Towards Applying Cryptographic Security Models to Real-World Systems

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Jochen Rill

Tag der mündlichen Prüfung:	21.04.2020
Erster Referent:	Prof. Dr. Jörn Müller-Quade
Zweiter Referent:	Prof. Dr. Hannes Hartenstein

Acronyms

ATM automated teller machine

EMT electronic money transfer

EMV A technical standard for payment cards, originally an acronym for Europay, Mastercard, and Visa

EUC Externalized Universal Composability

GUC Generalized Universal Composability

ITM Interactive Turing Machine

mPIN mobile personal identification number

mTIN mobile transaction identification number

PDP proof of data possession

PIN personal identification number

POS point of sale

PPT Probabilistic Polynomial Time

SSE Symmetric Searchable Encryption

TAN transaction authentication number

UC Universal Composability

Abstract

The cryptographic methodology of formal security analysis usually works in three steps: choosing a security model, describing a system and its intended security properties, and creating a formal proof of security. For basic cryptographic primitives and simple protocols this is a well understood process and is performed regularly. For more complex systems, as they are in use in real-world settings it is rarely applied, however. In practice, this often leads to missing or incomplete descriptions of the security properties and requirements of such systems, which in turn can lead to insecure implementations and consequent security breaches. One of the main reasons for the lack of application of formal models in practice is that they are particularly difficult to use and to adapt to new use cases.

With this work, we therefore aim to investigate how cryptographic security models can be used to argue about the security of real-world systems. To this end, we perform case studies of three important types of real-world systems: data outsourcing, computer networks and electronic payment.

First, we give a unified framework to express and analyze the security of data outsourcing schemes. Within this framework, we define three privacy objectives: *data privacy*, *query privacy*, and *result privacy*. We show that data privacy and query privacy are independent concepts, while result privacy is consequential to them. We then extend our framework to allow the modeling of *integrity* for the specific use case of file systems. To validate our model, we show that existing security notions can be expressed within our framework and we prove the security of CryFS—a cryptographic cloud file system.

Second, we introduce a model, based on the Universal Composability (UC) framework, in which computer networks and their security properties can be described We extend it to incorporate time, which cannot be expressed in the basic UC framework, and give formal tools to facilitate its application. For validation, we use this model to argue about the security of architectures of multiple firewalls in the presence of an active adversary. We show that a parallel composition of firewalls exhibits strictly better security properties than other variants.

Finally, we introduce a formal model for the security of electronic payment protocols within the UC framework. Using this model, we prove a set of necessary requirements for secure electronic payment. Based on these findings, we discuss the security of current payment protocols and find that most are insecure. We then give a simple payment protocol inspired by chipTAN and photoTAN and prove its security within our model.

We conclude that cryptographic security models can indeed be used to describe the security of real-world systems. They are, however, difficult to apply and always need to be adapted to the specific use case.

Zusammenfassung

Der formale Nachweis von Sicherheitseigenschaften ist eine grundlegende Anforderung bei der Entwicklung von modernen kryptographischen Verfahren. Die Vorgehensweise ist dabei in der Regel mehrstufig: Zunächst wird ein geeignetes *Sicherheitsmodell* ausgewählt und das zu untersuchende Verfahren wird darin beschrieben. Dafür existieren eine Vielzahl von unterschiedlichen Modellvarianten. Die Wahl der richtigen Variante hat einen entscheidenden Einfluss darauf, welche Sicherheitseigenschaften später überhaupt nachgewiesen werden können. Anschließend werden die gewünschten Sicherheitseigenschaften in einem sogenannten *Sicherheitsbegriff* beschrieben. Auch hier ist die richtige Wahl entscheidend – bezüglich eines zu schwachen Sicherheitsbegriffs lässt sich nahezu jedes Verfahren als sicher beweisen während ein zu starker Sicherheitsbegriff von keinem Verfahren realisiert werden kann. Schlussendlich wird ein *Sicherheitsbeweis* durchgeführt, mit dem gezeigt wird, dass das Verfahren die erwarteten Sicherheitseigenschaften besitzt.

Diese Vorgehensweise wird bisher jedoch hauptsächlich auf elementare Bausteine (wie beispielsweise *Commitments*), einfache Primitiven (wie beispielsweise Verschlüsselungsverfahren) und auf einfache Protokolle angewendet. Die einzelnen Schritte sind für diese Anwendungsfälle schon gut verstanden. So wird die Sicherheit von Verschlüsselungsverfahren beispielsweise in der Regel mit fest definierten Sicherheitsbegriffen in einem spielbasierten Modell beschrieben.

Bei komplexeren Systemen oder Protokollen, wie sie in der Praxis überwiegend im Einsatz sind, findet diese Vorgehensweise jedoch keine Anwendung. Nicht nur existieren für solche Verfahren keine Sicherheitsbeweise, meistens existiert noch nicht einmal eine formale Beschreibung des Verfahrens und der Annahmen, unter denen bestimmte Sicherheitseigenschaften erreicht werden sollen. Stattdessen wird häufig ein iterativer Entwicklungsprozess angewendet, bei dem der Entwickler aus den Fehlern der vorherigen Iteration lernt. Dies führt in der Praxis leider häufig dazu, dass Annahmen, die der Sicherheit des Systems zugrunde liegen, nur implizit getroffen und daher, meist unbeabsichtigter Weise, beim Einsatz des Systems nicht berücksichtigt werden.

Ein Grund für den Mangel an formaler Methodik bei dem Entwurf solcher Real-Welt-Systeme ist insbesondere, dass bereits die Auswahl und Anwendung eines Sicherheitsmodells ausgesprochen schwierig ist. An das Führen eines Sicherheitsbeweises ist dann nicht mehr zu denken. Selbst ohne Beweisführung bietet jedoch bereits die Modellbildung und -anwendung klare Vorteile: Sie zwingt den Entwickler dazu, auf systematische Art und Weise, getroffene Annahmen und potentielle Schwachstellen seines Verfahrens explizit zu machen. Bekannte kryptographische Modelle sind jedoch nicht ohne Weiteres auf komplexere Systeme übertragbar und es existiert auch

keine etablierte Vorgehensweise, solche Modelle für neue Anwendungsfälle anzupassen.

Diese Arbeit geht daher der Frage nach, inwiefern sich die aus der Kryptographie bekannte Methodik für die Modellierung von Real-Welt-Systemen anwenden lässt. Dazu wurde, anhand von drei konkreten Anwendungsszenarien, untersucht, welche Sicherheitsmodelle und -begriffe sich für das jeweilige Szenario am besten eignen und welche Anpassungen dafür möglicherweise vorgenommen werden müssen. Die Ergebnisse dieser Untersuchung wurden anschließend jeweils am Beispiel von konkreten Protokollen und Systemen validiert.

Im Einzelnen wurden die folgenden Anwendungsfälle betrachtet:

Die Modellierung von Computer-Netzwerken Computer-Netzwerke werden in der Regel mit bestimmten Sicherheitszielen im Hinterkopf entworfen. So sollen beispielsweise Firewalls die Computer in einem bestimmten Netzwerksegment vor Zugriffen aus anderen Segmenten schützen. Die Strukturierung des Netzwerks, sowie der Einsatz und die Verteilung von konkreten Schutzmaßnahmen, erfolgt jedoch häufig auf Basis von Richtlinien und Erfahrungswerten und nicht mithilfe formaler Methodik. Ob die gesetzten Sicherheitsziele durch die konfigurierte Netzwerkstruktur erreicht werden, kann dann nur empirisch evaluiert werden. In dieser Arbeit wurde untersucht, wie sich Computer-Netzwerke und deren Sicherheitseigenschaften in dem *Universal-Composability*-Framework (UC-Framework) beschreiben lassen. Auf die Modellierung von zeitlichen Zusammenhängen, die bei der Kommunikation zwischen Netzwerkteilnehmern häufig eine große Rolle spielen, wurde dabei ein besonderer Fokus gelegt. Da die Modellierung von Zeit in dem UC-Modell selbst für einfache Primitiven eine Herausforderung ist, wurden formale Hilfsmittel und Werkzeuge entwickelt, um die Anwendbarkeit des Modell in dem konkreten Anwendungsfall zu erleichtern.

Zur Validierung wurde das entwickelte Modell auf eine aus mehreren Firewalls bestehende Netzwerk-Architektur angewendet.

Die Modellierung von Verfahren für sicheres Daten-Outsourcing Sicheres Daten-Outsourcing wird in der Kryptographie vor allem im Kontext der *durchsuchbaren Verschlüsselung* erforscht. Für solche Verfahren existieren bereits eine Vielzahl von Sicherheitsbegriffen in unterschiedlichen Sicherheitsmodellen. Anders als bei etablierten Verfahren zur einfachen Verschlüsselung von Daten, gibt es für durchsuchbare Verschlüsselungen jedoch keinen Sicherheitsbegriff, der sich als Goldstandard etabliert hat. Das führt dazu, dass viele Verfahren ihren ganz eigenen, neuen und speziell zugeschnitten Begriff mitbringen. Die Sicherheitseigenschaften unterschiedlicher Verfahren miteinander zu vergleichen ist somit ausgesprochen schwierig. Zudem sind solche hochspezialisierten Sicherheitsbegriffe kaum auf andere, in der Praxis

häufiger zum Einsatz kommende, Outsourcing-Verfahren anwendbar.

In dieser Arbeit wurden daher, aufbauend auf den bestehenden Sicherheitsbegriffen und -modellen für durchsuchbare Verschlüsselung, neue, generalisierte Sicherheitsbegriffe in einem spielbasierten Sicherheitsmodell entwickelt, die sich für alle Arten von Daten-Outsourcing-Verfahren eignen. Um Vergleichbarkeit zu gewährleisten, wurde zusätzlich untersucht, in welchem Bezug die im Rahmen dieser Arbeit entwickelten Sicherheitsbegriffe zu verschiedenen bestehenden Sicherheitsbegriffen stehen.

Zur Validierung wurde das entwickelte Modell auf *CryFS* – ein verschlüsseltes Cloud-Dateisystem – angewendet.

Die Modellierung von elektronischen Bezahlverfahren Die EMV-Protokollfamilie ist weltweit die Grundlage für elektronisches Bezahlen und Geldabheben. Obwohl die Sicherheit von EMV hoch relevant für das Funktionieren der Wirtschaft weltweit ist, wurde keines der dafür relevanten Protokolle bisher in einem formalen Sicherheitsmodell beschrieben. Getroffene Annahmen, potentielle Schwachstellen, sowie erwartete Sicherheitseigenschaften sind daher nicht explizit beschrieben, sondern können nur durch genaues Studium der Protokollbeschreibung rekonstruiert werden. Insbesondere bei den Smart-Card-basierten Bezahlprotokollen zeigen sich genau aus diesem Grund in jüngster Zeit Schwächen. Der Sicherheit dieser Verfahren liegt die implizite Annahme zugrunde, dass die Kommunikation zwischen Smart Card und Point-of-Sale-Gerät bzw. Geldautomat nicht manipuliert werden kann. Diese Annahme ist heutzutage nicht mehr haltbar. Mit moderner Technik ist es möglich, einen unauffälligen Chip so auf der Smart Card anzubringen, dass jegliche Kommunikation abgehört und verändert werden kann. Da diese Annahme in dem Protokollstandard nicht explizit gemacht wurde, konnten die Anwender von EMV auch keine weiterführenden und gezielten Sicherheitsmaßnahmen treffen, um das Bestehen der Sicherheitsannahmen zu gewährleisten.

In dieser Arbeit wurde, erneut auf Basis des UC-Frameworks, ein Modell entwickelt, das es erlaubt, verschiedenste Arten von elektronischen Bezahlprotokollen zu modellieren. Dabei wurde insbesondere auch berücksichtigt, dass bei solchen Verfahren der Faktor Mensch eine große Rolle spielt. Da Menschen, anders als Computer, beispielsweise nicht die Fähigkeit haben kryptographische Signaturen zu prüfen, müssen andere Sicherheitsmaßnahmen zum Einsatz kommen, um die Gesamtsicherheit des Systems zu gewährleisten.

Zur Validierung wurde das entwickelte Modell auf *L-Pay* – ein Verfahren zum elektronischen Bezahlen und Geldabheben – angewendet.

Im Ergebnis wurden in dieser Arbeit für drei konkrete Anwendungsszenarien Sicherheitsmodelle -und begriffe erarbeitet, die es einfacher machen, die Sicherheitseigenschaften von Protokollen und Verfahren in diesen Anwen-

dungsfeldern formal zu beschreiben. Auch wenn hinsichtlich der Anwendbarkeit der kryptographischen Methodik für Real-Welt-Systeme auch weiterhin noch viel Forschungsbedarf besteht (insbesondere auch bei der Automatisierung und Softwareunterstützung), leistet diese Arbeit einen wichtigen Schritt in diese Richtung.

Acknowledgements

About eight years ago, I was quite determined on leaving university and science behind me and starting with a "serious" job in industry. I was then fortunate enough to be offered a topic for a diploma thesis by Dirk Achenbach which re-started my interest in science. As it would turn out, Dirk shared my curiosity about connecting cryptographic methodology with problems from the real world and through long and always fruitful discussions over many years, I think we managed to do just that. Thank you.

Creating this work would also have never been possible without the tremendous support from family, friends and colleagues. I am particularly grateful to Jörn Müller-Quade for supervising my dissertation and for always being available as a constant source of new ideas. My thanks also go to all of my wonderful co-authors but in particular to Jeremias Mechler, who provided the energy and determination to carry our work on electronic payment over the finish line in a time when I was burned out.

Last but not least, thank you Elfi for always providing me with stability, love, and patience, even through difficult times.

Contents

Chapter 1

Introduction

1.1 Motivation

Creating formal proofs of security is a basic requirement for the development of cryptographic protocols nowadays. In general, this task is approached in multiple steps. First, one has to choose an appropriate *security model* or *framework* and describe the chosen cryptographic scheme within that model. There exist quite a number of popular security models: game-based and simulation-based [74] models, Constructive Cryptography [82], UC [22], and many more. Choosing the right model for a specific scheme or protocol determines which security properties can later be proven and which can not. Trying to describe the security of a protocol which guarantees that a message will always be delivered on time, within the UC framework, will fail, since the model inherently does not support time.

Having chosen the appropriate security model, one describes the intended security properties within that model in form of a *security notion*. As with the security model, the correct choice of the security notion is critical. With regards to a security notion which is too weak, almost every scheme can be proven secure, whereas no scheme can fulfill a security notion which is too strong. It is also quite tempting to design security notions which perfectly fit the protocol to which they should be applied. This exhibits the danger that potential weak spots of the protocol do not become clear from the security notion and that the protocol is suddenly secure by definition. Ideally one would first create the security notion and check whether the protocol fits that notion.

Finally, one creates a *security proof*, which shows that the created protocol exhibits the intended security properties. Almost all cryptographic security proofs consist of arguments and reasoning delivered textually in prose, with the help of a formal calculus provided by the security model. Proofs are therefore not only difficult to create correctly, but also difficult to check. Lacking any form of formal verification tools, only experts in a particular

security model can verify proofs within that model—and they are also human and can therefore make mistakes. Indeed, using anecdotal evidence, the author of this thesis had a paper submission rejected because its main theorem was "obvious" and the proof "trivial". As it became clear later, the proof was completely wrong and the opposite of the theorem was true.

For all of the above reasons, applying the cryptographic methodology of models and proofs can be very difficult. Fortunately, with regards to basic cryptographic primitives (such as commitment or encryption schemes) and simple protocols, cryptographic research has established a well funded understanding of all necessary steps over the past decades. For proving the security of encryption schemes for example, there is a set of pre-defined security notions in a game-based security model. New encryption schemes are expected to fulfill these notions if they are to be considered secure.

However, for more complex systems or protocols, such as those used in applications employed in practice, this methodology is rarely applied. As a consequence there are little to no guidelines or predefined models and notions which can be used for these cases. Not only can one not expect to see a formal proof of the security properties attributed to such a system, they often lack any formalized description of their workings.

Instead, systems and protocols used in practice are often developed using an iterative development process, during which the developers learn from their past mistakes and errors are corrected as they occur. This often causes assumptions, upon which the security of the whole system relies, to not be made explicit. Users of such a system run the risk of overlooking these assumptions and thus using the system in an insecure way. This seems especially problematic in the case of EMV (the suite of protocols used for electronic payment), as we will discuss in Chapter 5.

This is not the fault of the system designers, however. As we have discussed, cryptographic security models are tremendously difficult to apply correctly even for experts and are often tailored for a specific use case and can not be applied to a different one without significant modifications. However, real-world systems could benefit immensely from applying the formal methodology used in cryptography. Even if creating an actual proof of security is out of scope, having an easy to apply cryptographic security model at hand, would allow the designers of a system to make underlying assumptions and potential weaknesses explicit. Additionally, formally describing the desired security properties of a system within a security notion before actually designing the system would allow security relevant design mistakes to become apparent earlier.

From all the above, there emerges a research question: can cryptographic methodology be applied to real-world systems, and if so, how? In this work, we aim to answer this question. We will focus our work on creating or adapting security models and notions for usage in real-world system and give security proofs to validate that these models work or to generate more general

knowledge about a system. The question of how to facilitate the creation of cryptographic security proofs for such systems (or in general), however, is its own field of research and—even though extremely important—will not be addressed by this thesis.

