Data Confidentiality and Reputation Schemes in Distributed Information Systems

DISSERTATION

zur Erlangung des akademischen Grades
doctor rerum politicarum
(Dr. rer. pol.)
im Fach Wirtschaftsinformatik

eingereicht an der
Wirtschaftswissenschaftlichen Fakultät
Humboldt-Universität zu Berlin

von
Herr Dipl.-Inform. Matthias Fischmann
geboren am 4. Januar 1973 in Tübingen

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Dr. h.c. Christoph Markschies

Dekan der Wirtschaftswissenschaftlichen Fakultät:
Prof. Oliver Günther, PhD

Gutachter:

1. Prof. Oliver Günther, PhD
2. Prof. Dr. Bettina Berendt

eingereicht am:	4. Februar 2008
Tag der mündlichen Prüfung:	23. Mai 2008

Abstract

In this thesis we discuss two demanding problems from the field of computer and communication security that involve trust.

The first is known as the database service provider problem: A database owner wants a database service provider (DSP) to host her database. She only trusts this DSP to a limited extent, so she does not want to rely solely on contractual solutions. It is therefore necessary to enforce confidentiality of her data by technical means. The second problem concerns a (potentially very large) number of network nodes in a peer-to-peer (P2P) environment.

Both problems are notoriously hard because, other than in traditional computer security problems, the adversary has a lot of control over the situation. The untrusted DSP needs to be able to process the data without learning anything about it, which seems to be a contradiction. In P2P applications it is desirable that nodes can join anonymously, but anonymity makes it easy to spread false reputation information. A node that enters a P2P application network for the first time needs to trust the claimed observations of other nodes, independent of the rate of malicious behaviour.

Our findings are not perfect solutions, but nevertheless instructive in several ways: We propose relaxed, but still practically useful, notions of security for the DSP problem; we identify theoretical limitations of the DSP solution space; and we gradually reduce the impact of adversarial behaviour in P2P reputation systems using heuristic methods. As a side effect of our work, we present a special-purpose framework for simulation of P2P reputation systems that can be used to compare and fine-tune previous and upcoming work.

Keywords:
confidentiality, databases, service provider, reputation

Zusammenfassung

Diese Arbeit betrachtet zwei anspruchsvolle Probleme aus dem Bereich Computer- und Kommunikationssicherheit und Vertrauen.

Beim Datenbank-Serviceprovider-Problem moechte ein Anwender seine Datenbank an einen Datenbank-Serviceprovider (DSP) uebergeben, damit dieser sie betreiben und ihm zur Verfuegung stellen kann. Er vertraut diesem DSP, und damit auch vertraglichen Abmachungen, nur bedingt und muss die Vertraulichkeit seiner Daten durch technische Massnahmen sicherstellen. Das zweite Problem ist das Verbreiten verlaesslicher Reputationsinformation ueber eine (moeglicherweise sehr grosse) Anzahl von Netzwerk-Knoten in einer Peer-to-Peer-Umgebung (P2P).

Beide Probleme straeuben sich hartnaeckig gegen einfache Loesungen. Im Gegensatz zu traditionellen Sicherheitsproblemen in der Informatik hat der Gegner in beiden ein hohes Mass an Kontrolle ueber die Situation. Der nicht ausreichend vertrauenswuerdige DSP muss in der Lage sein, die Daten seines Kunden zu verarbeiten, ohne etwas ueber sie zu lernen, was intuitiv wie ein Widerspruch erscheint. In P2P-Anwendungen ist es wuenschenswert, dass Knoten anonym beitreten und jederzeit wieder austreten koennen, aber diese Anonymitaet erleichtert es, falsche Reputationsinformation zu verbreiten. Ein Knoten, der erstmalig in ein P2P-Netzwerk eintritt, muss den behaupteten Beobachtungen anderer Knoten vertrauen.

Die Resultate dieser Arbeit sind keine Idealloesungen, und dennoch aufschlussreich in mehrerlei Hinsicht: Es werden gelockerte, aber immer noch nuetzliche Sicherheitsbegriffe fuer das DSP-Problem vorgeschlagen; es werden theoretische Grenzen des DSP-Loesungsraums gezogen; und die Auswirkung feindseligen Verhaltens in P2P-Reputationssystemen wird durch heuristische Methoden reduziert. Ein Nebeneffekt unserer Arbeit ist ein speziell fuer Reputationssysteme in P2P-Netzwerken geeignetes Simulations-Tool, das zum Vergleich und zum Fine-Tuning bestehender und zukuenftiger Forschungsarbeiten genutzt werden kann.

Schlagwörter:
Vertraulichkeit, Datenbanken, Service-Provider, Reputation

Contents

Chapter 1

Two Approaches to Information Security

Computing applications are increasingly moving from individual users' equipment into the network. In the early 1990s, a PC was hosting a range of applications that still to a large extent were completely unaware of the network, with e-mail and WWW being the two most prominent exceptions. Since then, more and more of the applications have migrated to server farms and have left behind only a user interface on the client machine. As a rule, newly emerging applications such as Web 2.0 are highly distributed by their very nature.

This imposes challenges of a new order of magnitude towards data integrity and confidentiality, among other security goals. If the personal information of the users is not stored on a local PC any more, it is insufficient to protect the data against unauthorized, external adversaries, because even legitimate participants such as a service provider or a peer in a community portal, that need some level of access to the sensitive data in order for the application to function, cannot always be trusted without reservation.

In other words, the user of a distributed system has to surrender control over her data to parties of limited trustworthiness at least to some extent, and naturally wants to maintain as much control as possible while sacrificing as little functionality as possible. The strictness of requirements varies between applications. Data integrity goals in file sharing networks may be satisfactorily met by making a posteriori decisions after download whether a given file is a valid response to a search query, but if confidentiality of sensitive business data is at stake, potential customers are less ready to make bold compromises.

Cryptography is a branch of applied mathematics that helps enforce security requirements in distributed computing systems in which not all partic-

ipants can be trusted to obey the rules of the protocol. Confidentiality and integrity are the most prominent security goals established by cryptographic means, and we will restrict ourselves mostly to those two:

1. Confidentiality is the need for information secrecy. For example, it is often required by two communicating parties that trust each other without reservation, but who fear of eavesdroppers on their communication channel. It is achieved by using a keyed transformation that turns the communicated data into apparent gibberish for any eavesdropper who does not know the key.

2. Integrity is the need for unmodified, authentic data flows. In the same scenario, but with an eavesdropper that owns the channel and can remove or change bits on the way, integrity is achieved by adding checksums that are computed by another keyed transformation with another secret key. By any changes to the communication, a potential adversary would break the verification process, since she does not know how to compute the changed checksums.

Both these requirements have robust and well-understood solutions that we use every day, perhaps even without being aware of it, by logging on to an SSL-protected web server. We are interested in more advanced security problems in the sense that the adversaries are *too strong*, i.e. it is impossible in principle to build secure systems. We accept this challenge and answer it by finding new notions of security, new methods of achieving it, and where nothing else works, solid theoretical proofs that some tasks are simply impossible. This thesis is comprised of two parts that consider two extreme scenarios.

In Part I, we consider a database service provider (DSP) who wants to convince client organizations to outsource storage and processing of their data to her server farm. The client, however, is uncertain about the confidentiality that the service provider can guarantee, and is only likely to sign a deal if security deteriorates not even marginally compared to local storage.

In Chapter 2, we lay the foundation for attacking this problem by giving an account of the according cryptographic machinery. This chapter is intended to be used by database researchers, so we will put particular emphasis on motivating the assumptions and definitions more thoroughly that in the basic texts on cryptgraphy that we know of. In Chapter 3, we will give a survey of the solutions that have been proposed to the service outsourcing problem (DSP or computations in other applications) and related fields, most importantly code obfuscation and private information retrieval. A code obfuscator is an operation that takes any program in a given language and

transforms it into an equivalent program that does not reveal anything of interest to an interpreter besides what can be learned from its input/output-behavior. A private information retrieval scheme allows for a client to access a database hosted by a DSP, only the confidentiality concerns the query text, not the queried data. Both obfuscators and private information retrieval will be helpful in our examination of DSP schemes.

In Chapter 4, we will then develop novel security definitions. The most important one is an adaptation on a common security definition from cryptography called indistinguishability. A variant of this takes into account that sometimes a client can be expected to switch into defense mode when the server turns against her. This is not realistic if the threat is a hacker breaking into the DSP host, but it may be what a company needs that wants to insure its clients against change of country of operation or change of ownership. We will use these definitions of security to derive efficient and secure cryptographic schemes for DSP based on existing full-text search algorithms, and prove their security. A treatment of the orthogonal question of data integrity in the database outsourcing scenario at the end of Chapter 4 concludes the constructive segment of this part of the thesis.

But there are also limits to what can be done. In Chapter 4, we will expose two popular results from related work, namely ATE (HILM02) and $\mathrm{Fts}^{[\cdot]}$ (SWP00), to extensive scrutiny, uncovering several flaws and inconsistencies. In Chapter 5, we will construct three independent proofs of impossibility for two of our stronger security definitions, covering three different subsets of relational algebra. Since private information retrieval can be reduced to Hom^{σ}, this yields a previously unestablished lower bound for the related problem of private information retrieval as well. Also, our reduction of homomorphic encryption to code obfuscation, plus a trivial reduction of code obfuscation on homomorphic encryption, will yield equivalence of the two problems.

In Part II, we consider fully decentralized peer-to-peer (P2P) networks and investigate into means of establishing trust between peers if nobody knows anybody a priori. Trust is useful for ensuring quality of service and for establishing incentives for cooperation by punishing defection. However, since every node in the network can ultimately only trust itself, trust grows too slow in large-scale networks to be of any use, and rumor schemes need to be deployed that propagate noise and sometimes intentionally false information about the behavior of other nodes. In order to decrease the noise level of these rumors, a number of techniques has been developed. These techniques are called reputation schemes. Ideally they consist of robust, efficient, distributed data structures and algorithms and advanced cryptographic primitives to reduce noise.

In Chapter 7, we will formally introduce the concept of trust and rep-

utation and motivate the relevance of this topic not only to P2P networks but to a much wider class of social systems. We will give a classification of P2P applications and one of adversarial node types, have explaine the complementary nature of monetary incentives and reputation, and introduce the relevant game theoretic concepts to assess the impact and relevance of our work.

In Chapter 8, we will establish a list of building blocks for constructing secure P2P networks, recycling and extending the cryptography from Chapter 2.3. Since most work on reputation schemes is based on existing P2P applications, which are in turn usually based on ad hoc network designs with inadequate cryptographic protection, this list should be of high value for future research.

We will conclude Chapter 8 with the description of three example reputation schemes, which will nicely lead over to Chapter 9, in which we will develop a new and generic P2P application model and a corresponding reputation scheme, and will use a generic P2P reputation simulator we have developed to implement and run this model and understand its characteristics.

We will design our model with the purpose of grasping the core idea of a wide variety of different papers on the subject in a concise way. It has a number of unusual aspects that make it not both challenging and potentially instructive to do this. It is very generic and it is based on advanced cryptographic distributed data structures, which together makes it more powerful than most existing application networks. It is orthogonal towards monetary incentive systems, and assumes the nodes are indifferent, too. It abandons the hope of strong results like game theoretic equilibria that force nodes to cooperate, and assumes certain levels of adversarial behavior independent of the rules of the game. Finally, it is deliberately too complex for examining it with functional analysis, and all our results stem from simulation. All these features will be explained and justified in the following.

The two problems are opposites in several ways. This explains why our results and even the methods of achieving them are so different, and it illustrates the vastness of the field of computer and communication security.

- The topology of DSP consists of a server node and a single client node (although there is more than one client, they do not interact, so the model only considers a two-node network) with fixed roles. P2P networks have many nodes with no fixed roles, interests, or communication channels.

- While in our treatment of the DSP problem, we face a situation with two strong, non-anonymous identities and a long-term relationship that

can be supported by off-line interactions. Therefore, we take data integrity for granted, while data confidentiality is a concern. In P2P networks, identities are anonymous (and this is by requirement), and connections are as numerous as they are volatile. Therefore, confidentiality in the strong sense is so hard to achieve that we only consider applications with low or no confidentiality requirements. However, (at least gradual) integrity of data is in important issue.

- And finally, while in Part I we strive at establishing perfect security for DSP, Part II is about incremental security improvement of P2P applications that will always contain a certain fraction of lazy or hostile nodes.

Why not strive for incremental security improvements for outsourced databases as well? In a sense we did that, by giving a more relaxed definition of what confidentiality means. However, the nature of the application gives us far less space to navigate than the P2P scenario. A DSP client has high standards according to security — those that are set by the alternative of running the database service on her own premises, which gives her virtually perfect access control, both physically and online. If security or performance is noticeably reduced by an outsourcing move, it will not happen, so we need very reliable technology to convince clients that security breaches in the DSP scenario are less likely than without outsourcing. In the following we will prove that this can only be done in very reduced notions of security, and thus debunk prominent research results in this direction as well as disencourage future work that will only lead to disappointments.

In P2P applications, the situation is quite different. First of all, integrity tends to be not quite as black-and-white as confidentiality: If my adversary learns about my secrets, all is lost, and the consequences can be arbitrarily damaging to me, including bankruptcy. If I misjudge the honesty of a node in a P2P network with whom I choose to cooperate, it may be quite possible to limit the damage that I risk to take. After all, I always know that all trust and reputation values in the system are only estimates, and I will not share my most sensitive secrets with a moderately trustworthy node. Instead, I might just contribute a few idle CPU cycles or memory blocks in the hope of receiving idle resources in return later when I need them.

Incremental decrease in security in P2P networks yields incremental decrease in quality of service, or incremental increase of the fraction of adversarial nodes, but a P2P application allowing for sporadic failures may still be far better than no P2P network at all. On the other hand, sporadic failures in the security of a DSP system are unacceptable, since even the first successful attack could lead to the revelation of valuable business secrets. The

two problems, although both are about highly distributed applications, after close inspection turn out to be close to opposites not only in their security requirements, but also in the techniques appropriate for assessing whether a given system can meet those requirements or not.

The days in which IT-security problems had sound, robust solutions that could keep the system secure against any conceivable adversary is over. With the growth of cyberspace, new adversaries arise that cannot be easily defended against using traditional encryption or message integrity preservation schemes, since the need for cooperation and data exchange conflicts with their assumptions and preliminaries. In this thesis, take the new situation into account, aiming for a new understanding of the nature of security.

By deciding early on which notions of security are most suitable in a given context, one can benefit in two ways:

1. Security of the system can be established in a meaningful way that does not lead to frustration and shrink on the demand side, and

2. ill-guided research that yields security-that-is-not-security can be avoided by proving over-ambitious security requirements unsatisfiable, and thus false allocation of precious intellectual resources can be avoided.

In this thesis, our aim is to benefit two extreme applications in both ways, and by choosing extreme techniques to establish security in these applications evince directions in which to advance research in the field of security in distributed computing.

Part I

Homomorphic Encryption of Databases

Chapter 2

Introduction

This part of the thesis, we consider applications in which a database system is operated by a service provider (*database service provider, short: DSP*). We address the problem of maintaining confidentiality of the information processed by the application, despite the fact that the client has only limited control over the trustworthiness of the database service provider.

2.1 Problem Statement

Traditionally, database security is concerned with two threats: Outsiders attacking the database system at its boundaries and malicious or confused users causing damage from within.

Attacks at the system boundaries have been a growing concern since those have become increasingly large and distributed, and more and more communication takes place on public networks like the Internet. It is especially problematic that those networks usually provide little protection as they have been designed with an adversary in mind that is external to the network (Rob66). The most immanent risks are those of eavesdropping or revelation of confidential information to others, data corruption such as injection of false (or deletion of legitimate) information, and denial of service such as disruption of a connection to a database server.

The former two can be solved effectively and with low performance overhead by public-key cryptographic algorithms for encryption and authentication, available through a large number of implementations of standards like SSL and TLS (DA99). All messages are transformed into ciphertext before transmission, and an adversary who is not in the possession of the legitimate credentials cannot recover the plaintext communication from intercepted ciphertext. Further, if she injects, removes, or alters ciphertext between client

and server, decryption will fail and the system will be notified of the attack. (See Section 2.3 for a more detailed treatment of encryption, and Section 2.3.5 for more on authentication.) Denial of service attacks are harder to address in general, and usually require risk management and redundancy and intrusion detection counter measures (Roe99; Koz03).

Threats caused by malicious (or confused, or incompetent) users are usually dealt with by access control mechanisms, which can become surprisingly complex, powerful, and hard to understand and deploy (Ben05). Access control presumes a private, authenticated channel on which database client and server both are confident that nobody eavesdrops and that they are talking to each other (and not to an imposter). Each time the client attempts to perform a transaction, the database server consults a set of access control rules. Only if the transaction is consistent with the rule set, the server processes it. Otherwise, an error is triggered.

The third potential threat besides user and outsider, the database server itself, has been paid less attention so far. There are good reasons to consider it last. A much higher level of protection is often affordable than for clients or network infrastructure. But even a server has to be maintained by staff that can only be trusted to a certain extent. If users can turn against the system, then so can administrators.

Outsourcing has become increasingly common in the last years, allowing users to benefit from better utilization of resources like hardware, experience, man power. This has mixed effects on the risks of privacy breach.

A service provider with specialized security engineers may be better suited to protect the server site against both hackers and physical threats, and achieve both at a lower price per customer than any small company could on its own. Non-disclosure agreements can be negotiated so that the customer can hope to get compensated for any incident that causes loss of confidential information to a third party.

Unfortunately in practice, laws and contracts have limited power (BG02). New risks emerge, like potential change of ownership of the service provider. In the merger of PeopleSoft with Oracle in January 2005 in the US, PeopleSoft's customer data became property of Oracle, leading to highly complex legal arguments about the validity of privacy agreements between PeopleSoft and its clients. Further, companies that considered Oracle as their direct competitor at the time of the merger may have failed to withdraw their data in time from the PeopleSoft databases. Back in 2000, Internet retailer Amazon had a very clear position on this subject:

> *In the unlikely event that Amazon.com Inc., or substantially all of its assets are acquired, customer information will of course*

be one of the transferred assets.
`http://www.wired.com/news/politics/0,1283,38572,00.html`

Amazon is not in the database service provider business, and the privacy policy has been changed since. But the quote still illustrates the potential risks that a client exposes herself to if she hands over confidential data to a third party. No need to say these risks are multiplied in international mergers, where many laws already in place on the national level in the US and many other countries still need to be established.

As information technology is gradually penetrating more and more aspects of our lives, confidentiality of data has seized to be an exclusively business-to-business topic. It is becoming a concern in people's everyday lives. The market for web and mobile services is growing, and a large fraction of all e-mail today is processed on a web-server instead of an application running on the user's computer.

Empirical research has given strong evidence that customers of such services are aware that there is a problem, and that this awareness is inhibiting market growth. Further, even if the customer base is still large enough to do business with, any DSP constantly risks being exposed in an identity theft scandal and losing market share to more lucky competitors. If all application data was securely encrypted without obstructing the DSP, crafting a privacy policy would be trivial, and the risks involved in hacking incidents would be minimized.

Finally, even if the effectiveness of legal and regulatory precautions is not to be questioned in any particular scenario, those precautions still come at a substantial cost. All parties involved can profit from technical solutions that make it impossible for a service provider to cause any privacy breach in the first place. The service provider will save legal infrastructure, may be able to reduce security precautions, and still face lower risks of unhappy clients. Clients can have higher confidence in the service provider and may save some real money if running the service will become cheaper due to the reduced resource requirements.

So even if the server is trustworthy it may prove beneficial for everybody to assume it is not. In that case, re-establishing security on a technical level is a big challenge. This is the challenge we will take on in the next three chapters. In Chapter 3, we will review and catalogue the solutions available in research literature. In Chapter 4, we will present a new class of solutions for a reduced subset of relational algebra that is provably secure, and establish the theoretical tools to prove its security. Finally, in Chapter 5, we will establish upper limits for the level of security against an adversarial service provider that can be achieved by mere technical solutions, and lower

limits for the overhead those solutions will always cost. In our eyes, these limits are particularly essential to direct and evaluate future research, and shed new light on available results.

In the remainder of this chapter, we will introduce the reader to the basic concepts of database theory and cryptography.

2.2 Database Foundations

Databases are traditionally organized *tables* (or *relations*) that have *rows* (or *records*, or *tuples*) and *columns* (or *features*, or *attributes*). Each record in a table lists one set of attribute values that satisfies the relation.

2.2.1 Relational Algebra

In 1970, Edgar F. Codd proposed using relational algebras to reason about database systems (Cod70). As a formalization of relations and the laws by which they are governed, it is still the theoretical foundation of the field of database systems in use today. For an extensive treatment of the subject, see cf. (RE03). A very concise summary of relational algebra, including implementation and complexity issues and a few simple query rewriting concepts can be found online (Nel99).

A relation R is a set of tuples $r_0, \ldots, r_n$. The *domain* $\mathcal{R}$ of R from which the tuples are chosen is the cross product of the attributes of the relation:

$$\begin{aligned} \mathcal{R} &= A_0 \times \cdots \times A_k \\ R &\in \mathcal{R}^P \\ r_i &= (a_0, \ldots, a_k) \end{aligned}$$

(With $\mathcal{R}^P$ we denote the power set of set $\mathcal{R}$.)

An algebra consists of operations and operands. The operands are the relations, so now we need some operations. Since relations are sets, we can start by using *union*, *intersection* and *difference* from set theory:

$$\begin{aligned} R \cup R' &= \{\ r \mid r \in R \vee r \in R'\ \} \\ R \cap R' &= \{\ r \mid r \in R \wedge r \in R'\ \} \\ R - R' &= \{\ r \mid r \in R \wedge \neg(r \in R')\ \} \end{aligned}$$

Databases are used to answer questions of the form "which of my tuples have property P?" This is done by the *select* operation σ:

$$\sigma_P(R) = \{\ r \mid r \in R \wedge P(r)\ \}$$

P is a *predicate*, or a boolean function on the domain $\mathcal{R}$ of R. It can be expressed using the logical operators $\wedge, \vee, \neg$, arithmetic operators $<, >, =, ...$ and attribute names A_i and constants $a \in A_i$ as operators. For example, for a suitable relation R over tuples representing employees, the query $\sigma_{\text{salary} \geq 1500.00}(R)$ returns all employees whose salary is higher than 1500.00.

In many contexts, we only want to look at a subset of all attributes stored in a relation. The *project* operation removes all but a given set of desired attributes:

$$\pi_{\mathcal{I}}(R) = \{\ (a_i, \dots) \mid (a_0, \dots, a_k) \in R \wedge A_i \in \mathcal{I}\ \}$$

$\mathcal{I} = \{A_{i_0}, \dots\}$ is an index set that contains the names of the attributes to be kept in the result of the projection. Keep in mind that relations are sets, so resulting duplicate tuples in the relation are implicitly removed.

The last operation is *join*. It is the counterpart of projection in the sense that it increases tuple size. In its most basic form, it is similar to the cross product, only that it does not produce pairs of tuples but concatenations:

$$R \bowtie R' = \{\ (a_0, \dots, a_k, a'_0, \dots, a'_{k'}) \mid (a_0, \dots, a_k) \in R \wedge (a'_0, \dots, a'_{k'}) \in R'\ \}$$

In terms of expressive power, this is all we will need. However, from an implementation perspective, $\bowtie$ is intolerably expensive: When applied to two large relations, it produces a result of the size the product of the sizes of the two input relations. Worse, one is rarely interested in the cross product of two tables, but in a maximum subset of the cross product on which the tuple pairs match certain attributes. For example the office number of an employee may be stored in one table, and her salary in another, but when we join the two tables we want the result to contain only concatenations of tuple pairs that represent aspects of the same employee.

In order to allow for a more concise representation of equality constraints, relational algebra usually contains a variant of $\bowtie$ that is called *equi-join*:

$$\begin{aligned} R \bowtie_P R' = \{\ & (a_0, \dots, a_k, a'_0, \dots, a'_{k'}) \\ & \mid (a_0, \dots, a_k) \in R \\ & \quad \wedge (a'_0, \dots, a'_{k'}) \in R' \\ & \quad \wedge P(a_0, \dots, a_k, a'_0, \dots, a'_{k'})\ \} \end{aligned}$$

A less flexible but better known and often more convenient variant of equi-join uses a set of attribute names that need to occur in both tuples and

have the same values:

$$\begin{aligned} R \bowtie_{\mathcal{I}} R' = \{\ & (a_0, \ldots, a_k, a'_0, \ldots, a'_{k'}) \\ & \mid (a_0, \ldots, a_k) \in R \\ & \quad \wedge (a'_0, \ldots, a'_{k'}) \in R' \\ & \quad \wedge \forall A_i \in \mathcal{I}, a_i, a'_i \in A_i . a_i = a'_i \ \} \end{aligned}$$

This establishes a sound and powerful core of relational algebra. Other common operations that are often added are grouping γ and sorting τ. τ_I transforms a relation into a sequence of tuples such that attributes in the attribute sequence $I = (A_{j_0}, \ldots)$ appear in some canonical order:

$$\tau_I(\{r_0, \ldots, r_N\}) = (r_{i_0}, r_{i_1}, \ldots, r_{i_N})$$

where

$$\forall A \in I, j < k \ : \ r_{i_j}.A < r_{i_k}.A$$

Grouping $\gamma_{I,F}$ collapses all tuples with all attributes in the set $I = \{A_{j_0}, \ldots\}$ equal into one tuple. This often requires an aggregation function $F : \mathcal{R}^p \to \mathcal{R}$ that aggregates all values of attributes that are not in I:

$$\gamma_{I,F}(\{r_0, \ldots, r_N\}) = \{F\{r_i \mid \pi_I(r_i) \text{ is the same throughout the set}\}\}$$

For instance, consider a table listing bank account transactions in the form (a_0, a_1), where a_0 is the day of the transaction and a_1 is the amount of money added to the account (negative for withdrawals). Then the total of transactions per day can be computed with $\gamma_{I,F}$, where $I = \{a_0\}$ and $F(R) = \sum_{r_i \in R} \pi_{\{a_0\}}(r_i)$.

(If F is not required since no non-grouped attribute is left over, simply choose $F(x) = x$ to be the identity function.)

2.2.2 SQL

In the remainder of this thesis, we will use relational algebra as a model for databases. However, many of the readers may have experienced databases from a different perspective, namely the SQL query language (CB74). SQL is designed to be more convenient to use than relational algebra, but the two translate into each other to a large extent. In this paragraph, we shed some light on this translation. In particular, we explain why, when reasoning about security, we can simply ignore the existence of those parts of SQL that are used to modify existing tables.

SQL allows for handling the database schema (creation, modification and deletion of tables and their attribute sets) as well as the data (feeding the tables and asking questions, or *queries*, about their contents). The following SQL command creates a table for storing employee information:[1]

```
CREATE TABLE employee {
  name       VARCHAR(50),
  born       DATE,
  position   VARCHAR(80),
  department NUMERIC,
  salary     NUMERIC
};
```

Each attribute in the table has a *data type* that restricts the contents to a certain form: `VARCHAR` fields may only contain character strings, `NUMERIC` fields numbers, etc. All SQL commands are terminated by a semicolon.

A table could be removed from the database like this:

```
DROP TABLE employee;
```

When a new employee joins the company, a new record needs is added to the table:

```
INSERT INTO employee (name, born, position, department, salary)
   VALUES ('Jean-Paul Sartre', '1905-06-21', 'Novelist',
           13, 4910.00);
```

And if Mr. Sartre leaves the company:

```
DELETE FROM employee WHERE name = 'Jean-Paul Sartre';
```

Finally, once we have established a database schema and fed all the records to it representing the data that we want to manage, we can start retrieving information. The following SQL command outputs a table containing all employees born before 1950:

```
SELECT * FROM employees WHERE born < '1950-01-01';
```

[1] The examples that are given here establish an SQL dialects many of which are in wide-spread use today. In this text we value clarity and simplicity over flexibility and expressive power, so the code may need small adaptations before it can be used with an SQL database production system.

In terms of relational algebra:

$$\sigma_{\text{born<'1950-01-01'}}(\text{employees})$$

Projection is already implicit in the `*`, and explicit in the following:

```
SELECT (name, salary) FROM employees;
```

is the translation of

$$\pi_{\text{name, salary}}(\text{employees})$$

For the example for join, consider a second table in which each department is listed with the name of its head and a description of the department's function. Then, the following gives a list of employees with additional department information:

```
SELECT * FROM employee, department
WHERE employee.department = department.name;
```

Or, in terms of relational algebra:

$$\text{employee} \bowtie_{\text{employee.department = department.name}} \text{department}$$

Up to here, relational algebra is a purely functional calculus in the sense that none of its objects have a state that changes over time. An expression creates new tables from existing ones, but no table is ever modified. How do we express `INSERT` or `DELETE` in this context? It is simple enough to use set union and intersection to model the creation of an updated *copy*, but this is such a strong abstraction that it may seem too inaccurate to represent reality: We want to manipulate an existing relation, not compute an entirely new one from it. The latter would be intolerably inefficient.

The mathematical tools for expressing manipulation of the objects and changing database state in relational algebra can be found in text books on programming language theory (Bar84; AC96; AS85). Expressions are evaluated in a well-defined order and in an environment of mutable variables that can be referenced in the expressions. The language is extended by operations that replace the contents of these variables with new data (e.g. the union of the contents of a table and the set of new records to be inserted). The resulting algebra is still simple and abstract, while allowing for efficient implementations of update operations.

Fortunately, knowing that it can be done is enough for our purposes. For most problems in database theory, the function view captures all the relevant aspects of database systems. In particular, for reasoning about data

security, we only need the simpler functional view on relations: If, say, the intersection operation that corresponds to a DELETE statement yields a fresh table and leaves the old one intact, the information on the database contents that can be gained from observing the system is the same as if the old table was modified and the observer kept a history of previous system states. The difference between the two models, namely the deletion of the old table and storage of the new table, does not introduce any new vulnerabilities to attack. This claim may appear to require further justification right now, but it will become self-evident once we start discussing security definitions for homomorphic database encryption in the next chapter.

2.2.3 Object-Oriented Databases

Today, object-oriented software engineering has become the norm. The data is no longer separated from the algorithm, but the algorithm is embedded into the data objects. With this paradigm shift, considerable improvements in code modularity, abstraction and therefore robustness have hit the mainstream. Due to its success, it has been applied to other fields of computer engineering, such as databases.

In object oriented databases, the data is not organized in records, but in objects. This allows for a more direct mapping of data structures of object-oriented programming languages to the data structures of the database. Also, most object-oriented databases come with a richer query language that allows for isolating further parts of the functionality of a software system into the database component.

Despite these two appealing advantages, even in 2006, 37 years after development of the object-oriented programming language Smalltalk began, databases are still relational to a large extent. This may be due to the many years of experience with relational databases, the generally better performance characteristics due to a simpler model, or last not least the fact that the weaker expressive power of relational databases is enough for most applications.

In this thesis, we will not consider object-oriented databases. Nevertheless, our results map directly to them, since objects can be stored in relations, and queries on objects can be transformed into (more complicated looking) queries on those translations. In fact, many object-oriented databases are implemented using such a mapping and a relational core engine, providing the gain in expressive power to the user without sacrificing relational database know-how. So by using a simpler (but not less general) model, we obtain results that apply to both relational and OO database systems.

2.3 Cryptographic Foundations

The second Volume of Goldreich's *Foundations of Cryptography* starts with an explanation of the term so concise that we feel it is best to simply repeat it here:

> *Cryptography is concerned with the construction of schemes that withstand any abuse. Such schemes are constructed so as to maintain a desired functionality, even under malicious attempts aimed at making them deviate from their prescribed functionality.*
> Oded Goldreich (Gol04), p. xiii

Two of the most prominent sub-disciplines of cryptography are encryption and authentication. *Encryption schemes* are used for keeping (part of) the data in the system hidden from an adversary. *Message authentication* and *signature* schemes ensure integrity of data, i.e. the fact that no adversary is able to change or delete existing data or inject forged data.

Part I of this thesis will only deal with those, and in fact only with encryption to a large extent. In Part II, we will introduce more advanced cryptographic schemes and applications. In this section, we have a closer look at the basics of cryptography and introduce the essential preliminaries. It is not necessary to take it all in at once; a more suitable approach may be to read it diagonally and return to it once the individual concepts are applied later on.

2.3.1 Probability Theory

To express the security of a cryptographic scheme, the winning probability of the adversary in an assumed game is computed. If that probability is small enough, the scheme is secure. So in order to do cryptography, we need a few basics from probability theory. The foundations of modern probability theory have been established by Kolmogorov in the 1930s (Kol33). For a more thorough treatment see (Var01) or (Vai06).[2]

An *experiment* is a device that, once set in motion, yields one of a number of possible *outcomes*, or *samples*. The set of all possible outcomes is called the *sample set*. An *event* is a subset of the sample set. For instance, when rolling two dice, the sample set is denoted by the cross product $\{1,2,3,4,5,6\} \times \{1,2,3,4,5,6\}$, and the event "doublets" is denoted by $\{(1,1),(2,2),(3,3),(4,4),(5,5),(6,6)\}$. An event $S_i \subseteq S$ is said to *occur*

[2] The more philosophical question of how the outcomes of using the devices introduced in this section can be interpreted is attacked in (Háj03).

if the outcome of the experiment lies in S_i. $\Pr(S_i)$ denotes the *probability distribution* (or *probability*, for short) of the event S_i to occur.

$\Pr : S \mapsto \mathbb{R}$ must satisfy Kolmogorov's probability axioms, which directly imply that Pr is a measure on the sample set:

(*i*) *(impossibility)* Every experiment run produces *some* outcome:

$$\Pr(\{\}) = 0$$

(*ii*) *(certainty)* Every possible outcome is contained in the sample set:

$$\Pr(S) = 1$$

(*iii*) *(additivity)* For every pair of disjoint events $S_1, S_2 \subseteq S$ with $S_1 \cap S_2 = \{\}$, the probability of either of two disjoint events occurring is the sum of the individual probabilities of the events:

$$\Pr(S_1 \cup S_2) = \Pr(S_1) + \Pr(S_2)$$

A *random variable* $X : S \mapsto \mathbb{R}$ is a function that maps samples onto real numbers. Random variables are convenient whenever numerical outcomes are needed if the sample space is non-numerical (such as $\{\text{heads}, \text{tails}\}$ in the case of a coin toss experiment). If the sample space is numerical (such as in dice experiments), a suitable random variable is the identity function $X(s) = s$.

This allows for a more convenient notation. Events are usually expressed as conditions on samples, for instance $\{s \in S | X(s) \leq 0\}$, so as shorthand for the probability $\Pr(\{s \in S | X(s) \leq 0\})$ we often write $\Pr(X \leq 0)$.

The probability distribution of the random variable X is often expressed in terms of its cumulative distribution function F, or its probability density function f. F is defined as follows:

$$F(x) = \Pr\left[X \leq x\right], \quad x \in \mathbb{R}$$

The density gives for each closed interval $[a, b] \in \mathbb{R}$ the probability that the outcome of an experiment is in $[a, b]$:

$$\Pr[a \leq X \leq b] = \int_a^b f(x)\, dx$$

Many distributions have been given enough attention to have their own names (McL99), such as the *normal distribution* (or *Gaussian distribution*) materializing in a vast number of natural phenomena, or the *exponential*

distribution useful for modelling growth processes. In cryptography, we are mostly interested in the *uniform distribution* that describes "pure randomness" in the sense that no information on the outcome is available prior to the experiment. In the discrete case,[3] a random variable is *uniformly distributed* iff the density function on individual events is constant:

$$\forall s \in S \,.\, f(s) = \frac{1}{|S|}$$

2.3.2 Algorithms, Complexity Theory, and Oracles

When assessing the security of cryptographic methods and tools we will have to prove that adversarial success is theoretically impossible ("there is no algorithm for launching a successful attack"). Sometimes, it makes life easier if we contend ourselves with proving that it is merely infeasible in practice ("there is an attack, but the adversary has not enough computing power to run it"). Either way, we will need a few basic concepts from complexity theory. Two of the many good standard text-books on algorithms and complexity are (HU00) and (HRL97).

The description of a cryptographic scheme and its security involves several parties, including at least one honest participant and one adversary. The adversary may have a legitimate role in the scheme (e.g., it may be the service provider running a database for a client, and behind its client's back attempt to steal the data in the database despite it being encrypted), or she may be an outsider (e.g., somebody eavesdropping on an e-mail correspondence).

Turing Machines

In order to obtain a mathematical description of the cryptographic scheme, we take all parties involved to be *Turing machines* (see cf. (HRL97; GJ79)). A Turing machine (TM) consists of a one-dimensional infinite tape of bit cells, a read-write head, and a directed graph of nodes representing abstract states with a cursor on a dedicated start node. The machine proceeds in steps. In each step, the read-write head reads a symbol, moves either to the right or to the left, and writes a symbol. Then, a state transition is performed moving a cursor on the state graph from the current node to the next along any suitable directed edge. The machine halts if the cursor hits a dedicated stop node (there may be one or more stop nodes, or even none).

A subset of all bit sequences that can be written on the tape is called a *language*, and the sequences in a language are called *words*. The TM is

[3] Since cryptography is a discrete discipline, we can ignore the possibility of continuous sample spaces.

said to *accept* a language $\mathcal{L}$ if it reaches a stop node when run with any word $l \in \mathcal{L}$ on its tape. It is said to *generate* $\mathcal{L}$ if for each $l \in \mathcal{L}$, there is an input such that it halts with l written on the tape. Both generating a language and accepting it is sometimes called *solving* it. In a way, while TMs are generic mathematical representations of *programs*, languages are generic mathematical representations of *problems*.

Originally, the TM is considered to perform a specific computation on its input data, and the state graph is representing the nature of that computation. But it is possible to construct a single TM whose purpose is to run computations that can be expressed in an *arbitrary* TM. In fact, this other TM is represented as a bit stream that can be stored on the tape together with its input. A TM that interprets such representations of any other TM is called a *universal Turing machine*. This type of Turing machines occurs wherever the boundaries between code and data are blurred, which happens in cryptographic theory (see Section 3.3) as well as in everyday 20th century IT, e.g. in electronic documents that support macro languages.

A few models of algorithms with less expressive power than Turing machines have been proposed, such as boolean circuits and stack automata. We will introduce circuits when thinking about code obfuscation in Sections 3.3 and 5.2, since conceivably programs may be easier to obfuscate if they are expressed in a more primitive language. However, since we cannot control the devices the adversary will use to launch an attack on an information system, we usually only need the most general model of computation.

Are there any stronger models? Turing machines come in many flavors: The tape alphabet can be extended from bits to arbitrary finite sets of symbols, the number of cursors or the number of tapes, or even the number of dimensions of each tape can be increased, and other ways in which the read-write head may move can be added. Further, several other formalizations of algorithms have been proposed over a short period of time in the history of computer science, such as the λ-calculus, recursive functions, Post automata, and probably others. However, equivalence results have emerged soon establishing that whenever a problem can be solved by an algorithm expressible in one model, that algorithm can be expressed in every other model. In particular, all programming languages can be mapped onto any of those models in terms of expressive power. This suggests that all models give an accurate account of what constitutes an algorithm and what problems can be solved by automated procedures, and it provides us with the luxury of using the most convenient one exclusively. For our purposes, this will be Turing machines, and we use the term synonymously with the terms algorithm and program.

Complexity

A TM is said to be *time bounded* by $f(n)$ if on an input word of length n, the TM halts after at most $f(n)$ steps. A language $\mathcal{L}$ is said to be bounded by f if there is a TM that solves $\mathcal{L}$ and that is bounded by f. A TM being *space bounded* by $f(n)$ means it never uses tape cells beyond cell $f(n)$. Although considering these two resources (and possibly others) separately in general has proven useful, no TM with space bound f can have time bound g if g grows slower than f: each cell that is written to the tape uses up one step of run time. In this thesis, we will therefore only consider time bounds.

This approach of measuring the time in individual steps turns out to be too accurate. If we want to optimize a program, we are interested in the exact number of steps the program requires to terminate, but first, a look at f's *complexity* is likely to be more productive: If $f(n) = 2^n$ and we can show that a language is bounded by f, no matter how hard we try to tweak any TM that solves it, we will always run out of time on moderate problem sizes.

To reason about classes of boundary functions, we write $O(f)$ for the set of functions that grow at most as fast as f:

$$O(f) = \{g : \mathbb{N} \to \mathbb{N} \mid \exists c \in \mathbb{R}, c > 0 \,:\, \exists N \in \mathbb{N} \,:\, \forall n \geq N \,:\, g(n) \leq c \cdot f(n)\}$$

We also say that a TM or a language is in $O(f)$ if its bound is in $O(f)$. When thinking about the adversary ("how good can he possibly get at solving this problem, and is it bad enough?"), or about the hardness of a problem ("how much work are we going to have to put into solving this, and will it ever work?"), we sometimes want to establish lower bounds. These can be written down the the Ω-notation in analogy to the upper bounds described using O:

$$\Omega(f) = \{g : \mathbb{N} \to \mathbb{N} \mid \exists c \in \mathbb{R}, c > 0 \,:\, \exists N \in \mathbb{N} \,:\, \forall n \geq N \,:\, g(n) \geq c \cdot f(n)\}$$

Finally, there is a notation for describing a corridor of upper and lower bound:

$$\Theta(f) = \{g : \mathbb{N} \to \mathbb{N} \mid \exists c, c' \in \mathbb{R}, c, c' > 0 \,:\, \exists N \in \mathbb{N} \,:\, \forall n \geq N \,:\, c \cdot f(n) \leq g(n) \leq c' \cdot f(n)\}$$

For example a language is in $O(n^2)$ if there is a solving TM that terminates in quadratically many steps over the input. Both language and TM are said to have quadratic complexity. A language that is in $O(p)$ for any polynomial p is said to have *polynomial complexity*. The class $\mathcal{P}$ of all such languages is called a *complexity class*. It forms the core of the complexity theory that we need in cryptography, so we name it:

$$\mathcal{P} = \{\mathcal{L} \mid \exists \text{ polynomial } p \,:\, \mathcal{L} \in O(p)\}$$

An analogous class of problems is

$$\mathcal{NP} = \{\mathcal{L} \mid \exists \text{ polynomial } p : \mathcal{L} \in O(p) \text{ for non-deterministic TMs}\}$$

Problems that are in $\mathcal{NP}$ or harder are called *$\mathcal{NP}$-hard*. It is usually assumed that there is a gap between $\mathcal{P}$ and $\mathcal{NP}$, and in fact many complexity classes have been defined that are known to be harder than the former or easier than the latter. However, it is one of the grand open problems of computer science to find a proof that $\mathcal{P} = \mathcal{NP}$ is false.

In most cases, an algorithm with reasonable performance can be found for a language in $\mathcal{P}$, whereas any language not in $\mathcal{P}$ is intuitively and practically infeasible to compute for realistic input word lengths. This observation has found its way into terminology: Languages in $\mathcal{P}$ are also called *efficient*. TMs in $\mathcal{P}$ are also called efficient TMs, but more commonly *polynomial time* TMs.

In many situations in cryptography, one cannot rule out that the adversary is implausibly lucky, for instance by guessing a secret encryption key correctly and using this key to decrypt an intercepted message. We can then only state that this luck is implausible, or so unlikely that we can ignore it. This is done by assigning the event a *negligible* probability (we will come to that soon).

A function f is called *negligible* if it grows slower than the inverse of any polynomial p:

$$\forall \text{ polynomials } p : \exists N \in \mathbb{N} : \forall n \geq N : f(n) \leq 1/p(n)$$

or, using O-notation:

$$\forall \text{ polynomials } p : f \in O(1/p)$$

For brevity, we use neg in place of an arbitrary negligible function and write "$\ldots \text{neg}(\dot{)} \ldots$" instead of "there exists a negligible f such that $\ldots f(\dot{)} \ldots$".

The original motivation for developing theories of computer performance were to gauge existing algorithms, to establish limits of how efficiently problems can be solved in principle, and to approximate these limits as tightly as possible. However, when the discipline of cryptography merged with mathematics and computer science, it turned out that complexity theory can also be used to establish lower limits to adversarial algorithms that are impractical for the adversary to overcome.

Complexity is a vast field in theoretical computer science. But when concerned with applications of cryptography as we are in this thesis, loosely speaking it boils down to making sure two things.

1. **f**easibility. The honest players have algorithms in $\mathcal{P}$.

2. **s**ecurity. The languages the adversaries have to solve in order to break the system are $\mathcal{NP}$-hard. (Ideally, the adversary has no algorithm at all, no matter how complex. However, complexity theory has become accepted as a sound basis for all of public-key cryptography for more than 30 years of open research now.)

Variants of Turing Machines

Several changes can be made to standard TMs that make them more suitable for certain situations. $\langle M \rangle^k$ is a machine that times out after k steps, whether a halt state has been reached or not. It is a straightforward exercise to give a generic construction of a timeout variant for an arbitrary TM.

Probabilistic TMs are more interesting. They have an extra tape with an extra head that moves one bit to the right in every step. The tape contains a random sequence of bits that the TM can base its moves on. Often, the random tape is modelled as the ability of the TM to toss a coin once in each round, and use the upper side of the coin as input for the state transition function.

The source of randomness adds something new to the computing model. Probabilistic TMs cannot be modelled using non-probabilistic TMs, despite the fact that randomness is available on every computer. Since randomness makes it harder to predict an algorithm, it is harder to fool a probabilistic TM it into overlooking a flaw in a security system, which makes probabilistic adversaries harder to defend against. Therefore, adversaries are thought of being able to build arbitrary *probabilistic polynomial-time Turing machines (PPTMs)* by default.

Non-deterministic TMs (or NTMs) are TMs that may have more than one possible transition for one specific situation (bit on tape and current state in transition graph). When an NTM is run, all steps are taken simultaneously, and as soon as a halt state is reached somewhere, all simultaneous runs are terminated and the corresponding tape contents is output.

NTMs are used to describe a class of computational problems, namely the class of languages in $\mathcal{NP}$:

$$\mathcal{P} = \{\mathcal{L} \mid \mathcal{L} \text{ is decidable in polynomial time by a non-deterministic TM}\}$$

Although is appears that languages in $\mathcal{NP}$ are very hard to solve in general, there are a number of problems for which practical solutions exist (e.g. linear optimization). The question whether $\mathcal{P} \neq \mathcal{NP}$ is one of the grand open problems of computer science.

Oracles

If an online banking web server is under attack from the Internet, the adversary can send arbitrary messages to the server and observe what the server does. If she is lucky, she can even submit randomly created messages, have the server treat them as valid encrypted messages, make an attempt to decrypt them, and pass the outcome back to her.[4]

The web server in this attack is called an *oracle* for the adversary. If the adversary is a TM A and the web server is a TM M, we say that A has oracle access to M.

Definition 2.1 (Oracles)**.** *If a TM A is allowed to compute values of a function f for arbitrary input of its choice, we say that A has* oracle access *to f and write*

$$A^f$$

Analogously, oracle access to a TM M is written A^M. In each step, A may feed one bit of input to the oracle or read one bit of output from it.

In the example, f would be the decryption function that the web server runs in incoming messages. Note that f is not required to be in $\mathcal{P}$, or even computable. However, there is a natural restriction on what an algorithm A with access to an oracle can do. If A is in $\mathcal{P}$, it only gets to ask polynomially many questions, and the questions and the answers must be polynomial in size.

Dice

We will often need samples from a uniform distribution over an arbitrary set, so we conclude this section with the definition of the PPTM RND^X. Formally, for any set X and any element $x \in X$:

$$\Pr[\mathrm{RND}^X = x] = \frac{1}{|X|}$$

2.3.3 Encryption Schemes

Consider two persons who want to exchange messages (in the form of bit streams) over a public telephone line. Because they are wary of eavesdroppers and want to keep the communication private, they decide to use an *encryption scheme*.

[4] Although in real life it is usually more complicated than that, successful attacks have been proposed based on this simple idea.

This scenario suggests a straightforward mathematical structure for an *encryption scheme*: An *encryption algorithm* transforms a *plaintext message* into a *ciphertext message* before it is sent over the insecure channel, and a corresponding *decryption algorithm* reverses this transformation.

All algorithms are available to the adversary. This way, the cryptographic community can scrutinize them and assess their security. Where they are kept secret, the team of engineers that develops them will usually overlook something that a team of reverse engineers slightly more creative and slightly more determined will find and exploit. Of course, only the legitimate receiver of a message should be able to decrypt an encrypted message. To enforce this requirement, both encryption and decryption algorithm require as additional input besides the message a value that is only known to the communicating parties, the *key*. The algorithm for key generation completes the description of an encryption scheme.

Definition 2.2 (Encryption scheme)**.** *An encryption scheme is a tuple* (G, E, D) *of three probabilistic polynomial-time Turing machines (PPTMs). The encryption algorithm*

$$E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$$
$$E_k(m) \mapsto c$$

computes a ciphertext $c \in \mathcal{C}$ *from a corresponding plaintext* $m \in \mathcal{M}$ *and a key* $k \in \mathcal{K}$*; the decryption algorithm*

$$D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$$
$$D_k(c) \mapsto m$$

reverses encryption; and the key generation algorithm

$$G : \{1\}^* \to \mathcal{K}$$
$$G(1^N) \mapsto k$$

maps integer numbers in unary notation ($\{1\}^*$ *denotes the set of all finite sequences of ones) to random keys from* $\mathcal{K}$*. An encryption scheme must satisfy the condition that decryption recovers the plaintext. In other words, encryption must be injective and invertible:*

$$\forall k \in \mathcal{K}, m \in \mathcal{M} : D_k(E_k(m)) = m$$

More accurately, since both E *and* D *are probabilistic we require that*

$$\Pr\left[\forall k \in \mathcal{K}, m \in \mathcal{M} : D_k(E_k(m)) = m\right] \geq 1 - \mathit{neg}(N)$$

The argument of G, a natural number N passed in binary notation 1^N mostly because it is a natural form for TMs to digest, is called the *security parameter*. Its necessity stems from the asymptotic nature of security: An encryption scheme is designed so that there is no adversarial algorithm in $\mathcal{P}$ that can do it any harm, i.e., every adversarial algorithm performs worse than any polynomial *for any* $n \geq N$. So in order for the system to be secure, we always need to choose an appropriate N first.

Because key management is often straightforward in our applications, we sometimes omit the key entirely, writing $E(m)$ and $D(c)$. Keep in mind that even then, security still depends on the security parameter implicit in the key.

It is easy to see why key generation needs to be probabilistic: An adversary who is kept from observing G's output must not be able to guess it, so G's output should in fact be uniformly random. The reason why E and D are probabilistic is less obvious, and will be discussed in the following.

2.3.4 Security Definitions

The following encryption scheme encrypts any plaintext to itself and decrypts any ciphertext to itself:

Example 2.1 (A Weak Encryption Scheme)**.**

$$\begin{aligned} G(1^N) &= 1^N \\ E_k(m) &= m \\ D_k(c) &= c \end{aligned}$$

(G, E, D) satisfies Definition 2.2. In particular, it is true that $D(E(m)) = m$ for all keys and messages, but something is still wrong: There is a very simple way for the adversary to decrypt ciphertexts (namely, by doing no transformation at all).

This shows that we need a notion of security of a cryptographic scheme that grasps the intuition that the adversary "should not figure out the plaintext". This notion should be as mathematically rigorous as possible so that we can be sure not to miss any attac angle.

Simulations (or *games*) in which the players follow the rules of the scheme while an adversary (who may be one of the honest players, such as a service provider, or an outside enemy, such as an eavesdropper) attempts to corrupt its outcome. The scheme is secure if the players can be confident that the adversary will lose any possible game (or more accurately, that her winning probability is negligible). We refer to such games as *security definitions*, or

notions of security. An example for such a game that is rather imperfect (in a moment it will become clear why) and thereby instructive is this:

Example 2.2 (A Weak Notion of Security)**.** *The system is secure if there is no adversarial PPTM A such that for every c, it holds that* $A(c) = D_k(c)$.

(Note that A is not provided with the secret decryption key.) If the players use the bad scheme from Example 2.1, the adversary chooses $\forall c \in \mathcal{C} : A(c) = c$ and plaintexts are always recovered in full. Hence, the scheme is not secure by this definition. And yet, an encryption scheme that securely hides the first bit of the message from the adversary but passes through all other bits unmodified is secure in this definition. But the fact that it leaks all but one bit of the plaintext to the adversary means it is only negligibly better than our first approach.

Before we look at better formalizations of what constitutes a secure encryption scheme, we need to discuss a few common mistakes that need to be avoided in the representation of the adversary, of her goal, and of the application that makes use of the scheme.

Assumptions on adversarial strategies and algorithms. First, it is not enough to consider any restricted class of adversarial algorithms. During the construction of a secure system, it is unknown what attacking algorithms an adversary will come up with. The adversary has the advantage of being confronted with the complete system before she has to make any decisions on how to attack. Once she comes up with an attack, the system designer can do very little about the design that has been finished earlier, even if the attack becomes publicly known. Therefore, security proofs are necessarily existential, i.e., of the form: *There is no attacking algorithm A such that...*. We will come back to this point in Section 3.6 in an attempt to give further motivation for the cryptographer's position in this argument.

On the other hand, making assumptions on the computational resources of the adversary can be justified. It is unlikely that any real system will ever face an adversary that can solve $\mathcal{NP}$-complete problems of a properly chosen size in the life time of the system under scrutiny.[5]

Partial information recovery by the adversary is harmful, too. Second, it needs to be decided what goal the adversary needs to accomplish in order to render a scheme insecure. Ultimately, she would of course be happy to recover the plaintext in full as required in Example 2.2. However, a basis for

[5]This, of course, is also an assumption, and can be rejected. However, it is more comforting to base a system's security on the assumption that the adversary will not solve a mathematical problem which generations of mathematicians world-wide have failed to solve than to assume that all effective attacks have been taken into account by a small group of engineers.

security proofs needs to be more conservative. Even if a small fraction of the plaintext is recovered successfully, e.g., a fraction of all confidential e-mails, the users of an encryption scheme may get burnt.

Assumptions on adversarial a priori knowledge. Further, the adversary also needs to be granted partial information on the plaintext. It cannot be made the responsibility of the encrypting parties to reveal absolutely no information to an adversary, because much of this information may be public. Figures from a company's share holder reports, the structure imposed on messages by industry standard protocols, or seemingly insignificant information provided by unsuspecting employees, suppliers, or clients, can all lead to disaster if the security of the encryption scheme relies on the adversary having no knowledge of the plaintext whatsoever.

Example 2.3 (Partial Plaintext Recovery Hurts)**.** *A supply chain management (SCM) application uses an application-layer protocol to send queries from the client to the service provider and results from the service provider back to the client. The encrypted communication stream of company A with the service provider is intercepted by competitor B's business intelligence unit.*

Because B requires the same products for its operation, B has fuzzy, but detailed knowledge about A's supply chains. It knows many of A's suppliers, what parts they supply, and at what likely price. It may also know the output of A, and what input that requires (which parts, and how many). Also, as A and B are clients to the same SCM service provider, B is in possession of the exact specification of the plaintext protocol.

Now B wants to address specific questions, and an answer to any of these question provides some benefit to B and does some damage to A. For an order of a known number of a known part, is the unit price that A pays lower than the one that B pays? Do patterns in the order volumes suggest that A has found a way to produce the same output as B with less input? Is the list of suppliers of A any different from what B expects? The reader will find it easy to extend this list ad infinitum.

The adversary may even be able to gain partial access to the secret key, in the form of ciphertexts to which she not only *knows*, but *chooses* the corresponding plaintexts (this circumstance is called adversarial oracle access; see Definition 2.1):

Example 2.4 (The Adversary may be able to Encrypt)**.** *B may contact A and talk an employee into triggering an SQL query to the SCM database, and then observe the traffic between A's site and the service. There are two messages, and A knows the complete plaintext of the first one, the query, and*

thus the exact meaning of the second one, the reply. If the answer contains a list of SQL tuples, B can compute the tuple count from the ciphertext without decryption, from the size of the ciphertext and some knowledge of the database scheme alone.[6]

If the adversary already suspects a certain message to be sent over the wire in a context that she either predicts or actively provokes, she only needs to recover one last bit, namely the answer to the yes-no-question: *Was it message m, or was it another one?* If the suspected contents of the message is a stock transfer order, the knowledge leaked to her with this one bit may be disastrous.

(Note that when we are talking about "one bit of information" in the last paragraph, we do not mean one bit of the plaintext representation of the message, but one bit in the information theoretic sense (Sha48; Mac03; Sti05): One bit of information can be smeared over many bits in the actual message, and be only explicit in the value a boolean function applied to the entire plaintext message.)

Assumptions on applications. Finally, although adversarial algorithms usually benefit greatly from being tailored for specific applications, this does not hold for the user of the encryption scheme. Making assumptions on what part of the ciphertext needs to remain confidential and what can be given away is difficult and risky.

This has two reasons. One is that is that software engineering is mostly about managing changes in requirements. Users are likely to learn when it is too late that something that has been specified as non-confidential turns out to be confidential after all. But the second one is more important: The database has internal functional dependencies that are extremely hard to formalize and understand. A partial encryption scheme has to be protected against an attack in which the adversary computes confidential contents of the database as the value of a function of publicly available information.

Work has been carried out to control these functional dependencies and allow for publication of partial messages while keeping other parts reliably confidential. However, these approaches usually make two other mistakes that we have already treated above: The violate the principles of arbitrary adversarial algorithm and of adversarial a priori knowledge. Particularly the latter one is further complicating the issue considerably. See Section 3.6 for a discussion.

To conclude, a sound and reliable security model for encryption schemes

[6] We assume that ciphertext length is roughly the same as plaintext length. All relevant encryption schemes have this property, since the performance penalty for padding does not justify the minimal gain in security.

- makes no assumptions on the adversarial strategy or algorithm, but only on her computing resources,
- considers an adversary who knows everything about a plaintext except for one last (information theoretic) bit, and attempts to recover that bit,
- might even grant her access to encryption or decryption oracles (see Definition 2.1), and
- is not tailored towards any specific application.

These characteristics have been discussed and motivated or simply implied in most text books on security in varying depth (Gol01; Gol04; Sta03; Sch96; MvOV01). They form a solid consensus from which few security engineers deviate, and if they do risk disaster (Neu85). The weaknesses we uncover in related work in Chapter 3 can be blamed to a large extent to violations of these principles.

Doing it right

Equipped with these principles, we can now look at a few more useful security definitions. The first, a basic form of semantic security, is both sound and relatively intuitive. The motivation of the second one, indistinguishability, might be more surprising, but it has its technical merits. Conveniently, the two turn out to be equivalent.

Definition 2.3 (Semantic Security)**.** *An encryption scheme (G, E, D) is semantically secure (has semantic security) in the known plaintext model iff for any plaintext m and any probabilistic polynomial-time adversarial algorithm A, there is an adversarial PPTM B such that*

$$|\Pr\left[A(E_k(m), 1^{|E_k(m)|}) = 1\right] - \Pr\left[B(1^{|E_k(m)|}) = 1\right]| \leq neg(N)$$

where $k = G(1^N)$.

Intuitively, any access to a ciphertext does not add to the adversary's knowledge about the corresponding plaintext: Everything that can be computed by A using the size of the ciphertext and the ciphertext itself, can also be computed by an B using the size of the ciphertext only.

Reading security definitions as games gives us a good intuitive understanding, which is crucial to accept (or reject) them for a given application as appropriate (or too strong, or too weak). But where is the game in this definition?

First, the honest user decides on A and notifies the adversary, then the adversary chooses B. In the next step, the honest user provides the adversary with input for B and lets her run B. She runs A on the corresponding input herself, and both players compare their results. If they match, the adversary has guessed correctly (wins), if not, she was wrong (loses). This is played for a number of rounds until the sample set allows for computing approximations of the probabilities occurring in the definition above. If and only if the adversary's winning probability is negligible, the definition will hold.

Although this may capture our feeling for what it means for an encryption scheme to be secure quite well, it turns out to produce clumsier proofs than the following

Definition 2.4 (Indistinguishability)**.** *An encryption scheme* (G, E, D) *is* indistinguishably secure (has indistinguishability) *iff for any adversary PPTM* A *and any* $m_0, m_1 \in \mathcal{M}, m_0 \neq m_1$*,*

$$\Pr\left[\begin{array}{rcl} k & \leftarrow & G(1^N) \\ i & \leftarrow & RND^{\{0,1\}} \\ i^* & \leftarrow & A(m_0, m_1, E_k(m_i)) \\ i = i^* & & \end{array}\right] \leq \frac{1}{2} + neg(N)$$

Again, we rephrase this as a game. The adversary chooses A, m_0, m_1, and the user tosses a coin and publishes $E_k(m_0)$ if the coin turns up heads and $E_k(m_1)$ if it turns out tails. The adversary wins if she can guess what happened in non-negligibly more than half of all runs.

Note that knowledge about m_0, m_1 can be build into A^*'s state graph, so strictly speaking, this makes it unnecessary to provide them as arguments. However, making them explicit makes the intention behind the definition more apparent. A^* knows everything about the encrypted plaintext except for one bit (namely, which of the two it is). An indistinguishable encryption scheme renders guessing this one last bit impossible from the information available to the adversary (namely, the ciphertext).

Indistinguishability is often more suitable for security proofs,[7], while semantic security intuitively seems better suited for describing practical security. Luckily, the two are interchangeable:

Theorem 2.1. *An encryption scheme* (G, E, D) *has semantic security iff it has indistinguishability.*

For a proof see cf. (Gol04) (pp. 383).

There is a definition of indistinguishability that looks different, but turns out to be equivalent to the above:

[7] And insecurity proofs, too!

Definition 2.5 (Indistinguishability: Variant)**.** *An encryption scheme* (G, E, D) *is* indistinguishably secure (has indistinguishability) *iff for any adversarial PPTM* A^* *and any* $m_0, m_1 \in \mathcal{M}, m_0 \neq m_1$*,*

$$\left|\Pr\left[\begin{array}{rcl} k & \leftarrow & G(1^N) \\ A^*(m_0, m_1, E_k(m_0)) & = & 1 \end{array}\right]\right.$$
$$\left. - \Pr\left[\begin{array}{rcl} k & \leftarrow & G(1^N) \\ A^*(m_0, m_1, E_k(m_1)) & = & 1 \end{array}\right]\right| \leq neg(N)$$

Theorem 2.2. *An encryption scheme* (G, E, D) *is secure in Definition 2.4 iff it is secure in Definition 2.5.*

Again, for a proof see cf. (Gol04).

Cryptographic literature knows several variants of indistinguishability that grant the adversary additional information in the form of limited access to oracles (see Definition 2.1) that perform encryption or decryption operations. Adversaries with oracle access to the cryptographic scheme under attack are called *active adversaries*. The idea of oracle access yields the following 2.4.[8]

Definition 2.6 (Chosen Plaintext Security)**.** *An encryption scheme* (G, E, D) *is (indistinguishably) secure in the* chosen plaintext model *iff for any probabilistic polynomial-time adversarial PPTM* A^**, and* $m_0, m_1 \in \mathcal{M}, m_0 \neq m_1$*,*

$$\Pr\left[\begin{array}{rcl} k & \leftarrow & G(1^N) \\ i & \leftarrow & RND^{\{0,1\}} \\ i^* & \leftarrow & A^{*E_k}(m_0, m_1, E_k(m_i)) \\ i = i^* & & \end{array}\right] \leq \frac{1}{2} + neg(N)$$

That is, A has now access to an encryption oracle. Now we can say explain why we defined E to be probabilistic in Section 2.3.3. If E_k was deterministic, i.e., two different runs of $E_k(m)$ would produce the same output, the adversary could simply have the oracle encrypt m_0 and compare the result with the ciphertext in its challenge. If the two are equal, the ciphertext is $E_k(m_0)$; if not, it is $E_k(m_1)$. Thus, a deterministic encryption scheme cannot be secure in this definition.

We can fix this definition to make it weaker, and allow for deterministic encryption: We simply replace the oracle E_k by

$$E_k^{\setminus\{m_0,m_1\}}(m) = \begin{cases} 0 & \text{if } m \in \{m_0, m_1\} \\ E_k(m) & \text{otherwise} \end{cases}$$

[8] There are natural analogs of the following definitions for semantic security that do not provide any further insight, so we will not consider those here.

However, keep in mind that this is a considerably weakens Definition 2.6. Whenever it is used, great care must be taken that it does not break security altogether.

Two further variants based on oracles describe, with increasing adversarial power, a priori and adaptive chosen ciphertext adversaries.

Definition 2.7 ((A Priori) Chosen Ciphertext Security)**.** *An encryption scheme* (G, E, D) *is (indistinguishably) secure in the* (a priori) chosen ciphertext model *iff for any pair of adversarial PPTMs* A, B *and any pair of messages* $m_0, m_1 \in \mathcal{M}, m_0 \neq m_1$*:*

$$\Pr\left[\begin{array}{rcl} k & \leftarrow & G(1^N) \\ i & \leftarrow & RND^{\{0,1\}} \\ \omega & \leftarrow & B^{E_k, D_k}(m_0, m_1, 1^N) \\ c & \leftarrow & E_k(m_i) \\ i^* & \leftarrow & A(m_0, m_1, 1^N, \omega, c) \\ i = i^* & & \end{array}\right] \leq \frac{1}{2} + neg(N)$$

ω is the a priori computation made by the adversary using oracle encryption *and* decryption. (Again, since B is in $\mathcal{P}$, there may be only polynomially many queries.)

Definition 2.8 (Adaptive Chosen Ciphertext Security)**.** *An encryption scheme* (G, E, D) *is (indistinguishably) secure in the* adaptive chosen ciphertext model (ACC) *iff for any pair of adversarial PPTM A and any pair of messages* $m_0, m_1 \in \mathcal{M}, m_0 \neq m_1$*:*

$$\Pr\left[\begin{array}{rcl} k & \leftarrow & G(1^N) \\ i & \leftarrow & RND^{\{0,1\}} \\ c & \leftarrow & E_k(m_i) \\ i^* & \leftarrow & A^{E_k, D_k^{\setminus\{c\}}}(m_0, m_1, 1^N, c) \\ i = i^* & & \end{array}\right] \leq \frac{1}{2} + neg(N)$$

As before, $D_k^{\setminus\{c\}}$ denotes an decryption oracle that decrypts anything but the ciphertext c under attack.

In practice, chosen ciphertext attacks may only reveal noisy plaintext information such as non-uniform probability distributions over the number of 1s in the plaintext (Koc96; Ble98). Such attacks are successful because of the high pace with which things happen in information systems: In order to recover a few kilobytes, millions of noisy decryption operations can be performed in a short period of time, and extensive computations on the noisy oracle answers can be used to recover the original plaintext or the

secret key. Therefore, the model grants the adversary clean access to an encryption oracle. If this does not help against a given scheme, we may hope to be secure using that scheme in practice as well.

We have stated that in general, it is a bad idea to make any assumptions on the adversary's context knowledge. However, despite its deficiencies, it appears in the literature every now and then, and can arguably be used to provide at least a better level of security than none at all. Therefore, where unavoidable we will drop the principle of adversarial context knowledge and relax Definition 2.4:

Definition 2.9 (Known Ciphertext Security)**.** *An encryption scheme* (G, E, D) *is indistinguishably secure (has indistinguishability) in the known ciphertext model iff for any two plaintexts* $m_0, m_1 \in \mathcal{M}, m_0 \neq m_1$ *and any probabilistic polynomial-time adversarial algorithm* A*,*

$$\Pr\left[\begin{array}{rcl} k & \leftarrow & G(1^N) \\ i & \leftarrow & RND^{\{0,1\}} \\ i^* & \leftarrow & A(E_k(m_i)) \\ i = i^* & & \end{array}\right] \leq \frac{1}{2} + neg(N)$$

All these models can be put into a hierarchy of strength as depicted in Figure 2.1.

Theorem 2.3 (Hierarchy of attack models)**.**

1. *Indistinguishability (Definition 2.4) implies known ciphertext security (Definition 2.9)*
2. *Chosen plaintext security (Definition 2.6) implies indistinguishability (Definition 2.4)*
3. *(A priori) chosen ciphertext security (Definition 2.7) implies indistinguishability (Definition 2.4)*
4. *Adaptive chosen ciphertext security (Definition 2.8) implies adaptive chosen ciphertext security (Definition 2.7)*

Proofs derive the information required for the weaker attack from the information available in the stronger attack. Most are straightforward. For instance, if the adversary gets granted access to an oracle while the game otherwise stays the same, every attack that works without access to the oracle will still work. Hence, chosen plaintext security is strictly more secure than basic indistinguishability. The full proofs can be found in (Gol04).

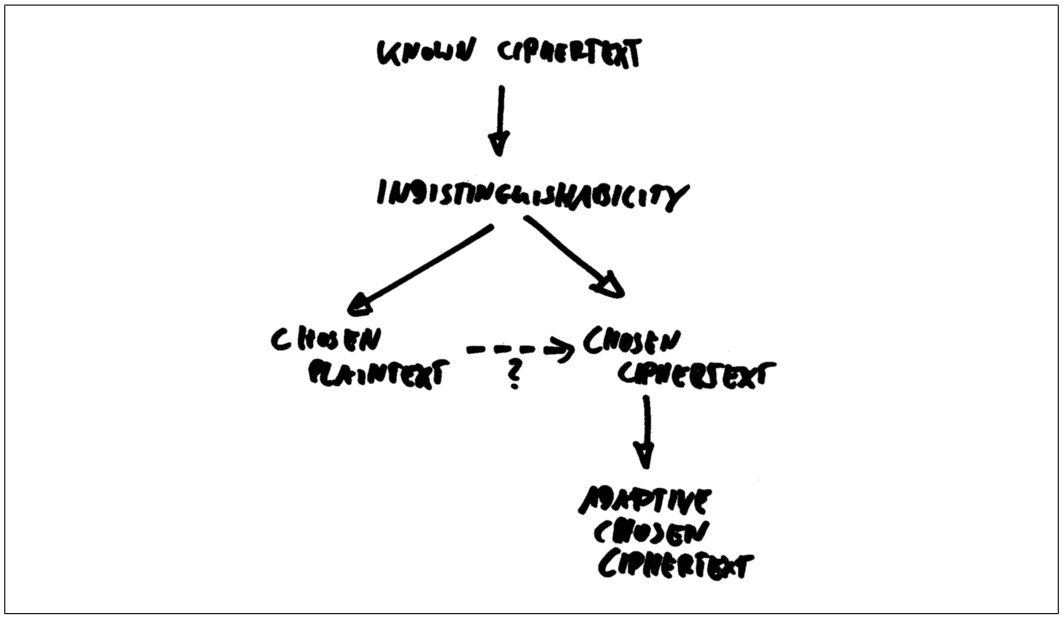


Figure 2.1: Hierarchy of attack models (illustration of Theorem 2.3).

Note that there is no reduction from chosen ciphertext security to chosen plaintext security: Although an adversary with access to a decryption oracle appears to be closer to its goal of decrypting secret messages than one with merely access to an encryption oracle, no reduction has been found so far. This is mostly a theoretical deficiency. Assessing the security of a particular scheme, a proof in one model often yields a similar proof in the other.

Adaptive chosen ciphertext is the strongest model, granting the most resources to the adversary, and there are good reasons to use this model instead of one of the weaker variants (Sho98). In realistic scenarios, the adversary has extensive capabilities of mangling authentic messages and forging new ones. There have been effective attacks against widely deployed cryptographic products caused by the wrong choice of security model (Ble98).

We have to add one last, orthogonal aspect to the hierarchy security notions developed in Theorem 2.3. We often want to use an encryption scheme over a longer period of time to encrypt many messages with the same key. (Otherwise, for each message transmitted securely over an insecure channel, we would have to transmit a key over a secure channel as well.) The extension of the above definitions to the multiple-message case is rather straightforward. We only list the variant for indistinguishability (the others can be found in (Gol04)).

Definition 2.10 (Multiple-Message Indistinguishability)**.** *Let* $\bar{m} = (m^1, \ldots, m^{O(N)})$ *be a sequence of messages of length polynomial in the security param-*

eter, and let $\bar{\mathcal{M}}$ be the set of sequences of polynomial length over elements of $\mathcal{M}$. Further, for any PPTM M and message sequence $\bar{m}$, let $\bar{M}(\bar{m})$ be the result of running M on all elements of $\bar{m}$ subsequently.

An encryption scheme (G, E, D) is indistinguishably secure (has indistinguishability) in the multiple-message case *iff for any adversarial PPTM A, and any $\bar{m_0}, \bar{m_1} \in \bar{\mathcal{M}}$,*

$$\Pr\left[\begin{array}{rcl} k & \leftarrow & G(1^N) \\ i & \leftarrow & RND^{\{0,1\}} \\ i^* & \leftarrow & A(\bar{m_0}, \bar{m_1}, \bar{E_k}(\bar{m_i})) \\ i = i^* & & \end{array}\right] \leq \frac{1}{2} + neg(N)$$

Note that the contents of M's randomness tape is different for each invocation of M in $\bar{M}(\bar{m})$. If two of the messages in $\bar{m}$ are the same, the output may still be different, because M tosses its coins anew each time.