1.2 Our Contribution

To answer the presented research question we chose three real-world use cases and investigated which cryptographic security models and which security notions can be used to describe systems within these use cases. We validated the results of this investigation by applying these models and notions to specific systems and protocols from these use cases.

Modeling secure data outsourcing In cryptography, research in practical schemes for secure data outsourcing is mainly being done with regards to searchable encryption.

For searchable encryption schemes, there exists a plethora of security notions in game-based and simulation-based security models. Different from regular encryption schemes however, there is no single gold standard security notion, which all schemes must fulfill. This leads to every scheme bringing its own, specifically tailored security notion. Thus, comparing the security properties of different searchable encryption schemes is next to impossible. Also, these highly specific security notions can rarely be applied to other searchable encryption schemes or even outsourcing schemes in general without major modifications.

In Chapter 3 we developed new and generalized security notions based on existing ones for searchable encryption, which are applicable to a wide range of data outsourcing schemes. To that end, we first precisely define security notions for three seemingly disparate security objectives: data privacy, query privacy, and result privacy. We show how these notions are related. Second, we give generalizations of these notions to make them applicable to both highly elaborate and practical schemes. To guarantee compatibility with existing security notions and to allow existing schemes to be compared among one another, we also investigated how our notions relate to popular existing notions. We show that several notions from literature are specializations of one of our notions, others intertwine them.

To validate our model, we tried to apply it to *CryFS*, an encrypted cloud file system. We found that the model lacked the ability to describe an important security property, which is crucial for file systems: integrity. We extended our model with a new security notions which allows to describe integrity of file systems.

Modeling computer networks Computer networks are usually designed with specific security goals in mind. Firewalls, for example, should isolate computers within a specific network segment from access from other segments. However, the structure of the network, as well as the distribution of specific security measures within it, is usually chosen based on experience and guidelines, lacking any kind of formal methodology. Whether the envisioned security goals are met by a specific network configuration, can only be validated empirically. In Chapter 4 we investigated how computer networks and their security goals and properties can be described and analyzed using the UC framework. Since latency and timing are often very important for networks, we specifically focus on how to describe theses properties. Since modeling time is impossible in the basic UC framework, we employ an extension created by Katz et al. [75]. We developed formal tools and modeling guidelines to facilitate the modeling of computer networks within this extension.

For validation, we applied the resulting security model and security notions to architectures of multiple firewalls and restate a popular theorem from literature within our model.

Modeling electronic payment The world-wide system for electronic payment and cash withdrawal is based on a family of protocols called EMV. Even though the security of EMV is extremely important for the functioning of the world-wide economy, none of the relevant protocols within the family have been described and analyzed within a formal security model. Underlying assumptions, potential weaknesses, as well as the expected security properties are therefore not described explicitly but can only be inferred through careful study of the protocol description. Smartcard based protocols in particular have shown weaknesses for this reason in recent times. The security of these protocols is based upon the implicit assumption that the communication between smartcard and point-of-sale device can not be manipulated. This assumption does not hold anymore. Using modern technology, it is possible to mount an inconspicuous chip on the smartcard which can intercept and manipulate all communication. Since this assumption has not been made explicit in the protocol standard, users and implementers of EMV are not able to take further security measures.

In Chapter 5, we developed a formal model based on the UC framework, which allows us to describe various kinds of electronic payment protocols and which incorporates a stronger but also more realistic adversarial model than has been used for the design of EMV. We first give a *formal description* of electronic payment which works for both payment at a point of sale (POS) and for the withdrawal of cash at an automated teller machine (ATM). Second, we provide an ideal functionality for electronic payment, which captures the desired security guarantees for such protocols. In particular, this model allows for modeling human-based protocol interactions. Since humans, in contrast

to computers, are for example not able to verify cryptographic signatures, other security measures have to been taken in order to ensure the security goals of the whole system.

We then prove a set of *general requirements* for designing such protocols. These requirements can act as a guideline for future protocol designers. Based on these results, we argue that a number of current payment systems are insecure, even on a conceptual level. Inspired by this analysis, we propose a simple electronic payment protocol which mainly requires secure communication between the bank and the initiator of a transaction. We use this protocol to validate our model.

1.3 Structure of this Work

Chapter 2 introduces basic definitions and notations used in the remainder of this work and gives an introduction into game-based and simulation-based security notions, as well as into the UC framework. In Chapter 3 we explore how data outsourcing schemes can be modeled. Chapter 4 discusses how to apply a cryptographic security framework to the case of computer networks. In Chapter 5 we investigate the case of electronic payment protocols. Chapter 6 concludes.

Chapter 2

Preliminaries

In this chapter, we give a brief introduction to the different kinds of security models used within this thesis, along with general definitions and notations.

2.1 General Definitions and Notation

Definition 1 (Negligable)**.** A function $f : \mathbb{N} \to \mathbb{R}$ is *negligable* if the following holds:

$$\forall c \in \mathbb{N} \exists k_0 \in \mathbb{N} \forall k > k_0 : |f(k)| < \frac{1}{k^c}$$

As an abbreviation, we write $f(k) < \mathsf{negl}(k)$.

Definition 2 (Indistinguishablity)**.** Two binary ensembles X and Y are *indistinguishable* ($X \approx Y$), if $\forall c, d \in N\ \exists k_0 \in N$, so that for all $k > k_0$ and all $a \in \cup_{\kappa \leq k^d} \{0,1\}^\kappa$ holds:

$$|\Pr(X(k,a) = 1) - \Pr(Y(k,a) = 1)| < k^{-c}$$

In other words, the difference between the two ensembles is negligable.

Definition 3 (Probabilistic Polynomial Time (PPT))**.** A probabilistic algorithm is PPT, if his time complexity is polynomial with regards to the length of its input.

Definition 4 (Symmetric Encryption Scheme)**.** A symmetric encryption scheme $\mathcal{E}$ is a tuple $\mathcal{E} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with

- $\mathsf{Gen} : 1^k \to \{0,1\}^k$ is a PPT algorithm which given a security parameter k, outputs a key K.
- $\mathsf{Enc} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$ is a PPT algorithm which given a key K and a plaintext outputs the corresponding ciphertext.

- $\mathsf{Dec} : \{0,1\}^k \times \{0,1\}^m \to (\{\perp\} \cup \{0,1\}^n)$ is a PPT algorithm which given a key K and a ciphertext outputs the corresponding plaintext. It outputs $\perp$ if K is wrong or the ciphertext was not valid.

Definition 5 (Public Key Encryption Scheme)**.** A symmetric encryption scheme $\mathcal{E}_{\text{pk}}$ is a tuple $\mathcal{E}_{\text{pk}} := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with

- $\mathsf{Gen} : 1^k \to \{0,1\}^k \times \{0,1\}^k$ is a PPT algorithm which given a security parameter k, outputs a private key sk and a public key pk.
- $\mathsf{Enc} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$ is a PPT algorithm which given a public key pk and a plaintext outputs the corresponding ciphertext.
- $\mathsf{Dec} : \{0,1\}^k \times \{0,1\}^m \to (\{\perp\} \cup \{0,1\}^n)$ is a PPT algorithm which given a secret key sk and a ciphertext outputs the corresponding plaintext. It outputs $\perp$ if sk is wrong or the ciphertext was not valid.

Encryption schemes, both public and symmetric, usually need to be *correct*. This means that $\forall k, m : \mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$ (with $pk = sk$ for symmetric encryption schemes).

2.2 Game-based Security

In a game-based security model, the security of a cryptographic scheme is described in terms of a game with two players: a challenger $\mathcal{C}$ (also called *experiment*) and an adversary $\mathcal{A}$. The challenger has a specific question in mind, which the adversary must answer in order to win the game. To this end, the adversary can usually ask the challenger various questions about his task before he has to give an answer. If the adversary can find an answer better than by guessing, he wins. The specific question the adversary has to answer, as well as the allowed interaction with the challenger, depends on the specific security properties (the *security notion*) one wants to describe within the security model.

We will illustrate this by example of encryption schemes. One of the basic security properties each encryption scheme has to have in order to be considered secure is *one-wayness*. This means that, only given a ciphertext, extracting the plaintext is difficult, even if one can encrypt arbitrary messages (which is always the case when using public key encryption schemes, since the encryption key is public by definition). This can be formalized by the following security game.

Security Game 1 ($\mathsf{OW\text{-}CPA}^{\mathcal{A}}_{\mathcal{E}_{\text{pk}}}(k)$)**.**

- The challenger generates a key pair $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$
- The adversary gets access to pk.

- The challenger chooses a random message m from the message space M and generates $c \leftarrow \mathsf{Enc}(pk, m)$
- The adversary gets the encrypted message c and outputs a plaintext m'.

The result of the game is 1 if $m = m'$ and 0 otherwise.

In order for an encryption scheme PKE to be secure with regards to this definion, we require that the advantage of the adversary in the security game is *negligable*.

$$\Pr[\mathsf{OW\text{-}CPA}^{\mathcal{A}}_{\mathcal{E}_{\mathrm{pk}}}(k) = 1] < \mathsf{negl}(k)$$

In this example, the specific question the adversary is asked by the challenger is "decrypt this ciphertext". Other than that, no further interaction with the challenger is possible.

If one wants to describe the security of symmetric encryption schemes within a game-based model, the introduction of so-called *oracles* becomes neccessary. Since with symmetric encryption schemes the encryption key is the same as the decryption key and has to remain secret, the adversary cannot easily generate encryptions himself. In order to capture the same security properties as stated above, the challenger has to provide the adversary with encryptions. We say, that the adversary has access to an *encryption oracle*. The following security game illustrates the difference.

Security Game 2 ($\mathsf{OW\text{-}CPA}^{\mathcal{A}}_{\mathcal{E}}(k)$).

- The challenger generates a key $K \leftarrow \mathsf{Gen}(1^k)$
- The adversary gets access to an encryption oracle $\mathsf{Enc}(K, \cdot)$.
- The challenger chooses a random message m from the message space M and generates $c \leftarrow \mathsf{Enc}(K, m)$
- The adversary gets the encrypted message c and outputs a plaintext m'.

The result of the game is 1 if $m = m'$ and 0 otherwise.

Note that we use $\cdot$ to denote free parameters of oracles, which can be chosen by the adversary.

Since in this work, we mainly consider schemes and protocols which use symmetric encryption, going forward when stating security notions, we will always use the oracle-based variant.

It is quite obvious, that one-wayness does not capture all security properties one would expect from a modern encryption scheme. In particular, one would expect an encryption to not only be resilient against decryption of random message, but of any given message. Indeed, even if the adversary gets to choose the message space, from which the challenger chooses the plaintext

for encryption, he should not be able to tell which message got encrypted. We expect two independantly created ciphertexts (even of the same message) must be *indistinguishable* from each other and call the resulting security notion *indistinguishability under chosen plaintext*.

Security Game 3 ($\mathsf{IND\text{-}CPA}^{\mathcal{A}}_{\mathcal{E}}(k)$).

- The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$ and a random bit $b \leftarrow \{0,1\}$.
- The adversary is given oracle access to $\mathsf{LR}(K, m_0, m_1)$, which outputs an encryption of m_b under K, if $|m_0| = |m_1|$.
- $\mathcal{A}$ submits a guess b' for b.

The result of the experiment is 1, if $b' = b$, and 0 otherwise.

So, even if the adversary can choose two arbitrary messages to his liking, he cannot tell which one of them got encrypted only by looking at the ciphertext.

Note that there are several equivalent formalizations for IND-CPA security [74]. In this work, we use the formalization with a *left-or-right oracle* to reduce the complexity of our models and proofs.

Keeping plaintexts hidden is not the only security property we expect from an encryption scheme nowadays. We also want an encryption to provide *integrity*, which means that any manipulation of a ciphertext will be detected during decryption. This is a security property which is orthogonal to indistinguishability under chosen plaintext and the resulting security notions is called *integrity of ciphertexts* [14].

Security Game 4 ($\mathsf{INT\text{-}CTXT}^{\mathcal{A}}_{\mathcal{E}}(k)$).

- The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$.
- The adversary is given oracle access to $\mathsf{Enc}(K, \cdot)$.
- The adversary is given oracle access to $\mathsf{Dec}(K, \cdot)$.

The result of the experiment is 1, if for any Dec oracle query: $\mathsf{Dec}(K, c) \neq \perp$ and c was never *output* by the Enc oracle.

Intuitively this states, that even if an adversary can perform arbitrary encryption and decryption operations, he cannot produce a new, valid ciphertext.

Proofs within this security model generally work with a technique called *reduction*. If one wants to prove that a specific scheme fulfills a specific security notion, one tries to construct a *reduction* algorithm. This algorithm uses an adversary with more than negligable advantage in the corresponding security game to break a fundamental security assumption upon which the

scheme is build (for example that factoring is hard). Since the security assumption is always true, by contradiction such an adversary cannot exist. Thus, a proof *reduces* the security properties of the particular scheme to one (or even multiple) security assumption.

2.3 Simulation-based Security

Another way of describing security properties is by comparing an actual execution of a protocol or scheme in presence of an adversary to an idealized execution, where there is strictly less information available. In the idealized execution a *simulator* takes the place of the adversary and tries to simulate the behavior of the real adversary as best as he can. If the outputs of these two executions are indistinguishable for a specific scheme, it is considered to be secure.

For encryption schemes, there also is a simulation-based security notion, which is called *semantic security*. We restate its definition as given by Katz and Lindell [74].

Definition 6 (Semantic Security)**.** A symmetric encryption scheme $\mathcal{E}$ is *semantically secure* if for every PPT algorithm $\mathcal{A}$ there exists a PPT algorithm $\mathcal{S}$ such that for any polynomial-time computable functions f and h, the following holds:

$$|\Pr[\mathcal{A}(1^k, \mathsf{Enc}(K, m), h(m)) = f(m)] - \Pr[\mathcal{S}(1^k, |m|, h(m)) = f(m)]| < \mathsf{negl}(k)$$

Informally speaking, this states, that each function (f), that an adversary can compute on a ciphertext, can also be computed by a simulator who receives no ciphertext at all, but only its length ($|m|$) and auxillary information ($h(m)$). Creating proofs in this model usually works by explicitly constructing such an algorithm $\mathcal{S}$, which in turn internally simulates the behavior of algorithm $\mathcal{A}$. One then has to argue, that the input algorithm $\mathcal{A}$ receives in both worlds, is indistinguishable.

Depending on the specific use case, simulation-based security notions can be more difficult to work with, since proofs are more difficult to perform. Fortunatelly, in the case of encryption, it can be proven, that semantic security is equivalent to indistinguishability [58].

2.4 The Universal Composability Framework

In this section we give a brief review of the UC framework by Canetti [21]. It is a tool for proving the security of complex multi-party protocols by comparing their execution with an idealized version of the protocol. We use this framework to model more complex systems like firewall architectures, computer networks and electronic payment, which cannot be adequately

captured by game- or simulation-based security notions. In many ways, the UC framework can be seen as a more strictly formalized version of the simulation-based security model presented above.

Since its introduction in 2001 the UC framework has changed quite a bit. In the following we will give a brief overview of the framework based on its most recent version from 2019. In the UC framework, participants in a protocol are modeled as Interactive Turing Machines (ITMs). Since there are different definitions of ITMs in literature, we will briefly summarize the definition given by Canetti [21].

Definition 7 (Interactive Turing Machine)**.** An ITM μ is a multi-tape turing machine with the following augmentations: A tape is *externally writeable* (EW), if it can be written by every other turing machine.

- an *identity tape*: This tape is read-only and mainly holds the program of μ.
- an *outgoing message tape*: This tape holds all messages μ wants to send, together with information about the recipient.
- Three externally writeable tapes:
  - an *input tape*: This tape can be used to provide input to μ by external programs.
  - an *subroutine-output tape*: This tape holds information generated by sub-programs running on μ.
  - a *backdoor tape*: This tape holds information coming from the adversary and can be used to influence the program running on μ.
- an *activation tape*: This tape holds the information, whether the ITM is currently running or not.
- An external-write instruction: This transfers messages from the outgoing message tape to a specified tape from another ITM.
- A read next message instruction: This instruction reads a complete message from tape.

We call an ITM *probabilistic*, if it additionally has a *random tape*, which contains a random bit string of a specific distribution.

Informally speaking, an execution of a protocol π in the UC framework, can be described by a number of interacting ITMs. This execution happens in the context of two additional ITMs: the adversary $\mathcal{A}$ and the environment $\mathcal{Z}$. The environment represents the perception of the execution from an outside point of view and is thus generally responsible for proving input to a protocol

(i.e. by writing a value to the input tape of an ITM) and receiving its output. The adversary represents the party which wants to attack the protocol.

The adversary and the environment are allowed to communicate freely. One can even proof that a dummy adversary, who is completely controlled by the environment, is equivalent to an adversary who acts on his own accord.

The adversary controls all communication between ITMs. That is, if an ITM wants to send a message to another ITM, it writes that message to the backdoor tape of the adversary, who in turn writes the message to the backdoor tape of the recipient. During this process, there is no guarantee that message will stay unmodified. Which tape can be written to by which ITM is enforced by a control function C.

The execution of the protocol is turn-based. If an ITM is activated, it can perform computations and write to a tape of any other ITM based on the aforementioned restrictions. Then its turn ends. If an ITM receives input on one of its tapes, it is the next to be activated. The first ITM to be activated is always the environment $\mathcal{Z}$.

It is obvious that in a network, where the adversary controls all communication, there can be no secure protocols without some kind of setup assumption (i.e. pre-distributed keys), since the adversary can change all messages at will. Indeed, a common reference string is sufficient to securely realize a broad variety of protocols [22].

While using some form of encryption in a protocol will prevent the adversary from arbitrarily changing messages, he can always choose to not deliver a particular message at all. This is nothing that can be prevented by using some kind of additional setup assumption or security measure, but is inherent to the framework. For this very reason, availability can not be expressed in the basic UC framework.

Setup assumptions, like a common reference string, can be expressed using so-called hybrid or ideal functionalities. Hybrid functionalities can be accessed by the ITMs running the protocol like a subroutine: they directly write to the input tape of the functionality and receive the output on their subroutine tape. With this mechanism, they can perform actions (i.e. receiving a secret common reference string or send a secret message to a party) without the adversary having full control over the communication. In theory, it is possible to define hybrid functionalities which completely leave the adversary out of the interaction. However, such a functionality will itself never be realizable by any protocol within the UC framework, since, as we discussed, in the basic UC framework, the adversary always has the power to stop messages from being delivered.

If one wants to model adaptive corruption, the adversary can send a special message to the backdoor tapes of other parties: the *corruption message*. If a party receives a corruption message, it stops executing its own program and instead gives complete control of its functions to the adversary. This includes disclosing its internal state. However, to simplify the analysis, we will use

static corruption, where all parties have been corrupted prior to the protocol execution.

The output of the whole protocol is the output of $\mathcal{Z}$ and we assume, without loss of generality, that it consists of one bit. The distribution of all outputs of $\mathcal{Z}$ is a random ensemble based on the two parameters z (the input) and k (the security parameter).

Definition 8 (Ensemble of a protocol execution)**.** We denote the random variable which describes the execution of a protocol π with adversary $\mathcal{A}$, environment $\mathcal{Z}$, input z, security parameter k as $\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$. The set of random distributions $\{\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)\}_{k\in N,z\in\{0,1\}^*}$ is denoted as $\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$.

The security of a protocol execution in the UC framework is based on a comparison with an execution of an idealized version of the protocol: the *ideal protocol*. The ideal protocol contains the *ideal functionality* $\mathcal{F}_{\mathsf{ideal}}$ which completely realizes the properties of the analyzed protocol. In the ideal protocol, all parties only act as dummies which directly give their input to the ideal functionality and receive back their output without performing any computation themselves. The ideal functionality may communicate with the adversary in order to model the influence $\mathcal{A}$ is allowed to have. We call this adversary the "adversary simulator" $\mathcal{S}$. Since all computations are performed by $\mathcal{F}_{\mathsf{ideal}}$, which is ideal by definition, the whole protocol execution is ideal and thus secure. Note that this does not model an absolute security guarantee but a guarantee relative to the defined ideal functionality.

Definition 9 (Ideal protocol)**.** Let $\mathcal{F}_{\mathsf{ideal}}$ be an ideal functionality. Then, the ideal protocol which realizes $\mathcal{F}_{\mathsf{ideal}}$ is denoted as $\mathsf{IDEAL}_{\mathcal{F}}$.

Informally, a protocol π is UC secure if, for every adversary $\mathcal{A}$ and every environment $\mathcal{Z}$, $\mathcal{Z}$ can not distinguish if it is interacting with π or with the ideal protocol implementing π. Because parties may behave indeterministically, their outputs are modeled as distributions. Further, since protocol runs are parameterized (e.g. by the security parameter k), the following definition uses probability ensembles.

Based on that notion, we now formalize the indistinguishability of two protocols in the UC framework. The simulator's job is to simulate the presence of $\mathcal{A}$ to the environment, so that it cannot distinguish the real protocol execution from the idealized version. The security notion requires that there is a successful simulator for every adversary.

Definition 10 (UC emulates)**.** Let π and ϕ be two protocols. Then π UC emulates the protocol ϕ, if $\forall\mathcal{A}\ \exists\mathcal{S}$, so that $\forall\mathcal{Z}$ holds:

$$\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}} \approx \mathsf{EXEC}_{\phi,\mathcal{S},\mathcal{Z}}$$

We can now formally state when a protocol *realizes* a functionality.

Definition 11 (UC realizes)**.** A protocol π (securely) *UC realizes* an ideal functionality $\mathcal{F}_{\mathsf{ideal}}$, if π UC emulates the corresponding ideal protocol $\mathsf{IDEAL}_{\mathcal{F}}$.

If a protocol π realizes a given ideal functionality, then we say π is *UC secure*.

The UC framework is a powerful instrument for analyzing the security of protocols because it provides a composition theorem.

Theorem 1 (The Composition Theorem)**.** *Let ρ, ϕ, π be protocols such that ϕ is a subroutine of ρ and π UC-emulates ϕ. Then protocol $\rho^{\phi \to \pi}$ UC-emulates ρ*

Informally speaking, the composition theorem states that if π securely realizes an ideal functionality $\mathcal{F}_{\mathsf{ideal}}$, one can use π instead of $\mathcal{F}_{\mathsf{ideal}}$ in other protocols without compromising security. This allows us to break down large protocols into smaller components (i.e. hybrid functionalities) and analyze their security separately.

2.5 Synchronized Universal Composability

As we have discussed, the UC framework is inherently asynchronous. Exactly one machine can run at any given moment and the adversary is in control of all communication.

Katz et al. [75] created an extension to the UC framework alleviate this issue. To this end, they introduced two new hybrid functionalities: a clock ($\mathcal{F}_{\mathsf{clock}}$, c.f. Figure 2.2) and a bounded delay channel ($\mathcal{F}^{\delta,l}_{\mathsf{BD\text{-}SMT}}$, c.f. Figure 2.1).

$\mathcal{F}_{\mathsf{clock}}$ allows the parties to wait for each other at synchronization points. A party can signal when its round is complete. When all parties have completed their round, the round counter is reset.

$\mathcal{F}^{\delta,l}_{\mathsf{BD\text{-}SMT}}$ allows to model communication channels where the adversary cannot delay the delivery of messages indefinitely. Each channel has an incoming queue and each protocol participants polls the queue regularly. The adversary may increase the delay on the channel up to a predefined limit (δ). When a party polls the incoming queue for a channel, the counter is decreased. When it reaches zero, the party receives the next element from the channel queue.

$\mathcal{F}_{\mathsf{clock}}$ together with bounded-delay channels are sufficient to prove *guaranteed termination* for multi-party protocols [75], i.e. the protocol does not "hang" indefinitely.

This somehow seems to circumvent the impossibility of expressing availability in the basic UC framework by the use of only two hybrid functionalities. Note however that both $\mathcal{F}_{\mathsf{clock}}$, as well as $\mathcal{F}^{\delta,l}_{\mathsf{BD\text{-}SMT}}$, require fundamental changes to the basic UC framework. $\mathcal{F}_{\mathsf{clock}}$ does not wait upon parties which

The bounded delay channel $\mathcal{F}_{\mathsf{BD\text{-}SMT}}^{\delta,l}(p_s, p_r)$

Initialise $M := \perp$ and $D := 1$, and $\hat{D} := 1$.

- Upon receiving a message m from p_s set $D := 1$ and $M := m$ and send $(\mathsf{send}, l(M))$ to the adversary.
- Upon receiving a message (fetch) from p_r:
  1. Set $D := D - 1$
  2. If $D = 0$, then send (sent, M) to p_r
- Upon receiving (delay, T) from the adversary, if $\hat{D} + T \leq \delta$, then set $D := D + T$,$\hat{D} := \hat{D} + T$, and return ($\mathsf{delay\text{-}set}$) to the adversary; otherwise ignore the message.
- Upon receiving $(\mathsf{replace}, m', T')$ from the adversary, if p_s is corrupted, $D > 0$, and delay T' is valid, then set $D := T'$ and set $M := m'$.

Figure 2.1: The bounded delay channel by Katz et al. [75]. δ is an upper bound on the delay the adversary can impose on the delivery of a single message.

are under adversarial control in order to prevent the adversary from delaying the protocol execution indefinitely. Regular hybrid functionalities in the UC framework however, are not aware which parties are corrupted. Also, in the basic framework, parties only get activated when they receive a message (i.e. through delivery by the adversary on their backdoor tape). Using $\mathcal{F}_{\mathsf{BD\text{-}SMT}}^{\delta,l}$ however, parties do not receive messages until they have polled the channel a sufficient number of times, which they can only do if they are active. To circumvent this, Katz et al. [75] change the model of execution in a way that the environment is responsible for activating all parties and advancing the protocol.

Even though this seems like a subtle change, we will see in Chapter 4 that it makes the framework more difficult to handle.

2.6 Generalized Universal Composability

When a protocol π has been proven secure in the UC framework, it remains secure even when composed concurrently with *arbitrary* protocols. However, since all parties are freshly created upon each execution of a protocol, UC

The clock function $\mathcal{F}_{\mathsf{clock}}$

Initialise for each party p_i a bit $d_i := 0$.

- Upon receiving message (RoundOK) from party p_i set $d_i = 1$. If for all honest parties $d_i = 1$, then reset all d_i to 0. In any case, send (switch, p_i) to $\mathcal{A}$.
- Upon receiving message (RequestRound) from p_i, send d_i to p_i.

Figure 2.2: The ideal $\mathcal{F}_{\mathsf{clock}}$ functionality by Katz et al. [75]. Parties can signal that they are done for the current round. When all honest parties have signaled RoundOK the round counter is reset. Further, parties can request the status of the round counter, learning whether the round has changed.

does not allow for shared state between different protocols. This makes it impossible to model a wide variety of protocols which sustain state between multiple executions of the protocol, such as a bank card. In real life, one would not use a new card every time one wants to withdraw money.

The Generalized UC framework (GUC) [24] solves this problem. It introduces so-called global functionalities which can not only be used in multiple instances of *the same* protocol, but also concurrently in multiple instances of *arbitrary* protocols, resulting in a much stronger expressiveness and security guarantees. Formally, this is captured by modifying the UC experiment. In the standard UC experiment, the environment $\mathcal{Z}$ has to distinguish between *one* execution of the real protocol π with the real adversary $\mathcal{A}$ (denoted by $\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$) and the ideal protocol with the ideal functionality $\mathcal{F}$ and the simulator $\mathcal{S}$ (denoted by $\mathsf{EXEC}_{\phi,\mathcal{S},\mathcal{Z}}$). In the GUC experiment the environment interacts with multiple challenge sessions and may *additionally* invoke multiple additional protocols $\rho_1, \ldots, \rho_n$ *of its choice* that may arbitrarily share state even with the challenge sessions.

We re-state the definition of GUC emulation from Canetti et al. [24]:

Definition 12 (GUC-emulation)**.** Let π and ϕ be PPT multi-party protocols. We say that π GUC-emulates ϕ if, for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{S}$ such that for any PPT environment $\mathcal{Z}$, we have:

$$\mathsf{GEXEC}_{\pi,\mathcal{A},\mathcal{Z}} \approx \mathsf{GEXEC}_{\phi,\mathcal{S},\mathcal{Z}}.$$

Since the enviroment can create arbitary parallel instances of the challenge protocol, prooving that a protocol GUC-emulates another protocol would require the proof to consider all possible protocols, which is tremendously difficult.

Fortunatelly, there also exists a simplified version of GUC that is called Externalized Universal Composability (EUC). Here, $\mathcal{Z}$ may invoke exactly *one* ITM instance with the code of the shared functionality $\overline{\mathcal{G}}$. $\mathcal{Z}$ may interact with $\overline{\mathcal{G}}$ directly and without the adversary. It also may invoke dummy parties with arbitrary party IDs (which might coincide with IDs of parties in the challenge protocol) and use them to access functionalities in $\overline{\mathcal{G}}$. Thus, "personalized" hybrid functionalities, such as the per-party global certification functionality $\overline{\mathcal{G}}^{P}_{\text{cert}}$ (cf. [64]) can be arbitrarily used by $\mathcal{Z}$ in the name of a *honest* protocol party the challenge session.

We say that a protocol π is $\overline{\mathcal{G}}$*-subroutine-respecting* if it only shares state via $\overline{\mathcal{G}}$.

Definition 13 (The EUC execution [24])**.** Let π and ϕ be PPT multi-party protocols, where π is $\overline{\mathcal{G}}$-subroutine-respecting. We say that π EUC-emulates ϕ with respect to a shared functionality $\overline{\mathcal{G}}$ (or in shorthand, that π $\overline{\mathcal{G}}$-emulates ϕ) if for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{S}$ such that for any $\overline{\mathcal{G}}$-constrained PPT environment $\mathcal{Z}$, we have:

$$\mathsf{EXEC}^{\overline{\mathcal{G}}}_{\pi,\mathcal{A},\mathcal{Z}} \approx \mathsf{GEXEC}^{\overline{\mathcal{G}}}_{\phi,\mathcal{S},\mathcal{Z}}.$$

For $\overline{\mathcal{G}}$-subroutine-respecting protocols, EUC and Generalized Universal Composability (GUC) are equivalent:

Theorem 2 (EUC-GUC equivalence criteria [24])**.** *Let π be any protocol which invokes no shared functionalities other than (possibly) $\overline{\mathcal{G}}$, and is otherwise subroutine respecting (i.e. π is $\overline{\mathcal{G}}$-subroutine respecting). Then,* π GUC-emulates *a protocol ϕ, if and only if π $\overline{\mathcal{G}}$-EUC-emulates ϕ.*

Chapter 3

Modeling Secure Data Outsourcing

The following chapter is based on joint work with Dirk Achenbach, Matthias Huber, Sebastian Messmer and Jörn Müller-Quade. Parts of the included content have already been presented in the following works:

- Dirk Achenbach, Matthias Huber, Jörn Müller-Quade, Jochen Rill: *Closing the Gap: A Universal Privacy Framework for Outsourced Data.* BalkanCryptSec 2015 [4].
- Sebastian Messmer, Jochen Rill, Dirk Achenbach, Jörn Müller-Quade: *A Novel Cryptographic Framework for Cloud File Systems and CryFS, a Provably-Secure Construction.* DBSec 2017 [83]
- Dirk Achenbach: *On Provable Security for Complex Systems.* PhD Thesis 2016 [1]
- Matthias Huber: *Provable and practical security for database outsourcing.* PhD Thesis 2016 [66]

As the inventor of CryFS, Sebastian Messmer deserves a special mention at this point. Even though his encrypted cloud file system is presented in some detail in this thesis, it only serves as a case-study for our security models and all credit for design and implementation goes to him.

3.1 Introduction

In recent years, cloud computing has transformed from a trend to a serious competition for traditional on-premise solutions. Elastic cost models and the availability of virtually infinite resources present an alternative to offers of a preset volume. The more bandwidth is available to consumers, the more economically reasonable it is to replace an on-premise solution with a cloud

solution. In the wake of the PRISM disclosures, it seems naïve to trust in the security of one's data in the cloud, however. The scientific challenge for security researchers is to solve this dilemma by finding solutions without sacrificing the economic benefits of cloud technology.

Cryptographic research offers methods that guarantee the confidentiality and integrity of data in the presence of an adversary. By expressing and analyzing the resulting schemes in a *formal model* (as detailed in Section 1.1), one can reduce trust requirements even further.

Yet, provably-secure schemes are rarely adopted in practice. The abstract computational models that form the basis of cryptographic frameworks don't usually facilitate a straightforward implementation. Also, the concept of efficiency in these models differs from practical efficiency notions, so that many schemes which are asymptotically efficient in theory, are next to unusable in practice. In contrast, there are many practical solutions to security challenges. They are deployed widely, but seldom lend themselves to a formal security analysis and are thus analyzed in an "ad-hoc" fashion.

One exemption from this rule is the field of searchable encryption. Researchers in this field aim to design efficient schemes (evaluated using benchmarks), as well as formal security notions. However, in general these security notions are custom-made for the particular protocol. While a tailored security notion helps accurately express a scheme's security properties, it makes comparing security properties difficult.

This chapter, we investigate the question, whether it is possible to provide a unified formal security model for data outsourcing schemes in general. In particular, our focus is on allowing to capture the security of schemes used in practice. We first identify three conceptually different privacy goals: keeping the outsourced data itself private, keeping the queries to the data private, and keeping the result of the query private. We show that data privacy and query privacy are independent concepts, while result privacy is consequential to them.

We applied our proposed model to different existing outsourcing schemes, to evaluate its usefulness. We found that our results are applicable to constructions from seemingly disparate fields of cryptographic research, e.g. private information retrieval, searchable encryption, and secure database outsourcing.

When applied to an encrypted cloud file system, however, our model fails. Different from other outsourcing schemes, file systems must protect the *integrity* of the outsourced data in particular. This means that a malicious server must not be able to provide the client with data that is modified in any way without the client noticing. In particular, this includes providing an old state of the data, even though the client has updated the data previously. We find that this property can not be expressed within our model by means of data, query or result privacy.

We extend our model by security notions for the integrity of file systems

and evaluate them by applying them to *CryFS* [1], which is a popular encrypted cloud file system, available by default in many modern linux distributions.

3.2 Related Work

In the following, we introduce new security notions for four concepts: data privacy, query privacy, result privacy and integrity of ciphertexts. Therefore, we divide the related work concerning security notions for data outsourcing schemes into three categories: notions which only consider the privacy of the outsourced data, notions which only consider the privacy of the queries, those which intertwine both and security notions for file systems.