Whether one should consider multiple-message or single-message security is always apparent from context and application: If the same key is used for a series of messages, single-message security is inappropriate. If not (for instance because the key is generated ad hoc in the context of a larger protocol and discarded in the next step) it is the better choice because it is easier to achieve.

We conclude this section with a few musings on the virtues and dangers of intuition. All of the thoughts expressed here are common sense. Some will re-emerge in Section 4.1 when we have a look at previously proposed security model for homomorphic encryption.

If a mistake has been made in the design of an airplane, the affected machines can be taken down after the first serious hint to trouble, and the pace of events will prevent a long series of crashes. On the Internet, an engineering mistake propagates at a pace that nobody has a chance to react until, to stay in the metaphor, hundreds of airplanes have crashed into the neighborhoods of hundreds of airports. The reason why this has not caused higher casualties yet is that most people, not only experts, are aware of this and avoid relying on the Internet to the same extent they rely on the safety of civil aviation.

Both scenarios deploy heuristic security precautions, which is the best that can be done given the resources available and the hardness of the task. Unfortunately, the heuristics of non-communication engineering do not work for data network engineering any more, since the nature of the two is fundamentally different, and thus so are the threats that both have to deal with.

- **O**pportunity boost. An adversary after the contents of your tool shed needs to meet two requirements in order to be successful: She needs

to know of your tool shed, and she needs to be ready to go through the trouble to get there. On the Internet, something whose presence is not publicly announced cannot automatically be considered secret. For example, if a flaw in a software server product is publicly announced, tools can be used to scan the entire IP address space for running instances of that software automatically. Further, there are no geographic distances any more: Every IP address can be reached effortlessly by the adversary. This increases the range of potential adversaries by orders of magnitude.

- **S**calability of attacks against information systems. It is a common misconception that "everything that slows an adversary down is good." This is in fact true for house locks, because it will slow down every adversary, in every instance of an attack. Time is very scarce for the adversary, and every minute she needs to spend on the attack increases the probability to get caught on site. However, attacks against information technology almost always come with outrageous scalability benefits due to attack automation.

  Breaking a cryptographic scheme is not like breaking a lock, but like crafting a key that fits all the locks of a certain brand in the world at the same time. If a standard cryptosystem is broken, *every* house in the entire world that is protected with it can be opened without additional effort. Hence, while it may be justified to obscure a system to establish an additional reverse engineering obstacle for the adversary, one always has to keep in mind that the benefit from doing the reverse engineering is higher by many orders of magnitude than in a traditional context. This is the reason *security by obscurity* has such a bad reputation in the computer security community.

- **R**emote attacks. The adversary need not be physically present at the site of the attack. Also, the goods stolen in data networks are not really stolen in a physical sense, but interception usually goes unnoticed. This has a number of negative effects on the effectiveness of traditional security heuristics. (1) A lower motivation is sufficient for an adversary to decide to engage in an attack, since the risk of getting caught is low. (2) The attack against a crypto scheme can be more resource intensive than an attack against a vault, since the ciphertext can be attacked in a hidden laboratory, and the attack may take many weeks and still be worth it. Finally, (3) it amplifies the effects of scalability effects weakening the effect of system obscurity on system security.

- **M**onocultures and centralization. Centralization promises synergy effects. Operating systems and mass application software arguably form natural monopolies that are hard (and probably undesirable) to counter on a political level. Unfortunately, both work together to increase the threats to the data that is processed by these systems. Centralization in computing centers run by service providers multiply the value of an attack target, while monocultures improve the scalability of automated attacks. Furthermore, so called bot networks (see e.g. (LAAA06)) can be set up by spreading malware, yielding exponential size/effort ratios. The computing resources available from these stolen networks can then be used to successfully launch attacks of intimidating complexity.

All these observations disembogue into the conclusion that we need to think harder about what we mean if we say an IT system needs to be secure, or confidential. New intuitions have to be developed for the capabilities of the adversary and the nature of the threats in general. The security definitions presented in this section span the current state of the art, but certainly do not mark the end point of the debate.

2.3.5 Other Cryptographic Building Blocks

Public Key Encryption

Encryption schemes (G, E, D), as introduced in Section 2.3.4, are more accurately called a *symmetric encryption schemes*. Symmetric encryption is characterized by the symmetric nature of $\mathcal{K}$: Both encryption and decryption algorithm are fed the same secret key, and once this key is revealed to the adversary, she can use it both for encrypting new plaintexts and decrypting intercepted ciphertexts.

Asymmetric encryption, or *public-key* encryption, has a key generation algorithm that generates *key pairs* $(e, d) \in \mathcal{K}$. Encryption

$$E : \{e|(e, d) \in \mathcal{K}\} \times \mathcal{M} \mapsto \mathcal{C}$$

relies on the public part only, only decryption

$$D : \mathcal{K} \times \mathcal{C} \mapsto \mathcal{M}$$

relies also on the secret part as before. In order for the system to be secure, knowledge of e must be useless for recovering the corresponding d.

An asymmetric system *allows for* the public key e to be known to the adversary, but does not *require* it. An adversary with strictly less information

can only be less successful. On the other hand, because they are based on mathematical structures that allow for functions that are hard to compute given some (m, e), but become easy given some (m, e, d), the performance of public-key cryptosystems is worse than that of symmetric key systems: The latter are based on relatively small circuits and highly efficient boolean operations. Therefore, even in applications that depend on the fact that key distribution is easier and therefore use public key cryptography, the bulk of the message is usually encrypted with a symmetric session key that is then encrypted in an asymmetric fashion (hybrid encryption).

Asymmetric cryptography is used in industry standards like SSL/TLS (DA99) and PGP (Zim95). The most prominent underlying algorithms are RSA (RSA77) and ElGamal (Gam85) on residue class rings.[9] Considerably younger but maturing rapidly are a number of computationally more efficient schemes based on elliptic curve groups (Was03). An efficient and provably secure system has been proposed (CS98), but we know of no widely used implementation.

Public-key cryptography is simplifying key distribution enormously. This has made it popular for e-commerce applications and given way for a large industry providing public-key infrastructure (PKI): If every client of a bank would need to negotiate a symmetric key before being able to use online-banking, not only would enormous computing and administrative resources be required by the bank to manage the vast number of keys make the business prohibitively expensive, but acceptance in the user base would likely deteriorate even independent of the increase in cost, solely because of the effort to handle the technology.

Homomorphic encryption (see Section 2.3.6) has been discovered on asymmetric encryption schemes, namely RSA (RAD78); symmetric schemes usually display no homomorphic structure. To our knowledge, we are the first to have used the term for schemes and applications outside asymmetric cryptography. This has proven a good idea in several ways. When looking at privacy of information stored in outsourced databases, there is usually no point in allowing the adversary access to the encryption key in our model: All encryption and decryption is carried out within the security realm of the database client (or several clients), and the adversary has only access to the server. The investigation into symmetric cryptography has resulted in efficient practical results that would have easily been overlooked otherwise (see Section 4.3). Finally, note that the move from the asymmetric to symmetric model is not a restriction, but a generalization: If the application requires

[9] We will explain the working of RSA in Section 3.1.2, since it turns out that RSA can be turned into a homomorphic encryption scheme very easily.

an encryption scheme that is secure in the symmetric case, any asymmetric scheme will do (even though it does not provide any additional benefits).

Pseudo-Randomness

Later in this thesis, we will make use of a class of symmetric encryption schemes called stream ciphers. These require a device that creates a long sequence of pseudo-random bits from a small fix number of input bits. This device is called a *pseudo-random generator*.

Definition 2.11 (Pseudo-Random (Bit) Generator)**.** *A* pseudo-random (bit) generator *is a (deterministic) PTM*

$$R : \{0,1\}^l \to \{0,1\}^m$$

where $m = O(2^l)$.

Examples for pseudo-random generator algorithms are linear shift feedback register constructions (see cf. (Sch96)) such as used in the global positioning system GPS, or the less efficient and little used, but also more secure Blum-Blum-Shub algorithm (BBS82; BBS86).

Again, security of pseudo-random generators is defined in terms of an indistinguishability game. A good pseudo-random generator makes it impossible for an adversary to tell the output apart from true randomness:

Definition 2.12 (Pseudo-Random Generator Security)**.** *A pseudo-random generator* $R : \{0,1\}^l \to \{0,1\}^m$ *is secure if for every adversarial PPTM A:*

$$\Pr[A(R_k) = 1] - \Pr[A(r) = 1] \leq neg(m)$$

where $k \in \{0,1\}^l$ *and* $r \in \{0,1\}^m$ *are chosen uniformly at random.*

From the perspective of the engineer, this raises the question of how a PRG is *seeded*, or how the seed k is chosen in a truly random process. This touches a rather philosophical question: What makes a sequence random in the first place? The problem of deciding for an individual sequence of bits whether it is random or not seems paradoxical — is 111111 more random than 101101? But even the problem of determining the level of randomness of languages (i.e., sets of sequences of bits rather than individual words), has puzzled mathematicians for centuries ((Vol02) is an excellently written survey, covering and exceeding the foundations of randomness laid in (Gol01)). However in practice, good sources of true randomness exist.[10] We will contend ourselves with assuming there is such a thing as a uniformly random

[10]Examples: Intel RNG, VIA PadLock Security Engine.

choice, without proving for any particular device that it falls into this category.

The following important result states that the number of samples (as long as it remains polynomial) does not affect the distinguishability of a PRG:

Theorem 2.4 (Pseudo-Random Generator Security for Multiple Samples)**.** *Given a PRG R and a PPTM A. If*

$$\Pr[A(R_k) = 1] - \Pr[A(r) = 1] \leq neg(m)$$

for uniformly chosen $k \in \{0,1\}^l$ and $r \in \{0,1\}^m$, then

$$\Pr[A(R_{k_0}), \ldots, R_{k_z})) = 1] - \Pr[A(r_0, \ldots r_z) = 1] \leq neg(m)$$

for any z and uniformly chosen $k_i \in \{0,1\}^l$ and $r_i \in \{0,1\}^m$.

(There is no need to restrict z, since A is polynomial and can therefore only process polynomially sized input.) A proof based on probability ensembles can be found in (Gol01). This result allows us to re-use the same algorithm in the same application with different keys, without sacrificing security. We will need this in the construction of our homomorphic database encryption schemes in Section 4.3.2.

We will also require the notion of *pseudo-random functions* and *pseudo-random permutations* that can be easily obtained from the definition of pseudo-random generators.

Definition 2.13 (Pseudo-Random Function)**.** *A* pseudo-random function *is a function $f : \{0,1\}^l \to \{0,1\}^m$ for some l, m.*

Definition 2.14 (Pseudo-Random Function Security)**.** *A* pseudo-random function $f : \{0,1\}^l \to \{0,1\}^m$ *is secure iff for every adversarial PPTM A,*

$$\Pr[A(f(k)) = 1] - \Pr[A(r) = 1] \leq neg(|k|)$$

where $k \in \{0,1\}^l, r \in \{0,1\}^m$ are chosen uniformly at random.

Definition 2.15 (Pseudo-Random Permutation)**.** *A* pseudo-random permutation *is a permutation $f : \{0,1\}^l \to \{0,1\}^l$ that is a pseudo-random function.*

The main purpose of pseudo-random functions (and permutations) in a cryptographic scheme is to generate outcomes that surprise the adversary from input that she knows of. This is impossible if the function is defined in the scheme and the adversary knows the scheme, so we need to *key* them. We omit the technical details (which differ only subtly from keying of encryption schemes that we have explained more thoroughly) here. We simply write f_k for a secret pseudo-random function (or permutation): k is a secret uniformly chosen key whose size is as usual determined by the security parameter, and f is known to the adversary.

One-Wayness

Cryptographic schemes always boil down to enforcing situations in which one agent is capable of doing a computation efficiently, and another one is not. Often, these situations can be described by a function that is easy to compute, but whose inverse is hard to compute. These functions are called one-way functions.

Definition 2.16 (One-Way Function)**.** *A* one-way function *is a function* $f : \mathcal{A} \to \mathcal{B}$ *such that*

- *There is a PPTM* M *such that for uniformly chosen* $a \in \mathcal{A}$*,*

$$\Pr[M(a) = f(a)] \geq 1 - neg(|a|)$$

- *For every PPTM* A *and uniformly chosen* $a \in \mathcal{A}$*,*

$$\Pr[A(f(a), 1^{|a|}) \in f^{-1}(f(a))] \leq neg(|a|)$$

In analogy to the grand open question of complexity theory whether $\mathcal{P} = \mathcal{NP}$, the grand open question of cryptography is whether there are functions that satisfy Definition 2.16.

And as in complexity theory, where many theorems begin with *unless* $\mathcal{P} = \mathcal{NP}$, ..., many fundamental results in cryptography have the form *if one-way functions exist...*.

For example, an encryption function E can be thought of as a one-way function if both key k and message m are taken as its input: If it is provably infeasible for an adversary to compute any valid (k, m) from $E_k(m)$, then the message is provably safe.

Further applications are numerous, such as producing a secure small fix-sized *witness* or cryptographic *checksum* for some large message. For example, the message could be a large contract in an electronic format, and the witness would be the input to a signature scheme (see Section 2.3.5) that can only handle small inputs. If the witness that comes with the document is intact and produced by a one-way function, the document is intact as well.

A function that computes witnesses from arbitrary-sized documents and that is hoped to be one-way is called a *hash function*.

Definition 2.17 (Hash Function)**.** *A* hash function *is a one-way function* $h : \{0,1\}^* \to \{0,1\}^l$ *for some fixed* $l \in \mathbb{N}$*. (A common value is* $l = 160$*.)*

Since one-wayness is out of reach, security of hash functions can be defined in several ways. For example, *collision resistance* states that for any given

x such that $h(x) = y$, it is hard to find an $x' \neq x$ such that $h(x') = y$. (x' is called a collision.) An orthogonal notion is *pre-image resistance*, which states that for any y it is hard to find an x such that $h(x) = y$.[11]

Widely used hash functions are MD5 (Riv92) and the SHA family (rJ01; rH06). Cryptographic hash function research has been in some turmoil recently. Theoretical weaknesses in SHA-1 indicate that it may soon be broken entirely (WYY05a; WYY05b), while MD5 has been known to be insecure for a long time (Ber92; Dob96). This has incalculable consequences – these functions are used in industry standards and legislation for electronic banking, electronic contracting, electronic passports and visa, and many others. Further, they are crucial for the security of virtually all larger cryptographic applications. In particular, SSL/TLS, PGP, SSH are directly affected.

Block Ciphers

There are essentially two classes of symmetric encryption algorithms: Block ciphers and stream ciphers. Before explaining the latter in the next section, we will now give an account of *block ciphers*.

Definition 2.18 (Block Cipher)**.** *A* block cipher *is a (symmetric) encryption scheme* (G, E, D) *with* $\mathcal{M} = \mathcal{C} = \{0,1\}^k$*. I.e., ciphertext space and plaintext space consist of bit words of fixed* block size k*.*

In practice, the block size usually varies between 128 and 512 bits (16 and 64 bytes). The most famous block cipher today is the advanced encryption standard (AES) (DR02). Its predecessor DES (ACF$^+$77) is slowly removed from the production cycle mostly due to key sizes that have grown insufficient to protect against vastly more powerful hardware.

We can make two observations on block ciphers:

1. The plaintext and ciphertext space are equal in size and every ciphertext needs to be mappable on the plaintext used to compute it. Hence, a block cipher must be a *bijection* from $\mathcal{M}$ to $\mathcal{C}$. In fact, this means it is a *permutation* on $\{0,1\}^k$.

2. For an adversary who has access to an encryption oracle E_k, but not to the key k, every newly encrypted message yields a ciphertext that surprises her, or *appears to be random*.

[11] Pre-image resitance makes hash functions look a little like encryption schemes, only there is no decryption operation. In fact, many pre-images map to the same hash value, so there is not even a decryption.

The former means that block ciphers cannot be probabilistic (with more than non-negligible probability). If the same plaintext would map on a different ciphertext each time E_k is invoked with the same k, there would not be enough ciphertexts to map to all plaintexts.

The latter observation is not implicit in Definition 2.18, but in a suitably chosen security definition. However, it has a useful implication: A pseudorandom permutation can be modelled as an encryption oracle to a block cipher. (Since the key is hidden from the oracle client, if the block cipher is indistinguishably secure the client cannot make any predictions on the outputs.)

One-Time Pads and Stream Ciphers

Consider for a moment plaintexts of length 1. The operation $c = m \oplus k$ of xoring a plaintext bit with a key of length 1 is a perfect encryption operation in the sense that no trace of the plaintext is left in c. The adversary knows that if $k = 1$, then $m = 0$, and vice versa. And since she knows nothing about k, she knows nothing about m. By extension, in order to encrypt plaintext bit streams of arbitrary length, there is a class of encryption schemes that "zip" a stream of seemingly random bits into plaintext bits. This scheme has interesting implications and deserves its own name and definition.

Definition 2.19 (One-Time Pad (OTP))**.** *Fix a message length $l \in \mathbb{N}$. A* one-time pad (OTP) *is an encryption scheme (G, E, D) with $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^l$, where encryption and decryption are defined as*

$$E_k(m) = m \oplus k, \qquad D_k(c) = c \oplus k$$

($\oplus$ is bit-wise XOR.)

Since the unary operation $\cdot \oplus k$ is its own inverse for any k, decryption inverts encryption. The idea was originally developed for a non-binary alphabet by Gilbert Vernam in 1917 in order to protect communication over teletype machines. That one-time pads, if used properly, can be ultimately secure is stated in the following

Theorem 2.5. *If G chooses keys from $\mathcal{K}$ uniformly at random, OTP is single-message adaptive chosen ciphertext secure. This is even true if the adversarial algorithms A, A' are not computationally bounded.*

(Security against computationally unbound adversaries is called *information-theoretic security*, as opposed to *complexity theoretic security*. Being the stronger notion, security against unbound adversaries is hard to achieve

in practice. Achieving either one is usually considered good enough for all practical purposes.)

A proof to Theorem 2.5 can be found in (Sha49). Note the restriction to the single-message case, i.e., the case that each key is discarded after having been used once. Since key size equals message size, this makes the scheme quite impractical. Further, the insecurity in the multiple-message case is very acute: If a key k is used to encrypt two messages m_0, m_1, an eavesdropper can compute

$$c_0 \oplus c_1 = k \oplus m_0 \oplus k \oplus m_1 = m_0 \oplus m_1$$

In other words, she can compute the sum of the two plaintext, and thus decide for each bit independently whether it is the same in both messages or not!

On the other hand, the single-message security of this scheme is not only the theoretical optimum in the sense that no information at all is contained in the ciphertext (whether extractable or not) as long as the key remains secret. It is also the *only* scheme (up to trivial variations) that has such a strong security. In particular, every scheme in which i key bits are used to encrypt j plaintext bits for $j > i$ is strictly weaker than OTP in the single-message case.

So a trick to overcome the key management problem of OTPs and make them practical as an encryption method would be very valuable: We are looking for an algorithm that computes a large amount of key bits from a small truly random seed key that are *random enough to fool the adversary*, and use these pseudo-random key bits in the construction of a "fake" one-time pad. In other words, instead of using G to generate the full OTP key, we use it to generate a small number of key bits for a PRG, and use that PRG to feed the OTP. The resulting encryption scheme is called a *stream cipher*.

Definition 2.20 (Stream Cipher)**.** *Let $(G^*, \oplus, \oplus)$ be an OTP, let f_k be a pseudo-random permutation, and let R be a pseudo-random generator. The* stream cipher *(G, R, f_k) is the encryption scheme (G, E, D) with*

$$E_k(m) = v \,\|\, R_{f_k(v)} \oplus m$$
$$v \,\|\, D_k(c) = R_{f_k(v)} \oplus c$$

where v is chosen uniformly at random for each encryption. (v is called an initialization vector *and is published to all involved parties, including adversaries.)*

($\cdot \,\|\, \cdot$ is stream or sequence concatenation.) There are other constructions of stream ciphers such as (Gol01), but ours has the advantage that its security

can trivially be reduced to the security of OTP, R, and f_k: The PRG R allows for short fix-sized keys, while the uniform distribution of the initialization vector v guarantees multiple-message security.

Theorem 2.6. *Let (G, R, f_k) be a stream cipher. If R is secure in the sense of Definition 2.12, and f_k is secure in the sense of Definition 2.14 for some encryption operation $\cdot$, then (G, R, f_k) is secure in the sense of Definition 2.10.*

Proof. The probability that v is chosen twice with the same value is negligible, and so is the probability that R is invoked with the same key twice. Hence, the security of f_k ensures that knowledge of v does not help the adversary in learning anything about R's actual key. □

Since R's key $f_k(v)$ is generated anew for every run of the stream cipher, we also call it a *session key*. Sometimes we omit the session key generation issue for the sake of simplicity of notation and simply write (G, R) instead of (G, R, f_k).

Integrity and Authenticity

So far, we have encountered cryptography as a discipline to keep information secret from an adversary. However, there are many other security requirements that can be met by using cryptographic building blocks. Perhaps the second most prominent one after confidentiality is information integrity or information authenticity.

A message is *authentic* if the recipient holds a sender ID together with the proof that the sender with that ID actually sent it. It is *intact* if, no matter who sent it, nobody else modified the document after it was sent on its way.

There is a very simple argument why there cannot be a reasonable distinction between the two:[12] If a message is not authentic, integrity loses all meaning, because integrity now means that the message arrives exactly in the state in which it left the adversary rather than the legitimate sender. A message that is not intact is not authentic because it does not apply to the message as it was sent, but to whatever the adversary makes of it. In both cases, there is an adversary that can mislead the recipient about what the sender wrote or not.

[12]The author of this thesis has made this argument before (`http://www.etc-network.de/blog/fis/crypto_metaphors.html`), mostly reiterating a note by Phil Rogaway earlier (`http://www.cs.ucdavis.edu/~rogaway/papers/draft-rogaway-ipsec-comments-00.txt`).

Subtle and esoteric differences could be made: An authenticity-preserving scheme *may under additional assumptions* prevent an adversary from sending messages without the legitimate sender having sent one first. If the message is a physical object like a CD or a Prada purse, some may argue that the CD itself is still authentic, even if the contents is overwritten with data by an adversary. Or that the purse is still authentic, even if somebody patched it with a non-brand piece of cloth. Or one could simply argue that the department of homeland security would see the difference between an adversary originating the message and another one messing with it on the way.

On the other hand, there are very good reasons to not make this distinction: Designing secure systems is an intricate task, and there are enough traps if things are kept as simple as possible. Fighting complexity is essential to minimize risks. Leaving two redundant concepts dangling in a design process is a good recipe for disaster. Engineers are bound to start arguing about which of the two should be given up in favor of the other, or just plain forget one of them, and thus unknowingly sacrifice the other.

The following definitions span the cryptographic primitives for message integrity. We distinguish between the asymmetric signature schemes and the symmetric message authentication schemes. Apart from key handling (which due to the differing mathematical structures available has considerable implications for implementation performance and proofs of security), the two are identical.

Definition 2.21 (Signature Scheme)**.** *A* signature scheme (G, S, V) *is a tuple of three PPTMs for key generation, message signing, and message verification, respectively such that*

$$\begin{aligned} G &: 1^N \to \mathcal{K} \\ S &: \{d \mid (e,d) \in \mathcal{K}\} \times \mathcal{M} \to \mathcal{S} \\ V &: \{e \mid (e,d) \in \mathcal{K}\} \times \mathcal{M} \times \mathcal{S} \to \{ok, error\} \end{aligned}$$

and such that for all $(e,d) \in \mathcal{K}$ *and* $m \in \mathcal{M}$*,*

$$V_e(m, S_d(m)) = ok$$

Definition 2.22 (Message Authentication Scheme)**.** *A* message authentication scheme (G, S, V) *is a tuple of three PPTMs for key generation, message authentication, and message verification, respectively such that*

$$\begin{aligned} G &: 1^N \to \mathcal{K} \\ S &: \mathcal{K} \times \mathcal{M} \to \mathcal{S} \\ V &: \mathcal{K} \times \mathcal{M} \times \mathcal{S} \to \{ok, error\} \end{aligned}$$

and such that for all $k \in \mathcal{K}$ *and* $m \in \mathcal{M}$*,*

$$V_k(m, S_k(m)) = ok$$

$S_k(m)$ *is called a* message authentication token*.*

Note that these definitions only rule out false negatives, i.e. require that legitimate signatures or message authentication tokens are always accepted. False positives, i.e. cases in which illegitimate ones are accepted, are treated in the security definitions:

Definition 2.23 (Integrity (or Authenticity) in the adaptive chosen ciphertext model)**.** *A message authentication scheme* (G, S, V) *provides* integrity (or authenticity) *iff for any adversarial PPTM A and any message m,*

$$\Pr\left[\begin{array}{rcl} k & \leftarrow & G(1^N) \\ V_k(m, A^{S_k, V_k}(m)) & \leftarrow & ok \end{array}\right] \leq neg(N)$$

Security of integrity schemes in other models in the attack hierarchy and of signature schemes are defined analogously.

2.3.6 Homomorphic Cryptography

Recall that the application that interests us most is not encryption of messages for transport over an insecure channel, but protection of the contents of a database against the database server. In other words, there is a player in the game that has a role in the desired functionality of the system, but at the same time attempts to make the system deviate by decrypting some of the ciphertexts that she holds.

Throughout this thesis, we will call this player, the service provider, Chantal. The client, or the user, is called Murat. The challenge consists in choosing an encryption transformation that allows Chantal to produce the result of some computation f on the plaintext hidden in the ciphertext, but does not allow her to learn anything (or at least: not too much) about the plaintext. An encryption scheme that delivers to these requirements is called a *homomorphic encryption scheme*, or a *privacy homomorphism*. Although the latter term is still used in database research, the former has become more common in cryptography. We use the two synonymously, tending towards the cryptographic terminology.

Definition 2.24 (Homomorphic Encryption Scheme)**.** *Let* $\Phi : \mathcal{M} \mapsto \mathcal{M}$ *be a set of operations on the set* $\mathcal{M}$ *of plaintexts (plaintext operations), let* $\Psi : \mathcal{C} \mapsto \mathcal{C}$ *be a set of operations on the set* $\mathcal{C}$ *of ciphertexts (ciphertext*

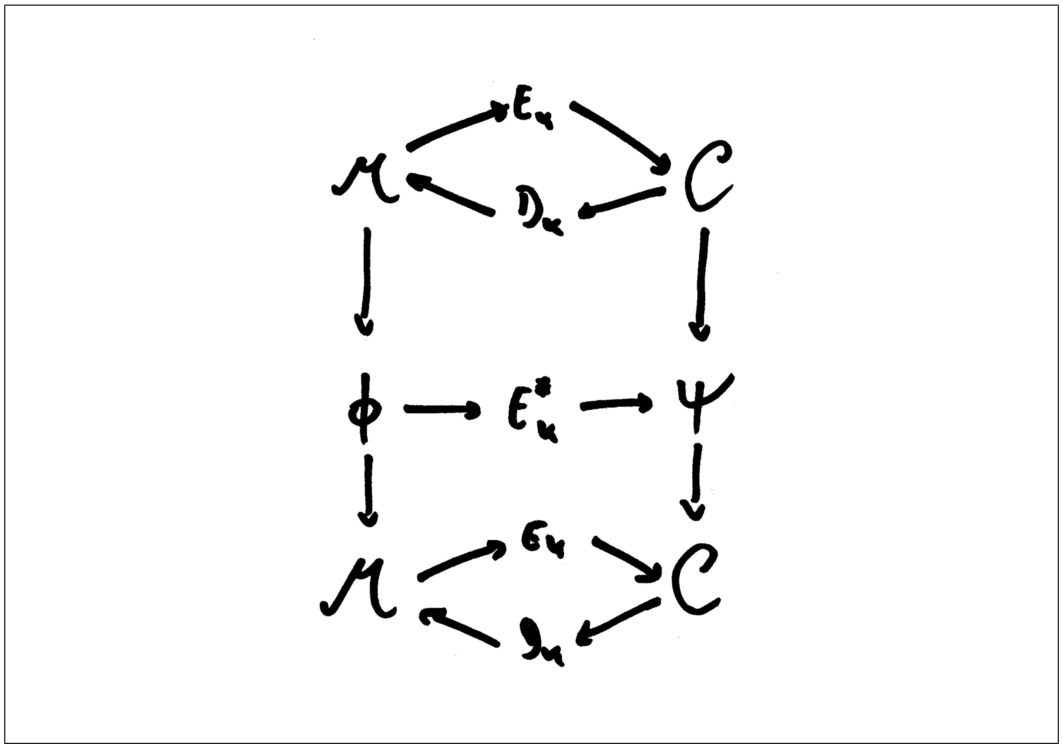


Figure 2.2: Homomorphic encryption.

operations), and let $E^* : \mathcal{K} \times \Phi \mapsto \Psi$ *be a mapping of plaintext operations to ciphertext operations.*

If (G, E, D) *is an encryption scheme, then* (G, E, E^*, D) *is a homomorphic encryption scheme (with respect to* Φ*) iff for any* $m \in \mathcal{M}, k \in \mathcal{K}, \varphi \in \Phi$*:*

$$E_k(\varphi(m)) = (E_k^*(\varphi))(E_k(m))$$

A homomorphism is a structure preserving mapping between two algebraic structures: Plaintexts $(\mathcal{M}, \Phi)$ and ciphertexts $(\mathcal{C}, \Psi)$. The structure is preserved with respect to a set Φ of operations on $\mathcal{M}$. This means that every computation that is carried out on a plaintext has an equivalent computation that can be carried out on the corresponding ciphertext. Figure 2.2 contains an illustration.

Contradictory a creature as it may seem (encryption should *hide* all plaintext structure in the ciphertext, not *maintain* it!), the first homomorphic encryption scheme has been proposed in the early years of civil cryptographic research in 1978 (RAD78),[13] shortly after the discovery of public-key cryptography and the development of the first public-key encryption scheme RSA (RSA77).

[13]The term *privacy homomorphism* has been coined in this same article.

Rivest, Adleman and Dertouzos discovered that if used properly, RSA is homomorphic with respect to multiplication of plaintexts: For every pair of integers $m_1, m_2 \leq (p-1)(q-1)$ (p, q are two large primes that have been generated by G):

$$E_k(m_1)E_k(m_2) \equiv E_k(m_1 m_2) \mod (pq)$$

This is covered nicely by Definition 2.24: If (G, E, D) is traditional RSA and $E^* : \{\cdot\} \mapsto \{\cdot\}$ is the identity mapping of residue-class ring multiplication on $\mathcal{M}$ to itself, then (G, E, E^*, D) is a homomorphic encryption scheme.

This allows Chantal to offer multiplication in residue class rings (or on integers with client-side overflow detection), with some hope for privacy. However, the overhead for Murat of doing the relatively complex RSA encryptions and decryptions locally is considerably bigger than the multiplication task that is outsourced. Also, note that the definition of homomorphic encryption still lacks a notion of security similar to those for non-homomorphic encryption discussed in Section 2.3.4. (See also Section 3.2 and Chapter 3). For now we only point out that the basic form of RSA that has the homomorphism property is insecure in most adversary models. For instance, since it is deterministic, Chantal can decide for two encrypted operands whether they are identical or not, breaking indistinguishability in the multiple-message case.

Are there any other homomorphic encryption schemes? A few articles have been published on this topic since 1978, many of which have been broken again (Section 3.1 gives a summary of the state of the art). Proposals to apply the notion of homomorphic encryption to relational algebra are quite recent (EFG06; FG03; BG02), and despite many open questions and tight theoretical limits, a few positive results are beginning to emerge, such as the encryption scheme developed in Section 4.3.

Usually, $\mathcal{M}$ and Φ are relatively simple, such as groups or rings, but this is not required by Definition 2.24. In fact, Φ does not even have to be finite. This observation is essential for capturing the idea of homomorphic encryption of relational algebra or its subsets, where Φ is an infinite enumeration of all query terms that the client is allowed to apply to any relation.

Definition 2.25 (Homomorphic encryption scheme for relation algebra)**.** *A homomorphic encryption scheme for relational algebra (or Hom$^\sigma$ for short) is a homomorphic encryption scheme (G, E, E^*, D) as in Definition 2.24, where $\mathcal{M}$ is the set of relations and Φ is (a subset of) relational algebra.*

Data Integrity in the Service Provider Model

The problem of maintaining confidentiality against a service provider by means of homomorphic encryption is hard, since the two goals of allowing the

service provider to process the data and keeping her from learning anything about it are in conflict.

Integrity does not conflict with the service provider model as severely. In order to establish that the data at the service provider's site has been compromised, the service provider does not need to transform the ciphertext as if he wants to compute functions on it. Traditional, non-homomorphic integrity-preserving methods can be applied without much change:

```
\begin{algorithm}
  \caption{Trivial Data Integrity}
  \Begin{
   download the entire database from Chantal\;
   compute message authentication token and store it locally\;
   repeat from time to time\;
   \If{authentication token has changed}
      {file a lawsuit against Chantal.}
  }
\end{algorithm}
```

Further, there are incremental algorithms and data structures such as hash trees (Mer80; Bau04; MRK03) that improve on the above.

Note that there is a difference between enforcing integrity and being able to detect whether there has been a security breach. When meeting confidentiality requirements, detecting a security breach usually is not enough. Once the breach has occurred, nothing can be done any more, so one wants to make very sure it simply does not happen.

Fortunately, as we have seen there are efficient schemes to do that. Enforcing integrity, however, is hard: If Chantal destroys some parts of the database she is holding, she cannot be forced to undestroy them. Hence, Murat is forced to be content with a proof of integrity. This reduces Chantal's incentive to destroy information in the first place, because she is aware of the legal consequences. Also, Murat may store a number of backups with different independent services providers, and if he realizes one of them has corrupted the database, he can restore it using those backups.

A Taxonomy of Security Requirements

The symmetric non-homomorphic security notions do not map very smoothly to homomorphic encryption. First of all, there is a new encryption operation E^* that is not treated by any of those. If the only operation carried out on ciphertexts is multiplication, E^* cannot hide any information (the attacker

knows that any encrypted operator is always multiplication), and this may not be a problem at all as long as the actual numbers that are processed remain secret. However, the more complex Φ gets, the bigger the danger grows that the operations to be run by Chantal may contain secrets of their own.

Assume Chantal knows everything about Murat's database scheme (remember that she is allowed all context knowledge up to the last bit she is required to guess in order to win the attack game), and she is recording Murat's query stream. Murat is first browsing the revenues of a particular branch of his company, and then starts mining into his employees database submitting queries about all the income distribution and sick leave statistics in that branch, she might consider, given the right adversarial motivation, passing that information on to the employees of that branch, and Murat might not be very happy about that. Hence, at least *sometimes*, E^* should be a good encryption algorithm, too. (As we will see in the following Chapters, in order to make the overall system secure in a meaningful sense of the word, we need more than that.)

So whereas in non-homomorphic encryption we have one stream of bits to protect, now we not only have an arbitrary set $\mathcal{M}$ of operands to protect, but there may be confidentiality issues with the operator set Φ, too. Integrity taken into account as well, this yields the following taxonomy of possible security requirements for homomorphic cryptography in the service provider model:

$$\begin{bmatrix} & \text{data} & \text{code} \\ \text{integrity} & (1) & (2) \\ \text{confidentiality} & (3) & (4) \end{bmatrix}$$

In the last section we have made the point that (1) is not as challenging as confidentiality requirements, so we give the latter a lot more room. This also applies to (2), only the problem is more damaging to performance here: If Murat does not trust Chantal to return complete and sound results, he has three options:

1. Do everything locally, and compare the results to those returned by Chantal (not very attractive).

2. Do *something* locally every now and then and sample Chantal's reliability. If an error occurs, file a lawsuit.

3. Deploy a number of redundant service providers and compare their results.

(3), (4), and the interdependencies between the two are the main subject of this part of the thesis.

Chapter 3

Related Work

In the last chapter, we have established the basics of database theory and cryptography to express the problem. The following is a survey of related work relevant for our results.

3.1 Homomorphic Encryption Schemes

A number of homomorphic encryption schemes have been proposed, for relational algebra as well as for other structures. We will now sketch their most important ideas and shortcomings.

3.1.1 ATE: Homomorphic Encryption in Databases

In 2002, Hacıgümüş et al. have proposed an encryption scheme for relational algebra (HILM02), although neither a notion of homomorphic encryption nor the term "relational algebra" are mentioned (the scheme refers directly to SQL). The paper is important for three reasons: First, it has been quoted extensively, mostly due to the practical relevance of the claimed results and the high performance and completeness of coverage of relational operators. Second, although the authors where not aware of it, it is the first that we know of to attack the problem of secure database outsourcing in a way that *looks like* a homomorphic encryption scheme. And finally, its flaws make it an almost perfect counter-example by demonstrating how *not* to solve the problem at hand.

We call this encryption scheme *aggregate-then-encrypt (ATE)*. ATE proceeds in rows and in two phases: First, it securely but non-homomorphically encrypts a row with a symmetric encryption scheme $(G^\circ, E^\circ, D^\circ)$ called *hard encryption* in the following. Then, it adds additional data to each row that

allows for computation of relational algebra terms on the resulting encrypted table. We call this additional information *soft encrypted*. Hard encryption hides all the necessary information, and this is where new means of encryption need to be developed that reveal *just enough* information, but *never too much*. Since hard encryption $(G^\circ, E^\circ, D^\circ)$ does not need to be homomorphic and can be implemented using traditional cryptography,[1] all security relies on soft encryption. Hence, the latter deserves further elaboration.

To recall the notation introduced in Section 2.2, the plaintext set $\mathcal{M}$ consists of relations R that in turn consist of rows r_i, which are tuples of attributes $(a_0, \dots) \in (A_0 \times \dots)$. On the other hand, ciphertext in ATE consists of relations $R^\mathcal{C} \in \mathcal{C}$ that in turn consist of one ciphertext row $r_i^\mathcal{C}$ for each plaintext row $r_i \in R$. Ciphertext rows contain the strongly encrypted plaintext row $E_k^\circ(r_i)$, plus a number of soft encrypted attributes A_j.

For a given attribute A_j, we are now going to describe how a corresponding soft encryption function $h_j : A_j \to A_j^\mathcal{C}$ is defined. Note that h_j does not depend on any key known only to Murat, so Chantal can compute h_j on any given plaintext value by herself. For a relation $R \in \mathcal{M}$, an encrypted row $r^\mathcal{C} \in R^\mathcal{C} \in \mathcal{C}$ will look as follows:

$$r^\mathcal{C} = (E_k^\circ(r), h_{j_1}(a_{j_1}), h_{j_2}(a_{j_2}), \dots)$$

First, h_j splits the attribute domains A_j into *partitions* and maps each attribute value a_j to its partition.

Definition 3.1 (Partitioning)**.** *A* partitioning $Q_A : A \to \mathbb{N}$ *of any finite set A maps each element of A to a natural number (its* partition*) such that $a, b \in A$ are in the same partition iff $Q_A(a) = Q_A(b)$. A partitioning is* ordered *iff $a_j \leq a_k$ whenever $Q_A(a_i) \leq Q_A(a_k)$. Q_A provides order-preserving, but inaccurate identifiers for elements of A (the identifier $Q_A(a)$ of $a \in A$ does not reveal a in full, but its interval in A).*

(A partitioning of A establishes an equivalence relation on A.) Soft encryption $h_i(a)$ of an attribute value $a \in A_i$ consists of (at most) two operations:

1. An ordered partitioning $Q_{A_i} : A_i \to \mathbb{N}$. (If there are no user preferences from the semantic point of view, use the lexicographical order over the bit representations of the attribute values.)

[1] Of course, this does not imply that the implementation should not be carried out carefully. For instance, if one is as unaware as the authors of the fact that RSA is not a block cipher or of the difference between randomness and pseudo-randomness, one risks to make other mistakes as well that open vulnerabilities even in parts of the system that can easily be secured in principle.

2. A hash function H that maps partitions to hash values such that Chantal may be aware of the hash of a partition, and still cannot (in any straightforward way) compute the partition itself.

Two types of soft encryption are described by the authors (and we will discuss further variants later on): $h^{\mathbf{ord}} = Q_{A_i}$ is order-preserving and $h^{\mathbf{random}} = H \circ Q_{A_i}$ is pseudo-random.

Which one is used depends on the privacy requirements for the attribute in question. Whenever $h^{\mathbf{ord}}$ is used, it is obvious to the server which of a, a' is bigger just from looking at $h^{\mathbf{ord}}(a), h^{\mathbf{ord}}(a')$. If $h^{\mathbf{random}}$ is used, the server can only identify rows that fall into the same partition, but not a useful order on the partitions. Wherever it is obvious or irrelevant which one is meant, we simply write h.

In the following, we write h for soft encryption of arbitrary attributes, attribute names, conditions on attributes, terms of relational algebra, and complete relations. This puts aside the question of which partitionings should be used and whether h should be random or ordered on any specific attribute, and makes the crucial concepts more transparent.

The definition of translation functions that map a query on a database $\bigcup R_i$ to a query on the encrypted counterpart $\bigcup R_i^{\mathcal{C}}$ is now rather straightforward. Two transformations are defined, one for conditions and one for the relational query operators. In order to access the database, the client first transforms the query into a less precise query suitable for $\bigcup R_i^{\mathcal{C}}$ rather than $\bigcup R_i$. The server processes this transformed query on the encrypted relation and returns the relation consisting of the w encrypted rows $(E_k^{\circ}(r_{i_1}), \dots)$ that approximates the results to the query (projecting away the soft-encrypted key attributes). Finally, the client decrypts the rows and runs the plaintext query on $(r_{i_1}, \dots)$ once more, which is hopefully much smaller than the original relation.

Figure 3.1 shows the transformation of conditions. If an attribute is required to have a given value, the client soft-encrypts that value and makes the server compare the result with the soft-encrypted values in the encrypted relation (3.1). To refer to all values *smaller* than a given value v, a set of partitions that contain values smaller than v is computed, and thus all rows on the server side with soft-encrypted values in that set will match (3.2). (Note that if we were using $h^{\mathbf{ord}}$ instead of $h^{\mathbf{random}}$, inequality relations could be used on soft encrypted attribute values just as on the plaintext, resulting in better performance and lower security). To probe two attribute values for equality, the client enumerates all pairs of partitions that contain pairs of equal plaintext values (3.3). The rules for comparison operators are defined analogously (3.4). The logical operators $\wedge$ and $\vee$ are not affected by soft

encryption. Negation is not considered in the basic scheme.

$$
\begin{aligned}
h(A_i = v) &= h(A_i) = h_i(v) && (3.1)\\
h(A_i < v) &= h(A_i) \in \{h_i(v') | v' < v\} && (3.2)\\
h(A_i = A_j) &= \bigvee_{v \in A_i, v' \in A_j, v = v'} (h(A_i) = h_i(v) \wedge h(A_j) = h_j(v')) && (3.3)\\
h(A_i < A_j) &= \bigvee_{v \in A_i, v' \in A_j, v < v'} (h(A_i) = h_i(v) \wedge h(A_j) = h_j(v')) && (3.4)\\
h(C \wedge C') &= h(C) \wedge h(C') && (3.5)\\
h(C \vee C') &= h(C) \vee h(C') && (3.6)\\
h(\neg C) &= \text{n/a} && (3.7)
\end{aligned}
$$

Figure 3.1: ATE: Transformation of conditions ($h = h^{\mathbf{r}\text{random}}$).

Figure 3.2 formalizes the transformation of relational algebra terms. Selection is quite straightforward (3.8). Relation and condition are transformed as described in Figure 3.1, the server computes a conservative approximation of the outcome, and the client runs the query on the decrypted outcome in order to obtain the exact result. Nothing can be done to help Murat when performing a projection (3.9): The hard encrypted tuples are out of reach for Chantal, and the full relation needs to be downloaded and projection applied locally. An unconditional join would be the same on $h(R), h(T)$ as on R, T (simply compute all possible pairs of rows from the two relations). If a condition is given, the server can filter out those row pairs that do not match the ciphertext condition (3.10). Groups with respect to attribute set I can be formed by the server on the ciphertext attributes $h(I) = \{h(A_i) | A_i \in h(I)\}$, so that the client only needs to group each partition locally. (3.11, 3.12). Sorting only makes sense for ordered soft encryption (3.13). If the soft encrypted attributes are still in order, the sort operation in the post-processing step may become considerably faster. However, when $h^{\mathbf{r}\text{random}}$ in place of $h^{\mathbf{o}\text{rd}}$ is used, sorting on the server side has no positive effect.

One year after (HILM02), another research group has published a follow-up article that improves on its results (DdVJ+03). A method is proposed that is based on ATE, but supports a metrics for quantifying the information exposed to the server and deals better with interval queries. Our analysis covers variants of ATE like this one.

We conclude this section with an example of using ATE, a discussion of its performance and a brief assessment of its security. In Section 4.1, we will

$$
\begin{aligned}
h(\sigma_q R) &= \sigma_q(h^{-1}(\sigma_{h(q)}(h(R)))) && (3.8)\\
h(\pi_{\mathcal{I}} R) &= \pi_{\mathcal{I}}(h^{-1}(h(R))) && (3.9)\\
h(R \bowtie_q T) &= \sigma_q(h^{-1}(h(R) \bowtie_{h(q)} h(T)) && (3.10)\\
h(\gamma_I R) &= \gamma_I(h^{-1}(\gamma_{h(I)}(h(R)))) && (3.11)\\
h(\gamma_{I,F} R) &= \gamma_{I,F}(h^{-1}(\gamma_{h(I),h(F)}(h(R)))) && (3.12)\\
h(\tau_I R) &= \tau_I(h^{-1}(\tau_{h(I)}(h(R)))) && (3.13)
\end{aligned}
$$

Figure 3.2: ATE: Transformation of relational algebra terms.

give an extensive discussion of its (in-)security.

A Simple Example

Consider a relation of employees with three attributes `id`, `name`, and `salary` (similar to the one in (HILM02)):

	`id`	`name`	`salary`
r_1	323	Tom	91,130
r_2	860	Mary	63,200
r_3	320	John	50,000

Presume that `id` and `salary` are the only relevant attributes, i.e., the server does not even know about the existence of the names, and cannot compute any query conditions on them. (Note that the client could still run queries on the names locally after decryption.) The partitionings are the following:

$$
Q_{\texttt{id}}(a) = \begin{cases} 0 & \text{if } i < 100 \\ 1 & \text{if } 100 \leq a < 200 \\ \ldots & \\ 9 & \text{if } 900 \leq a \leq 999 \end{cases}
$$

$$
Q_{\texttt{salary}}(a) = \begin{cases} 0 & \text{if } a < 20,000 \\ 1 & \text{if } 20,000 \leq a < 40,000 \\ 2 & \text{if } 40,000 \leq a < 60,000 \\ 3 & \text{if } 60,000 \leq a < 80,000 \\ 4 & \text{if } 80,000 \leq a \end{cases}
$$

Further, chose id soft encryption $h_{\mathtt{id}} = Q_{\mathtt{id}}$ (in the hope that an id being in a specific partition does not tell Chantal anything interesting) and salary soft-encryption $h_{\mathtt{salary}} = H \circ Q_{\mathtt{salary}}$, and assume

$$H = \{0 \to 1, 1 \to 4, 2 \to 0, 3 \to 2, 4 \to 3\} \tag{3.14}$$

(Chantal may very well be curious about an approximation to the employees' incomes). Thus, r_1 is becomes:

$$\begin{aligned} h(r_1) &= (E_k^\circ(r_1), h_{\mathtt{id}}(323), h_{\mathtt{salary}}(91{,}130)) \\ &= (E_k^\circ(r_1), 3, 3) \end{aligned}$$

The other rows are encrypted analogously. Having processed R this way and passed $h(R)$ to Chantal, Murat can start having Chantal run queries for him. Consider the following plaintext query:

$$\sigma_{(\mathtt{salary} \geq 55{,}000 \wedge \mathtt{salary} \leq 65{,}000)} R$$

With the selected soft-encryption methods, this translates to:

$$\sigma_{\begin{array}{l}\mathtt{s}\text{alary} \in \{h_{\mathtt{salary}}(40,000), h_{\mathtt{salary}}(60,000), h_{\mathtt{salary}}(80,000)\} \wedge \\ \mathtt{salary} \in \{h_{\mathtt{salary}}(0), h_{\mathtt{salary}}(20,000), h_{\mathtt{salary}}(40,000), h_{\mathtt{salary}}(60,000)\}\end{array}} h(R)$$

Of course, there is an obvious optimization:

$$\sigma_{\mathtt{s}\text{alary} \in \{h_{\mathtt{salary}}(40,000), h_{\mathtt{salary}}(60,000)\}} h(R)$$

This can be processed by Chantal and yields $\{h(r_2), h(r_3)\}$, and by decryption Murat obtains $\{r_2, r_3\}$. Now the preliminary result contains the two employees Mary (which is good) and John (which is not so good, because John does not match the non-aggregated query). Therefore, Murat needs to re-run the original query on the smaller result relation to produce the correct and accurate result $\{r_2\}$.

Performance

ATE has been tested using the TPC-H benchmark[2] on databases of size 10MB and 100MB. Although the slowdown appears reasonable at the first glance, there are a number of caveats.

If an encryption scheme does not scale to hundreds of Gigabytes or more, its use in practice for outsourcing database services is quite limited — every

[2] `http://www.tpc.org`

cell phone today is capable of hosting Gigabytes of data. As long as we are attacking the problem with the service provider model in mind, the provided benchmarks are meaningless.

As long as we do not talk about security, there is no trade-off between performance and security, but only performance. But the authors of ATE make no claims about the security of their scheme. In particular, there is no model of the adversary, and no notion of what it means for the scheme to be broken. So to be perfectly fair, the benchmark should be compared to not using any security precautions at all.

Finally, some of the query transformation cases, e.g. the case $A_i = A_j$, can get quite bulky. Since all the intersecting partition pairs need to be enumerated, the size of the encrypted query is

$$O(|Q_{A_i}| \cdot |Q_{A_j}|)$$

for given partitionings Q_{A_i} and Q_{A_j}, i.e. quadratic in the number of partitions.[3] This not only has disastrous effects on the performance, but it also allows Chantal to compute more fine-grained partitionings on the ciphertext. So in contrast to the performance / privacy tradeoff observed everywhere else, increasing the number of partitions not only reduces the effectiveness of the encryption, but also has a negative impact on performance in the presence of conditions of the form $A_i = A_j$.

Security

So far we have only explained how the scheme serves its purpose as long as Chantal cooperates. As already mentioned, the authors say very little about the security. One of the contributions of this thesis is to examine ATE security in great depth and point out several flaws together with repair strategies, as well as some that cannot be fixed.

We present our results in Section 4.1. For now we merely point out that ATE is trivially secure as long as no soft encrypted attribute contains any confidential information. Without loss of generality, fix a database schema

$$\mathcal{M} \times \mathcal{M}' = (A_0 \times \cdots \times A_m \times A'_0 \times \cdots \times A'_{m'})$$

where attributes A'_i are soft encrypted, and attributes A_i are not. We write $R \parallel R' \in \mathcal{M} \times \mathcal{M}$ for the row-wise concatenation of $R \in \mathcal{M}, R' \in \mathcal{M}'$. It holds that $|R| = |R'| = |(R \parallel R')|$. As above, we write h for encryption of queries. Further, we write h_k for encryption of complete relations, including the hard encryption of the entire tuples for which we require a secret key k.

[3] $h^{\mathbf{ord}}$, which we suspect has been used for the benchmarks, does not have this problem.

Theorem 3.1 (Security of ATE)**.** *If* $(G^\circ, E^\circ, D^\circ)$ *is indistinguishable, then ATE is secure in the following sense: For any adversarial PPTM* M*, any two relations* $R_0, R_1 \in \mathcal{M}$*, any relation* $R' \in \mathcal{M}'$*, any function* $f : \mathcal{M} \to \{0,1\}$*, and any sequence of terms of relational algebra* $\bar{t}$*,*

$$\begin{aligned} &| \Pr[M(R_0, R_1, h_k(R_0 \parallel R'), \bar{h}(\bar{t})) = f(R_0)] \\ &- \Pr[M(R_0, R_1, h_k(R_1 \parallel R'), \bar{h}(\bar{t})) = f(R_1)] | \leq neg(n) \end{aligned}$$

where $k = G(1^N)$*.*

Proof. The soft encrypted attributes in the output of h_k are independent of R_i. Furthermore, the output of h only depends on soft encrypted attributes in its input by construction. Therefore, the query results are the same in both probability terms:

$$\Pr[\bar{h}(\bar{t})(h_k(R_0 \parallel R')) = \bar{h}(\bar{t})(h_k(R_1 \parallel R'))] = 1$$

Therefore, if M can distinguish between

$$(R_0, R_1, h_k(R_0 \parallel R'), \bar{h}(\bar{t}))$$

and

$$(R_0, R_1, h_k(R_1 \parallel R'), \bar{h}(\bar{t}))$$

it in fact distinguishes between $E_k^\circ(R_0)$ and $E_k^\circ(R_1)$. However, $(G^\circ, E^\circ, D^\circ)$ is indistinguishable. □

Unfortunately, the odd separation of soft encrypted from hard encrypted attributes makes this result quite impractical. If we want to be sure that it is safe to use ATE for a particular database schema and a particular relation, we first need to make sure that hard encrypted attributes do not depend on soft encrypted ones, however noisy. If that was the case, Chantal could simply compute the distinguishing bits of the hard encrypted data from the soft encrypted data. We will return to the security of ATE in Section 4.1.

3.1.2 Field Arithmetics

We have briefly sketched the concept of public-key encryption in Section 2.3.5. Let us now have a brief look at RSA (RSA77), the first and most famous public-key encryption schemes. As already discussed, keys consist of a public part e and a private part d, such that

$$\begin{aligned} E &: \{e | (e,d) \in \mathcal{K}\} \times \mathcal{M} \mapsto \mathcal{C} \\ D &: \mathcal{K} \times \mathcal{C} \mapsto \mathcal{M} \end{aligned}$$

RSA is based on $\mathbb{Z}_n$, the residue class field[4] of order $n = pq$ for two large prime numbers p, q. Intuitively, all operations in $\mathbb{Z}_n$ are the same as in $\mathbb{Z}$, except that numbers that yield the same residue when divided by n fall into an equivalence class and represent the same element of $\mathbb{Z}_n$. More simply put, in $\mathbb{Z}_n$, everything is implicitly computed $\mod n$. $\mathbb{Z}_n$ has a few interesting properties for our purposes. For $a, b \in \mathbb{Z}_n$, it has proven easy to compute products $c = ab$. But *integer factorization*, the problem of computing the prime factors a, b from c without knowing either, is generally assumed to be infeasible. Further, since the multiplicative subgroup of $\mathbb{Z}_n$ has order $(p-1)(q-1)$, we obtain $a^{(p-1)(q-1)} = a$.

Now if the public key is a small constant, say, $e = 3$, then if we can compute a secret key d from e, p, q for which de is a multiple of $(p-1)(q-1)$, write:

$$(p-1)(q-1)|ed$$

then we have our encryption and decryption:

$$E_{(n,e)}(m) := m^e$$
$$D_{(n,d)}(c) := c^d = m^{ed} = m^{x(p-1)(q-1)} = m$$

d can be computed from e, p, q using the extended Euclid algorithm for finding the greatest common divisor (see Algorithms 1 and 2). Note that since the gcd is computed in the multiplicative subgroup, we have

$$\gcd(e, (p-1)(q-1)) = 1$$
$$v(p-1)(q-1) = 0 \quad \text{for any } v \in \mathbb{Z}$$

Therefore, $eu = 1$ for the u that is returned by the extended Euclid algorithm, and we can set $d := u$.

By the *RSA assumption* that has been widely accepted after almost 30 years of academic scrutiny, there is no efficient algorithm to compute d from n, e alone, without knowledge of q, p. Therefore, as long as p, q and the private key d are kept secret from an adversary, she will be unable to apply the decryption function to some ciphertext $c \in \mathbb{Z}_n$, even if she knows the public key.

In this simple form, RSA is homomorphic. Decrypting the product of two ciphertexts yields the product of the two corresponding plaintexts:

$$D(E(m_1)E(m_2)) = m_1 m_2$$

[4]In the following, we will need a few basics in algebra and number theory. One of the many suitable text books to read up on these would be (Kob94). The impatient reader can skim through the rest of this section without missing anything that is essential later.

Algorithm 1: xEuclid: A recursive definition of the extended Euclid algorithm for computing the gcd of two integers. (For an iterative construction and further discussion see (CLRS01).)

Input: $a, b \in \mathbb{Z}_n$ such that $a \leq b$
Output: $x, u, v \in \mathbb{Z}_n$ such that
x is common divisor: $x|a \wedge x|b$
x is maximal: $\forall x' > x : x' \not| a \vee x' \not| b$
$au + bv = 1$

$d := a/b$ (divisor)
$m := a\%b$ (remainder)
if $m = 0$ **then**
  return $(a, 1, 0)$
else
  $(x, u', v') :=$ xEuclid(m, a)
  return $(x, v' - du', u')$

This has been discovered shortly after the development of RSA by the authors (RAD78), who also have coined the term "privacy homomorphism". To re-iterate, RSA being homomorphic allows Murat to deliver arithmetic jobs to Chantal, and Chantal to perform the jobs on her hardware, delivering the output to Murat without learning anything about the numbers involved.

A number of the cryptosystems in (RAD78) have been broken subsequently (BY87) (although not the basic one outlined above). Much later, Domingo-Ferrer and others have attempted to extend the results to full field arithmetics, including addition, subtraction and division (DF96; DFHJ98; DF02). We have applied these ideas in a preliminary feasibility study for application service providing (BF03), and others have tried to use them in the protection of software agents on untrusted hosts (LAFH04). However, attacks against these extended schemes (Wag03; CKN06) have emerged since then.

The status quo is not very encouraging: RSA stands undefeated, but only for weak security definitions. All solutions for other operations have been broken, and simple indistinguishability for addition has been shown to be impossible (ALN87). Plaintext numbers are chosen from a finite algebraic structure, requiring expensive overflow checks. Finally, encryption and decryption alone are computationally more expensive than the alternatively of performing the field arithmetics on the plaintext locally, which rules out savings in client hardware as a possible motivation for using it in applications.

Algorithm 2: RSA key generation. (d is directly returned by the extended Euclid algorithm, since $(p-1)(q-1) = 1 \mod n$.)

Input: key size k in bits
Output: public key n, e and private key n, d

while *! (p, q are both prime)* **do**
  Choose $p, q \in [0, 2^k - 1]$ uniformly at random
$n := pq$
$e := 5$
$d := u$ (where $(x, u, v) := \gcd(e, (p-1)(q-1))$)

It is nevertheless instructive to consider Domingo-Ferrer's novel and creative (if ultimately invalid) way of looking at security of homomorphic encryption. Therefore, we conclude this section with a discussion of his scheme for full-fledged field arithmetics, called $(G^{\mathrm{df}}, E^{\mathrm{df}}, D^{\mathrm{df}})$ in the following (DF02). Later in Section 3.2.1 we will return to his security definitions and proofs.

As in many other public-key encryption schemes, the plaintexts of $(G^{\mathrm{df}}, E^{\mathrm{df}}, D^{\mathrm{df}})$ are chosen from a finite ring $\mathbb{Z}_n$ for some integer n.[5] The ciphertexts are chosen from a much larger ring of polynomials over $\mathbb{Z}_N$ of degree d, $\mathbb{Z}_N^d$. N is an integer with many small divisors, one of which must be n. E^{df} attempts to conceal a given plaintext from Chantal by mapping each plaintext at random to one of many different ciphertexts.

To be more specific, the parameters of the encryption scheme are d, N, n, r such that:

1. $d \in \mathbb{Z}$, e.g. $d = 3$
2. $N \in \mathbb{Z}$, e.g. $|N| = 512$
3. $n > 1$ such that $n|N$
4. $r \in \mathbb{Z}_N$ such that $r^{-1} \in \mathbb{Z}_N$

In order to encryption a plaintext $m \in \mathbb{Z}_n$, we create a polynomial of degree d at random that we will show has the desired properties:

$$E^{\mathrm{df}}_{r,n}(m) = (m_1 r, m_2 r^2, \ldots, m_d r^d)$$

[5]Note however that the structure is more similar to private-key encryption schemes: In contrast to public-key encryption schemes, in $(G^{\mathrm{df}}, E^{\mathrm{df}}, D^{\mathrm{df}})$ the public parameters available to the adversary are not meant for encrypting plaintexts.

where $m_1, \ldots, m_d$ are chosen at random such that $\sum_{i=1..d} m_i = m$. Observe that the ciphertext is not a polynomial function, but a sequence of terms of a polynomial function that have been evaluated with argument r. And this is what $\mathbb{Z}_N^d$ really is. We will see why we need to think about polynomials when we look at ciphertext arithmetics in a second.

But first, decryption:

$$D_{r,n}^{\mathrm{df}}(c_1, c_2, \ldots, c_d) = \sum_{i=1..d} c_i r^{-i} = \sum_{i=1..d} m_i (r^{i-i}) = m$$

Since we know the secret parameter r, we can compute a number in $\mathbb{Z}_N$ that is congruent to m, and can thus easily coerced into $\mathbb{Z}_n$.

Addition and subtraction of two ciphertexts is carried out componentwise:

$$(c_1, \ldots, c_d) + (c'_1, \ldots, c'_d) = (c_1 + c'_1, \ldots, c_d + c'_d)$$

Multiplication is performed according to the laws of polynomial arithmetics:

$$(c_1, \ldots, c_d)(c'_1, \ldots, c'_d) = (0, c_1 c'_1, c_1 c'_2 + c_2 c'_1, \ldots, c_{d-1} c'_d + c_d c'_{d-1}, c_d c'_d)$$

For these operations to work, the elements in the sequences need to be ordered (or labelled) according to their degree. Empty places need to be padded with zeros. During generation of a product, all elements of one degree are collapsed in a sum in the corresponding place.

The laws of arithmetics valid in $\mathbb{Z}_n$ and $\mathbb{Z}_N^d$ make sure that the homomorphism property is maintained throughout all operations, i.e. that:

$$D^{\mathrm{df}}(E^{\mathrm{df}}(m) + E^{\mathrm{df}}(m')) = m + m'$$

and likewise for subtraction and multiplication. Since $\mathbb{Z}_N$ is a ring, and not a field, the ciphertext space has elements that are not contained in the multiplicative subgroup of $\mathbb{Z}_N$, i.e. do not have an inverse. This does not render division impossible, but complicates it a little. We omit the answers to these complications for the sake of compactness.

Again, there is an attack against $(G^{\mathrm{df}}, E^{\mathrm{df}}, D^{\mathrm{df}})$ that both breaks the security in the definition claimed in the paper as well as intuitively weakens the system to practical uselessness. We will discuss this further in Section 3.2.1.

3.1.3 Full-Text Search

Assume we are running an online e-mail service on which users can store their archives in encrypted form. The interface allows for downloading encrypted

e-mails into the browser where they can be decrypted (e.g., by a JavaScript function embedded into the page) and displayed.

This setting seems to conflict with full-text search on the archive: Either the client needs to download all e-mails and search them locally, or the server needs to be provided with the cryptographic key to look into the encrypted archive. Or, of course, we can use a secure encryption scheme homomorphic with respect to full-text search, if one exists:

Definition 3.2 (Homomorphic Encryption for Full-Text Search)**.** *Let l be the global* word length*, and let $W = \{\{0,1\}^l\}$ be the set of all* words *of that length. A* homomorphic encryption scheme for full-text search *is a homomorphic encryption scheme (G, E, E^*, D) as defined in Definition 2.24, where $\mathcal{M} = \{(W^*)^*\}$ is the set of all sets* (archives) *of word sequences* (documents)*, and $\Phi = \{\gamma_w | w \in W\}$ is the set of full-text search queries. γ_w yields all documents containing w: If $\gamma_w \in \Phi$ and $m = \{m_0, m_1, \dots\} \in \mathcal{M}$, then $\gamma_w(m) = \{m_i \in m | m_i = (w_0, w_1, \dots) \wedge \exists j. w_j = w\}$.*

For example:

$$\text{if} \quad U = \left\{ \begin{array}{c} \{(\text{ab}, \text{ac}, \text{bc})\} \\ \{(\text{a}, \text{cba}, \text{c})\} \\ \{(\text{ac}, \text{ca}, \text{aa})\} \end{array} \right\} \subseteq W, \quad \text{then} \quad \gamma_{\text{ac}}(U) = \left\{ \begin{array}{c} \{(\text{ab}, \text{ac}, \text{bc})\} \\ \{(\text{ac}, \text{ca}, \text{aa})\} \end{array} \right\}$$

The globally constant word length will prove useful in the construction of schemes that satisfy this definition. Luckily, it does not restrict the application range, as we can use *padding* and *chopping* to allow for shorter and longer words, respectively. When a word is too long, simply chop it into l-sized pieces and count them as separate words. Then, search for each word separately, and intersect the results.[6]

(SWP00) introduces a scheme for homomorphic encryption for full-text search. In the following, we will refer to this scheme as $\text{Fts}^{[\cdot]}$. In order to do introduce it, we need to make use of a few building blocks introduced in Section 2.3.5. In particular, we are going to talk about stream ciphers, block ciphers, and pseudo-random functions (or hash functions). At the core, the scheme consists of a stream cipher, only Chantal, given a properly encrypted query, can decide which locations in which documents match the query. The search word is encrypted not only in the searched text, but also in the query.

For this, the pseudo-random stream needs to have some additional structure that can be obtained by interleaving an ordinary pseudo-random bit stream with *suffixes* of bits that depend on (among other things) a fixed

[6]Performance needs to be taken into account, but many schemes have straightforward extensions in which the overhead in computation and communication is a small constant.

number of preceding bits. We start by listing a number of requirements for the function generating these suffixes. Then, following the structure of (SWP00), we introduce a simplified version of the scheme and improve on it step by step. For illustration of the full scheme, see Figure 3.3.

Let l be the word length in bits, and let $m < l$. (For instance, $l = 512, m = 96$.) The stream cipher uses a pseudo-random bit stream as usual. But for each word w_i, it generates only $l - m$ pseudo-random bits, $s_i \in \{0,1\}^{l-m}$, and concatenates them with m bits generated by a pseudo-random function F, and xors the resulting concatenation $s_i \,\|\, F_{...}(s_i)$ onto w_i. But how is F keyed? Before we elaborate on this, observe that F needs to have the following properties:

1. If the pseudo-random generator R is secure, then R^* must be a secure pseudo-random generator, where

$$R^*(k) = s_0 \,\|\, F_{...}(s_0) \,\|\, s_1 \,\|\, F_{...}(s_1) \,\|\, \ldots$$

2. Murat must be able to efficiently compute F during encryption.

3. Loosely speaking, it must be infeasible for Chantal to compute F on her own.

4. Search needs to work: Given an encrypted query for word w_i and a ciphertext word c_i, Chantal must be able to decide whether $c_i = w_i \oplus (s_i \,\|\, F_{...}(s_i))$ for some s_i or not. (If and only if it is, an occurrence of the search word has been found.)

Requirements (2) and (3) imply that F needs to be keyed with something only known to Murat. The obvious candidate is w_i, but then Chantal could find all occurrences of words of her choice in the encrypted text base, even before receiving any queries from Murat. For each c_i, she could simply compute s_i as the prefix of $c_i \oplus w_i$, compute $F_{w_i}(s_i)$, and see whether remaining bits in $c_i \oplus w_i$ match. If Chantal is an e-mail service provider, by using an ordinary English dictionary for options of w_i she quickly would have decrypted the entire text base up to typing errors!

Therefore, instead of using w_i to key F, we introduce a second pseudo-random function f that maps w_i on $f_{k'}(w_i)$. So in addition to the key $k \in G(1^n)$ for the pseudo-random generator R, Murat generates an additional key $k' \in G(1^n)$ that keys the suffix key generator f. This way, the suffix for encryption of not only depends on w_i, but also on k'. Since Chantal does not know k', she cannot compute $f_{k'}(w_i)$ herself, and the search query for word

w_i must now contain both w_i and $f_{k'}(w_i)$ for Chantal to be able to process it. This means that Chantal can only run queries crafted by Murat.

But Chantal can still learn all those words that Murat lets her search for, plus their positions in the plaintext! Since the scheme should satisfy a strong security definition, this is clearly a big disadvantage, even if Murat considers the non-searchable parts of his text base secret and cannot think of a reason why Chantal should not learn occurrence of the search terms.

To improve the secrecy of search words, we chose a deterministic encryption scheme $(G, E^{\text{pre}}, D^{\text{pre}})$ and encrypt each word w_i before feeding it to (G, R^*). This new layer of encryption requires a third secret key, $k'' \in G(1^n)$. $(G, E^{\text{pre}}, D^{\text{pre}})$ has to be deterministic: If Murat encrypts a word that he wants to search for, he must obtain the same encryption as when encrypting words in the text base. See discussion below.

Queries of this new form, $(E^{\text{pre}}_{k''}(w_i), f_{k'}(w_i))$, do not reveal the words that are searched to Chantal any more. But they cause a new problem: If Murat receives an encrypted document from Chantal and wants to decrypt any ciphertext word c_j, he needs to compute the stream cipher bits to be xored off c_j to yield $E^{\text{pre}}_{k''}(w_j)$. But since the suffix has been generated using the plaintext w_j, he can only decode c_j if he already knows w_j. This threatens to render the whole exercise of encryption meaningless.

To allow for recovering of w_j without knowledge of w_j, we need to fix keying of F one last time. Observe that Murat is able to generate s_i from R_k, so he can retrieve the first $l - m$ bits of $E^{\text{pre}}_{k''}(w_j)$ without computing $F(s_i)$. Therefore, if the key of F only depends on those bits, Murat can compute $F(s_i)$ from the ciphertext and k and k' alone. To obtain such a key, we split up $E^{\text{pre}}_{k''}(w_j) = L_i \,\|\, R_i$ such that $|L_i| = |s_i|$, and the final suffix of s_i for encryption of plaintext word w_i is

$$F_{f_{k'}(L_i)}(s_i)$$

During decryption, L_j is computed via s_i from the first $l-m$ bits of c_j and k; the suffix is computed from s_j, L_j, k'; R_j from this suffix and the remaining m bits of c_j; and w_j from k'' and $L_j \,\|\, R_j$ by means of D^{pre}.

When encrypting a query w_i, Murat pre-encrypts w_j, takes the first $l-m$ bits L_i and computes $f_{k'}(L_i)$, and submits $(E^{\text{pre}}_{k''}(w_i), f_{k'}(L_i))$ to Chantal.

Other approaches

Bloom filters (Blo70) can be used as searchable index data structures and attached to the encrypted documents (Goh03).

Also, public-key encrypted databases have been considered: If Chantal stores all of Murat's e-mail, and all of Murat's contacts encrypt their corre-

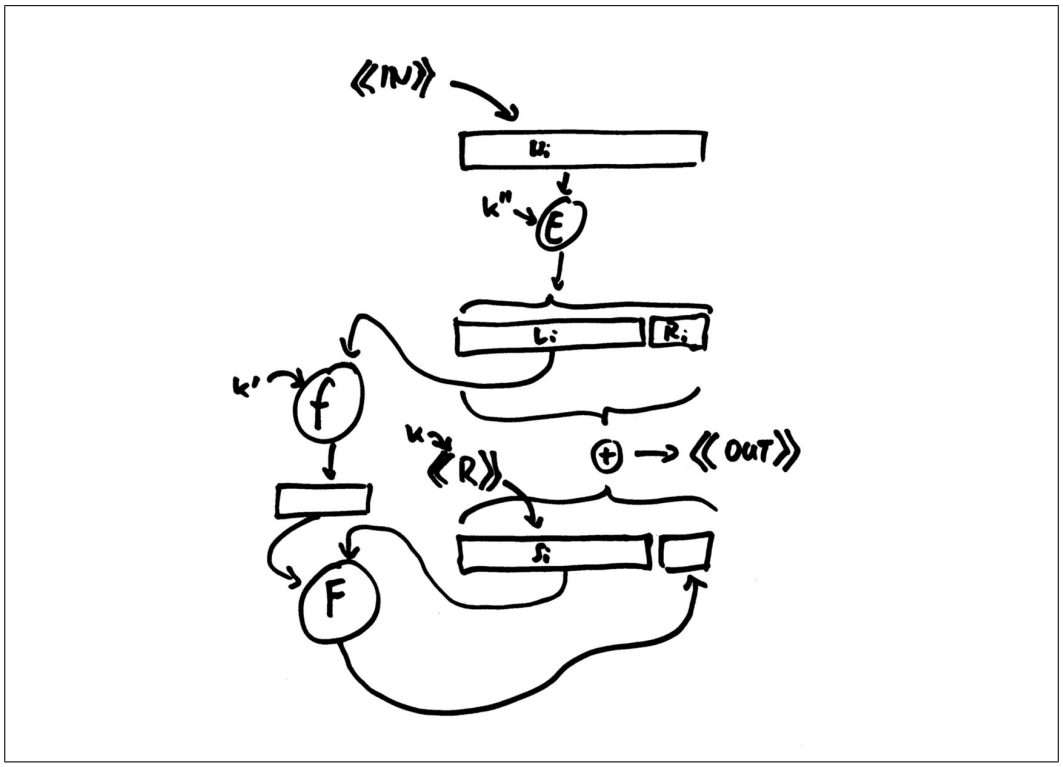


Figure 3.3: $\text{Fts}^{[\cdot]}$: Homomorphic encryption for full-text search as in (SWP00).

spondence with Murat's public key, there is a scheme that allows Murat to retrieve only those documents that contain a given keyword, without allowing Chantal to decrypt his e-mails (BCOP04).