Security Notions for Data Privacy There is a rich body of literature on data outsourcing schemes which only consider the privacy of the outsourced data in both static and adaptive settings. There are game-based notions [55, 67, 49], simulation-based notions [27, 26, 71], and notions that use the Universal Composability framework [21, 94, 78]. A well-known example for an adaptive security notion is IND-CKA established by Goh [55]. The intuition is that an adversary should not be able to distinguish two sets of data of his choosing based on the generated index even if he can issue and observe queries. However, in Goh's notion, the queries the adversary can choose are strongly restricted: he is not allowed to query for words that are exclusive to one of the two sets he chooses as challenge. This severely restricts the kinds of schemes which can be described within this model.

An example for a notion which only considers static security is Huber et al.'s IND-ICP [67]. Here, the idea is that an adversary should not be able to distinguish the encryptions of two databases. However, the databases the adversary is challenged on are restricted to being independent permutations of one another.

Security Notions for Query Privacy Hiding queries on outsourced data on a single server has been studied in the context of Single-Server Private Information Retrieval [31] (PIR). The PIR notion requires that an adversary who observes access patterns cannot distinguish any two queries. PIR does not guarantee that the data itself is kept private [31]. There is a rich body of literature on PIR schemes which have sublinear *communication complexity* ([79, 20, 53]). However, all PIR schemes inherently have a *computational complexity* for the server which is linear in the size of the data [92]. The PIR security notion is thus not applicable to efficient schemes.

The privacy of queries on data has also been investigated in the context of Oblivious RAMs (ORAMs) first introduced by Goldreich and Ostrovsky [56] and further explored and improved upon by others [87, 91, 38]. Similar to

[1] https://www.cryfs.org/

PIR, an "oblivious" RAM is one that cannot distinguish access patterns—the data itself is not required to be private. As is the case with PIR, all ORAM constructions can not be considered efficient in our sense. They either have polylogarithmic computation cost while requiring the client to store a constant amount of data [91] or have logarithmic computation cost, but require the client to store at least a sublinear amount of data dependent on the size of the RAM [59]. Therefore, the security notion for ORAM is not suitable for our cause.

Security Notions for Data Privacy as well as Query Privacy There are security notions in the literature which consider both data privacy as well as query privacy. Chase et al. [29] introduce the simulation-based notion of "chosen query attacks" which models both the privacy of queries and that of the data. However, in their notion, the concepts of privacy for data and privacy for queries are intertwined. Haynberg et al. [62] try to separate both properties: they introduce the notion of "data privacy" and complement it with "pattern privacy", which is similar to PIR. However, their notion for data privacy only allows the adversary to observe the execution of one query. While the notion works for their scheme, this limitation is too strict for other schemes.

Modeling Information Leakage A reoccurring pattern in security notions for practical schemes is the use of a *leakage function* which describes the information the scheme leaks to the adversary during execution. A certain amount of leakage seems necessary in order for schemes to be efficient. Cash et al. investigate the construction of efficient and practical schemes that also have a formal security analysis [27, 26]. Their analyses follow a simulation-based approach. The constructions leak information about the plaintext and the query which they explicitly model by a *leakage function* $\mathcal{L}$. This is similar to Chase et al. [29], whose notion allows to describe the information that leaks through the encryption itself ($\mathcal{L}_1$) and the information about the ciphertext *and* the queries combined that is leaked by evaluating queries ($\mathcal{L}_2$). Stefanov et al. [94] employ the same technique in the Universal Composability Framework. In game-based notions such leakage is modelled by restricting the challenges the adversary can choose. Thus, in our framework we define "leakage relations" that model information leakage.

Secure Cloud File Systems There are various existing commercial and free solutions for secure cloud storage [2][3][4]. None of them have a formal proof of security. There has been research into how to model the security

[2] `https://spideroak.com`
[3] `http://tresorit.com`
[4] `http://www.boxcryptor.com`

of file systems, however, most of this research is directed at disk encryption schemes. Damgård et al. [37] for example introduce a formalization of encryption schemes for file systems that is based on the Universal Composability framework. However, there are many artefacts in their model which are not relevant in the cloud setting (e. g. they explicitly model physical and logical sectors). Their model also misses important components on which security is regularly based upon (i.e. different states for client and server) and thus is not well suited for our setting. Kristian Gjøsteen [54] and more recently Khati et al. [76] both introduce a game-based security model, which, however, is also only suited for modeling full disk encryption. There is a rich body of work regarding outsourcing schemes and corresponding security models which provide proofs of data possession (PDPs) and proofs of retrievability (e. g. Zhang et al. [99], Erway et al. [48] and Cash et al. [25]). Similar to our goals, all these schemes provide integrity for outsourced data. However, their requirements are fundamentally different. The goal of a PDP scheme is for a cloud provider to be able to prove that he has all of the outsourced data and that he did not modify it maliciously without requiring the user to hold a copy of the data himself and without having to download it. This is very useful if the server performs computations on the outsourced data without interaction of the user and the user wants to verify if all the data is still correct. In our case however, the server is only used for storage and users interact with the data only locally. Thus, all integrity checks can be performed by the user on the data itself. In order to achieve these particular integrity guarantees, PDP schemes require design and performance trade offs, which are also reflected in their security models. This makes the schemes incomparable to our scheme and the security models hard to adapt to our case.

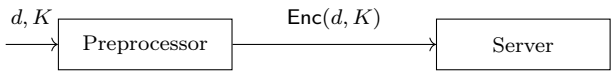
3.3 A Model for Outsourced Data

Our basic object of interest is a *data set*—be it a database, an e-mail archive or a collection of images. One can execute *queries* on this data. The *result* of the execution of a query on a data set can be any function of the data.

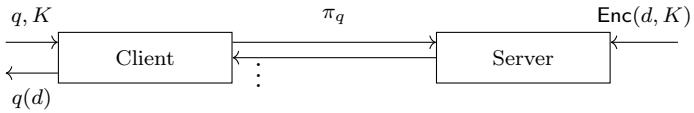
We first focus on queries that only return a function of the data they are executed on. Note, however, that updating data is also important and will be investigated further later on.

In an outsourcing scenario, a *client* transfers the data to a *server* for storage. Before the data is transferred, the client *encrypts* it using a private key. Instead of executing queries locally, the client transforms them into an interactive *protocol* that it runs with the server. The client's input into the protocol is its private key, the server's input is the encrypted data. See Figure 3.1 for an illustration of our outsourcing model.

We assume the server may be under adversarial influence. We restrict



(a) initialisation phase



(b) query phase

Figure 3.1: We model the interaction of the client with the server in two phases. In the initialization phase (a) a preprocessing agent receives a data set d and an encryption key K, prepares the encrypted data $\mathsf{Enc}(d, K)$ and uploads it to the server. In the query phase (b) the client issues queries q using encryption key K by running protocol π_q with the server. The server's input is the encryption $\mathsf{Enc}(d, K)$. After the interaction the client outputs the query result $q(d)$.

the adversary to honest-but-curious behavior however—he may not deviate from the protocol as to interfere with its correct execution, but he may try to learn private information from the interaction.

Efficiency

We are particularly interested in capturing the security properties of *efficient* schemes within our framework. In practice, the size of outsourced data sets can easily measure terabytes. Since data is uploaded to the server only once while queries get executed repeatedly, it is not practical to use encryption schemes which process the whole data set on each query—or even a fraction of it. Therefore, we consider schemes efficient which have strictly logarithmic *communication and computation complexities* per query—for the client as well as for the server. This is in contrast to many schemes in the literature which are considered efficient even though they have a polynomial overhead.

Privacy

There are three privacy objectives: Keeping the data private, keeping the queries private, and keeping the results private.

Keeping the Data Private Bob runs an e-mail service. Alice uses the service and is concerned Bob might snoop through her private e-mail. *Data Privacy* guarantees that Bob does not learn anything about the content of Alice's e-mail.

Keeping the Queries Private In this scenario, Bob runs a patent search service. Clients can submit construction plans and Bob's service looks for any patents the construction is infringing. Alice is worried that Bob might learn her ideas and register them as patents himself. If the query protocol has *query privacy*, Bob cannot learn the content of Alice's requests.

Keeping the Results Private Bob also owns a database of DNA markers that are a sign of genetic diseases. He offers his customers the possibility to check blood samples against the database. Alice runs a clinic and is interested in Bob's services. She has no interest in disclosing to Bob whether her patients suffer from any genetic diseases. If the method of accessing Bob's database has *result privacy*, the results to Alice's requests are hidden from Bob. As we will show, result privacy implies database privacy as well as query privacy and vice versa.

3.4 Security Notions for Data Outsourcing Schemes

In this section we define precise terminology for securely outsourcing data and establish fundamental relations.

3.4.1 Notation and Conventions

We use the probabilistic-polynomial time (PPT) model, i.e. we assume all machines, algorithms, parties and adversaries to be restricted to a polynomial number of computation steps (in the security parameter). (Note that the formal notion of efficiency here is different to our idea of a scheme's *practicality* (see also Section 3.3).) Similarly, we say a protocol is *efficient* if its number of communication rounds does not exceed a fixed polynomial (in the security parameter). We denote the set of all efficient two-party protocols with Π. Our definitions can be extended to allow for the interaction of multiple servers with multiple clients. For the sake of clarity, we focus on the single-server-single-client case and leave a further generalisation of the definitions for future work.

We define our algorithms and protocols to operate on a *domain*. In this work, a domain is the set of all possible values in the given context—for example, in a database context, a domain would be the set of all databases. Our algorithms and protocols operate on three domains: a domain of the data to be encrypted (i.e. plaintexts) Δ, a domain of ciphertexts Γ, and a domain of results P.

3.4.2 Outsourcing Data

In this section, we define the elementary concepts used in the rest of the paper.

Definition 14. A *data set* $d \in \Delta$ is an element of a domain Δ. By $|d|$ we denote the length of its (unique) binary representation.

For example, in the scenario of an outsourced e-mail archive, Δ is the set of all mailboxes and a concrete data set (mailbox) $d \in \Delta$ is a set of e-mail messages. To outsource data, one requires an algorithm that makes the data ready to be uploaded to a server. We call this process "encrypting" the data, as we will require later that no adversary can learn the original data from the outsourced data.

Definition 15. An *outsourcing scheme* for data sets is a tuple $(\mathsf{Gen}, \mathsf{Enc})$ such that

$$\mathsf{Gen} : 1^k \to \{0,1\}^n$$
$$\mathsf{Enc} : \Delta \times \{0,1\}^n \to \Gamma$$

We call an outsourcing scheme for a data set *retrievable* if there is a function $\mathsf{Dec} : \Gamma \times \{0,1\}^n \to \Delta$ such that $\forall K \in \{0,1\}^n, d \in \Delta : \mathsf{Dec}(\mathsf{Enc}(d,K),K) = d$.

We do not require that encrypted data sets be decryptable. The main purpose of outsourcing data in this work is to remotely execute queries on it.

Definition 16. A *query* q is a PPT algorithm that, on input of a data set $d \in \Delta$ returns a *result set* $q(d) \in P$ and an updated data set $d_q \in \Delta$.

We point out that, to simplify notation, we do not model parameters for queries explicitly. Our model supports parameterized queries nevertheless, as for each pair of a query q with parameters p, there is an equivalent query $q(p)$ that has the parameters "hard coded". Without loss of generality we assume that queries are functions of the data, i.e. $\forall q \exists d_1, d_2 \in \Delta : q(d_1) \neq q(d_2)$, and the query result is the same if evaluated twice.

Our idea of the "correctness" of a protocol is relative to a given query q.

Definition 17. A two-party protocol $\pi_q \in \Pi$ between a Server $\mathfrak{S}$ and Client $\mathfrak{C}$ *executes a query* q for a outsourced data set $\mathsf{Enc}(d,K)$ if

- The Client, on input of a key K, outputs the result set $q(d) = \pi_q^{\mathfrak{C}}(K, \mathsf{Enc}(d,K))$.
- The Server, on input the outsourced data set $\mathsf{Enc}(d,K)$, outputs an updated outsourced data set $\mathsf{Enc}(d_q,K) = \pi_q^{\mathfrak{S}}(\mathsf{Enc}(d,K))$.

Definition 18. A *queryable outsourcing scheme* for data sets is a tuple $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})$ such that

- $(\mathsf{Gen}, \mathsf{Enc})$ is an outsourcing scheme for data sets, and
- $\mathsf{Q} \subseteq \Pi$ is a non-empty set of efficient two-party protocols that execute a query for outsourced data sets.

We stress that the client has no direct access to the data when interacting with the server in order to execute a query.

To be able to argue about privacy in the presence of queries to outsourced data, we require a notion of what a protocol party "sees" during the execution of a protocol.

Definition 19. A *view* of a protocol party is the totality of its inputs, received messages, sent messages and outputs. To denote the view of protocol party $\mathfrak{P} \in \{\mathfrak{C}, \mathfrak{S}\}$ in protocol π with inputs c and K, we write

$$\mathsf{view}^{\pi}_{\mathfrak{P}}(c, K).$$

In particular, the encrypted data is part of the server's view.

3.4.3 Privacy Notions for Outsourced Data Sets

Static Security

The static notion of privacy for outsourced data captures the intuition that no adversary may deduce any information about the data from its ciphertext alone. We model it closely after the IND-CPA (Indistinguishability Under Chosen Plaintext Attack) [74] notion.

Security Game 5 ($\mathsf{IND\text{-}CDA}^{\mathcal{A}}_{(\mathsf{Gen},\mathsf{Enc})}(k)$).

1. The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$ and a random bit $b \leftarrow \{0, 1\}$.
2. The adversary $\mathcal{A}$ is given input 1^n and oracle access to $\mathsf{Enc}(\cdot, K)$.
3. $\mathcal{A}$ outputs two data sets d_0 and d_1 of equal length to the experiment.
4. $\mathcal{A}$ is given $\mathsf{Enc}(m_b, K)$.
5. $\mathcal{A}$ outputs a guess b' for b.

The result of the experiment is 1 if $b' = b$ and 0 else.

Definition 20 (Static Security)**.** An outsourcing scheme $(\mathsf{Gen}, \mathsf{Enc})$ has *indistinguishable encryptions under chosen-data attacks* IND-CDA or *static security*, if

$$\forall \mathcal{A}, c \in \mathbb{N}\, \exists k \in \mathbb{N} : |\Pr[\mathsf{IND\text{-}CDA}^{\mathcal{A}}_{(\mathsf{Gen},\mathsf{Enc})} = 1]| \leq \frac{1}{2} + k^{-c}$$

Privacy in the Presence of Queries

When outsourced data sets are queried three conceptually different privacy goals can be distinguished: keeping the data private, keeping the queries private, and keeping the results private. We model these privacy goals as security games. The adversary is supplied an oracle for views on the interaction between client and server and tries to discern the challenge bit b.

In all three security experiments, in addition to a challenge oracle, the adversary is supplied with an "open" view oracle. The oracle provides views for arbitrary queries executed on an encryption of arbitrary data sets *using the challenge key*. It implies that the scheme must have a probabilistic property in the sense that two identical queries on two different encryptions of the same plaintext will not access the same parts of the ciphertext. This can either be done by randomizing the structure of the encrypted ciphertext (as in the work of Haynberg et al. [62]) or by randomizing the protocol which realizes the query.

Security Game 6 ($\mathsf{D\text{-}IND}^{\mathcal{A}}_{(\mathsf{Gen},\mathsf{Enc},\mathsf{Q})}(k)$)**.**

1. The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$.
2. $\mathcal{A}$ receives access to an oracle for $\mathsf{view}^{\pi_\cdot}_{\mathfrak{S}}(\mathsf{Enc}(\cdot, K))$ and continues to have access to it. The oracle takes a query q and a data set d as input and returns $\mathsf{view}^{\pi_q}_{\mathfrak{S}}(\mathsf{Enc}(d, K))$.
3. $\mathcal{A}$ outputs two data sets d_0 and d_1 of equal length to the experiment.
4. The experiment draws a random bit $b \leftarrow \{0, 1\}$.
5. Challenge oracle: $\mathcal{A}$ is given access to an oracle for $\mathsf{view}^{\pi_\cdot}_{\mathfrak{S}}(\mathsf{Enc}(d_b, K))$. That is, the oracle takes any query q such that $\pi_q \in \mathcal{Q}$ as input, internally runs the protocol π_q on $\mathsf{Enc}(d_b, K)$, and outputs $\mathsf{view}^{\pi_q}_{\mathfrak{S}}(\mathsf{Enc}(d_b, K))$ to the adversary.
6. $\mathcal{A}$ outputs a guess b' for b.

The result of the experiment is 1 if $b' = b$ and 0 else.

Security Game 7 ($\mathsf{Q\text{-}IND}^{\mathcal{A}}_{(\mathsf{Gen},\mathsf{Enc},\mathsf{Q})}(k)$)**.**

1. The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$.
2. $\mathcal{A}$ receives access to an oracle for $\mathsf{view}^{\pi_\cdot}_{\mathfrak{S}}(\mathsf{Enc}(\cdot, K))$ and continues to have access to it. The oracle takes a query q and a data set d as input and returns $\mathsf{view}^{\pi_q}_{\mathfrak{S}}(\mathsf{Enc}(d, K))$.
3. $\mathcal{A}$ outputs two queries q_0 and q_1 to the experiment. q_0 and q_1 must yield protocols π_{q_0} and π_{q_1} with the same number of protocol messages.

4. The experiment draws a random bit $b \leftarrow \{0,1\}$.

5. Challenge oracle: $\mathcal{A}$ is given access to an oracle for $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_b}}(\mathsf{Enc}(\cdot, K))$. That is, the oracle takes any data set $d \in \Delta$ as input, internally runs the protocol π_{q_b} on $\mathsf{Enc}(d, K)$, and outputs $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_b}}(\mathsf{Enc}(d, K))$ to the adversary.