(CM05) has proposed an incremental encryption scheme that maintains a separate index data structure and is thus independent of the encryption scheme used. Since search takes place on the separate index structure, this scheme does not give away locations of found keywords to Chantal, which provides for stronger security in a theoretical sense. Further, it allows for compression of the documents to be stored. However, if many search terms are to be stored for each document, performance in space and computing time degrades considerably.

During completion of this thesis, two new schemes have been proposed that show better server efficiency (CGKO06). Interestingly, they also satisfy more robust security definitions than $\mathrm{Fts}^{[\cdot]}$ (see Section 3.2.2). These schemes create search data structures on the client side that allow the server to process queries in time constant in the size of the searched data. The structure of the index data is highly specialized for text search purposes.

3.1.4 Homomorphic Signatures

In contrast to encryption schemes, signature schemes are used to enforce message integrity (see Section 2.3.5). It seems odd to think about homomorphic signature schemes. Given two messages m_1, m_2 with signatures $S(m_1), S(m_2)$, this means we can compute the signature $S(m_1 \circ m_2) = S(m_1) \circ S(m_2)$ for the message $m' = m_1 \circ m_2$, and we can go on combining resulting message / signature pairs to obtain new ones ($m'' = m' \circ m_1$, $m''' = m'' \circ m_1, \dots$) recursively. But none of the resulting messages was ever intended to be signed by the signer of m_1 and m_2.

However, there are applications. For example, in some hierarchical routing protocols, routers are faced with the problem of aggregating routing information. If routing information is signed to counter redirection attacks, the signatures need to be aggregated as well. It is possible to represent the routing information in a binary tree such that the aggregations are parent nodes of their unaggregated components. In (CRR02), a digital signature scheme on such binary trees is presented that allows for the signature of a node to be computed from the signatures of its children (or, by recursion, from the signatures of all leafs under that node), without the private key. The security definition states that no signature of any node can be computed *without* the knowledge of the two child signatures (recursive or direct).

A more academic problem is the computation of transitive hulls on discrete finite graphs: Given the signatures of two edges $(u, v), (v, w)$, can it be

possible to compute a valid signature for (u, w) without the private key, but not for any other edge in the graph for which no signed path is available? A solution for undirected graphs is proposed in (MR02). Applications could lie in the field of military hierarchy management. However, a scheme for directed graphs would be more suitable here, but such a scheme does not exist. Also, a simpler approach usually works that relies on non-homomorphic signature schemes: In order to check legitimacy of an edge (u, v), search for a path from u to v, and collect signatures for all edges on that path. If all those signatures are valid, so is the (non-existing) signature for (u, v). (If signatures are missing, search all other paths from u to v.)

(JMSW02) describes two further applications: Text redaction (the process of replacing parts of a text document with a special censor symbol) and set union and subsets, presents security definitions and schemes for solving these application, and proves that there is no solution for the problem of homomorphically signing integer addition structures.

Service outsourcing does not usually require homomorphic signatures for integrity enforcement. Murat can use conventional message authentication techniques on the database and on each incremental change to it. In fact, homomorphic signatures are less suitable than conventional ones: If Murat has to use the former for some reason, Chantal has subtly more freedom over the database states that have legitimate signatures (although if the schemes presented in (JMSW02) is used, all redacted data is clearly marked so).

3.2 Homomorphic Security Definitions

Homomorphic encryption schemes are encryption schemes, so the security definitions for the latter technically can be used on the former without change. Unfortunately, now there is additional information for the adversary in the form of encrypted queries and matching encrypted query results. In this section, we look into the consequences, and summarize existing work on dealing with them.

3.2.1 Small and Finite Operand Domains

The first homomorphic encryption scheme, plain RSA (RAD78), has only one operator, namely multiplication in the RSA residue class field:

$$\Phi = \{\cdot\}$$

This means that confidentiality requirements for operators make no sense — the adversary knows the decryption of any $E(\cdot) \in \Psi$, independent of the size

of Ψ or of the encryption algorithm E.

All confidentiality requirements for algebras with $|\Phi| = 1$ are about the operands, so if Murat passes two messages $m_0, m_1 \in \mathcal{M}$ to Chantal, we can reuse any security definition from Section 2.3.4 without change.

However, note that the homomorphicity requirement introduces new obstacles for security: As we have mentioned earlier, there is a tension between the wish to allow Chantal to process ciphertexts in a meaningful way and the wish to keep her from being able to gain knowledge about the plaintext.

For example, RSA is not chosen plaintext secure (see Definition 2.6). Since two plaintexts are equal iff two corresponding ciphertexts are equal, the adversary can work around her problem that the oracles do not give away the plaintext under attack: First, Chantal simply generates four plaintext messages $m_{0,0}, m_{0,1}, m_{1,0}, m_{1,1}$ at random and has Murat encrypt them into the ciphertexts $c_{0,0}, c_{0,1}, c_{1,0}, c_{1,1}$. To prompt for Murat's challenge, she computes $m_0 = m_{0,0} \cdot m_{0,1}$ and $m_1 = m_{1,0} \cdot m_{1,1}$, obtains c_i, and has to guess $i \in \{0, 1\}$. But no matter what i, Chantal can compute $c_i = c_{i,0}c_{i,1}$. She merely has to do two multiplications and two comparisons to achieve a winning probability of 1. Hence,

Theorem 3.2. *No deterministic encryption scheme homomorphic over a multiplicative group is secure in Definition 2.6.*

Small $|\Phi| > 1$, such as in complete field arithmetics, are similar in their security characteristics. Consider a security definition game in which the adversary receives two ciphertext operands and one ciphertext operator. Under the best circumstances, the number of bits the adversary needs to guess correctly for retrieving the operator is increased by $O(\log |\Phi|)$ with respect to a successful attack against a scheme with $|\Phi| = 1$. If the adversary manages to get these guesses right by chance, she will know which operator has been used. However, since we are only interested in asymptotical performance, the effect of decreasing the odds of the adversary by a small constant factor will not suffice to satisfy harder security definitions. (In Section 4.1, we will consider several wrong ways of hardening cryptographic schemes that cover this one.)

Operand Confidentiality

In (DF02), Domingo-Ferrer claims to have found a homomorphic encryption scheme and a suitable security definition that satisfies it. We have already outlined the encryption scheme in Section 3.1.2; the following definition is used in an attempt to establish its security:

Definition 3.3 (Domingo-Ferrer Security (DF-Security))**.** *An encryption scheme* (G, E, D) *is* DF-secure *if, for any fixed number* k *of plaintext-ciphertext pairs* $m_1, c_1, \ldots, m_k, c_k$ *known to Chantal, and any probability* ϵ*, there is a suitable choice of parameters such that Chantal cannot infer* $m = D(c)$ *for any ciphertext* c *with probability higher than* ϵ*.*

In case of $(G^{\mathrm{df}}, E^{\mathrm{df}}, D^{\mathrm{df}})$, the parameters to calibrate to the desired security level are d, N, n, r. We can assume a globally constant k: The easiest way to achieve this is for Murat to change the key every k encryptions, thereby rendering all previously captured plaintext-ciphertext pairs useless for any attack on future ciphertexts.

This is an information theoretic security definition, meaning it does not restrict Chantal's computing power. Instead, it limits the level of certainty she can possibly have over the correctness of the result of any attack.

Note one peculiar gap, though. Traditional information-theoretic security (see Section 2.19) always bounds adversarial success probability as the inverse of a super-polynomial function in some security parameter that increases linearly with the increase of work required by the honest parties involved. SF-security, on the other hand, defines ϵ as just any function in the parameters. It does not require the function to be in $O(1/p(\ldots))$ for every polynomial in these parameters. As a consequence, feasibility in terms of performance and security have been merged in one definition: An encryption scheme may be secure, but only at the cost of an inacceptably high workload of the honest parties in order to achieve the desired ϵ.

This may be a mere inconvenience, and does not invalidate the security notion. Furthermore, it is unclear to us to what extent $(G^{\mathrm{df}}, E^{\mathrm{df}}, D^{\mathrm{df}})$ has this problem. Inefficiency is not a necessary consequence of Definition 3.3, but it is consistent with it: The usual upper bound on the complexity of the encryption scheme that grows polynomially in the parameters is simply missing.

A more severe problem is that for all practical purposes, one would like to have a way to derive actual parameters for any given ϵ, or at least vice versa. The abstract knowledge that parameters exist to render a scheme "secure enough" is not sufficient.

Domingo-Ferrer's proofs attempt to provide such a relation for $(G^{\mathrm{df}}, E^{\mathrm{df}}, D^{\mathrm{df}})$, but has been proven to be far too optimistic (Wag03). To our knowledge, no fix for the original scheme has been proposed since then. Hence, at least for all practical purposes, we have to leave the question whether Definition 3.3 has an instance or not unanswered for the time being.

The (flawed) argument for the security of Domingo-Ferrer's encryption scheme follows the following form: (1) For any single plaintext-ciphertext

pair, there are many possible keys (r, n); (2) only a small fraction of these keys is equivalent to the real key; (3) therefore, no key can be retrieved (as always, with sufficiently high probability); (4) therefore, plaintexts cannot be retrieved from ciphertexts.

These steps bear one, at least in retrospect, rather crude mistake, and a more subtle one. First, if (4) would follow from (3) without further assumptions, the best encryption scheme would be the identity "encryption" mentioned from Example 2.1: Since the key is completely ignored during encryption and decryption, it is perfectly safe from Chantal, who has only access to plaintexts and ciphertexts. Hence, although $c = m$, she cannot compute c from m. The missing assumption is that there is no *other* way to retrieve the plaintext from the ciphertext that does not require knowledge of the key.

The more subtle mistake is rooted in the implicit assumption that every bit of the key (r, n) is required for a successful decryption. However, since $r \in \mathbb{Z}_N$, and the plaintext $m \in \mathbb{Z}_n$ with n being much smaller than N, the key space collapses into equivalence classes that are considerably fewer in numbers. Using well-known properties of resultants, r' can be chosen such that

$$r' \equiv r \mod n$$

In other words, such that r' and r are both representatives of the same equivalence class of keys. (r', n) can then be used instead of (r, n). We are omitting the actual algorithm for inferring (r, n). The curious reader is referred to (Wag03).

3.2.2 Full-Text Search

The appendix of (SWP00) contains a proof of security, where security is implicitly defined in the following theorem (rewritten to fit our notation):

Theorem 3.3. *If f, F are secure pseudo-random functions in the sense of Definition 2.14 and R is a secure PRG in the sense of Definition 2.12, then the PRG component of $Fts^{[\cdot]}$ that outputs $(S_0, F(S_0), S_1, F(S_1), \dots)$ (the lower half of Figure 3.3) is secure in the sense of Definition 2.12.*

In other words, instead of indistinguishability (or some variant thereof) of the full-text search encryption scheme, the proof establishes security of the underlying PRG. What the paper does not make very explicit is that deriving security of the full-text search encryption scheme from this result requires a considerable relaxation of the security definition.

This leaves a number of open issues. For a start, multi-message security needs to be established explicitly, which is doable but yields a few key management issues that need to be solved as well.

Further, each occurrence of some word w in the corpus that has been searched for is available in the raw pre-encryption $E^{\text{pre}}_{k''}(w)$ to the adversary. Each query pokes holes into the stream cipher, and in the extreme case that all words have been searched for at least once, the security provided by the stream cipher, no matter how provable, has been completely annihilated. This issue can be addressed by sporadic re-encryptions, but it is much harder to talk about provable security in its presence.

The key shortcoming of the treatment of security is the failure to take the full homomorphic structure of the ciphertext, both data and operators, into account. Since both interact in some meaningful way even when encrypted, this leads to the revelation of confidential data. This is a common pattern in this area of research, and we will uncover more of its instances in the following. Independently of our work (EFG06; EFG07), new security definitions for full-text search have been proposed together with encryption schemes that satisfy them (CGKO06). Adaptive indistinguishability is equivalent to our Definition 4.4, and we will compare our results to theirs in further detail in Section 4.3.2.

3.2.3 Relational Algebra

Consider a database PH (G, E, E^*, D) such that the encryption scheme (G, E, D) is indistinguishably secure and such that E^* is the identity function on queries, i.e. for all q, $E^*(q) = q$. (Remember the main concern of homomorphic encryption is data confidentiality, not query confidentiality.) Assume an adversary Chantal runs the query $\sigma_{a_i:d_i} = E^*(\sigma_{a_i:d_i})$ on the encrypted table. As result of her computation, Chantal obtains a set of encrypted tuples. Although she cannot decrypt them, she can infer that the value of the attribute a_i of these tuples is d_i merely from the fact that they are in the query result. So even though we used a rigorous security definition and assumed a scheme that meets this definition, in a way (G, E, E^*, D) is considerably less secure than ATE (see Section 3.1.1).

How is it possible that a database PH based on a perfectly secure table encryption scheme can still leak so much information? What tricked us here is an ill-applied adversary model. The classical model does not consider that the adversary gains additional information about the plaintext data when receiving and processing the queries. The above example shows that these computations can cause unsuspected leaks. In order to capture this new situation, we need to craft new security definitions that are aware of the

adversary's new capabilities. In other words, we need to take E^* into account. (KC04) propose a definition that addresses this problem and requires that, in addition to encrypting tables, the queries should be securely encrypted as well:

Definition 3.4 (Kantarcıoğlu-Clifton security for databases)**.** *A database PH is secure according to the Kantarcıoğlu-Clifton model (or KC-secure) if*

1. *Any two tables of the same size are indistinguishable (i.e., secure in the sense of Definition 2.4).*
2. *Two queries that run on the same set of tables and return results with the same number of tuples are indistinguishable.*

Kantarcıoğlu and Clifton suspected that their definition may already be too strong for any scheme to exist that satisfies it. Although we show that this is not true, there is another more serious problem with it. It is in fact still too *weak* to match our intuition of what security means: A scheme that is KC-secure may still reveal some information about an encrypted tables.

Example 3.1 (Attacking Homomorphic Database Encryption)**.** *Suppose a KC-secure homomorphic encryption for relation algebra with exact select queries only (See Definition 4.7), and a patient database with statistics for three hospitals. For each patient, we store HIV status and the hospital they are assigned to:*

id	*hospital*	*hiv*

Now suppose that Chantal knows the database schema, the number of hospitals, has good estimates of the distribution of patients among hospitals $(0.2, 0.3, 0.5$*, resp.) and the percentage of HIV-positive patients (say,* 8%*).*

Murat issues the following queries:

```
SELECT * FROM table WHERE hospital = 1;
SELECT * FROM table WHERE hospital = 2;
SELECT * FROM table WHERE hospital = 3;
SELECT * FROM table WHERE hiv = "positive";
```

Since the results are different in size, Chantal can guess the queries with high confidence from the size of the output they produce. This is not in itself a problem, as long as our concern is privacy of the database, not privacy of the queries.[7] *But by intersecting the answers to the first and the fourth query, Chantal can now infer the ratio of HIV-positive patients in hospital* 1*!*

[7] Query privacy without data privacy is called private information retrieval (PIR), and is treated in Section 3.4. As we will see, the two affect each other in many ways, so constructing hybrids will prove worthwhile.

Note that in this scenario, Chantal is a passive adversary with some background knowledge. Because exact selects are allowed (See Definition 4.7), she knows enough about the queries from the known-ciphertext perspective. The problem of Definition 3.4 is that the indistinguishability requirement for encrypted queries is restricted to those that return results of the same size. Queries violating that restriction are still likely to leak information.

ATE (see Section 3.1.1) does not come with a security definition. We have published an extensive list of alternatives, plus a number of modifications to the original scheme, and shown that ATE is inherently insecure, and repairing it is exceedingly hard (if possible) (FG03). We will present our results in Section 4.1.

3.3 Code Obfuscation

In mobile agents research, we are confronted with the problem that a program (an agent) is run on a system (a host) that may act in conflict with the legitimate agenda of the agent. It is easy to see that protecting hosts against malicious agents is far easier than protecting agents against malicious hosts since ultimately, the host is always in charge. It may simply refuse to run the agent if it is not convinced that the agent will behave well, or grant it access to only those parts of the system that are not sensitive (the latter is called the *sandbox approach*). On the other hand, an agent that has reached the host can do nothing to prevent inspection of its code, arbitrarily many re-runs, timing attacks (an example involving smart cards is Kocher's attack against a number of public-key cryptosystems (Koc96)), and so forth. This may provide the host with information about the agent's code that is (a) not needed for serving the agent, and (b) confidential.

Code obfuscation promises to prevent malicious hosts from learning the agent's secrets. Intuitively, an obfuscator transforms a program into another program with equivalent functionality, but so that nothing can be learned from an obfuscated program except what can be learned from oracle access to it (see Section 2.3.2). An obfuscated agent on a malicious host is worse off than a host running a malicious agent: The host can still decide on which input to run the agent, where to interrupt it, and has still a range of analysis techniques at its disposal. Hence, this is as far as we can hope to possibly get in mobile agent applications security.

The same problem arises in other contexts such as intellectual property protection. Code obfuscation does not only help against patent theft because the latter usually requires reverse engineering, but is similar in nature to techniques like *watermarking*, where the code hides a message of authentic-

ity that is hard to find or remove, and *fingerprinting*, where the authenticity message contains an additional serial number that can be linked to the legitimate owner of a specific copy, who can then be charged with the copyright infringement.[8]

What is more interesting to us, homomorphic encryption and code obfuscation are closely linked, too. If we can generate a secure obfuscation $\mathcal{O}(M)$ of an arbitrary program M, homomorphic encryption with respect to arbitrary operator sets becomes very simple. For pair $(\mathcal{M}, \Phi)$ of operand set and operator set, simply take a symmetric non-homomorphic encryption scheme $(\mathcal{K}, E, D)$, generate a secret key k, and define the secure equivalent of any operation ϕ to be $\phi^* = \mathcal{O}(E_k \circ \phi \circ D_k)$. Trivially, ϕ^* (on ciphertexts) is equivalent to ϕ (on plaintexts). By the security of $(\mathcal{K}, E, D)$, the key k cannot be observed from the IO-behavior of the program. Because the obfuscation operation is secure, the obfuscated ϕ^* does not tell any more than that. An adversary having access to ϕ^* can perform the operation ϕ on ciphertexts chosen from $\mathcal{M}$ without learning anything about them. Hence, we have a homomorphic encryption scheme that is as secure as the symmetric (non-homomorphic) encryption scheme that we used in the construction.

In the opposite direction, given the dual nature of code and data captured in the idea of universal Turing machines, any language for which we can find a homomorphic encryption scheme can be securely obfuscated. In particular, if we could find a homomorphic encryption scheme for a Turing-complete language of operators, we would have a general solution for the problem of code obfuscation.

Unfortunately, no such general solution exists. More seriously, it turns out that finding useful language classes that can be securely obfuscated is extremely hard. In this section, we will give a brief example for a code obfuscation technique that (like all the proposals we are aware of) has been proposed as a pragmatic solution to intellectual property issues rather than with a rigorous cryptographic security notion in mind. Then, we elaborate on the what can not be done in principle. This will have to be done in some detail, since we will need the theoretical devices developed in the proofs for our results on homomorphic encryption of relational algebra.

[8] Although we are not concerned with sociology here, for the sake of completeness we would like to point out that these technical solutions to a social problem have problematic economical, ethical and legal implications. For example, a law-abiding owner of a fingerprinted software is vulnerable to weaknesses in the fingerprinting scheme that allow an adversary to blame her for illegal copying, and weaknesses in software are common enough to worry about this. While the client has a strong interest in the fingerprinting scheme to be secure, the vendor (whose responsibility it is to guarantee its quality) has the (at best) orthogonal interest of finding a culprit for the crime.

3.3.1 Proposed Solutions

There is an abundance of techniques for code obfuscation. Many are ad-hoc and only available in the form of software products, but some have been given a more scientific treatment (CTL98a; CTL98b; BDK+01; Wro02).[9] Let us have a closer look at a prominent example. (CTL98b) propose to use *opaque predicates* in branches that are added to the control flow of the code. The value of an opaque predicate is known to the obfuscator, but not to the reverse engineer. This allows the obfuscator to add misleading code that is never reached during any execution of the program.

Opaque predicates are constructed using dynamic pointer-based data structures that are hard to understand using static code analysis techniques such as abstract interpretation (NNH99). These structures are nested into the *live* data structures (those required by the function that the code computes) in a way that makes them hard to identify as *dead* data (added merely for the sake of obfuscation). An opaque predicate may have the form "pointer x_1 and pointer x_2 point to the same address", for two dead pointers somewhere on the heap, and dead code can be added to modify those two pointers every now and then during the program run. This allows the obfuscator to modify the branch pointers in sync with the opaque predicate such that as long as the predicate's value is unknown, the two branches are made sure to be functionally equivalent, but as soon as it is known, the branch that becomes dead may contain deceptions.

This technique was proposed in 1998, two years before the discovery that code obfuscation is not only hard, but in fact unsolvable (see Section 3.3.2). However, in 2002 in a survey on intellectual property protection technology (CT02), the authors still claim that their solution provides "*some* degree of protection". Although such a soft claim is safe from outright refutation, we argue that it is at least dubious.

The assumption that the reverse engineer is restricted to static analysis of the code is wrong: Since she has access to the (obfuscated but still operational) code, she can run it on arbitrary inputs, produce large amounts of traces, and analyze those instead of abstract representations of a superset of all traces theoretically possible. If the input distribution is random, an obfuscator that is aware of this attack algorithm may be able to find a way

[9] Off-topic and on a more humorous note, there are several competitions that reward manually obfuscated code: The International Obfuscated C Code Contest (`http://www.ioccc.org/`) demonstrates the value of a transparent programming style by counter example, and the Underhanded C Contest (`http://www.brainhz.com/underhanded/`) challenges its participants to write code that appears to implement an innocent algorithm but contains malicious code. See Figures 3.3.1 and 3.3.1.

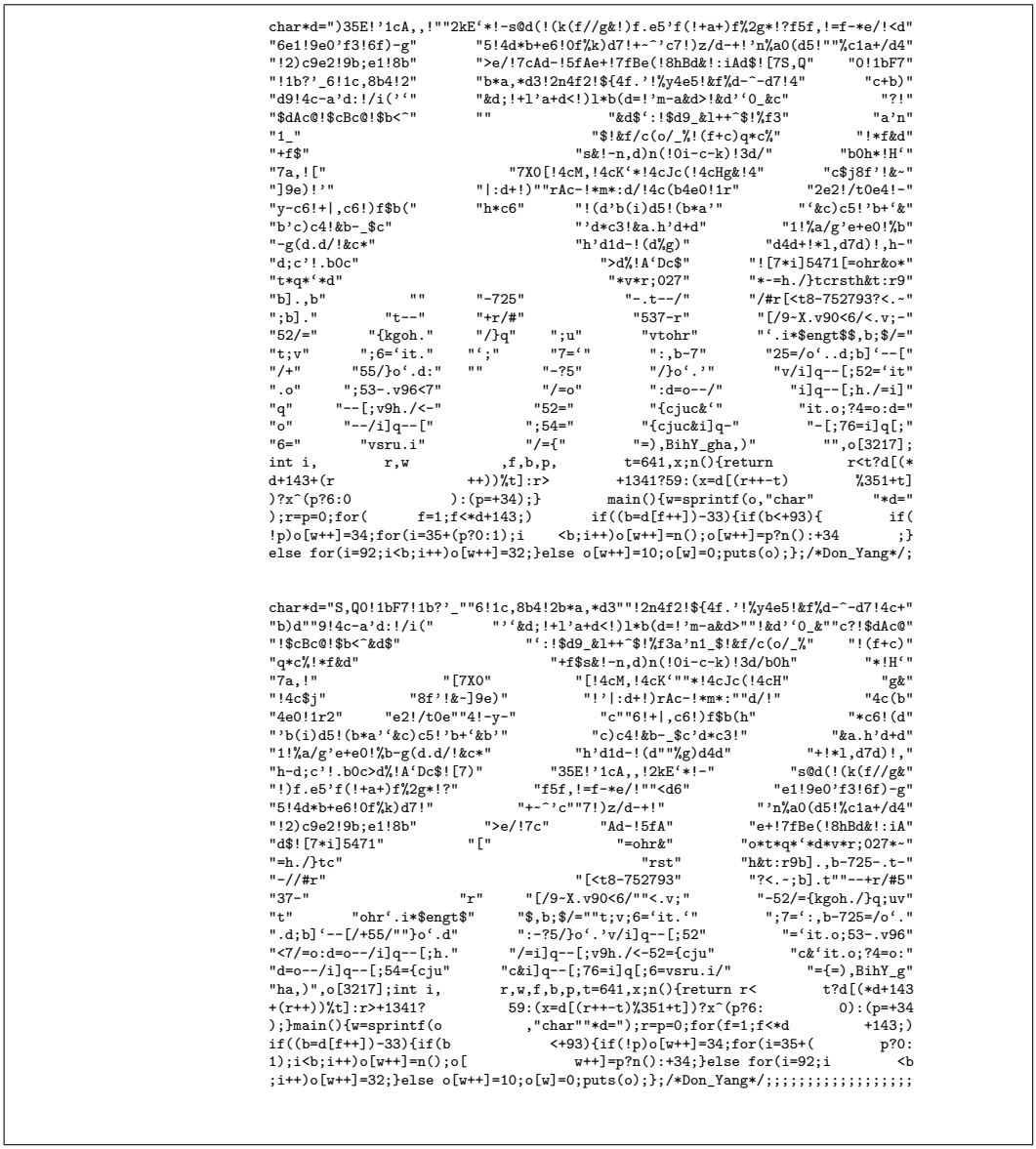


Figure 3.4: Don Yang's submission to the 2000 Obfuscated C Contest is a self-replicator. Each picture is a valid C program that prints the next one to stdout. There is also a third block (not shown here) produced by the second, which in turn produces the first block again.

```
size_t hash(wchar_t* word) {
  char* s=(char*)word;
  size_t x;
  ssize_t i;
  /* looping backwards is more efficient */
  for (x=i=wcslen(word)*sizeof(wchar_t); i>=0; i--) {
    x += ~(s[i+1]) + s[i];
  }
  return x;
}
```

Figure 3.5: Felix von Leitner's submission to the 2006 Underhanded C Contest illustrates a problem related to code obfuscation: Obfuscation of a specific aspect of a piece of code, or of a private agenda of the authors of the code. The challenge was to write code that runs extremely slow on a "competitor's platform" of choice, while efficient on another. This hash function can be used to implement poorly balanced hash tables. The value of x subsequently adds and subtracts each character in a string once, ending up with the first character of a file as the hash value. On little-endian systems this yields a poor distribution, but for English texts there should still be around 60 possible return values (one for each character used). However, since the return value is 32 bits, on big-endian systems, the return value is the constant 0.

to trick it. However, the reverse engineer has many ways to produce input of higher relevance to her problem at hand (e.g., by examining the control structures in the code or by analyzing the heap and the traces of preliminary program runs). Furthermore, we suspect that security is increased linear in the security parameter since these attack techniques are incremental, i.e., the reverse engineer may be able to remove one opaque predicate after the other, simplifying the code step by step.

Other obfuscation strategies have proven to be similarly weak. A recent NAI report comes to the conclusion that previous work has been not as successful as hoped (DMR+03) (although expressing the new hope that there may be certain useful subsets of programs for which obfuscation is possible). Therefore, in the following section, instead of refuting particular (classes of) obfuscation techniques individually, we will present the strictly more general and, in our eyes, more convincing result that *any possible* code obfuscation technique is necessarily insecure.

3.3.2 Impossibility Results

In this section, we give a summary of the results of (BGI+01). It consists of a notion of what an obfuscator does and under what circumstances it does it securely, and of a proof that the existence of obfuscators is impossible. We start with the definition of an obfuscator.

Definition 3.5 (TM Obfuscator)**.** *A* TM obfuscator *is a TM $\mathcal{O}$ such that for any TM M:*

1. *(functional equivalence) For every input x, $\mathcal{O}(M)(x) = M(x)$;*
2. *(polynomial slow-down) There is a polynomial p such that $|\mathcal{O}(M)| \leq p(|M|)$ and if $M(x)$ terminates in $\leq t$ steps, then $\mathcal{O}(M)(x)$ terminates in $\leq p(t)$ steps;*
3. *(black-box property) For every PPTM A, there is a PPTM S such that:*

$$|\Pr[A(\mathcal{O}(M), 1^{|M|}) = 1] - \Pr[S^M(1^{|M|}) = 1]| \leq neg(|M|)$$

A TM obfuscator is called efficient *if it is in $\mathcal{P}$.*

(Note that A and S have not only access to an oracle, but also are provided with the size of the unobfuscated TM. If that was not the case, A could try to use its knowledge of the size of $\mathcal{O}(M)$ to gain an advantage over S.)

Given there is no TM obfuscator, perhaps it is possible to securely obfuscate a class of algorithms that can be expressed with a computation model

less powerful than Turing machines? A far simpler computation model is that of boolean circuits: They have input and output of fixed finite length, the halting problem for circuits is decidable, and there is even an efficient algorithm to compute the running time of a circuit that is the same for any input. For instance, if we (falsely) assume for a moment that any obfuscator always fails on some TM, then there still *might* be circuit obfuscators. The reason for TM obfuscators to fail may be that it fails to conceal halting properties of those TMs, and since the halting proprerties of circuits are much simpler, in their case there is simply nothing to conceal, so the existence of circuit obfuscators was not ruled out yet. because it is impossible to conceal halting properties of Turing machines,

Therefore, two independent proofs are required for the two computation models, the very restricted one using circuits and the most general one using TMs, that no obfuscators exist. The two proofs together then indicate that obfuscation is impossible in general independently of the class of algorithms to be obfuscated.

Definition 3.6 (Circuit)**.** *An* n-m circuit *is a boolean function* $C : \{0,1\}^n \to \{0,1\}^m$ *with* n *input bits and* m *output bits.* C *can be expressed as a boolean expression composed of the operators* $\vee, \wedge, \neg$ *on the input bits.*

Definition 3.7 (Circuit Obfuscator)**.** *An* (n-m) circuit obfuscator $\mathcal{O}$ *is an algorithm such that for any* (n-m) *circuit* C*:*

1. *(functional equivalence) For every input* x*,* $\mathcal{O}(C)(x) = C(x)$*;*
2. *(polynomial slow-down) There is a polynomial* p *such that* $|\mathcal{O}(C)| \leq p(|C|)$*;*
3. *(black-box property) For every PPTM* A*, there is a PPTM* S *such that:*

$$|\Pr[A(\mathcal{O}(C), 1^{|C|}) = 1] - \Pr[S^C(1^{|C|}) = 1]| \leq neg(|C|)$$

In order to prove that neither of the two definitions can be satisfied, i.e., that there are no TM obfuscators and no circuit obfuscators, a few preliminaries are needed. As they are crucial for the understanding of our results in Chapter 5, they need to be developed in more detail.

We start by introducing the concept of 2-TM obfuscators (or 2-circuit obfuscators), and prove that no such thing exists. The counter example makes use of the circumstance that even under the (false) assumption that both are securely obfuscated, one TM (circuit) can use the other as input. On the other hand, with polynomially many oracle queries only, a carefully chosen

first TM (circuit) cannot be guessed with non-negligible success probability, and thus no input can be provided for the second.

After that, by explaining how two TMs (circuits) can be composed into one we will extend this result to 1-TM (or 1-circuit) obfuscators in Theorems 3.5 (easy) and 3.6 (complicated by the fixed input size of circuits).

Definition 3.8 (2-TM Obfuscator)**.** *A* 2-TM obfuscator $\mathcal{O}$ *is a TM obfuscator such that for every pair* M, N *of TM and every PPTM* A*, there is a PPTM* S *such that:*

$$\begin{aligned} |\Pr[A(\mathcal{O}(M), \mathcal{O}(N), 1^{|M|+|N|}) = 1] \\ - \Pr[S^{M,N}(1^{|M|+|N|}) = 1]| \\ \leq \mathit{neg}(\min\{|M|, |N|\}) \end{aligned} \tag{3.15}$$

Definition 3.9 (2-Circuit Obfuscator)**.** *A* 2-circuit obfuscator $\mathcal{O}$ *is a circuit obfuscator such that for any pair* C, D *of circuits and for every PPTM* A*, there is a PPTM* S *such that:*

$$\begin{aligned} |\Pr[A(\mathcal{O}(C), \mathcal{O}(D), 1^{|C|+|D|}) = 1] - \Pr[S^{C,D}(1^{|C|+|D|}) = 1]| \\ \leq \mathit{neg}(\min\{|C|, |D|\}) \end{aligned}$$

Lemma 3.4. *Neither* 2*-TM obfuscators nor* 2*-circuit obfuscators exist.*

Proof. For the case of Turing machines, we assume a 2-TM obfuscator and construct three Turing machines M, M^*, N. Then, we show that either M, N or M^*, N will necessarily violate Equation (3.15).

First, for any given $\alpha, \beta \in \{0,1\}^k$, let

$$M_{\alpha,\beta}(x) = \begin{cases} \beta & x = \alpha \\ 0^k \text{otherwise} \end{cases}$$
$$M^*_k(x) = 0^k$$

That is, M and M^* can be distinguished on their output only in α, and behave equivalently on all other inputs.

As we want to obfuscate *pairs* of TM, we need a second TM to pair with both M and M^*, respectively: Let $N_{\alpha,\beta}$ be the TM that checks whether its input is a TM that implements the same function as $M_{\alpha,\beta}$ in the domain point α:

$$N_{\alpha,\beta}(M) = \begin{cases} 1 & M \text{ is a TM} \wedge M(\alpha) = \beta \\ 0 \text{otherwise} \end{cases}$$

The trick is that N is configured with the parameters α, β matching those of M *a priori* to the obfuscation. (If you are beginning to suspect a gap in the argument here you are correct: N has a halting problem. However, this can be easily fixed. We will deal with it in the end of this proof.)

We have designed M, M^* such that their behavior can not be learned from interpolating oracle answers in polynomial time with non-negligible probability: The value α is unknown and can only be guessed with probability 2^{-k}. Formally, for every PPTM S, if α, β and the coin tosses of S are chosen uniformly at random, we have:

$$|\Pr[S^{M_{\alpha,\beta},N_{\alpha,\beta}}(1^k) = 1] - \Pr[S^{M^*_k,N_{\alpha,\beta}}(1^k) = 1]| \leq \mathrm{neg}(k) \tag{3.16}$$

In order to complete the proof, we need to show that the respective obfuscated representations $(\mathcal{O}(M), \mathcal{O}(N))$ and $(\mathcal{O}(M^*), \mathcal{O}(N))$ are more informative than the oracle access given to S. To be more precise, that there is a PPTM A to distinguish the two with non-negligible probability. And here it is:

$$A(\mathcal{O}(M), \mathcal{O}(N)) = N(M)$$

That is, A simply invokes N with input M. Because obfuscation does not change the behavior of a TM, we have:

$$\Pr[A(\mathcal{O}(M_{\alpha,\beta}), \mathcal{O}(N_{\alpha,\beta})) = 1] = 1 \tag{3.17}$$

and:

$$\Pr[A(\mathcal{O}(M^*_k), \mathcal{O}(N_{\alpha,\beta})) = 1] = 0 \tag{3.18}$$

Equations (3.15) and (3.17) imply that $\Pr[S^{M_{\alpha,\beta},N_{\alpha,\beta}}(1^k) = 1] = 1$, and Equations (3.15) and (3.18) imply that $\Pr[S^{M^*_k,N_{\alpha,\beta}}(1^k) = 1] = 0$ (both up to a negligible error). But Equation (3.16) cannot be consistent with both. Hence, (3.15) must be false, and the assumption of the 2-TM obfuscator $\mathcal{O}$ is invalid.

The refutal of 2-circuit obfuscators as described in Definition 3.9 is completely analogous. Two technicalities conclude the proof.

The TM case and halting issues. N does not terminate in general, because question whether a TM returns β on input α is undecidable. To take this into account, we introduce a timeout t such that $\mathcal{O}(N_{\alpha,\beta})$ is terminated with output 0 after t steps. t is chosen greater than the time it takes for $\mathcal{O}(N_{\alpha,\beta})$ to terminate on input $\mathcal{O}(M_{\alpha,\beta})$. (If there is no other way to estimate t, simply run $\mathcal{O}(N_{\alpha,\beta})(\mathcal{O}(M_{\alpha,\beta}))$ and measure the time before submitting it to the adversarial algorithms.)

The circuit case and input size issues. Because we want to feed the circuit M to N as input, there is a lower bound on the number of input gates of

N. Furthermore, we want to feed $\mathcal{O}(M)$ to the (obfuscated, but functionally equivalent) N, so the lower bound is increased by the polynomial that bounds the growth of M during obfuscation. This is easy to achieve, since $\mathcal{O}(M)$ is known at the time of construction of N, and because of the polynomial slow-down property of $\mathcal{O}$. (Note that smaller input can always be fit into bigger circuits by using padding.) □

In order to extend this result to the non-existence of TM obfuscators, we introduce a simple composition device: For two functions, TMs, or circuits $f_0, f_1 : X \to Y$, we define the function, TM, or circuit

$$\begin{aligned} f_0 \# f_1 &: \{0,1\} \times X \to Y \\ (f_0 \# f_1)_i(x) &\mapsto f_i(x) \end{aligned}$$

Access to a composition gives direct access to its components by simply fixing the first input bit. Therefore, we may hope that situation in which we are in possession of one composite TM is equivalent to the situation in which we are in possession of the two components. This can easily be proven for TMs.

Theorem 3.5. *TM obfuscators as in Definition 3.5 do not exist.*

Proof. Assume there is a TM obfuscator $\mathcal{O}$. Then we can use $\mathcal{O}$ to obfuscate compositions of the TMs $M_{\alpha,\beta}, M_k^*, N_{\alpha,\beta}$ as constructed in the proof of Lemma 3.4. In particular, there are no successful adversaries against $M_{\alpha,\beta} \# N_{\alpha,\beta}$ and $M_k^* \# N_{\alpha,\beta}$.

However, for an adversary

$$A(M) = (M)_1((M)_0)$$

Equations (3.16), (3.17), and (3.18) from Lemma 3.4 apply, and thus A wins the game in the same way as it wins the game against any 2-TM obfuscator. □

Unfortunately, establishing the equivalent result for circuits requires quite a few afterthoughts. Also, we can only achieve the impossibility result for *efficient* (not for computationally unbound) circuit obfuscators. (Note, however, that since we reasonably assume computationally bound adversaries in the rest of this thesis, the two are equivalent for all practical purposes.)

Theorem 3.6. *Efficient circuit obfuscators as in Definition 3.7 do not exist.*

Proof. (As before, unless specified otherwise see proof of Lemma 3.4 for definitions of the parameters used here.)

If we simply try the composition trick that worked for TMs, the proof does not work: The two circuits $(M\#N)_0$ and $(M\#N)_1$ both have (at least) the same size as $M\#N$, so the fixed-size input of $(M\#N)_1$ cannot consume all of $(M\#N)_0$. Further, since the adversary has only access to the obfuscated composition $\mathcal{O}(M\#N)$, any trick that can be used to compress $(M\#N)_0$ can be used in a successful attack against $\mathcal{O}$. However, $\mathcal{O}$ is secure by assumption, and our aim in this proof is to derive a contradiction (and not cause one by introducing more assumptions).

Somehow, we need construct a more sophisticated N such that we can pass M to N (or an N-oracle) *in chunks* without learning anything about M from its chunks or from the partial results of the computation. This N needs to be a composition of three algorithms: One to prepare an input for M and pass it to the adversary; one to do a gate-wise encrypted evaluation of M on an encryption of its input α, yielding an encryption of $M(\alpha)$; and one to decide whether $M(\alpha) = \beta$ without revealing either.

Given a bit-wise encryption scheme

$$(G^{\text{bit}} : \{1\}^* \to \mathcal{K}, E^{\text{bit}} : \{0,1\} \to \{0,1\}, D^{\text{bit}} : \{0,1\} \to \{0,1\})$$

that is (a priori) chosen ciphertext secure and chosen plaintext secure. It is safe to assume the existence of such a scheme (see the last paragraph of this proof). Notice that in order to be secure in such a strong sense, $(G^{\text{bit}}, E^{\text{bit}}, D^{\text{bit}})$ needs to be highly non-deterministic. Each time we encrypt a single bit of plaintext, the output must appear completely random to the adversary, even if she is allowed access to encryption and decryption oracles.

Now define three circuits: N^{in} encrypts a given input α, N^{thru} computes an encryption of the output of one gate of $M(\alpha)$, and N^{out} decides whether the output of $M(\alpha) = \beta$ using only an encryption of $M(\alpha)$.

$$N^{\text{in}}(\alpha_0, \ldots, \alpha_{k-1}) = (E^{\text{bit}}(\alpha_0), \ldots, E^{\text{bit}}(\alpha_{k-1}))$$

$$N^{\text{thru}}(x_0, x_1, \odot) = E^{\text{bit}}(D^{\text{bit}}(x_0) \odot D^{\text{bit}}(x_1))$$

$$N^{\text{check}}(\beta_0, \ldots, \beta_k, x_0, \ldots, x_k) = \begin{cases} 1 & \text{if } D^{\text{bit}}(x_0) = \beta_0 \wedge \ldots \\ \quad \wedge D^{\text{bit}}(x_k) = \beta_k & \\ 0 & \text{otherwise} \end{cases}$$

To be able to fix α, β, we need a circuit that just outputs its input:

$$N^{\text{id}}(x) = x$$

Let

$$N_{\alpha,\beta} = (N^{\text{in}} \circ N^{\text{id}}(\alpha))\#N^{\text{thru}}\#(N^{\text{check}} \circ N^{\text{id}}(\beta))$$

where the first component is a function composition that returns an encryption of α when run with no input, and the third is the test whether a given input is an encryption of β without revealing β.

Consider two inputs in analogy to the proof of Theorem 3.5 (only that they are now implemented as circuits), namely $M_{\alpha,\beta}\#N_{\alpha,\beta}$ and $M_k^*\#N_{\alpha,\beta}$.

Now we construct an adversarial PPTM $A(\mathcal{O}(M\#N_{\alpha,\beta}), 1^{|M\#N_{\alpha,\beta}|})$ such that no PPTM $S^{M\#N_{\alpha,\beta}}(1^{|M\#N_{\alpha,\beta}|})$ exists as described in the last condition of Definition 3.7. In other words, we show that oracle access to the obfuscated circuit reveals strictly less information about the algorithm than the circuit itself.

In fact, A works just like in the previous proofs in this section, only that it utilizes the decomposition of $N_{\alpha,\beta}$ to compute M gate by gate: First, it retrieves an encryption of α from $(N)_0$, then it splits M up in its individual gates and runs them one by one on the encrypted intermediate bits using $(N)_1$. Finally, it passes the encrypted result to $(N)_2$ and outputs the result.

This allows A to distinguish reliably between $M_{\alpha,\beta}\#N_{\alpha,\beta}$ and $M_k^*\#N_{\alpha,\beta}$. However, any S that has mere oracle access to its input circuit can not split up M in its individual circuits; all it can do is probe M for polynomially many inputs of its choice. The argument is analogous to those above in this section. See (BGI$^+$01) for the full proof. □

Curiously, this proof uses a homomorphic encryption scheme with respect to circuit evaluation with $\Phi = \{\vee, \wedge, \neg\}$, $\Psi = \{N^{\text{thru}}(\cdot,\cdot,\odot) \mid \odot \in \Phi\}$. This seems like exciting news: Can a homomorphic circuit encryption scheme be used as a circuit obfuscator?

Unfortunately, the answer is no. The proof introduces a homomorphic encryption scheme, but its application context and the resulting security notions are quite different from what is required to deploy it in the service provider model. The function that performs the operation is in possession of the secret key used for encryption of operators and operands in the form of keyed decryption and encryption functions. This conflicts with everything that has been said about security so far: If Chantal is allowed to decrypt the operands, perform the computation on the plaintext, and then encrypt the result, nothing has been gained for Chantal being adversarial. But this is not how the proof uses the homomorphic encryption scheme. Security in our sense is not an assumption but *absence* of security for this *particular* scheme is the reason why circuit obfuscation cannot work in general.

However, the way in which a notion of homomorphic encryption is used in the analysis of code obfuscators suggests that the two are closely related. We will use this fact in the other direction by proving the non-existence of secure homomorphic encryption schemes based on the non-existence of code

obfuscators in Section 5.2.

3.4 Private Information Retrieval

The problem of secure outsourcing of database services on confidential data has close connections to the problems of *oblivious transfer (OT)* and *private information retrieval (PIR)*.

Oblivious transfer (NP00; NP01) is a very generic cryptographic primitive in which Chantal maintains an unencrypted bit sequence, and Murat sends queries to retrieve parts of it. A secure OT protocol ensures two things: (1) Chantal does not accidentally reveal more than the requested parts of the bit sequence, and (2) Murat does not reveal which part of the data he learns.[10] Private information retrieval (CGKS95; KO97; AF02) is a relaxation of OT in which Chantal is not interested in protecting her data any more: A trivially secure (but infeasible) PIR protocol is that there is only one query Murat can send, and he always obtains the entire database in response.

There are two interesting entanglements between these two: (a) coming from homomorphic database encryption as we have defined it in Definition 2.25 and with secure outsourcing of processing confidential information, the privacy requirement is turned around in that no longer is the data protected from Chantal, but the query; and (b) both problems are computationally hard.

Oblivious transfer is needed in a variety of cryptographic protocols. A simple motivational example is the online movie retailer. Chantal maintains a database of movies, and Murat retrieves his favorite selection by passing their offsets in the bit stream to Chantal (all movies have the same length, which is one DVD). For business reasons, Chantal does not want to deliver more movies than have been paid for. On the other hand, for privacy reasons Murat does not want to reveal which movies he is interested in.

In applications in which Murat has access to the entire database beforehand, such as when he outsources his own data to Chantal, the requirements of OT are too strong. We are therefore more concerned with PIR, which is the exact equivalent of homomorphic encryption with confidentiality required for queries, not relations.

Definition 3.10 (Querying Protocol)**.** *A* querying protocol *is a language $\mathcal{T}$ of tables and a language $\mathcal{Q}$ of queries such that $\forall q \in \mathcal{Q} : q : \mathcal{T} \to \mathcal{T}$.*

[10]Note that any participant in an OT protocol can be considered the adversary. This is different from symmetric encryption schemes, where the adversary is a dedicated participant that has no legitimate interests, but also from the service provider model in which only one of the seemingly legitimate parties is adversarial.

Querying protocols model a client-server context: A query $q \in \mathcal{Q}$ is sent by a client to a server to retrieve a part of some given table $t \in \mathcal{T}$ stored on that server.

Relational algebra is a special case of a querying protocol, relations being tables and queries being terms. But the concept is more general, since nothing is said about the structure of table or query language in this definition. In most of the literature on PIR, the tables in $\mathcal{T}$ are simply bit arrays, and $\mathcal{Q}$ is the range of those arrays. (This very simple setting already proves astonishingly challenging.)

Private information retrieval is defined on arbitrary querying protocols, so as we will see later it is directly applicable to relational algebra.

Definition 3.11 (Private Information Retrieval)**.** *Given a querying protocol* (Q, T)*. A pair of PPTM* $f : \mathcal{Q} \to \mathcal{Q}$*,* $g : \mathcal{T} \to \mathcal{T}$ *provides* private information retrieval (PIR) *iff*

1. *(information retrieval)* $\forall q \in \mathcal{Q}, t \in \mathcal{T} \,:\, g(f(q)(t)) = q(t)$
2. *(privacy) For any adversarial PPTMs* A, B

$$\Pr\left[\begin{array}{rcl} i & \leftarrow & RND^{\{0,1\}} \\ t \in \mathcal{T}, \bar{q_0}, \bar{q_1} \in \bar{\mathcal{Q}} & \leftarrow & B(t) \\ i^* & \leftarrow & A(t, \bar{q_0}, \bar{q_1}, \bar{f}(\bar{q_i})) \\ i = i^* & & \end{array}\right] \leq \frac{1}{2} + neg(|t|)$$

Chantal (the adversary) selects a table and two query batches to her liking, then Murat chooses one of the two query batches and "encrypts" them (although this term is not used in the context of private information retrieval, it gives a rather accurate account of what f does), and passes it on to Chantal. Now Chantal has to guess which query batch has been "encrypted".

By putting condition *1.* of Definition 3.11 and Figure 2.2 together, one can see that the transformations f, g look suspiciously homomorphic. In fact, it will turn out to be instructive to use PIR as a building block for constructing database encryption schemes. We will come back to this in Sections 4.3 where we use it to construct a secure encryption scheme.

PIR is hard. The trivial solution of passing the entire table as the result of any query to Murat is hard to improve upon. Poly-logarithmic complexity has been achieved in (KY01), but the hidden constants are likely to be rather large. Multiple servers that can not share information for a concerted attack can be used to obtain schemes with communication complexity $O(|t|^{1/3})$ (CGKS95). However, the assumption that servers do not collude is

arguably fragile, and the communication complexity and the administrative and legal overhead of this approach are rather large. An efficient solution using secure co-processors (see next Section) is presented by Asonov et al. in (AF02). We should stress that efficiency is essential when using PIR in the context of database outsourcing, so any scheme with an overhead above constant is likely to be too costly for our concerns. Further, the querying protocol for all these results is simple bit array lookup rather than full-fledged relational algebra.

3.5 Secure Co-Processors

So far, in addressing the problem of untrusted service providers we have focussed on cryptographic solutions. But there is a pragmatic solution that has proven quite useful in a few applications: Build a tamper-proof device into the service provider's host that does all the confidential part of the computation.

Such devices are called *secure co-processor (SCP)* and can be obtained from a number of vendors, e.g. IBM's 4758 (DLP$^+$01). SCPs serve the same end as ordinary processors, i.e. they have a small constant size memory and perform computations on the data that is made available to them via the system bus. They differ from ordinary processors in two ways:

- An SCP cannot be opened or tampered with without being destroyed. Therefore, the service provider, or any adversary with physical access to the running system containing an SCP, has no access to the SCP other than via the specified interface.

- A SCP has built-in crypto functionality to establish a secure channel between its secure perimeter and the client host. This is usually achieved by a pair of private and public server keys. The public key is delivered to the client in a certificate signed by the manufacturer, and the private key stays on the processor. This way, the service provider cannot replace the SCP by an ordinary one that merely claims to be secure. She does not have the secret key to sign its output, so the client would notice the replacement immediately. Analogously, a pair of private and public keys can be used by the client to ensure exclusive access to the SCP.

These two features, cryptographic protection of the communication between client and SCP, and tamper resistance, extend the perimeter of trust into the service provider's hardware.

The claim that tampering is impossible is not to be understood in an absolute sense. For unlimited financial resources and with sufficiently lax time constraints, there are a number of techniques like timing analysis, power analysis, or electromagnetic radiation analysis for inferring the secrets hidden in the chip from subtle physical characteristics (AK97; KJJ99). An adversary can even build a laboratory, steal the chip, and dissect it using microscopic layout analysis or physical on-chip wiretapping. But these threats do not invalidate the approach in general. They simply constitute an additional security requirement:

> A system relying on a SCP that defeats all attacks that cost less than x is secure if an adversary that can make a profit of at most x from attacking it.

This means that before a system can be designed to be secure using SCP, the potential profits of the adversary must be bounded, and the paramters of the SCP must be set according to this bound.

SCPs have been considered for content protection measures in standard PC hardware and entertainment electronics, but the PC industry has a much higher inertia than the market for server hardware, and there have been strong privacy concerns about the consumer giving up control over his laptop to the content providers, so progress has been slow on that front.

We conjecture that the most promising applications, although also the least widely distributed, are in hardware-enhanced homomorphic encryption protocols. SCPs have been used for several years in high-security web services like online-banking to manage cryptographic keys. The next step would be to examine more advanced problems, and find new algorithms that exploit the existence of a trusted processor in the realm of the adversary.

Private information retrieval, as introduced in the last Section, is (to our knowledge) the first problem that has been attacked in this way. In (SS01), an algorithm for PIR with simple array access using an SCP is proposed that has constant communication and client computation complexity.[11] However, it achieves these at the cost of linear complexity in the size of the array on the server side.

The algorithm is quite simple: The array is not modified in any way. Each query is received by the SCP in encrypted form, inaccessible to Chantal. The co-processor then retrieves each element in the array once, storing the one that is requested by the client, and discarding all the others. Once all elements have been touched once, the one to be returned is encrypted and sent to the client, again outside the reach of Chantal.

[11] An array is a database with one table that has only attributes `index` and `contents`.

A considerable improvement, rendering this basic approach useful for practical purposes, has been proposed in (AF02; Aso03). By encrypting the entire array and permuting the elements into an encrypted index in $O(n^2)$ server computation time, the server does not know which element is stored where in the array: Let s be a secret permutation only known to client and SCP. If the client requests array element i from the SCP, the SCP requests array element $s(i)$ from the server, decrypts it, and forwards it to the client. Since s is secret, the server learns nothing about i, or the element stored in the array under i.

Of course a PIR system should support multiple queries throughout its life time. This requires a little sophistication for the algorithm to remain secure. With the system implemented like this, Chantal could keep track of which element has been downloaded how many times, yielding opportunity for statistical attacks. Both problem and solution are similar to what surfaced in the discussion of $\mathrm{Fts}^{[\cdot]}$ in Section 3.2.2: In order to remain secure in the presence of multiple queries, the data must be re-shuffled in regular intervals, and a threshold for the desired security needs to be decided upon. For more details on the algorithm see (AF02).

Can SCPs help us in our goal to find homomorphic encryption schemes for relational algebra? A positive answer cannot be ruled out. In fact, a connection between PIR and a reduced flavor of relational algebra can be established that yields at least a theoretically sound Hom^{σ}-scheme (see Section 4.4). But there is good reason to be pessimistic.

First of all, Hom^{σ} is a query language much more powerful than the one used for the PIR-schema just outlined. What a co-processor can do is move expensive computations offline, i.e. have them done by the server side instead of burdening the network connection and client resources. However, the basic problem remains that secure realm covers only a small constant amount of memory, whereas the data to be queried is potentially huge. Hence, from a complexity point of view, the situation is not all that different: We still need to find a way to transform the data such that we can run transformed queries on transformed data, and the transformation still needs to effectively destroy all information for an adversarial observer. Pre-computation and trade-off schemes are conceivable, but with increasing query complexity, trade-off schemes are becoming less and less likely.

But most importantly, research for Hom^{σ}-schemes has not even matured to the point where we really know what we are looking for. We are still investigating preliminary security definitions, and most of those published so far have turned out to be unsatisfactory. In this thesis, we attempt to make a contribution to laying out the foundations of the field. Optimization of existing schemes can only come when such schemes (a) exist and (b) have

proven to satisfy a reasonable notion of security.

Cryptographic hardware may be facing a bright future, and be used ubiquitously in twenty or thirty years, but some important open questions have to be addressed before that.

3.6 Data Mining and Privacy

We will now have a very brief look at another related field of research that touches both cryptography and database theory. Consider an address database of all customers of an online retailer. In order for the retailer to process orders and ship goods, he needs to be able to retrieve the address matching a given customer identifier. On the other hand, for privacy reasons, he should not be able to mine the database for a list of all customers with a certain item in their purchase records.

Orthogonally, consider two banks that want to compare their transaction logs to make sure that every transaction between the two has been noted and processed by both ends and no money has been lost. The challenge is that the transaction logs are sensitive trade secrets, so neither bank wants to hand over theirs to the other.

3.6.1 Negative Databases

These are examples of databases that are not privately run by a single organization, but shared between different organizations that do not necessarily trust each other. The problem consists in allowing clients to answer some questions, e.g.

Is record X contained in the database? *(1)*

while ruling out others, e.g.

Give me all records in the database that have property C! *(2)*

For this specific example, an elegant solution has been proposed: Define a finite set $\mathcal{U}$ of all possible records, and store the complement set of all records in a table, i.e., the set that is *not* in the database. The result is called a *negative database* (EFH04; Esp05; EAH+06). Query *(1)* can still be answered by simply sending it to the negative database and negating the result, but query *(2)* has no straightforward equivalent other than subtracting the entire negative database from $\mathcal{U}$, and running a select operation on the result.

There are two challenges in implementing negative databases: How can we store a set that is almost as huge as $\mathcal{U}$ efficiently? And how can be make sure it is secure, i.e. there is no more efficient way to process queries like *(2)*?

The first one is addressed with a compression scheme that exploits the fact that the set is extremely dense. Almost all records are stored in the negative database if its complement has any realistic size. Automatically derived and updated regular expressions can be used as a compressed representation. The negative database thus has roughly the same size as its complement. To consider an extreme example, if k is the fixed length of all records, the negative of the empty database $\{\}$ is simply $\{\{0,1\}^k\}$. Existential queries are then mere pattern match operations.

The second question: How do we rule out privacy-breaching queries?, is addressed with reduction the $\mathcal{N}P$-hard problem of deciding whether a given boolean formula is satisfiable or not. This problem is known as SAT, and instances of the problem that are in fact hard to solve can be constructed efficiently (JMS05). By deriving database representations from hard-to-solve SAT instances, hard-to-invert negative databases can be constructed that represent any given complementing positive database.

3.6.2 Privacy-Preserving Data Mining

There is a number of related problems and solutions, directly expressible in relational algebra or not, that all have in common that a set of users wants to process some pool of information, while certain partitions of that pool, and certain computations on those partitions, need to be infeasible due to confidentiality goals. These problems have been subsumed under the label *privacy-preserving data mining (PPDM)* (AS00), and a lot of research has been done, identifying a huge number of problems and a few possible solutions, e.g. (DA00; EGS03; XT06).

How does PPDM relate to homomorphic encryption? As we have argued in 2.3.4, it is dangerous for a database user to commit to a class of harmless queries that the service provider can run on the data without violating confidentiality requirements. The risks that the released parts of the data, together with the adversary's context knowledge, allow for inferring (parts of the) confidential information are too hard to estimate and control. For all practical purposes, we need absolute privacy, i.e. the amount of leaked information must be 0.

PPDM is addressing an orthogonal class of problems in which the clients, despite their confidentiality concerns, have a need to share their data in order to accomplish a common goal. The default solution of the database outsourcing model: *do not outsource and keep your data locked up in your*

basement, does not work in PPDM.

The results in PPDM that have been published so far, or that are likely to be published in the foreseeable future, all involve not only a deterioration of privacy and increase in complexity of risk and threat management, but also considerable performance penalties. The benefits from pooling information in a controlled way may be worth those penalties. But database outsourcing does not provide these benefits, so both performance and rigorous security are essential for a cryptographic scheme to be of practical value.

Consequently, despite is intellectual elegance and practical relevance in related applications, PPDM in its current state does not provide us with satisfactory tools to build strong homomorphic encryption schemes. Although the structure of the data and query languages are very similar, the requirements differ considerably.

Chapter 4

Adversary Models and Analysis

In Section 3.2, we have presented the state of the art of security definitions for homomorphic encryption. In this chapter, we are going develop our contributions to this field.

4.1 Heuristic Security: Analysis of ATE

We will now see that the homomorphic database encryption scheme ATE[1] is insecure if we use even weak traditional notions of security (or KC-Security,[2] for that matter). We also develop a follow-up to the discussion in Section 2.3.4 on what constitutes a good security definition by demonstrating how a number of appealing, but not very suitable candidates fail to yield intuitive security for (variations of) ATE. An earlier version of this discussion has been published in (FG03).

Before we start, we introduce a rather odd security definition that we will need for both lower and upper security bounds in the following. δ-indistinguishability is defined in analogy to indistinguishability (see Definition 2.4), but security is already given if the adversary wins in at most a fraction $x \leq \delta$ of all cases.

Definition 4.1 (δ-Indistinguishability)**.** *Let* $0 \leq \delta \leq \frac{1}{2}$*. An encryption scheme* (G, E, D) *is* δ-indistinguishably secure (has δ-indistinguishability)

[1] See Section 3.1.1.
[2] See Definition 3.4.

iff for any adversarial PPTM A and any $m_0, m_1 \in \mathcal{M}$*,*

$$\Pr\left[\begin{array}{rcl} k & \leftarrow & G(1^N) \\ i & \leftarrow & RND^{\{0,1\}} \\ i^* & \leftarrow & A(m_0, m_1, E_k(m_i)) \\ i = i^* & & \end{array}\right] \leq \frac{1}{2} + neg(N) + \delta$$

Transforming any other security definition into its δ-analogon is straightforward. A crucial difference between the adversarial winning probability threshold δ and the security parameter N is that the former does not take security to be a function asymptotic in N. We can increase the security of the encryption scheme (usually this is done by increasing the size of the key), but this does not affect δ-indistinguishability non-negligibly.

To our knowledge, this is the first proposition of δ-security in the research literature. In most cases, 0-indistinguishability (which is just plain indistinguishability), i.e. a scheme that reduces adversarial success probability to negligibility, is what one wants. However, δ-indistinguishability for greater δ has at least two potential uses:

1. It is useful for applications in which something can be gained from rendering even only a fraction of all attacks useless. For instance, consider a database that contains intellectual property, and users that are only granted 30 access operations a day. If they can only corrupt the system in one out of thousand access operations, the number of security breaches will be reduced considerably, which may have a positive impact on legitimate sales and thus profits.

2. It can be used to establish lower security-bounds: Instead of the rather abstract and unintuitive result that some scheme is asymptotically insecure, we can often find large δ values for which the system is always insecure, no matter what the key size, thus giving a measure to compare the severity of different security problems.

In the business of database service outsourcing, security is an all-or-nothing thing: The risk of revealing data to adversaries if the service is run in-house at the client site is minimal, and must not be increased by an outsourcing move. Hence, although potentially interesting in the long term, for our purposes, δ-security is mostly good for establishing explicit lower security bounds.

ATE is insecure in almost every applicable definition contained in this thesis. However, this does not follow directly from our infeasibility theorems (see Sections 5.1 and 5.2), because Chantal never gains access to the

unaggregated information necessary to compute an exact query result. Instead, Chantal can only compute a conservative approximation (conservative meaning "no false negatives") of the result, and Murat needs to participate a round of post-processing himself. The pattern in Figure 2.2 is therefore subtly violated: For exact homomorphic encryption schemes, Chantal is responsible for the right half of the picture (and capable of handling it all by herself), and Murat is responsible for the left part. For ATE, D_k has the form $\phi \circ \tilde{D}_k$, where $\tilde{D}_k$ is a decryption function that only inverts E_k *approximately*, and $\phi \in \Phi$ is the plaintext query for which the encrypted result contains a conservative approximation.

As it is not a direct implication of our proof sketch, we now demonstrate that ATE is not secure, and give a sense of the severeness of the threat. We first expose order revelations through range queries, then small attribute domains, and then the impact of aggregation on security in general.

As an afterthought, the last point, an argument that aggregation does not provide security in any reliable sense, can be considered a third application of the notion of δ-security, which is closely related, but not identical, to the second one. It goes roughly like this: The aggregation before encryption turns out to have a multiplicative effect on δ. Therefore, if a scheme is is not δ-secure without aggregation for some $\delta > 0$, then for any aggregating enhancement of that scheme there is some $\delta' > 0$ such that it is not δ'-secure.

4.1.1 Order Revelation through Range Queries

Assuming the database schema is confidential in itself, Chantal can get considerable insight in the number and types of attributes in the plaintext database relations. In particular, each soft-encrypted attribute reveals at least:

- The fact that there is an attribute, and how many bits are needed at most to store an attribute value.
- The number of partitions of a given attribute. (In principle, there may be empty partitions, but the configurations on which ATE has been tested set number of partitions to very small numbers, and the figures suggest that this is in fact necessary for performance reasons.)
- The rough size of a given partition value (if $h^{\mathbf{ord}}$ is used).
- Which rows have attribute that are similar (if $h^{\mathbf{random}}$ is used).
- If the database service is embedded in an application service known to (or run by) Chantal, then she is likely to guess the semantics of

the attributes (e.g. Employee Name, Type of Position, Salary, ...), and even if that does not reveal the actual values for given encrypted rows, it makes guessing and certain types of cryptographic attacks much easier.

We shortly sketch an example for an attack revealing the order of plaintext values despite pseudo-random soft encryption $h^{\mathbf{r}\text{random}}$, and then give a formal account of it.

Example 4.1 (A Simple Attack Against ATE)**.** *Consider the example in Section 3.1.1 again. The conditions:*

$$\mathtt{salary} \geq 55{,}000, \mathtt{salary} \leq 65{,}000$$

which do not trivially reveal anything about $h_{\mathtt{salary}}$ (which is pseudo-random), are transformed into

$$\mathtt{salary} \in I_1 = \{0, 2, 3\}, \mathtt{salary} \in I_2 = \{0, 1, 2, 4\}$$

(Using Equation 3.2.) Each of these two set member tests turns out to tell a great deal about the secret permutation, assuming that Chantal guesses correctly that the expressions have been transformed from comparison operations: Chantal can derive from I_1 that partitions $0, 2, 3$ are all smaller (or greater) than partitions $h_{\mathtt{salary}}(\mathtt{salary}) - I_1 = \{1, 4\}$.[3] *Similarly, she can derive from I_2 that partitions $0, 1, 2, 4$ are smaller (or greater) than 3. Guessing $\geq$ for I_1 and $\leq$ for I_2 correctly, she obtains:*

$$\begin{aligned} &\{\, \rho_{\mathtt{range}}(a) \geq \rho_{\mathtt{range}}(b) \mid a \in \{0, 2, 3\}, b \in \{1, 4\} \,\} \\ \wedge &\{\, \rho_{\mathtt{range}}(a) \geq \rho_{\mathtt{range}}(b) \mid a \in \{3\}, b \in \{0, 1, 2, 4\} \,\} \end{aligned}$$

which enables her to define:

$$\begin{aligned} I_1^* &= \{1, 4\} \\ I_2^* &= \{0, 2\} \\ I_3^* &= \{3\} \end{aligned}$$

And know that all partitions in I_i^* have smaller plaintext partitions than any partition in I_j^* for all $i < j$.

In other words, the order of partitions hidden by $h^{\mathtt{random}}$ has been revealed partially, but to a great extent, by a single range condition and two correctly guessed bits, without even looking at the encrypted data!

[3] As usual for denoting the set of all values of a function for some domain, we use $h_{\mathtt{salary}}(\mathtt{salary})$ to refer to the set of all soft-encrypted salary attribute.

In this example, Chantal had to guess two bits: The decisions where to use which of $\leq, \geq$, because the interval does not necessarily reveal towards which end of the attribute domain it is open. We now demonstrate that there is a way for Chantal to recover all of a set of operators up to one bit which she has to guess, and we make a first attempt at gauging the strength of this attack by using the newly introduced notion of δ-indistinguishability.

Recall that a comparative query condition of the form $C = A < v$ is soft-encrypted into $h(C) = h(A) \in I$, where $I = \{h_i(v') \mid v' < v\} \subseteq h(A)$ is the set of all partitions that contain at least one value smaller than v (see Figure 3.1). We now estimate the information leakage from one such condition.

If we had only one comparison operator $<$, each $I \subseteq h(A)$ would provide Chantal with the constraint

$$\forall a, b \in A : h(a) \in I \wedge h(b) \notin I \Rightarrow a < b$$

What remains to be shown is that Chantal can recover this information up to one bit despite the fact that she is not told which comparison operators are used.

First, assume that we only have $<, >$ and consider this example: Given 6 partitions with encrypted partitions $0, ..., 6$. If Chantal has seen two sets $\{0, 1, 2, 3\}, \{0, 1, 2\}$ and guesses $\{0, 1, 2, 3\} < \{4, 5\}$ and $\{0, 1, 2\} > \{3, 4, 5\}$, then she can derive an inconsistency: Both $0 < 4$ and $0 > 4$ are implications of these guesses. Hence, if the first operator is $<$, the second must be $<$ also.

More generally speaking, for each pair I, I' extracted from a valid pair of inequality conditions on the same attribute, we can find a pair of values that tells Chantal whether the two inequalities are different or not:

$$\exists x \in I, x' \in h(A) - I : x \in I' \wedge x' \in h(A) - I'$$

This means that in a larger number of inequalities, we only need to guess once whether an operator is $<$ or $>$, and the decryptions of the other operators follow. This gives Chantal the one bit of uncertainty about her success.

As for $\leq, \geq$, let $v' = v + 1$ and $Q_A(v') = Q_A(v) + 1$. (If the attribute type from which v, v' are picked is not numeric, use the bit representation.) Then the two queries $A < v$ and $A \leq v'$ are indistinguishable from Chantal's perspective, and they both provide her with a precise split in $h(A)$. Further, observe that Chantal can at best reveal the order of the partitions — the actual contents never leaves Murat's part of the system. Since Chantal is not interested in the exact values but in revealing the partition order, and assuming that the range bounds are suitably distributed, the attack is equally effective for operators $<, >, \leq, \geq$ and for operators $<, >$ only.

As a preliminary attempt to summarize the conclusion from this example attack, we formulate the following

Conjecture 4.1. *Assume A is soft encrypted with some $h^{\mathbf{r}random}$ (order revelation trivially holds for $h^{\mathbf{o}rd}$). If $|A| > 1$, then security of ATE deteriorates rapidly with the number of distinct range queries of the form $\sigma_{A\cdot v}$ (where $\cdot \in \{<, >\}$). After $|A|$ distinct range queries, the order of $Q_A(A)$ is revealed but for one bit.*

The problem with finding a proof is that we have not found a satisfactory security definition for homomorphic encryption with relational algebra yet. We will make up for this in the following, and establish more rigorous and general results after that. These general results will be applicable to this analysis of ATE.

The most straightforward fix of order revelation would be to not allow for the comparison operators $<$, $\leq$, $>$, $\geq$ in query conditions, sacrificing expressive power for security. However, this would only prevent order revelation by the specific attack technique outlined here, while there may be more sophisticated attacks that still work and can be found with little more dedication. Also, this does not help against the more severe flaws we are going to explore below.

Is there anything else that can be done, short of reducing expressive power of the application of our scheme? Noise queries, or queries that have an invalid form and thereby mislead Chantal when she takes them for legitimate range queries, are not an answer (see Section 4.1.4). PIR would be, but our infeasibility results (see Chapter 5) suggest that there are no schemes that are flexible enough to allow for homomorphic range query encryption.

So we seem to have tracked down a principle privacy limitation here: Prompted accordingly, Chantal must be capable of returning all rows with values $h(a)$ such that $a < v$ and not too much noise, so she is bound to learn the answer, even if she does not know v. But gathering answers for many vs, the order becomes necessarily apparent to her. In conclusion, $h^{\mathbf{r}\mathrm{random}}$ appears to be no stronger than $h^{\mathbf{o}\mathrm{rd}}$ in the presence of range queries.

4.1.2 Small Attribute Domain Size

One of the primary purposes of aggregating plaintexts into partitions is to make it hard to decide equality (or inequality) of two attributes given just their soft encryptions. We now demonstrate that if the domain size of an attribute is very small, this technique is ineffective.

To consider an extreme example, a boolean attribute type has only two partitionings and both are trivial: $\{\{1, 0\}, \{\}\}$ and $\{\{1\}, \{0\}\}$. The former is as efficient as no soft encryption at all, so the only option is to create one partition per plaintext value. Since soft encryption needs to be deterministic,

this has the effect that the set of all values of a boolean attribute in a relation are known to Chantal except for the last bit (the one that decides which is 1 and which is 0). Taking into consideration that Chantal should be granted all but one bit of knowledge of the plaintext, soft encryption for boolean attributes is clearly and fatally insecure. Similar arguments hold for attributes with marginally more than two values, although the problem gets less severe with growing domains.

In order to obfuscate values of this type effectively and avoid attacks that decide $a =^{?} b$ given only their soft encryptions $h(a), h(b)$, more than one partition can be assigned to each partition ID. As a range of plaintext salary values could be mapped on one partition, now one partition can be mapped on a range of partitions, which, if encrypted, can not trivially be associated with the same partition.

A probabilistic partitioning

$$Q_A^{(\tau)} : A \rightarrow^{(\tau)} N$$

has more than one partition for each partition, and $Q_A^{(\tau)}(a)$ is picked uniformly at random from a set of τ partitions assigned to a. If $\tau = 1$, the partitioning is deterministic. Note that Figure 3.1 must be modified slightly. $h(v)$ is not a unique value any more, but one of a set of values. Instead of testing for equality, we need to test for set membership.

This approach has a fatal weakness, however: Every select with equality condition causes Murat to consistently request one of a number of unique and disjoint sets of ciphertext partitions (two such disjoint sets total exist in the case of booleans), so Chantal can conclude that members of those groups represent the same plaintext partition. We now outline an algorithm to reveal these partition groupings.

Let A be any attribute with "small" domain size and $h_A = H \circ Q_A^{(\tau)}$ probabilistic.[4] If h_A is deterministic ($\tau = 1$), the attack is trivial, as the only useful partitioning is a one-to-one mapping from plaintext values to partitions. For probabilistic h_A ($\tau > 1$), Chantal needs to be more intelligent. The idea is to scan the available queries for *clues* to the partition sets, and to collect all clues in a matrix

$$Clue : h(A) \times h(A)$$

Whenever a clue is found that $x, y \in h(a)$ for some $x, y \in h(A), a \in A$, increase $Clue_{x,y}$; if a clue is found that $x \in h(a), y \notin h(a)$, decrease $Clue_{x,y}$. Note that *Clue* is symmetric, or triangular.

[4] There is no need to specify "small" any further here. Security will simply be a function of $|A|$.

Algorithm 3: Feeding the *Clue* matrix.

```
/* initialization */
begin
    Initialize all cells with 0
end
/* range heuristic */
begin
    for every (sub-)condition of the form h(A) ∈ I do
        for x, y ∈ I do
            Clue_{x,y} := Clue_{x,y} + 1
        for x ∈ I, y ∉ I do
            Clue_{x,y} := Clue_{x,y} − 1
end
/* exact select heuristic (A = v) */
begin
    if |I| = τ then
        /* all elements in I are ciphertexts of the same plaintext. */
        for x ∈ I, y ∈ h(A) do
            if y ∈ I then
                Clue_{x,y} = +∞
            else
                Clue_{x,y} = −∞
end
```

The attack consists of two steps: Feeding the *Clue* matrix (see Algorithm 3), and consulting it for deciding plaintext equality for two soft encrypted ciphertexts (see Algorithm 4).

Since it is meaningless to retrieve only an unspecified subset of all tuples in a relation that satisfy a given query condition, $|I|$ is always divisible by τ. The smaller $|I|$, the greater the odds that two tuples have identical values if their soft encryptions are both in I. In the exact select heuristics, Chantal can make a definite statement and not investigate any further: All of I encrypts the same plaintext attribute. If $|I|$ grows, the range heuristics kicks in and computes probabilities that depend on the number of clues encountered so far. (There is a more accurate algorithm that increments $Clue_{x,y}$ by $\frac{\tau}{|I|}$, taking the probabilities into account that change with $|I|$, but for our purposes the above version will suffice.)

Both update steps in Algorithm 3 are heuristic, so in some cases, Al-

Algorithm 4: Consulting *Clue* for equality decision problems.

Input: $x, y \in h(A)$
if $\text{Clue}_{x,y} > 0$ **then**
  output *identical*
else
  output *different*

gorithm 4 does not produce accurate results. However, if Murat does not submit noise queries, we obtain the following

Lemma 4.2. *Let ATE with a deterministic partitioning operation Q_A is δ-insecure for some δ.*[5] *Then ATE with probabilistic partitioning $Q_A^{(\tau)}$ is δ-insecure after $|A|$ distinct exact select queries,*

It is easy to see that it is true independent of the actual security definition: After $|A|$ distinct exact selects, $Q_A^{(\tau)}$ has been completely determined, i.e. all equivalence classes of its domain have been determined by the adversary, so no additional security can be provided by probabilistic partitioning.

4.1.3 On the Impact of Aggregation

In this section, we will consider the order over the plaintext domain A as a source of information for Chantal. We will only consider $h^{\mathbf{r}\text{andom}}$ here, but a similar argument trivially holds for $h^{\mathbf{o}\text{rd}}$. Assume that Q_A is deterministic and that all partitions in the partitioning Q_A have the same size:

$$\exists Z \,:\, \forall x \in Q_A(A) \,:\, |\{a \in A \mid Q_A(a) = x\}| = Z$$

Further, we define a predicate $p : A \to \{0, 1\}$ that Chantal wants to compute from the ciphertext alone:

$$p(a, b) = \begin{cases} 1 & \text{if } |a - b| < Z \\ 0 & \text{otherwise} \end{cases}$$

This may be interesting for salary attributes, or any other attribute for which any one or more of the following conditions hold:

1. Partial information on the plaintext is already available.

[5] Note that this is not an assumption, but a fact. Even if ATE was perfectly secure, it was still $\frac{1}{2}$-insecure.

2. Approximations of plaintext values are of some value already. (An approximate salary is more likely to be of value than an approximate customer id.)

3. Classification of encrypted tuples is more interesting than decryption and recovering actual values. (If a boolean attribute states, say, HIV test results, a distribution of 1/1 is different from 38/1, even if the adversary does not know in the latter case which boolean value has which of the two relative frequencies. Further, one of the two alternatives can often be ruled out by context knowledge, such as that there are more negative outcomes of HIV tests than positive ones.)

We can now derive a very low upper bound for the protection of ATE-encrypted data against revelation of p:

Lemma 4.3. *Given a relation with Q_A, Z as introduced above. For*

$$\delta = 1 - \frac{Z(|A| - Z)}{2|A|^2}$$

p can be recovered from an encrypted relation with probability δ. Furthermore, for all $Z, |A|$, $\delta \geq \frac{7}{8}$.

Proof. The attack is very simple: When faced with a challenge $h(a), h(b) \in h(A)$, Chantal simply returns 1 if $h(a) = h(b)$, and 0 otherwise. Let α the Chantal's answer.

The success probability is 1 in the case of $h(a) = h(b)$, as the maximum distance of two members of one partition is $Z-1$. If a, b are in two neighboring partitions, and a is the upper half of the lower partition and b is in the lower half of the upper partition (or vice versa), then the adversary loses, since she returns 0 although $|a - b| < Z$. Otherwise, $|a - b| \geq Z$, and the adversary correctly returns 0.

Hence, the adversary loses if and only if a, b are inhabiting two neighboring partitions with a distance smaller than Z. The probability of this event, i.e., the probability with which the attack fails, is computed as follows:

$|A|/Z$ is the number of partitions, so $Z/(2|A|)$ is the probability of a plaintext lying in any specific partition half. For each partition except the last, the probability that a lies in its upper half and b in the upper neighboring partition's lower half is $(Z/(2|A|))^2$, and as the condition is symmetric (b may be in the smaller partition and b in the upper), we need to double this probability. There are $|A|/Z - 1$ occurrences of this event (each partition except the last one has an upper neighbor). Hence, we need to multiply

again and obtain as the success probability of the attack

$$\begin{aligned}\delta &:= \Pr[\,\alpha = p(a,b)\,] \\ &= 1 - (|A|/Z - 1) \cdot 2(Z/(2|A|))^2 \\ &= 1 - (\frac{|A|}{Z} - 1)\frac{Z^2}{2|A|^2} \\ &= 1 - \frac{1}{2}(\frac{Z}{|A|} - \frac{Z^2}{|A|^2}) \\ &= 1 - \frac{Z(|A| - Z)}{2|A|^2}\end{aligned}$$

For $Z = \frac{|A|}{2}$ (we need at least two partitions, otherwise there is no risk, but also no virtue in soft encryption), we have

$$\delta = 1 - \frac{\frac{|A|}{2}(|A| - \frac{|A|}{2})}{2|A|^2} = \frac{7}{8}$$

δ grows monotonously with decreasing Z. This concludes the proof. □

Conjecture 4.4. *If the relation contains a soft encrypted attribute A with Q_A, Z as in Lemma 4.3, then ATE is not $\frac{\delta}{2}$-secure for $\delta = \frac{7}{8}$.*

Intuitively, this means that ATE is inappropriate for keeping a rough similarity of attribute values private. The only thing that Alice can achieve is enforcing a "level of roughness" that suits her by choosing a large enough partition size Z. Note that this problem arises from the partitioning alone, so better query *or* data encryption cannot be used to overcome it.[6]

As an interesting aside concerning e.g. equi-joins, even if an feasible trade-off between performance and security can be found, query conditions of the form $A_i = A_j$ can get quite bulky when encrypted. Since all the intersecting partition pairs need to be enumerated, the size of the encrypted query is

$$O(\max\{|Q_{A_i}|, |Q_{A_j}|\}$$

for given partitionings Q_{A_i} and Q_{A_j}, i.e., linear in the number of partitions. The consequences of this seem paradoxical at first: In contrast to the performance-privacy tradeoff observed elsewhere, increasing the number of partitions not only reduces the effectiveness and security of the encryption scheme, but also has a negative impact on performance in the presence of conditions of the form $A_i = A_j$.

[6]Keeping the partitioning secret is a bad idea. The partitioning is part of the scheme, not part of the key. Choices affect security in non-trivial ways, so they have to be made available for scrutiny and auditing to a wide audience.

4.1.4 Noise

Recall the clue matrix attack algorithms in Section 4.1.2 once more. One possibility to prevent this sort of attacks is to inject noise into exact select queries that trick Chantal into false conclusions about the contents of soft encrypted attribute values. (Similar strategies as proposed here for clue matrices can be deployed against other forms of attacks outlined above, with similar limitations.)

ATE with noise queries: A query of the form $\sigma_{A=a}(R)$ is split up into ρ different queries as follows:

1. Let $I = \{x_0, \dots\}$ be the domain of $h_A(a)$, i.e.

$$h_A(A = a) \;=\; A \in I$$

2. Let $I_i = \{x_i, \chi_{i,1}, ..., \chi_{i,\tau-1}\}, i \in [1, \rho]$, where $\chi_{i,j}$ are uniformly distributed (pseudo-)random values.

3. Instead of the noise-free

$$\sigma_{A \in I}(R)$$

submit the queries

$$\sigma_{A \in I_i}(R) \quad \forall i \in [1, \rho]$$

However, the additional effort that Murat has to put into computation and communication is too high, even under the (rather strong) assumption that Chantal has no way of heuristically connecting queries that occur in the same burst and recovering the original (noise-free) query from the burst.

Lemma 4.5. *Given a notion of security that states that the clue matrix attack outlined in Section 4.1.2 is ineffective.*

Consider a database schema with a soft encrypted attribute A. Let ATE without probabilistic soft encryption and noise be δ-insecure for some δ. Then there exists a function

$$f : [0, 1] \times [1, |A|] \times [1, |A|] \to [0, 1]$$

that maps values of δ, τ, ρ to new values of δ such that

1. *$f(\delta, |A|, |A|) = 0$.*
2. *$f(\delta, 1, 1) = \delta$.*
3. *f is monotonously growing with τ and ρ: If $\tau' > \tau$ and $\rho' > \rho$, then $f(\delta, \tau', \rho) > f(\delta, \tau, \rho)$ and $f(\delta, \tau, \rho') > f(\delta, \tau, \rho)$.*

4. *ATE with probabilistic soft encryption with randomness τ and noise level ρ is $f(\delta, \tau, \rho)$-insecure.*

Proof. Queries in ATE with randomness and noise levels both $|A|$ does in fact not reveal any clue that could be used for the attack: Every query explodes into $|A|$ queries for the entire domain of $|A|$, so the encrypted condition contains no information about the plaintext condition. On the other hand, if $\rho = \tau = 1$, the probabilistic noisy enhancement of ATE collapses to the original scheme. The observation that f grows monotonously in τ and ρ is straightforward. □

In this result, we are not interested in other potential attacks. We merely want to assess the effectiveness of noise as a countermeasure against this particular attack. Unfortunately, the additional security comes with a super-linear increase in communication costs: If $c(\rho, \tau)$ is the bit size of the set of queries a non-probabilistic noise-less query is compiled into, then $c \in O(\tau\rho)$. This is prohibitively expensive.

This is in fact the primary benefit of the notion of δ-security: It establishes that there is no easy way around asymptotic security, i.e. the notion that linear growth in the security parameter and the costs for the legitimate user of a cryptographic scheme should result in super-polynomial growth of the cost for a successful attack. Any attempt to increase the security polynomially by making it polynomially more expensive results in infeasible cryptography for all practical purposes.

To conclude this section, we have found a number of fatal security issues in ATE and shown that most options to fix them, or at least alleviate their impact, are either ineffective, or overly costly, or both. The generality of our arguments suggests that the idea of security by aggregation is fundamentally flawed and should not be pursued any further.

4.2 Partial Security: Analysis of Full-Text Search

In Section 4.3.2, we will develop a homomorphic encryption scheme for databases based on homomorphic full-text search encryption schemes, and we will establish two relaxed security properties. In this section, we assess the security of $\mathrm{Fts}^{[\cdot]}$ and a new variant called $\mathrm{Fts}^{\{\cdot\}}$, reveal an important gap in its security proof, and provide a new security definition that can be satisfied nevertheless.

4.2.1 The Problem

In the brief discussion of the security of Fts$^{[\cdot]}$ in Section 3.2.2, it became apparent that there is a problem with queries: Each time an encrypted query γ_w is sent, Chantal can decrypt all occurrences of the word w in the document base down to $E^{\text{pre}}(w)$. For the scheme to work at all, E^{pre} needs to be non-probabilistic, i.e. the same input always yields the same output, no matter what the location of w in the document or document base. But non-probabilistic block-wise encryption of plaintexts breaks indistinguishability.

There are two straightforward attacks that demonstrate this. The first makes use of unsafe queries to identify the encrypted result by its size.

Example 4.2 (Attack using Result Size)**.** *Let* $U_0 = U_1 = \{\{rosebud\}\}$*,* $q_0 = \gamma_{rosebud}$*, and* $q_1 = \gamma_{winter}$*. The size of the encrypted result fully reveals* i*.*

The second is stronger in the sense that it does not rely on the significance of the size of the result, but identifies it by the location of the match.

Example 4.3 (Attack using Match Positions)**.** *Let* $U_0 = \{(rosebud, spring)\}$*,* $U_1 = \{(spring, rosebud)\}$*, and* $q_0 = q_1 = \gamma_{rosebud}$*. Then the position of the match, which is revealed to Chantal by the algorithm, reveals* i*.*

While the first problem is intuitively hard to fix without having fixed output size, for the second there is a solution: Assume the position of words is destroyed during encryption, or speaking more abstractly: Define documents to be sets, not sequences of words.

Definition 4.2 (Homomorphic Encryption for Search in Sets)**.** *Let* l *be the global* word length*, and let* $W = \{\{0,1\}^l\}$ *be the set of all* words *of that length. A* homomorphic encryption scheme for search in sets *is a homomorphic encryption scheme* (G, E, E^*, D) *as defined in Definition 2.24, where* $\mathcal{M} = \{\{W^*\}^*\}$ *is the set of all sets* (archives) *of word sets* (documents)*, and* $\Phi = \{\gamma_w | w \in W\}$ *is the set of full-text search queries.* γ_w *yields all documents containing* w*: If* $\gamma_w \in \Phi$ *and* $m = \{m_0, m_1, \dots\} \in \mathcal{M}$*, then* $\gamma_w(m) = \{m_i \in m | m_i = \{w_0, w_1, \dots\} \wedge \exists j. w_j = w\}$*.*

A homomorphic encryption scheme on set documents can be derived from Fts$^{[\cdot]}$: Before encrypting a document $m_i = \{w_0, w_1, \dots\}$, store the words in a sequence in arbitrary order (the canonical way to do this is to merely keep the words in the order in which they are already stored on the computer that runs the encryption algorithm), and apply a (pseudo-)random permutation κ_{m_i} to that sequence. Since documents are sets, the destruction of any order by $\kappa(m_i)$ does not affect the contents. Since the order appears random to

the adversary, implementation details about set representation cannot be exploited for an attack.

We call the resulting set document encryption scheme $\mathrm{Fts}^{\{\cdot\}}$. It is easy to see that the attack in Example 4.3 fails against $\mathrm{Fts}^{\{\cdot\}}$. Unfortunately, the attack in Example 4.2 is still effective. We will consider countermeasures in our construction of secure $\mathrm{Hom}^{=/\sigma}$-schemes in Section 4.3.

The security of both $\mathrm{Fts}^{[\cdot]}$ and $\mathrm{Fts}^{\{\cdot\}}$ is so weak for a deeper reason that is crucial to understand: *The rule that same plaintexts yield same ciphertexts.* This touches on a basic notion called *mode of operation* that is used to reason about symmetric encryption schemes. Before we proceed to establish a limited level of security for these schemes, we will now explain this in more depth.

Since symmetric encryption schemes are often circuits with a fixed number of input and output bits, strategies are needed to build encryption schemes from them that work on plaintexts of arbitrary length. These strategies are called *modes (of operation of a block cipher).*[7]

The simplest mode of operation is called *ECB (Electronic Code Book)* and works as follows: Given a block cipher (G, E, D) a plaintext $(m_0, \ldots, m_n)$ of arbitrary length divisible by the block size,[8] and encrypt each block individually:

$$\begin{aligned}\mathrm{ECB}_k(E, (m_0, \ldots, m_n)) &= (E_k(m_0), \ldots, E_k(m_n))\\ \mathrm{ECB}_k(D, (c_0, \ldots, c_n)) &= (D_k(c_0), \ldots, D_k(c_n))\end{aligned}$$

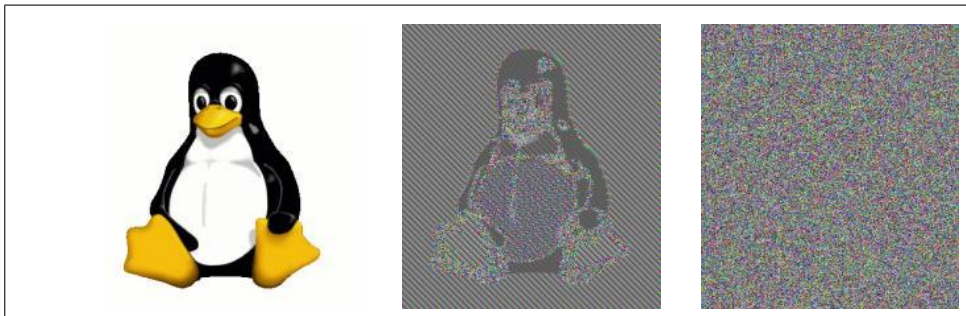
A more secure variant called *CBC (Cipher Block Chaining)* seeds each plaintext block with the preceding ciphertext block in order to destroy homomorphicity with respect to block equality:

$$\begin{aligned}&\mathrm{CBC}_k(E, (m_0, m_1, \ldots, m_n)) =\\ &\qquad (r, E_k(m_0 \oplus r), E_k(m_1 \oplus E_k(m_0)), \ldots, E_k(m_n \oplus E_k(m_{n-1})))\\ &\mathrm{CBC}_k(D, (r, c_0, \ldots, c_n)) =\\ &\qquad (D_k(c_0) \oplus r, D_k(c_1) \oplus c_0, \ldots, D_k(c_n) \oplus c_{n-1})\end{aligned}$$

CBC is probabilistic and precedes each ciphertext block sequence with an initial random *initialization vector* r. If the plaintexts with identical first blocks $m_0 = m_0'$ are encrypted, this makes sure that the corresponding encryptions $c_0 \neq c_0'$ (probabilistically). It is easy to see that this is a necessary

[7] Block ciphers are introduced in Definition 2.18. Since we will not need modes of operation anywhere outside this section, we only introduce them now.

[8] Modes of operation usually come with a *padding scheme* with which the plaintext can be extended to a length divisible by the block size in a way that can be reversed during decryption.



(a) Original (b) ECB (c) Other modes

Figure 4.1: ECB is weak: Even if every block is encrypted using a secure scheme, the full picture remains visible because identical plaintext blocks yield identical ciphertext blocks. (Source: http://en.wikipedia.org/)

condition for indistinguishability of encryptions. Newer modes provide integrity and confidentiality at the same time and are thus more robust against advanced active attacks (Jut00; Rog00; Hal01; BKN02).

Figure 4.1 gives an impressive demonstration of the weakness of ECB, and how CBC can overcome this weakness: While (c) apparently does not contain any structure at all, in (b), the original unencrypted image (a) is clearly recognizable.

Unfortunately, $(G^{\text{pre}}, E^{\text{pre}}, D^{\text{pre}})$ in our full-text search scheme of choice cannot implement any mode other than ECB, as those all have the (usually desirable) property that encryption of the same block in two different locations in the plaintext block sequence yields two different ciphertext blocks. But two occurrences of the same word w need to result in identical pre-encryptions $E^{\text{pre}}(w)$. (If not, which one would be the encryption of w provided with the query?) This is the reason why a stream cipher needs to properly encrypt the already pre-encrypted words.

Trade-offs

In (SWP00), the authors propose to fix this problem by periodic re-encryptions of the entire database: Every time "too many" pre-encrypted words have been unwrapped from the stream cipher layer of encryption by Chantal, the entire document base is downloaded, re-encrypted, and uploaded again. Note that even with a sophisticated key management, it may be impossible in general to do this incrementally for those documents that have been matched only. Such a scheme would need to prevent Chantal from annotating a newly encrypted document with the information she already learned before re-encryption, which seems ambitious to say the least.

This is similar to the trade-off tricks discussed in Section 3.5, but there are three obstacles to consider:

- Even if re-encryption is done at high frequency, it will not resurrect security in any of the definitions in which it was broken without it. (This flaw is also immanent in the trade-offs for secure co-processors.)

- The client has to contribute considerable communication, computation, and storage work. Since one of the most prominent motivations for database outsourcing is the more efficient use of these resources, this disadvantage can be quite prohibitive. In particular, mobile applications are ruled out by this restriction, or require a trusted stationary host with more resources than a mobile device that supports the mobile user. To an extent, however, this problem can be addressed with trusted hardware (see Section 3.5.)

- The computing resources that the server needs for performing its task are considerably higher than the resources needed for the alternative local solution without any outsourcing. This puts a weight on the hardware prices needed per customer, and makes outsourcing less profitable and competitive.

4.2.2 Partial Security

The weakness of $\mathrm{Fts}^{\{\cdot\}}$ and $\mathrm{Fts}^{[\cdot]}$ suggests a relaxed security definition that captures and dismisses the attacks we have outlined above. In particular, we want to allow for the adversary to use the query result sizes in her attack, and only rule out attacks that recover more information than can be recovered from those sizes. This idea is expressed in the following

Definition 4.3 (Full-Text Search Indistinguishability up to Result Size)**.** *A homomorphic full-text search encryption scheme* (G, E, E^*, D) *is indistinguishably secure up to result size if for any adversarial PPTM* A *there is a corresponding adversarial algorithm* A' *such that for any set of document sets* $\mathcal{U}$*, any two document sets* $U_0, U_1 \in \mathcal{U}$ *with* $|U_0| = |U_1|$*, and any two query*

sequences $\bar{q_0}, \bar{q_1}$ *with* $|\bar{q_0} = \bar{q_1}|$*:*

$$
\begin{aligned}
&|\Pr\left[\begin{array}{rcl} i & \leftarrow & RND^{\{0,1\}} \\ i^* & \leftarrow & A(\mathcal{U}, U_0, U_1, \bar{q_0}, \bar{q_1}, E_k(U_i), \bar{E_k^*}(\bar{q_i})) \\ i = i^* & & \end{array}\right] \\
&- \Pr\left[\begin{array}{rcl} i & \leftarrow & RND^{\{0,1\}} \\ i^* & \leftarrow & A'(\mathcal{U}, U_0, U_1, \bar{q_0}, \bar{q_1}, E_k(U_i), |E_k^*(q_{i,0})(E_k(U_i))|, \dots) \\ i = i^* & & \end{array}\right]| \\
&\leq neg(N)
\end{aligned}
$$

The following conjecture states that $\text{Fts}^{\{\cdot\}}$ is secure beyond the notion that has been established in Section 3.2.2.

Conjecture 4.6. *$Fts^{\{\cdot\}}$ is secure in the sense of Definition 4.3.*

4.2.3 A Note on Multi-Message Security

In Section 3.2.2, we have mentioned that there are issues with key management and multi-message security. We will now elaborate on this. As we have hinted, the argument applies easily to the Hom^σ-schemes derived from $\text{Fts}^{\{\cdot\}}$.

Have another look at Figure 3.3. Note that so far, the PRG $\langle\langle \text{R} \rangle\rangle$ takes a key k that is constant over all documents. However, this allows for trivial distinguishing attacks: If every document is encrypted using the same PRG stream, then two identical documents will yield two identical encryptions. But since the homomorphism is about document sets and not documents, multi-message (i.e., multi-document) security is necessary (SW05).

If each document has a different key $k_{(\text{document})}$, then multi-message indistinguishability of the stream cipher is preserved, and the search operation is still computable by Chantal alone. From her perspective, the S_i cannot be distinguished from true randomness, but further knowledge about the S_i is not required for performing her task. Given the pre-encrypted search word, she merely has to check whether some potential S_i generates the right suffix $F(S_i)$.

Different k for each document require us to maintain a mapping from documents (or some sort of document identifiers) to their corresponding keys. There are several standard techniques to achieve this. The simplest one is to use a multi-message secure symmetric encryption scheme, encrypt each document key with that scheme, and attach the ciphertext to the encrypted document. Since $k_{(\text{document})}$ is only needed during decryption, after match retrieval, this is both safe and convenient, and reasonably efficient. A variant would be a PRF that computes the key from the document identifier. (This requires the application to guarantee uniqueness of document identifiers.)

4.3 The "Good Server Going Bad" Model

It is time for some constructive results. In this chapter, we will develop novel security definitions that are, although relatively weak in some applications, both intuitively sound in others and satisfiable in general. Further, we will instantiate the main new notion of security, the *good server going bad model*, with a class of schemes that satisfies it.

We start with the motivation behind the good server going bad model. Assume, as before, that Chantal is a database service provider, and Murat is a company that outsources (parts of) his database operations to Chantal. Both have come to an agreement that requires Chantal to not do any analysis on the data, and Chantal will honor this agreement. So far, no cryptography would be necessary in the first place for the situation to be secured for Murat. However, at some point in time, Chantal makes an announcement to turn adversarial to Murat. From that point in time on, Chantal is a traditional adversary as developed in Section 2.3.4. In particular, Murat cannot choose to force Chantal to delete his database, since by the time Murat makes his first move knowing that Chantal has turned adversarial, she already has. What Murat can do is not provide any specific queries to Chantal that would allow her to launch an attack. Roughly speaking, this implies that for a Hom^σ-scheme (G, E, E^*, D) to be secure in this model, it is enough to require (G, E, D) to be secure, and to require Murat to not send any queries after Chantal's announcement to turn adversarial.[9]

There are at least two interpretations for this model:

- On page 10, we have already used the take-over of PeopleSoft by Oracle. If we assume for a moment that PeopleSoft is honoring its contracts, but Oracle argues that contracts closed by PeopleSoft do not apply to the newly merged company, the service provider may have to be considered an adversary under this new model in the eyes of Murat. If the two companies involved are located in different jurisdictions, the situation is complicated further.

- If a server is hosted in a country with poor (implementation of) civil rights standards, it may be seized by the authorities and analyzed offline. In this case, Chantal's announcement assumes the form of denial of service: When Murat connects to the server the first time after the adversarial turn, the server simply will be down. Examples of this

[9] Murat can still download the entire database, decrypt it, and move his business elsewhere. This can be considered a query of the form $\sigma_{\mathtt{true}}(R)$, but we will see that it is still safe.

case can be found even in countries with a good reputation for legal standards. For instance, Cicero, a conservative political and lifestyle magazine, has been illegally raided by the German police authorities after allegations of treason.[10]. Further police raids have been targeted against operators of the Internet anonymization network TOR.[11] In countries like China or Singapore, stories like this one are likely to be more numerous. Commentators disagree on the question whether the Cicero raid was consistent with legislation or not, but it certainly would make Murat's life easier if he did not have to worry about this distinction in the first place, and simply keep his secrets secret by means of cryptography rather than contracting and legislation.

Further, if for some limited time the server hosts an intruder that Chantal knows nothing about, our model is not directly applicable, but the effort for launching a successful attack is significantly increased: The intruder cannot simply walk away with the entire database, but needs to intercept communication between Chantal and Murat, gathering enough information to launch an attack. (Murat not running any queries always turns good-server-going-bad-security into the equivalent server-always-bad-security.)

We will now develop this new adversary model, and then proceed with a scheme that instantiates it. An earlier version of the material in this Section has been published in (EFG06; EFG07).

4.3.1 Definitions

Recall the definition of KC-Security in Section 3.2.3. We have exposed a weakness in the definition that stems from the independent treatment of relation and term, although terms can be used to partially reveal a relation they are applicable to. The following definition binds terms and queries into one ciphertext structure, thus repairing this flaw.

Definition 4.4 (Hom$^\sigma$-Indistinguishability)**.** *A homomorphic database encryption scheme (G, E, E^*, D) is indistinguishably secure iff for any adversarial PPTM A and any database schema $\mathcal{R}$, any two relations $R_0, R_1 \in \mathcal{R}$*

[10] "Razzia bei Cicero.", `http://www.faz.net/`, September 29th 2005. On February 27th, 2007, the German Constitutional Court ruled the raid to violate the right to freedom of press (`http://www.bundesverfassungsgericht.de/pressemitteilungen/bvg07-021.html`)

[11] "Anonymisierungsserver bei Razzia beschlagnahmt", `http://www.heise.de/newsticker/meldung/77915`, September 8th 2006.

such that $|R_0| = |R_1|$*, and any two query sequences* $\bar{q_0}, \bar{q_1}$ *such that* $|\bar{q_0}| = |\bar{q_1}|$*:*

$$\Pr\left[\begin{array}{rcl} i & \leftarrow & RND^{\{0,1\}} \\ i^* & \leftarrow & A(\mathcal{R}, R_0, R_1, \bar{q_0}, \bar{q_1}, E_k(R_i), \bar{E_k^*}(\bar{q_i})) \\ i = i^* & & \end{array}\right] \leq \frac{1}{2} + neg(N)$$

This provides Chantal with all the information that Murat cannot guarantee to keep from her in the worst case, namely all context knowledge about the ciphertext but the last bit, plus an encryption of a fitting pair of relation and query term sequence. Although this definition is not the strongest possible, any scheme that satisfies it will be sufficiently secure for many applications.

The trivial scheme based on an indistinguishable encryption scheme (G, E, D) with $E^* : \Phi \rightarrow \{0\}$ that maps every query to some singular constant ciphertext and forces Murat to retrieve the entire database for each query (doing the actual computation in the post-processing step) satisfies indistinguishability for Hom^σ. But is there a practical scheme that does so, too?

As in our analysis of $\text{Fts}^{[\cdot]}$, if we put reasonable restrictions on the queries that are submitted to the adversary, a tradeoff can be established with traditional security in the one extreme. We now give a security definition analogous to Definition 4.3 that accounts for certain attacks, and then define a class of *unsafe queries* that spans the gap between the two notions: As long as Murat does not submit unsafe queries when using a scheme being secure in the latter, the scheme is secure in the former.

Definition 4.5 (Hom^σ-Indistinguishability up to Result Size)**.** *A homomorphic database encryption scheme* (G, E, E^*, D) *is indistinguishably secure up to result size if for any adversarial PPTM* A *there is a corresponding adversarial PPTM* A' *such for that for any database schema* $\mathcal{R}$*, any two relations* $R_0, R_1 \in \mathcal{R}$ *with* $|R_0| = |R_1|$*, and any two query sequences* $\bar{q_0}, \bar{q_1}$ *with* $|\bar{q_0} = \bar{q_1}|$*:*

$$\begin{aligned} &|\Pr\left[\begin{array}{rcl} i & \leftarrow & RND^{\{0,1\}} \\ i^* & \leftarrow & A(\mathcal{R}, R_0, R_1, \bar{q_0}, \bar{q_1}, E_k(R_i), \bar{E_k^*}(\bar{q_i})) \\ i = i^* & & \end{array}\right] \\ &- \Pr\left[\begin{array}{rcl} i & \leftarrow & RND^{\{0,1\}} \\ i^* & \leftarrow & A'(\mathcal{R}, R_0, R_1, \bar{q_0}, \bar{q_1}, E_k(R_i), |E_k^*(q_{i,0})(E_k(R_i))|, \dots) \\ i = i^* & & \end{array}\right]| \\ &\leq neg(N) \end{aligned}$$

As long as Chantal merely used the size of the output in her attack, she has no advantage from running (A, B) over merely running (A', B'), since

both are provided with those sizes. We now can search for schemes that have *no further weaknesses* beyond the one described above, which is a good start. Also, it gives us this

Lemma 4.7. *Hom^σ-indistinguishability implies Hom^σ-indistinguishability up to result size.*

Proof. Let

$$\nu_A := \Pr\left[\begin{array}{rcl} i & \leftarrow & \mathrm{RND}^{\{0,1\}} \\ i^* & \leftarrow & A(\mathcal{R}, R_0, R_1, \bar{q_0}, \bar{q_1}, E_k(R_i), \bar{E_k^*}(\bar{q_i})) \\ i = i^* & & \end{array}\right]$$

and

$$\mu_A := \Pr\left[\begin{array}{rcl} i & \leftarrow & \mathrm{RND}^{\{0,1\}} \\ i^* & \leftarrow & A(\mathcal{R}, R_0, R_1, \bar{q_0}, \bar{q_1}, E_k(R_i), |E_k^*(q_{i,0})(E_k(R_i))|, \dots) \\ i = i^* & & \end{array}\right]$$

Definition 4.4 states that $\nu_A \leq \frac{1}{2} + \mathrm{neg}(N)$. for all A. This implies that $\nu_A \geq \frac{1}{2} - \mathrm{neg}(N)$. (Otherwise, an algorithm A' that outputs the inverse of A would violate the first inequation.)

If there was an algorithm A^* such that $\mu_{A^*} \geq \frac{1}{2} + \mathrm{neg}(N)$, then there would be an algorithm A^+ such that $\nu_{A^+} \geq \frac{1}{2} + \mathrm{neg}(N)$: Since the scheme is homomorphic, the sizes of the query results can be computed given the encrypted queries and the encrypted data, so the input to A^* contains strictly less information than the input to A^+.

Hence, both probability values in the definition of indistinguishability up to result size are $\frac{1}{2}$ with negligible error, and therefore

$$|\nu_A - \mu_A| \leq (\frac{1}{2} - \frac{1}{2}) + (\mathrm{neg}(N) - \mathrm{neg}(N))$$

for all A, which means that the scheme is secure in Definition 4.5. □

However, for Murat to comfortably use such a scheme in practice, he needs to make sure that he is actually willing to dismiss all attack algorithms (A', B') as acceptable, which would require an assessment of what harm can actually be done by such attacks. Since this in turn requires to limit the context knowledge of Chantal significantly, we are faced with the plethora of obstacles on the way to reasonably relaxing traditional security definitions that we have listed in Section 2.3.4.

Definition 4.5 can be further strengthened by means of the query language. By carefully restricting the class of queries whose encryptions are exposed to Chantal, we can close the gap between indistinguishability up to result size and plain indistinguishability.

Definition 4.6 (Safe and Unsafe Queries)**.** *A* safe query *with respect to some subset $\mathcal{R}' \subseteq \mathcal{R}$ of all possible instances of $\mathcal{R}$ is a query that yields always exactly 1 output tuple r if run on any instance $R \in \mathcal{R}'$.* Unsafe queries *are queries that are not safe.*

To simplify notation, we omit the subset $\mathcal{R}'$ in the following. The reader may interpret this as treatment of the special case of safe queries that are safe for all of $\mathcal{R}' = \mathcal{R}$, but all the theorems directly extend to true subsets.

The class of safe queries allows for upgrading security in the sense of Definition 4.5 to security in the sense of Definition 4.4:

Theorem 4.8. *Let $(G, E, E^* : \Phi \to \Psi, D)$ be a Hom$^\sigma$-scheme, and let*

$$E^+ : \{\, \phi \in \Phi \mid \phi \text{ is a safe query} \,\} \to \Psi$$
$$E^+(\phi) = E^*(\phi)$$

be encryption of safe queries only. If (G, E, E^*, D) exists and is Hom$^\sigma$-indistinguishable up to result size, then (G, E, E^+, D) is Hom$^\sigma$-indistinguishable.

Proof. We show that falsity of the implication implies falsity of the assumption. Let A be an adversarial PPTM that breaks Hom$^\sigma$-indistinguishability for the scheme (G, E, E^+, D), and let $\mathcal{R}, R_0, R_1, \bar{q}_0, \bar{q}_1$ be the attack parameters for which A can guess i with non-negligible probability.

Let ν_A, μ_A be defined as in the proof of Lemma 4.7 (with E_k^* replaced by E_k^+). Since all queries are safe, the result sizes that are passed to A in μ_A are all 1, and therefore the attack algorithm A has strictly less information in the probability μ_A than in ν_A.

Let $A' = \mathrm{RND}^{\{0,1\}}$ be an algorithm that outputs bits uniformly at random, no matter what the input. Then the distinguishing attack algorithm A together with A' satisfy

$$|\nu_A - \mu_A| \leq \mathrm{neg}(N)$$

and thus constitute a distinguishing attack up to frequency against (G, E, E^+, D). Finally, since $E^* = E^+$ on all queries used in our attack, by being successful against (G, E, E^+, D) it is also successful against (G, E, E^*, D). □

Safe queries are rather common. For instance, consider an RFID reader that reads a unique identification code from an tag attached to some item and extracts information associated with this code (name, price, etc.) from an encrypted outsourced database. Since for every item the code is unique and the reader sends requests only for existing codes, the reader issues only safe queries.

Are there queries that are intuitively safe, but are not covered by this definition? In other words: Is there a way to safely extend the class of safe queries?

First, assume deterministic encryption, at least in the places that match some query. (This is the case for a wide class of encryption schemes, including the one we will use in the next section to establish indistinguishability up to result size.) If the number of resulting tuples varies freely, Example 3.1 demonstrates how Chantal can run a successful attack on the encrypted database and infer some sensitive information. (A variant of this attack also works if only empty results and results of size 1 are allowed.) If all the queries return exactly k tuples for $k > 1$, it is possible to craft tables with two attributes a_0 and a_1 and queries q_0, q_1 such that for one table the queries produce intersecting and for another non-intersecting result sets. This allows Chantal to infer dependencies between values of different attributes and use them for an attack. For example, if the adversary suspects that attribute a_0 contains names of the hospitals and a_1 contains family names of the patients, by observing such intersections she can try to estimate the number of family members that were treated in the same hospital. Finally, although if all queries are guaranteed to return no tuple rather than one, Murat would also be safe, our definition is clearly more useful.

4.3.2 Cryptographic Schemes

We now finally present a class of Hom^{σ}-schemes for the restricted query language of exact selects. We start with a definition of a term that we have casually used a few times already.

Definition 4.7 (Exact select relational algebra and $\mathrm{Hom}^{=/\sigma}$)**.** Exact select relational algebra*, or* relational algebra with exact select*, is the subset of relational algebra in which only queries of the form $\sigma_{v=A}$ and the usual boolean and set operations are allowed. Such queries are called* exact select *queries. A* $\mathrm{Hom}^{=/\sigma}$-scheme *is a Hom$^{\sigma}$-scheme for relational algebra with exact select.*

Our approach is based on the intuitive analogy between running exact selects on a database relation and searching a set of documents by keyword (see Section 3.1.3). A relation can be seen as a set of documents: Each tuple is one document, and the attribute / value pairs of the tuples are the words of the document. We formally establish a structure-preserving mapping from a database table to such document sets. A scheme for full-text search as in Definition 3.2 can then be used to construct a $\mathrm{Hom}^{=/\sigma}$-scheme. We use a variant of the full-text search encryption scheme $\mathrm{Fts}^{[\cdot]}$ to demonstrate the

practical effectiveness and efficiency of our results, but any other scheme satisfying Definition 3.1.3 can be used as well.

Mapping

First of all, we define a mapping from relations to sets of documents. Given some relational schema (or relation domain) $\mathcal{R} = A_0 \times \cdots \times A_{l-1}$, let $\mathbb{D} = \{\, i : d_{ij} \mid i \in [0..l-1], d_{ij} \in A_i \,\}$ be the set of all attribute-value pairs in $\mathcal{R}$, and let $W = \{w_i\}$ be a set of words of globally fixed length such that the total number of words $|W| = |\mathbb{D}| = \sum_{i=1\ldots l} |A_i|$. Then there is always a bijective mapping from attribute-value pairs to words:

$$\diamondsuit : \mathbb{D} \mapsto W,$$
$$\diamondsuit(a_i : d_{ij}) = w_m$$

For any tuple $r_j = (1 : d_{1j}, \ldots, l : d_{lj})$ there is a corresponding sequence $V_j = (\diamondsuit(1 : d_{1j}), \ldots, \diamondsuit(l : d_{lj}))$ of corresponding words. V_j is called the *document* corresponding to tuple r_j. For every relation $R = \{r_0, \ldots, r_m\}$, there is a corresponding document set $U = \{V_1, \ldots, V_m\}$. Analogous to $\mathcal{R}$, we write $\mathcal{U} = \{U\}$ for the set of all document sets. Using the bijection $\diamondsuit$, we obtain a mapping $\heartsuit$ of relations to document sets:

$$\heartsuit : \mathcal{R} \mapsto \mathcal{U}$$
$$\heartsuit(\{(1 : d_{i1}, \ldots, l : d_{il}) \mid 0 \le i \le m\}) = \{(\diamondsuit(1 : d_{i1}), \ldots, \diamondsuit(l : d_{il})) \mid 0 \le i \le m\}$$

Since $\diamondsuit$ is bijective, $\heartsuit$ is also bijective.

Homomorphism

Next, we use $\heartsuit$ to map a full-text search problem on a document set U to an equivalent $\mathrm{Hom}^{=/\sigma}$-problem on a relation $R = \heartsuit^{-1}(U)$. In other words, we define a homomorphism which projects keyword searches into exact selects. No claims about security are made yet. In this section we only establish the connection between the two problems, in the next we add encryption and thus obtain a complete $\mathrm{Hom}^{=/\sigma}$ scheme, and after that we present the security proofs of this scheme.

Consider the structures $(\mathcal{R}, \{\sigma_{a_i:d_j}\})$ on the one hand and $(\mathcal{U}, \{\varphi_{w_{ij}}\})$ on the other. Using the bijection $\diamondsuit$ between attribute-value pairs and words, we can define a mapping between the two search operations (again, note that

this mapping is bijective):

$$\clubsuit : \{\sigma_{a_i:d_j}\} \mapsto \{\varphi_{w_{ij}}\}$$
$$\clubsuit(\sigma_{a_i:d_j}) = \varphi_{\diamondsuit(a_i:d_j)}$$

$\heartsuit$ is homomorphic with respect to $(\mathcal{R}, \sigma_{a_i:d_j})$ and $(\mathcal{U}, \varphi_{\diamondsuit(a_i:d_j)})$: If $R \in \mathcal{R}$, then

$$\heartsuit(\sigma_{a_i:d_j}(R)) = \clubsuit(\sigma_{a_i:d_j})(\heartsuit(R))$$

Adding Encryption

Using the above mappings, we can now construct a $\text{Hom}^{=/\sigma}$-scheme using an arbitrary homomorphic encryption scheme for full-text search.

Lemma 4.9. *If (G, E, E^*, D) is a homomorphic encryption scheme for full-text search, then $(G, E \circ \heartsuit, E^* \circ \clubsuit, \heartsuit^{-1} \circ D)$ is a $\text{Hom}^{=/\sigma}$-scheme.*

Proof. Directly follows from construction. □

Note that this Lemma is constructive: It provides an algorithm to transform any homomorphic encryption scheme for full-text search (G, E, E^*, D) into a $\text{Hom}^{=/\sigma}$-scheme as depicted in Algorithm 5.

Algorithm 5: Constructing $\text{Hom}^{=/\sigma}$-schemes.

Input: Relation R; encryption scheme for full-text search (G, E, E^*, D)

begin

Choose a set $W \subseteq \mathcal{M}$ such that $|W| = |\mathbb{D}|$;
Choose a bijective mapping $\diamondsuit : \mathbb{R} \mapsto W$;
Define $\heartsuit$ and $\clubsuit$ as in the text

end

Output: $(\mathcal{K}, E \circ \heartsuit, E^* \circ \clubsuit, \heartsuit^{-1} \circ D)$

Security Proof

It remains to be shown that this $\text{Hom}^{=/\sigma}$-scheme inherits the security characteristics of the underlying full-text search encryption scheme, i.e. that it is indistinguishable up to result size, and by Theorem 4.8 therefore secure if Murat is not making Chantal process any unsafe queries.

This yields our main

Theorem 4.10. *Let $\Gamma = (G, E, E^*, D)$ be a secure (in the sense of Definition 4.3) searchable encryption scheme. Then the $Hom^{=/\sigma}$-scheme $\Delta = (G, E \circ \heartsuit, E^* \circ \clubsuit, \heartsuit^{-1} \circ D)$ as constructed in Lemma 4.9 is secure (in the sense of Definition 4.5).*

Proof. Directly follows from construction. □

Corollary 4.11. *If Γ is secure in the sense of Definition 4.3, then the reduction of Δ to safe queries is secure in the sense of Definition 4.4.*

Proof. Directly follows from Theorem 4.8. □

A Partially Secure Example

In Section 4.2.2, we have presented $\text{Fts}^{\{\cdot\}}$, a set-based variant of the full-text search encryption scheme $\text{Fts}^{[\cdot]}$ that satisfies Definition 4.3. We will now use this scheme to construct our first concrete solution to the Hom^σ problem.

Let word length l and document length m be global constants. (There are simple extensions that allow for variable-length words and documents, but this is more essential for the practical utility in the context of text search than in the context of pre-specified database schemas, and we skip it here for the sake of clarity.) Remember that we write $a \,\|\, b$ for concatenation of strings a and b. '#' is the padding symbol.

For our example we use the database schema

$$\texttt{E}mp(name : string[9], dept : string[5], salary : int)$$

The privacy homomorphism is defined as follows:

$$\begin{aligned} W &= \{d_1 \,\|\, \texttt{NM}, d_2 \,\|\, \texttt{DP}, d_3 \,\|\, \texttt{SL} \mid d_1, d_2, d_3 \in \texttt{string}[9]\} \\ \diamondsuit(\texttt{name}{:}d) &= \texttt{pad}(9, d) \,\|\, \texttt{NM} \\ \diamondsuit(\texttt{dept}{:}d) &= \texttt{pad}(5, d) \,\|\, \texttt{\#\#\#} \,\|\, \texttt{DP}, \\ \diamondsuit(\texttt{salary}{:}d) &= \texttt{pad}(4, \texttt{toString}(d)) \,\|\, \texttt{\#\#\#\#} \,\|\, \texttt{SL}, \text{ where } d \in \texttt{int} \end{aligned}$$

($\texttt{pad}(i, s)$ for any string s is guaranteed to have length i.) Then, $\heartsuit$ maps tuples to documents as follows:

$$\begin{aligned} V_k &= \heartsuit(\{\texttt{name}{:}\text{Montgomery}, \texttt{dept}{:}\text{HR}, \texttt{salary}{:}7500\}) \\ &= \{\diamondsuit(\texttt{name}{:}\text{Montgomery}), \diamondsuit(\texttt{dept}{:}\text{HR}), \diamondsuit(\texttt{salary}{:}7500)\} \\ &= \{\text{MontgomeryNM}, \text{HR\#\#\#\#\#\#\#\#DP}, 7500\text{\#\#\#\#\#\#SL}\} \end{aligned}$$

Remember that sets are represented as secure (pseudo-)random permutations, and thus without order from the perspective on an adversary. The

actual string that is stored on Chantal's host is obtained by making this permutation explicit:

$$\kappa(\{\texttt{MontgomeryNM}, \texttt{HR}\#\#\#\#\#\#\#\#\texttt{DP}, \texttt{7500}\#\#\#\#\#\#\texttt{SL}\})$$
$$\rightarrow \texttt{HR}\#\#\#\#\#\#\#\#\texttt{DPMontgomeryNM7500}\#\#\#\#\#\#\texttt{SL}$$

Now all that's left to do is encryption of the resulting document strings using the searchable encryption scheme, and the ciphertexts can be shipped to Chantal's server.

♣, connects exact select and keyword search. For $\sigma_{\texttt{name:Montgomery}}$, we get

$$\clubsuit(\sigma_{\texttt{name:Montgomery}}) = \varphi_{\texttt{MontgomeryNM}}$$

which is encrypted using full-text search query encryption, and the encrypted query is run by Chantal to produce a set of encrypted strings. This set is then decrypted using $\heartsuit^{-1}$ and mapped to the corresponding tuples.

By Conjecture 4.6, the full-text search encryption scheme we used indistinguishably encrypts sets of documents and reveals nothing but the number of documents sharing the queried word. Hence, according to Theorem 4.10, the resulting $\text{Hom}^{=/\sigma}$-scheme is secure in the sense of Definition 4.5, and its restriction to safe queries is secure in the sense of Definition 4.4.

A $\text{Hom}^{=/\sigma}$-Scheme with Hom^{σ}-Indistinguishability

Independent of us (EFG06; EFG07), a security definition very similar to Definition 4.4 has recently been proposed (CGKO06). It is based on the *history* of a query protocol, consisting of data, queries, and corresponding results, and defines the input for the adversarial algorithm as the *view* on such a history. The view is composed of all the encrypted information passed to her during the protocol. Together with the definitions, two homomorphic encryption schemes for full-text search are proposed. The nature of these schemes rules out both attacks discussed in the examples in Section 4.2.1.

Neither of these attacks invalidates Conjecture 4.6, but they demonstrate a weakness in both $\text{Fts}^{[\cdot]}$ and our improvement $\text{Fts}^{\{\cdot\}}$. In particular, we can show that using either scheme in our construction of a $\text{Hom}^{=/\sigma}$-scheme breaks Hom^{σ}-indistinguishability:

Lemma 4.12. *$\text{Hom}^{=/\sigma}$ based on $\text{Fts}^{\{\cdot\}}$ or $\text{Fts}^{[\cdot]}$ is insecure in Definition 4.4.*

Proof. The attack outlined in Example 4.2 can be adapted to the database case. □

However, both schemes from (CGKO06), namely SSE-1 and SSE-2, can be shown to satisfy Definition 4.4 by straightforward modifications of the proofs in the paper. Giving an account of the details of these schemes here would not provide any new insights, and we refer the reader to the paper. However, this gives us the much stronger

Theorem 4.13. *$\mathrm{Hom}^{=/\sigma}$ based on SSE-1 or SSE-2 is secure in Definition 4.4.*

Proof. Follows from the security definition outlined in (CGKO06). □

This is an impressive demonstration of the virtue of our abstract construction connecting Hom^{σ} and full-text search: By using this construction on novel, improved results on one problem, we can directly and mechanically obtain equivalent new results for the other.

4.3.3 Discussion

Performance. The performance of $\mathrm{Hom}^{=/\sigma}(\mathrm{Fts}^{\{\cdot\}})$ is rather promising: All of stream cipher and pre-encryption and the search operations come at the cost of running comparably small boolean circuits for both Murat and Chantal. The overhead in terms of storage space is a small constant. Finally, there are simple and efficient solutions for key management. So the main part of the cost in run-time is due to the actual select (or full-text search) operation. Depending on the underlying scheme being used, this may take between linear time in the number of documents (Blo70; Goh03) and, noticeably worse, linear in the total number of words in the document set (SWP00).

From the complexity theoretic point of view, $\mathrm{Hom}^{=/\sigma}$(SSE-*) performs even better. Due to its independent and very efficient indexing structure (similar but superior to the one proposed in (CM05)), it consumes constant server complexity as opposed to the linear server complexity required for $\mathrm{Hom}^{=/\sigma}(\mathrm{Fts}^{\{\cdot\}})$. However, it is not clear how the structure of the searchable encrypted data in $\mathrm{Hom}^{=/\sigma}$(SSE-*) performs in the special case established by our construction, where the search and result patterns diverge from those for, say, search of keywords on e-mail. In particular, in many database applications the entire document (or tuple) consists of words that occur equally likely in queries. This reduces the advantage an independent index structure has over true searchable encryption, and increases the severity of imposed space overhead. Comparing the two approaches in practice would make an intriguing goal for an experimental setup.

Unfortunately, there is no straightforward way to implement indexing data structures and algorithms without losing security. We are not aware of

any homomorphic encryption schemes for databases that take indexing into account. Without further research, this shrinks the realm of applications to the few that have very low performance requirements and those where the index can be maintained on Murat's host.

Applications. There are few differences between exact select relational algebra and file systems. Both allow for retrieval of data items given a handle to that data item. Only the structure of files may differ from a relational tuple. On modern workstation and PDA operating systems, files usually have very little structure other than the order of bytes, whereas on main frames they may be data structures far more complex than common relational tuples. Further, a key (or file path) is a unique identifier for a file, whereas in exact select relational algebra it is possible to obtain more than one tuple for one key. (In a way, exact select relational algebra could be called a multi-set files system.)

Other systems that match the structure of exact select are e.g. DNS (Moc87a; Moc87b), LDAP (KDZ06), the less well-known, but potentially much bigger ONS devised by EPCglobal (Mea04; FGS05), and certainly many others.

Theorem 4.10 can be adapted to file systems and all these applications. If Murat has a local index of all existing file paths to be able to resolve file-not-found errors locally (in order to avoid the unsafe 0-tuple query results), exact selects on primary keys are always safe. Hence, Definition 4.5 is satisfied by all queries, and we obtain the following

Corollary 4.14. *There is a homomorphic file system encryption scheme secure in the sense of Definition 4.5 given the client makes sure all file look-ups are safe.*

The problem with files of widely differing size can be dealt with by splitting and padding schemes (similar to the concept of i-nodes in Posix file systems). Alex would retrieve the first block of a file, which then contains the key to the second block, and so on, until he has retrieved the entire file and assembled it locally.[12] More advanced data structures such as specialized search trees would also allow for a seek operation to reach arbitrary locations in files of size n in $O(\log n)$, or even $O(1)$.

[12]Note that this raises the issue of traffic analysis attacks, since the query sequence will develop patterns that can be used by Chantal to guess which tuples are part of the same file. How to address these issues in a provably secure way is another open research question.

4.4 Private Information Retrieval, revisited

Private information retrieval (see Section 3.4) is useful even in the setting where the database is secret and the query public. Because of the entanglements between queries and database, by keeping the queries private from Chantal, Murat can keep the encryption of the data securely confidential, too. The following theorem establishes a sufficient condition for Hom^σ-indistinguishability.

Theorem 4.15. *A Hom^σ-scheme $(G, E : \mathcal{M} \to \mathcal{C}, E^* : \Phi \to \Psi, D : \mathcal{C} \to \mathcal{M})$ is Hom^σ-indistinguishable if (1) the encryption scheme (G, E, D) is indistinguishable and (2) the transformations (f, g) such that*

$$f(q)(t) = E_k^*(q)(E_k(t))$$
$$g = D_k$$

provides PIR for the querying protocol $(\mathcal{M}, \Phi)$.

Proof. For each condition, we construct a direct contradiction to Definition 4.4 from assuming its violation. *(1)* Assume (G, E, D) is not indistinguishable. Then there exist R_0, R_1 such that Chantal can distinguish between $E_k(R_0)$ and $E_k(R_1)$ with non-negligible probability. This allows her to distinguish between $(E_k(R_0), \bar{E^*}_k(\bar{q}_0))$ and $(E_k(T_1), \bar{E^*}_k(\bar{q}_1))$ by simply ignoring the query sequences. *(2)* If (f, g) is not PIR, then Chantal can use the relation R and two query sequences $\bar{q_0}, \bar{q_1}$ from any PIR attack to distinguish between $(E_k(R), \bar{E^*}_k(\bar{q}_0))$ and $(E_k(R), \bar{E^*}_k(\bar{q}_1))$. □

This is clearly an interesting subject to future research, in particular with PIR schemes such as the one using SCPs mentioned in 3.5 whose performance may be enough for some applications. On the other hand, PIR has very hard lower complexity bounds: Even with secure co-processors, the overall work load of the system is amortized quadratic — too for most everyday database applications.

Although not strictly constructive, this theorem also suggests the construction of secure Hom^σ-schemes for array access. (Hom^σ for array access is the restriction of $\mathrm{Hom}^{=/\sigma}$ to the setting where the only searchable attribute is integer and continuous.)

Let (f, g) be a PIR-scheme for array access, i.e. for operations $\phi_i : \mathcal{M}^A \to \mathcal{M}$ on arrays from $\mathcal{M}^A$ over an element set $\mathcal{M}$. Further, let $(G, E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}, D : \mathcal{K} \times \mathcal{M} \to \mathcal{M})$ be an encryption scheme providing indistinguishability on the element set $\mathcal{M}$. Also, let (G, E^A, D^A) be the extension of (G, E, D) to entire arrays such that each element is encrypted individually.

From the first part of Definition 3.11, we know that for all ϕ_i and for all $a \in \mathcal{M}^A$, $g(f(\phi_i)(a)) = \phi_i(a)$. From the second part, we know that i is private, or intuitively, that Chantal will not know which elements Murat has retrieved.

This gives us everything we need to construct our Hom^σ-scheme for array access: For data encryption, simply use E^A to encrypt all array elements individually. For query encryption, transform ϕ_i into $f(\phi_i)$ such that Chantal will not know i, but will still be able to compute some x from an array a with it such that $g(x)$ is the i-th element of a. Finally, decryption is transformation of that x into the proper array element by means of g, plus array element decryption. In total, our Hom^σ-scheme for array access is

$$(G, E, f, D^e \circ g)$$

and from the definitions we obtain

Theorem 4.16. *If (f, g) is PIR for querying protocol $(\mathcal{M}^A, \{\phi\})$, then $(G, E, f, D^e \circ g)$ provides Hom^σ-indistinguishability.*

Proof. The two conditions from PIR and Hom^σ-indistinguishability can be directly transformed into each other. □

4.5 A Note on Data Integrity

On the last roughly one hundred pages, we have slowly developed an awareness that confidentiality on outsourced databases has proven to be surprisingly hard. We have developed definitions that are not entirely satisfactory, reduced the set of operations that we expect to run on the encrypted data, and still there are many caveats about performance, feasibility, and remaining threats by malicious service providers.

The other famous problem of cryptography, establishing not data confidentiality but integrity, has only briefly been touched in our introduction to cryptography. This is partly because in contrast to confidentiality, integrity is rewardingly simple to achieve. In fact, we almost get it for free, without any restrictions on indexing algorithms, and with small space or communication overhead.

The easiest integrity-preserving scheme simply attaches an additional attribute to every record stored on the database server containing a message authentication code (or MAC, see Definition 2.22). If a record is altered,

the MAC will not match the contents any more, and a small efficient MAC-verification routing will trigger an alert.[13]

When using full-text search schemes that establish index structures independent of the stored documents, we are free to choose any record encryption scheme that suits us, and we could use one that provides message confidentiality and integrity at the same time, such as for example (Jut00; Rog00; Hal01; BKN02). There will still be a few bits of overhead per record, but encryption and decryption will be faster since there is no need for a separate MAC verification routine.

These methods will be sufficient if Murat has an easy way to decide which records should be present in the database and which not, for instance because there is a simple algorithm to compute all serial numbers that have been allocated. In this case, if Chantal silently drops a record, an alert will be triggered by that algorithm.

If there is no way for Murat to efficiently decide which records are simply not there and which have been "lost" by Chantal, the simplest solution would be to compute the MAC of the ordered list of all record MACs. But even for databases with low dynamics, this would constitute an excessive workload for both Murat and Chantal, since the entire database needs to be touched every time a single record is modified, removed, or added.

Fortunately, there are more advanced data structures that serve the same purpose as the MAC over the list of all MACs, but they are more efficient to handle by orders of magnitude. An early one is known as Merkle hash trees (Mer80), and has subsequently been improved many times. The most recent and efficient solution we are aware of is (Bau04; MRK03), which also contains an introduction to the history of the problem.

We strongly believe this is more or less all there is to say about data integrity in the database outsourcing scenario. Efficient, effective solutions exist, and any application considering enforcing confidentiality by cryptographic means could be doing this as an aside, with no significant further cost.

[13] And being notified about an integrity violation after it has happened is the best possible outcome for Murat: There is no way for him to force Chantal to leave his data intact, once he has handed it over to her. However, usually this is sufficient, too. For example, if there is an integrity breach, Murat can retrieve a backup from a low-bandwidth system unsuitable for the database application and move to another vendor.

Chapter 5

Security and Performance Bounds

Up to this point, we have developed a formal notion for the problem of database outsourcing, Hom^σ, layed out solid cryptographic foundations to tackle it, and proposed a novel set of solutions for a subset of relational algebra. In this chapter, we conclude this part of the thesis by establishing solid theoretical feasibility bounds.

5.1 Strong Security Definitions

We start with a new security definition that encompasses what would be nice to have.

Definition 5.1 ((Adaptive) Semantic Chosen Plaintext Query Security for Hom^σ)**.** *A Hom^σ-scheme (G, E, E^*, D) is* semantically secure (has semantic security) in the chosen plaintext query model *iff for any plaintext relation R and any probabilistic polynomial-time adversarial algorithm A, there is an adversarial PPTM S such that*

$$|\Pr\left[A^{E_k^*}(E_k(R), 1^{|E_k(R)|}) = 1\right] - \Pr\left[S^{E_k^*}(1^{|E_k(R)|}) = 1\right]| \leq \mathit{neg}(N)$$

where $k = G(1^N)$.

This is a variant of semantic security (Definition 2.3), and of course there is a corresponding variant of indistinguishability:

Definition 5.2 ((Adaptive) Chosen Plaintext Query Indistinguishability for Hom^σ)**.** *A homomorphic database encryption scheme (G, E, E^*, D) is* indistinguishably secure in the chosen plaintext query model *iff for any adversarial*

PPTM A and any database schema $\mathcal{R}$, any two relations $R_0, R_1 \in \mathcal{R}$ such that $|R_0| = |R_1|$, and any two query sequences $\bar{q_0}, \bar{q_1}$ such that $|\bar{q_0}| = |\bar{q_1}|$:

$$\Pr\left[\begin{array}{rcl} i & \leftarrow & RND^{\{0,1\}} \\ i^* & \leftarrow & A^{E_k^*}(\mathcal{R}, R_0, R_1, \bar{q_0}, \bar{q_1}, E_k(R_i), \bar{E_k^*}(\bar{q_i})) \\ i = i^* & & \end{array}\right] \leq \frac{1}{2} + neg(N)$$

where $k = G(1^N)$.

Using an adaptation of the standard proof for Theorem 2.1, the two can be shown to be equivalent:

Corollary 5.1. *In the chosen plaintext query model, a Hom$^\sigma$-scheme (G, E, E^*, D) has semantic security iff it has indistinguishability.*

This notion of security accounts for the fact that Chantal can do computations on the ciphertext in an extreme sense: Any computation that Chantal might come up with during the attack is encrypted for her using a the query encryption oracle E_k^*.

As we have explained in Section 2.3.4 in the discussion of Definitions 2.7 and 2.8, encryption oracles are standard in many security definitions and can be motivated with scenarios in which the honest party is a software agent tricked into revealing oracle replies. Even if these replies are partial and require sophisticated analysis by the adversary, they may reveal the data needed for an attack. Oracles are especially likely if the adversary is not an intruder, but extensive communication with her is required by the protocol, as in our case here.

However, despite the fact that this definition looks highly useful from the perspective of the requirements engineer, it can be shown that there is no Hom$^\sigma$-scheme that satisfies it:

Theorem 5.2. *There is no Hom$^\sigma$-scheme secure in Definition 5.2.*

Before we can prove this result, we make a restriction to the Hom$^\sigma$-schemes: We only consider those schemes that allow Chantal to distinguish between empty and non-empty output. We justify this restriction with a lemma that states no schemes violating this restriction are significantly more secure.

Lemma 5.3. *If there is a secure Hom$^\sigma$-scheme (in the sense of any of Definitions 4.5, 4.4, 5.1, or 5.2)), then there is a secure Hom$^\sigma$-scheme (G, E, E^*, D) in the same Definition such that Chantal can compute the function*

$$empty : E(\mathcal{R}) \to \mathbb{B}$$

$$empty(E(R)) = \begin{cases} 0 & \text{if } R = \emptyset \\ 1 & otherwise \end{cases}$$

In particular, all schemes that encrypt tuple-wise fall naturally into this class. The only existing schemes we are aware of that do not encrypt tuple-wise are newer full-text search schemes that maintain a separate index, such as the already mentioned (CM05; CGKO06). Not surprisingly (given we are about to establish a Lemma that says so), even these allow for Chantal to identify empty query results, or even the precise size of the output (at least probabilistically, which will prove enough for a successful attack). If the homomorphic index processing returns a list of document IDs, then these documents still need to be retrieved in a second step, and this second step usually reveals the number of documents retrieved.

The proof of Lemma 5.3 is founded on the fact that we can not hope to gain much from spreading one plaintext tuple over many ciphertext bits:

Proof. For Definitions 4.5 and 4.4, we have already delivered schemes that encrypt tuple-wise and thus allows Chantal to compute empty on ciphertexts.

Note that as long as Murat does not provide Chantal with any queries, it security of the Hom^{σ}-scheme (G, E, E^*, D) collapses to the corresponding security of the underlying data encryption scheme (G, E, D).

Now assume a Hom^{σ}-scheme that conceals the size of the plaintext output in the encrypted output of a computation that is available to Chantal. In order to process queries in this scheme, either Chantal has to apply some transformation on the ciphertext to shrink it into the size of the plaintext, or communication complexity, which is the bottleneck of most distributed algorithms, is increased. Either way, the complexity of the attack can only be grown linearly in the communication overhead imposed on Murat for additional security. This violates the nature of the security definitions, which requires an exponential increase in attack complexity.

Consequently, the size of the encrypted output that Chantal produces must be within a (small, if the scheme is to be practical) constant factor of the size of the plaintext output. So no matter how the bits are distributed over the ciphertext data, each result will allow Chantal to transform this distribution into one that is still encrypted, but that is also row-wise. By a similar argument for the communication complexity, security of the scheme grows only linearly with the server computation complexity of the scheme, which (given that in a service provider setting ultimately Murat will have to pay for the consumed computing resources himself) is unacceptable. □

In analogy, consider one-way functions, which are used in the theoretical foundations of cryptography to define concepts like encryption and authentication (see Section 2.3.5). A (trap-door) one-way function is a function that can be computed in polynomial time, whereas its inverse can not (unless

one has the key to the trap-door). If encryption is a function that can be proven to be (trap-door) one-way, then the adversary can not compute the plaintext from the ciphertext in polynomial time (unless she has the key). This means that if the lower bound on the attack complexity against an encryption scheme is only doubled by doubling the effort of Alex to perform a round of encryption and decryption, the underlying function is not one-way.

With this result at hand, we can prove Theorem 5.2, or the impossibility of query-oracle-security:

Proof. Let (G, E, E^*, D) be any Hom^σ-scheme for any subset of relational algebra that allows for a relation R and two queries q_0, q_1 such that $q_0(R) = \emptyset$ and $q_1(R) \neq \emptyset$. Then, by homomorphicity, Chantal can compute $C' := E_k^*(q_i)(R_i)$, and by Lemma 5.3, Chantal can compute $\mathrm{empty}(C')$. The outcome of this computation is her guess in the attack game, and her winning probability is 1. □

5.2 Relational Algebra, Code Obfuscation

In this section, we consider Hom^σ, i.e. full-fledged relational algebra including projections, joins, exact select, and negated exact select. We propose a suitable security model and present two variants of a novel reduction proof that there is no secure Hom^σ-scheme in that model.

This does not follow from our insecurity results for $\mathrm{Hom}^{=/\sigma}$. Since Chantal is always aware of the involved language, the two are in fact orthogonal: If $\mathrm{Hom}^{=/\sigma}$ is used, she can make use of the knowledge that only specific plaintext queries are possible due to the restrictions in the query language, and thus the ciphertext result can only assume the form of exact selects; if Hom^σ is used, there may be a way to exploit the additional structure necessarily revealed in the encrypted queries as well as relations in order for Chantal to be able to process a query.

Therefore, the impossibility of secure Hom^σ-encryption gives us something new. Those who suggest that the impossibility results for exact select relational algebra are only due to an over-restricted query language are now confronted with strong evidence that lifting these restrictions will not help. In particular, any relaxation beyond the languages used in our proofs of Theorems 5.4 and 5.5 are ruled out.

Roughly speaking, our argument goes as follows: We can encode a circuit (a TM) into a relation and construct terms in relational algebra such that an evaluation of those terms yields the outcome of a run of that circuit (TM). Now assume there was a homomorphic encryption scheme for relational algebra, this would allow for a scenario in which Chantal can evaluate

an encrypted circuit (TM), obtaining an encrypted output, without learning anything about the plaintext circuit (TM). In other words: Our implementation of circuits (TMs) in relational algebra together with the assumed homomorphic encryption scheme yields a code obfuscator $\mathcal{O}$. However, we have presented a proof from (BGI+01) that such a code obfuscator does not exist in Section 3.3, so any subset or superset of relational algebra that allows for our encodings rules out secure homomorphic encryption.

This reduces the search space for existing, feasible, and secure schemes to languages strictly smaller than Hom^{σ} and strictly larger than $\mathrm{Hom}^{=/\sigma}$. The problem space in which solutions can be hoped for is thereby reduced considerably with respect to the space that is still currently explored.

We will start with two sections treating implementations of circuits and TMs in relational algebra, respectively. Then, in the third section, we will derive a reduction of homomorphic encryption of relational algebra to code obfuscation.

5.2.1 Circuits

We are now establishing the fact that we can encode circuits in relational algebra. Corollarily, if we have access to a database service provider, we can have her evaluate circuits for us.

Theorem 5.4. *There is a pair of transformations ρ from n-m circuits to relations R and ρ^c from circuit inputs $x \in \{0,1\}^n$ to queries Q such that:*

1. *Chantal can efficiently compute $C(x)$: For any circuit C, input x, relation $R = \rho(C)$, and query $Q = \rho^c(x)$, running time of $Q(R)$ is linear in $|C|$. the bit string $C(x)$ can be extracted from $Q(R)$ in time linear in x.*

2. *Q does not reveal any information about C other than the number of gates, the number of input bits n, and the number of output bits m.*

3. *Running time and output size of ρ, ρ^c are linear in the size of C and x, resp.*

Proof. The relation generated by ρ assumes the following form:

R	(circuit)	
X	`Int`	gate number
T	`String`	gate type (one of $\{\mathbf{i}\mathrm{n}, \mathbf{o}\mathrm{ut}, \mathbf{a}\mathrm{nd}, \mathbf{o}\mathrm{r}, \mathbf{n}\mathrm{ot}\}$)
V	`Bool`	input bit
I_1	`Int`	gate number of input gate 1
I_2	`Int`	gate number of input gate 2

Each gate is represented by a pair of rows with gate number X and gate type T set appropriately. The two rows are identical except that V is 0 in the first row and 1 in the second. Evaluating a gate does not require any insertions or updates in the query Q: Instead, the row with the correct V is already there, and σ-operations will serve to filter out the row with the incorrect V value. The input gate numbers I_1, I_2 point to the gates that provide it with input bits. They point to random gates wherever they are not needed. (Otherwise an adversary may be able to distinguish **i**n from **a**nd later in the encryption.)

For example, if gate 319 is an **a**nd computing the conjunction of the output bits of gates 2 and 196, its unevaluated form is

$$\{(319, \mathbf{a}\text{nd}, 0, 2, 196), (319, \mathbf{a}\text{nd}, 1, 2, 196)\}$$

This completes the representation of a circuit constructed by ρ. The meaning will become clearer while we construct the query that actually loads it with an input word and runs it. Note that the rows can be sorted using key X in a way that avoids forward references in I_1 and I_2, since a circuit is a directed acyclic graph.

Now for the query Q. First, we need to load the actual input bits that are chosen here. Let $\bar{c} = (c_0, \ldots, c_{n-1})$ be the input vector. Then for each input bit x_i, compute

$$R := \sigma_{X \neq i \vee V = c_i}(R)$$

This removes all input bit rows that have a value of V inconsistent with the actual input.

By the same trick, we then continue evaluating every gate: Consider evaluation of the ith gate in R, and assume that all previous gates have been evaluated, i.e. all possible input bits for gate i are available in R. We now compute a relation R' from R in which the value of gate i is available, i.e. only the row with the correct value of V is left in R'.

We start by constructing three relations R_1, R_2 and R_3 that will be used to look up the input bits from the references:[1]

$$\begin{aligned} R_1 &:= \sigma_{X=i, V=1} R \\ R_2 &:= R_1 \bowtie_{R_1.I_1 = R.X} R \\ R_3 &:= R_1 \bowtie_{R_1.I_2 = R.X} R \end{aligned}$$

[1]Notation: When joining two relations R_1 and R_2, attribute type A_i of R_j becomes $R.R_j.A_i$ in $R = R_1 \bowtie R_2$.

In R_1, we isolate the gate that is currently being processed (the condition arbitrarily discards the one with $V = 0$ because we are not going to use $R_1.V$ anywhere). In R_2 (R_3), this row is concatenated with the row representing the row that provides its output bit as first (second) input bit. R_1, R_2 and R_3 thus all contain one row only.