6. $\mathcal{A}$ outputs a guess b' for b.

The result of the experiment is 1 if $b' = b$ and 0 else.

Definition 21 (Data Privacy)**.** An outsourcing scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})$ has *Data Privacy*, if

$$\forall \mathcal{A}, c \in \mathbb{N}\, \exists k \in \mathbb{N} : |\Pr[\mathsf{D\text{-}IND}^{\mathcal{A},R_d}_{(\mathsf{Gen},\mathsf{Enc},\mathsf{Q})}(k) = 1]| \leq \frac{1}{2} + k^{-c}$$

The privacy notion of Query Privacy captures the goal of hiding the queries themselves from the server. The notion is equivalent to Private Information Retrieval (see Section 3.6.1 for a discussion and proof).

Definition 22 (Query Privacy)**.** An outsourcing scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})$ has *Query Privacy*, if

$$\forall \mathcal{A}, c \in \mathbb{N}\, \exists k \in \mathbb{N} : |\Pr[\mathsf{Q\text{-}IND}^{\mathcal{A},R_q}_{(\mathsf{Gen},\mathsf{Enc},\mathsf{Q})}(k) = 1]| \leq \frac{1}{2} + k^{-c}$$

The third privacy goal, Result Privacy, captures the idea that the adversary must not learn the result of any query executed on any data. To state this idea formally, we allow the adversary to output two data-set-query pairs (d_0, q_0) and (d_1, q_1), as a result is always determined by a query and a data set on which it is evaluated. We then challenge the adversary on the view of query q_b executed on data set d_b.

Security Game 8 ($\mathsf{R\text{-}IND}^{\mathcal{A}}_{(\mathsf{Gen},\mathsf{Enc},\mathsf{Q})}(k)$)**.**

1. The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$.

2. $\mathcal{A}$ receives access to an oracle for $\mathsf{view}_{\mathfrak{S}}^{\pi_\cdot}(\mathsf{Enc}(\cdot, K))$ and continues to have access to it. The oracle takes a query q and a data set d as input and returns $\mathsf{view}_{\mathfrak{S}}^{\pi_q}(\mathsf{Enc}(d, K))$.

3. $\mathcal{A}$ outputs two data-set-query pairs (d_0, q_0) and (d_1, q_1) to the experiment. ($|d_0| = |d_1|$ and q_0 and q_1 must yield protocols π_{q_0} and π_{q_1} with the same number of protocol messages.)

4. The experiment draws a random bit $b \leftarrow \{0,1\}$.

5. Challenge: The experiment runs the protocol π_{q_b} on $\mathsf{Enc}(d_b, K)$ and outputs $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_b}}(\mathsf{Enc}(d_b, K))$ to the adversary.

6. $\mathcal{A}$ receives oracle access to $\mathsf{view}_{\mathfrak{S}}^{\pi_\cdot}(\mathsf{Enc}(d_b, K))$ and $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_b}}(\mathsf{Enc}(\cdot, K))$.

7. $\mathcal{A}$ outputs a guess b' for b.

The result of the experiment is 1 if $b' = b$ and 0 else.

Definition 23 (Result Privacy)**.** An outsourcing scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})$ has *Result Privacy*, if

$$\forall \mathcal{A}, c \in \mathbb{N}\, \exists k \in \mathbb{N} : |\Pr[\mathsf{R\text{-}IND}^{\mathcal{A},R_d,R_q}_{(\mathsf{Gen},\mathsf{Enc},\mathsf{Q})}(k) = 1]| \leq \frac{1}{2} + k^{-c}$$

Fundamental Relations Among the Basic Security Notions

We establish fundamental relations among the three concepts of Data Privacy, Query Privacy, and Result Privacy.

Theorem 3 ($\mathsf{D\text{-}IND} \not\Longleftrightarrow \mathsf{Q\text{-}IND}$)**.** *If a data outsourcing scheme that has Data Privacy exists, there is a data outsourcing scheme that has Data Privacy but no Query Privacy.*

Proof. Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})$ be a data outsourcing scheme that has Data Privacy. We modify it in a way that we violate Query Privacy, but keep Data Privacy intact. To this end, we amend the protocols that execute queries to have the client transmit the executed query in the clear after the actual protocol is complete. We have to show that the modification violates Query Privacy, but does not violate Data Privacy.

With the modification, the adversary in experiment $\mathsf{Q\text{-}IND}$ can easily extract the executed query from any view and thus determine the challenge bit with certainty. Thus the modification violates Query Privacy. To see that this modification does not violate Data Privacy, first note that the modified scheme retains its Data Privacy up until the point of the modification. We argue that the transmission of the query in the clear does not break Data Privacy. Consider experiment $\mathsf{D\text{-}IND}$. Because the experiment draws the key K after the scheme is fixed, the scheme is independent of the actual key used to encrypt the data set d. Further, because the query is supplied by the adversary in the experiment and the adversary has learned neither d_b nor K up to this point, the query is also independent of d_b and K. This concludes the argument. ■

Theorem 4 ($\mathsf{Q\text{-}IND} \not\Longleftrightarrow \mathsf{D\text{-}IND}$)**.** *If there is a retrievable data outsourcing scheme that has Static Security, there is a data outsourcing scheme that has Query Privacy and Static Security, but no Data Privacy.*

Proof. Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})$ be a retrievable data outsourcing scheme that has Static Security. We construct a modified scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q}')$ that suits our purposes. By adopting Gen and Enc, we retain static security. We design

Q' such that it has Query Privacy, but trivially loses Data Privacy. Q' is constructed iteratively, starting with an empty set. For each protocol $\pi_q \in \mathsf{Q}$ that realizes a query q, we define a protocol π'_q to Q' as follows:

(Recall that the client's input is the encryption key K and a query q; the server's input is an encrypted data set $\mathsf{Enc}(d, K)$.)

1. Client: Transfer K to the Server.
2. Server: Decrypt $\mathsf{Enc}(d, K)$ and send $d = \mathsf{Dec}(\mathsf{Enc}(d, K), K)$ back to the Client.
3. Client: Execute query q locally on d and output $q(d)$.

Protocol π' transmits the data set d in the open, violating Data Privacy. Because the client executes q locally and never transmits any information that depends on q, π' does have Query Privacy.

■

The following theorems show that Result Privacy is equivalent to both Data Privacy and Query Privacy (at the same time).

Theorem 5 ($\mathsf{R\text{-}IND} \implies \mathsf{D\text{-}IND}$)**.** *There is no data outsourcing scheme that has Result Privacy but no Data Privacy.*

Proof. Assume a data outsourcing scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})$ for which there is an efficient adversary $\mathcal{A}$ against experiment $\mathsf{D\text{-}IND}$. We give an efficient reduction for $\mathcal{A}$ that breaks the Result Privacy (experiment $\mathsf{R\text{-}IND}$) of the scheme, contradicting the assumption.

The reduction is straightforward. It has to provide a challenge oracle $\mathsf{view}_{\mathfrak{S}}^{\pi,\cdot}(\mathsf{Enc}(d_b, K))$. Such an oracle is provided by experiment $\mathsf{R\text{-}IND}$ and only has to be "passed through".

■

Theorem 6 ($\mathsf{R\text{-}IND} \implies \mathsf{Q\text{-}IND}$)**.** *There is no data outsourcing scheme that has Result Privacy but no Query Privacy.*

Proof. The proof of Theorem 6 is analogous to the proof of Theorem 5 and omitted here. ■

Theorem 7 ($\mathsf{D\text{-}IND} \wedge \mathsf{Q\text{-}IND} \implies \mathsf{R\text{-}IND}$)**.** *Data Privacy and Query Privacy together imply Result Privacy, i.e. there is no data outsourcing scheme that has Data Privacy and Query Privacy but no Result Privacy.*

We prove the statement using a game-hopping technique. Assume any adversary against $\mathsf{R\text{-}IND}$. We replace both view oracles for d_b and q_b, respectively, with an oracle for fixed challenges d_0 and q_0. We argue the indistinguishability of these steps with Data Privacy and Query Privacy.

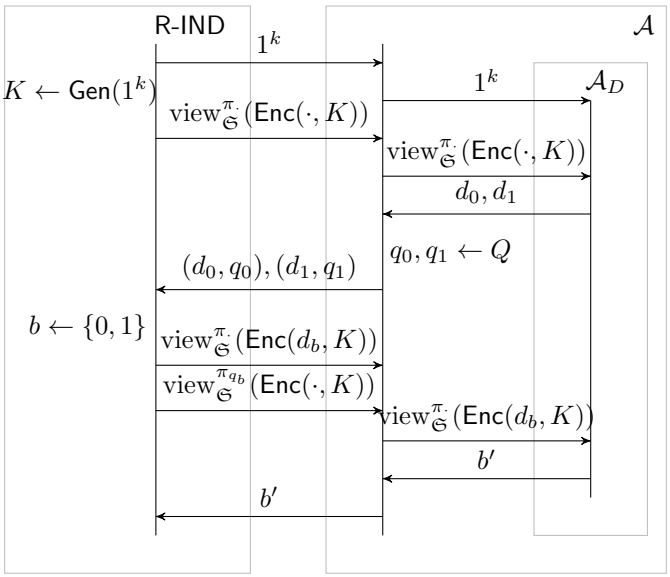


Figure 3.2: Sketch for the proof of Theorem 5.

Finally, in the now-transformed experiment, the adversary has no advantage since his input is independent of b. Concluding, given a scheme with Data Privacy and Query Privacy, no adversary against Result Privacy has a non-negligible advantage.

Proof. We define two game transformations, R-IND′ and R-IND″, starting from the Result Privacy experiment R-IND. In the unmodified experiment R-IND, the adversary is supplied with two view oracles $\mathsf{view}_{\mathfrak{S}}^{\pi_\cdot}(\mathsf{Enc}(d_b, K))$ and $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_b}}(\mathsf{Enc}(\cdot, K))$. In R-IND′ we replace the $\mathsf{view}_{\mathfrak{S}}^{\pi_\cdot}(\mathsf{Enc}(d_b, K))$ oracle by an oracle for $\mathsf{view}_{\mathfrak{S}}^{\pi_\cdot}(\mathsf{Enc}(d_0, K))$. In R-IND″ we further replace $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_b}}(\mathsf{Enc}(\cdot, K))$ by $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_0}}(\mathsf{Enc}(\cdot, K))$. In R-IND″ the adversary receives no input that is dependent on the challenge bit b. He thus has no advantage over guessing b. We have to argue that R-IND″ is indistinguishable from R-IND for the adversary. To this end, we prove the indistinguishability of R-IND from R-IND′ in Lemma 1 and the indistinguishability of R-IND′ from R-IND″ in Lemma 2. ■

Lemma 1. *An adversary who can distinguish between running in experiment* R-IND *and experiment* R-IND′ *yields a successful adversary against database privacy.*

We model the distinction as an experiment D-Oracle-IND in which the adversary must decide whether he is running in R-IND or in R-IND′. It is modeled closely after R-IND. In experiment R-IND′ the challenge database d_b is replaced with the fixed database d_0. Thus, in D-Oracle-IND the adversary is

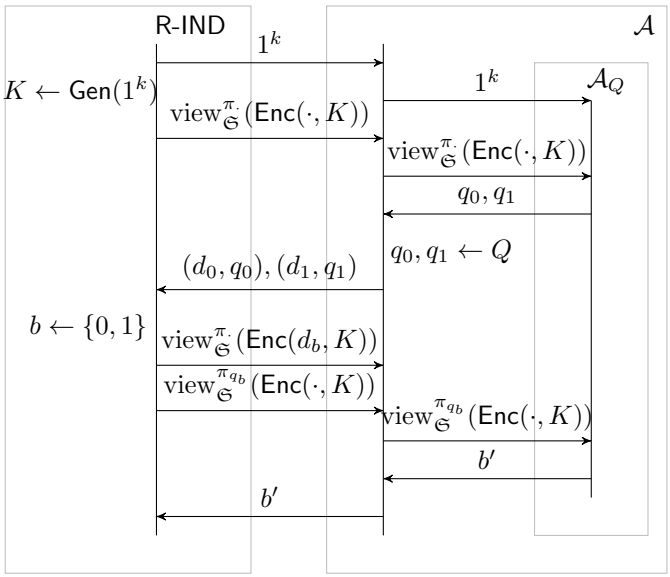


Figure 3.3: Sketch of the proof for Theorem 6.

challenged on deciding whether he has access to an oracle $\mathsf{view}^{\pi_\cdot}_{\mathfrak{S}}(\mathsf{Enc}(d_b, K))$ which is based on the actual challenge or whether he is accessing the fixed oracle $\mathsf{view}^{\pi_\cdot}_{\mathfrak{S}}(\mathsf{Enc}(d_0, K))$. (The query view oracle $\mathsf{view}^{\pi_{q_b}}_{\mathfrak{S}}(\mathsf{Enc}(\cdot, K))$ is provided with no change.) We give a reduction $\mathcal{R}$ which transforms this adversary into an adversary on database privacy. To clearly separate the different challenge bits, we name the challenge bit in the distinction experiment c.

Security Game 9 ($\mathsf{D\text{-}Oracle\text{-}IND}^{\mathcal{A}}_{(\mathsf{Gen},\mathsf{Enc},\mathsf{Q})}(k)$).

1. The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$.
2. $\mathcal{A}$ receives access to an oracle for $\mathsf{view}^{\pi_\cdot}_{\mathfrak{S}}(\mathsf{Enc}(\cdot, K))$.
3. $\mathcal{A}$ outputs two data-set-query pairs (d_0, q_0) and (d_1, q_1) to the experiment (under the restriction that $|d_0| = |d_1|$ and that π_{q_0} and π_{q_1} exchange the same number of messages).
4. The experiment draws two random bits $b \leftarrow \{0,1\}$ and $c \leftarrow \{0,1\}$.
5. Challenge: If $c = 0$, output $\mathsf{view}^{\pi_{q_b}}_{\mathfrak{S}}(\mathsf{Enc}(d_b, K))$ to the adversary. Else, output $\mathsf{view}^{\pi_{q_b}}_{\mathfrak{S}}(\mathsf{Enc}(d_0, K))$.
6. Oracles: If $c = 0$, $\mathcal{A}$ receives oracle access to $\mathsf{view}^{\pi_\cdot}_{\mathfrak{S}}(\mathsf{Enc}(d_b, K))$, else to $\mathsf{view}^{\pi_\cdot}_{\mathfrak{S}}(\mathsf{Enc}(d_0, K))$.
7. $\mathcal{A}$ receives access to $\mathsf{view}^{\pi_{q_b}}_{\mathfrak{S}}(\mathsf{Enc}(\cdot, K))$.

8. $\mathcal{A}$ outputs a guess c' for c.

Proof. Assume an adversary $\mathcal{A}$ with a non-negligible advantage in experiment D-Oracle-IND. We construct a reduction that has a non-negligible advantage in experiment D-IND. The experiment D-IND provides us with access to two oracles: $\mathsf{view}_{\mathfrak{S}}^{\pi_\cdot}(\mathsf{Enc}(\cdot, K))$ and $\mathsf{view}_{\mathfrak{S}}^{\pi_\cdot}(\mathsf{Enc}(d_b, K))$. We use these oracles to simulate the two oracles and the challenge expected by D-Oracle-IND:

- Oracle $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_b}}(\mathsf{Enc}(\cdot, K))$: Fix $q_b := q_1$ and use the $\mathsf{view}_{\mathfrak{S}}^{\pi_\cdot}(\mathsf{Enc}(\cdot, K))$ oracle provided by D-IND to provide $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_1}}(\mathsf{Enc}(\cdot, K))$ to $\mathcal{A}$.
- Oracle $\mathsf{view}_{\mathfrak{S}}^{\pi_\cdot}(\mathsf{Enc}(d_b, K))$: This oracle is provided by D-IND and is relayed.
- Challenge $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_b}}(\mathsf{Enc}(d_b, K))$: Fix $q_b := q_1$ and provide $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_1}}(\mathsf{Enc}(d_b, K))$ as the challenge.

We can now distinguish two cases. If $b = 1$ in D-IND, the challenges and oracles provided to $\mathcal{A}$ running in D-Oracle-IND are consistent (the selected database matches the selected oracle) as they would be when used in R-IND. Therefore the correct challenge bit in the reduction is $c = 0$. On the other hand, if $b = 0$, the views are inconsistent as they would be in R-IND$'$ and $c = 1$. Thus, we return $\mathcal{A}$'s guess as the reduction's own guess $b' = 1 - c'$ to inherit $\mathcal{A}$'s success probability.

■

Lemma 2. *An adversary who can distinguish between* R-IND$'$ *and* R-IND$''$ *is also a successful adversary on Query Privacy.*

The proof of Lemma 2 is analogous to that of Lemma 1. We omit it.

Corollary 1 (R-IND $\iff$ D-IND $\wedge$ Q-IND)**.** *Result Privacy is equivalent to both Data Privacy and Query Privacy (at the same time).*

3.5 Generalized Security Notions for Data Outsourcing Schemes

In this section, we generalize the security notions introduced in Section 3.4 to make them applicable to a wide variety of practical schemes. Protocols that—for the sake of efficiency—base decisions on the content of the queried data are bound to leak information about it [27, 26, 29] or are only secure for a limited number of queries [62].

Therefore, we first introduce bounds for the number of oracle calls. A special case is a bound of 1 that renders the notions non-adaptive.

Second, we define "leakage relations" R_d and R_q. Challenges the adversary can choose are subject to equivalence under these relations. This way, one

can explicitly rule out specific distinction advantages. To model the leakage of the length of a data set for example, one would define $R_d \subset \Delta^2$ as the set of all data set pairs with equal length.

Third, we explicitly model the issuing of queries independently of handing out the results. This allows us to capture security notions where the adversary can alter the state of the database, but can not see the immediate result (e.g. he can only observe the result of the last issued query).

Goh [55] introduces restricting parameters into his security notion as well. They allow for a bound on the running time, the advantage, and the number of oracle calls. Our work in this section can be seen as a generalization of his concept.

In Section 3.6 we showcase case studies that are direct applications of our generalized notions. We only give the security definitions here and defer discussion to the following section.

Security Game 10 ($\mathsf{IND\text{-}CDA}^{\mathcal{A},R_d}_{(\mathsf{Gen},\mathsf{Enc})}(k)$)**.**

1. The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$ and a random bit $b \leftarrow \{0,1\}$.
2. The adversary $\mathcal{A}$ is given input 1^k and oracle access to $\mathsf{Enc}(\cdot, K)$.
3. $\mathcal{A}$ outputs two data sets d_0 and d_1 to the experiment. The choice of d_0 and d_1 is restricted to data set pairs that are equivalent with regard to equivalence relation $R_d \subseteq \Delta^2$, i.e. $(d_0, d_1) \in R_d$.
4. $\mathcal{A}$ is given $\mathsf{Enc}(m_b, K)$.
5. $\mathcal{A}$ outputs a guess b' for b.

The result of the experiment is 1 if $b' = b$ and 0 else.

Definition 24 (Static Security)**.** An outsourcing scheme $(\mathsf{Gen}, \mathsf{Enc})$ has *indistinguishable encryptions under chosen-data-attacks (*$\mathsf{IND\text{-}CDA}$*)* or *static security* with respect to R_d, if

$$\forall \mathcal{A}, c \in \mathbb{N}\, \exists k \in \mathbb{N} : |\Pr[\mathsf{IND\text{-}CDA}^{\mathcal{A},R_d}_{(\mathsf{Gen},\mathsf{Enc})} = 1]| \leq \frac{1}{2} + k^{-c}$$

Security Game 11 ($\mathsf{D\text{-}IND}^{\mathcal{A},R_d,n_1,n_2,n_3}_{(\mathsf{Gen},\mathsf{Enc},\mathsf{Q})}(k)$)**.**

1. The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$.
2. $\mathcal{A}$ receives access to an oracle for $\mathsf{view}^{\pi}_{\mathfrak{S}}(\mathsf{Enc}(\cdot, K))$, and continues to have access to it. $\mathcal{A}$ is only allowed to query the oracle for a total number of n_1 times.

3. $\mathcal{A}$ outputs two data sets d_0 and d_1 to the experiment. The choice of d_0 and d_1 is restricted to pairs of data sets that are equivalent with regard to equivalence relation $R_d \subseteq \Delta^2$, i.e. $(d_0, d_1) \in R_d$.

4. The experiment draws a random bit $b \leftarrow \{0, 1\}$.

5. Challenge oracle: $\mathcal{A}$ is given access to an oracle for $\mathsf{view}_{\mathfrak{S}}^{\pi_\cdot}(\mathsf{Enc}(d_b, K))$, and continues to have access to it. $\mathcal{A}$ may call the challenge oracle for a total number of n_2 times.

6. Run oracle: $\mathcal{A}$ is given access to an oracle $\mathrm{run}_{\mathfrak{S}}^{\pi_\cdot}(\mathsf{Enc}(d_b, K))$. The run oracle executes queries just as the view oracle does, but has no output. $\mathcal{A}$ is allowed to call the run oracle for a total number of n_3 times.

7. $\mathcal{A}$ outputs a guess b' for b.

The result of the experiment is 1 if $b' = b$ and 0 else.

Security Game 12 ($\mathsf{Q\text{-}IND}^{\mathcal{A},R_q,n_1,n_2,n_3}_{(\mathsf{Gen},\mathsf{Enc},\mathsf{Q})}(k)$)**.**

1. The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$.

2. $\mathcal{A}$ receives access to an oracle for $\mathsf{view}_{\mathfrak{S}}^{\pi_\cdot}(\mathsf{Enc}(\cdot, K))$, and continues to have access to it. $\mathcal{A}$ is only allowed to query the oracle for a total number of n_1 times.

3. $\mathcal{A}$ outputs two queries q_0 and q_1 to the experiment. The choice of q_0 and q_1 is restricted to query pairs that are equivalent with regard to equivalence relation $R_q \subseteq \Pi^2$, i.e. $(q_0, q_1) \in R_q$.

4. The experiment draws a random bit $b \leftarrow \{0, 1\}$.

5. Challenge oracle: $\mathcal{A}$ is given access to an oracle for $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_b}}(\mathsf{Enc}(\cdot, K))$. $\mathcal{A}$ may call the challenge oracle for a total number of n_2 times.

6. Run oracle: $\mathcal{A}$ is given access to an oracle $\mathrm{run}_{\mathfrak{S}}^{\pi_b}(\mathsf{Enc}(\cdot, K))$, and continues to have access to it. The run oracle executes queries just as the view oracle does, but has no output. $\mathcal{A}$ is allowed to call the run oracle for a total number of n_3 times.

7. $\mathcal{A}$ outputs a guess b' for b.

The result of the experiment is 1 if $b' = b$ and 0 else.

Definition 25 (n_1, n_2, n_3-Data Privacy)**.** An outsourcing scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})$ has n_1, n_2, n_3-*Data Privacy* with respect to R_d, if

$$\forall \mathcal{A}, c \in \mathbb{N} \exists k \in \mathbb{N} : |\Pr[\mathsf{D\text{-}IND}^{\mathcal{A},R_d,n_1,n_2,n_3}_{(\mathsf{Gen},\mathsf{Enc},\mathsf{Q})}(k) = 1]| \leq \frac{1}{2} + k^{-c}$$

Definition 26 (n_1, n_2, n_3-Query Privacy)**.** An outsourcing scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})$ has n_1, n_2, n_3-*Query Privacy* with respect to R_q, if

$$\forall \mathcal{A}, c \in \mathbb{N} \exists k \in \mathbb{N} : |\Pr[\mathsf{Q\text{-}IND}^{\mathcal{A}, R_q, n_1, n_2, n_3}_{(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})}(k) = 1]| \leq \frac{1}{2} + k^{-c}$$

Security Game 13 ($\mathsf{R\text{-}IND}^{\mathcal{A}, R_d, R_q, n_1, n_2, n_3}_{(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})}(k)$)**.**

1. The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$.
2. $\mathcal{A}$ receives access to an oracle for $\mathsf{view}^{\pi_\cdot}_{\mathfrak{S}}(\mathsf{Enc}(\cdot, K))$, and continues to have access to it. $\mathcal{A}$ is only allowed to query $\mathsf{view}^{\pi_\cdot}_{\mathfrak{S}}(\mathsf{Enc}(\cdot, K))$ for a total number of n_1 times.
3. $\mathcal{A}$ outputs two data-set-query pairs (d_0, q_0) and (d_1, q_1) to the experiment. The choice of d_0, d_1, q_0, and q_1 is restricted to $(d_0, d_1) \in R_d$ and $(q_0, q_1) \in R_q$.
4. The experiment draws a random bit $b \leftarrow \{0, 1\}$.
5. Challenge: The experiment runs the protocol π_{q_b} on $\mathsf{Enc}(d_b, K)$ and outputs $\mathsf{view}^{\pi_{q_b}}_{\mathfrak{S}}(\mathsf{Enc}(d_b, K))$ to the adversary.
6. $\mathcal{A}$ receives oracle access to $\mathrm{view}^{\pi_\cdot}_{\mathfrak{S}}(\mathsf{Enc}(m_b, K))$ and $\mathrm{view}^{\pi_{q_b}}_{\mathfrak{S}}(\mathsf{Enc}(\cdot, K))$. He is only allowed to call the oracles a total number of n_2, respectively n_3, times.
7. $\mathcal{A}$ outputs a guess b' for b.

The result of the experiment is 1 if $b' = b$ and 0 else.

Definition 27 (n_1, n_2, n_3-Result Privacy)**.** An outsourcing scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})$ has n_1, n_2, n_3-*Result Privacy* with respect to R_d and R_q, if

$$\forall \mathcal{A}, c \in \mathbb{N} \exists k \in \mathbb{N} : |\Pr[\mathsf{R\text{-}IND}^{\mathcal{A}, R_d, R_q}_{(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})}(k) = 1]| \leq \frac{1}{2} + k^{-c}$$

3.6 Case Studies

In this section we review security notions from the literature and examine how they fit into our framework. To that end, we translate these notions to our formalisms.

3.6.1 Private Information Retrieval

We give a definition of the original (single-server) Computational Private Information Retrieval (cPIR) [79] notion using our conventions.

Definition 28 (Private Information Retrieval)**.** A queryable outsourcing scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \{\pi\})$ exhibits *Computational Single-Server Private Information Retrieval* (PIR) when two conditions hold for any $n \in \mathbb{N}$, any security parameter $k \in \mathbb{N}$, and any data set d over $\Sigma = \{0,1\}^n$:

1. Correctness: $\forall i \in \{0, \ldots, n-1\} : \pi_i^{\mathfrak{C}}(d) = d[i]$.
2. Privacy: $\forall c \in \mathbb{N}, i, j \in \{0, \ldots, n-1\}, \forall \mathcal{A} \exists K \in \mathbb{N}$ such that $\forall k > K$

$$|\Pr[\mathcal{A}(\mathsf{view}_{\mathfrak{S}}^{\pi_i}(\mathsf{Enc}(d, K))) = 1] - \Pr[\mathcal{A}(\mathsf{view}_{\mathfrak{S}}^{\pi_j}(\mathsf{Enc}(d, K))) = 1]| < \frac{1}{\max(k, n)^c}.$$

Theorem 8. *Private Information Retrieval is equivalent to Query Privacy.*

In our proof we implicitly assume a queryable database outsourcing scheme that has, for each bit in the database, a query that retrieves it, i.e. we exclude schemes that store unretrievable information in the database. This is not a restriction, as one can easily construct a "non-redundant" scheme from one that stores unretrievable information.

Proof. For this proof, let the domain of all data sets be $\Delta = \{0,1\}^*$. Fix a security parameter $k \in \mathbb{N}$. W.l.o.g., assume any queryable outsourcing scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Q})$ with a protocol π_i that outputs the $i+1$th bit of the database for all $i \in \{0, \ldots, n-1\}$, where n is the length of the data set $d \in \Delta$. We prove the theorem in two steps.

PIR $\implies$ Q-IND. Assume any efficient adversary $\mathcal{A}$ who is successful in Q-IND with a non-negligible advantage. We show that there are $i, j \in \{0, \ldots, n-1\}$ and an efficient algorithm $\mathcal{A}'$ such that they violate the Privacy condition of Definition 28.

Construct $\mathcal{A}'$ as follows: Simulate experiment Q-IND to obtain π_i, π_j from $\mathcal{A}$. i and j are the required indices. Now relay the input $\mathsf{view}_{\mathfrak{S}}^{\pi_b}(\mathsf{Enc}(d, K))$ (for $b \in \{i, j\}$) to $\mathcal{A}$. Output $\mathcal{A}$'s guess b'.

Q-IND $\implies$ PIR. Assume any $i, j \in \{0, \ldots, n-1\}$ and any efficient algorithm $\mathcal{A}'$ such that $\mathcal{A}'$ violates the Privacy condition of Definition 28 at indices i and j. We construct an efficient adversary $\mathcal{A}$ that has a non-negligible advantage in Q-IND: Output π_i and π_j as the challenge queries. Output $b' = \mathcal{A}(\mathsf{view}_{\mathfrak{S}}^{\pi_b}(\mathsf{Enc}(d, K)))$ (for $b \in \{i, j\}$). (For any adversary with a success probability $< \frac{1}{2}$ there is an adversary with a success probability $> \frac{1}{2}$, easily obtained by flipping its guess.) ■

3.6.2 Searchable Encryption using Directed Acyclic Word Graphs

Haynberg et al. [62] construct an efficient scheme for symmetric searchable encryption and a corresponding security notion. Their construction is not based on a dictionary, but realizes incremental pattern matching.

Security Game 14 ($\mathrm{PrivK}^{\mathrm{cppa}}_{\mathcal{A},\mathsf{Enc}}(k)$).

1. The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$ and a random bit $b \leftarrow \{0,1\}$.
2. The adversary receives oracle access to $\mathsf{view}^{\pi_\cdot}_{\mathfrak{S}}(\mathsf{Enc}(\cdot, K))$, and continues to have access to it.
3. $\mathcal{A}$ outputs two plaintexts m_0 and m_1 of the same length to the experiment.
4. $\mathcal{A}$ outputs a number of queries $x_0, \ldots, x_q$ and an integer $i \in \{0, \ldots, q\}$ to the experiment.
5. The queries $x_0, \ldots, x_q$ are evaluated in that order.
6. $\mathcal{A}$ is given the view on the challenge ciphertext to query i: $\mathsf{view}^{\pi_{x_i}}_{\mathfrak{S}}(\mathsf{Enc}(m_b, K))$.
7. $\mathcal{A}$ submits a guess b' for b.

The result of the experiment is 1 if $b' = b$ and 0 else.

The security notion can be directly instantiated in our framework.

Theorem 9. $\mathrm{PrivK}^{\mathrm{cppa}}$ *is equivalent to Data Privacy.*

Proof. $\mathrm{PrivK}^{\mathrm{cppa}}_{\mathcal{A},\mathsf{Enc}}(k)$ can be directly instantiated from $\mathsf{D\text{-}IND}^{\mathcal{A},R_d,n_1,n_2,n_3}_{(\mathsf{Gen},\mathsf{Enc},\mathsf{Q})}(k)$ with parameters $R_d = \{d_0, d_1\}$, $|d_0| = |d_1|$, $n_1 = \mathsf{poly}(k)$, $n_2 = 1$, and $n_3 = \mathsf{poly}(k)$.

■

3.6.3 Indistinguishability under Independent Column Permutations

Huber et al. [67] present a provably-secure database outsourcing scheme which is es efficient as the underlying database. In their notion the encryptions of two databases must be indistinguishable if they can be transformed into each other by permuting attribute values within columns. Since our generalized notions allow for defining a restriction on the plaintexts, this database-specific security notion also fits into our framework.

Definition 29 (Independent Column Permutation [67])**.** Let Φ be the set of database functions $\mathfrak{p} : \Delta \to \Delta$ such that each $\mathfrak{p} \in \Phi$ permutes the entries within each column of a database. We call $\mathfrak{p}$ an *independent column permutation.*

Security Game 15 ($\mathsf{IND\text{-}ICP}^{\mathcal{A}}_{\mathsf{Gen},\mathsf{Enc},\Phi}(k)$)**.**

1. The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$.
2. $\mathcal{A}$ outputs one plaintext m and an independent column permutation $\mathfrak{p} \in \Phi$ to the experiment.
3. The experiment chooses $m_0 := m$ and $m_1 := \mathfrak{p}(m)$ draws $b \leftarrow \{0,1\}$ uniformly at random.
4. $\mathcal{A}$ is given $\mathsf{Enc}(m_b, K)$.
5. $\mathcal{A}$ submits a guess b' for b.

Theorem 10. IND-ICP *is equivalent to static security.*

Proof. IND-ICP is a direct instantiation of $\mathsf{IND\text{-}CDA}^{\mathcal{A},R_{\mathrm{ICP}}}_{(\mathsf{Gen},\mathsf{Enc})}(k)$, where $R_{\mathrm{ICP}} \subset \Delta^2$ is the set of all pairs of databases that are independent column permutations of each other: We set $\Delta = DB$ and $R_{\mathrm{ICP}} = \Delta_{/\Phi(\Delta)}$. Then, each adversary that has a non-negligible advantage in $\mathsf{IND\text{-}ICP}^{\mathcal{A}}_{\mathsf{Gen},\mathsf{Enc},\Phi}(k)$ can efficiently be reduced to an adversary that has non negligible advantage in $\mathsf{IND\text{-}CDA}^{\mathcal{A},R_{\mathrm{ICP}}}_{(\mathsf{Gen},\mathsf{Enc})}(k)$ and vice versa. The reduction from IND-ICP to $\mathsf{IND\text{-}CDA}^{\mathcal{A},R_{\mathrm{ICP}}}_{(\mathsf{Gen},\mathsf{Enc})}(k)$ sets $m_0 := m$ and $m_1 := \mathfrak{p}(m)$, while the reduction from $\mathsf{IND\text{-}CDA}^{\mathcal{A},R_{\mathrm{ICP}}}_{(\mathsf{Gen},\mathsf{Enc})}(k)$ to IND-ICP determines $\mathfrak{p}$ with $\mathfrak{p}(m_0) = m_1$. ■

3.6.4 Semantic Security Against Adaptive Chosen Keyword Attacks

Goh [55] presents a security notion for index-based searchable encryption schemes. We give a translation into our formalisms. Trapdoors are translated to a view oracle.