Next, this information is attached to each row of R, then it is used on the current gate for the filtering step. Finally, we project the result back onto the original attributes of R:

$$\begin{aligned} R_4 &:= R \bowtie R_2 \bowtie R_3 \\ R_5 &:= \sigma_{(R.X \neq i) \vee C} R_4 \\ R' &:= \pi_{R.*} R_5 \end{aligned}$$

R_5 looks like the update step by which the input bits have been initialized, except that there is now a condition C. C is the core of our construction. It encodes the truth tables defining the gate types. For gates of type **n**ot, C looks like this:

$$\begin{aligned} C_{\text{not}} := &(R_2.R.V = 0 \wedge R.V = 1) \\ &\vee (R_2.R.V = 1 \wedge R.V = 0) \end{aligned}$$

For **a**nd, two input references must be resolved:

$$\begin{aligned} C_{\text{and}} := &(R_2.R.V = 0 \wedge R_3.R.V = 0 \wedge R.V = 0) \\ &\vee (R_2.R.V = 0 \wedge R_3.R.V = 1 \wedge R.V = 0) \\ &\vee (R_2.R.V = 1 \wedge R_3.R.V = 0 \wedge R.V = 0) \\ &\vee (R_2.R.V = 1 \wedge R_3.R.V = 1 \wedge R.V = 1) \end{aligned}$$

or, and **o**ut are treated similarly

$$\begin{aligned} C_{\text{or}} := &(R_2.R.V = 0 \wedge R_3.R.V = 0 \wedge R.V = 0) \\ &\vee (R_2.R.V = 0 \wedge R_3.R.V = 1 \wedge R.V = 1) \\ &\vee (R_2.R.V = 1 \wedge R_3.R.V = 0 \wedge R.V = 1) \\ &\vee (R_2.R.V = 1 \wedge R_3.R.V = 1 \wedge R.V = 1) \end{aligned}$$

$$\begin{aligned} C_{\text{out}} = &(R_2.R.V = 0 \wedge R.V = 0) \\ &\vee (R_2.R.V = 1 \wedge R.V = 1) \end{aligned}$$

Finally all four gate types can be combined with a case switch:

$$\begin{aligned} C := &(R.T = \mathbf{a}\text{nd} \wedge C_{\text{and}}) \\ &\vee (R.T = \mathbf{o}\text{r} \wedge C_{\text{or}}) \\ &\vee (R.T = \mathbf{n}\text{ot} \wedge C_{\text{not}}) \\ &\vee (R.T = \mathbf{o}\text{ut} \wedge C_{\text{out}}) \end{aligned}$$

Let Q_i be the query sequence that evaluates gate i and computes R' from R. Further, let $Q_{\text{initialize}}$ be the query that computes the table with input bits all set from the raw uninitialized circuit. Finally, let

$$Q_{\text{finalize}} := \sigma_{T=\mathbf{o}\text{ut}}$$

be the query that isolates the output bits. Finally, let N be the number of gates of C (input and output gates included). Then the query

$$Q := Q_{\text{finalize}} \circ Q_{N-1} \circ \cdots \circ Q_n \circ Q_{\text{initialize}}$$

Stores an input into the circuit, computes the output, and returns it. Without any optimization, Q creates a constant number of intermediate tables the size of the original circuit. Therefore, running time of the whole transformation is quadratic in the the size of the circuit. Output size is linear of $Q(R)$ in the size of $C(x)$.

In fact, a simple optimization of the functional structure of our construction yields that running time is linear as well. However, for our reduction proofs, polynomial complexity is sufficient. □

5.2.2 Turing Machines

We are now doing what to TMs what we just did to circuits. We define one relation R_t for the tape, one R_g for the transition graph, and one R_c for the cursor:

R_t	(tape)
location	`Int`
value	`Bool`

R_g	(transition)
from	`Int`
read	`Bool`
to	`Int`
write	`Bool`
move	$\{+1, -1, 0\}$

R_c	(cursor)
location	`Int`
state	`Int`

Before the TM is run, the tape R_t is initialized with the input contents. The cursor R_c is set to the start state and start position on the tape.

There is one crucial difference between the set of queries that interprets these three relations as a TM and the analogous set for circuits: Our trick to start with all possible values on all gates of a circuit and then subsequently eliminate false values helped us to stay within the purely functional part of relational algebra. However, with potentially infinite tapes, this does not work any more. We need to do real updates to both tape and cursor relations here.

We start with the definition of a few "macros". The first two simply extract the current state from the cursor relation, and third reads the bit on the tape currently under the read/write head, and finally, the fourth is for looking up the transition about to be taken determined by the current state and tape contents.

$$\begin{aligned}
\mathrm{LOC}(R_c) &:= \pi_{\mathrm{location}}(R_c) \\
\mathrm{STATE}(R_c) &:= \pi_{\mathrm{state}}(R_c) \\
\mathrm{READ}(R_c, R_t) &:= \pi_{\mathrm{value}}(\sigma_{\mathrm{location} \in \mathrm{LOC}(R_c)}(R_t)) \\
\mathrm{TRANS}(R_c, R_t, R_g) &:= \sigma_{\mathrm{from}=\mathrm{STATE}(R_c) \,\wedge\, \mathrm{read}=\mathrm{READ}(R_c,R_t)}(R_g)
\end{aligned}$$

Since relational algebra is purely functional, an update operation is modelled as the construction of a new table from an old one. We use an update operation that is a little simpler than the one provided by SQL. Ours inserts a tuple t and removes all tuples with the same values for attribute k. (If k is a primary key, then at most one row is removed for each insertion.)

$$\mathrm{UPDATE}(R, k, t) := \sigma_{k \neq \pi_k(t)}(R) \cup t$$

The updated table R' is obtained by function assignment

$$R' := \mathrm{UPDATE}(R, \dots)$$

For example:

$$R'_c := \mathrm{UPDATE}(R_c, (\mathrm{location}, \mathrm{state}), (15, \mathrm{STATE}(R_c)))$$

sets the new current location to 15, leaving the pointer to the current state in the transition graph intact.

Now we can carry out one full state transition:

$$\begin{aligned}
x &:= \mathrm{TRANS}(R_c, R_t, R_g) \\
R'_t &:= \mathrm{UPDATE}(R_t, \mathrm{location}, (\mathrm{LOC}(R_c), \pi_{\mathrm{write}}(x))) \\
R'_c &:= \mathrm{UPDATE}(R_c, (\mathrm{location}, \mathrm{state}), (\mathrm{LOC}(R_c) + \pi_{\mathrm{move}}(x), \pi_{\mathrm{to}}(x)))
\end{aligned}$$

Algorithm 6: Looping over a TM transition sequence in relational algebra.

Input: R_t, R_g, R_c, R_s as constructed above

while $|R_c| > 0$ **do**
$x := \text{TRANS}(R_c, R_t, R_g)$;
$R_t := \text{UPDATE}(R_t, \text{location}, (\text{LOC}(R_c), \pi_{\text{write}}(x)))$;
$R_c := \text{UPDATE}(R_c, (\text{location}, \text{state}), (\text{LOC}(R_c) + \pi_{\text{move}}(x), \pi_{\text{to}}(x)))$;
$R_c := \sigma_{\text{state} \notin R_s}(R'_c)$;

Output: R_t

There is only one little technicality to be taken care of. The tape of a TM can be thought of as initialized with in infinite number of 0 bits. Short of constructing a relation of infinite size, we need to treat a failed lookup ("this location on the tape has never been visited before") as a 0 bit. This can be done by modifying the READ macro. If the tape contains 1, READ* returns 1, otherwise it returns 0.

$$\text{READ*}(R_c, R_t) := \max(\text{READ}(R_c, R_t) \cup \{0\})$$

Note that this is the only place that requires boolean comparison (or any other). If there was a way to initialize the tape with enough 0 bits before the machine is run, our encoding would work with relational and set operations alone.

How does it all end? If a dedicated stop state is entered, then no further step is taken, and the current contents of the tape is the output of the TM. Stop states are listed in a fourth relation that is simply a set:

R_s	(stop)
stop_state	`Int`

When a stop state has been reached in R_c, the next state R'_c is the empty relation, so as the next current state we simply select all non-stop states from R'_c:

$$R''_c := \sigma_{\text{state} \notin R_s}(R'_c)$$

Finally, we need to construct a loop that starts from an initial state and traverses a sequence of transitions that is either infinite or terminates with a stop state eventually.

Relational algebra lacks the expressive power to express such a loop recursively as one closed expression. Instead, we carry out Algorithm 6. There are two things to note about this algorithm.

1. Since it assumes the tape has already been loaded with the TMs input, it is completely independent of that input. Furthermore, since transition graph, start state and stop state are all encoded in the relations, it is also independent of any specific TM, and runs on all TMs without change or calibration.

2. In the next section we are going to face the question whether the algorithm works on encrypted tables, too. To answer it, it is important to see that there is only one place where the plaintext result of an operation on the data is needed by the algorithm: The loop condition that requires R_c to be non-empty. All other operations could be encrypted by Murat in constant time to fit the encrypted data and produce encrypted intermediate data and output.

We are now ready to state the analogous to Theorem 5.4.

Theorem 5.5. *There is a pair of transformations ρ from TMs to relations R and ρ^c from TM tape contents $x \in \{0,1\}^n$ to queries Q such that:*

1. *Chantal can efficiently compute $M(x)$: For any TM M, input x, let $R = \rho(M)$, and $Q = \rho^c(x)$. Running time of $Q(R)$ is linear in x. Algorithm 6 can be used to load the tape with the outcome of $M(x)$ with polynomial overhead. Algorithm 6 terminates if and only if $M(x)$ terminates. The contents of the tape can be extracted in time linear in its current size.*

2. *Q does not reveal any information about M.*

3. *Running time and output size of ρ, ρ^c are linear in the size of M and x, resp.*

Proof. The second and third item are straightforward. To see why ρ^c together with an empty tape relation R_t can replace a loaded R_t, return to the proof of Theorem 5.4 in the last section, where we simply transformed every record into an `insert` statement. This can be done in linear time, as can the inverse operation to produce the desired output in its original form. That Algorithm 6 is a generic TM interpreter has been established during its construction above. None of the operations involved that implement constant-time operations on TMs have super-polynomial complexity. □

5.2.3 Reduction Proofs

Given a secure Hom^σ-scheme, Chantal can now evaluate encrypted circuits (or TMs) given encrypted input and producing encrypted output, without learning anything about what she is doing. This does not establish a code obfuscator yet, though. Chantal still needs to have the means to feed a plaintext input of her choice to the circuit and retrieve the plaintext output. For now, both are safely encrypted and out of her reach.

We tackle this issue by introducing encryption oracles for queries. Chantal is allowed to choose an input, transform it into a query using ρ^c, and hand that query over to the oracle for encryption. Then, she can run the circuit without further help by Murat.

In this section, we define a reduction of Hom^σ-schemes to code obfuscators using such oracles. This directly yields to further proofs for Theorem 5.2, i.e. that one using circuits and one using TMs that no Hom^σ-scheme can be secure in Definition 5.1 (which is equivalent to 5.2). Then we propose two new security definitions that are restrict the powers of the adversary, and extend the reduction proofs to those.

We start with two proofs for Theorem 5.2, one based on circuits and one based on TMs. (Since the two use slightly different subsets of Hom^σ, they are both interesting in their own right.)

Proof. We prove Theorem 5.2 using Theorem 5.4. We show that if there is a Hom^σ-scheme secure in Definition 5.2, then there is a circuit obfuscator secure in Definition 3.7.

Assume the existence of a secure Hom^σ-scheme. By Lemma 5.3, the encryption of a result alone suffices to decide whether it contains 0 tuples or more.

We use this assumed scheme in the construction of a circuit obfuscator according to Theorem 5.4. Murat encrypts his circuit, without input, and hands it over to Chantal together with the query that will interpret it after it has been loaded with input. Chantal, with her access to the query encryption oracle, obtains a suitably encrypted input of her choice and evaluates the circuit.

Now the result is available in encrypted form. Chantal probes the oracle with $2|R|$ more queries (two for each tuple in the encoded circuit):

$$q_{2i} = \sigma_{X=i \wedge T=\mathbf{o}\text{ut} \wedge V=0}(R)$$
$$q_{2i+1} = \sigma_{X=i \wedge T=\mathbf{o}\text{ut} \wedge V=1}(R)$$

There will be exactly m non-empty results, one for each output gate. From the queries that have provoked those, Chantal can deduce the output

bit sequence. The entire process of recovering the result takes $O(|R|)$ time. Hence, the obfuscator is efficient, i.e. runs in polynomial time.

Let the obfuscation $\mathcal{O}(C) := (E_k(\rho(C)), E_k^*)$ be the pair of the encrypted relation representing the circuit and the query encryption oracle, so for the obfuscator $\mathcal{O}$ we have just constructed, any algorithm $A(\mathcal{O}(C), \dots)$ that receives such an obfuscation as input can be rewritten as

$$A^{E_k^*}(E_k(\rho(C)), \dots)$$

(We have not discussed obfuscated code with oracles in Section 3.3, but the extension is straightforward. For instance, in the proof of Lemma 3.4, we construct two circuits, and we demonstrate that any obfuscation yields more information than pure oracle access to the implemented functions. We allow the obfuscator to output circuits that come with an oracle. But the argument of the proof, namely that two circuits can be designed to recognize each other while independent oracle access to both circuits does not allow for this recognition, would remain valid.)

It remains to be shown that from the security of Hom$^\sigma$-scheme in Definition 5.1, it follows indeed that our obfuscator satisfies Definition 3.7. Or, equivalently, that violation of Definition 3.7 by our obfuscator implies violation of security of the underlying Hom$^\sigma$-scheme.

Assume the obfuscator violates Definition 3.7. The first two conditions of the definition are met by construction, so it must be the black-box property that is violated: There must be a circuit C and a TM A such that for any S:

$$|\Pr[A(\mathcal{O}(C), 1^{|C|}) = 1] - \Pr[S^C(1^{|C|}) = 1]| > \mathrm{neg}(|C|)$$

Let $R := \rho(C)$. By construction, we know that $|E_k(R)| = O(|C|)$. Since $\mathcal{O}(C) = (E_k(R), E_k^*)$, we have

$$|\Pr[A^{E_k^*}(E_k(R), 1^{|E_k(R)|}) = 1] - \Pr[S^C(1^{|E_k(R)|}) = 1]| > \mathrm{neg}(|E_k(R)|)$$

Note that by padding C with gates that do not affect the output, we can obtain a circuit whose size $|C| > O(N)$ is at least linear to the security parameter. This way, for any negligible function f, we can make sure that $f(|C|) > f(N)$.

Finally, observe that no matter how sophisticated the choice of q, $E_k^*(q)$ in itself does not yield any information about R, but only allows to extract it from $E_k(R)$. Since $E_k(R)$ is unavailable to S, whatever $S^{E_k^*}$ can do with $1^{|E_k(R)|}$, S^C can do as well.

In other words, $S^C(1^{|E_k(R)|})$ is strictly better than $S^{E_k^*}(1^{|E_k(R)|})$ in emulating A, and in conclusion we obtain

$$|\Pr\left[A^{E_k^*}(E_k(R), 1^{|E_k(R)|}) = 1\right] - \Pr\left[S^{E_k^*}(1^{|E_k(R)|}) = 1\right]| > \mathrm{neg}(N)$$

which is a direct contradiction to Definition 5.1. □

In order to show that if there is a Hom^σ-scheme secure in Definition 5.1, then there is a TM obfuscator secure in Definition 3.5, we use the same argument.

Proof. We prove Theorem 5.2 using Theorem 5.5. We construct a TM obfuscator according to 5.5. Injection of input and extraction of output follows the same idea as in the circuit case. Algorithm 6 can still be run on the obfuscated circuit without change: All steps are terms in relational algebra that translate directly into the encrypted case because Hom^σ is homomorphic. The only exception is the condition $|R_c| > 0$, which can be decided from knowledge of $|E_k(R_c)|$ alone because of Lemma 5.3.

As in the circuit proof above, it merely remains to be shown that from the resulting obfuscator $\mathcal{O}$ violating the black-box property in Definition 3.5, it follows that the Hom^σ-scheme violates Definition 5.1: Choose M such that

$$|\Pr[A(\mathcal{O}(M), 1^{|M|}) = 1] - \Pr[S^M(1^{|M|}) = 1]| > \text{neg}(|M|)$$

(As before, we know such an M exists, since no obfuscators exist.) Let $R := \rho(M)$. By construction, we know that $|E_k(R)| = O(|M|)$. Since $\mathcal{O}(M) = (E_k(R), E_k^*)$, we have

$$|\Pr[A^{E_k^*}(E_k(R), 1^{|E_k(R)|}) = 1] - \Pr[S^M(1^{|E_k(R)|}) = 1]| > \text{neg}(|E_k(R)|)$$

and by the same arguments as before, this yields

$$|\Pr\left[A^{E_k^*}(E_k(R), 1^{|E_k(R)|}) = 1\right] - \Pr\left[S^{E_k^*}(1^{|E_k(R)|}) = 1\right]| > \text{neg}(N)$$

which is the same contradiction to Definition 5.1 (only using a slightly different set of operators from the relational algebra). □

So far we have presented new and more complex proofs for what we already know from Section 5.1. In order to make this result more meaningful, we can exploit the fact that the attack does not require adaptive access to the query oracle: Since the algorithm under de-obfuscation attack is chosen by the adversary, the attacks can be pre-computed as well, and provided as queries together with the circuit. This suggests a new definition with a less powerful adversary:

Definition 5.3 ((Non-Adaptive) Semantic Chosen Plaintext Query Security for Hom^σ)**.** *A Hom^σ-scheme (G, E, E^*, D) is* semantically secure (has semantic security) in the chosen plaintext query model *iff for any plaintext*

relation R, any sequence of queries $\bar{q}$, and any probabilistic polynomial-time adversarial algorithm A, there is an adversarial PPTM S such that

$$|\Pr\left[A(E_k(R), 1^{|E_k(R)|}, E_k^*(\bar{q})) = 1\right] - \Pr\left[S(1^{|E_k(R)|}, E_k^*(\bar{q})) = 1\right]| \leq neg(N)$$

where $k = G(1^N)$.

Note that the ciphertext containing the query is provided to both algorithms. It does not help the adversary to attack the query encryption E^*, since we are only interested in hiding the data.

By the standard technique used in the proof of Theorem 2.1, non-adaptive semantic chosen plaintext query Hom^σ-security can be shown to be equivalent to Hom^σ-indistinguishability (Definition 4.4). We will conclude this section by extending our reductions to code obfuscation to these two security notions.

Theorem 5.6. *There is no Hom^σ-scheme secure in Definition 4.4.*

Proof. We prove semantic insecurity (insecurity in Definition 5.3); distinguishability follows from the equivalent of the two definitions.

A reduction proof has been carried out above for the case where A has access to a query encryption oracle. However, the use of this encryption oracle is unnecessary. The adversary can choose R, $\bar{q}$ in advance, and so she can play through the attack against the obfuscator before the game starts. When she is presented with the obfuscated circuit, she has already pre-encrypted all the input and output operations that she needs. □

How many proofs for one theorem should there be?

In Chapter 4 we have developed solutions for homomorphic encryption of databases where the homomorphism property was restricted to exact select queries. In this Chapter, we have provided evidence that for more complex query languages, no such results exist in principle. In retrospect, Hom^σ-indistinguishability seems to be the most fruitful definition of homomorphic database security: Equivalent non-homomorphic indistinguishability definitions have proven to be strong enough to be of practical relevance and to be accepted by the requirements engineer, and we have shown Hom^σ-indistinguishability to be satisfiable for the restricted $\text{Hom}^{=/\sigma}$. Above, we have proven it to be out of reach for a richer subset of relational algebra.

But why have we presented so many different proofs of impossibility? First, every proof makes use of a slightly different subset of relational algebra (one based on circuit emulation, one based on TM emulation, and a very simple one based on the result emptiness operator alone).

None of these query languages constitutes a core language for all supersets of which impossibility of indistinguishable homomorphic encryption is established. Instead, it is possible that by making the query language more complex and query encryption more opaque, the possibilities of what the encrypted query result might contain multiply, and the adversary gains less information from analyzing them. Therefore, the use of three proofs using three different sets of query operations in the attack is instructive.

Second, a vast number of publications exists that propose Hom^{σ}-schemes of one kind, some of them very recent, and many of these solutions fail to provide a rigorous demonstration of their security for the very reason that the schemes provide none. By developing different arguments along different lines, we hope to convince the casual reader that many of the options are simply unavailable, and to give a better intuition of what might still be possible. With this thesis, we hope to help the community to save a lot of futile effort to invent the perpetuum mobile.

Third, the impossibility of code obfuscation is a very deep result with many side tracks proving impossibility of obfuscation various special subclasses of algorithms such as signature schemes, encryption schemes, and pseudo-random functions. This means that even if the counter-examples we used to establish our results look highly artificial, for many patterns that may occur in a concrete database application there are adaptions of our counter-examples so that these patterns are instantiated, and the impossibility extends to these applications. (This is to deal with a line of reasoning that we have already debunked in Section 2.3.4, namely that security should only be necessary for those inputs that actually occur in practice.)

And finally, the connection of two important problems in cryptography by reduction of one to the other is of theoretical interest. Note that the reduction of TM-obfuscatability to Hom^{σ} (with any subset of relational algebra) is trivial. Databases, like any other system in computer science, can be implemented on a Turing machine. Hence, assuming suitable security definitions, by our reduction proof we have established an equivalence between code obfuscation and homomorphic encryption of relational algebra.

5.3 Open Problems

Despite the class of practicable $\mathrm{Hom}^{=/\sigma}$-schemes we have constructed in the last Chapter, the question whether homomorphic database encryption has the potential to ever become commercially viable is still open. We have approached the question mathematically, and presented various novel proofs that there are tight bounds for feasibility even on the level of complexity

theory. We will conclude this part of the thesis with the discussion of a few more practical reasons why the problem is so stubbornly resisting elegant solutions.

5.3.1 Performance

Databases are some of the most demanding applications conceivable in terms of computing resources. The amounts of data are huge even for modern hard drives, and systems have to cope with frequent complex changes in the data. It has become the exception that one user accesses a single private database. Even e-mail has been integrated in groupware systems in which many users share their data, which makes sophisticated role user and role management and access control necessary, further complicating the application, and adding to the challenge of getting the system running fast enough.

Certainly the majority of systems is not as extreme in all of these respects as we picture, but it is a fact that databases in real life require considerable hardware resources, despite the fact that decades of research have resulted in algorithms with surprising performance characteristics.

Now when we consider two alternative system designs:

1. The server sits in a trusted realm (e.g., on Murat's company premises), and no cryptography is involved in database access beyond protection of the communication between client and server.

2. Server time is rented from an untrusted service provider run by Chantal, and the application is hardened by the cryptographic machinery presented in this thesis.

1. has the disadvantages we have discussed in Section 2.1. But being realistic, we have to worry that 2. may have serious performance issues.

Even the $\mathrm{Hom}^{=/\sigma}$-schemes presented in Chapter 4, although featuring good complexity characteristics, have a constant overhead, and constant overhead may already be fatal to our applications, since there simply is no money for a machine ten times as fast, or with ten times as much storage space, or both.

And even if there is, or if the numbers are better, there is the question of how much overhead is caused by the impossibility of optimization. Chantal deals with encrypted data and has a very limited understanding of what she does. This is a requirement, and we have struggled hard to satisfy it. But its down side is that optimization techniques like caching, query planning, or advanced indexing algorithms have to be re-examined from scratch. In the end, they may turn out to be considerably less effective. Indexing in

particular is only in its infancy for Hom^σ-schemes (as is the field of Hom^σ-schemes itself, to a slightly lesser extent), and the theoretical optimum in slowdown may be quite significant.

In light of these thoughts, the parameters and interpretations of the experiments to gauge the performance of the often-cited ATE algorithm (HILM02) appear overly optimistic. The databases of size 10MB and 100MB can be held and processed on almost any embedded device. This would arguably be cheaper than the hardware required to handle the cryptographic and communication overhead.

There may be applications for which the state of the art is suitable, but it is one of the challenges of the field to point them out. For that, more research needs to be done in several directions:

- **P**erformance, not complexity: What is the concrete performance on existing database engines that have been enhanced with homomorphic database encryption? How do the different schemes compare, and where are the bottlenecks?

- **A**pplications: Which benchmarks and parameters are suitable for assessing the utility of new encryption schemes? In other words: What are the conceived applications the technology is expected to one day benefit?

People have attempted to look in these directions, and (HILM02) is one of the preliminary results. Now, with the cryptographic theory of homomorphic database encryption developed in this thesis, it may be time for their agenda again.

5.3.2 Covert Channels and Advanced Cryptanalysis

In Section 3.5, we have briefly mentioned that there are advanced attacks against secure co-processors that make use of the information gained by observing the co-processor in operation, such as electro-magnetic radiation, the precise time it takes the processor to perform a certain operation, or power consumption.

Similar issues may occur in distributed scenarios where the adversary has access to part of the networked hosts, observes communication and computations, and deduces confidential information from measuring *the wrong thing*.

For example, timing attacks (EH96) can be used to compute the exponent d from some g and the time it takes some player to compute g^d. The algorithm

relies on the fact that exponentiation by d is almost always implemented as a series of shift operations on g for the individual bits in d, and that the amount of CPU cycles treatment of one bit of d takes depends on the value of that bit. This can be used by a malicious agent with sufficient capabilities to trick a person into revealing her secret RSA key by making her either sign or decrypt several messages.

The timing information that the adversary makes use of in this example is called a *covert channel*. Covert channels have received some attention for a number of years now, and are generally considered to be exceedingly hard to detect and eliminate. Covert channels usually involve an eavesdropper on a connection who uses traffic flow characteristics to infer information about the contents of a communication.

For instance, the SSH protocol for secure remote Unix shell operation has suffered an attack recovering passwords typed in by hand to log in on a remote host. The attack exploits the fact that characters are transmitted in individual packets, and password characters can be recognized from the ciphertext stream because they are not echoed. Given enough timing information and a machine learning algorithm that predicts locations of keys on the keyboard from delays between key presses, the password can be recovered with astonishing accuracy.[2] Generally speaking, the more complex a protocol or application becomes, the higher the risk of covert channels leaking information about secrets held in some part of the system that is considered protected.

When a cryptographic algorithm is defined and analyzed, and ultimately proven secure, the abstract representation of the algorithm is an extreme simplification of its implementation and run-time characteristics. For example, in theory there is no run-time or power-consumption behavior associated with exponentiation. There is just an atomic operation that has no characteristics besides mapping certain inputs to certain outputs. Only later, when the provably secure algorithm is implemented, the covert channels emerge, and operations that are absolutely and positively secure in the research paper suddenly start leaking secrets.

This is not to say theory is futile. It is a vital first start to establish a system that is worth being protected against covert channels in the first place. It just is not enough yet.

Homomorphic encryption for complex languages appears to be particularly vulnerable to covert channels. The adversary has an excessive amount of information about data, operations, results, has an active part in the communication, and can often observe automatic responses of the client ap-

[2] `http://www.dailycal.org/sharticle.php?id=19585`

plication in real time. Such an adversary has a huge advantage over a passive eavesdropper on an encrypted communication channel, so there are likely to be more holes to plug as well.

Should we elaborate further on this issue then? Unfortunately, the status quo of homomorphic encryption poses many problems with which we would have to struggle first. Covert channels change nothing about the fact that we need rigorous definitions of what we mean when we say we want a system to be secure. Algorithms that do not deliver to these definitions need not be protected against attacks based on implementation glitches. And as long as there are no algorithms in the first place for many of problems, it makes little sense to protect them. But given that the technology will mature to a point where it meets its first production systems, it will be essential to keep in mind that Chantal has excessive control over the application run, and to think hard about what she can do with it.

Chapter 6

Conclusions

If a database production system is deployed, nobody questions the virtue of a sound theory of databases that the system is based on. If an insecure network connection is used between client and server for sensitive applications, reliable encryption and authentication mechanisms are used to protect the user against attacks. However, solutions for the problem that the database server itself goes adversarial have so far often been based on much lower standards: Rather than focusing on robust worst-case scenarios, some (not all) researchers have been more concerned with minimizing performance overhead or increasing expressive power of the query language.

This thesis is an effort to change this. We have rigorously approached the problem from a theoretical angle, and made an attempt to put a problem that has been generating research publications for several years on a solid foundation.

Foundations and Definitions

In Chapter 2, we have given an original account of the foundations of cryptography that we feel is unique in its emphasis on giving intuitive and motivating examples. Most of the arguments we elaborate on in great detail are not considered worth mentioning even in basic cryptographic text books, which has often led to misunderstanding between the security and the database community. We have contributed to resolving these problems and given readers with a strong background in databases a new perspective on security. In Chapter 8, we will re-apply these foundations to the field of P2P network security.

In Chapter 3, we have given a survey of the state of the art in homomorphic cryptography, with a classification of operator languages and an extensive enumeration of encryption schemes and security definitions from

the research literature. We also have covered the context in which research in homomorphic security is located, and from which we have used results later to advance it.

In Chapter 4, we have developed new security definitions. The two most important ones are the good server going bad model that models situations in which the client knows if the server is to be trusted or not, and can take action accordingly; and the strictly more powerful (and thus harder to achieve) Hom^{σ}-indistinguishability.

In (KM04), the question what security proofs can do for practical security is given some thought on a more philosophical level. Our thesis contributes a few insights that complement this paper nicely. We have discussed the full-text encryption scheme $\mathrm{Fts}^{[\cdot]}$ that is "provably secure", and for all we can tell the proof is sound and complete. Nevertheless, we have found that a correct implementation may provide very little actual security, since the notion of security underlying the proof is inadequate. We have strived for contributing to the search for *better* security definitions, where *better* can mean any of (1) better matching our intuition of what "secure" means; (2) easier to instantiate; (3) more suitable for disqualifying schemes with poor security.

We believe we have achieved all three of these goals. At the core of our collection of new security definitions lies Hom^{σ}-indistinguishability, which has been independently proposed by us (EFG06; EFG07) and others (CGKO06). We have used this basic model to span a family of relaxations and restrictions, which have lead us to a range of insights, both constructive (by inspiring new solutions) and destructive (by inspiring proofs that other solutions are impossible). In our analysis of ATE and in an effort to establish a limited, but reliable level of security, we have developed the notion of δ-security that we hope will be of independent interest for analysis of homomorphic crypto schemes in the future.

Solutions and Limits

The observation that full-text search and a limited set of database operations can be mapped into each other has led us to merging two independent fields of investigation. We have proposed a construction that transforms any homomorphic encryption scheme for full-text search into a $\mathrm{Hom}^{=/\sigma}$-scheme for database encryption. Using two exemplary full-text search schemes, we have obtained two secure solutions for a restricted query language, one of them with good server going bad security, and one satisfying our core notion of Hom^{σ}-indistinguishability.

We also have developed a reduction of PIR schemes to Hom^{σ}-schemes.

The encrypted query (and in particular the result obtained by it) is often used by an adversary to derive information about the encrypted data. This has led us to the insight that if we encrypt the query in a PIR way, i.e. such that the adversary cannot see what the query does, the data becomes more secure by extension. We have used this insight to construct a simple Hom^σ-scheme for array access from an arbitrary PIR scheme.

Last but not least, we have covered the orthogonal question of data integrity in the database outsourcing scenario in a few paragraphs as an afterthought to Chapter 4. Perhaps because it turns out to be so easy to address, to our knowledge this is the first account of this matter.

There are limits to what can be done. In Chapter 4, we have exposed two very popular results from related work, namely ATE (HILM02) and $\mathrm{Fts}^{[\cdot]}$ (SWP00), to extensive scrutiny, uncovering several flaws and inconsistencies.

In Chapter 5, we have provided three independent proofs of impossibility for two of our stronger security definitions, covering three different subsets of relational algebra. Since PIR can be reduced to Hom^σ, this yields a previously unestablished lower bound for the related problem of private information retrieval as well.

Also, our reduction of homomorphic encryption to code obfuscation, plus the trivial reduction of code obfuscation on homomorphic encryption,[1] yields equivalence of the two problems. This may be of great value for future infeasibility proofs.

The field is still very young and the challenges hard. With our new the assembly and additions to a formal treatment of the field, new constructions of homomorphic encryption schemes, and an extensive assessment of what cannot be done, we hope to have given it a small jolt in the right direction.

[1] If there is code obfuscation, modify the database engine so that it decrypts all data *from* client or database and encrypts all output *to* client or database with a non-homomorphic symmetric scheme. The key is stored inside the code, so obfuscation will keep it safe. Since this engine can be obfuscated, its non-homomorphic symmetric scheme and key can be used for a database privacy homomorphism.

Part II

Trust and Reputation in P2P Networks

Chapter 7

Introduction

Whenever individuals or organizations, or nodes in a social network, engage in resource transactions, whether in a file-sharing network, on NYSE, or in a kibbutz, trust plays a crucial role in the decisions they make. This is independent of whether the underlying economic rules impose monetary incentives, rely on altruism or on a general willingness to play by the rules, or other behavioral drivers. If the nodes do not know each other and change business partners often, a small number of adversarial nodes can do a lot of damage to the functioning of the overall system. However, the trust that is necessary to identify and exclude adversaries is hard to earn because the only a priori reliable source of information of any node on the behavior of others is itself. Hence, reputation distribution in form of rumors is common, if not ubiquitous, in most social and economical systems.

In this part of our thesis, we focus on P2P resource pooling networks as a large and growing class of distributed information systems. We propose an abstract network model that keeps track of the cooperation level of the participating peers (or nodes) that helps to grasp those aspects of rumor-spreading agent systems relevant for reputation.

This model is based on state-of-the-art cryptographic research and makes explicit, plausible assumptions on the honesty of the users. Using this model, we examine a number of methods for reputation tracking using simulations to obtain comparable utility results.

7.1 Concepts and Outline

The Merriam-Webster Online[1] defines

[1] `http://www.m-w.org/`

Trust: firm belief in the integrity, ability, effectiveness, or genuineness of someone or something.

Reputation: overall quality as seen or judged by people in general.

We are going to follow a similar slightly more formal notion.

Definition 7.1 (P2P Network, Trust, Reputation)**.** *A* P2P network *(or an* economy*) is a graph* (V, E) *in which the nodes (or vertices)* $i, j, \cdots \in V$ *are* peers*, or* individuals*, and the edges* $(i, j), \cdots \in E$ *are* channels of resource exchange *between those nodes. Every node can have both incoming and outgoing edges, i.e.both contribute and consume resources. In a* fully connected P2P network*, every node can provide (or consume) resources to (or provided by) every other node.*

Assume $(i, j) \in E$*. Then the* trust *of node* i *in node* j *is any real number* $\tau_{i,j} \in \mathbb{R}$ *(at a given time). The higher* $\tau_{i,j}$*, the more trustworthy* i *considers* j*.*

The reputation *of node* j *as observed by node* i *is an aggregation* $\Omega_{i,j} := f_i(\{\tau_{k,j} \mid k \in V\})$ *(at a given time) for some suitable aggregation function* f*.*

What resources are exchanged strongly varies from application to application and is not subject of this general definition (see Section 7.2 for an incomplete list).

Trust is a subjective concept. Two nodes may disagree on the behavior of a third node, based on the different experiences they have made. Reputation is the aggregation over all subjective reputations, and represents the view on a node of the network as a whole. P2P networks can exist without any technical notion of trust or reputation. However, the introduction of these concepts can affect the behavior of the network positively by giving nodes incentives to cooperate more.

A *reputation scheme* is a method for computing the reputation of a node j from (possibly incomplete or inaccurate) trust data on j. A robust reputation scheme is a necessary condition for effective reputation-based incentives, and should withstand as wide a range of types of adversarial behavior as possible. This part of this thesis is about the construction, comparison, and evaluation of such reputation schemes.

Reputation schemes have been around as long as economics. In the 15th century, informal trade networks between merchants across Europe established a vital backbone of the Hanse (GMW94; Ogi00; Str04; ES06; Thr41). These networks provided the infrastructure of simple *multilateral reputation mechanisms*, in which a traitor was excluded from trade with anybody in the network. A traitor is any node, whether market place in a town or individual

merchant, that has achieved a negative trust level with any trusted node. Formally (and slightly simplified),

$$\Omega_{i,j} := \begin{cases} -1 & \text{if } \exists k \,:\, \tau_{k,j} < 0 \text{ and there is a path in } E \text{ from } i \text{ to } k \\ 0 & \text{otherwise} \end{cases}$$

The nodes inside one network could trust each other to a large extent, because belonging to a large network of trust was exceedingly profitable, and that any member would risk being excluded was unlikely. Hence, these networks could also be used by others as shortcuts for long-distance transactions which, because of the security overhead, would have been prohibitively expensive to carry out between untrusted parties.

A related phenomenon from today's highly specialized economy is the computation of reputation by financial analysts such as Standard & Poor's or Moody's that base their judgements neither on personal experience from previous transactions nor on claimed experience of others, but on objective rating criteria. However, since these ratings are necessarily incomplete and noisy, neither personal experience nor rumors have ceased to play an essential role in most business decisions.

Despite their perseverance and apparent superiority as a basis for decision making, reputation mechanisms tend to be excessively under-specified and all subject to the intuition of the individual nodes. Trust values $\tau_{i,j}$, aggregated reputation values $\Omega_{i,j}$, and the rules according to which decisions are derived from those values have usually been poorly studied, if available at all.

With the rise of distributed file sharing and the notion of P2P networks (MHS$^+$01; SE05), reputation systems have received a more technical treatment. However, an ideal solution in which a new node enters a completely unknown network and can efficiently and accurately compute whom to trust clearly does not exist for principal reasons: No matter which rumors are taken seriously, they might be wrong, so if no further assumptions on the system can be made, a malicious node can hurt every new node in the network at least once before being effectively excluded from the resource pool. Since in most cases the network allows new nodes to join at any time, there will always be somebody to cheat.

All solutions so far proposed have made daring assumptions on the limits on adversarial power and creativity. Worse, many consistency guarantees that can be achieved by using recent cryptographic protocols for distributed computing have not been taken into account. So previous work is often both too lax and too restrictive. Furthermore, a large number of solutions attempts to achieve game theoretic equilibria (BAS03; OR94), i.e. enforce a situation in which it is rational for any individual node to cooperate, *even*

if all other players are willing to defect as soon as it gives them an advantage. A large number of successful file P2P networks (see Section 7.2) proves this assumption wrong, and suggests that game theory, at least in its most common interpretation, is too pessimistic about the behavior of nodes.

We approach the problem from a different perspective. In Chapter 8, we extend the cryptographic framework established in Chapter 2 and assemble a set of building blocks that allows us to enforce as many data consistency properties on the reputation values stored in a distributed fashion on the nodes of the network. This yields a basic model that is much more robust than the ones underlying almost all P2P reputation mechanisms we are aware of. We review a few selected reputation schemes previously published to demonstrate this point. In Chapter 9, we abandon the assumption that all players behave rational and that any situation that is not an equilibrium will rapidly spiral out of control. Instead, we assume that

1. a fraction of the network sticks with the rules, even if they mean a limited deviation from the rational behavior (in the game theoretic sense); and that

2. malicious nodes have a well-defined agenda. In other words, there is a security definition that explains what it means for an adversarial node to successfully attack the network.

We then run simulations to examine the impact of the remaining adversarial nodes on the performance of the entire network, and the performance experienced by the honest nodes in particular. I.e., an attack is not either successful or not, but it is successful to a certain degree that can be measured and bounded.

Are reputation schemes relevant? Sometimes, such as in well-established business contexts that rely more on complex legal devices and informal bonding to enforce good behavior, or in open source software projects where the feeling of having made a contribution is enough incentive for many participants, reputation is not essential to make an economic system work. Alternative to reputation have been considered as well, monetary ones being the most popular idea (see Section 7.4). But neither constitutes a reason for not being interested in reputation schemes. Both systems free of explicit internal incentives and monetary economic systems can benefit from reputation schemes and become more robust against an increase in hostility and egotism. As long as the computational and communication overhead is acceptable, this should always be a good reason to built trust and reputation into a system early on.

7.2 Application Scenarios

So far we have left the question of what constitutes a resource very much in the abstract. This is intentional, as it allows us to model a wide range of different applications that only have a few crucial properties in common. Yet in this section, we will motivate our work by listing a number of more or less concrete applications that can potentially benefit from it. Having these applications in mind should make it easier to follow the construction of our model.

Due to the high volume of research and development in this field, our list has to stay incomplete.

- **F**ile sharing. Certainly the most famous application of P2P networks, despite the legal and ethical controversy it ignited (OGS07). Started with Napster[2] by Fanning and Parker in the summer of 1999, it has produced a confusing abundance of competing protocols, software projects and products, and companies such as Gnutella[3], LimeWire[4], BitTorrent[5], eMule[6], KaZaA[7], and many others.

  There has been some controversy in academia as to what constitutes a resource in these networks. Traditional economics interprets the files that are shared as electronic goods with similar scarcity properties as rice or TV sets. This would mean that giving a file to a peer causes costs. However, files shared in P2P networks often have paradoxical scarcity properties. As soon as part of a large file is downloaded to a node, that node starts to upload this part to others. Therefore, the more often a file is downloaded, the more people are offering parts of the file, and consequently the easier it becomes available to everybody. A better way to model the resources pooled in a P2P network is to consider the bandwidth that is allocated for file transport. This is done in many practical reputation systems, most prominently in BitTorrent (see Section 8.4.2).

- **D**istributed backup and storage systems. In contrast to file sharing networks, where files are of common interest and downloaded to make them accessible to new nodes, in P2P backup solutions the participating nodes give parts of their mass storage space away to other nodes,

[2] `http://www.napster.de`
[3] `http://www.gnu.org/philosophy/gnutella.html`
[4] `http://www.limewire.com`
[5] `http://www.bittorrent.com`
[6] `http://www.emule-project.net`
[7] `http://www.kazaa.com`

and store their own data in turn on those nodes for backup. The data is usually encrypted, so the only risk involved in this backup strategy is data loss due to unreliable or malicious nodes, not breach of confidentiality.

Products like Pensamos Magic Mirror Backup[8] are intended for setting up an intra-office P2P network such that if a small number of PCs fail, others in the same office have all the data necessary to reconstruct them. Larger solutions in which the data is stored in other office buildings, ideally on other continents, are conceivable.

This application is interesting because it is one of the few in which the valuable resource is not bandwidth, but storage space. This subtly changes the nature of how reputation is established.

- **G**rid computing. Grid computing is a related and equally vivid field of research as P2P networks (FK98; BFH03; LB05). It is hard to find clear criteria to tell the two concepts apart, but while P2P networks are usually pictured a loose assembly of nodes that know little about each other, such as global file sharing networks, grids are more organized. A typical example is the Enabling Grids for E-Science (EGEE) project[9], a network of over 90 organizations that provides and utilizes resources for computation-intensive research projects such as the Centre Européenne pour la Recherche Nucléaire (CERN)[10].

  Since the values invested and expected from individual participants in grids are considerably higher than in P2P networks, monetary incentive systems or legally binding service level agreements are usually more attractive, but there may be exceptions or hybrid solutions that trade reputation for resources like CPU cycles, bandwidth, and storage space (see Section 7.4).

- **A**nonymizer networks. *Onion routing*, or *mix networks schemes* (Cha81), are distributed cryptographic algorithms for keeping communication links in a network secret from an eavesdropper. The message is rerouted via a path that is (with high probability) hard to observe from an eavesdropper's perspective, and encryption ensures that the intercepted communication on different edges on the path through the network can not be linked.

[8] `http://www.pensamos.com/mmb/`
[9] `http://www.eu-egee.org`
[10] `http://www.cern.ch`

Assume node i wants to send a message m to node j via routing nodes r_0, r_1, r_2. Every node has a key pair for public-key encryption, i encrypts m in layers:

$$c = E_{r_0}(r_1, E_{r_1}(r_2, E_{r_2}(j, E_j(m))))$$

and sends the result c to r_0, the first node on the route. r_0 decrypts the message and obtains

$$c' = (r_1, E_{r_1}(r_2, E_{r_2}(j, E_j(m))))$$

so it knows the next recipient is r_1, and it forwards c' accordingly. r_1 repeats the process, decrypts c':

$$c'' = (r_2, E_{r_2}(j, E_j(m)))$$

and forwards the (still encrypted) result c'' to r_2. c' and c'' are encrypted using two different keys. Given there is a large number of messages going through the network, an eavesdropper who has neither of the two keys will find it infeasible to link the two to the same communication link. In some schemes, the nodes accumulate a certain number of incoming messages, permute them, and forward them in a new random order. This prevents timing attacks in which the adversary simply assumes that messages take a certain time to be processed, and link incoming and outgoing messages based on the times they are coming in and out of an observed node.

There are competing adversary models for mix networks, and the solutions depend on which adversary they defend against. If the entire network is under surveillance by a powerful government, a network with very few paths of highly trusted nodes that process a large number of messages is the best solution. On the other hand, if any adversary only has access to a small fraction of the nodes in the network, a distributed routing infrastructure in which each client picks random paths that change every few minutes is more secure.

The former model is assumed in the JAP[11] project for web anonymization, the latter in the TOR[12] TCP anonymizer and the older e-mail anonymizer Mixminion[13]. A large collection of research papers can be found on the homepage of the FreeHaven project[14].

[11] `http://anon.inf.tu-dresden.de/`
[12] `http://tor.eff.org`
[13] `http://mixminion.net`
[14] `http://freehaven.net`

The resources provided by nodes in an anonymization network are bandwidth, and to a lesser extent CPU cycles (on high-volume nodes, the cryptography requires a noticeable fraction of the CPU). Another resource that could be taken into account is trust itself: A node that has either auditing certificates or a recorded history of good behavior is more valuable than one that is unknown, or even known to attack the anonymity of other nodes. However, when designing reputation schemes for anonymity networks, we face a new obstacle: There is a conflict between the interest of a node to record its good behavior for others to trust it, and the interest of other nodes to remain unobservable when routed through that node. The question whether reputation systems can possibly be used here at all is still open (ADS03).

- **M**essaging systems, groupware, server-less web servers.

  Several technologies have been developed and suggested as alternatives to the SMTP protocol (Pos82) underlying our everyday e-mail communication (KRT03; MHPD05; SMPD05). Internet telephony applications like Skype[15] or Gizmo[16] often use P2P networks for routing tasks to enable large networks with low-cost server hardware. Groove Networks[17] is a groupware suite that allows an organization of users to collaboratively process office documents, e-mail, schedules, and so forth. It is based on P2P network technology as well, and enforces simple confidentiality and integrity properties and role-based as well as user-based access control policies using cryptography.

  In principle, every application that is based on a Web server today could be redesigned to be based on a P2P network and a local client delivering that network to the same Web Browser-based user interface. Resources traded in such a network could be any of bandwidth, storage space with different persistence requirements (ranging from caching small data packets for a few minutes to integrated backup services for many months), or processing power.

- **R**ecommender systems, friends and business networks.

  All the above applications are arguably possible without P2P technology. The reasons for choosing a distributed infrastructure are performance and scalability considerations and trust. However, if the resource units that are traded involve human interaction, P2P networks are not

[15] `http://www.skype.net`
[16] `http://www.gizmoproject.com`
[17] `http://www.groove.net`

an alternative to a centralized system, but intrinsic to the application: Every human becomes a node, whether maintaining her profile on a centralized server or in a distributed application.

In fact, examples of networks where humans trade their attention, competence, time, and so on can be found on top of decidedly centralized technical infrastructures. Every large online bookstore allows users to publish reviews. Automatic recommender systems (SM95) correlate books that are often purchased by the same users, and suggests new books that have a high correlation with those purchased before.

Recommender systems can be used for any class of goods, not only books, music or movies. Adapted to e-mail, they can serve as collaborative spam filters (*"those e-mails that have been labelled as spam by many others are likely to be labelled spam by you as well"*). Friends and business networks such as friendster[18], XING[19], or LinkedIn[20], or online dating sites could provide recommender systems that decide how likely two users are to want to get together before they are even aware of each other.

If experts form groups to author a shared document set (Wikipedia[21]) or to pool their knowledge and answer each others' questions (Lycos-IQ[22]), they need to assess and register the quality of each others' contributions.

In all these applications, computing and communication resources play a role to the extent the network infrastructure is P2P. However, note that many collaborative projects are based on a traditional, centralized infrastructure and yet *form a P2P network on the user level*. This brings other resources with different characteristics into play, such as high connectivity and strong trust relationships (if a person needs to be found on XING that can do a consulting job at hand), or user time (if knowledge is to be shared).

7.3 The Adversary

The goals of the opponent are as diverse as the application scenarios. Some users might simply oppose the idea of having to contribute their own re-

[18] `http://www.friendster.com`
[19] `http://www.xing.com` (formerly OpenBC)
[20] `https://www.linkedin.com/`
[21] `http://www.wikipedia.org`
[22] `http://www.lycos-iq.de`

sources in order for the network to provide utility for all. This is called *free riding*.

If the resources are scarce and free riding is common, resource starvation may cause the death of the network. Fortunately, there is a simple solution to it: Make sure free riders starve first, and the network containing only good contributors will heal quickly.

But how do we decide who is a free rider and who is a contributor? The problem of free riding can thus be stated as follows:

> **P**reventing free riding: Given a network of peers in which each peer only knows few other peers (if any), how can we contribute resources with higher priority to those peers who contribute more?

This is where reputation finds one of its two main applications: If we can attribute a reputation to each node proportional to its contributions to other nodes in the network, we can give more to nodes with high reputation, or give to nodes with high reputation only, and free riders will starve.

One adversarial strategy in a network like this will be to tamper with the reputation schemes. So the scheme needs to protect against adversaries that attempt to boost reputation of one or several specific nodes.

But there are malicious users that have other goals than drawing resources for free. For instance, the editors of Wikipedia[23] have to deal with malicious content on a regular basis that is injected as advertisement for a person, a company, or a political cause, or simply for purposes of defamation. In file sharing networks, files containing viruses are distributed by hackers in order to create bot networks, or movies of poor quality are provided by the content industry to frustrate users into abandoning downloading copyrighted content and going to the store to make a traditional purchase. This problem becomes essential in applications where resources are costly to retrieve, whether because they are large files downloaded via a narrow channel, or whether there is an exchange of money at the same time.

Suitable reputation systems can help here, too: If by looking at a node, or a resource, we can estimate its quality with high accuracy, no adversary will be able to inject virus-infected files. (In fact, existing anti-virus software can be viewed as a simple, centralized reputation system: Patterns that are known to be malevolent are labelled with reputation -1, and all other patterns are implicitly 0. Files that have patterns with negative reputation are removed from the system, or otherwise contained.) Opinions or content contributions of low quality will be marked as such, and ignored, thus improving the overall efficiency and effectiveness of the system.

[23] `http://www.wikipedia.org`

And as before, a node compromising the reputation scheme and illegitimately increasing its reputation can render the reputation scheme useless. In summary, the question is this:

> **A**ssessing resource quality: Given a network of peers in which each peer only knows few other peers (if any), how can we estimate the available resource contributors' reliability levels and choose a reliable contributor?

These two goals of the adversary, free riding and resource poisoning, are the motivation for reputation schemes and determine the nature of the security notions those schemes have to satisfy.

7.4 Monetary Incentives vs. Reputation

Why are we even considering reputation? Our civilization has made an excessive amount of experience with monetary incentive systems (i.e., resources are contributed only in exchange for money) as tools for optimizing resource allocation. Money has been subject to many theoretical treatments of P2P networks (BKO02; BAS03; JF05). In practice, attempts have been made to grow P2P networks that run their own currency (e.g. MojoNation[24]). Alternatively, one could of course always connecting any national currency to the system via an Internet banking service such as PayPal[25]. Research suggests even anonymous digital cash is feasible (Cha91) (although so far it requires a trusted third party as well).

The volatile nature of the Internet makes it crucial for user acceptance to think about trust. This is the reason why online auction sites, being about trading goods for money, have made positive experiences with reputation systems (or feedback systems).

Often the resource in a network does not behave like gas used up by a car, but rather like radio broadcasts that are available independently of the number of radios that tune in. In auction feedback schemes, the resource is node honesty, which clearly does not diminish with higher utilisation. In file sharing networks, consumption and production are inseparably connected. Every consumer is also a contributer, and thus the files paradoxically suffer inverse scarcity, i.e. scarcity decreases with increasing consumption.

In these cases, to pay in exchange for a piece of a good does not provide the right incentives, if it makes sense at all. What we really want to know

[24] `http://www.mojonation.com`
[25] `http://www.paypal.com`

is which nodes act cooperatively, in a much more general and flexible sense than that its production is high and its consumption is low.

Market-based mechanisms have theoretical limitations (MS83): Finding optimal prices for both buyers and sellers is at least in $\mathcal{NP}$, even if a trusted third party is available.

In his case against micropayments (Odl03), Odlyzko argues that Internet economics will have to exist without ubiquitous inexpensive payment schemes for small amounts of money for the foreseeable future. As reasons for this claim he lists the benefits of bundling and flatrates that are established both in economic theory and by their popularity; wrong incentives justified by wrong scarcity assumptions and in spite of economies of scale; and the long time it takes for payment schemes to penetrate society compared to other technologies.

Reputation schemes do not have these disadvantages. Systems can be designed to also benefit users poor in reputation. Therefore, they can be easily erected and dismantled. And utilization of resources can be encouraged by simply not decreasing reputation for it.

Finally, a more principal aspect is that of control. Reputation owned by node j is collaboratively controlled by all the surrounding nodes, while money is controlled by j itself. The latter makes capital and power accumulation effects, which are undesirable for public welfare, harder to prevent. In our opinion, this relocation of control over the wealth of an individual into its economic environment is at the root of all the benefits of reputation over monetary incentives. Even if this strong claim is not shared by the reader, it makes for an interesting subject of economic research.

7.5 Game Theory

Economic and social systems that involve interactions of several players and can have good or bad outcomes for each are often formally described as *games*. Game theory (vNM07; OR94), initially developed by Morgenstern and Neumann in 1944, is the mathematical discipline of these games. It is widely used in economics, but computer science makes use of it in distributed systems research to a lesser extent as well. In the following it is not strictly required, but awareness of the basic concepts will help the reader to better understand some of the related work as well as the rationale behind the simulation model developed in Chapter 9.

A *game* consists of n *players* $p_0, \ldots, p_{n-1}$. Every player p_i has a set of *actions* $A_i = \{a_{i,0}, \ldots\}$ to choose from, and a *strategy* $S_i : \Gamma_i \to A_i$, where Γ is the context knowledge the player's action is based on. The *outcome*

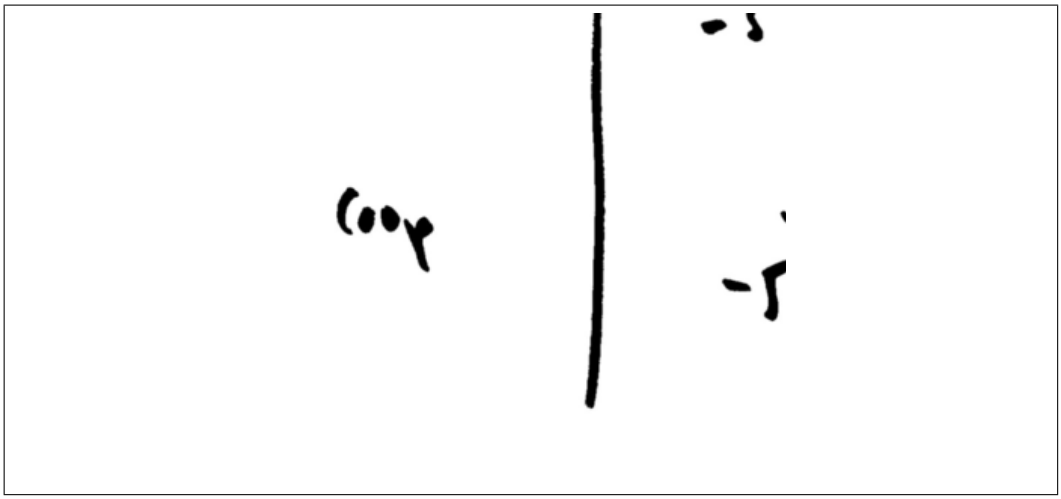


Figure 7.1: The Prisoner's Dilemma.

of *round* k is a set $O_k = \{a_{0,j_0}, a_{1,j_1}, \ldots, a_{n-1,j_{n-1}}\}$ of actions chosen by all players.

In one-round games, $\Gamma_i = \emptyset$. Each player chooses an action before the game starts and once the game starts each player makes her move independently of what the other players do. In multi-round games, each player can learn from the previous outcomes and adapt her own behavior to the social context: In round k, $\Gamma_i = (O_0, \ldots, O_k)$.

Each outcome yields a certain *utility* $U_i : O_k \to \mathbb{R}$ for each player p_i. Utility is a measure of the happiness of the players, and every player is assumed to act only to maximize its expected utility.

Perhaps the most famous example of a game in this sense is the *Prisoner's Dilemma* (PD). Two players suspected of a bank robbery are interrogated in separate rooms so that there is no negotiation between them. Each interrogator tells his suspect that there is evidence to convict one of the two, but it is not clear which one, so he offers the following deal: If p_0 witnesses against p_1, but p_1 does not witness against p_0, p_1 goes to jail for a long time, while p_0 goes free, and vice versa. If both talk, jail time is still long for both. If neither talks, both go to jail for a short time.

Figure 7.1 contains the possible combinations of actions (where defection is the act of witnessing against the other, and cooperation is remaining silent). In this simple scenario, U_i is simply the number of years in jail multiplied with -1 (highest utility is always best).

Assume p_0 cooperates. Then p_1 goes to jail for one year if he cooperates as well, but he goes free if he defects, so in order to maximize utility, he should defect. On the other hand, if p_0 defects, the choice is to cooperate and go to jail for five years, or to defect and go to jail for three years, so the

utility-maximizing choice is again defection. This game is symmetrical, so the choices, outcomes, and utilities are the same for p_0.

This is where the dilemma becomes apparent: No matter what the other player does, each player should choose to defect. If there was a way to make a deal for the two players, both would be better off, but game theory assumes that each player always maximizes utility for the outcomes in which all other players do so as well.

This insight is captured by the term *Nash Equilibrium*. A Nash equilibrium (or simply equilibrium) is a set of strategies $S_0, \ldots, S_{n-1}$ that no player has an incentive to deviate from. In a slight abuse of notation:

$$\forall S_i' : U_i(S_0, \ldots, S_i', \ldots, S_{n-1}) \leq U_i(S_0, \ldots, S_i, \ldots, S_{n-1})$$

Any change in strategy would, given all other players keep theirs, reduce the individual utility of the deviator. In the PD game described above, $(\text{defect}, \text{defect})$ is an equilibrium, but $(\text{cooperate}, \text{cooperate})$, despite its better total utility, is not. Either player would be better off when switching strategy.

Equilibria are closely linked to *Pareto optimality*. A strategy set $S_0, \ldots, S_{n-1}$ is Pareto optimal if no change in strategies can improve the total utility over all players:

$$\forall S_i' : \sum_{i=0..n-1} U_i(S_0, \ldots, S_i', \ldots, S_{n-1}) \leq \sum_{i=0..n-1} U_i(S_0, \ldots, S_i, \ldots, S_{n-1})$$

While equilibria are the *most likely* strategy sets to be witnessed in real life (as long as the assumption that players always act selfish is not falsified), Pareto optimality represent the *most desirable* strategy sets. Games with a Pareto optimal equilibrium are ones in which selfish play yields outcomes that are optimal for all players (on average). PD is not such a game.

There is an n-player variant of PD called *tragedy of the commons* (TC). It is about a village of shepherds that share a single pasture ground that can only sustain a maximum number of sheep before it is over-grazed and all sheep starve. Each shepherd benefits from any other shepherd not overgrazing, but benefits more from overgrazing himself: The profit from having sustained more sheep goes to the individual shepherd, while the damage to the common resource is distributed evenly among all. The formalization of this game has first been developed in (Har68).

The multi-round case looks better than the single-round case: Now it makes sense for a single player to play *tit-for-tat* (always copy the move of your opponent / some threshold of your opponents, starting with cooperation). If all other players do so as well, total utility will be maximal, while

individual outcome will be far from minimal and impossible to improve any further by defection (because it will make others defect). If not, he will only loose one round of defection, which is negligible for the total individual outcome over time.

Note that tit-for-tat does not represent an equilibrium. If all other strategies remain unchanged over time (no player adapts to the other player's actions), any player can improve its utility by at least this first round of defection bonus (if all other players constantly defect), and potentially much more (if other players cooperate to a certain extent). This is arguably a defect of the theory, and not the world it has been developed to describe. But on the other hand, tit-for-tat, or other forms of negotiation, require a high degree of organization. In a P2P network where it is hard to see whether one is talking to the same player under many different aliases, or to different players under the same alias, this degree may be out of reach. And if it is, the PD may turn out to be the most suitable model after all.

Game theory is often criticized for its assumption of the homo economicus, an unconditionally rational and selfish player. There are countless situations in which individuals act against their selfish interest, and game theory does not account for that (unless what looks like acting for the benefit of others is in fact just optimizing the game's outcome according to some highly unusual utility function). P2P applications in particular are a great source of such examples.

But game theory can be interpreted in at least two different ways. The first one, that no individual would ever act other than selfish, is not very plausible. But the second one, that there are certain forces that drive systems of social and economic interactions into certain stable states, is easier to believe. That there may be other forces possibly reduces the predictive power of a game theoretic model (since it only explains a system to a certain extent), but does not invalidate any of its assumptions.

Game theory and P2P networks

Game theoreticians have tried to design adversary-proof P2P systems, usually with disappointing results. We now look at two symptomatic examples, one that attempts a constructive result and one that claims defeat of the P2P paradigm, and reason that game theory is not suitable because a theoretical solution is out of reach, just like theoretical solution for secure outsourcing of database services to an adversarial service provider is impossible. In the remainder of this thesis, in contrast to what we did in Part I, we will see that satisfactory solutions do exist in a weaker sense: We do not build a system that is secure under the game theoretic assumption that everybody

always acts selfish, but instead we build *what-if machines* (RKCG02) that allow us to select plausible levels of hostility and then assess the impact on the performance of a system.

Equilibria. A model of a P2P system based on TC is used in (FLSC04) to examine several variations over tit-for-tat games. A node decides whether to cooperate or defect with a given node based on the complete and noise-free history of previous actions of that node. This essentially assumes a solution to the problem we are addressing in this part of the thesis since the biggest challenge in designing reputation systems is that nodes have no way of being sure which reputation information is signal and which is noise. While this is a legitimate approach, the P2P networks used here suffer from unnecessary as well as from implausible assumptions:

1. It has untraceability not only of players (IP addresses can change any time), but also of resources: A resource that reaches a node does not allow for reliable identification of the sender. As we will see in Chapter 8, this is unnecessarily pessimistic and rules out many obvious incentives. In particular, it does not allow to punish sabotage directly, since it is impossible to identify the source of malicous communication.

2. As a consequence, adversaries need to be weakened to an extent that is not very convincing. In particular, the fact that nodes are capable of lies is considered, but essentially rejected as irrelevant. But while it is true that both liars and untraceability of defection provide an insufficient basis for game theoretic equilibria, inconvenience is in turn an insufficient justification for denouncing a scientific assumption.

PD/TC. In (AH00), Adar and Huberman try to give empirical evidence that P2P applications are subject to TC and that free riding is the most popular strategy. They provide statistics of node responses to different types of requests in an early version of the Gnutella network (RFI02), and argue that negative responses are due to defection.

This approach suffers from several flaws ranging from errors in the representation of the Gnutella protocol up to misconceptions about the nature of the application. We only name a few. (1. has been pointed out independently in (GF03), along with a number of interesting philosophical arguments.)

1. *Unrealistic utility function.* The authors claim that

   > ... it appears rational for people to download music files without contributing by making their own files accessible to other users.

However, it is more realistic to assume that upload does not decrease the utility of a player, but in fact slightly increase it, because the node that I upload to does not have to download from somewhere else, so there is a chance that I will get better download performance because there is one competing client less.

2. *IP addresses are used as identity tokens.* Most Gnutella clients use dynamic DSL connections and change their IP every few hours, but at least once every day. IP addresses are therefore no suitable identity token for assessing cooperation, especially not if there is no clear distinction between node unavailability and node defection: A node may still be available after having obtained a new IP address, but the figures will mark it as defective.

3. *Over-generalizations.* Gnutella is an old system, and one of its many shortcomings is that it does not have an incentive mechanism. However, there are systems like BitTorrent that do (see Section 8.4.2). A flaw in a protocol is not the same as a flaw in any possible protocol serving a specific purpose.

In Section 9.2.3 we will see how PD-like settings can lead to altruistic behavior under certain conditions. So not only is it difficult to establish that an existing P2P applications is subject to prisoner's dilemma / tragedy of the commons, but it may be feasible (although very challenging) to design novel P2P networks for which it can be established that they are immune.

Chapter 8

Building Blocks and Related Work

Integrity of reputation information in distributed, unreliable information systems poses a few peculiar problems. In this chapter, in analogy to Chapter 2 in Part I, we collect the building blocks to solve these problems.

Some of these building blocks have been well-known for many years, but some are quite recent, and have therefore enjoyed little consideration in the design of existing P2P networks. The extensions to the blind signature schemes in Section 8.1.2 are original results first published in this thesis.

In the first part of this thesis, we have already given a comprehensive introduction to the field of cryptography. Back then, we were mostly interested in data confidentiality, whereas now we have to put a stronger emphasis on data integrity. Nevertheless, the material presented there remains the groundwork of what we are going to do in the following, so at this point, those readers who have skipped Part I may want to have a quick a look at the material in Section 2.3.

8.1 Identity

In every distributed system, the communicating parties need identities, and these identities can best be established by cryptographic means. In Part I of this thesis, we have considered two parties, Murat and Chantal, and based our considerations on SSL/TLS (DA99) as a solution that is both robust and readily available in existing IT systems.

The cryptographic structure of TLS is that of a two-layer (public-key) digital signature scheme (see Definition 2.21). Both Murat and Chantal have a pair of private and public key. The own private key is used to sign all

outgoing data, and the other party's public key is used to verify all incoming data.

Since Chantal and Murat have a long-term relationship, we have said little about key agreement. In the database outsourcing scenario, we can safely assume that during contract agreement, finding some secure means of exchange of public keys will not be the biggest obstacle. If no other means can be found, there are companies that provide this service known as *Public-Key Infrastructure (PKI)*. A PKI is a network of service providers and parties holding or verifying keys in which the service providers make sure that all public keys are held by the organizations and individuals that claim to hold them. This is done by means of *certificates*, or documents containing a public key and the name and address of the key holder. These certificates are in turn signed by the PKI service providers.

In the case of P2P networks, however, this is infeasible for at least three reasons:

1. Too expensive. Certificates require manual verification of identity documents, which results in high fees compared to the expected benefit of running a P2P application.

2. Too cumbersome. The P2P application software should be able to run out of the box, without the need to purchase PKI certificates, submission of other identity documents, etc.

3. Not anonymous. PKI services usually bind the public key to a person's or organization's true identity. However, in many P2P applications this is neither technically necessary nor desired. There are not only the obvious reservations of users of file sharing networks regarding legal hazards caused by intellectual property rights issues. The P2P applications we are having in mind here cover a far larger class of applications, including community portals covering health issues, communication networks for political activists, online dating networks, and many others. All of these allow for a legitimate desire of its users to remain anonymous, or at least pseudonymous.

Therefore, although it may be applicable in some selected applications, in our work we reject PKI as a solution for equipping P2P applications with identity. Instead, we impose the following requirements:

1. A user can choose to link several of her actions to each other under an assumed name, or *pseudonym*. These links between pseudonym and actions can be verified by others.

2. A user can decide to discard a pseudonym at any time, and create a new one that can not be linked to the old one.

3. Despite these facts, free riding can be kept low, and lemon (i.e., defective) resources can be identified before costs arise by consuming them (see Section 7.3.)

Requirements 1. and 2. are easy to satisfy. All we need is a digital signature scheme *without* PKI. Each message that is sent through the network is signed by the sender, and the network provides some means of looking up the public key that belongs to the signature of a packet. This can be done using a subset of TLS. That this is feasible has been demonstrated in many applications, see e.g. the TOR network that provides TCP-level sender and recipient anonymity (DMS04).

The challenge consists in the third one, namely that despite the fact that there are no visible links between actions and users, no adversary can effectively launch any attack against the P2P application that could be prevented if such links existed. In the following, we give a list of counter measures against such attacks. But before we can discuss these, we need to talk about the attacks first.

8.1.1 Forms of Identity Abuse

Pseudonymity implies that nodes are anonymous, i.e. that no pseudonym can be linked to any other, or to a "true identity" such as a passport number or residential address. We now present a categorization of attacks made possible by pseudonymity-preserving identity management.

Dropping unpopular pseudonyms

This is the easiest and most basic form of attack. If a user has established a reputation as an adversarial peer, she simply drops her present pseudonym and creates a new one. To all other players, it now appears as if there was a new user who has done nothing wrong, and the rogue user will have the opportunity for further misbehavior. An extensive game theoretical analysis of the effects of this freedom under various assumptions has been delivered in (FR01).

To prevent identity drop, one could treat new peers as rogues, waiting for them to make an effort and convince everybody of their honesty. The problem with this approach is that new nodes will experience poor quality of service, which may result in a low acceptance of some new application and ultimately cause its death. Furthermore, some applications, e.g. the file

sharing tool BitTorrent (see Section 8.4.2) would be entirely dysfunctional if new peers would not be given some credit: If all server peers wait for a new client peer to provide a piece of a large file that it wants to download because it does not have it yet, no data will be distributed at all!

In BitTorrent, the problem is solved by setting the absolute lower trust bound to ϵ for some small $\epsilon > 0$ (where 0 means that the node has no trust at all and does not deserve cooperation). Every new peer thus obtains for free a small piece of a file it wants to download, and can use it to start trading for more. This way, new nodes can gather positive reputation quickly. This works well for high-performance file sharing, but in other applications, more advanced protective measures may prove more effective.

Sybil attacks

Instead of creating a new pseudonym each time the old one becomes unpopular, an adversary may also create a huge army of pseudonyms that act in a concerted attack like they belonged to an equally huge number of independent users. The act of a single user (or small group of users) creating such an army is called a *Sybil attack* (Dou02). Formally,

Definition 8.1 (Sybil Attack)**.** *Let (V, E) be a P2P network (at a certain point in time). Further, let U be the set of users, and let $f : V \to U$ be a mapping of nodes to the users that control the nodes (respectively at that same point in time). Then, a* Sybil adversary *is a user $u \in U$ such that $|f^{-1}(u)| > 1$. The nodes in $f^{-1}(u)$ are called* Sybil nodes*, or* Sybil*.*

Sybil attacks have become common since file sharing networks are sabotaged by the content industry and used to spread malware. If a P2P network is designed under the assumption that there is a bijective mapping between users and nodes, an adversary can easily convince the community that a resource is highly popular because of its integrity and high quality, whereas really it is only provided by a single malicious user.

Reputation systems can suffer particularly from Sybil attacks. If an adversary commands a large number of nodes, the effect her claims have on the reputation of any node is likely to be considerably higher than that of an honest user (see the next two forms of abuse).

A simple and very popular defense against Sybil adversaries is to use the address space of the underlying network as a name space for the nodes in the P2P network. For example, if the P2P network is based on TCP/IP directly, then each node assumes its IP address as its pseudonymous identity. This violates the requirements we have listed in the introduction to this chapter in two ways. First, it does not account for anonymity, since IP addresses can

often be mapped to the real identity of a user. Second, this mapping can be easily corrupted by an adversarial user by launching the attack via a bot network (see e.g. (LAAA06)), rendering the defense ineffective.

In principle, Sybil attacks can only be prevented if the adversary's resources are bound in some way (Dou02). Just as when considering outsourcing of encrypted databases, there is very little to be done in the model that we would *like* to use to define our notion security.

However, in P2P networks, we will show that it is much easier to find models that provide heuristic security that is still acceptable. Recently, an anonymity-preserving, robust defense against Sybil attacks has been proposed (YKGF06). We will describe this solution in Section 8.1.4.

False lack of trust and denial of service

Naturally, in any robust and secure reputation system we must assume that nodes can lie about the events they witnessed. In particular, it is possible that a node attempts to reduce the reputation of another node by complaining about its performance. In a tit-for-tat style reputation system that correlates reputation with service quality, this may result in resource starvation of the attack target, better known in computer science as denial of service attacks.

The effect of lies is increased if the adversary launches a Sybil attack at the same time. But it may also be effective if there is a large enough group of uncoordinated adversarial users. For example, a portal for libertarian political discussion may be sabotaged by a large number of users under the command of a fundamentalist religious leader.

False trust

Equally damaging, users can trade undeserved positive reputation. In some cases, the user has to tolerate some cost for providing resources to others. This may not be the case in file sharing networks, where upload bandwidth is free and unused for the most part. But in others, such as distributed knowledge bases such as Wikipedia,[1] resource contribution may cause a user to spend considerable amounts of his spare time doing research and authoring documents.

If resources are costly, and reputation is linked to the amount of resources provided to others, there is an incentive to lie about one's resource contributions.

Again, a single adversary can do this by launching a Sybil attack first, and then having her Sybils claim to have received resources from each other.