Security Game 16 ($\mathsf{IND\text{-}CKA}^{\mathcal{A}}_{(\mathsf{Gen},\mathsf{Enc},\mathsf{Q})}$)**.**

1. The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$ and a random bit $b \leftarrow \{0,1\}$.
2. The adversary $\mathcal{A}$ is given input 1^k and access to an oracle $\mathrm{view}^{\pi}_{\mathfrak{S}}(\mathsf{Enc}(\cdot, K))$.

3. $\mathcal{A}$ outputs two plaintexts m_0 and m_1 of the same length to the experiment. The adversary must not have queried the oracle for words that are only in one of the two plaintexts.

4. $\mathcal{A}$ is given $\mathsf{Enc}_K(m_b)$ and access to an oracle $\mathrm{view}_{\mathfrak{S}}^{\pi_\cdot}(\mathsf{Enc}(m_b, K))$.

5. $\mathcal{A}$ submits a guess b' for b.

The result of the experiment is 1 if $b' = b$ and 0 else.

IND-CKA is a weaker form of Data Privacy. Were the adversary not restricted in choosing queries (Line 3. in Security Game 16), the notions would be equivalent. As in the case of Curtmola et al.'s notion (Section 3.6.5), we prove the relation of Goh's notion to our model without considering this restriction. We point out that one could easily further generalize our security notions to include this restriction by additionally forcing queries to adhere to a relation to the data set. However, we decided against this, as the applications of such restrictions seem limited.

Theorem 11. IND-CKA *implies static security.*

The proof is a straightforward reduction and we omit it here.

Theorem 12. *Database privacy implies* IND-CKA.

Proof. Assume an adversary $\mathcal{A}$ who who has a non-negligible advantage in experiment IND-CKA. We give an efficient reduction that breaks database privacy. The reduction forwards the two challenge data sets from $\mathcal{A}$ to experiment D-IND. All queries $\mathcal{A}$ executes using via the view oracle (before and after submitting the two challenge databases) can easily be simulated by using the oracles from the D-IND experiment (this is because the set of valid queries in D-IND is a superset of the valid queries in IND-CKA). The reduction forwards the adversary's guess b' to the experiment. ■

3.6.5 Adaptive Security for SSE

Curtmola et al.'s notion *adaptive indistinguishability security for SSE* [36] is also a security notion for symmetric searchable encryption based on indices.

Security Game 17 ($\mathsf{Ind}^*_{\mathbf{SSE},\mathcal{A},(\mathsf{Gen},\mathsf{Enc},\mathsf{Q})}(k)$ [36]).

1. The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$ and a random bit $b \leftarrow \{0,1\}$.

2. The adversary $\mathcal{A}$ is given input 1^k and outputs two plaintexts d_0 and d_1 of the same length to the experiment.

3. $\mathcal{A}$ is given $\mathsf{Enc}_K(d_b)$.

4. $\mathcal{A}$ can output polynomialy many pairs of queries (q_0, q_1) and is given $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_b}}(\mathsf{Enc}(d_b, K))$.

5. $\mathcal{A}$ submits a guess b' for b.

The result of the experiment is 1 if $b' = b$ and 0 else.

Note that in Curtmola et al.'s notion, Query Privacy and Data Privacy are intertwined. Thus, it can not be directly instantiated from our framework. We instead show how his notion relates to our notions. $\mathsf{Ind}^*_{\mathrm{SSE}}$ also requires the adversary to only choose plaintexts and corresponding search queries which have the same views for both databases. This is a very strict restriction which we do not take into consideration here. We instead focus on the more general notion.

Theorem 13 ($\mathsf{Q\text{-}IND} \wedge \mathsf{D\text{-}IND} \implies \mathsf{Ind}^*_{\mathrm{SSE}}$)**.** *If a queryable outsourcing scheme has Query Privacy, Data Privacy implies* Ind^*_{SSE}.

We prove the statement using a game-hopping technique. The argument is very similar to the proof of Theorem 7. We only sketch the technique.

Proof. Modify $\mathsf{Ind}^*_{\mathrm{SSE}}$ so that the challenge oracle always returns $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_1}}(\mathsf{Enc}(d, K))$, independently of b. An adversary who can distinguish between the two games also breaks query privacy. Database privacy implies the modified $\mathsf{Ind}^*_{\mathrm{SSE}}$ experiment. The reduction is straightforward. If the adversary against $\mathsf{Ind}^*_{\mathrm{SSE}}$ requests a view $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_b}}(\mathsf{Enc}(d_b, K))$ for input (q_0, q_1), the reduction forwards q_1 to the oracle of Database Privacy and returns $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_1}}(\mathsf{Enc}(d_b, K))$. ■

Theorem 14 ($\mathsf{Ind}^*_{\mathrm{SSE}} \implies \mathsf{D\text{-}IND}$)**.** Ind^*_{SSE} *implies database privacy.*

Proof. We give a reduction from database privacy to $\mathsf{Ind}^*_{\mathrm{SSE}}$. It is straightforward. When the adversary against database privacy requests $\mathsf{view}_{\mathfrak{S}}^{\pi_q}(\mathsf{Enc}(d_b, K))$, the reduction fixes $q_0 := q_1 := q$ and returns $\mathsf{view}_{\mathfrak{S}}^{\pi_{q_b}}(\mathsf{Enc}(d_b, K))$. ■

3.7 A Security Model for Cryptographic File Systems

In this chapter, we expand our model for secure data outsourcing to apply to encrypted cloud file systems which covers both security and integrity in a non-adaptive as well as in an adaptive setting. We first give security definitions in the chosen plaintext attack scenario and then show how to extend them to the chosen ciphertext attack scenario. Further, we show that chosen ciphertext security for file systems can be achieved by combining plaintext security and integrity.

3.7.1 Basic Definitions

We give a formal definition of an encrypted file system. In general, a file system needs four algorithms: one for initializing the file system (like setting up data structures), one for updating the file system (like adding and removing files), one for decrypting the file system and one for generating the cryptographic keys. The file system, and all algorithms which interact with it, are stateful.

Definition 30 (Encrypted File System)**.** Let $\mathbb{F}$ be the set of plaintext file systems, $\mathbb{C}$ the set of ciphertext file systems, and $\mathbb{S}$ the set of client states. Let $\mathbb{K} = \{0,1\}^k$ be the set of keys and $\mathcal{E} = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ be a symmetric encryption scheme. An encrypted file system $\mathcal{C}$ is a tuple $\mathcal{C} := (\mathsf{Gen}, \mathsf{Init}, \mathsf{Update}, \mathsf{Dec}, \mathcal{E})$ with

- $\mathsf{Gen} : \{1\}^k \to \mathbb{K}$ is a PPT algorithm which generates a key K.
- $\mathsf{Init} : \mathbb{K} \to \mathbb{C} \times \mathbb{S}$ is a PPT algorithm which takes the key K and initializes an empty ciphertext file system C, and the client state s.
- $\mathsf{Update} : \mathbb{K} \times \mathbb{C} \times \mathbb{F} \times \mathbb{S} \to (\{\perp\} \cup \mathbb{C}) \times \mathbb{S}$ is a PPT algorithm used to update the file system. It is given the key K, an old ciphertext file system C, a new plaintext file system F and a client state s. It outputs $\perp$ if the decryption of C fails, else a new ciphertext file system C', and a new client state s'.
- $\mathsf{Dec} : \mathbb{K} \times \mathbb{C} \times \mathbb{S} \to (\{\perp\} \cup \mathbb{F}) \times \mathbb{S}$ is a PPT algorithm which is given a key K, a ciphertext file system C, and the client state s and outputs $\perp$ if the decryption fails, else the decrypted file system F, and a new client state s.

3.7.2 Modelling Non-Adaptive Security

Traditionally, security against non-adaptive adversaries requires that an adversary cannot gain any information from a scheme which they did not observe or interact with before. In the case of file systems however, we additionally require that the adversary could have interacted with other encrypted file systems using the same key. We allow the adversary to create an arbitrary but constant number of file systems, which are available before and after he chooses the challenge. Also, we do not require the client state to be kept secret. We allow the challenges to be restricted by a relation R_d (e. g. both file systems must store the same amount of data). This means that from looking at a freshly encrypted file system, an attacker cannot deduce any information even if he observed modifications on different file systems using the same key. In particular, this requires the file system to introduce measures to be secure under key reuse (e. g. a user encrypting two different file systems with the same password). We call this security notion *indistinguishability under non-adaptive chosen file system attacks* (IND-naCFA).

Security Game 18 ($\mathsf{IND\text{-}naCFA}^{\mathcal{A},R_d}(k)$). • The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$ and a random bit $b \leftarrow \{0,1\}$.

- The adversary is given oracle access to $\mathsf{Init}(K)$. The j-th query returns a new ciphertext file system (C_j, s_j) using the same key and the following oracle to interact with it:
  - $(C'_j, s'_j) \leftarrow \mathsf{Update}_j(K, C_j, \cdot, s_j)$. The game sets $(C_j, s_j) := (C'_j, s'_j)$.

  The number of Init queries is bounded by an adversary-chosen constant q_{Init}.
- The adversary outputs two file systems F^0 and F^1 with $(F^0, F^1) \in R_d$.
- The experiment generates $(C, s) \leftarrow \mathsf{Init}(K)$.
- The experiment computes $(C', s') \leftarrow \mathsf{Update}(K, C, F^b, s)$.
- $\mathcal{A}$ is given (C, s) and (C', s').
- $\mathcal{A}$ submits a guess b' for b.

The result of the experiment is 1 if $b' = b$, and 0 else.

Definition 31 (Nonadaptive Security)**.** A file system is $\mathsf{IND\text{-}naCFA}$ secure, if

$$\forall \mathcal{A}, c \in \mathbb{N} \exists k_0 \in \mathbb{N} \forall k > k_0 : |\Pr[\mathsf{IND\text{-}naCFA}^{\mathcal{A},R_d}(k) = 1]| \leq \frac{1}{2} + k^{-c}$$

3.7.3 Modelling Adaptive Security

Intuitively, while $\mathsf{IND\text{-}naCFA}$ models security of a file system directly after creation, adaptive security models the security of a file system later in its life. To achieve this, we allow the adversary to choose a file system as challenge with which he already interacted. We then require that he cannot distinguish which of two modifications he chose is performed. Again, we allow to restrict the adversary's choice of challenge by a relation R_d. We call this security notion *indistinguishability under adaptive chosen file system attacks* and it is a direct extension of $\mathsf{IND\text{-}naCFA}$.

Security Game 19 ($\mathsf{IND\text{-}aCFA}^{\mathcal{A},R_d}(k)$). • The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$ and a random bit $b \leftarrow \{0,1\}$.

- The adversary is given oracle access to $\mathsf{Init}(K)$, which on the j-th query initializes $F_j = \perp$ (empty file system), returns a new ciphertext file system (C_j, s_j) using the same key and an oracle to interact with it.
  - $(C'_j, s'_j) \leftarrow \mathsf{Update}_j(K, C_j, \cdot, s_j)$. The game remembers the most recent input F_j and sets $(C_j, s_j) := (C'_j, s'_j)$.

The number of Init queries is bounded by a constant q_{Init} chosen by the adversary.

- The adversary outputs j and two file systems F^0, F^1 with $(F_j, F^0, F^1) \in R_d$.
- The experiment computes $(C'_j, s'_j) \leftarrow \mathsf{Update}_j(K, C_j, F^b, s_j)$ and passes (C'_j, s'_j) to the adversary.
- $\mathcal{A}$ submits a guess b' for b.

The result of the experiment is 1 if $b' = b$ and 0 else.

Definition 32 (Adaptive security)**.** A file system is $\mathsf{IND\text{-}aCFA}$ secure, if

$$\forall \mathcal{A}, c \in \mathbb{N} \exists k_0 \in \mathbb{N} \forall k > k_0 : |\Pr[\mathsf{IND\text{-}aCFA}^{\mathcal{A},R_d}(k) = 1]| \leq \frac{1}{2} + k^{-c}$$

3.7.4 Modelling Integrity

To provide integrity, a cloud file system must ensure that a malicious server cannot alter the file system in any way, even though the server can observe every modification made to this file system and to other file systems using the same key. In particular, a server must not be able to provide the client with old states of the file system. This results in the following security model, which we call *integrity of file systems*.

Security Game 20 ($\mathsf{INT\text{-}FS}^{\mathcal{A}}(k)$).

- The experiment chooses a key $K \leftarrow \mathsf{Gen}(1^k)$.
- The adversary is given oracle access to $\mathsf{Init}(K)$. The j-th query returns a new ciphertext file system (C_j, s_j) using the same key and the following oracles to interact with it:
  - $(C'_j, s'_j) \leftarrow \mathsf{Update}_j(K, C_j, \cdot, s_j)$. The game sets $(C_j, s_j) := (C'_j, s'_j)$.
  - $(F, s'_j) \leftarrow \mathsf{Dec}_j(K, \cdot, s_j)$. The game sets $s_j := s'_j$ for the next query.

  The number of Init queries is bounded by an adversary-chosen constant q_{Init}.

The result of the experiment is 1 if for any of the decryption oracle queries $\mathsf{Dec}_j(K, C', s_j) \neq \perp$, $C_j \neq C'$.

Definition 33 (Integrity)**.** A file system is $\mathsf{INT\text{-}FS}$ secure, if

$$\forall \mathcal{A}, c \in \mathbb{N} \exists k_0 \in \mathbb{N} \forall k > k_0 : |\Pr[\mathsf{INT\text{-}FS}^{\mathcal{A}}(k) = 1]| \leq k^{-c}$$

3.7.5 Security Against Chosen Ciphertext Attacks

Like IND-CCA security is an extension of IND-CPA security, we extend IND-naCFA to IND-naCCFA and IND-aCFA to IND-aCCFA. The security games are identical to their chosen plaintext counterparts, except that Init returns an additional decryption oracle $\mathsf{Dec}_j(K, \cdot, s_j)$, which is modeled like in the INT-FS game.

For basic encryption schemes, ciphertext security (IND-CCA) can be achieved by combining plaintext security (IND-CPA) with integrity (INT-CTXT) [14]. We show that this is also true for file systems within our security framework.

Lemma 3. *A file system* $\mathcal{F} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{Update}, \mathsf{Dec})$ *is* IND-(n)aCCFA *secure, if it is* IND-(n)aCFA *and* INT-FS *secure.*

Proof. Assume a modified version of IND-(n)aCCFA, where the Dec_j oracle only works for the most recent output of the corresponding Update_j oracle, or (if Update has not been called yet) for the output of Init. For all other queries, it returns $\perp$. We call this modified game IND-(n)aCCFA′. It is straightforward to reduce an adversary against IND-(n)aCCFA′ to an adversary against IND-(n)aCFA by remembering the most recent Update_j queries and answer the decryption query accordingly. We now show that any adversary with non-negligible success probability against IND-(n)aCCFA also has a non-negligible success probability against IND-(n)aCCFA′. Assume towards a contradiction an adversary A with a non-negligible different success probability in playing IND-(n)aCCFA and IND-(n)aCCFA′. We transform this adversary into an adversary A' against INT-FS. When A requests access to the Init oracle, A' forwards the calls to the Init oracle provided by INT-FS, returning (C_j, s_j) and the Update_j oracle. When A requests access to the Dec_j oracle, A' calls the Dec_j oracle provided by INT-FS, but ignores the response and implements the behavior described for the IND-(n)aCCFA′ game by remembering the most recent Update_j query. For IND-aCCFA′, the challenge (C'_j, s'_j) is generated by another call to the Update_j oracle. For IND-naCCFA′, the challenge (C, s, C', s') is generated by calling Init and then using the freshly returned Update_j oracle. Since the success probability of A is non-negligibly different for IND-(n)aCCFA and IND-(n)aCCFA′, and the only difference in the games is the behavior of Dec_j oracle queries that are not the most recent output of the Update_j oracle but decrypts successfully, we know such a query must happen with non-negligible probability. This query can be used directly to win the INT-FS game.

■

3.8 Case Study: CryFS

CryFS is an overlay file system which can be mounted to a virtual folder. It is available by default in most major Linux distribution. Everything the user

stores in this virtual folder is encrypted in the background. The ciphertexts are stored on the hard disk (through the underlying file system) and can be picked up by third party synchronisation clients like Dropbox and uploaded to a cloud storage. This allows for a flexible use on top of any file system or cloud storage provider. In contrast to many other encrypted file systems, we do hide file contents as well as metadata like file sizes, file permissions and directory structure. We achieve this by splitting all file system data into same-sized blocks. These blocks are then individually encrypted using an authenticated cipher. Using a specifically tailored data structure, we ensure that all file operations are still fast and we induce little space overhead, even though all files are segmented into small blocks (see Section 3.8.1). To prevent malicious storage providers from violating the integrity of the file system, we introduce additional measures to prevent rollback, deletion and re-introduction of deleted blocks (see Section 3.8.3). We point out that we decided against using hash trees to protect integrity: The primary reason behind this decision is our goal to support concurrent access to the file system. Hash trees induce changes from the affected block up to the root node, thus increasing the chance of edit conflicts. The second reason for avoiding hash trees are performance considerations. Although hash trees have only logarithmic overhead in the size of the file system, any non-constant overhead is prohibitive for file systems with many frequent changes in many small files. Even though these integrity protections are only fully effective when the file system is used by a single user, CryFS is designed to work well with multiple users. As most other encrypted file systems, CryFS uses two keys: a file system key for encrypting the file system blocks and a master key for encrypting the filesystem key. This makes it easy to change passwords for example.