[1] `http://www.wikipedia.org/`

In a minority of applications (Wikipedia being one), resource contribution is public and can be witnessed by all peers. This makes it harder to establish false trust. However, if Wikipedia was extended such that expert opinions can be requested on confidential topics, resources would be delivered from server peer to client peer in private. This is what happens in file sharing, backup systems, or social networking applications. In those cases, confidentiality requirements, but also performance considerations alone, make it hard to render resource contribution events publicly observable.

8.1.2 Blind Signatures and Trusted Third Parties

To reiterate: Identity is based on a public-key signature scheme. Every user exclusively owns a private key, and proves possession with the associated public key. In this way, the actions of a single user can be all associated with the same public key, and trust relationships can emerge *as long as the owner of the secret key desires it*. If he has accumulated bad reputation, he can choose to drop the identity at virtually no cost, rendering negative reputation ineffective (FR01).

Fortunately, there is a simple way to prevent this from happening using standard cryptography and a trusted third party (TTP). To our knowledge, it has been proposed first in (FR01) (but it is very straightforward and may have been mentioned earlier).

Before we can define the solution, we need another cryptographic building block derived from signature schemes (see Definition 2.21): Blind signatures (Cha82).

Definition 8.2 (Blind Signature Scheme)**.** *A* blind signature scheme *is a tuple* (G, S, V, b, b^{-1}) *of five PPTMs for key generation, message signing, message verification, message blinding, and message unblinding, respectively, such that*

$$\begin{aligned} G &: 1^N \to \mathcal{K} \\ b &: \mathcal{K} \times \mathcal{M} \to \mathcal{M} \\ b^{-1} &: \mathcal{K} \times \mathcal{S} \to \mathcal{S} \\ S &: \{d \mid (e,d) \in \mathcal{K}\} \times \mathcal{M} \to \mathcal{S} \\ V &: \{e \mid (e,d) \in \mathcal{K}\} \times \mathcal{M} \times \mathcal{S} \to \{ok, error\} \end{aligned}$$

and such that for all blinding keys $k \in \mathcal{K}$*, signing keys* $(e,d) \in \mathcal{K}$*, and messages* $m \in \mathcal{M}$*,*

$$V_e(m, b_k^{-1}(S_d(b_k(m)))) = ok$$

To get an intuitive grasp of what blind signature schemes do, we return to Murat and Chantal, our protagonists from Part I. Murat now has a message m, and wants Chantal to sign it. But he does not want her to look at it. If the two where doing business in the year 1928, Murat could write m on a sheet of paper, cover it with a sheet of carbon paper, and put the two sheets in a sealed envelope. Chantal would sign the envelope and hand it back to Murat without breaking the seal. Murat could retrieve a piece of paper that states m (in ink) and is signed by Chantal (in carbon), and Chantal would never know what she signed.

The cryptographic construction is of course slightly different. Murat uses a function to blind m. The signature function produces a blinded signature from a blinded message. By unblinding the signature using b^{-1}, Murat obtains a pair of unblinded message and unblinded signature. Verification then works just as in unblind signature schemes.

To construct a blind signature scheme, we can use RSA again once more and simply swap encryption and decryption. Let (G, E, D) be RSA encryption as introduced in Section 3.1.2. Then (G, S, V) is a signature scheme with

$$\begin{aligned} S_{(n,d)}(m) &:= D_{(n,d)}(m) = m^d \\ V_{(n,e)}(s) &:= E_{(n,e)}(s) = s^e = m^{de} = m \end{aligned}$$

d cannot be computed from the public key (n, e). Hence, just as decryption is impossible without d, no valid, verifiable signatures can be created without d.[2]

For blinding, we once more exploit the fact RSA is homomorphic with respect to multiplication:

$$(rm)^d = r^d m^d \mod n$$

Let Murat pick a random number r (such that $\gcd(r, n) = 1$), keep it secret from Chantal, and compute

$$b_r(m) = r^e m \mod n$$

Chantal cannot compute m from $b_r(m)$. She knows e, but lacks r, so she needs to compute factors from the product $r^e m$, which is hard by the RSA assumption. However, she can blind-sign anything:

$$s = (r^e m)^d$$

[2]Technically, in order to satisfy our definitions V would have to accept m, s as arguments and decide whether $m = s^e$. We chose this representation for compactness and clarity.

Algorithm 7: Secure pseudonyms using a TTP.

Input: A blind signature scheme (G, S, V, b, b^{-1}); a true identity i; a pseudonym v; a set I of previously served users
Output: A signature s for v; an updated user set I

Murat sends $(i, b(v))$ to Chantal
if $i \in I$ **then**
  fail
else
  Chantal computes $I := I \cup i$
  Chantal sends $S(b(v))$ to Murat
Murat computes $s := b^{-1}(S(b(v)))$
return (s, I)

Further, Murat can unblind the signature because he knows r:

$$b_r^{-1}(s) \;=\; (r^e m)^d \cdot r^{-1} \;=\; r^{ed} m^d \cdot r^{-1} \;=\; m^d \mod n$$

So without Chantal having seen m, Murat has received a valid RSA signature for m.

A formal treatment of security of blind signatures can be found in the historical research article (Cha82). Roughly, a blind signature scheme (G, S, V, b, u) is secure if (G, S, V) is a secure signature scheme and (G, b, u) is a secure encryption scheme. In an interesting recurrence of the leitmotiv of Part I, observe that (G, S, V) is homomorphic with respect to (G, b, u).[3]

Blind signatures have been introduced for untraceable payment schemes (Cha82; Cha85; Cha92), but have turned out have many other uses as well. One of them is the one we are interested in: To have a TTP sign an identity token for a P2P network without revealing it. Chantal receives a pair of true identity and blinded pseudonym, and records the true identity as already served in order to avoid signing multiple signatures for the same user. Murat can unblind the signature and thereby obtains a certificate by Chantal that his pseudonym is not Sybil. Algorithm 7 summarizes this procedure. The condition in Definition 8.2 ensures that the resulting signature is valid.

[3]Turning this understanding into definitions is complicated by the fact that (G, S, V) is not necessarily a signature scheme. Without the blinding operations, it may violate the condition that valid signatures are verified successfully.

Extensions

As one of the contributions of this thesis, we propose several extensions and variations of this basic algorithm. First of all, the user set I need not be permanent. If a user lost her pseudonym due to carelessness or identity theft, her demand to have another pseudonym signed would be legitimate. However, the overall scheme needs to make sure when accounting for this legitimate requirement that security is not compromised. This can be done in two ways:

- The signing authority could include a time stamp with the pseudonym in the signature and thereby expire the pseudonym after a certain amount of time. This would require all peers to occasionally re-register for new pseudonyms, but it would both make sure that no two pseudonyms are ever held by the same person and that pseudonym loss only causes temporary exclusion from the network.[4]

- Additional pseudonyms could be signed for a fee that grows exponentially larger with each additional pseudonym. Instead of money, proofs of work are conceivable as suggested in the next Section.

  This idea is inspired by a popular, if rather utopian, approach in spam prevention: If each e-mail costs an insignificant amount of money, normal users might be happy to pay while spammers will not see any return on investment due to the large volumes of ineffective spam mails they have to send in order to get one hit. Variants of this scheme have been proposed in which the receiver of an e-mail can decide to send the money back to the sender if he does not consider it spam.

  Alternatively to the hybrid approach in which the first pseudonym is free and all others are priced based on the track record of the user, a purely monetary solution could be used and the registry I abandoned entirely. This would provide better anonymity and require no user data management on the side of the TTP.

  The drawback is that it involves either real money, which complicates IT systems enormously and comes with a plethora of new security requirements. Alternatively, if no real money but hash cash or captchas are used, the question of effectiveness has not been agreed on by the community (see below).

[4]The former assumes that users do not deliberately give away their pseudonyms to others who already have one. For more on this, consult Section 8.5.

Another extension allows for using the TTP for a number of different pseudonyms for different purposes. Some applications may account for different roles that require different identities held by the same user. Even desktop operating systems know different users and roles, e.g. "administrator" and "user". Further, the TTP could be run by a governmental, administrative, or profit organization and provide pseudonyms for any P2P network that is designed to use its service.

Partially blind signature schemes (AF96; MB02; CZL07) are schemes for signing a message (m, m') such that the signing party learns m, but not m'. This allows for a TTP that records pairs (i, m) of true identity and application context, and that issues one unique pseudonym per such pair, not per identity.

Partially blind signatures also give way to quality of service constructions: The TTP can let the user decide which payment scheme to use and how much to pay, and the pseudonym used in the P2P network can then contain a signed record of the user's choices. The more a user has paid, the better he is treated by the other peers. [5]

Note that the basic system and all its variants can be used as incremental extensions of a plain unprotected pseudonym system. They introduce a superior type of pseudonyms, and pseudonyms of this type are more valuable to the holder because others can see that the trust established with it is more meaningful. However, there is no reason not to allow for the common, uncertified pseudonyms at the same time, in the same system. The choice whether to upgrade or not on the one hand, and how to treat ordinary pseudonyms compared to certified ones on the other, can be left to the user.

A Case against TTPs

TTPs have several disadvantages. Unless carefully designed, they represent single points of failure, or at least a small number of hot spots, that can be attacked in order to tear down the entire network. Further, the operation of TTPs often implies that users need to pay for it (either in terms of consuming advertisement or in terms of membership fees). This erodes two of the main benefits of P2P networks.

Failure of a TTP can happen in two ways: It may have down times due to an attack, or it may be corrupted by an adversary, and participate in an

[5] If such a record was one-shot, i.e., good for a single transaction only, it could be considered electronic cash. What we want is a re-usable identity token that gives a certain level of confidence to the verifier. Electronic cash is more complicated than that and often reduces anonymity, since it needs to be make sure that no such record is spent more than once (Cha91).

attack, giving the adversary the power to hand out certificates with contents of her choice. On the other hand, if the system is fully distributed, its security can often be based on a majority of honest nodes, so the adversary needs to corrupt a large number of nodes instead of just one.

While it is still interesting to understand hybrid systems in which certain high-security tasks such as provision of unique pseudonyms are delegated to a centralized sub-system, our ideal is a purely distributed network. Luckily, there are distributed alternative approaches for robust identity management that only require local a priori trust. In Section 8.1.4, we will describe such a method to enforce tight bounds on the number of Sybil nodes based on the assumptions about the number of honest users, not the nodes. Also, future advances in cryptography and distributed algorithms research may generate means of removing the need for a TTP from the above schemes altogether.

8.1.3 Hash Cash and Enforcing Exclusivity

A reputation value Ω_j for some node j of 0 means that the network is undecided on j, or knows nothing about it. Positive Ω_j means that j has earned some respect and above-average treatment, and negative Ω_j means that j has defected in some way and should be punished.

Let us return to the setting without TTP, and consider a different approach: If j can not be punished, because each time its reputation drops below 0, it simply disappears, and the user creates a new pseudonym, then an option is to simply punish everybody in advance, at network entry time, by decreasing reputation of new nodes by an amount corresponding to the severity of the punishment. This way, a user whose pseudonym j ends up at 0 can still discard j, but that means it has to take punishment again. Therefore, users have an incentive to stay in the network even when they enjoy moderate negative trust, because they are worse off with a new pseudonymous identity. See Figure 8.1 for an illustration.

Proofs of Work

Hash cash (Bac02) has been proposed as one such punishment method that new nodes can take at startup time. As the name suggests, it is a method of "paying" not with money but with solving a challenge based on hash functions (see Definition 2.17). The challenge is easy to impose on a new node by the others and the solution is easy to verify, but it is hard to compute. This way, with little effort for the nodes already in the network, the new node is forced to spend computing resources and, more importantly, time, before it is let in.

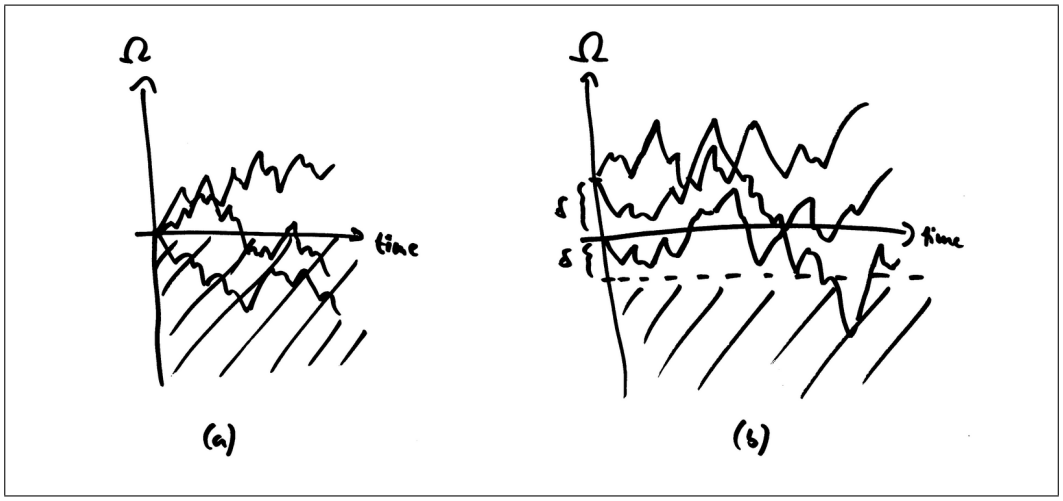


Figure 8.1: Proofs of work. With initial reputation 0, identities with negative reputation are dropped by rational users (a). If the initial reputation is increased to δ, an incentive is imposed on the holder to keep the pseudonym until its reputation has dropped to $-\delta$ (b). This makes it possible to punish rational users by giving them negative reputation.

The hash cash challenge uses the fact that hash functions are hard to invert: For a given hash function h and output x, brute force search over all inputs is the best known method to come up with an input y such that $h(y) = x$. The challenge consists in finding such a y for any x' that has a prefix of length l in common with x. The amount of work can be set by the challenger by choosing an appropriate l.

Challenges based on other mathematical structures have been considered, for example factorization of the product of two primes chosen just large enough to keep the user busy for the desired amount of time, or discrete logarithm problems are most popular. In principle, every problem in computer science that is considered infeasible, but that has an algorithm to efficiently verify a given solution, can be used. A large number of such problems can be found in $\mathcal{NP}$ (HU00).

In some articles, the outcomes of these challenges are called *proof of work*, since their point is that everybody can see that the new node has accepted its punishment. Among other things, proofs of work have been used to build incentives for cooperation in P2P routing protocols (BB03; Buc06; BBvdW06). We will come back to this in Section 8.4.1.

Captchas

Proofs of work can be carried out in a P2P network without any human intervention. In order to solve a challenge, a new node requires system resources, mostly time, and this alone is hoped to reduce the amount of one-time pseudonyms created by a single user.

There is a slightly different kind of challenge that does not require computer resources but human interaction. These challenges are Turing tests (Tur50), or AI problems that are considered hard and can be used to decide whether the agent accepting the challenge is human or machine: If the challenge is solved correctly, the agent must be human.

These challenges are known as *captchas* (vABL04), or *Completely Automated Public Turing Test to Tell Computers and Humans Apart*. The most prominent captchas can be found on e-mail web services to prevent spam bots from registering new addresses automatically, and often consist of a few distorted characters that cannot be identified by today's OCR software.

In P2P networks, captchas can often be used as generic proof-of-work schemes. A user that wants to create an army of Sybil nodes has to go through the manual work of solving many captchas, and might decide to rather cooperate and obtain the good reputation. Further, it has been suggested to use captchas to prevent access to sites for search engine bots. This may be desirable in some P2P applications as well. Spam will likely to be a problem in all data networks of sufficient size and popularity, and captchas have effectively been used in various ways to reduce the effectiveness of spam attacks.

Putting it into Perspective

Captchas have one severe weakness: Users are willing to solve them to get access to services on the Internet. Therefore, an adversary that is dedicated enough to set up a service (fake or real) can proxy the captchas she receives from the attack target to this site, and have them solved by her own users. This can be done automatically, and if the site has a sufficiently high traffic volume, with only insignificant delay. This reduces the applicability of captchas to low-security situations in which no dedicated adversaries have to be expected.

When using computer-solvable challenges, there is another, more subtle problem. The parameters of the challenge have to be chosen such that (a) the honest verifier (or group thereof) does not need to *work too hard* to verify all the challenges that have been solved, and (b) the solver still needs to *work hard enough*. These parameters vary between applications and implementa-

tions, but this window of useful parameters is arguably empty in the scenario where each e-mail is only forwarded if the sender presents a proof of work (LC04): Either honest users are punished too much, or malicious users are not punished enough. This is not necessarily true for other applications, but makes the point that choosing parameters is a tricky business.

This extends to techniques of outsourcing the creation of the proofs of work from adversarial solvers to unsuspecting third parties. However, solving hash cash challenges is easier to outsource than solving captchas, since the adversary only needs to purchase a bot net instead of making humans do his bidding.

We believe that proofs of work techniques, both those based on computational and those based on AI problems, are interesting tools. If used with care, they can amplify the effectiveness of other security precautions without imposing too much cost on honest users. (Buc06) gives evidence for this belief and suggests interesting applications.

8.1.4 Graph-Theoretical Approaches

Younger approaches to hardening identities are based on graph theory. They do not require a TTP, and do not have the problems of proof of work schemes.

In (BK05), instead of allowing peers to issue their identity tokens individually, a distributed PKI is erected that validates identities based on data derived from the physical location of the assigned peer. This provides a limited defense against Sybil attacks, but its effectiveness is reduced even by minor node mobility (such as laptops moving between office, customers, and an apartment). Worse, it assumes that an adversary controls a small number of nodes in the network. As long as bot nets are readily available, this is a major deficiency.

(CF05) goes one level up and proposes a way to limit the damage that can be done to the reputation scheme by Sybil nodes, without attempting to limit their number.

As the most recent and one of the most powerful defense known to us, we discuss SybilGuard in a little more depth in the following (YKGF06).

The Idea. SybilGuard is based on a strong but sparse social network. A social network is a graph (U^s, E^s). The nodes represent users of the P2P network, and edges represent personal, off-line acquaintances. A small fraction of the users in U^s is malicious and attempts to launch a Sybil attack in order to outvote the majority. Assume there is exactly one adversarial node $u^* \in U^s$ (SybilGuard does not impose this restriction on its applications).

Social networks have certain interesting properties (New03; NBW06). In particular, they tend to contain no or few "small cuts", i.e. clusters of

nodes with many internal edges between each other, but few external edges connecting them to the rest of the graph.

Let (V, E) be the projection of the social network into a P2P network of nodes V. If all users in the social network are honest, then (V, E) is isomorphic to (U^s, E^s) (there is a bijection between users and nodes). Therefore, the nodes span a small world graph just as the users do. On the other hand, assume u^* creates a large number of Sybil nodes $V^* \subseteq V$ and creates edges between them. Since honest users only accept one node key from every acquaintance, the nodes in V^* can have only g trust relationships to honest nodes, where g is the number of acquaintances of u^*. Therefore, (V, E) has a small cut between V^* and $V - V^*$: Honest nodes are acquainted with each other with a certain density following a uniform pattern, and Sybil nodes among themselves can model this pattern. But paths from Sybil nodes to honest nodes all go through a small number g of bottlenecks, so called *attack edges*.

SybilGuard is a distributed algorithm that establishes an equivalence relation on a subset of V of *accepted nodes* with the following properties:

1. Nodes that are accepted are likely to be honest. Nodes that are not accepted are likely to be Sybil.

2. The number of equivalence classes containing Sybil nodes is limited by the number of attack edges g.

3. The size of any equivalence class is limited by a small constant w that is defined as a system parameter. (For one million nodes, the authors suggest $w = 2000$.)

The algorithm assumes the perspective of a node that attempts to fight Sybil nodes. The equivalence classes differ from node to node (and so do the few false positives, and therefore no node suffers considerable loss of quality of service due to probabilistic errors). Nevertheless, the above two requirements hold for all nodes.

Application. In voting schemes, if the total number of equivalence classes is more than twice as large as g and all honest users agree on which vote to cast, a majority vote will always produce the desired result. Such votes could be called on unambiguous questions, e.g. whether an observable event has taken place or not. If not all honest users agree, the ratio of honest to Sybil equivalence classes must be larger.

For instance, in distributed backup applications, if there are at most g corrupted equivalence classes, it is enough to store $g + 1$ backups in order to be able to retrieve at least one later on. (If integrity of the backup is not

apparent from an intact copy, it needs to be established by a majority vote, and the number of copies needs to be $2g + 1$.)

In Section 9.2, we will use SybilGuard as a justification for imposing bounds on the fraction of adversarial nodes in our model. If Sybil attacks were possible, this assumption could be violated easily.

Equivalence class construction. In a nutshell, a node establishes the equivalence classes by projecting a *random route* of onto (U^s, E^s). A random route is similar to a *random walk*, i.e. a path that is initialized by the source, and extended by every next node choosing an outgoing edge (and thereby successor node) at random. The difference is that in order to generate a random route, a node does not choose edges at random, but based on a random permutation chosen at system boot time.

Let (V, E°) be the extension of (V, E) that contains all edges along which communication takes place in the P2P application. Since E° is considerably larger than E (which is isomorphic to E^s), there are a lot of nodes to which no direct acquaintance has been established. These nodes are called *suspects*, and the node that runs the SybilGuard algorithm to establish an equivalence relation on suspects is called the *verifier*. In order to do that, the verifier first contacts all nodes on a random route of length w.[6] With high probability (see the remarks on loops and attack routes below), the route contains neither loops nor attack edges and remains entirely within the set $V - V^*$ of honest nodes. This means that although the verifier has no control over the other nodes on the route, he can trust that they will behave according to protocol. Only attack nodes deviate from that behavior, but no attack nodes will be reached.

If the verifier wants to determine the equivalence class of some suspect, it searches for a crossing in the two respective random routes. If a crossing is found, the nature of random routes imposes a number of useful invariants on the state of the network (see Figure 8.2).

1. **C**onvergence. Random routes are *converging*, i.e. two routes that enter an honest node over the same edge will leave it over the same edge as well (they are both treated by the same fixed, if random, permutation).

2. **E**quivalence classes. Let $v_0, v_1 \in V$ be two nodes (honest or Sybil). v_0, v_1 are said to be equivalent (belong to the same equivalence class) if v_0 lies on the random route of v_1 or vice versa. If v_0, v_1 are suspects, a verifier $v \in V - V^*$ decides whether they are equivalent by looking at

[6]For redundancy, it may be indicated for a node to have multiple random routes (up to the number of outgoing edges). We are ignoring this complication and assume a unique route in the following where not stated otherwise. See the paper for further details.

the crossings between their random routes and its own: If v's random route is touched in the same node via the same edge by v_0, v_1, the two are equivalent.

3. **T**otal number of equivalence classes is $d \cdot w$. Let $r_0, \ldots, r_{w-1} \in V - V^*$ be the random route of verifier r_0, and let $\deg(\cdot)$ be the node degree (the social network and therefore (V, E) are undirected and in-degree equals out-degree). The number of r_0's equivalence classes is $\sum_{i=0\ldots w-1} \deg(r_i)$. In an approximation, assuming equal degree d for all verifiers, the number of equivalence classes is $d \cdot w$.

4. **N**umber of Sybil groups is g. An equivalence class that contains Sybil nodes is called a *Sybil group*. Because of the convergence property, each Sybil group requires an edge in the social network between u^* and some honest node u: If two random routes reach the same $u \neq u^*$ over the same edge, they converge and are considered equivalent by all honest nodes. Hence the number of Sybil groups is bounded by g. (This bound exploits the assumption that the verifier's path lies entirely in $V - V^*$.)

5. **S**ize of equivalence classes is w. The only way for two nodes to fall into the same equivalence class is to lie on the same random route. Since all such routes have length w, the size of equivalence classes is bound by w. As soon as $w + 1$ Sybil nodes attempt to use the same attack edge to cross the cut with their "random" routes, the honest node located at the attack edge knows that something went wrong and its neighbor must be Sybil. Therefore, this bound holds for both honest and Sybil equivalence classes.

The authors dedicate a section of their paper to the treatment of dynamic networks. This is beyond the scope of our models, so we will not discuss this here. But we have to sketch how certain undesired forms of random routes can be dealt with.

Loops. It is possible that a random route enters a certain node via a certain edge twice, with the effect that the route is less then w nodes long. Luckily, due to the convergence property, any loop must contain the source of the random route. If the outgoing edge is identical for two places in the route, then the incoming edge must be, too. By a recursive argument, this proves that if any node in the route lies on the loop, then so does its source node. Therefore, loops are both unlikely and easily detectable and can be avoided by the source choosing an other outgoing edge to start the random route with. (Also, the only harm that is done by loops is that intersections

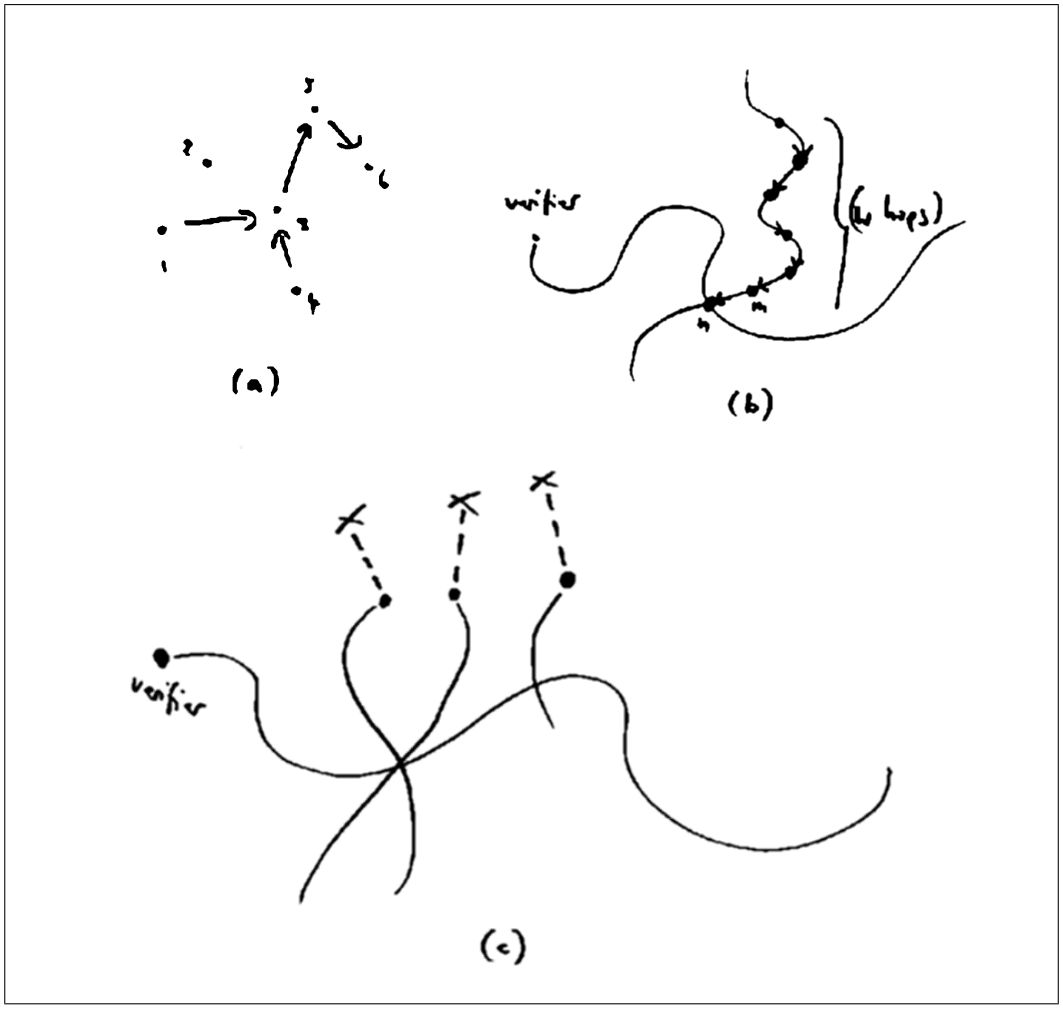


Figure 8.2: Random routes in SybilGuard. (a) Convergence: The two routes $a = (1, 3, 5, 6), b = (4, 3, 5, 6)$ share edge $(3, 5)$. If a proceeds to 6, b cannot proceed to 2; (b) equivalence classes: The equivalence class for node n and incoming edge (m, n) contains the w last hops on (the infinite extension of) the random route uniquely determined by (m, n); (c) maximum number of Sybil groups: Dashed lines are attack edges. All nodes having a given attack edge in their "random" route are equivalent.

with suspects are becoming less likely. This only reduces quality of service for some suspects, not the effectiveness of SybilGuard against attacks.)

Attack routes. An attack route is a random route that contains an attack edge, which implies that it also contains at least one Sybil node. Since Sybil nodes will deviate from the protocol in any way u^* finds beneficial, this node may then claim to have acquaintances to arbitrarily many sibyl nodes. Consequently, a verifier using an attack route for verification accepts a number of Sybil nodes only bound by plausibility considerations. Since attack edges are very rare, a wise choice of w can help reducing the probability of this happening. Furthermore, since a verifier usually has more than one acquaintance to start random routes over, it can replay via several routes, and use a threshold parameter to decide how many routes need to clear the suspect for an accept. An accepted node is then assigned an equivalence class based on some default route.

Implementation. Initially, all nodes forward their random permutations to their neighbors. If a node "projects" a random route, it contacts all nodes on that route and exchanges identity tokens based on a public-key authentication scheme. If a node $v \in V - V^*$ wants to verify some node $v^? \in V$, $v^?$ sends the node lists of its routes to v, and v can contact the node on the crossing to verify whether $v^?$ is listed with the route it claims to be. An honest nodes only allows the projection of a random route through itself if it is convinced that the route so far is not corrupted (e.g., longer than w hops to the source, or containing invalid hops). This ensures that all properties of random routes such as convergence hold even if the source of a route is under control of the adversary.

8.2 Routing

At the core of every distributed application with dynamic node set lies a routing algorithm that enables the nodes to find each other and establish contact. In IP networks, this is done in a relatively static way and with a considerable amount of manual intervention. Each Internet router needs to be given rules in form of finite state automata that decide where to find which classes of IP addresses. In P2P networks, manual intervention is highly undesirable, and nodes can change location (at least in the IP topology) all the time, so novel solutions are required.

In this section, we will show that routing in decentralized environments is practically feasible. We will give examples for routing algorithms from different categories and compare their strengths and weaknesses. The model we develop in the next Chapter will abstract away from the routing problem

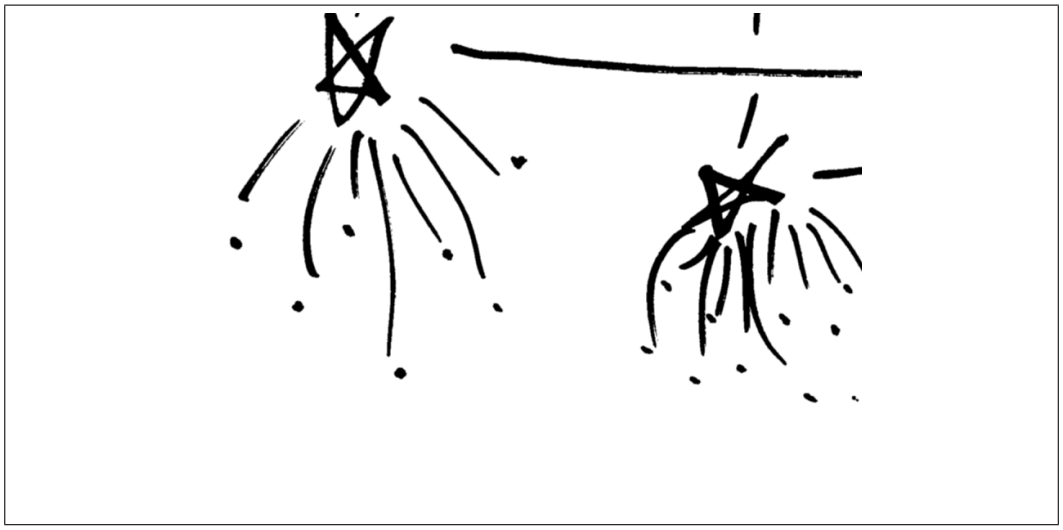


Figure 8.3: Flooding with super nodes.

discussed here. Nevertheless, in order to interpret the results that it produces, it is necessary to have an idea of the technical realities that lie behind it.

8.2.1 Flooding Strategies

To keep it simple, some applications use *broadcast* or *flooding* schemes. Each node obtains a set of IP addresses before joining the network (from a web page, hard-wired into the client, or similar), and simply sends every data packet to all nodes that it knows, hoping that those recipients that are supposed to act upon it will receive it. The Gnutella network is the most popular representative of this approach. As broadcasts are forwarded from all of v's neighbors to all of v's neighbors' neighbors, the number of nodes grows exponentially in the TTL (time to live, or number of forwards before the last recipient gets impatient and drops the packet).

The simplicity of understanding and implementing broadcast schemes comes at a cost: Whereas the computation and storage cost for any given node is $O(1)$ (just maintain a small table of neighbor IPs), the cost of sending a single packet to a single recipient that is not a direct neighbor of the sender is $O(|V|)$, since in the worst case every node in the network needs to be contacted before the target node is found. This may lead to the overwhelming majority of all data packets transported being routing requests rather than payload. Gnutella suffers greatly from this: According to some studies, 88.8% of its traffic volume consists of query messages, which leaves less than 12 for the actual payload (IEPN04).

Furthermore, if it becomes necessary to drop the requests early on in order

to avoid network suffocation, fewer and fewer of the theoretically available resources can actually be found, leading to further decrease in quality of service.

There are several ideas to make flooding schemes more scalable. One is to introduce a second layer of *super nodes* that segment the network (see Figure 8.3). Each super node is responsible for many (sub-)nodes and maintains routing information for them. Queries only need to be broadcast among super nodes, but will be sent directly to the sub-nodes that are actually interested. This reduces intra-segment traffic significantly at the cost of introducing a weak form of trusted third parties, or central servers.

Semantic routing algorithms adjust routing tables (or edges in the network graph, or the neighboring relation) to suit the communication needs of the nodes better. If two nodes need to talk more, such schemes try to ensure a direct link between them (LST05). While this works well on paper, applications using it need to have highly local traffic patterns. Expert networks where nodes are humans that bounce ideas off each other may have this property: An expert on tropical fish is likely to talk more often to other biologists rather than to political scientists. However, even in this scenario it may be desirable to switch disciplinary boundaries, but this is harder than in plain broadcast networks if semantic routing is effective and edges are lumped in the local environment. Other applications such as anonymous routing or distributed backup systems have near-uniform distribution of traffic as a requirement,[7] and are not suitable for semantic routing.

8.2.2 Distributed Hash Tables

These deficiencies of the flooding approach have led to advanced routing algorithms that span suitable data structures over the network that allow for fast lookup of any node in the network. These data structures are virtually always based on the notion of a *hash table*.

A hash table allows for $O(1)$-time storage and lookup of values under keys of arbitrary type. The keys are mapped to array indices using a hash function. These array elements contain sets of values that have constant size and thus can be searched in constant time. Constant set size of array elements is ensured by adapting the array length to a growing number of elements contained in the table.

Other than plain arrays, hash tables do not restrict the application to dense numeric key types. Keys can have large gaps between them or be

[7] Backup systems need to minimize probability of content loss, and anonymity networks need to minimize the observable structure of the (link-encrypted) network traffic.

non-numerical, and the good time and space characteristics of arrays still hold.

The hash function needs to guarantee probabilistic uniform distribution of its values. This is a necessary (although not sufficient) requirement for good cryptographic hash functions that were introduced in Section 2.3.5, so any cryptographic hash function can be used to construct hash tables. See cf. (CLRS01) for more details on this data structure in the non-distributed case.

A *distributed hash table* (DHT) distributes the elements of the array to a network of nodes such that each node is responsible for maintaining a subset of the elements. If one of the nodes has a key and wants to look up (or update, or insert) the corresponding value into the table, it computes the hash of the key, finds the node responsible for that hash using a distributed routing algorithm, and sends a query (or update, or insert command) to that node.

In the non-distributed case, finding the target array element is a matter of a simple direct memory access operation. In a DHT, this step is replaced by a routing algorithm that allows each node to maintain partial information about the network and still construct a route to any node faster than by a broadcast. Every DHT needs to provide algorithms for

1. establishing the routing tables on each node that (implicitly) assign a different range of DHT addresses to every node such that the entire DHT address space is distributed among all participating nodes;[8]

2. forwarding a packet with given DHT address to the neighbor closest to the target node (The DHT itself is not concerned with whether this packet contains a write or read operation or something else – the payload is processed on the higher architectural levels);

3. and adapting routing structure to cope with joining and leaving nodes.

We will now look at two example DHTs that differ in their structure and performance characteristics. Many more could be listed and studied here, but are left unmentioned instead since they do not contribute greatly to our agenda.

[8]This step can often be ignored in the introduction of a DHT because it follows from 3. below: Initially, the network consists of one node that covers the entire network. Network build up is done incrementally from there, following the rules for node joins.

CAN

The first distributed hash table that carried this name was the *content-addressable network* (CAN) (RFH+01). The address space of CAN is a d-dimensional torus of size $[0,1] \times \cdots \times [0,1]$.[9] Each node allocates a (hyper-)square and registers size and location of all neighboring squares, together with the IP address of the responsible node. Two squares are neighbors if they fully or partially overlap in all but one dimension. Keys are hashed to points on the surface of the torus. We call the torus fully partitioned if exactly one node is responsible for any possible key.

Routing. Routing is localized, but approximates the shortest straight line from source to target (see Figure 8.4): For a given packet with DHT address x, a node forwards among its neighbors to the one whose address range is closest to x. If that neighbor's range actually contains x, the packet has reached its target. If it does not, that neighbor re-iterates. If that neighbor is unavailable, localized flooding is used until a node is reached that is closer to x than the last sender, and forwarding along the shortest straight line to x is resumed.

Maintenance. If a new node enters the network, it contacts an arbitrary node and asks that node to send a join request to a node that is chosen uniformly at random. (This double-step approach prevents clustering of nodes around the neighborhood of bootstrapping nodes.) The node that receives the join request splits its area up in two halfs and hands one over to the new node, updating both its own and the new node's routing table appropriately.

Node exit is a little more complicated. If a node leaves, it may be impossible to combine the abandoned area with a neighboring area into a rectangle, because along no dimension the overlap is total (black area in Figure 8.4). In this case, a neighboring node must take up maintaining two areas for a while, which complicates routing table maintenance.

CAN has problems with spontaneous node failures. If too many nodes in one place fail, the remaining neighbors' negotiation algorithm can under certain circumstances lead to inconsistent routing tables. To avoid this, additional repair algorithms need to be put in place that are explained in detail in (RFH+01).

Chord

Chord imposes a much simpler circular structure on the network (SMK+01). When using hash function H with domain $\{0,1\}^m$, the nodes are arranged in

[9] A 2-dimensional torus is merely a rectangle with top and bottom edge touching and left and right edge touching, like a Pac-Man world.

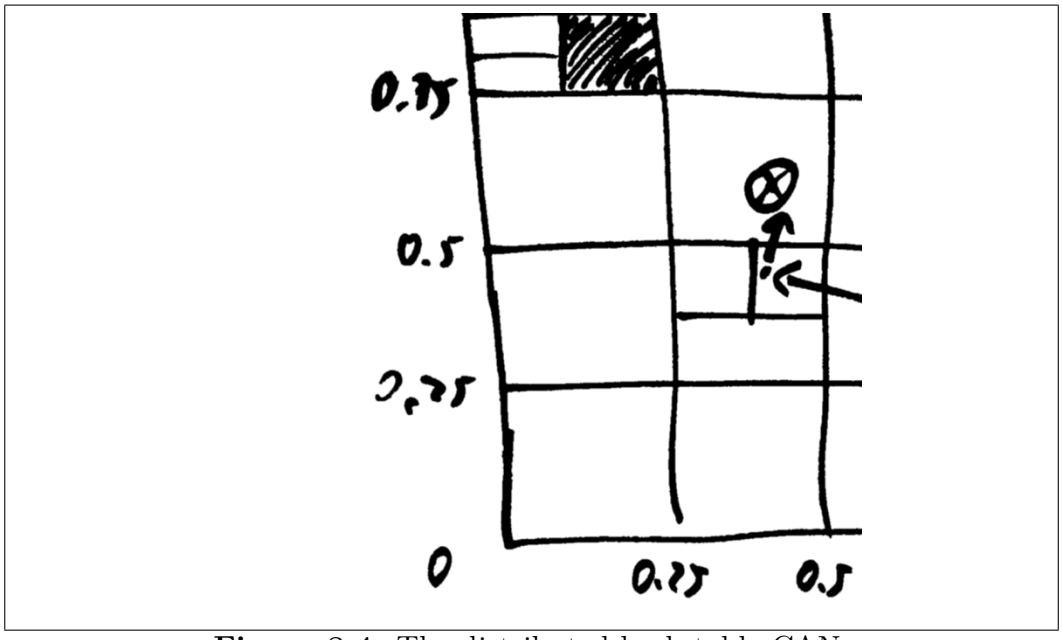


Figure 8.4: The distributed hash table CAN.

a routing circle (increasing clock-wise). All computations are done modulo 2^m, i.e. 0 is the successor of $2^m - 1$. A node i assumes position $H(i)$ on that circle. If j is i's counter-clockwise neighbor, i.e. $H(j) < H(i)$, then i is responsible for all keys x such that $H(j) < x \leq H(i)$ (see Figure 8.5b).

If a new node k enters the network, it assumes its position on the circle according to H as any other node and takes over its partition of the key space from its successor (see Figure 8.6). If k leaves, i takes over k's partition.

Routing. Each node maintains a routing table with entries consisting of key $H(\cdot)$ and IP address for a direct communication link. This table has two parts: The *neighbor table* with one pointer each to its direct successor and predecessor, and the *finger table* with shortcuts to distant nodes. The constant-size first part would be sufficient to find the responsible node for any given key by hopping along the circle in at most $|V|$ steps of length 1. Only this is no better than flooding.

Therefore, in the finger table each node i maintains a list of nodes whose distance from i grows exponentially that can be used as shortcuts. To improve on the direct successor $\mathbf{s}\text{ucc}(H(i) + 1)$ from the neighbor table, the finger table contains the nodes

$$\begin{aligned}&\mathbf{succ}(H(i)+2^1)\\&\mathbf{succ}(H(i)+2^2)\\&\dots\\&\mathbf{succ}(H(i)+2^{m-1})\end{aligned}$$

This results in a finger table of size $O(\log|V|)$ (see Figure 8.7).

If i wants to contact the node $\mathbf{succ}(H(q))$ responsible for storing the value of some key q, it looks up the node j in the finger table that directly precedes $H(q)$. If j's successor $\mathbf{succ}(H(j)+1)$ is responsible for q, j sends i the IP address of $\mathbf{succ}(H(j)+1)$. Otherwise, j consults its finger table, and the algorithm is re-iterated.

This routing table is not only small, but it also guarantees a bound of $O(\log|V|)$ on the length of routing paths. To see why, note that every step of length, say, 2^v is chosen maximal. Since it is not of length 2^{v+1}, the next step cannot be 2^v (which would reach the same target node, only with one superfluous intermediate step), but must be shorter. Hence, each step is shorter than the last one. Since there are only m possible choices in a finger table and the relative distances are the same for all nodes, the route must reach the target after at most $m = \log|V|$ steps.

Maintenance. For an entering node k to take over its partition, it must initialize its routing table and update the routing tables of existing nodes in accordance with its new presence. For finding its own neighbors, it merely needs to run a lookup on $H(k)$, and ask the answering node (say, i) for its predecessor (say, j; see Figure 8.6 again). j, k, i all update their neighbor tables to represent the new situation, and all other neighbor tables are still valid. To initialize its finger table, k can use a simple trick: Because the relative distances of the nodes in the finger table are the same for every node, each node to be found and included in k's finger table is very close to the corresponding node in j's finger table. So k merely needs to ask j for its fingers and do a local search of constant cost for each. This takes $O(\log|V|)$.

In order to find out which nodes need notification to replace i by k in their finger tables, k needs to route backwards: Iff the node preceding $H(k)-2^v$ contains i in entry 2^v, it needs to be updated. The straight-forward way to do this would do $\log|V|$ many updates that take time in $O(\log|V|)$ each, resulting in a total time of $O(\log^2|V|)$. However, there is a non-trivial scheme to do it in total time in $O(\log|V|)$. A variant of this scheme can also handle concurrent node failures.

Note that strong variance in node or query target density would kill performance. For Chord to work as described here it is crucial that both nodes

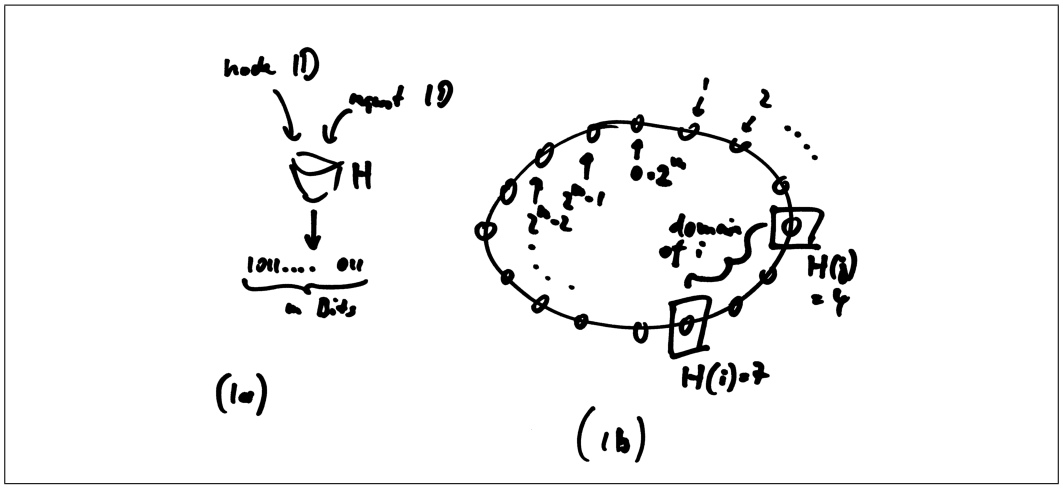


Figure 8.5: The distributed hash table Chord (1). The nodes assume positions on a circle of all hash values. Each node answers to queries sent to keys in the interval between his own and his predecessor's.

and queries are uniformly distributed. This is ensured even for non-uniformly distributed node or target names by using a cryptographic hash table.

8.2.3 Discussion

Which data structure is the best heavily depends on the structure of the application in question. The flooding approach was used in earlier stages of the development of purely distributed P2P networks, since its theory as well as its realization is much simpler than that of DHTs. Flooding performs relatively well on small networks, or if nodes are very unreliable and the network is in heavy motion, with nodes joining and leaving and changing their name all the time.[10] DHTs need to update routing tables and propagate these updates for every change in topology, and the overhead caused by this in computation and communication can exceed the benefits of higher routing performance. Furthermore, routing performance may be impossible to maintain if updates are slower than the actual changes in the network and routing tables are constantly out of date.

However, the benefits of DHTs are so high that most applications are likely to be better off with one. Although flooding imposes no restrictions on the queries sent and thus appears more suitable for complex application protocols, it turns out that with a little care, most common applications

[10]Changing names is not necessarily an attack: In Gnutella, IP addresses are used as identifiers. If a node connected via a DSL modem gets cut off, it will lose its dynamically assigned IP address and is forced to re-enter with a new identity.

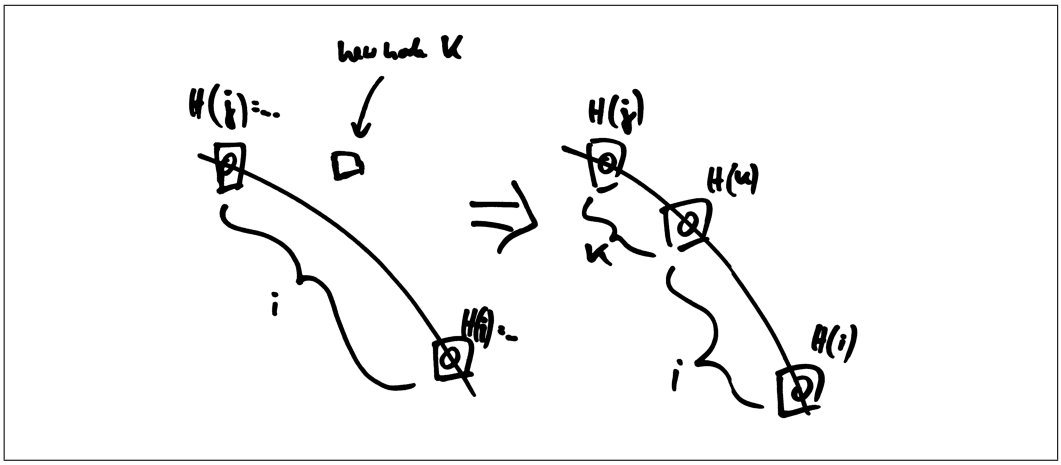


Figure 8.6: The distributed hash table Chord (2). If a new node k joins the network, its successor's range is cut into two, and k assumes responsibility for its part.

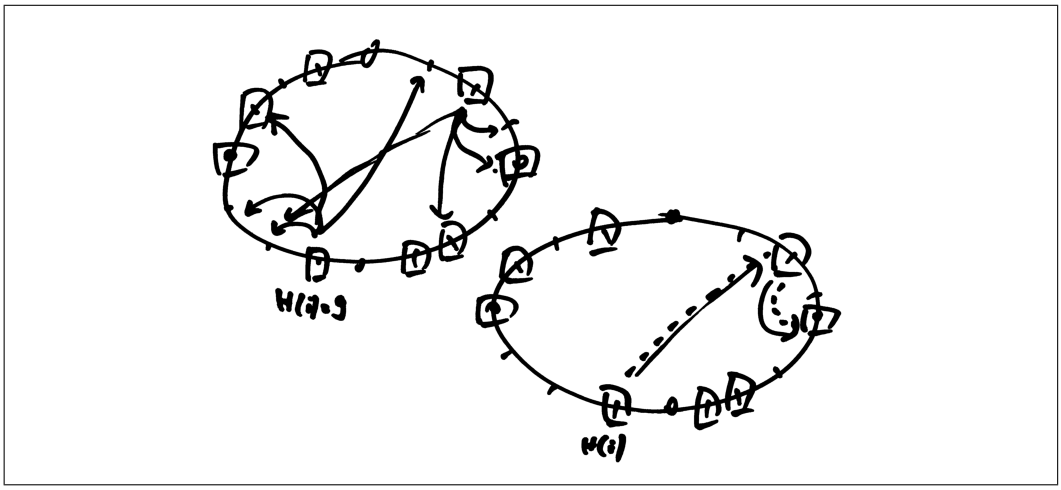


Figure 8.7: The distributed hash table Chord (3). Routing takes place along the finger table entries to ensure logarithmic routing path length.

can be implemented on DHT as well. For example, full-text or database search queries are conceivable in DHTs if appropriate index structures are maintained. To give a very simple example, document tagging can be implemented in a DHT as follows: A tag is a document containing the keys of all documents matching that tag. In order to tag a document, lookup the document representing the tag in the DHT, and append the key to the new document to it. In order to retrieve all documents matching a tag, retrieve the tag first, and then all the documents whose keys are stored in it.

The differences between the two approaches boil down to a shift of some of the work from the point in time when a node actually sends out its first request (this is where flooding starts causing costs by exponential increase in messages) to the point where a node enters the network (this is where a DHT is expensive by establishing the node's routing table). If nodes stay in the network sufficiently long, DHTs can be used with benefit. Hybrid approaches are conceivable, but we are not aware of any considerable work in this direction. Given the potentially enormous overhead of both approaches and the worst-case duplication in effort if a flooding algorithm and a DHT are used in parallel, this should be an interesting line of applied research.

As to the question of which DHT to use: It is hard to look at a distributed algorithm and have an intuition of how expensive it is under a given set of system parameters without extended experimentation. And it is more difficult to establish realistic parameters without seeing an application in wide spread use.

CAN seems to suffer from overly long routing tables that are therefore expensive to keep up to date. The bound on routing path length is $\sqrt[d]{N}$, which outperforms flooding in terms of complexity, but at the a relatively high cost: First of all, the constants in the complexity class are much higher than in straightforward flooding, and second, d constantly needs to be adapted to changing N for optimal results, but large d causes the number of neighbors to explode, which again is bad for performance.

Chord has a much simpler design with fewer rules and exceptions. Path lengths are bounded by $\log N$ steps, at a routing table size of $\log N$. Logarithmic path length is usually better in practice than polynomial already for small network sizes, but some applications may turn out to be exceptions to this rule of thumb. Other distributed routing algorithms, e.g. Pastry (RD01), Tapestry (ZHS+04), MadPastry (ZS05), or Kademlia (MM02), are competing with the options outlined above.

Our concern has been to establish that solutions for the problem of routing in decentralized environments do exist. Our focus on reputation allows us to abstract away from any particular solution in the following, and leave selection of the proper algorithm to others.

8.3 Deletion-Proof Data Structures

If reputation information is stored in a distributed datastructure on untrusted nodes that have a direct, selfish interest in this information, several things can go wrong:

1. A node may have not received a message for storage, but present a forged one claiming to have received it. This way, good reputation can be generated without transactions ever having taken place that would justify it.

2. A node may change a message that it has received. Say, it could change a claim that a transaction went wrong into the claim that the same transaction went well.

3. A node may drop messages received for storage. This is a convenient strategy to get rid of bad reputation.[11]

1. and 2. can be prevented by message authentication and signature schemes, depending on the application scenario. (In applications where only the node that submitted a message for storage needs to verify its authenticity, symmetric key management and message authentication is enough; in reputation systems, every node needs to be able to verify what any other node has stored, and thus a public-key signature scheme is likely to be necessary. Both are practicable and require little treatment beyond what has already been said in Chapter 2.)

Countering 3. is more challenging: If node i sends a message m to node j (and forgets about the process, since keeping track of all transactions for later verification is as costly as the transactions themselves), but j drops the message plus signature, signature schemes are useless. j claims to never have received m, which is consistent with there being no signature in the first place.

Reputation systems are highly vulnerable to such *erasure attacks*: If a number of nodes cooperate to drop (or erase) all negative reports on their behavior, their reputation will increase significantly. Clearly, we need a trick to make this impossible, or at least easy to detect and punish.

The simplest solutions is to fall back to a trusted third party. TTPs do not act against the protocol, so no information that is ever stored there in

[11] Similarly, if a message is deleted by the author, the node may keep it and pretend it is still valid. In reputation schemes, we assume that messages are permanent and are never deleted, so we ignore this attack. If occurring, it can be addressed using the methods effective against message drop.

the first place will ever be lost. However, we have rejected TTPs in Section 8.1.2 as a last resort, and would like to find something that works in a pure P2P network.

There is a class of signature schemes for creating *undeniable signatures* (CA89), but they require online interaction with the signer during the verification process. Alternatively, each message could be stored on the node that created it, basing consistency of the available information solely on the originator of the information. However, in both cases, whenever some i wants to compute the reputation of j, every k who knows some j needs to be online and be available for communication. Node availability is even harder to guarantee than node cooperation. Therefore, we are rejecting these approaches as infeasible.

The next-best solution is to choose a finite set $\mathcal{M}$ of messages, and register *all* messages on every node: Both those messages to be stored and, in a different way, those *not to be stored*. An enumeration function that maps $\mathcal{M}$ to $\mathbb{N}$ and a bit array could be used for that. If m is to be stored on j, the bit under the index representing m in the array on j is set to 1. Otherwise, it is set to 0. Signatures can be used to enforce integrity of the array. Since the array makes an explicit claim about whether a message is there or not, if node j drops a message, any other node can detect this by verifying the array signature.

The array is called a *state credential*, and it can used as a replacement for individual message signatures. Its advantage over individual message signatures is that it not only allows to verify integrity of existing messages, but also claims of absence of messages.

The idea of using a bit array ranging over the entire message space is of theoretical interest. For most message domains, it is prohibitively expensive. But it is the idea underlying a number of more sophisticated schemes that are practicable and provide the same consistency guarantees. In order to develop them, observe that the array is usually extremely sparse. Using hash functions (see Definition 2.17), we can easily develop very compact state credentials: Given a hash function h and signature scheme (G, S, V), the state credential for message set $M \subseteq \mathcal{M}$ is

$$c = S(h(M))$$

However, if a node claims that $m \notin M$, the verifying node still needs to download M, c, check that $m \notin M$ by hand, and then check that $V(h(M), c) =$ `true`. Further, each time a new message is added to M, M has to be downloaded in order to create a new state credential. So the challenge consists in finding a state credential scheme that allows for efficient verification and incremental update.

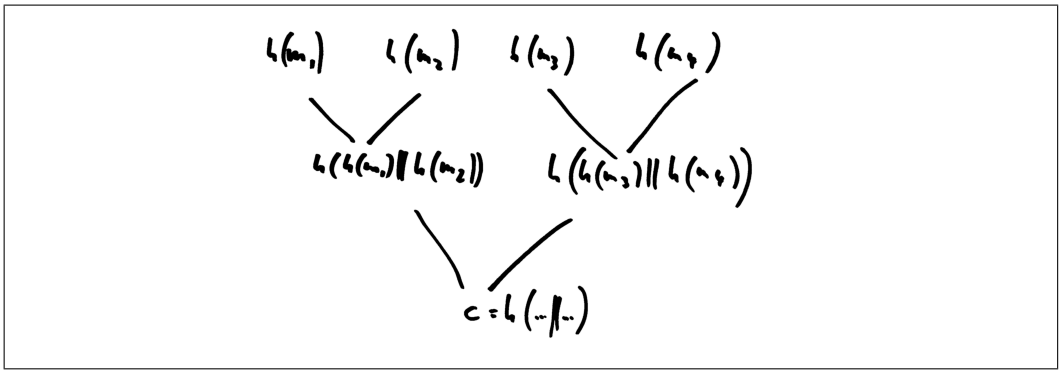


Figure 8.8: Merkle hash trees.

A scheme that allows for incremental update has been proposed in (Mer80) (see Figure 8.8). It sorts $M \subseteq \mathcal{M}$, and computes individual hashes for every $m_i \in M$ (starting with a hash for the empty database, since if there are no messages at all we need proof for that, too), and then constructs a binary tree with those hashes as its leaves. Assume for a moment the number of messages is always a power of 2. Then the hash tree H has depth $l := \lceil \log_2(|M|) \rceil$ and is constructed bottom-up, and the credential $c = H(1,1)$ is the root of the tree. Assume for a moment that $|M|$ is a power of 2. Then the H is constructed recursively as follows:

$$H(l,i) = h(m_i) \forall i \in 1..2^l$$
$$H(j,i) = h(H(j+1,2i) \parallel H(j+1,2i+1)) \forall j \in 1..l,\ i \in 1..2^j$$

If $|M|$ is not a power of 2, a bigger number of roots needs to be maintained instead of the single root $H(1,1)$ (the idea should get clear from Figure 8.9 much easier than from the more complex algebraic definition, so we omit the latter). Each time a message is added to M, at most l nodes need to be reconstructed, and thus at most l hashes signed in order to compute an updated state credential.

To confirm set membership for $m \in M$, node j presents all pairs of hashes from leaf m to the root, together with the root signature (created by some other node i who feeds M with messages). If (a) the root signature matches and (b) all pairs of hashes are consistent with the hash on the next level, then $m \in M$ indeed. Since the tree is binary, this makes for logarithmic communication complexity. On the other hand, if $m \notin M$, it is still necessary to completely retrieve M and re-compute the state credential, just as in the naïve approach described above.

In (Bau04), a scheme is presented that has both efficient update and verification, and provides protection against all four attacks listed in the

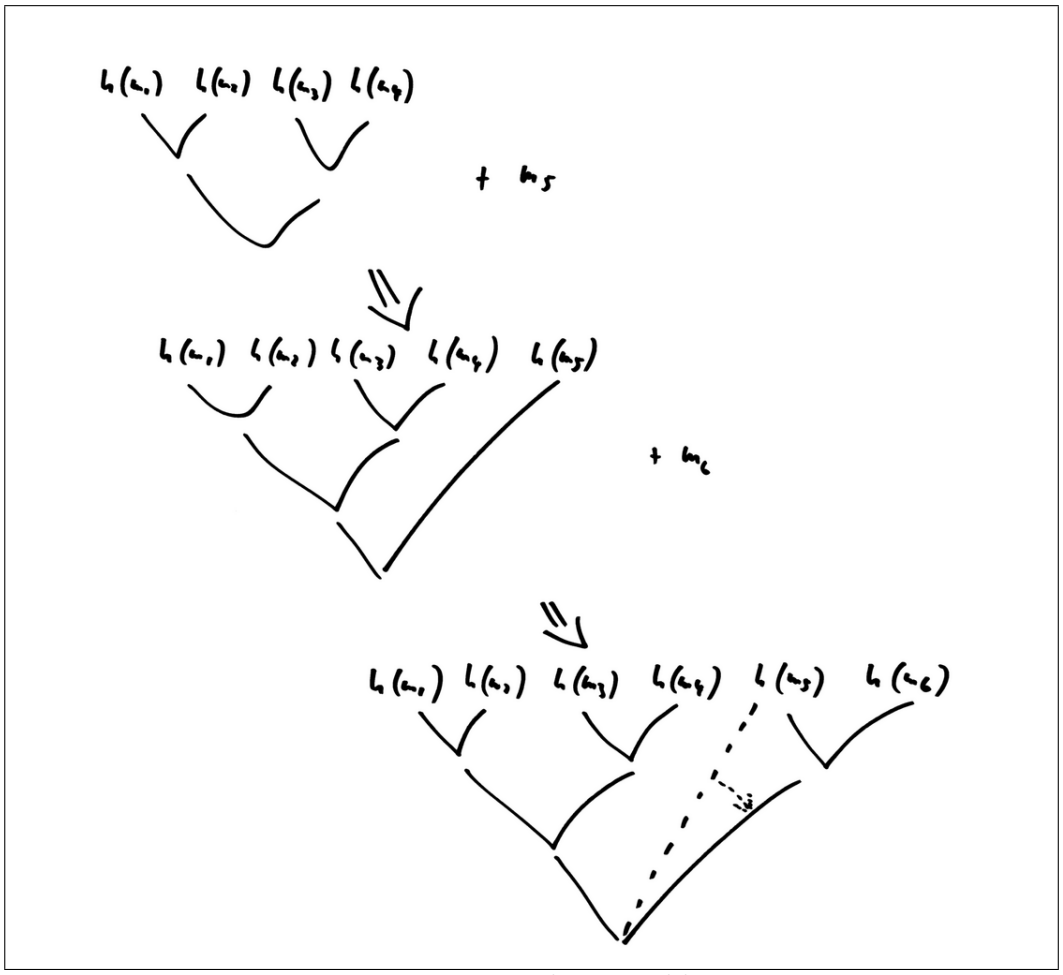


Figure 8.9: Extending Merkle trees.

beginning of this section. The state credentials created by this scheme are called *keyed hash trees*.[12]

As a Merkle tree, a keyed hash tree is a binary tree, but it is not constructed based on the ordered sequence of elements of M, but on the elements themselves. In fact, it is much closer to our original array construction, with three important differences: (1) Instead of storing messages directly, it stores their hashes; (2) instead of an array, the binary search tree is used to find the 1s and 0s that signify presence or absence of m, respectively; (3) it comes with a representation linear in size to M.

As an ordinary binary search tree, to decide whether a given message m is present or absent, we construct a path through the tree based on the bits in $h(m)$ as sketched in Figure 8.10: For each 0, make a left, and for each 1, make a right, until a leaf is reached. If the leaf is 1, m is present, otherwise it is absent. Update works analogously: Find the leaf corresponding to $h(m)$ for a newly inserted m, and flip its value from 0 to 1.

Similar to Merkle trees, the node values are computed incrementally as follows: Each leaf contains the presence bit for the corresponding (hash of) message m. Each internal node assumes the hash of the concatenation of the hashes of its two children. The (signed) root of the keyed hash tree is the state credential.

Assume node j maintains some set M of messages and node i wants to know whether $m \in M$ for some m. If $m \in M$, then j presents pairs of signatures as in a Merkle tree. But in keyed hash trees, this is also true if $m \notin M$. The difference is that we end up at an "empty" leaf, not at a "full" one.

For hash values of 160 bits (which is rather common), this tree has depth 160 and 2^{160} leaves. Storage of keyed hash trees only is practical because there is a compact representation for the vast regions of the tree that only have empty leaves.

Let H^k be a keyed hash tree (as in the description of Merkle trees above represented as the function mapping nodes to their values). An empty subtree is a sub-tree of H^k that has only empty leaves. A maximum empty sub-tree is an empty sub-tree whose sibling is not empty. Note that empty sub-trees of the same depth all are identical: All leaf values are $h(0)$, all values of leaf parents are $h(h(0) \,\|\, h(0))$, and so on. Further, note that if a path points into

[12](Bau04) solves the slightly more general problem of finite maps from arbitrary keys to arbitrary values. Our application is the special case where keys are our messages and values are set membership bits. Independently of (Bau04), (MRK03) proposes a very similar data structure for a signature scheme that has far stronger security properties. In particular, it allows for verification of claims of the form $m \notin M$ while keeping all other information on M secret from the verifier.

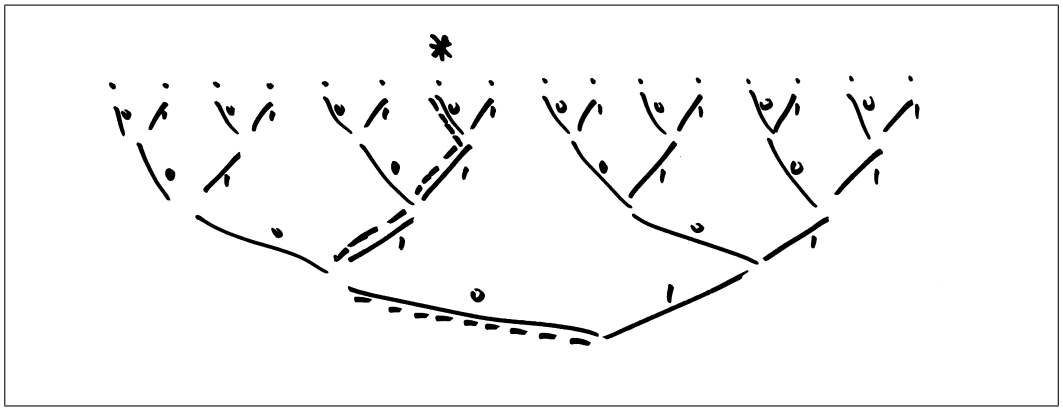


Figure 8.10: Keyed hash trees for $|h(\cdot)| = 4$. Initially for $M = \emptyset$, all leaves are 0. To insert some new m with $h(m) = 0110$, set leaf $*$ to 1.

an empty sub-tree, there is no need to know at which leaf it ends. The hash of the root of that empty sub-tree and the pairs of hashes up to the root of H^k alone allow for verification of a state credential: If all hashes add up, the root is a proof that $m \notin M$.

Note that since empty sub-trees are not stored explicitly and thus their internal node values are unavailable for look-up, this requires every verifying node to maintain a table of $|h(\cdot)|$ pre-computed hash values — one for each possible size, and thus root node value, of an empty sub-tree. This imposes a memory overhead logarithmic in the length of the output of h on every node, which can be considered a small constant.

Finally, h is collision resistant. Modifying any one sub-tree or flipping any two sub-trees always yields a new root value:

$$\forall a \neq b \,:\, h(a) \neq h(b) \;\wedge\; h(a \,\|\, b) \neq h(b \,\|\, a)$$

Therefore, keyed hash trees are secure against the three attacks described in the beginning of this section.

8.4 Reputation

To conclude this Chapter, we have a quick glimpse at three reputation schemes previously proposed.

8.4.1 CAN with Feedback

In (BB03; BBvdW06; Buc06), an extension to CAN is proposed that maintains and spreads track records of the behavior of neighboring nodes.

Gathering feedback. A node obtains feedback information by (1) direct experience in routing service quality, by (2) collecting proofs of work, and by (3) participating in rumor spreading.

(1) A node that performs a packet forward according to protocol instead of silently dropping the packet gains reputation.

(2) New nodes that have not had a chance to gather any reputation can be forced to present proofs of work (see Section 8.1.3). This is meant to disencourage identity replacement by lazy nodes.

(3) Incoming rumors on neighbors are cached for a certain amount of time and then flushed to make room for newer feedback. There are a number of caching rules, such as discarding feedback from untrusted nodes immediately, and replacing old feedback if an update from the same node on the same node arrives.

Making use of it. There are two ways in which a node's reputation level affects its experience of the network. First, packets from nodes with high reputation are forwarded with higher priority than packets from low-reputation nodes. The latter may in fact be asked to perform a proof of work, which in many applications may render the data to be sent obsolete before it can be done. Second, in order to increase expected quality of service, nodes with low reputation are avoided by the routing protocol.

Properties. CAN with feedback discriminates between cooperative and defective nodes such that the cost for defective nodes to obtain the same level of service quality is higher than that for cooperative nodes. Furthermore, since this affects node behavior and reduces the number of defects as nodes learn that cooperation leads to better results, overall performance increases. This has been demonstrated both in an analytical model and with simulations on networks of robot nodes (BBvdW06).

The semantics of reputation is complicated, since it both involves aspects of service quality and truthful reputation propagation. This leads to the dangerous assumption that a cooperative node will never lie. But it is possible that lying while cooperating yields better results than both telling the truth while defecting and telling the truth while cooperating. If this was the case, the majority of lazy nodes would choose to lie and cooperate, rendering the assumption faulty.

A more tricky incentive issue that is very hard to avoid evolves around the question of how to act on reputation. If a node chooses to burden popular cooperative nodes with more work, there is an incentive to achieve a low reputation. This can only be helped by assuming that cooperative nodes

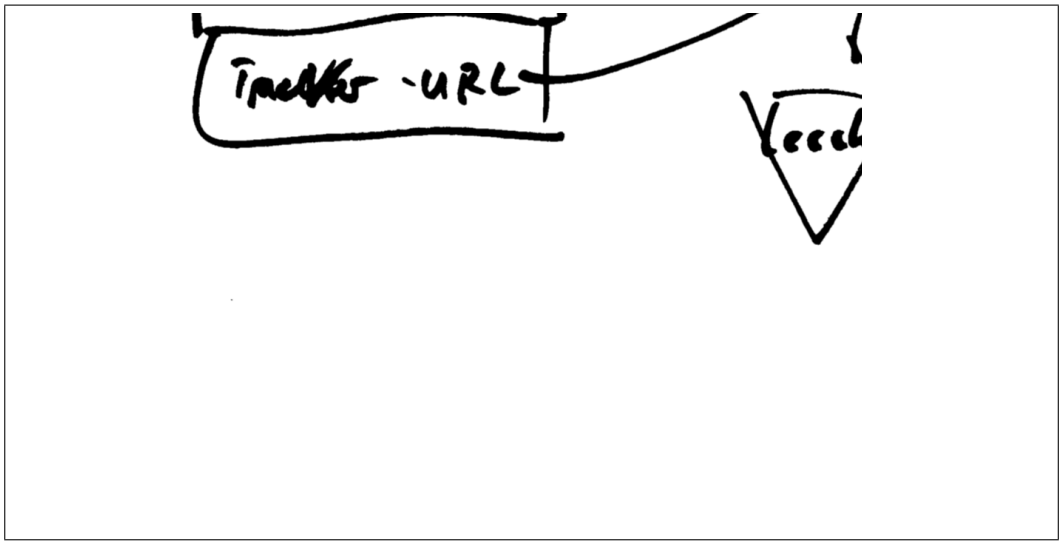


Figure 8.11: The BitTorrent architecture.

cooperate out of their intrinsic non-economic motivation. (Fortunately, this assumption finds ample support in the real world.)

A minor defect of CAN with feedback is that it relies on proofs of work, which decrease total benefit of a network by artificially increasing the amount of necessary effort. This reiterates the discussion in Section 8.1.3.

Finally, we have reasoned that CAN is not the most efficient DHT available today, so it would be nice to have adaptations of the CAN feedback scheme to arbitrary DHTs. It turns out this is feasible and has in fact been done (Buc06).

8.4.2 BitTorrent

BitTorrent was born not as a theoretical result but as an application wrapped around an elegant and powerful file distribution protocol. It addresses the problem of distributing large files with minimum server resource requirements by making the downloading clients share the parts they have already downloaded among each other (Coh03). The architecture is very simple (see Figure 8.11). The server distributes an index file called the *torrent*. A torrent contains hash values of the chunks of the files to be downloaded, together with a URL to the *tracker* that provides lists of downloading nodes to new downloaders via HTTP.

Once the torrent is distributed to everybody (e.g. via a link in a traditional web page) and the tracker is set up, one node needs to connect to the tracker that makes the entire file available (the *seed*). Nodes that want the file obtain

the torrent, consult it to learn where to contact the tracker, and obtain lists of nodes to start downloading from (and uploading to) from there. These nodes are called *leeches*. A leech that has completed the download but does not log off becomes a seed, so there may be many seeds at any given time.

On each connect of a new node (seed or leech), the tracker hands out a random subset of all nodes of constant size. This is the only thing it does. After this, the nodes connect to each other and work independently.

Each node sends requests for some of the file chunks it does not possess yet to its neighboring nodes. It faces two types of decisions: (1) Whom to ask for which chunk, and (2) whom to provide with chunks that have been requested.

(1) Selection criteria. Several heuristics are used to decide on the order in which the chunks are downloaded. Scarce chunks have higher priority, since every download reduces scarcity. Nodes that repeatedly refuse to allow downloads are excluded from requests. During the end run when waiting time for a specific last chunk should be minimal, broadcasts are used instead of targeted requests. Finally, within all the other rules, chunks are downloaded in random order (this avoids hot spots such as the first chunk in the file when it first gets published).

(2) Cooperation criteria / feedback. This is the more interesting of the two questions, since it determines the economic incentives the nodes experience and thereby the total upload (which in turn equals total download). Each node chooses four nodes that it uploads to, and changes them occasionally if it is likely to increase payload traffic. The rules are:

- Tit-for-tat. Those neighbors that provide for the fasted download connections are served first. If the download gets slower and a node moves on to download from other nodes, those other nodes will become the new tit-for-tat partners. This is the core rule of BitTorrent and fights off the prisoner's dilemma (see Section 7.5) and at the same time increases the file's availability (faster nodes will distribute what they download faster than slower nodes).

- Neediest first. Once a node is done downloading and stays online for a while (as is considered polite), tit-for-tat is not an option any more. In this case, the neighbors are served based on the upload they can consume. (Again, since each chunk downloaded increases total availability of the file, the faster the downloads now, the faster they will be in the future).

- Perturbation. The neighbor that downloads least is occasionally replaced by a random neighbor from the list. This is called *optimistic*

un-choking. It yields a certain degree of change. Because it prevents situations in which the most desperate nodes are forgotten, it gives a better incentive for sticking with the strategy than plain tit-for-tat. This brings BitTorrent closer to the ideal of Pareto-optimality.

- Anti-Snubbing. If a neighbor is not cooperating any more, a node moves on to another one and stops uploading. This makes BitTorrent more robust agains denial-of-service attacks, but it also prevents unmalicious stalemate situations.

Discussion

BitTorrent's simple design emphasis on high network saturation result in near-optimal content distribution performance (QS04). The theory behind it makes for a valuable building block for high-level P2P applications. However, there is a list of things that BitTorrent does not (and is not designed to) do.

First, there is no concept of long-term reputation. Instead, each node takes the current experienced upload rate of a neighbor as input for its strategy (averaged over a 20-second window). Although this works well for BitTorrent, it is not a reputation scheme, in the sense that no reputation information is maintained or propagated.

Second, there are no adversaries in BitTorrent. Since reputation is not distributed, there obviously are no liars. The most malicious thing a node can do is to have a low (or zero) upload rate. This is far easier to counter than nodes that upload corrupted content or disobey the protocol in other ways to disrupt service.

Third, the resource model is very narrow (large blobs of unstructured and atomic content such as CD images with free software), which on the one hand yields very good results for one application, but on the other needs to be re-invented for others.

Finally, centralized tracking is a pragmatic solution, since the original motivation to invent BitTorrent was to allow for web server style systems with minimum server load. But a truly decentralized P2P system would need to use a distributed algorithm for tracking, and it would be necessary to ask the questions of robustness against adversaries again for this new system.

8.4.3 EigenTrust

In (KSGM03), the *EigenTrust* algorithm for reputation tracking based on the Gnutella network is proposed that targets soundness and quality of shared

documents. For each node i, EigenTrust stores information on past transactions with other nodes j under the index ij in a *reputation matrix*, and reputation information is computed from that matrix.

The trick that renders EigenTrust efficient in its fully distributed version is the interpretation of an eigenvector of the reputation matrix as reputation figures. The reputation matrix M contains in each cell M_{ij} an aggregated figure on the experiences of node i with node j. If node j has uploaded virus-free files of high quality and with correct meta-data, M_{ij} is high; if j has been observed to upload bogus data, interrupt uploads etc., M_{ij} is low. In order for node i to accomplish a more objective opinion on node j's expected performance t_{ij}, it consults its peers on their opinions:

$$t_{ij} = \sum_k M_{kj}$$

To reduce the impact of lying peers, i weighs the assembled opinions of other nodes by its own opinion on their truthfulness:

$$t_{ij} = \sum_k M_{ik} M_{kj}$$

Note the implicit assumption that being truthful within the reputation system is the same as being highly reputable. As long as we want to isolate peers that pollute a content distribution network with false information, this is justified. But if reputation is a performance figure that may be low for honest but weak nodes, this assumption does not hold any more. EigenTrust in its current form is limited to one type of reputation, namely truthfulness.

The above equation can be rewritten in matrix notation. If t_i is the vector of opinions of i on all js:

$$t_i = M^T M_i$$

Now i wants to know the opinions of the peers of its peers. Each of i's peers k gives an opinion held by each of k's peers l on each of l's peers j, and that opinion is weighted first by k's opinion on l, and then by i's opinion on k:

$$t_{ij}^{(1)} = \sum_l \sum_k M_{ik} M_{kl} M_{lj} = \sum_l t_{il} M_{lj}$$

Or in other words, and iterating further:

$$t_i^{(c)} = M^T t_i^{(c-1)}$$

In order for i to decide what to make of j, it would make sense to take all $t^{(c)}$ into account, possibly weighing earlier iteration steps more because they are closer to personal experience, and thus less distorted by untruthful nodes.

Because it is hard to develop an efficient algorithm for this approach, EigenTrust takes a different route using two observations: (1) iteratively left-multiplying M^T to $t_i^{(0)} = t_i$ converges to an eigenvector (or to be more specific, the left principal eigenvector) of M; (2) the outcome only depends on M^T, any t_i converges to the same eigenvector, and thus all nodes perceive the same reputation information.[13] EigenTrust computes the eigenvector for a source of global consensus on the individual nodes' reputations, which it can do efficiently in a distributed fashion.

Discussion

For an individual node i this means to discard direct evidence, or evidence reported by nodes j that i knows personally. On the other hand, the weights put on each matrix cell in each round repair this shortcoming to an extent, and the simulation results in (KSGM03) suggest that this move is valid.

Gnutella as a network technology is quite outdated, and has been demonstrated to have scalability and robustness issues (see Section 8.2.1 and e.g. (IEPN04; RFI02)). EigenTrust (in a refinement of the reputation function sketched above) makes the relatively strong assumption of a small number of pre-trusted peers. Worse, it assumes strong identity tokens. Since Gnutella lacks cryptographic message authentication, there is no way of telling which node sent which message, and thus any bound on adversarial nodes is dubious.

In order to make full use of an advanced reputation system like EigenTrust, one would need to port it to a more modern network technology making use of the building blocks listed above. This, however, requires intricate changes and re-evaluation of the effectivity of the changed system. For instance, the problem EigenTrust attempts to address is data quality, which is far easier to enforce when the system makes use of hash functions (which comes for free with DHT routing) and digital signatures. Reputation tracking is much more valuable when used for propagating information like upload bandwidth.

Independently of EigenTrust, we have developed a model that is similar in that it also stores reputation in a matrix over all pairs of nodes in the network (see Section 9.1.1). Other than EigenTrust, our work does not include a detailed performance analysis. On the plus side, it is not restricted to one type of reputation. This abstraction from any real-world application allows for a more straight-forward system with fewer artifacts and thus gives way for more general insights into the mechanics of reputation.

[13] It is not necessary to know what an eigenvector is, or to be able to prove these two facts. Consult any other book on linear algebra for the relevant definitions and proofs.

8.5 Inverse Sybil Attacks

We conclude this chapter with a slightly off-topic shift in perspective. Up to here we have aimed at enabling P2P applications in which identities are easy to create in abundance, and cannot be linked to each other due to a lack of surveillance infrastructure that is a desired feature of the infrastructure. We have then struggled to keep users of this infrastructure from abusing their freedom through Sybil and other attacks.

In this, we always assumed that it is in the interest of honest users to safeguard their credentials and protect them against identity theft. Technically, this is simple to achieve by means of public key cryptography. Identity theft requires the thieve to get access to the private cryptographic key, but that key should never leave the owner node. In this section, we scratch on the surface of a complementary and much harder problem: Assume that nodes have an interest in trading their credentials, but this violates the legitimate interests of others. How can the system prevent identity trade by technical means, and thus make sure that the same person is consistently hidden behind the same pseudonym?

We will start from society at large as an application. Here, requirements such as accountability and forgery-resistance of identity tokens are widely accepted by the "users". And yet, the more data is aggregated by numerous private and public organizations, the bigger the problems that result from an *unnecessary* transparency of the individual. In order to avoid this transparency, the notion of *pseudonym systems* has been introduced (Cha85). Instead of allowing for only one identity token that is applied everywhere, a pseudonym system allows for the individual to use different local identity tokens for doing business with different organizations. Pseudonym systems are related to the blind signature schemes discussed in Section 8.1.2. However, the security requirements are complementary: A pseudonymous identity token used to commit an online auction fraud needs to give the authorities a proof of the fraud, so the authorities can find and prosecute the person behind the pseudonym. The interesting property of a pseudonym system is that the auction portal operator can only use the pseudonym for presenting it as a proof of fraud (and, of course, for processing legitimate transactions); she cannot contact the person with marketing campaigns, or sell personal information, or register the person's online behavior in user profiling databases.

These requirements are too strict to be of direct interest for the goals of this thesis. In particular, they are hard to imagine without a TTP, an assumption we have refuted because of the benefits of a lean, anonymous infrastructure. Still, pseudonym systems are related enough to deserve mentioning. In particular, it allows to consider a problem that is in a sense

opposite to the problem of Sybil attacks. Instead of one user creating many pseudonyms to do business with one organization such as a P2P application, it is sometimes undesirable that many users share a single pseudonym to do business with an organization.

A flatrate content provider or a public transport system who decides on a pricing policy based on the expected number of customers, experienced maximum consumption level, and cost of providing a service or good, could take considerable damage from involuntarily issuing multi-user pseudonyms. The further a pseudonym system permeates society, the more severe the possibilities of abuse. Neither medical prescriptions nor drivers licenses can be allowed to be transferable.

But the problem can also arise in P2P networks. Consider an expert employment agency over which users can hire consultants online to answer questions that require highly specialized knowledge but very little time. A low-maintenance, self-organizing network with a distributed reputation system (and optionally a separate micropayment system) would be ideal for this task, but it could suffer from an expert selling her pseudonym to someone with little experience who can then use it to earn money for inadequate service until the reputation associated with the pseudonym drops sufficiently.

(LRSW00) addresses this question and proposes a solution based on incentives. Each user has a master key that gives away all pseudonyms and allows an illegitimate holder to impersonate the legitimate one in any possible context. The legitimate user is reasonably expected to keep this key private for her own sake. Now, the authors propose a scheme that guarantees that if any user transfers a non-transferable pseudonym, she gives away her master key. Consequently, non-transferable pseudonyms can be expected not to be transferred by anyone.

Chapter 9

Modelling Reputation Schemes

In this chapter, we will develop a novel abstract model of reputation schemes for P2P networks making use of the cryptographic foundations established earlier on in this thesis. We will motivate this model, and then assess its characteristics and performance on a generic simulator we have developed for P2P system reputation schemes.

This simulation serves two purposes: Obviously, we want to demonstrate the strengths of the model and reputation scheme we proposed. But more importantly, we hope to lay the groundwork for comparing a wide range of other reputation schemes by expressing them in the same idioms, and for exploring their strengths and weaknesses with our simulator.

An earlier version of this chapter has been published in (Fis06).

9.1 The Model

The network consists of a set $\mathcal{N}$ of pseudonymous $i, j, \ldots$, or nodes. A node enters the network at the beginning of time and never leaves. Node i's behavior is defined by

- supply $v_i \in \mathbb{N}$
- demand $w_i \in \mathbb{N}$
- hostility $e_i \in \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$

The hostility parameter e_i determines with which probability a given peer misbehaves in a given round. Misbehavior may assume different forms: The node may not contribute anything at all, even if the other parameters demand some limited degree of cooperation by everybody in every round, or it may feed false information to the data structures for reputation tracking.

See Section 9.1.2 below. v_i, w_i, e_i are drawn from the random variables W, V, and E, respectively.

Request / response: In round r, every peer i generates w_i discrete resource requests, write: $i \xrightarrow{r} j$, where j is picked according to some fixed probability distribution over all peers. (For now, assume uniform probability for every peer other than the source to become the target of a request.) As soon as all requests have been sent, each request target, considering circumstances like v_j and the reputation of peer i (see below), decides whether to respond with $i \xleftarrow{r} j$ or not. The total of all events (requests and responses) forms the network trace Δ.

Response rate: The response rate of peer i is defined by the ratio

$$\pi_i(\Delta) = \frac{|\Delta \cap \{i \xleftarrow{r} j\}_{j \in \mathcal{N}, j \neq i}|}{|\Delta \cap \{i \xrightarrow{r} j\}_{j \in \mathcal{N}, j \neq i}|}$$

between resources received and resources requested in the entire simulation. Note the implicit validity conditions: The number of requests must be ≥ 1 to avoid division by zero. Also, in the logarithmic case below, the number of responses must be ≥ 2.

Neither the cost of responding to requests nor the benefit of charging for delivered resources is taken into account here; peers are invariant to whether they can provide a resource or not. Our focus is not on the profit made possible by established business transactions, but on comparing the impact of different notions of reputation on the number of profitable transactions established.

In order to reduce the effect of extraordinary resource hunger on performance (very high w_i), it may be more interesting to use a logarithmic scale:

$$\pi_i^{\log}(\Delta) = \frac{\log |\Delta \cap \{i \xleftarrow{r} j\}_{j \in \mathcal{N}, j \neq i}|}{\log |\Delta \cap \{i \xrightarrow{r} j\}_{j \in \mathcal{N}, j \neq i}|}$$

Network performance: We use two functions to measure the quality of a set of network parameters from the point of view of the network designer who has the common good in mind. First, we average evenly over the individual response rates, taking the needs of each peer equally serious:

$$\Pi^{(1)}(\Delta) = \frac{\sum_{i \in \mathcal{N}} \pi_i(\Delta)}{N}$$

Then, alternatively, we put weights on hostile peers because we are less interested in their needs than in the needs of honest peers:

$$\Pi^{(2)}(\Delta) = \frac{\sum_{i \in \mathcal{N}} (1 - e_i) \pi_i(\Delta)}{N}$$

The logarithmic variants $\Pi^{\log,u}(\Delta)$ and $\Pi^{\log,w}(\Delta)$ are defined accordingly. Where possible, we omit explicit mention of Δ for the sake of simplicity.

As an alternative to hostility, individual performance can be weighted in accord with the supply parameter v_i, given that we have some means of normalization and the normalized $\bar{v_i}$ such that $0 \leq \bar{v_i} \leq 1$:

$$\Pi^{(3)}(\Delta) = \frac{\sum_{i \in \mathcal{N}} \bar{v_i} \pi_i(\Delta)}{N}$$

Finally, we could consider hybrid weights:

$$\Pi^{(4)}(\Delta) = \frac{\sum_{i \in \mathcal{N}} (1 - e_i) \bar{v_i} \pi_i(\Delta)}{N}$$

9.1.1 Reputation

We now outline how peers can keep track of who did what to whom, so the effect of lazy and hostile peers on the performance can be minimized by either sending requests to more reliable peers, or responses to more worthy ones, or both.

The core of our reputation scheme is a matrix $M^{N \times N}$ in which each row represents the opinions *of some* peer *on all* other peers, and each column represents the opinions *of all* peers *on some* peer. M is called the *reputation matrix*. If $i \xrightarrow{r} j$ occurs, then the outcome (whether there is a response or not) is recorded in m_{ij}. M is a mutable data structure that changes its state after each round.

Each cell m_{ij} has the form $(c, d) \in \mathbb{R}^+ \times \mathbb{R}^+$, where c is a counter on the number of responses, and d is a counter on the requests that went unresponded. We write c_{ij} and d_{ij} respectively for the two parts of cell m_{ij}. For each $i \xrightarrow{r} j$, if $i \xleftarrow{r} j$ occurs, then $c_{ij} := c_{ij} + 1$; if $i \not\xleftarrow{r} j$ occurs, then $d_{ij} := d_{ij} + 1$.

To reduce the impact of the distant past on the current reputation values, we introduce a decay factor $0 \leq \gamma \ll 1$ to slowly erase the traces of behavior: Before updating cell m_{ij},

$$c_{ij} := (1 - \gamma) c_{ij}$$
$$d_{ij} := (1 - \gamma) d_{ij}$$

Where not stated otherwise, we consider $\gamma = 0$ in the following.

The intention behind the reputation matrix is that the information stored in M can be used by any peer i to estimate the supply level v_j of any other peer j prior to the first encounter. To get there, i needs to run a *reputation function* $\Omega_i(M, j)$ on the information stored in the matrix that correlates

with v_j as closely as possible (if there is only one global reputation function used by all nodes, we write $\Omega(M, j)$, omitting the i subscript). Here are two straightforward alternatives, average and median:

$$\Omega^{a*}(M,j) = \begin{cases} \frac{\sum_{k \in I_j} c_{kj} - d_{kj}}{|I_j|} & \text{if } I_j \neq \emptyset \\ 0 \quad \text{otherwise} \end{cases}$$

$$\Omega^{m*}(M,j) = \begin{cases} \mathbf{M}\mathrm{D}(\{c_{kj} - d_{kj}\}_{k \in I_j}) & \text{if } I_j \neq \emptyset \\ 0 \quad \text{otherwise} \end{cases}$$

$$I_j = \{k \in \mathcal{N} | c_{kj} \neq d_{kj}\}$$

where I_j is the set of peers that are not indifferent about j. (A peer k is indifferent about j iff $c_{kj} = d_{kj}$.)

Only taking non-indifferent opinions into account is crucial, particularly when computing the median. For instance, consider a network of 64 peers in which 30 peers hate j, only 1 peer loves j, and all remaining 33 peers have never met j ($c_{\cdot j} = d_{\cdot j} = 0$). Then the $\Omega^{m*}(M, j)$ would be 0 if all opinions, indifferent or not, would be considered. As a consequence of this decision, if $I_j = \emptyset$, then $\Omega(M, j)$ needs to be explicitly set to 0, meaning undecidedness (else it would be undefined).

The output of any Ω function is more convenient to handle if its domain is restricted to $[0, 1]$, so we normalize.

$$\Omega^{a}(M,j) = \frac{1}{2} + \frac{\Omega^{a^*}(M,j)}{2 \cdot \tau_j}$$

$$\Omega^{m}(M,j) = \frac{1}{2} + \frac{\Omega^{m^*}(M,j)}{2 \cdot \tau_j}$$

$$\tau_j = \max_{k \in I_j} \{|c_{kj} - d_{kj}|\}$$

τ_j is the most extreme absolute of all opinions on j. It is easy to see that $\Omega^{a,m}(M, j) \in [0, 1]$, and that $\Omega^{a,m}(M, i) > \Omega^{a,m}(M, j)$ holds iff $\Omega^{a^*,m^*}(M, i) > \Omega^{a^*,m^*}(M, j)$. A reputation value of 0 means 'as bad as it gets in this network', 1 means 'as good as it gets', and $\frac{1}{2}$ means 'average'.

This normalized reputation can be used as probabilities of cooperation, and renders $\Omega(..) = 1$ the maximal reputation at all times. Further, it differentiates between negative reputation and indifference.

However, it still fails to distinguish "has had mixed experiences" from "has had no experiences at all". This distinction is covered by the following

$\Omega^{a/w}$ that puts a weight on a reputation value that is derived from the number of transactions that the node has been involved in (normalized versions as above).

$$\Omega^{a/w*}(M,j) = \begin{cases} \frac{\sum_{k\in I_j}(c_{kj}+d_{kj})(c_{kj}-d_{kj})}{|I_j|} & \text{where } I_j \neq \emptyset \\ 0 \quad \text{otherwise} \end{cases}$$

$$\Omega^{m/w*}(M,j) = \begin{cases} \mathbf{M}\mathrm{D}(\{(c_{kj}+d_{kj})(c_{kj}-d_{kj})\}_{k\in I_j}) & \text{where } I_j \neq \emptyset \\ 0 \quad \text{otherwise} \end{cases}$$

$$I_j = \{k \in \mathcal{N} | c_{kj} \neq d_{kj}\}$$

A few idealized reputation functions will help to understand the visualization of our results below:

$$\Omega^v(M,j) = v_j$$
$$\Omega^w(M,j) = w_j$$
$$\Pr[\Omega^{\mathrm{rnd}}(M,j) < x] = x \quad \text{(output of } \Omega^{\mathrm{rnd}} \text{ is uniformly random)}$$

Ω^v provides noise-free access to supply levels, which helps direct resources to the most cooperative nodes and thereby counters free riding; Ω^w provides noise-free access to demand levels and allows to identify greedy nodes; and Ω^{rnd} provides maximum noise and returns uniformly random values that are fully dis-correlated from any relevant features of the subject node. This is the worst reputation system possible and can be used as the extrem opposite to Ω^v (or Ω^w).

Our reputation matrix, proposed independently of the EigenTrust matrix (KSGM03), differs from its competitor in a few crucial aspects. For once, EigenTrust does not differentiate between no experience and bad experience. Also, normalization takes place on a subjective level. $\Omega_i^{\mathrm{EigenTrust}}(M,j) = 1$ means that node j has given node i the best subjective experience, but it does not say anything about node j's objective quality. This allows for node-local normalization, which gives better performance at lower accuracy. What makes up for the loss in accuracy is the trick of finding a node's reputation in an efficiently computable eigenvector of the reputation matrix that gave EigenTrust its name (see Section 8.4.3). If reputation is about anything but truthworthiness (e.g., upload performance), there is a difference between considering a node a reliable source of reputation information and considering it a highly reputable node, so the EigenTrust algorithm does not apply any more. Whether the two approaches can be combined to get the benefits from

both is an intriguing, but non-trivial question that exceeds the scope of this work.

9.1.2 Hostile Peers

When it comes to understanding the adversary, different approaches can be found in the literature, ranging from outright avoiding the problems imposed by the nature of pseudonymous P2P applications (Miq04), over populating a fraction of the network with honest but lazy peers and forbidding liars (BB03), to the rigid construction of equilibria in non-hostile strategies for games where corrupting the reputation information is a valid option (KSGM03). In this thesis, we attempt none of these. The first is not very useful, and the last requires very strong assumptions elsewhere in the formalism for the equilibria to hold, e.g., the assumption of pre-trusted peers as in (KSGM03). Instead, we allow for plugging arbitrary assumptions on the hostility of peers into the model and predict their impact on overall system performance.

Note that we still give a formal description of the adversary in the spirit of Section 2.3. The difference to Part I is that there the system was broken once the adversary managed to have *non-negligible* impact on it. In P2P networks, we accept what cannot be helped (that the the risk of encountering adversarial nodes is non-negligible), and rather than considering P2P networks dead, we measure the negative impact for any chosen set of system parameters.

Remember a peer's hostility parameter e_i is defined as the probability that i "misbehaves". As a first option for a semantics of misbehavior, we propose the *cautious liar* model in which adversaries only record one event per request sent, but may be lying about the response:

Definition 9.1. Cautious liars (CL). *Peer i is said to misbehave in transaction $i \xrightarrow{r} j$ if instead of recording the real outcome, i records $i \xleftarrow{r} k$, where k is chosen uniformly at random such that $k \neq i, e_k > 0$.*

This definition creates a scenario in which adversaries are organized in a group in which everybody knows each other, and reputation within that group is boosted uniformly instead of the reputation of the real contributors. By modelling a less organized set of adversaries, we might obtain insights into the performance of different hostile strategies, but we do not expect the impact on the experience of honest nodes to be affected too much.

We assume that only peer i has write access to the row vector m_i. Not even hostile peers can alter or suppress other peers' recordings. The cryptographic data structures capable of giving these guarantees if M is stored

and maintained in a distributed fashion by all peers are under development, and several promising approaches are already available (see Section 9.2). Of course, implementation is trivial if we allow for one or more centralized TTPs (e.g., an organization that also coordinates a monetary infrastructure, given there is one).

But another form of malicious behavior is possible: In many scenarios, liars are likely to be lazy at the same time. In fact, the only reason for an adversary to contribute resources may be that others are less likely to track it down if it behaves exactly as everybody else.

Definition 9.2. Lazy liars (LL). *Lazy liars misbehave just like cautious liars. In addition, in the lazy liars setting, the supply parameter of peer i is $v_i = 0$ if $e_i > 0$.*

So in the lazy liars setting, misbehavior consists of both poisoning the reputation matrix and starving the network of resources.

A hybrid variant between these two adversary types is the lazy liar that *sometimes* contributes, but not quite enough:

Definition 9.3. Half-Lazy liars (HL). *A Half-Lazy liar i misbehaves just like a cautious liar. In addition, whenever an honest node would respond to a request, the half-lazy liar only does so with probability e_i.*

An analogous set of liar types covers *saboteur nodes* whose aim is to lower the reputation of productive and cooperative nodes for reasons of vandalism or hostility towards the goal of the network's application (such as the content industry in the case of unlicensed content distribution networks):

Definition 9.4. Cautious saboteurs. *Every hostile peer maintains a measured approximation V^* of the supply level distribution V in the network. Peer i is said to misbehave in transaction $i \xrightarrow{r} j$ if instead of recording the real outcome, i records $i \not\xrightarrow{r} k$, where k is chosen uniformly at random such that $v_k > avg(V^*)$.*

Variants analogous to the lazy liar types are straight-forward. Despite saboteurs being an interesting topic, we will focus on reputation boosting liars in our simulations. We expect that our results apply equally to both categories, and will leave the confirmation for future research.

9.1.3 Distribution Strategies

Which strategy should a peer follow when distributing its resources such that (a) individual response rates and (b) the network performance are maximized? There are often several options for the network designer, or for a peer

during the network run, that have non-trivial consequences. We consider two basic strategies and combinations and variants of them:

Definition 9.5. Constant resource limit (CRL). *In each round, peer j registers all requests $i \xrightarrow{r} j$ received and orders them with respect to the senders' reputation values $\Omega(M, i)$. The v_j senders with the highest reputation are served, the remaining requests are ignored.*

CRL is useful for describing resources such as upload bandwidth in traditional file pooling networks: The resource is limited, but the peer does not really care how it is used, or whether it is used at all.

Definition 9.6. Constant reputation threshold (CRT^α). *If $i \xrightarrow{r} j$, i is served by j iff $\Omega(M, i)^\alpha \geq \Omega(M, j)$.*

$\alpha \geq 0$ is a global threshold parameter. If $\alpha = 0$, then any request is answered; if $\alpha = 1$, i must have higher reputation than j for j to answer i's request. For $\alpha \to \infty$, j gets more and more picky.

That we use $\Omega(M, j)$ here to approximate v_j, although the latter is directly available to j and may contain considerably less noise, may seem a little odd. However, it makes notation easier, since we can plug in different reputation functions and always get comparable figures for both peers involved in a transaction. In fact, in the simulations we have also experimented with globally constant thresholds and v_j instead of $\Omega(M, j)$.

CRT^α is tailored towards applications in which resource limits are more dynamic than bandwidth. For instance, a peer providing access to a virtual machine or to a virtual hard disk may decide to leave a certain fraction of its resources unused for times of heavy load, and thus achieve short reaction times for peers with high supply values; or a hosted server running on a monthly traffic limit may decide to use this traffic for peers that exceed a given reputation limit.

Further distribution strategies will be introduced together with the resulting simulations.

9.2 Motivation

Our model is both very abstract and flexible, and restricts adversaries in ways that are not usually assumed by related research or implemented in existing P2P applications. Therefore, before the presentation of the simulation results, we need to discuss a few aspects of realization. In particular, we need to hint at how we will enforce the assumptions made by the model.

9.2.1 Routing and Identity

Often, P2P networks use merely the IP address or a cookie created by a newly connected client (as there is no web server that can create it). This gives room to all the identity attacks considered in Section 8.1. In contrast, our model makes a number of assumption on precautions that are taken on the network level (without bothering the user).

1. Each node has a strong identity established by an asymmetric key pair. All outbound messages will be signed using this key pair, and signatures on all incoming messages are verified using the public keys (for performance, hybrid schemes are used that encrypt all traffic after the initial handshake symmetrically). Public keys are exchanged when two nodes connect for the first time.

   This does not keep an adversary from joining a network, but it rules out the possibility that an adversary assumes the identity of a node that has a high reputation, since the secret key needed in order to establish a connection under that node's name is safely stored on the node and never revealed.

2. To reduce the risks of Sybil attackers that create many identities, the techniques presented in 8.1.1 such as SybilGuard are deployed.

   Our model takes the number of adversarial nodes and the level of cooperation of those nodes as parameters. Hence, depending on assumptions of the dedication of hostile players and on the robustness of the countermeasures taken, we can use the model to describe a wide range of scenarios.

   By using SybilGuard to assume limits on the number of Sybil nodes in the system, we also suggest that the two problems of Sybil attack and reputation scheme are entangled (instead of orthogonal, as stated in (YKGF06) in the related work section).

3. If we assume a TTP, each adversary actually only gets to play one identity. This comes at the cost that has been outlined in Section 8.1.2, but the advantages may be greater than the drawbacks.

This cryptographic infrastructure works nicely together with DHTs if the DHT address space is spanned by the (hashes of the) public keys of the nodes. If i wants to talk to j, it obtains the public key, uses it first to connect to j via the DHT, and then uses it again to establish an authentic communication link.

Note that since public key cryptography is used for (anonymous) authenticity, confidentiality comes virtually for free, both on the design and on the implementation level. The same cryptographic schemes can be used for both, and most libraries provide combined functionality.

9.2.2 The Reputation Matrix

The reputation matrix is essentially a two-dimensional hash table that is indexed with node names. This suggests using a DHT for storing reputation as well. Since the node responsible for storing cell (i, j) may be lazy or hostile, data can get corrupted. So in order to keep it reliable, signature schemes need to be used. As before in establishing secure direct communication links, these can be based on the identities of the nodes.

Assume node j wants to record the outcome of $i \rightarrow j$. In the simplest implementation, let M be a DHT that stores the reputation matrix in form of a collection of cell updates that all nodes can download and assemble at any time. j writes the update record to be stored in m_{ij} into a data packet, signs it, and submits the result to the DHT M under the key j. If k wants to compute the reputation of node j, its collects all the records stored in M under j, checks the signatures, and adds up the contents.

Queries are frequent events in many applications, and the performance is an issue. In this simple implementation, every record ever stored needs to be signature-checked over and over again, and many signatures need to be checked in order to compute a single reputation value. Hence, there is needs for improvement.

Further, there is no precaution against an adversary removing records from the DHT, since k knows only of those records that it can download, and can not distinguish between a record that has been stolen and a record that never was submitted. So a cryptographic data structure needs to be installed to make record loss detectable.

The answer to both needs lies in finding a good hash tree structure (see Section 8.3). If records are not signed individually, but as updates to an already existing collection of records that has a single tree signature, then k only needs to do one signature verification for each reputation value it needs to compute. Furthermore, if a node in the hash tree is missing, signature verification will fail, so k can act upon it.

There are many leads to be followed in order for the application designer to find the best solution. Most of them represent trade-offs between security and performance.

- Fully computed reputation values can be signed and cached, and if

enough signatures on a cached value have been stored, a new hash tree is created with the result of the old tree as the root update.

- Backup nodes can be used to recover stolen data if hash tree checks fail.

- A meta-reputation system can be used to give incentives to the nodes of M to not steal any records. This is a particularly interesting challenge to realize, and it is completely unclear how the two reputation systems would interact. It also constitutes even better proof than the CAN feedback scheme mentioned in Section 8.4.1 for the claim that mixing different levels of reputation is risky business (here: reputation for being a good peer, and reputation for taking good care of the reputation matrix). If a node can gather negative reputation by stealing previous bad reputation, it will probably not care, since it can steal the new bad reputation as well.

- If an uninvolved *witness node* is able to observe a transaction, it can contribute an additional signature to the record of that transaction. This increases the robustness of the overall scheme, but open the door to other forms of lying and complicates our assumptions on what an adversary can do to the reputation matrix.

In our model, we have made the simple assumption that node i can record any events on outcomes of queries that it has sent itself. If i tries to record reputation on l that has been given by j, the signature will not check, and the DHT M will reject storage.

If a request has never been sent and i attempts to submit a record on its outcome anyway, cryptography is ineffective. However, statistical methods like SybilGuard can help identifying nodes that send out significantly more records than queries. (This is another promising idea that to our knowledge has not been tackled elsewhere yet.)

9.2.3 Is There a Prisoner's Dilemma?

Our model assumes unchanging node characteristics. In particular, it assumes that a fixed fraction of all users is infallibly honest. This is consistent with observation in existing P2P networks, but stands in sharp contrast to the traditional game theoretical point of view that each individual does whatever gives it the highest chance to achieve the highest individual utility. Game theoreticians, when confronted with their inability to describe

cooperation-driven phenomena, argue that their models are not accurate descriptions of every single state of a system in time, but a fix point into which the system will collapse eventually. (We have touched this issue briefly in the introduction to game theory in Section 7.5.)

This means that if game theory is correct, this could mean that Napster was bound to fail merely because it represents another instance of the tragedy of the commons, even without the opposition that it faced from the content industry and the competition to more modern file sharing networks that it eventually lost. The participants simply would have contributed less and less content for others to download until nothing would have been left. Worse, it could mean that newer file sharing networks, and the very idea of P2P networks, are threatened by the real, non-transitory behavior of peers in a matured system.

Are P2P networks bound to suffer from PD? Although no theory can give a definite answer to this question, there are models that leave room for optimism. One example is an evolutionary game proposed in (FZ00) that shows an increase of cooperative nodes over time despite its PD-like structure.

In an evolutionary game, nodes do not change their strategy over time, but reproduce and mutate. Strategies of descendents are either the strategies of their ancestors, or deviate from that in certain ways. In (FZ00), two tribes of individuals are formed, and every individual is either cooperative or defective by genetics. After each round, utility is processed on two levels. On the intra-tribe level, nodes with low utility reproduce slower than nodes with high utility, and due to the PD-like structure of the game, this gives an advantage to defective nodes. On the inter-tribe level, those tribes that have higher cooperation levels grow faster. This gives an individual in a more successful tribe an advantage over an individual in a less successful tribe.

With the right system parameters, the inter-tribe utility counters intra-tribe utility, as a cooperative node in a cooperative tribe is more successful than a defective node in a defective tribe. In other words, cooperative tribes produce more cooperative nodes than defective tribes can produce defective nodes (although defective tribes produce even fewer cooperative nodes). This effect is called the *Simpson's Paradox (SP)*, named after the statistician who first described it (Sim51).

Can SP be imposed on a P2P network to compensate PD? There is an apparent mismatch between an evolutionary game and a network in which nodes can alter their strategy freely at any moment. There are intricate psychological effects in any social context, but in P2P networks with high anonymity, those are not usually considered particularly suitable for breeding loyalty and altruism. How can we design a computer network that makes

nodes (a) stick with their tribe and (b) develop strategies over time according to the evolutionary model?

Inspired by the game sketched here and returning to the assumption of pre-trusted peers also made in (KSGM03), we propose *tribe seeds* that consist of P2P network nodes that know and trust each other personally. Keeping the level of trust above a certain threshold, each tribe seed carefully admits new and unknown nodes, and ejects those nodes whose performance deteriorates. Tribes of this kind could leverage collective reputation and be better off than individuals whose reputation is based merely on their own contributions. At the same time, a good ejection algorithm would limit the amount of free-riding in each tribe. Both this and the higher attractivity of cooperative tribes would makes cooperation guarantee faster tribe growth.

Maintaining tribe membership is a complex, but possible task. Each individual creates a public-key identity that is signed beforehand by its personal contacts. An additional public-key identity is created collectively by the tribe so that each tribe member can sign for the tribe and both boost and leverage reputation for the tribe. Intra-tribe reputation mechanisms need to be maintained in order to identify and eject defective nodes. Inter-tribe reputation can be used to decide on new admission rates.

Networks with tribes are likely to level the playing field, making underperformers better and over-performers worse off. This may create incentives for defection, but the incentives for cooperation (cooperators get to be part of cooperative groups, which are more successful) may still be stronger. Cryptography, existing or yet to be developed, will also play an important role in the future in making it harder to defect undetected.

However, these are just preliminary ideas for future research. The complexities of any model necessary for representing multiple reputation schemes and multiple levels of social interaction are prohibitive at this point. In our much simpler model, we use these ideas as theoretical evidence that high levels of cooperation can last in P2P networks, complementing the much stronger empirical evidence.

9.3 Simulation Results

On of the main contributions of this thesis to the subject of P2P networks is the *esim* tool for exploring new and original situations in P2P networks, and for looking at previously explored topics from a different angle. In this section, we demonstrate what esim can do and present the output of selected simulation runs. This will lead to a few interesting observations about the mechanics of reputation systems.

We have implemented esim from scratch in the lazy functional programming language Haskell[1]. It is based on an earlier implementation in Python[2], which in turn was an adaption of Phil Jones' optimaes[3]. esim outputs LaTeXand gnuplot[4] source. The results in this thesis have been directly generated and included from our simulator. esim is open source and available for download.[5].

All simulations are run on fully connected networks with 70 nodes that run 100 rounds. For the statistics, each simulation is run 10 times, and the average over all statistics is used in the output. We have run selected experiments with larger parameters with similar results to confirm these choices are valid. To avoid cluttering the layout, we have placed most of the Figures in Section 9.3.5. The data displayed using gnuplot is perturbed by a factor of 0.01 (0.005 in every direction). This keeps lines from fully overwriting identical other lines, which makes the results in some of the outputs more visible.

9.3.1 Idealized reputation functions Ω^V, $\Omega^{\mathbf{rnd}}$

The first networks we run have honest players only:

$$Pr[E = 0] = 1$$

For illustration, we start with the pathological case that the values returned by the reputation scheme are random, i.e. there is no information on supply levels contained in reputation levels. The output is represented in a graph with one point for each node in each round (see Figure 9.1, top). The game proceeds in rounds from back to front, the vertical planes represent supply levels, and the (much less distinct) horizontal planes are reputation levels. A second view on the same game is a 2-dimensional projection on the average over all rounds, visualizing the correlation between supply level, reputation, and utility (see Figure 9.1, bottom). The lines in the 2-dimensional figure always connect the highest y-point on one x-point with the lowest on the next one, so diagonal lines suggest uniform randomness. In marks the extreme points on the y-axis for each point on the x-axis, and gives an idea of the distribution (in this case, both reputation and utility are uniformly random). (Note that this idea is not always reliable, and the distribution may be highly skewed and the extreme points still be the same for each x.)

[1] `http://www.haskell.org/`
[2] `http://www.python.org/`
[3] `http://www.nooranch.com/synaesmedia/optimaes/optimaes.cgi?HomePage`
[4] `http://www.gnuplot.org/`
[5] `http://www.etc-network.de/~fis/software/esim/`

Pearson correlation values over supply level V and reputation $\Omega(V)$, reputation $\Omega(V)$ and utility $\pi(V)$, and supply level V and utility $\pi(V)$, are all near 0:

	avg	sdev
$\rho(V, \Omega(V))$	0.00	0.01
$\rho(\Omega(V), \pi(V))$	-0.00	0.01
$\rho(V, \pi(V))$	-0.00	0.01

The *Pearson correlation coefficient* ρ_{xy} is defined on two sample sets x, y as follows:

$$\rho_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y}$$

where $\bar{x}$ is the arithmetic mean over set x, and s_x is the standard deviation. Pearson correlation is 1 if there is an $y_i \in y$ for each $x_i \in x$ such that $x_i = y_i$; it is -1 if such a mapping exists with $x_i = -y_i$, and it is 0 if the two sets are statistically independent. We use ρ on random variables and sample sets interchangeably, and for any function f and random variable X we write $f(X)$ for the distribution of $f(x)$ if x is drawn from X. The table above shows two numbers for each pair of sample sets: Average and standard deviation of correlation over the 10 simulation runs.

The next artificial extreme we consider is ideal supply-based reputation Ω^v, such that each node has noise-free information on the supply levels v_i of all other nodes (see Figure 9.2). This is the ideal that any reputation scheme wants to approximate, and that liars try to distort.

This achieves the expected correlation of 1.0:

	avg	sdev
$\rho(V, \Omega(V))$	1.00	0.00
$\rho(\Omega(V), \pi(V))$	0.61	0.00
$\rho(V, \pi(V))$	0.61	0.00

Correlation with utility is still less than 1 because the supply and demand distributions allows for situations in which four nodes with $v = 3$ send queries to the same node with $v = 1$, and three end up empty-handed despite their optimal reputation. With more complex rules that model more sophisticated resource allocation schemes, these figures are likely to improve, but at the cost of simplicity of our model.

9.3.2 Distribution strategies: CRL vs. CRT

Now that we have seen the worst and best reputation functions possible, we can look at how reputation matrices perform. To begin with, CRL with Ω^{avg} yields high positive correlation between production and reputation (see Figure 9.3). As long as nobody lies, productive nodes have higher reputation, and higher reputation means higher utility. In other words, both reputation and utility correlate with productivity:

	avg	sdev
$\rho(V, \Omega(V))$	0.94	0.00
$\rho(\Omega(V), \pi(V))$	0.63	0.00
$\rho(V, \pi(V))$	0.61	0.01

Next, to find out how robust this outcome is with respect to distorted initial system states, we initially assign uniformly random reputation values to all nodes (see Figure 9.4).

The following simulations have been run with such a pre-seeded reputation matrix: Initial reputation (n, m) means that n cooperation events and m defection events have been recorded. Initial reputation $(X, 10 - X)$ means that each cell in the matrix is initialized with two random numbers for cooperation and defection such that $c + d = 10$.

Due to the randomness early on in the simulation, the entire game exhibits worse correlation values than in the above experiments where the matrix was initialized with $(0, 0)$. However, there is a big improvement when looking only at last third of the rounds, after the initial noise has been sanitized with some rounds of node experience:

$M = (X, 10 - X)$, all rounds	avg	sdev
$\rho(V, \Omega(V))$	0.64	0.06
$\rho(\Omega(V), \pi(V))$	0.65	0.00
$\rho(V, \pi(V))$	0.42	0.04
$M = (X, 10 - X)$, last third	avg	sdev
$\rho(V, \Omega(V))$	0.80	0.04
$\rho(\Omega(V), \pi(V))$	0.67	0.01
$\rho(V, \pi(V))$	0.53	0.04
$M = (0, 0)$, all rounds	avg	sdev
$\rho(V, \Omega(V))$	0.94	0.00
$\rho(\Omega(V), \pi(V))$	0.63	0.00
$\rho(V, \pi(V))$	0.61	0.01
$M = (0, 0)$, last third	avg	sdev
$\rho(V, \Omega(V))$	0.97	0.00
$\rho(\Omega(V), \pi(V))$	0.64	0.00
$\rho(V, \pi(V))$	0.62	0.01

This proves that our reputation scheme is self-stabilizing, and that it can recover from noise, at least if it is temporary.

Next, we will play with scarcity and abundance of resources. For extremely high demand (every node gets a request from every other node every round, see Figure 9.5, top), correlation between supply V and reputation $\Omega(V)$ increases, whereas correlation between reputation $\Omega(V)$ and utility $\pi(V)$ decreases due to the many defections in case of resource exhaustion. Nevertheless, both are still quite high.

$V = [1, 2, 3, 4], W = [70]$, last third	avg	sdev
$\rho(V, \Omega(V))$	0.99	0.00
$\rho(\Omega(V), \pi(V))$	0.33	0.01
$\rho(V, \pi(V))$	0.34	0.02

(Average utility and reputation deteriorate, but this is due to scarce resources, not to the reputation scheme.)

If there are more than enough resources, but not enough to rule out exhaustion in hotspots, reputation is less accurate, i.e. advantageous for less productive nodes (see Figure 9.5, bottom).

$V = [1, 2, 3, 4], W = [1]$, last third	avg	sdev
$\rho(V, \Omega(V))$	0.80	0.02
$\rho(\Omega(V), \pi(V))$	0.17	0.02
$\rho(V, \pi(V))$	0.16	0.01

Note that utility correlations are both quite low, although still positive, which makes free riding more attractive. In the extreme case where all requests are answered, it is obvious that any reputation scheme must defeated:

$V = [70], W = [1]$, last third	avg	sdev
$\rho(V, \Omega(V))$	0/0	0/0
$\rho(\Omega(V), \pi(V))$	0/0	0/0
$\rho(V, \pi(V))$	0/0	0/0

(Pearson correlation on sample sets with only 1 sample is undefined, as the definition yields 0 in both numerator and denominator. Similarly to the case of $\rho = 0$, the interpretation is that there is no correlation.)

However, high supply and low demand lead to resources being not scarce any more, and utility is likely to be high even without a reputation scheme. These three example configurations give evidence that in most cases where a reputation scheme is actually needed, they can do good.

Resource distribution using CRT is generally more challenging to configure properly (see Figure 9.6). CRT has bad correlation properties for virtually any threshold. Either nodes are too picky, reputation deteriorates, and utility converges to 0. Or nodes are not picky enough, and utility converges to 1.

With node-local thresholds (Figure 9.7), every node serves only nodes with reputation, say, at least as high as its own supply level (ideal reputation). Correlation $\rho(\Omega(V), \pi(V))$ is high, only unfortunately correlation $\rho(V, \Omega(V))$ is negative:

$t_i = v_i$, all rounds	avg	sdev
$\rho(V, \Omega(V))$	-0.39	0.12
$\rho(\Omega(V), \pi(V))$	0.51	0.07
$\rho(V, \pi(V))$	-0.16	0.07

Paradoxically, nodes with lower supply levels are more forgiving about low supply levels of others, so they give more and get better reputation. CRT is effective at punishing weak suppliers, but at the price of reducing the reputation of strong suppliers over time, ending up punishing all nodes, independently of their behavior.

$\alpha < 1$ makes nodes more generous, which makes slightly sharper correlation between $V, \Omega(V), \pi(V)$ than in Figure 9.6 (both on the desired positive figures and on the undesired negative figures):

$\alpha = 0.7$, all rounds	avg	sdev
$\rho(V, \Omega(V))$	-0.80	0.15
$\rho(\Omega(V), \pi(V))$	0.65	0.09
$\rho(V, \pi(V))$	-0.56	0.17

During our research, up to this experiment we have always been able to explain the outcomes where we did not anticipate them. The higher correlation figures for smaller α is the first effect that we have not been able to explain satisfactorily.

$\alpha > 1$ makes nodes act uncooperative towards nodes with higher supply levels than their own. This has two negative effects: It makes the negative correlation between supply and reputation deteriorate (with reputation converging to zero), and it makes utility converge to zero, because no queries are answered any more. Consequently, the figures are not very good, and we can conclude that α should be chosen such that $0 < \alpha < 1$:

$\alpha = 1.3$, all rounds	avg	sdev
$\rho(V, \Omega(V))$	-0.28	0.02
$\rho(\Omega(V), \pi(V))$	0.50	0.04
$\rho(V, \pi(V))$	-0.09	0.01
$\alpha = 3.0$, all rounds	avg	sdev
$\rho(V, \Omega(V))$	0.00	0.05
$\rho(\Omega(V), \pi(V))$	0/0	0/0
$\rho(V, \pi(V))$	0/0	0/0

So far, CRT has not established the accuracy that we have hoped for. To conclude the discussion of distribution strategies, we therefore combine the two strategies CRL and CRT into a hybrid variant of CRL with local dynamic threshold.

Definition 9.7. CRL/CRT$^\alpha$ hybrid. *If $i \xrightarrow{r} j$, if $\Omega(M, i)^\alpha < \Omega(M, j)$, i is not served by j. Otherwise, it enters a CRL queue on node j and is served iff it is one of the v_j requesting nodes with the highest reputation.*

A node playing CRL/CRT sometimes runs out of resources and fails to cooperate with nodes with high reputation, just as nodes playing CRL. As nodes playing CRT, it sometimes holds resources back and defects although it is still able to cooperate.

While we can hope for stronger incentives to cooperate, the correlation values in our experiments are unaffected by the hybrid approach:

CRL only, all rounds	avg	sdev
$\rho(V, \Omega(V))$	0.94	0.00
$\rho(\Omega(V), \pi(V))$	0.63	0.00
$\rho(V, \pi(V))$	0.61	0.00
CRL/CRT$^{0.7}$, $t_i = 0.2$, all rounds	avg	sdev
$\rho(V, \Omega(V))$	0.94	0.00
$\rho(\Omega(V), \pi(V))$	0.63	0.00
$\rho(V, \pi(V))$	0.61	0.00

9.3.3 Hostile nodes

We now take a closer look at hostile nodes, their strategies, and the effect that has on the effectiveness of the reputation scheme and overall utility.

To achieve a drastic effect, we start with an overwhelming majority of 10 hostile nodes on one good node:

$$\Pr[E = 1] = \frac{10}{11}; \quad \Pr[E = 0] = \frac{1}{11}$$

This experiment shows two things (see Figure 9.8, top): Overall correlations deteriorate, and there are clear branching in the development of reputation, one branch (presumably populated by hostile nodes) pointing upwards and one (good nodes) downwards. A new projection of the simulation run confirms that the branching separates good from hostile nodes (see Figure 9.8, bottom): Here, the z-axis does not span rounds, but hostility levels, so we get one column for each pair (v_i, e_i) of productivity and hostility level (all points averaged over the entire game).

all nodes	avg	sdev
$\rho(V, \Omega(V))$	0.61	0.08
$\rho(\Omega(V), \pi(V))$	0.61	0.01
$\rho(V, \pi(V))$	0.51	0.04
$e_i = 0$	avg	sdev
$\rho(V, \Omega(V))$	0/0	0/0
$\rho(\Omega(V), \pi(V))$	0.09	0.07
$\rho(V, \pi(V))$	0/0	0/0
$e_i = 1$	avg	sdev
$\rho(V, \Omega(V))$	0.83	0.02
$\rho(\Omega(V), \pi(V))$	0.62	0.01
$\rho(V, \pi(V))$	0.55	0.01

It is obvious from the second block of these figures (honest nodes) that a majority of hostile nodes can kill our reputation scheme: All correlation between supply level and reputation is drowned by false data. Hostile nodes still benefit from cooperation (last block). However, computing an additional figure we can see that hostility gives better reputation than supply:

all nodes	avg	sdev
$\rho(E, \Omega(E))$	0.65	0.05

On the other hand, the LL strategy turns out to be a bad idea. For the same parameters as above, but with hostile nodes that simultaneously lie and defect, the reputation scheme is defeated once more, but both for honest nodes and for liars:

all nodes	avg	sdev
$\rho(V, \Omega(V))$	0.04	0.10
$\rho(E, \Omega(E))$	-0.63	0.11
$\rho(\Omega(V), \pi(V))$	0.08	0.03
$\rho(V, \pi(V))$	0.00	0.02
$e_i = 0$	avg	sdev
$\rho(V, \Omega(V))$	0/0	0/0
$\rho(\Omega(V), \pi(V))$	0.00	0.06
$\rho(V, \pi(V))$	0/0	0/0
$e_i = 1$	avg	sdev
$\rho(V, \Omega(V))$	0.00	0.08
$\rho(\Omega(V), \pi(V))$	0.09	0.02
$\rho(V, \pi(V))$	0.00	0.02

In fact, the strong negative correlation between hostility and reputation means that a liar can expect to be far less popular than an honest node, if supply levels are identical. So although sabotage may still be a valid motive for choosing LL, but for free riding nodes, CL is the only reasonable choice in this scheme.

Despite the conclusion that LL is a bad strategy for a malicious node, in the presence of a vast majority of malicious nodes who know that and choose CL, our reputation system fails. But what about smaller fractions of defectors?

If we consider two thirds, or even only one third or nodes with $e_i = 1$, the picture changes significantly to the better (see Figure 9.9, top and bottom, respectively).

	$\Pr[E=0]=\frac{1}{3}$ $\Pr[E=1]=\frac{2}{3}$		$\Pr[E=0]=\frac{2}{3}$ $\Pr[E=1]=\frac{1}{3}$	
all nodes	avg	sdev	avg	sdev
$\rho(V,\Omega(V))$	0.48	0.12	0.67	0.06
$\rho(E,\Omega(E))$	0.79	0.04	0.63	0.07
$\rho(\Omega(V),\pi(V))$	0.64	0.00	0.65	0.00
$\rho(V,\pi(V))$	0.37	0.08	0.43	0.04
$e_i=0$	avg	sdev	avg	sdev
$\rho(V,\Omega(V))$	0.79	0.07	0.91	0.01
$\rho(\Omega(V),\pi(V))$	0.28	0.08	0.54	0.03
$\rho(V,\pi(V))$	0.20	0.06	0.50	0.03
$e_i=1$	avg	sdev	avg	sdev
$\rho(V,\Omega(V))$	0.83	0.02	0.86	0.04
$\rho(\Omega(V),\pi(V))$	0.57	0.01	0.57	0.05
$\rho(V,\pi(V))$	0.51	0.02	0.49	0.04

Hostility and reputation still correlate positively, so defecting is still attractive. Correlation between supply and reputation over all nodes is at 0.67 again for one third of malicious nodes, and still 0.48 for two thirds. More importantly, if we consider honest nodes only, these figures are similar to those for simulation runs with no liars at all.

9.3.4 Sparse networks

All networks that we have simulated up to this point have been fully connected. In practice, this is the case only in some scenarios, such as in DHTs where the end points of each communication are determined by uniformly distributed hash values. However, this uniform behavior is not always desirable:

- A system in which each node knows only a constant number of neighbors independent of the network size scales better. These networks tend to have constant routes, too, and therefore constant routing costs.
- If nodes can benefit better from each other if they share common interests or other properties, then communication local on this topology of interests is more efficient. For example, a distributed research article library is more useful at the same cost if each node only stores articles relevant to its user.

For these reasons, we close this chapter with a quick look at sparse network graphs. A sparse uniform simulation run has an additional parameter,

density $0 \leq S \leq 1$. S is the probability of any edge from the fully connected network graph to exist, so $S = 1$ yields fully connected graphs, and $S = 0$ yields the graph with no edges.

We play the game once more with one third of cautiously hostile nodes with $e = 1$, and two thirds of honest nodes. We take statistics over the last third of the game.

	$S = 1.0$		$S = 0.3$		$S = 0.1$	
all nodes	avg	sdev	avg	sdev	avg	sdev
$\rho(V, \Omega(V))$	0.67	0.09	0.74	0.05	0.61	0.09
$\rho(E, \Omega(E))$	0.70	0.04	0.50	0.05	0.11	0.05
$\rho(\Omega(V), \pi(V))$	0.67	0.00	0.66	0.02	0.61	0.04
$\rho(V, \pi(V))$	0.44	0.06	0.49	0.04	0.46	0.05
$e_i = 0$	avg	sdev	avg	sdev	avg	sdev
$\rho(V, \Omega(V))$	0.97	0.01	0.89	0.02	0.67	0.09
$\rho(\Omega(V), \pi(V))$	0.53	0.02	0.61	0.03	0.66	0.06
$\rho(V, \pi(V))$	0.50	0.02	0.54	0.03	0.53	0.06
$e_i = 1$	avg	sdev	avg	sdev	avg	sdev
$\rho(V, \Omega(V))$	0.92	0.03	0.79	0.07	0.69	0.13
$\rho(\Omega(V), \pi(V))$	0.53	0.06	0.57	0.04	0.30	0.10
$\rho(V, \pi(V))$	0.48	0.06	0.45	0.05	0.28	0.12

Fewer contacts at constant traffic volumes mean more experiences with those contacts. So our initial assumption was that sparseness would increase the quality of the reputation information. Instead, at least if fewer edges are still randomly distributed, and not concentrated in cliques, these results show a decrease in $\rho(V, \Omega(V))$ on both good and bad nodes individually, and in $\rho(E, \Omega(E))$ for all nodes, with decreasing density S. The former implies that if all contacts are random, more contacts give a node an advantage. The latter implies that the advantage of hostile nodes over honest nodes deteriorates with decreasing density (possibly due to the fact that the effectiveness of the reputation system in general deteriorates, which renders exploiting it more difficult).

The question of an optimal sparsity parameter is relevant for finding an optimal strategy in super-node topologies (see Figure 8.3 on p. 196). Remember that super-nodes have been introduced to overcome scalability issues by forming a backbone network at which sub-nodes connect in one point only. Routes to sub-nodes connected to the same super-node may be direct, but all routes to sub-nodes connected to the backbone elsewhere lead through their respective super-nodes.

It is plausible to assume that at least in some applications, the number of direct neighbors maintained by each node is independent of the network size. Under this assumption, the density of each network (or in super-node structured networks: of each sub-network) is a function of its size. By controlling the size of their sub-networks, super-nodes thus control their density. This raises the question of how dense a network should be in order to yield optimal performance.

In our case, the data suggests that optimal density is 1, which depending on the circumstances may be impractical for super-nodes to maintain. But an optimum that can not be reached can be aimed for, which may already help increasing output substantially. Further, if other reputation schemes are considered, new simulations need to be carried out that may produce different optima.

9.3.5 Figures

On the remaining pages of this chapter, we present the figures discussed in the preceding sections.

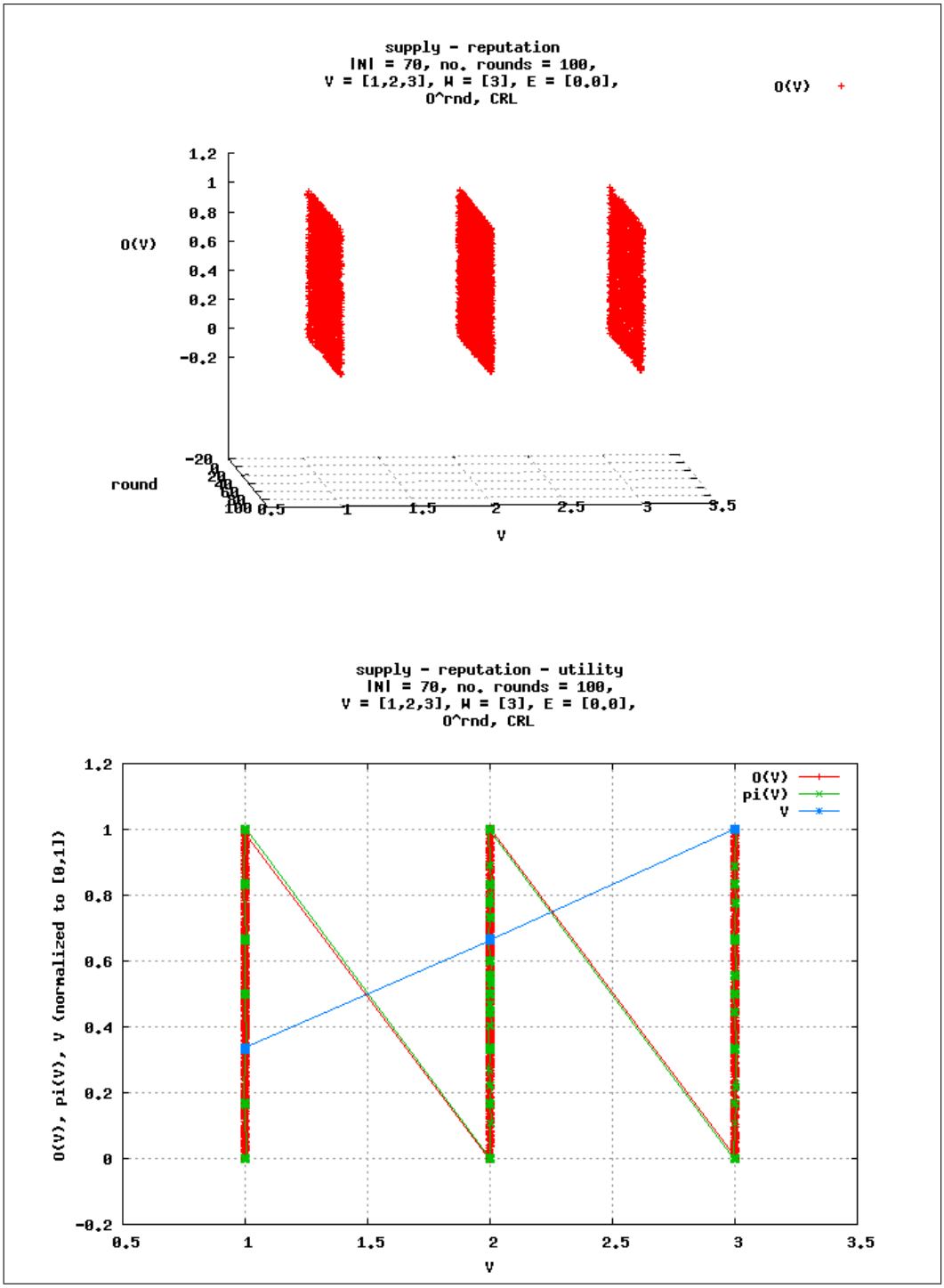


Figure 9.1: Ω^{rnd}

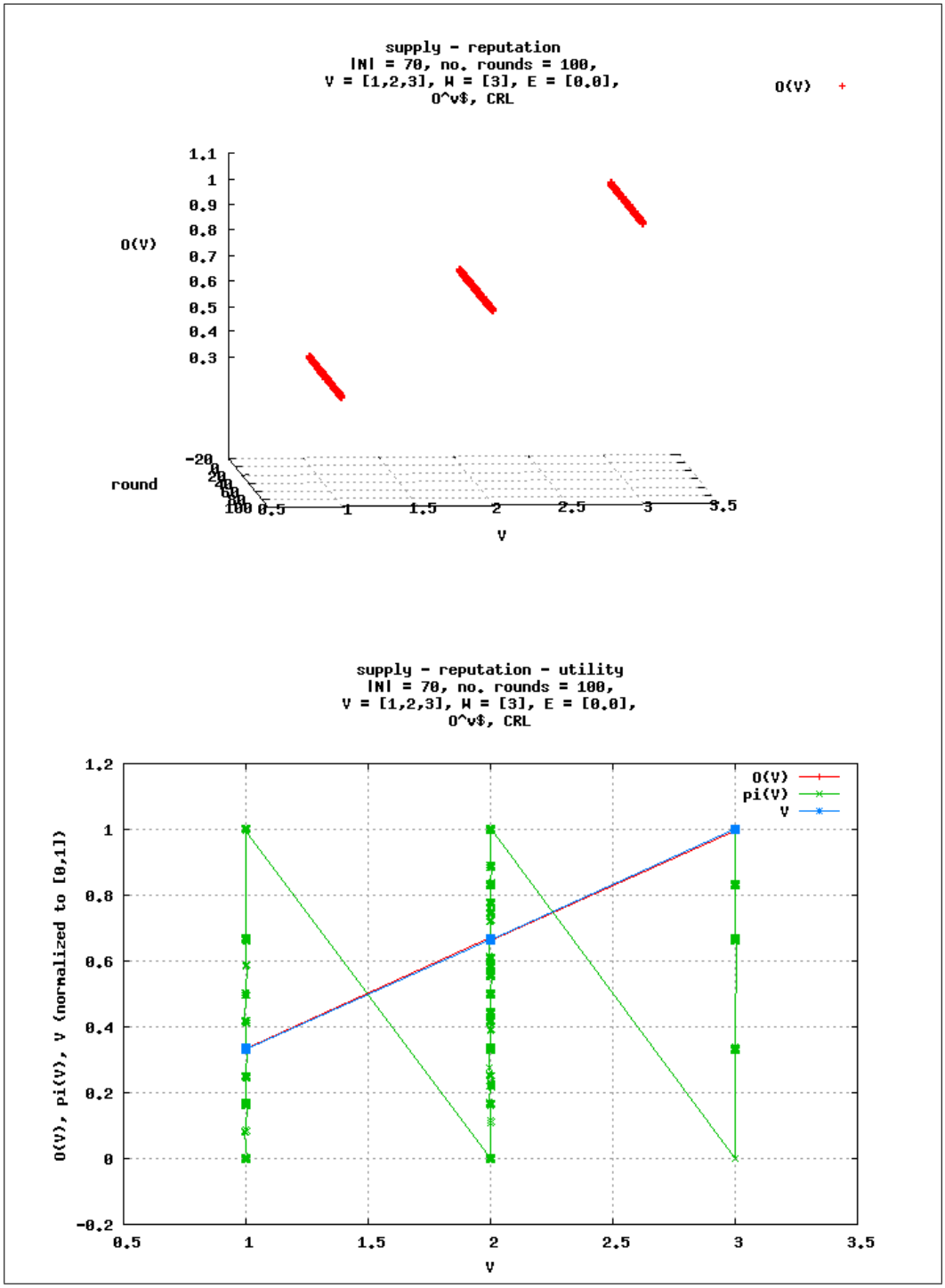


Figure 9.2: Ω^v

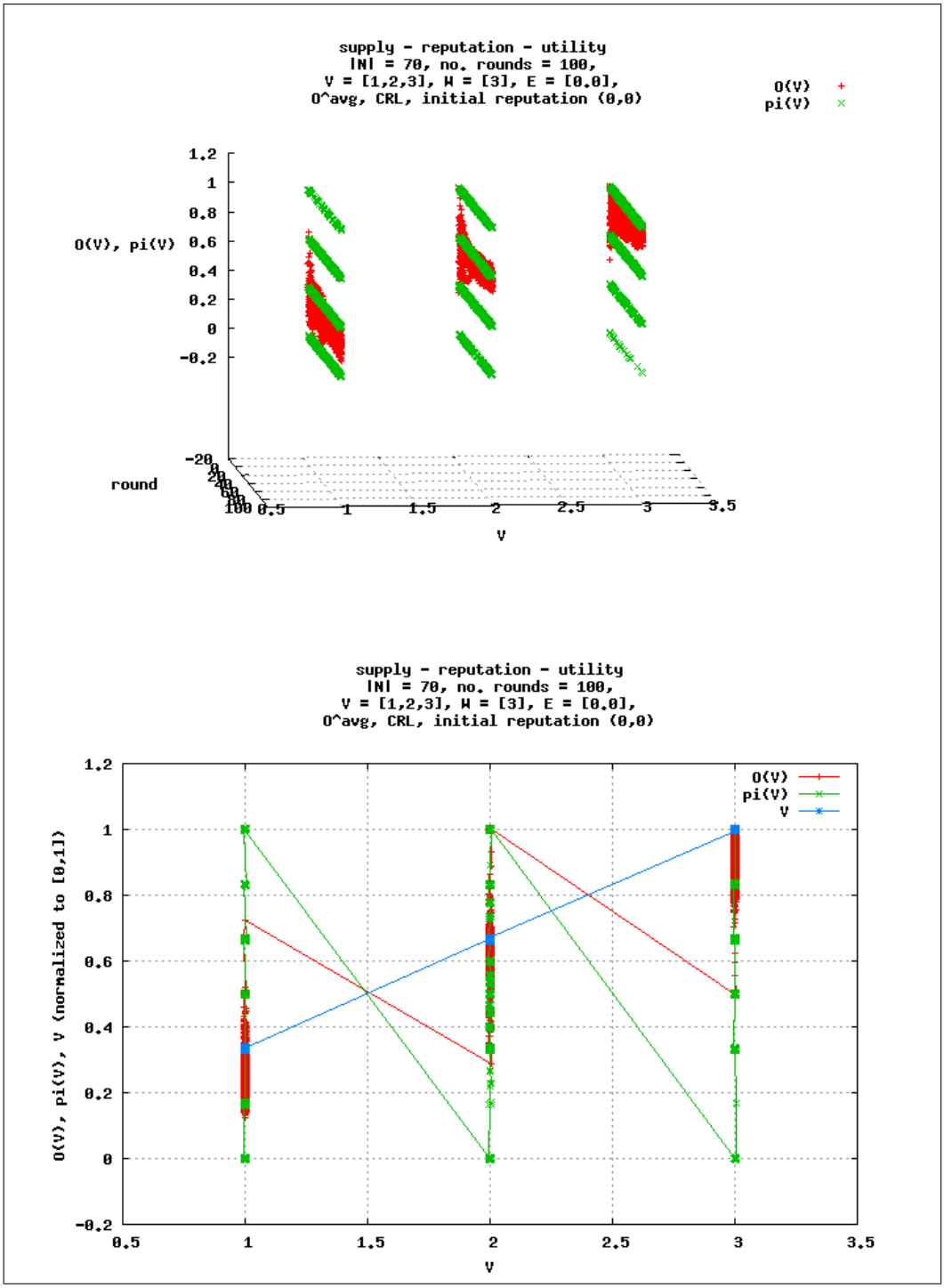


Figure 9.3: Ω^{CRL}

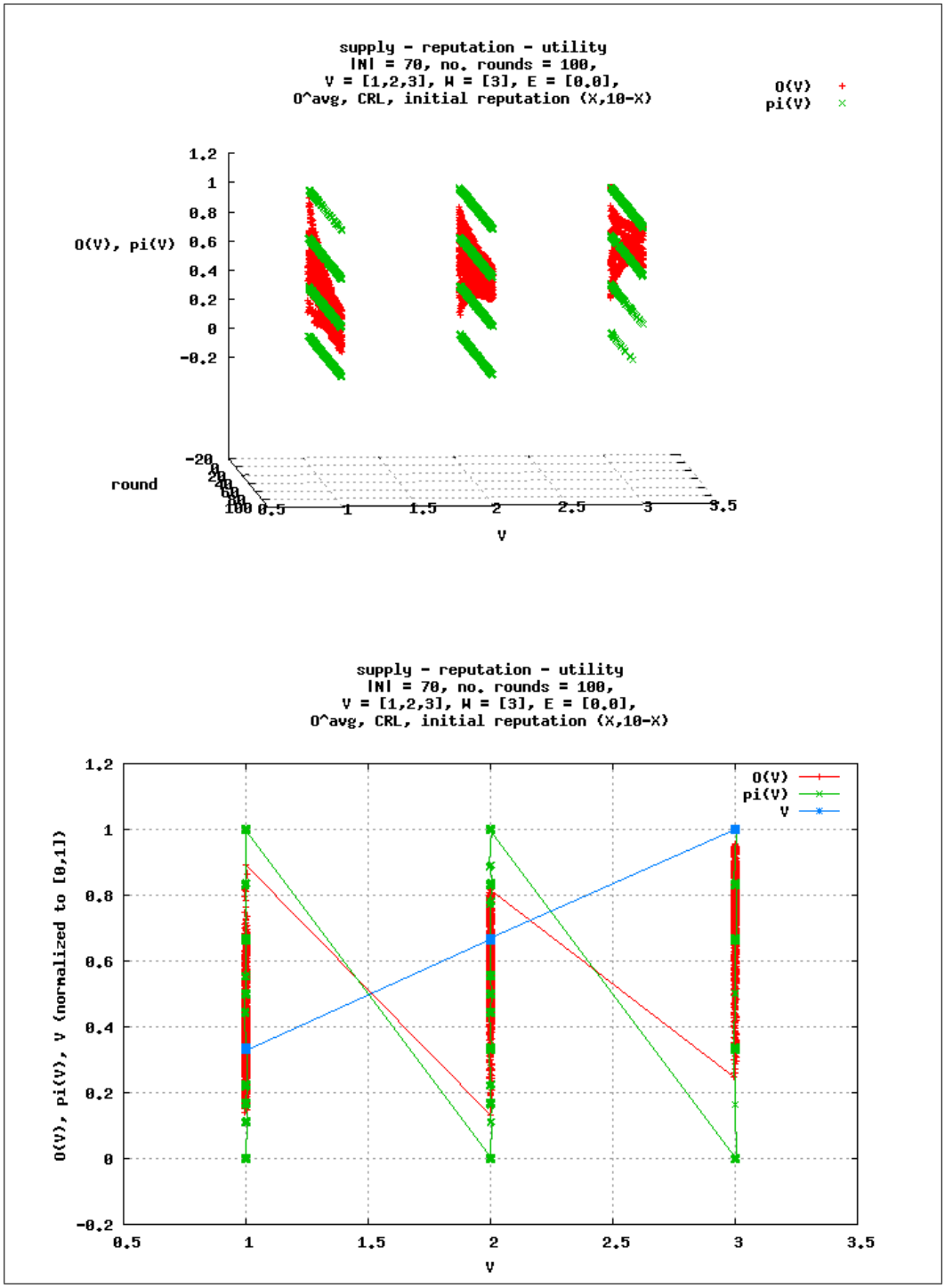


Figure 9.4: Ω^{CRL} with randomly pre-seeded reputation matrix.

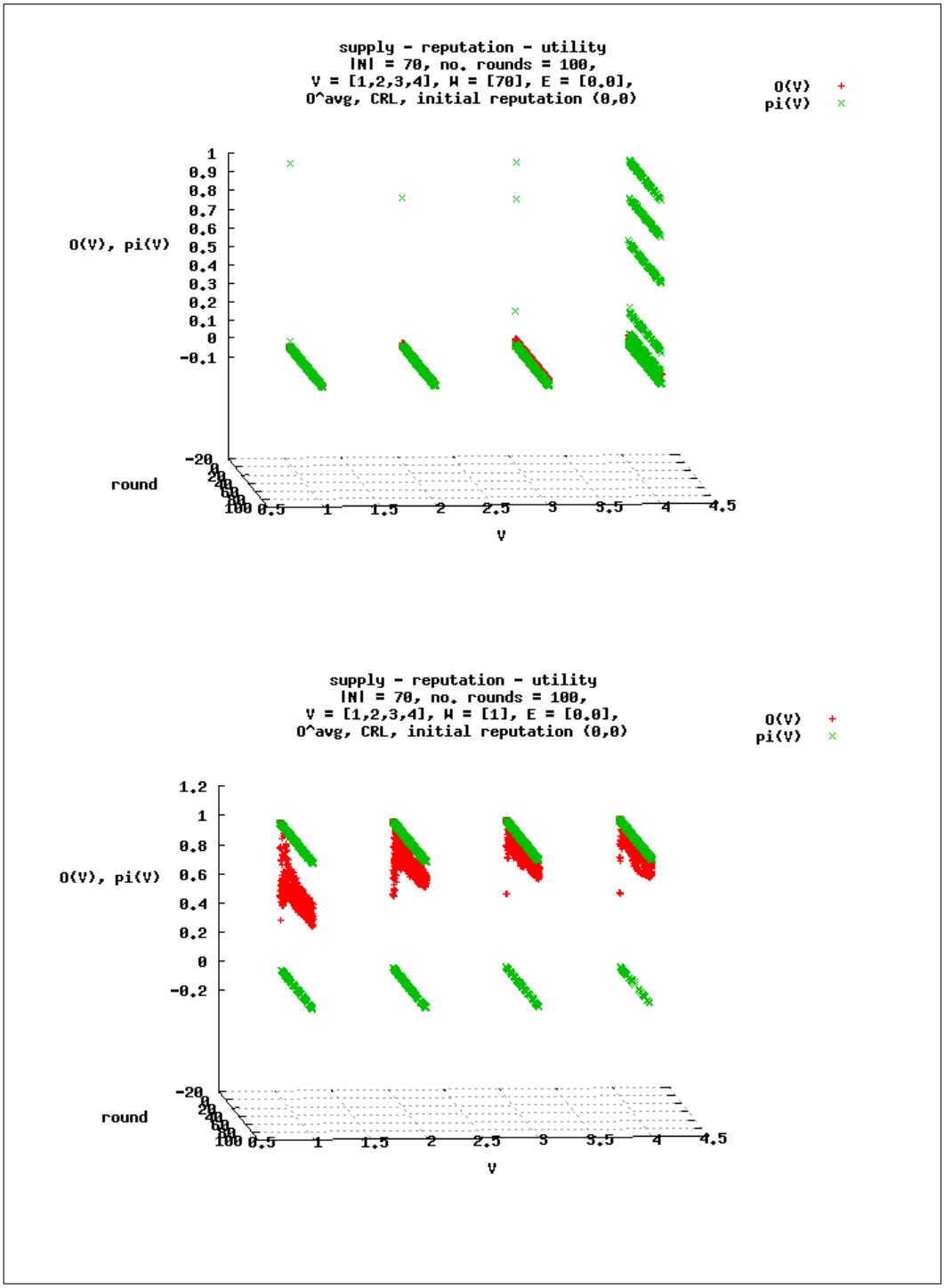


Figure 9.5: Ω^{CRL} with scarce and abundant resources.

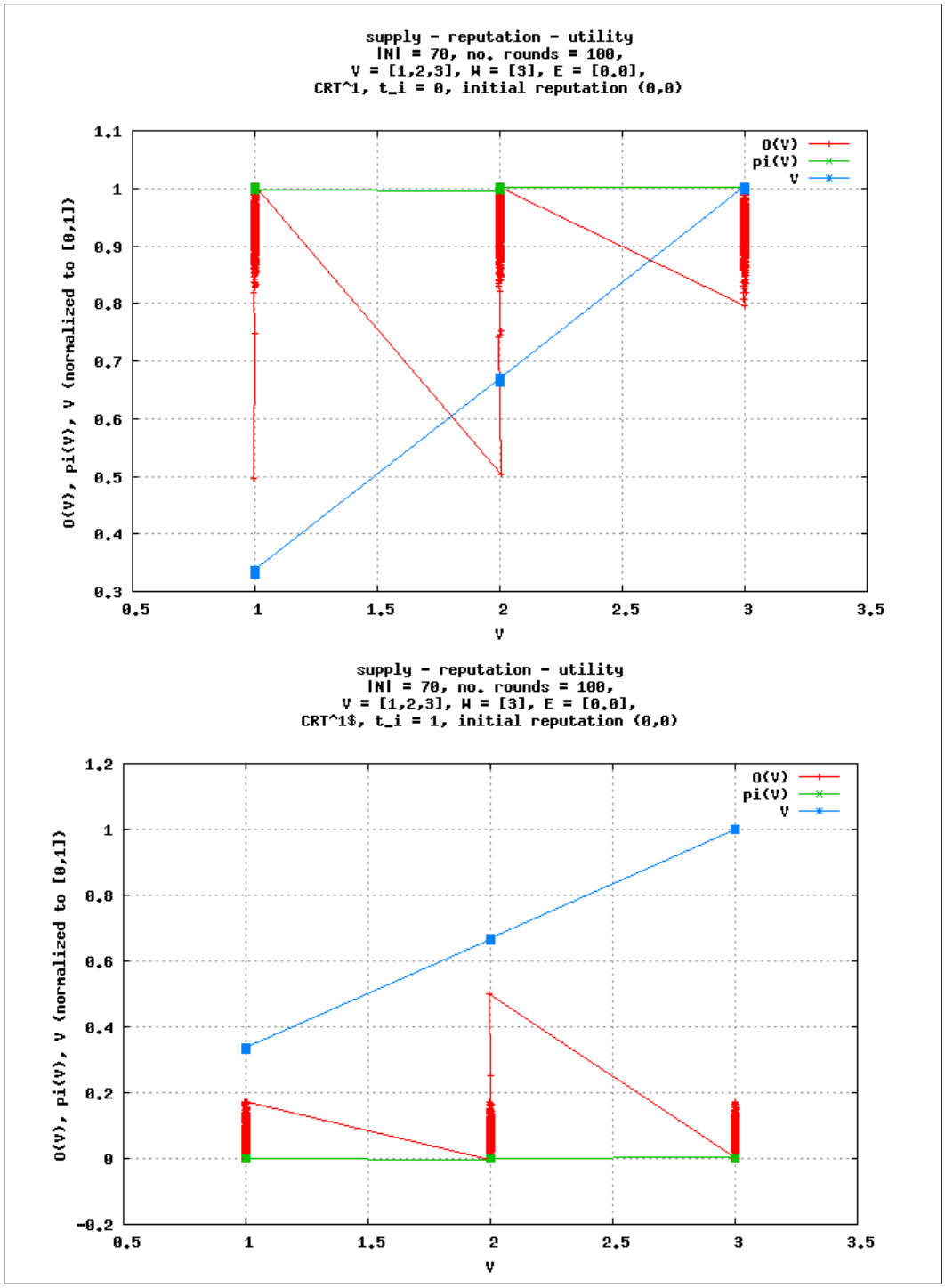


Figure 9.6: Ω^{CRT} with globally constant reputation threshold.

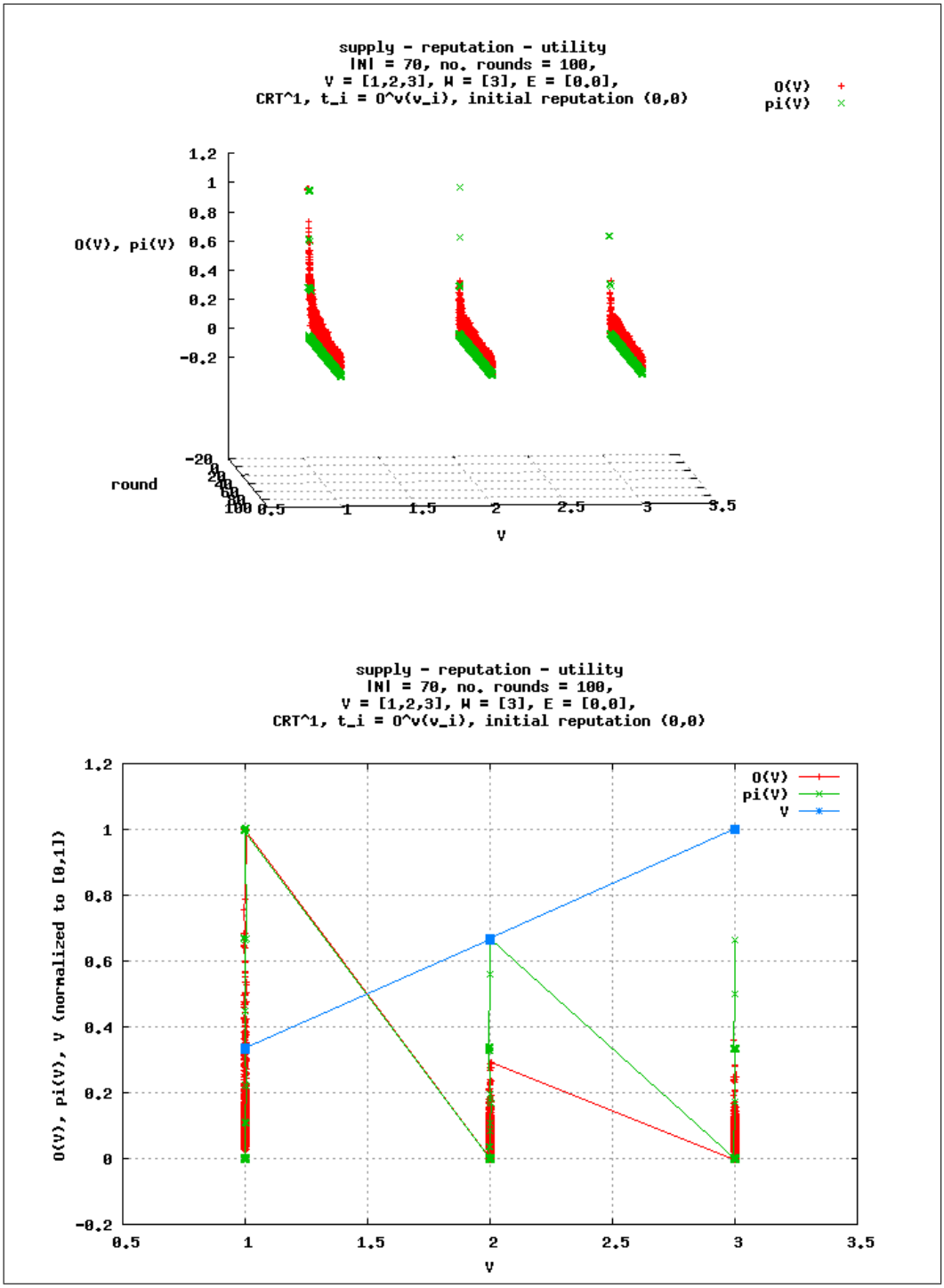


Figure 9.7: Ω^{CRT} with threshold $t_i = v_i$.

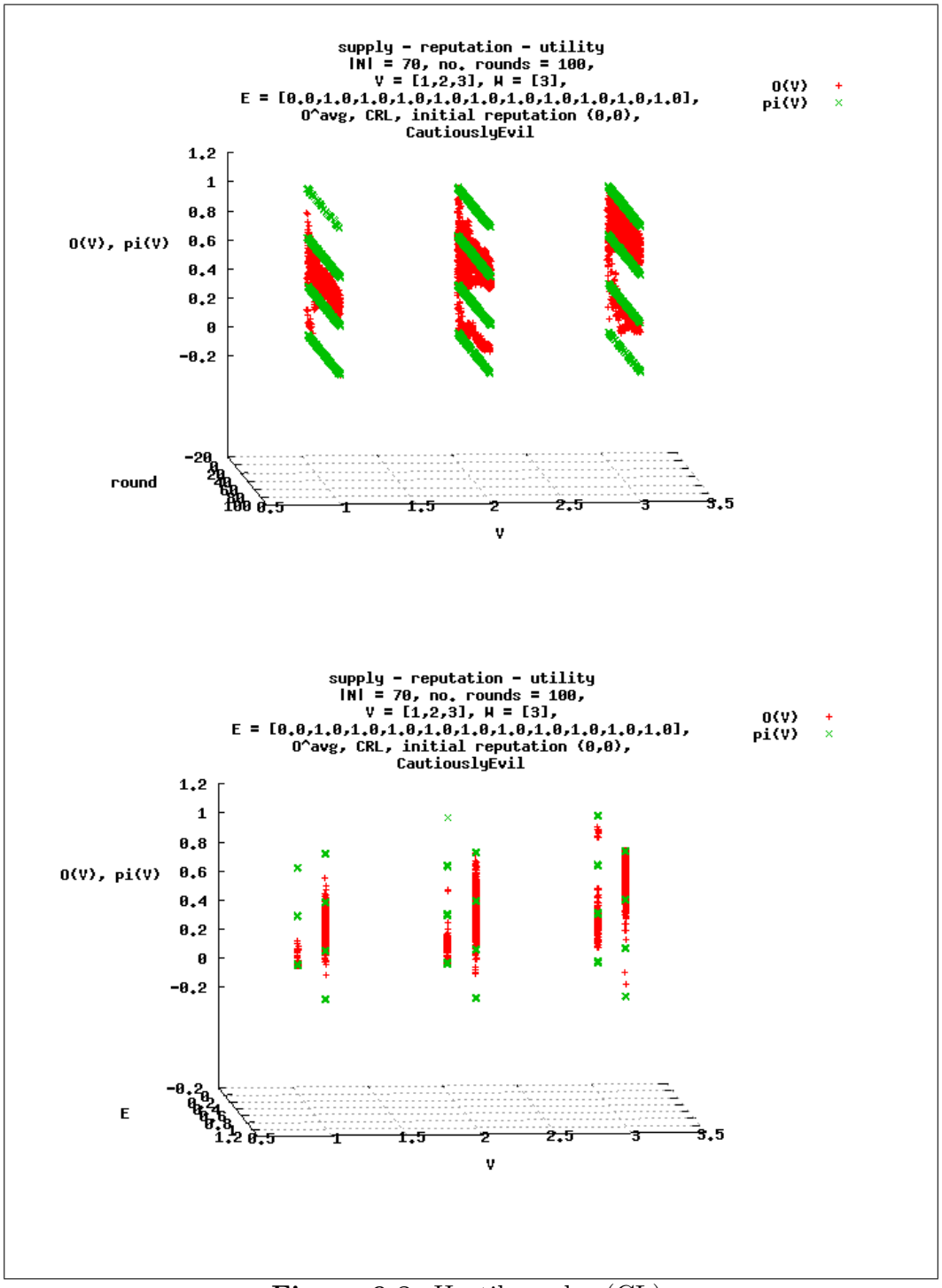


Figure 9.8: Hostile nodes (CL).

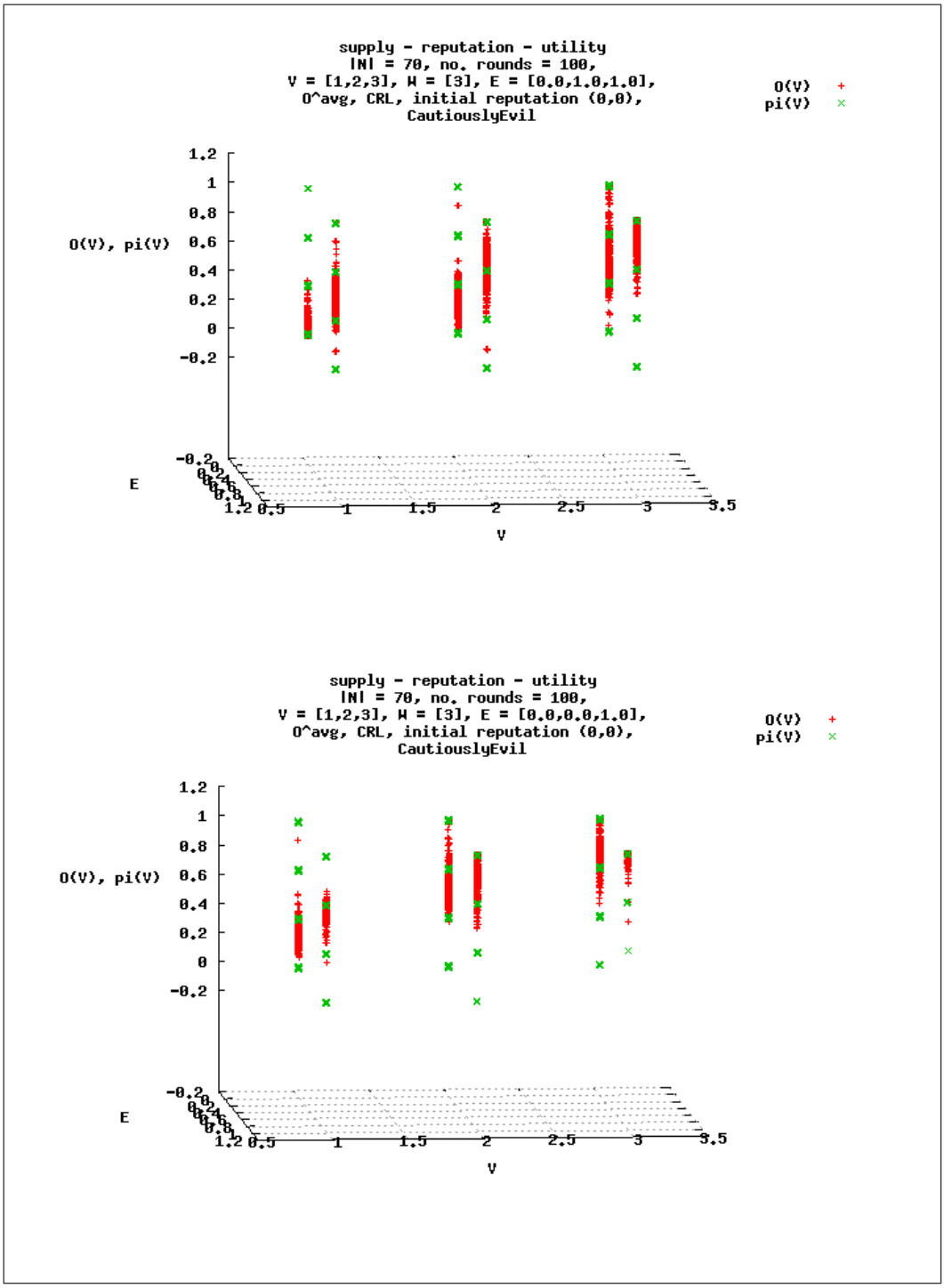


Figure 9.9: Fewer hostile nodes (CL).

Chapter 10

Conclusions

P2P applications, despite all their appeal and advantages, are inherently unhealthy environments. Anonymity is often a requirement, or a least a desirable side-effect, and not only has each node its own agenda potentially conflicting with that of other nodes, but worse, some of these agendas are purely destructive and targeted at causing damage to the application and its users.

To reduce the damage that can be caused to nodes when they are treated badly by others, they need to have accurate information on other nodes' identities, previous behavior, and agenda. This information is called reputation and trust information. It needs to be shared, since for every single node gathering sufficient experience is prohibitively expensive, and one needs to make sure that the shared information is correct in the presence of all those untrusted nodes that should be exposed in this information. For sharing it, a reputation scheme is needed.

Reputation schemes make use of robust, efficient, distributed data structures and algorithms and advanced cryptographic primitives to reduce noise. We have established a state-of-the-art common ground on which reputation schemes can start. We have then developed a model that has security characteristics unusually strong for current P2P applications, but that can be justified on this common ground. Finally, we have presented an implementation of a simple reputation scheme in this model and explore its characteritics.

In Chapter 7, we have formally introduced the concept of trust and reputation and motivated the relevance of this topic not only to P2P networks but to a much wider class of social systems. We have given a classification of P2P applications and one of adversarial node types, have explained the orthogonal and complementary nature of monetary incentives and reputation, and introduced the relevant game theoretic concepts to assess the impact and relevance of our work.

In Chapter 8, we have established a list of building blocks for constructing secure P2P networks, recycling and extending the cryptography from Chapter 2.3. Since most work on reputation schemes is based on existing P2P applications, which are in turn usually based on ad hoc network designs with inadequate cryptographic protection, this list should be of high value for future research.

We have started with an explanation of the identity problem, a classification of identity attacks, and a treatment of the three most important countermeasures. SybilGuard is a recent graph theoretical algorithm that finds clouds of adversarial robot nodes by the fact that they (claim to) have high traffic volumes among themselves, but the network as a whole has low traffic volumes into the cloud. Less recent and more controverial, hash cash and other proofs of work impose a job on a node that it can carry out to boost its reputation without the need of other nodes to risk interacting with it. Finally, there is a cryptographic signature scheme that makes identities irreplacable. Signatures are most reliable, but also hard to decentralize. We have developed a number of formerly unpublished variants that increase the domain of applicability and harden the original scheme against some risks such as theft of pseudonymous identities.

More important to our task is the construction of deletion-proof distributed data structures. It is easy to hand over a large document to a storage server and keep the hash, and if the storage server is unable to return the document, or returns an altered version of it, this can always be detected (with high probability) by comparing the hash of the retrieved document with the stored one. If the documents are as small as hashes, but there is a large number of them, it is possible that the server may get away unnoticed with dropping some of the documents. In storing a large number of transaction reports, we always face this problem when adversarial nodes are trusted with storing reputation on themselves or on colluding nodes. Hash trees are a class of data structures that help protect against this sort of attack, and we have listed a less well-known variant that is particularly powerful and suitable for the task at hand.

We have concluded Chapter 8 with the description of three example reputation schemes: CAN feedback, BitTorrent, and EigenTrust. In Chapter 9, we have developed a new and more generic application model and a corresponding reputation scheme, and used our generic P2P reputation simulator esim to implement it and understand its characteristics.

We have designed our model with the purpose to help grasping the core idea of a wide variety of different papers on the subject in a concise way. It has a number of unusual aspects that make it not both challenging and potentially instructive to do this.

1. **H**ighly generic. We do not make any assumptions on the underlying data structure, and thus our model applies to most of the data structures for which manually crafted incentive systems have been proposed, and hopefully also to future ones.

2. **U**tilizing cryptographic state of the art. Unlike in most current P2P applications, we use advanced building blocks from cryptography and other fields of computer science to motivate the modelled behavior of the nodes. For instance, our nodes have a persistent identity that cannot be changed even by adversaries (see Section 8.1). Without this restriction, it is very hard to say anything about system performance in the presence of adversaries, because a single player can impersonate arbitrarily many nodes. These cryptographic techniques are chosed carefully according to the requirement that the network be decentralized.

3. **B**eyond monetary incentives. Unlike in most of the economics literature, we are not restricting ourselves to monetary incentives. Neither the cost of responding to requests nor the benefit of charging for delivered resources are considered in our model. Nodes are invariant to whether they can provide a resource or not. Micropayments and monetary infrastructures in general have many drawbacks that may render their deployment unattractive for many P2P applications (see Section 7.4). Hence, we feel it is important to focus on reputation as an alternative class of incentives for cooperation. On the other hand, extending our model to monetary or hybrid incentive schemes is quite straightforward. Any insights gained on reputation systems should be valuable for those schemes as well.

4. **B**eyond equilibria and perfect solutions. Most previous work we are aware of has taken the approach of making inconvenient assumptions on both the distributed data structures the network is based on and the incentives involved, and then attempting to find cooperative equilibria (either in simulations or with rigorous game theoretical proofs). We have chosen this rigorous approach in Part I of our thesis to address a different problem, but in this part we have argued that a different approach is needed. We have built *what-if machines* that allowed us to describe a concrete system and a concrete adversary, and then assess the impact of the adversary on the performance of the system. Using our model we can decide to live with adversaries if they are unavoidable, and still find optimal parameters for a given application. The history of

P2P applications gives us reason to hope that this will make a situation that is already productive for a large user bases more so.

5. **B**eyond linearity. In economics, one can find linear models for many fundamental truths. These have the advantage that their behavior over time is easy to predict and calculate, but this is also their weakness: Complex dynamic systems behave in ways that make it impossible for easy-to-handle models to adequately describe their behavior. We have abandoned the ambition to provide analytic proofs for our findings. We claim that a model should be simple enough to be understood by simulation, which is a far higher bound than analysis.

To conclude Part II of this thesis, we have demonstrated how esim can be used to visualize and quantify effects of reputation systems on P2P networks, but we also have established a number of properties of our model (and thus, hopefully, on a number of real-life P2P networks as well). For a start, it has become clear that CRL yields more accurate reputation than CRT: If every node only gives to those nodes that it deems honest and highly cooperative, fewer resources are contributed. On the other hand, CRL, while performing better in terms of contribution rates, is not as targeted in punishing defectors with resource deprivation, and thus induces weaker incentives to contribute.

This dilemma demonstrates that a choice between different distribution strategies which does not matter for a single node may make a big difference for the overall system performance. In our setting, CRL may be better if we expect too many free riders, because even lying only helps to an extent and even selfish players have an incentive to supply values, and CRT (or, more likely, a combination of CRL and CRT) may be better if we expect a high fraction of honest players, because then even small changes in reputation values can be taken seriously by the nodes reading them.

More liars make $\rho(V, \Omega(V))$ blur and $\rho(E, \Omega(E))$ sharpen. This is unfortunate, but had to be expected, and our intension was not to weed out all adversarial behavior, but rather to determine its impact.

On the bright side, our experiments have shown that a smaller number of liars does not only have a smaller negative impact on system performance, but also makes lying less effective for the individual node. If one accepts the evolutionary interpretation of the prisoner's dilemma that systems may start off with good individuals, but then deteriorate if defectors are given an advantage, then this is an important result. It sets the threshold that nodes need to pass in our model to turn into defectors high in a young system, and this in turn should make it easy to keep a system young.

We have also examined sparse graphs in the hope of finding an optimal

density level that would help design and calibrate backbone network structures. The simulations show that our reputation scheme suffers in network areas with few connections. This suggests that medium and large networks may benefit from assuming super-node structures in which arbitrary nodes act as hubs, and thus the network is composed of many small networks that are better connected, instead of a large, sparse one.

There is one problem that has become appearent during our research that we have not explicitly mentioned yet. In our model (at least if CRL plays a role in the distribution strategy), nodes can get bad reputation for two reasons: (a) from malicious or lazy defection, and (b) from resource exhaustion or network problems. This has distorted our results considerably and should be the next thing to work on. Is there a way to distinguish deliberate defection from resource exhaustion? Which techniques for circumvention of hot spots are most suitable for our system? If this will reduce the noise levels in the simulation output as expected, will new trends be visible that we have missed here?

Bibliography

[AC96] Abadi, Martín; Cardelli, Luca: *A Theory of Objects*. Monographs in Computer Science. Springer, 1996. ISBN: 0-387-94775-2.

[ACF+77] Adler, Roy; Coppersmith, Don; Feistel, Hort; Grossman, Edna; Konheim, Alan; Matyas, Mike; Meyer, Carl; Notz, Bill; Smith, Lynn; Tuchman, Walter; Tuckerman, Bryant: Data Encryption Standard. FIPS-46, National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.

[ADS03] Acquisti, Alessandro; Dingledine, Roger; Syverson, Paul F.: On the Economics of Anonymity. In: *Financial Cryptography, 7th International Conference, FC 2003, Guadeloupe*, pp. 84–102. 2003.

[AF96] Abe, Masayuki; Fujisaki, Eiichiro: How to Date Blind Signatures. In: *Advances in Cryptology ASIACRYPT*, pp. 244–251. 1996.

[AF02] Asonov, Dimitri; Freytag, Johann Christoph: Almost Optimal Privacy Information Retrieval. In 2nd Workshop on Privacy Enhancing Technologies (PET2002), 2002. URL `http://citeseer.ist.psu.edu/636156.html`.

[AH00] Adar, Eytan; Huberman, Bernardo A.: Free Riding on Gnutella. First Monday, October 2000, `http://www.firstmonday.dk/issues/issue5_10/adar/index.html`, 2000.

[AK97] Anderson, Ross; Kuhn, Markus: Low Cost Attacks on Tamper Resistant Devices. In: *Security Protocol Workshop*, April 1997. `http://www.cl.cam.ac.uk/ftp/users/rja14/tamper2.ps.gz`.

[ALN87] Ahituv, Niv; Lapid, Yeheskel; Neumann, Seev: Processing Encrypted Data. In: *CACM*, volume 30(9):pp. 777–780, 1987. URL http://www.informatik.uni-trier.de/~ley/db/journals/cacm/cacm30.html#AhituvLN87.

[AS85] Abelson, Harold; Sussman, Gerald Jay: *Structure and Interpretation of Computer Programs*. MIT Press, 1985. ISBN: 0-262-51087-1.

[AS00] Agrawal, Rakesh; Srikant, Ramakrishnan: Privacy-Preserving Data Mining. In: *Proc. of the ACM SIGMOD Conference on Management of Data*, pp. 439–450. ACM Press, May 2000. ISBN 1-581-13218-2. URL http://www.almaden.ibm.com/cs/people/srikant/papers/sigmod00.pdf.

[Aso03] Asonov, Dmitri: *Querying Databases Privately*. Ph.D. thesis, Humboldt University Berlin, 2003.

[Avi03] Aviram, Amitai: The Paradox of Spontaneous Formation of Private Legal Systems. Technical report, Univeristy of Chicago law school, July 2003.

[Axe84] Axelrod, Robert: *The Evolution of Cooperation*. New York: Basic Books, 1984. ISBN 0-465-02121-2.

[Bac02] Back, Adam: Hash Cash — Amortizable Pubicly Auditable Cost Functions. http://www.cypherspace.org/hashcash/, August 2002.

[Bar84] Barendregt, Henk: *The Lambda Calculus, its Syntax and Semantics*. North-Holland, 1984. ISBN: 0-444-86748-1.

[BAS03] Buragohain, Chiranjeeb; Agrawal, Divyakant; Suri, Subhash: A Game Theoretic Framework for Incentives in P2P Systems. Preprint on http://arxiv.org/abs/cs.GT/0310039, 2003.

[Bau04] Bauer, Matthias: Proofs of Zero Knowledge. peer-reviewed preprint #cs.CR/0406058, arXiv.org, 2004. URL http://arxiv.org/abs/cs.CR/0406058.

[BB03] Buchmann, Erik; Böhm, Klemens: Reputation-Sensitive Message Passing in Content-Addressable Networks. Preprint 12, Otto-von-Guericke University Magdeburg, 2003. URL http://wwwiti.cs.uni-magdeburg.de/iti_dke/forschung/buchmann03reputation.pdf.

[BBS82] Blum, Lenore; Blum, Manuel; Shub, Michael: Comparison of two pseudo-random number generators. Advances in Cryptology: Proceedings of Crypto '82, 1982.

[BBS86] Blum, Lenore; Blum, Manuel; Shub, Michael: A Simple Unpredictable Pseudo-Random Number Generator. SIAM Journal on Computing, Vol. 15, May 1986.

[BBvdW06] Buchmann, Erik; Böhm, Klemens; von der Weth, Christian: Towards truthful feedback in p2p data structures. In: *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, OTM Confederated International Conferences, Part I*, pp. 498–515. 2006.

[BCOP04] Boneh, Dan; Crescenzo, Giovanni Di; Ostrovsky, Rafail; Persiano, Giuseppe: Public Key Encryption with Keyword Search. In: *in Proceedings of Eurocrypt 2004, LNCS 3027*, pp. 506–522. 2004.

[BDK+01] Badger, Lee; D'Anna, Larry; Kilpatrick, Doug; Matt, Brian; Reisse, Andrew; Vleck, Tom Van: Self-Protecting Mobile Agents Obfuscation. Technical Report #01-036, NAI Labs, November 2001. URL `http://opensource.nailabs.com/jbet/papers/obfeval.pdf`.

[Ben05] Benantar, Messaoud: *Access Control Systems: Security, Identity Management and Trust Models, 1st edition*. Springer, December 9 2005. ISBN 0387004459.

[Ber92] Berson, Thomas A.: Differential Cryptanalysis Mod $2^3 2$ with Applications to MD5. In: *Proceedings of Eurocrypt 2001*, volume 658 of *LNCS*, pp. 71–80. 1992.

[BF03] Boyens, Claus; Fischmann, Matthias: Profiting from Untrusted Parties in Web-based Applications. In: *Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies (EC-Web'03)*. LNCS, Springer Verlag, 2003.

[BFH03] Berman, Fran; Fox, Geoffrey; Hey, Anthony J.G., editors: *Grid Computing: Making The Global Infrastructure a Reality*. Wiley, 2003. ISBN: 0470853190.

[BG02] Boyens, Claus; Günther, Oliver: Trust is not Enough: Privacy and Security in ASP and Web Service Environments. In:

ADBIS '02: Proceedings of the 6th East European Conference on Advances in Databases and Information Systems, pp. 8–22. Springer-Verlag, London, UK, 2002.

[BGI+01] Barak, Boaz; Goldreich, Oded; Impagliazzo, Rusell; Rudich, Steven; Sahai, Amit; Vadhan, Salil; Yang, Ke: On the (Im)possibility of Obfuscating Programs. In: *Lecture Notes in Computer Science*, volume 2139, 2001. URL `http://citeseer.nj.nec.com/barak01impossibility.html,http://www.math.ias.edu/~boaz/Papers/obf_informal.html`.

[BK05] Bazzi, Rida A.; Konjevod, Goran: On the Establishment of Distinct Identities in Overlay Networks. In: *PODC*, pp. 312–320. 2005.

[BKN02] Bellare, Mihir; Kohno, Tadayoshi; Namprempre, Chanathip: Authenticated Encryption in SSH: Provably Fixing the SSH Binary Packet Protocol, 2002.

[BKO02] Bolton, Gary E.; Katok, Elena; Ockenfels, Axel: Bridging the Trust Gap in Electronic Markets. Discussion Paper, 2002.

[Ble98] Bleichenbacher, Daniel: Chosen Ciphertext Attacks against Protocols Based on RSA Encryption Standard PKCS #1. In: *Advances in Cyptology – CRYPTO' 98*, pp. 1–12. Springer-Verlag, Berlin, 1998. URL `http://www.bell-labs.com/user/bleichen/bib.html`.

[Blo70] Bloom, Burton H.: Space/Time Trade-Offs in Hash Coding with Allowable Errors. Communications of the ACM, Vol. 13(7), pp. 422-426, 1970.

[Buc06] Buchmann, Erik: *Erkennung und Vermeidung von unkooperativem Verhalten in Peer-to-Peer-Datenstrukturen.* Ph.D. thesis, Otto-von-Guericke University Magdeburg, Juny 2006.

[BY87] Brickell, Ernest F.; Yacobi, Yacov: On Privacy Homomorphisms (Extended Abstract). In: *EUROCRYPT*, pp. 117–125. 1987.

[CA89] Chaum, David; Antwerpen, Hans Van: Undeniable Signatures. In: *Advances in Cryptology CRYPTO'89, LNCS 435*, pp. 212–216. 1989.

[CB74] Chamberlin, Donald D.; Boyce, Raymond F.: SEQUEL: A structured English query language, 1974.

[CF05] Cheng, Alice; Friedman, Eric: Sybilproof Reputation Mechanisms. In: *P2PECON '05: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pp. 128–132. ACM Press, New York, NY, USA, 2005. ISBN 1-59593-026-4. doi:http://doi.acm.org/10.1145/1080192.1080202.

[CGKO06] Curtmola, Reza; Garay, Juan; Kamara, Seny; Ostrovsky, Rafail: Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In: *CCS'06: Proceedings of the 13th ACM conference on Computer and communications security*, pp. 79–88. ACM Press, New York, NY, USA, 2006. ISBN 1-59593-518-5. doi:http://doi.acm.org/10.1145/1180405.1180417.

[CGKS95] Chor, Benny; Goldreich, Oded; Kushilevitz, Eyal; Sudan, Madhu: Private Information Retrieval. In: *IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 41–50. 1995. URL `http://citeseer.ist.psu.edu/article/chor95private.html`.

[Cha81] Chaum, David: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. In: *Communications of the ACM*, volume 24(2):pp. 84–88, 1981.

[Cha82] Chaum, David: Blind Signatures for Untraceable Payments. In: *Advances in Cryptology CRYPTO*, pp. 199–203. 1982.

[Cha85] Chaum, David: Security Without Identification: Transaction Systems to Make Big Brother Obsolete. In: *Communications of the ACM*, volume 28(10):pp. 1030–1044, 1985.

[Cha91] Chaum, David: Numbers Can Be a Better Form of Cash than Paper. In: *Computer Security and Industrial Cryptography - State of the Art and Evolution*, pp. 174–178. 1991.

[Cha92] Chaum, David: Achieving Electronic Privacy. In: *Scientific American*, August 1992. An online version is available from `http://www.chaum.com/articles/Achieving_Electronic_Privacy.htm`.

[CKN06] Cheon, Jung Hee; Kim, Woo-Hwan; Nam, Hyun Soo: Known-plaintext cryptanalysis of the Domingo-Ferrer algebraic privacy

homomorphism scheme. In: *Inf. Process. Lett.*, volume 97(3):pp. 118–123, 2006.

[CLRS01] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford: *Introduction to Algorithms, Second Edition*. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7.

[CM05] Chang, Yan-Cheng; Mitzenmacher, Michael: Privacy Preserving Keyword Searches on Remote Encrypted Data. In: *International Conference on Applied Cryptography and Network Security (ACNS), LNCS*, volume 3. 2005.

[Cod70] Codd, Edgar F.: A relational model of data for large shared data banks. Communications of the ACM, Vol. 13(6), pp. 377–387, 1970.

[Coh03] Cohen, Bram: Incentives Build Robustness in BitTorrent. published on `http://www.bittorrent.org/`, May 2003.

[CRR02] Chari, Suresh; Rabin, Tal; Rivest, Ronald: An Efficient Signature Scheme for Route Aggregation. Draft, 2002. URL `{http://citeseer.ist.psu.edu/chari02efficient.html},{http://theory.lcs.mit.edu/~rivest/publications.html}`.

[CS98] Cramer, Ronald; Shoup, Victor: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. Advances in Cryptology — CRYPTO'98: 18th Annual International Cryptology Conference, 1998. URL `http://www.springerlink.com/app/home/contribution.asp?wasp=92udc9wgxg3unplxgndm&referrer=parent&backto=issue,2,35;journal,1100,1213;linkingpublicationresults,1,1`.

[CT02] Collberg, Christian S.; Thomborson, Clark D.: Watermarking, Tamper-Proofing, and Obfuscation — Tools for Software Protection. In: *IEEE Trans. Software Eng*, volume 28(8):pp. 735–746, 2002.

[CTL98a] Collberg, Christian S.; Thomborson, Clark D.; Low, Douglas: Breaking Abstractions and Unstructuring Data Structures. In: *ICCL*, pp. 28–38. 1998. URL `http://computer.org/proceedings/iccl/8454/84540028abs.htm`.

[CTL98b] Collberg, Christian S.; Thomborson, Clark D.; Low, Douglas: Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs. In: *POPL*, pp. 184–196. 1998. URL `http://doi.acm.org/10.1145/268946.268962`.

[CZL07] Chen, Xiaofeng; Zhang, Fangguo; Liu, Shengli: ID-based Restrictive Partially Blind Signatures and Applications. In: *Journal of Systems and Software*, volume 80(2):pp. 164–171, 2007. ISSN 0164-1212. doi:http://dx.doi.org/10.1016/j.jss.2006.02.046.

[DA99] Dierks, Tim; Allen, Christopher: The TLS Protocol, Version 1.0. RFC 2246, IETF, Network Working Group, `http://www.ietf.org/rfc/rfc2246.txt`, January 1999.

[DA00] Du, Wenliang; Atallah, Mikhail J.: Protocols for Secure Remote Database Access with Approximate Matching. In: *Proc. of the First Workshop on Security and Privacy in E-Commerce*. November 2000. URL `http://citeseer.ist.psu.edu/du00protocols.html/`.

[DdVJ+03] Damiani, Ernesto; di Vimercati, S. De Capitani; Jajodia, Sushil; Paraboschi, Stefano; Samarati, Pierangela: Balancing confidentiality and efficiency in untrusted relational DBMSs. In: *CCS'03: Proceedings of the 10th ACM conference on Computer and communications security*, pp. 93–102. ACM Press, New York, NY, USA, 2003. ISBN 1-58113-738-9. doi:http://doi.acm.org/10.1145/948109.948124.

[DF96] Domingo-Ferrer, Josep: A new privacy homomorphism and applications. In: *Information Processing Letters*, volume 60(5):pp. 277–282, 1996. URL `http://citeseer.ist.psu.edu/ferrer96new.html`.

[DF02] Domingo-Ferrer, Josep: A provably secure additive and multiplicative privacy homomorphism. In: *Proceedings of the 5th Information Security Conference*, pp. 471–483. 2002.

[DFHJ98] Domingo-Ferrer, Josep; Herrera-Joancomartí, Jordi: A Privacy Homomorphism Allowing Field Operations on Encrypted Data, 1998. URL `http://citeseer.nj.nec.com/domingo-ferrer98privacy.html`.

[DLP+01] Dyer, Joan G.; Lindemann, Mark; Perez, Ronald; Sailer, Reiner; van Doorn, Leendert; Smith, Sean W.; Weingart, Steve: Building the IBM 4758 Secure Coprocessor. In: *Computer*, volume 34(10):pp. 57–66, 2001. ISSN 0018-9162. doi:http://dx.doi.org/10.1109/2.955100.

[DMR+03] D'Anna, Larry; Matt, Brian; Reisse, Andrew; Vleck, Tom Van; Schwab, Steve; LeBlanc, Patrick: Self-Protecting Mobile Agents Obfuscation Report. Technical Report #03-015, NAI Labs, June 2003. URL `http://opensource.nailabs.com/jbet/papers/obfreport.pdf`.

[DMS04] Dingledine, Roger; Mathewson, Nick; Syverson, Paul F.: Tor: The Second-Generation Onion Router. In: *USENIX Security Symposium*, pp. 303–320. 2004.

[Dob96] Dobbertin, Hans: The Status of MD5 After a Recent Attack. CryptoBytes Vol. 2 No. 2, RSA Laboratories, 1996.

[Dou02] Douceur, John R.: The Sybil Attack. In: *IPTPS*, pp. 251–260. 2002.

[DR02] Daemen, Joan; Rijmen, Vincent: *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002. ISBN 3-540-42580-2.

[EAH+06] Esponda, Fernando; Ackley, Elena S.; Helman, Paul; Jia, Haixia; Forrest, Stephanie: Protecting Data Privacy through Hard-to-Reverse Negative Databases. In the Proceedings of the 9th Information Security Conference, September 2006, `http://crypto.stanford.edu/portia/papers/HardNDB.pdf`, 2006.

[EFG06] Evdokimov, Sergei; Fischmann, Matthias; Günther, Oliver: Provable Security for Outsourcing Database Operations. In: *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*. IEEE Press, 2006.

[EFG07] Evdokimov, Sergei; Fischmann, Matthias; Günther, Oliver: Provable Security for Outsourcing Database Operations. In: *(to be submitted)*, 2007.

[EFH04] Esponda, Fernando; Forrest, Stephanie; Helman, Paul: Enhancing privacy through negative representations of data. Technical report, University of New Mexico, 2004.

[EGS03] Evfimievski, Alexandre; Gehrke, Johannes; Srikant, Ramakrishnan: Limiting Privacy Breaches in Privacy Preserving Data Mining. In: *PODS '03: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 211–222. ACM Press, New York, NY, USA, 2003. ISBN 1-58113-670-6. doi:http://doi.acm.org/10.1145/773153.773174.

[EH96] English, Erin; Hamilton, Scott: Network security under siege: The timing attack. In: *Computer*, volume 29(3):pp. 95–97, 1996. ISSN 0018-9162. doi:http://doi.ieeecomputersociety.org/10.1109/2.485898.

[ES06] Ewert, Ulf Christian; Selzer, Stephan: Bridging the Gap: The Hanseatic Merchants' Variable Strategies in Heterogeneous Mercantile Environments. In: *XIVth International Economic History Congress, Helsinki, Session 110: Tools of Trade in Late Medieval Commercial Cities*. August 2006.

[ESM03] Endorf, Carl; Schultz, Gene; Mellander, Jim: *Intrusion Detection and Prevention*. McGraw-Hill Osborne Media, December 2003. ISBN 0072229543.

[Esp05] Esponda, Fernando: *Negative Representations of Information*. Ph.D. thesis, University of New Mexico, 2005.

[FG03] Fischmann, Matthias; Günther, Oliver: Privacy Tradeoffs in Database Service Architectures. In: *Proceedings of the 1st International Workshop on Business driven security (BIZSEC'03)*. 2003.

[FGS05] Fabian, Benjamin; Günther, Oliver; Spiekermann, Sarah: Security Analysis of the Object Name Service. In: *Proceedings of the 1st Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU 2005), in conj. with IEEE ICPS 2005, Santorini*, pp. 71–76. 2005.

[Fis06] Fischmann, Matthias: Modelling Reputation-Based Resource Pooling in P2P Systems. In: *Proceedings of the 1st International Conference on Scalable Information Systems (INFOSCALE'06)*. IEEE Press, 2006.

[FK98] Foster, Ian; Kesselman, Carl, editors: *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1998. ISBN: 1558604758.

[FLSC04] Feldman, Michal; Lai, Kevin; Stoica, Ion; Chuang, John: Robust Incentive Techniques for Peer-To-Peer Networks. In: *ACM Conference on Electronic Commerce*, pp. 102–111. 2004. URL `http://citeseer.ist.psu.edu/feldman04robust.html/`.

[FR01] Friedman, Eric J.; Resnick, Paul: The Social Cost of Cheap Pseudonyms. Journal of Economics and Management Strategy 10(2): 173-199., 2001. URL `http://www.si.umich.edu/~presnick/papers/identifiers/`.

[FZ00] Fletcher, Jeffrey A.; Zwick, Martin: Simpson's Paradox Can Emerge from the N-Player Prisoner's Dilemma. In: *Proceedings of The WorldCongress of the Systems Sciences and ISSS 2000, Toronto, Canada: International Society for the Systems Sciences*. 2000. URL `http://www.sysc.pdx.edu/download/papers/isss_fl_zw.html`.

[Gam85] Gamal, Taher El: A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. In: *Proceedings of CRYPTO 84 on Advances in Cryptology*, pp. 10–18. Springer-Verlag New York, Inc., 1985. ISBN 0-387-15658-5.

[GF03] Greco, Gian Maria; Floridi, Luciano: The Tragedy of the Digital Commons. IEG Research Report 14.10.03, `http://web.comlab.ox.ac.uk/oucl/research/areas/ieg`, 2003.

[GJ79] Garey, Michael; Johnson, David S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of Books in the Mathematical Sciences. W. H. Freeman, 1979. ISBN: 978-0716710455.

[GMW94] Greif, Avner; Milgrom, Paul; Weingast, Barry R.: Coordination, Commitment, and Enforcement: The Case of the Merchant Guild. In: *Journal of Political Economy*, volume 102(4):pp. 745–776, August 1994. Available at `http://ideas.repec.org/a/ucp/jpolec/v102y1994i4p745-76.html`.

[Goh03] Goh, Eu-Jin: Building Secure Indexes for Searching Efficiently on Encrypted Compressed Data. Cryptology ePrint Archive:

Report 2003/216. http://eprint.iacr.org/2003/216/, 2003. URL citeseer.ist.psu.edu/goh03secure.html.

[Gol01] Goldreich, Oded: *Foundations of Cryptography – Volume I Basic Tools*. Cambridge University Press, 2001. ISBN: 0-521-79172-3, http://www.wisdom.weizmann.ac.il/~oded/foc-vol1.html.

[Gol04] Goldreich, Oded: *Foundations of Cryptography – Volume II Basic Applications*. Cambridge University Press, 2004. ISBN: 0-521-83084-2, http://www.wisdom.weizmann.ac.il/~oded/foc-vol2.html.

[Háj03] Hájek, Alan: Interpretations of Probability. Stanford Encyclopedia of Philosophy, http://plato.stanford.edu/entries/probability-interpret/, 2003.

[Hal01] Halevi, Shai: An Observation Regarding Jutla's Modes of Operation. IBM Watson technical report, 2001.

[Har68] Hardin, Garrett: The Tragedy of the Commons, 1968.

[HILM02] Hacıgümüş, Hakan; Iyer, Bala; Li, Chen; Mehrotra, Sharad: Executing SQL over Encrypted Data in the Database-Service-Provider Model. In: *Proceedings of the 28th SIGMOD Conference on the Management of Data*. ACM, 2002.

[HRL97] Harry R. Lewis, Christos H. Papadimitriou: *Elements of the Theory of Computation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1997. ISBN: 0132624788.

[HU00] Hopcroft, John E.; Ullman, Jeffery D.: *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, November 2000. http://www-db.stanford.edu/~ullman/ialc.html.

[IEPN04] Ilie, Traffic Dragos; Erman, David; Popescu, Adrian; Nilsson, Arne A.: Measurement and Analysis of Gnutella Signaling. IPSI, 2004.

[JF05] Jurca, Radu; Faltings, Boi: Enforcing Truthful Strategies in Incentive Compatible Reputation Mechanisms. In: *Internet and Network Economics*, volume 3828 of *Lecture Notes in Computer Science*, pp. 268 – 277. Springer Verlag, 2005.

[JMS05] Jia, Haixia; Moore, Cristopher; Strain, Doug: Generating Hard Satisfiable Formulas by Hiding Solutions Deceptively, 2005.

[JMSW02] Johnson, Robert; Molnar, David; Song, Dawn Xiaodong; Wagner, David: Homomorphic Signature Schemes. In: *CT-RSA*, pp. 244–262. 2002. URL `http://citeseer.ist.psu.edu/johnson02homomorphic.html`.

[Jut00] Jutla, Charanjit S.: Encryption Modes with Almost Free Message Integrity. in Proceedings of Eurocrypt 2001, LNCS, Springer Verlag, 2000. Earlier version in Cryptology ePrint Archive, Report 2000/039, `http://eprint.iacr.org/`.

[KC04] Kantarcıoğlu, Murat; Clifton, Chris: Security Issues in Querying Encrypted Data. Purdue CS technical report, March 2004.

[KDZ06] Kurt D. Zeilenga, Ed.: Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map, June 2006. RFC 4510, IETF, Network Working Group, `http://www.ietf.org/rfc/rfc4510.txt`.

[KJJ99] Kocher, Paul C.; Jaffe, Joshua; Jun, Benjamin: Differential Power Analysis. In: *CRYPTO 1999, LNCS*, pp. 388–397. 1999.

[KM04] Koblitz, Neal; Menezes, Alfred J.: Another Look at "Provable Security", 2004.

[KO97] Kushilevitz, Eyal; Ostrovsky, Rafail: Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval. In: *Proceedings of Thirty-Eighth Annual IEEE Symposium on the Foundations of Computer Science (FOCS-97)*. 1997.

[Kob94] Koblitz, Neal: *A Course in Number Theory and Cryptography*. Springer Verlag, September 1994. ISBN 0387942939.

[Koc96] Kocher, Paul C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: *CRYPTO 1996, LNCS*, pp. 104–113. Springer Berlin / Heidelberg, 1996.

[Kol33] Kolmogorov, Andrei Nikolajevich: *Grundbegriffe der Wahrscheinlichkeitrechnung*. Springer, Berlin, 1933.

[Koz03] Koziol, Jack: *Intrusion Detection with Snort*. Sams, May 2003. ISBN 157870281X.

[KRT03] Kangasharju, Jussi; Ross, Keith W.; Turner, David A.: Secure and Resilient Peer-to-Peer E-Mail: Design and Implementation. In: *IEEE International Conference on Peer-to-Peer Computing*. 2003.

[KSGM03] Kamvar, Sepandar D.; Schlosser, Mario T.; Garcia-Molina, Hector: The EigenTrust Algorithm for Reputation Management in P2P Networks. In: *WWW'03: Proceedings of the 12th international conference on World Wide Web*, pp. 640–651. ACM Press, New York, NY, USA, 2003. ISBN 1-58113-680-3. doi:http://doi.acm.org/10.1145/775152.775242. `http://www.stanford.edu/~sdkamvar/papers/eigentrust.pdf`.

[KY01] Kiayias, Aggelos; Yung, Moti: Secure Games with Polynomial Expressions. In: *LNCS*, volume 2076, 2001. URL `http://citeseer.ist.psu.edu/kiayias01secure.html`.

[LAAA06] Lam, V. T.; Antonatos, Spyros; Akritidis, Periklis; Anagnostakis, Kostas G.: Puppetnets: Misusing Web Browsers as a Distributed Attack Infrastructure. In: *ACM Conference on Computer and Communications Security*, pp. 221–234. 2006.

[LAFH04] Lee, Hyungjick; Alves-Foss, Jim; Harrison, Scott: The Use of Encrypted Functions for Mobile Agent Security. Proceedings of the 37th Hawaii International Conference on System Sciences, 2004.

[LB05] Li, Maozhen; Baker, Mark: *The Grid: Core Technologies (Paperback)*. Wiley, 2005. ISBN: 0470094176, `http://coregridtechnologies.org/`.

[LC04] Laurie, Ben; Clayton, Richard: "Proof-of-Work" Proves Not to Work. `http://www.hashcash.org/papers/proof-work.pdf`, May 2004.

[LRSW00] Lysyanskaya, Anna; Rivest, Ronald L.; Sahai, Amit; Wolf, Stefan: Pseudonym Systems. In: *Selected Areas in Cryptography: 6th Annual International Workshop, SAC'99, Kingston, Ontario, Canada, August 1999.*, LNCS, pp. 184–199. Springer, 2000. URL `http://theory.lcs.mit.edu/~rivest/publications.html`.

[LST05] Löser, Alexander; Staab, Steffen; Tempich, Christoph: Semantic Methods for P2P Query Routing. In: *Multiagent System Technologies, Third German Conference, MATES 2005, Koblenz*, LNCS, pp. 15–26. Springer, 2005.

[Mac03] MacKay, David J. C.: *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. ISBN: 9780521642989, URL `http://www.inference.phy.cam.ac.uk/mackay/itila/`.

[MB02] Maitland, Greg; Boyd, Colin: A Provably Secure Restrictive Partially Blind Signature Scheme. In: *Public Key Cryptography*, pp. 99–114. 2002. URL `citeseer.ist.psu.edu/665591.html`.

[McL99] McLaughlin, Michael P.: A Compendium of Common Probability Distributions. Appendix A of the Regress+ User Manual. `http://www.causascientia.org/software/Regress_plus.html`, 1999.

[Mea04] Mealling, Michael: EPCglobal Object Naming Service (ONS) 1.0, 2004. `http://www.epcglobalinc.org/standards/Object_Naming_Service_ONS_Standard_Version_1.0.pdf`.

[Mer80] Merkle, Ralph C.: Protocols For Public Key Cryptosystems, April 1980.

[MHPD05] Mislove, Alan; Haeberlen, Andreas; Post, Ansley; Druschel, Peter: ePOST. In: *Peer-to-Peer Systems and Applications*, pp. 171–192. 2005.

[MHS+01] Minar, Nelson; Hedlund, Marc; Shirky, Clay; O'Reilly, Tim; Bricklin, Dan; Anderson, David; Miller, Jeremie; Langley, Adam; Kan, Gene; Brown, Alan; Waldman, Marc; Cranor, Lorrie Faith; Rubin, Aviel; Dingledine, Roger; Freedman, Michael; Molnar, David; Dornfest, Rael; Brickley, Dan; Hong, Theodore; Lethin, Richard; Udell, Jon; Asthagiri, Nimisha; Tuvell, Walter; Wiley, Brandon; (ed.), Andy Oram: *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, February 2001. `http://www.oreilly.com/catalog/peertopeer/`.

[Miq04] Miquel, Jordi Palau: Collaboration Analysis in Recommender Systems using Social Networks. To be presented to the Eighth International Workshop on Cooperative Information

Agents (CIA'04), 2004. http://eia.udg.es/~mmontane/palauaamas04.pdf.

[MM02] Maymounkov, Petar; Mazières, David: Kademlia: A Peer-to-peer Information System Based on the XOR Metric. In: *1st International Workshop on Peer-to-peer Systems*, 2002.

[Moc87a] Mockapetris, Paul: Domain Names — Concepts and Facilities, November 1987. RFC 1034, IETF, Network Working Group, http://www.ietf.org/rfc/rfc1034.txt.

[Moc87b] Mockapetris, Paul: Domain Names — Implementation and Specification, November 1987. RFC 1035, IETF, Network Working Group, http://www.ietf.org/rfc/rfc1035.txt.

[MR02] Micali, Silvio; Rivest, Ronald L.: Transitive Signature Schemes. In: *Proceedings of the Cryptographer's Track at the RSA Conference 2002*, pp. 236–243. Springer Verlag CT-RSA, LNCS 2271, 2002.

[MRK03] Micali, Silvio; Rabin, Michael O.; Kilian, Joe: Zero-Knowledge Sets. In: *44th Symposium on Foundations of Computer Science (FOCS 2003)*, pp. 80–91. IEEE Computer Society, 2003.

[MS83] Myerson, Roger B.; Satterthwaite, Mark A.: Efficient Mechanisms for Bilateral Trading. In: *Journal of Economic Theory*, volume 29(2):pp. 265–281, April 1983.

[MvOV01] Menezes, Alfred J.; van Oorschot, Paul C.; Vanstone, Scott A.: *Handbook of Applied Cryptography*. CRC Press, 2001. ISBN: 0-8493-8523-7, URL http://www.cacr.math.uwaterloo.ca/hac/about/order.html.

[NBW06] Newman, Mark; Barabási, Albert-László; Watts, Duncan J.: *The Structure and Dynamics of Networks*. Princeton University Press, 2006. First chapter available from http://press.princeton.edu/titles/8114.html.

[Nel99] Nelson, Randal: Relational Algebra. http://anon.cs.rochester.edu/u/www/u/nelson/courses/csc_173/relations/algebra.html, 1999.

[Neu85] Neumann, Peter G.: The Risks Digest. http://catless.ncl.ac.uk/risks, forum on risks to the public in computers and related systems, since 1985.

[New03] Newman, Mark E. J.: The Structure and Function of Complex Networks. In: *SIAM Review*, volume 45:pp. 167–256, 2003. URL `http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0303516`.

[NNH99] Nielson, Flemming; Nielson, Hanne R.; Hankin, Chris L.: *Principles of Program Analysis*. Springer-Verlag, 1999. ISBN: 3540654100, URL `http://www.libri.de/shop/action/productDetails?artiId=1402188`.

[NP00] Naor, Moni; Pinkas, Beny: Distributed Oblivious Transfer. In: *Advances in Cryptology – Asiacrypt '00*, volume 1976 of *LNCS*, pp. 200–219. Springer-Verlag, December 2000.

[NP01] Naor, Moni; Pinkas, Benny: Efficient Oblivious Transfer Protocols. In: *Proceedings of SODA 2001 (SIAM Symposium on Discrete Algorithms), Washington DC.* January 2001.

[Odl03] Odlyzko, Andrew M.: The Case Against Micropayments. In: *Financial Cryptography: 7th International Conference, FC 2003, LNCS 2742*, pp. 77–83. 2003.

[Ogi00] Ogilvie, Sheilagh: Social Capital, Social Networks, and History. Cambridge University memo, `http://www.econ.cam.ac.uk/faculty/ogilvie/social-capital-and-history.pdf`, June 2000.

[OGS07] Oberholzer-Gee, Felix; Strumpf, Koleman: The Effect of File Sharing on Record Sales: An Empirical Analysis. In: *Journal of Political Economy*, volume 115(1):pp. 1–42, February 2007.

[OR94] Osborne, Martin J.; Rubinstein, Ariel: *A Course in Game Theory*. MIT Press, 1994. ISBN 0-262-65040-1. `http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=5073`.

[Pos82] Postel, Jonathan B.: Simple Mail Transfer Protocol (SMTP). RFC 821, IETF, Network Working Group, `http://www.ietf.org/rfc/rfc821.txt`, August 1982.

[QS04] Qiu, Dongyu; Srikant, Rayadurgam: Modeling and Performance Analysis of Bittorrent-like Peer-To-Peer Networks. In: *Proceedings of ACM SIGCOMM*. August 2004. URL `http://citeseer.ist.psu.edu/qiu04modeling.html`.

[RAD78] Rivest, Ron L.; Adleman, Leonard; Dertouzos, Michael L.: On Data Banks and Privacy Homomorphisms. In: DeMillo, R.; Dobkin, D.; Jones, A.; Lipton, R., editors, *Foundations of Secure Computation*. Academic Press, 1978.

[RD01] Rowstron, Antony; Druschel, Peter: Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In: *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany*, pp. 329–350, 2001.

[RE03] R. Elmasri, S. Navathe: *Fundamentals of Database Systems. 4th edition*. Addison-Wesley, 2003. ISBN: 0321204484.

[RFH+01] Ratnasamy, Sylvia; Francis, Paul; Handley, Mark; Karp, Richard; Schenker, Scott: A Scalable Content-Addressable Network. In: *SIGCOMM '01: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 161–172. ACM Press, 2001. ISBN 1-58113-411-8. doi:http://doi.acm.org/10.1145/383059.383072. URL `http://citeseer.ist.psu.edu/ratnasamy01scalable.html`.

[RFI02] Ripeanu, Matei; Foster, Ian; Iamnitchi, Adriana: Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design. IEEE Internet Computing, 6(1), February 2002. `http://people.cs.uchicago.edu/~matei/papers/ic.pdf`, 2002.

[rH06] 3rd, Donald E. Eastlake; Hansen, Tony: US Secure Hash Algorithms (SHA and HMAC-SHA), July 2006. RFC 4634, IETF, Network Working Group, `http://www.ietf.org/rfc/rfc4634.txt`.

[Riv92] Rivest, Ron: The MD5 Message-Digest Algorithm, April 1992. RFC 1321, IETF, Network Working Group, `http://www.ietf.org/rfc/rfc1321.txt`.

[rJ01] 3rd, Donald E. Eastlake; Jones, Paul E.: US Secure Hash Algorithm 1 (SHA1). RFC 3174, IETF, Network Working Group, `http://www.ietf.org/rfc/rfc3174.txt`, September 2001.

[RKCG02] Rogers, Eric; Keeler, Ken; Cohen, David X.; Groening, Matt: Futurama: Anthology of Interest. 20th Century Fox, production code 2ACV16, 2002.

[Rob66] Roberts, Lawrence G.: Towards a Cooperative Network of Time-Shared Computers. Technical report, MIT, Lincoln Laboratory, Lexington, Massachusetts, October 1966. `http://www.packet.cc/files/toward-coop-net.html`.

[Roe99] Roesch, Martin: Snort - Lightweight Intrusion Detection for Networks. In: *Proceedings of the 13th USENIX Systems Administration Conference LISA'99*. USENIX, 1999. `http://www.usenix.org/event/lisa99/roesch.html`.

[Rog00] Rogaway, Phil: OCB Mode: Parallelizable Authenticated Encryption. NIST workshop on modes of operation, `http://csrc.nist.gov/encryption/modes/workshop1/`, October 2000.

[Ros86] Rose, Carol: The Comedy of the Commons: Custom, Commerce, and Inherently Public Property. In: *The University of Chicago Law Review*, volume 53(3):pp. 711–781, 1986. Doi:10.2307/1599583.

[RSA77] Rivest, Ron L.; Shamir, Adi; Adleman, Leonard M.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Technical Report MIT/LCS/TM-82, 1977. URL `http://sherry.ifi.unizh.ch/rivest78method.html`.

[Sch96] Schneier, Bruce: *Applied Cryptography, 2nd Edition*. John Wiley & Sons, 1996. ISBN: 0-471-59756-2.

[SE05] Steinmetz, Ralf; (Eds), Klaus Wehrle: *Peer-to-Peer Systems and Applications*. Lecture Notes in Computer Science, Volume 3485, September 2005. ISBN 3-540-29192-X.

[Sha48] Shannon, Claude E.: A Mathematical Theory of Communication. Bell System Technical Journal, Vol. 27, pp. 379–423, 623–656, Eprint, 1948.

[Sha49] Shannon, Claude E.: Communication Theory of Secrecy Systems. Bell System Technical Journal 28 (4): 656ó715., 1949.

[Sho98] Shoup, Victor: Why Chosen Ciphertext Security Matters. Research Report RZ 3076 (#93122), IBM Research, 1998. URL `http://citeseer.ifi.unizh.ch/547052.html`.

[Sim51] Simpson, Edward H.: The Interpretation of Interaction in Contingency Tables. In: *Journal of the Royal Statistical Society, Ser. B 13*, pp. 238–241, 1951.

[SM95] Shardanand, Upendra; Maes, Patti: Social Information Filtering: Algorithms for Automating "Word of Mouth". In: *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pp. 210–217. 1995. URL `http://citeseer.ist.psu.edu/shardanand95social.html`.

[SMK+01] Stoica, Ion; Morris, Robert; Karger, David; Kaashoek, M. Frans; Balakrishnan, Hari: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In: *Proceedings of the ACM SIGCOMM '01 Conference*. San Diego, California, August 2001.

[SMPD05] Sandler, Daniel; Mislove, Alan; Post, Ansley; Druschel, Peter: FeedTree: Sharing Web Micronews with Peer-to-Peer Event Notification. In: *International Conference on Peer-to-Peer Systems IPTPS'05*, pp. 141–151. 2005.

[SS01] Smith, Sean W.; Safford, David: Practical Server Privacy with Secure Co-Processors. In: *IBM Systems Journal*, volume 40(3):pp. 683–695, 2001.

[Sta03] Stallings, William: *Cryptography and Network Security, 3rd edition*. Prentice Hall, 2003. ISBN: 0-13-111502-2.

[Sti05] Stinson, Douglas: *Cryptography Theory and Practice*. CRC Press, 2005.

[Str04] Streb, Jochen: Die politische Glaubwürdigkeit von Regierungen im institutionellen Wandel. Warum ausländische Fürsten das Eigentum der Fernhandelskaufleute der Hanse schützten. In: *Jahrbuch für Wirtschaftsgeschichte*, volume 1:p. 141156, 2004.

[SW05] Song, Dawn Xiaodong; Wagner, David: personal correspondence, 2005.

[SWP00] Song, Dawn Xiaodong; Wagner, David; Perrig, Adrian: Practical Techniques for Searches on Encrypted Data. In: *IEEE Symposium on Security and Privacy*. 2000. URL `http://citeseer.nj.nec.com/song00practical.html`.

[Thr41] Thrupp, Sylvia L.: Social Control in the Medieval Town. In: *The Journal of Economic History*, volume 1, Supplement: The Tasks of Economic History:pp. 39–52, December 1941. Available from `http://www.jstor.org`.

[Tur50] Turing, Alan M.: Computing Machinery and Intelligence. In: *Mind*, volume 59, 236:pp. 433–460, 1950.

[vABL04] von Ahn, Luis; Blum, Manuel; Langford, John: Telling Humans and Computers Apart Automatically. In: *Communications of the ACM*, volume 47(2):pp. 56–60, 2004. ISSN 0001-0782. doi: \url{http://doi.acm.org/10.1145/966389.966390}.

[Vai06] Vaillant, Noel: Probability Tutorials. `http://www.probability.net/`, 2006.

[Var01] Varadhan, S. R. Srinivasa: *Probability Theory*. American Mathematical Society, Courant Lecture Notes, 2001. ISBN: 0-8218-2852-5.

[vNM07] von Neumann, John; Morgenstern, Oskar: *Theory of Games and Economic Behavior (60th-Anniversary Edition)*. Princeton University Press, 2007. ISBN: 978-0-691-13061-3, `http://press.princeton.edu/titles/7802.html`.

[Vol02] Volchan, Sérgio B.: What is a Random Sequence? The Mathematical Association of America Monthly, Vol. 109, 2002.

[Wag03] Wagner, David: Cryptanalysis of an Algebraic Privacy Homomorphism. In: *Proceedings of the 6th Information Security Conference*. 2003. URL `http://citeseer.ist.psu.edu/wagner03cryptanalysis.html`.

[Was03] Washington, Lawrence: *Elliptic Curves: Number Theory and Cryptography*. Chapman & Hall/CRC, 2003. ISBN 1-58488-365-0.

[Wro02] Wroblewski, Gregory: *General Method of Program Code Obfuscation*. Ph.D. thesis, Wroclaw University of Technology, Institute of Engineering Cybernetics, 2002. URL `http://citeseer.ist.psu.edu/wroblewski02general.html`.

[WYY05a] Wang, Xiaoyun; Yu, Hongbo; Yin, Yiqun Lisa: Efficient Collision Search Attacks on SHA-0. In: *CRYPTO 2005, LNCS*, volume 3621, 2005.

[WYY05b] Wang, Xiaoyun; Yu, Hongbo; Yin, Yiqun Lisa: Finding Collisions in the Full SHA-1. In: *CRYPTO 2005, LNCS*, volume 3621, 2005.

[XT06] Xiao, Xiaokui; Tao, Yufei: Anatomy: Simple and Effective Privacy Preservation. In: *VLDB*, pp. 139–150. 2006.

[YKGF06] Yu, Haifeng; Kaminsky, Michael; Gibbons, Phillip B.; Flaxman, Abraham: SybilGuard: Defending Against Sybil Attacks via Social Networks. In: *Proceedings of the SIGCOMM Conference on Communication*. ACM, 2006.

[ZHS+04] Zhao, Ben Y.; Huang, Ling; Stribling, Jeremy; Rhea, Sean C.; Joseph, Anthony D.; Kubiatowicz, John D.: Tapestry: A Resilient Global-Scale Overlay for Service Deployment. In: *IEEE Journal on Selected Areas in Communications*, volume 22(1), 2004. `http://pdos.csail.mit.edu/~strib/docs/tapestry/tapestry_jsac03.pdf`.

[Zim95] Zimmermann, Philip: *PGP Source Code and Internals*. MIT Press, 1995. ISBN 0-262-24039-4.

[ZS05] Zahn, Thomas; Schiller, Jochen: MADPastry: A DHT Substrate for Practicably Sized MANETs. In: *Proc. of 5th Workshop on Applications and Services in Wireless Networks (ASWN2005)*. Paris, France, June 2005.

Acknowledgements

I thank my advisor Oliver Günther, who not only gave insightful directions, arranged my funding, and sponsored me during my application to the graduate school, but also supported my work with great enthusiasm and inexhaustible patience; Bettina Berendt for her interest in my research and invaluable feedback to earlier drafts of this thesis; and the professors of the Graduate School of Distributed Information Systems, especially Johann Christoph Freytag and Hans-Joachim Lenz, for their time spent in long discussions and for their valuable feedback.

I thank the members of the Humboldt University Institute of Information Systems, especially Sergei Evdokimov, Seda Gürses, and Benjamin Fabian, Anett Kralisch, and Claus Boyens, who greatly inspired and advanced my research in countless cooperations.

Further I feel greatful to my graduate school fellow students; the participants and organizers of the Dagstuhl seminar on algorithms for sensor and ad hoc networks in 2005; the students in the seminar on economics in P2P networks held by Oliver Günther and me in 2003; and Helger Lipmaa, Elmar Wolfstetter, Klemens Böhm, Matthias Bauer, Markus Leypold, as well as many unnamed others for their interest in and impact on my academic life (both those aspects that have made it into this thesis and more or less unrelated ones).

Also, I want to thank the Deutsche Forschungsgesellschaft and the Wirtschaftswissenschaftliche Gesellschaft at Humboldt University for generous research grants, and the Institute of Information Systems for a position as a research assistant.

Most important if somewhat self-evident, I want to thank my parents, my family, and my friends, all those people who choose to be in my life and make it fun and interesting.

Selbständigkeitserklärung (declaration of originality)

Hiermit erkläre ich, dass ich über den üblichen wissenschaftlichen Austausch mit Kollegen, der in meinen Veröffentlichungen und in dieser Arbeit dokumentiert ist sowie die Betreuung durch Prof. Oliver Günther hinaus bei der Verfassung dieser Arbeit keine Hilfe von dritten in Anspruch genommen habe. Als Hilfsmittel für meine Forschung habe ich ausschließlich die zitierten Quellen, die üblichen Recherchewerkzeuge, um diese Quellen zu finden, sowie die in der Arbeit beschriebene Hard- und Software verwendet. Frühere Begutachtungen der hier vorliegenden Arbeit haben nicht stattgefunden.

Ich bezeuge, dass meine Angaben über die bei der Abfassung meiner Dissertation benutzten Hilfsmittel, über die mir zuteil gewordene Hilfe sowie über frühere Begutachtungen meiner Dissertation in jeder Hinsicht der Wahrheit entsprechen.

Berlin, den 31. Juli 2008

Matthias Fischmann