3.8.1 Data Structures, Blocks and Files

As already mentioned, CryFS does not encrypt files individually. Rather, it splits every file into same-sized blocks, which are then encrypted. A tree data structure then associates blocks to files and files to directories. We base our construction on Dielissen et al.'s work on left-perfect binary trees [42] and generalize their definition to *left-max-data trees*.

The main idea for this data structure is that all nodes in the tree are as far left as possible. The actual binary file data is always stored in the left-most leaves of the file system tree and in-order. All leaves in the tree are at the same depth, and with exception of the right-most one, store exactly the same amount of data. This allows to represent arbitrary file sizes. Internal nodes contain only pointers to other blocks. If the block size is chosen appropriately (and thus the number of available pointers in each block), even large files can be represented by a tree with little depth. Every block is identified by a unique id, which is randomly chosen each time a block is created. See

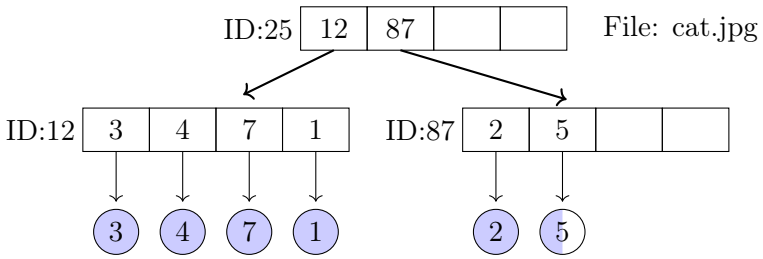


Figure 3.4: The tree for an exemplary file "cat.jpg". Each tree node is one same-sized block in CryFS. The actual file data is stored in the leaves, whereas inner nodes store only pointers. For determining the file size, one only has to descend into the right-most branch of the tree and examine how much data is stored in the right-most leaf. Since all leaves are at the same depth and only the right-most elements are allowed to contain a less-than-maximum amount of data, this descend suffices to know how many blocks the file contains and thus the total file size.

Figure 3.4 for an example file represented as a left-max-data tree. This structure leads to very efficient algorithms for file system access. When trying to read a certain position in a file, one only needs to compute the respective block number from the total number of blocks in this file and the fixed block data size. Also, small changes to a file are particularly efficient: only a small block has to be changed (and synchronized to the cloud) not the whole file. Increasing the file size is described in Algorithm 1, decreasing is similar. Since only the right-most leaf can contain a less-than-maximum amount of data, determining the file size can also be achieved without reading all blocks by determining the amount of data in the right-most leaf. In our reference implementation with 32kb blocks and 16 byte block ids, this data structure induces a space overhead of roughly 0.05% for inner nodes plus an additive overhead of at most one leaf node's size if the right-most leaf is not full.

3.8.2 Directory Structure

Directories in CryFS are basically files themselves. Directories, however, do not store binary data but store a list of the directory's entries—i. e. pointers to the root block of files and directories. To allow for an efficient listing of all directory entries without having to descend into all individual file trees, we store the name of each entry, as well as all file metadata (like permission bits) along with the corresponding pointer in the directory structure. This layout allows for fast modifications of the directory structure. Moving a large directory only requires re-encrypting both the old and the new parent directory. See Figure 3.5 for an example of a file system tree with one directory and one file.

Algorithm 1 Grow an existing tree by one leaf

```
function GrowTree(treeRoot, newBlock)
    ℓ ← LowestNonFullInnerNode(treeRoot)
    if ℓ = ⊥ then /* All nodes are full. We need to add a level. */
        ℓ ← NewInnerNode() /* Create a new root block */
        ℓ.AddChild(treeRoot)
        treeRoot ← ℓ
    end if
    while depth(ℓ) < depth(leaves) − 1 do
        n ← NewInnerNode()
        ℓ.AddChild(n)
        ℓ ← n
    end while
    ℓ.AddChild(NewLeaf(newBlock))
    return treeRoot
end function
```

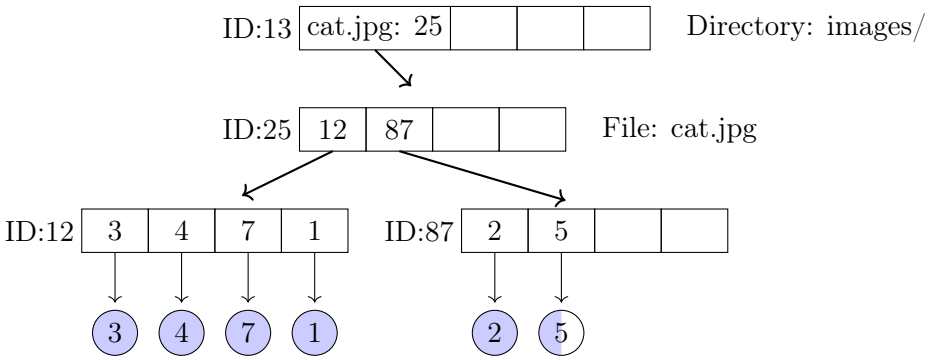


Figure 3.5: The file "cat.jpg" is contained in a directory "images". To list all files of a directory efficiently, the name of each file is included with the respective pointer. As it is the case with files, once the number of entries in a directory exceeds the size of one block, the directory itself is represented as a tree.

3.8.3 Encryption and Integrity

Encryption is on the block level—i. e. each block is encrypted individually. This allows for good performance because blocks can be encrypted in parallel. We use a cipher with an authenticated operation mode (e. g. AES-GCM) to prevent an adversary from altering the content of the blocks themselves. However, this is not yet sufficient to protect the integrity of the file system as a whole, since the connections between different blocks are not protected. An adversary can still try to reorder blocks, replace newer blocks with older versions, delete or re-add already deleted blocks.

We use a number of different mechanisms to prevent these attacks. First, we store the block ID in the header of the block, where it is integrity-protected by the authenticated encryption scheme. This ensures that an attacker cannot assign a different ID to a block (by changing the name of the file storing the block) and therefore prevents reordering. To prevent an attacker from replacing a block with a previous version of the same block, a block also stores a version counter in its header. Clients store a local list of all known blocks with a flag whether the block still exists, and their corresponding version numbers and check that it does not decrease. This list is also used to prevent an attacker from deleting or re-adding already deleted blocks without the client noticing. Additionally, the clients remember the master-key-encrypted file system key to prevent an adversary from replacing the whole file system including the key. In Section 3.8.4, we formally prove that this approach achieves the desired security goals. See Algorithm 1 – Algorithm 5 for a description of relevant file system algorithms in pseudo-code.

3.8.4 Proving the Security of CryFS

In this section, we prove the adaptive and non-adaptive security of CryFS and show that it also provides integrity. Further, we show that CryFS also achieves ciphertext indistinguishability. We first give a formal description of CryFS. To simplify notation, we represent the tree structure of CryFS as a set of node blocks.

Definition 34 (CryFS)**.** Let $\mathbb{I}$ be the space of block IDs, $\mathbb{I} \times \{0,1\}^n$ the set of plaintext blocks, and $\mathbb{I} \times \{0,1\}^m$ the set of ciphertext blocks. $\mathsf{CryFS}^{\mathcal{E}_1,\mathcal{E}_2}$ is an *encrypted file system* $(\mathsf{Gen}, \mathsf{Init}, \mathsf{Update}, \mathsf{Dec})$ with $\mathcal{E}_1 = (\mathsf{Gen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ and $\mathcal{E}_2 = (\mathsf{Gen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$. The client state $\mathbb{S} \subseteq 2^{\mathbb{I}\times\mathbb{N}\times\{0,1\}} \times \{0,1\}^{k'}$ stores a set of all known blocks with their id $i \in \mathbb{I}$, current version $v \in \mathbb{N}$ and a flag whether the block still exists (1) or was deleted in the past (0). The state also stores $c_{\text{fs}} \in \{0,1\}^{k'}$, an encrypted version of the file system key. For the sake of clarity of the exposition, we first define intermediate functions:

- $\mathsf{Repr} : \mathbb{F} \to 2^{\mathbb{I}\times\{0,1\}^n}$: Takes a plaintext file system and generates its representation as a set of plaintext blocks.

Algorithm 2 Returns a new block with a unique id and the version number set to 0

```
function CreateBlock
    i ← GenerateUniqueID()
    return (i, i||0)
end function
```

Algorithm 3 Add a file or a folder tree to a directory

```
function AddToDirectory(directory, newEntry)
    if RightmostLeaf(directory).IsFull() then
        GrowTree(directory, CreateBlock())
    end if
    RightmostLeaf(directory).AddData(newEntry)
end function
```

Algorithm 4 Creates a tree data structure from a file and returns the root node

```
function CreateFile(file)
    D := (d_0, ..., d_n) ← SplitData(file)
    t ← CreateBlock()
    t.AddData(d_0)
    for all other d_i ∈ D do
        b_i ← CreateBlock()
        b_i.AddData(d_i)
        t ← GrowTree(t, b_i)
    end for
    return t
end function
```

Algorithm 5 Creates the data structure for a complete file system

```
function CreateFileSystem(sourceFileSystemRoot)
    rootBlock ← CreateBlock()
    for all Directories dir in sourceFileSystemRoot do
        rootBlock.AddToDirectory(CreateFileSystem(dir))
    end for
    for all Files file in sourcFileSystemRoot do
        rootblock.AddToDirectory(CreateFile(file))
    end for
    return rootBlock
end function
```

- $\mathsf{EncBlock} : \mathbb{K} \times (\mathbb{I} \times \{0,1\}^n) \times \mathbb{N} \to (\mathbb{I} \times \{0,1\}^m)$: Takes a key K_{fs}, a plaintext block (i, b) and a version number $v \in \mathbb{N}$. Prepends block ID and version number to the data and encrypts it. Outputs (i, c) with $c := \mathsf{Enc}_2(K_{\text{fs}}, i||v||b)$.
- $\mathsf{DecBlock} : \mathbb{K} \times (\mathbb{I} \times \{0,1\}^m) \to \{\perp\} \cup [(\mathbb{I} \times \{0,1\}^m) \times \mathbb{N}]$: Takes a key K_{fs} and a ciphertext block (i, c). Decrypts it to $i'||v||b := \mathsf{Dec}_2(K_{\text{fs}}, c)$. If decryption fails or $i \neq i'$, returns $\perp$. Otherwise, returns the plaintext block (i, b) and the version number v.

Now we define the functions forming an encrypted file system.

- $\mathsf{Gen}(1^k) \mapsto (K_{\text{master}})$: Uses Gen_1 to generate a master key K_{master}.
- $\mathsf{Init}(K_{\text{master}}) \mapsto (C, s)$: Takes K_{master} and generates $K_{\text{fs}} \leftarrow \mathsf{Gen}_2(1^k)$. Encrypts it with the master key to $c_{\text{fs}} = \mathsf{Enc}_1(K_{\text{master}}, K_{\text{fs}})$. Computes $B := \mathsf{Repr}(F) = \{(i_0, b_0), \ldots, (i_n, b_n)\}$, a set of blocks representing an empty file system F.
  Sets $C := (c_{\text{fs}}, \mathsf{EncBlock}(K_{\text{fs}}, (i_0, b_0), 0), \ldots, \mathsf{EncBlock}(K_{\text{fs}}, (i_n, b_n), 0))$ and $s := (\{(i_0, 0, 1), \ldots, (i_n, 0, 1)\}, c_{\text{fs}})$ and outputs (C, s).
- $\mathsf{Dec}(K_{\text{master}}, C, s) \mapsto (F, s)$: Reads c_{fs} from C and compares it with the c_{fs} stored in s. If they differ, returns $\perp$. Otherwise, decrypts it to $K_{\text{fs}} := \mathsf{Dec}_1(K_{\text{master}}, c_{\text{fs}})$. Then, computes
  $D := \{((i', b), v) \mid ((i', b), v) = \mathsf{DecBlock}(K_{\text{fs}}, (i, c)), (i, c) \in C\}$. Outputs $\perp$ in the following cases:
  - Dec_1 fails to decrypt c_{fs} (wrong key or an integrity violation).
  - $\mathsf{DecBlock}$ fails to decrypt c (wrong key, an integrity violation, or $i \neq i'$).
  - There is an $((i, b), v) \in D$ for which there is no $(i, v', 1) \in s$
  - There is an $((i, b), v) \in D$ for which there is an $(i, v', 1) \in s$ with $v < v'$
  - There is an $(i, v, 1) \in s$ for which there is no $((i, b), v') \in D$

  Otherwise, computes the plaintext file system $F := \mathsf{Repr}^{-1}(\{(i_0, b_0), \ldots, (i_n, b_n)\})$ and outputs (F, s). The client state is not changed.
- $\mathsf{Update}(K_{\text{master}}, C, F', s) \mapsto (C', s')$: Decrypts the old file system state to $F := \mathsf{Dec}(K_{\text{master}}, C, s)$. Then, reads c_{fs} from C and decrypts it to K_{fs}. If either decryption fails, returns $\perp$. Initializes $s' := s$. Compares $\mathsf{Repr}(F)$ and $\mathsf{Repr}(F')$ and does the following:
  - For each block $(i, b) \notin \mathsf{Repr}(F)$, $(i, b') \in \mathsf{Repr}(F')$:
    - If $(i, v, 0) \in s$, replace it in s' with $(i, v + 1, 1)$. Else, add $(i, 0, 1)$ to s'

* Note: if $(i, v, 1) \in s$, Dec would have failed above.

– For each block $(i, b) \in \mathsf{Repr}(F)$, $(i, b') \in \mathsf{Repr}(F'), b \neq b'$
  * Replace $(i, v, 1)$ in s' with $(i, v' + 1, 1)$, where v' is the version number returned from $\mathsf{DecBlock}$ on decryption.
  * Note: $(i, v, 1) \in s \wedge v' \geq v$, otherwise Dec would have failed above.

– For each block $(i, b) \in \mathsf{Repr}(F)$, $(i, b') \notin \mathsf{Repr}(F')$
  * Replace $(i, v, 1)$ with $(i, v, 0)$ in s'.
  * Note: $(i, v, 1) \in s$ otherwise Dec would have failed above.

Then, encrypts F' using $\mathsf{EncBlock}$ with updated version numbers and outputs the new ciphertext file system C' (including c_{fs}), and the modified state s'.

We now show that CryFS exhibits non-adaptive security according to Definition 31. We set R_d to restrict the challenge file systems to be representable using the same number of blocks. Formally, this means

$$R_d = \{(F^0, F^1) \in \mathbb{F} \times \mathbb{F} : |\mathsf{Repr}(F^0)| = |\mathsf{Repr}(F^1)|\}$$

Theorem 15 (Nonadaptive Security of CryFS)**.**
$\mathsf{CryFS}^{\mathcal{E}_1,\mathcal{E}_2} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{Update}, \mathsf{Dec})$ *is* $\mathsf{IND\text{-}naCFA}$ *secure, if* $\mathcal{E}_1 = (\mathsf{Gen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ *and* $\mathcal{E}_2 = (\mathsf{Gen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$ *are* $\mathsf{IND\text{-}CPA}$ *secure encryption schemes.*

Proof. We prove the claim by reduction using two steps. First, we modify $\mathsf{IND\text{-}naCFA}$ to $\mathsf{IND\text{-}naCFA'}$ such that when the adversary gets the challenge $(C, s), (C', s')$, it does not contain an encryption of K_{fs}, but an encryption of 0s instead. We prove that an adversary which has a different advantage in $\mathsf{IND\text{-}naCFA}$ and $\mathsf{IND\text{-}naCFA'}$ can be used to break the $\mathsf{IND\text{-}CPA}$ security of $\mathcal{E}_1$. Second, we give a reduction from $\mathsf{IND\text{-}naCFA'}$ to the $\mathsf{IND\text{-}CPA}$ security of $\mathcal{E}_2$.

Consider the following modification to $\mathsf{IND\text{-}naCFA}$: When the adversary expects the challenge (C', s'), replace the encrypted file system key $\mathsf{Enc}_1(K_{\text{master}}, K_{\text{fs}})$ in state and ciphertext with $\mathsf{Enc}_1(K_{\text{master}}, 0)$. We call this modified game $\mathsf{IND\text{-}naCFA'}$. Now, assume towards a contradiction an adversary A with a probability of success p against $\mathsf{IND\text{-}naCFA}$ and p' against $\mathsf{IND\text{-}naCFA'}$, where $p = p' + d$ for a positive non-negligible d. This adversary can be used to construct an adversary B with a non-negligible advantage of $\frac{d}{2}$ against the $\mathsf{IND\text{-}CPA}$ security of $\mathcal{E}_1$. The reduction works as follows: The $\mathsf{IND\text{-}CPA}$ game draws $K_{\text{master}} \leftarrow \mathsf{Gen}_1(1^k)$ and a random bit b. When A uses the Init oracle, B generates $K'_{\text{fs}} \leftarrow \mathsf{Gen}_2(1^k)$ and (C_j, s_j) using the algorithms described in Definition 34 and uses the encryption oracle of $\mathsf{IND\text{-}CPA}$ to generate c'_{fs} as an encryption of K'_{fs}. Since B knows K'_{fs} it can also build

the Update_j oracle. When the adversary outputs F^0, F^1, B generates another independent $K_{\text{fs}} \leftarrow \mathsf{Gen}_2(1^k)$ and passes 0 and K_{fs} as challenge to the IND-CPA game. The game returns c_{fs}. When $b = 0$, this is an encryption of 0. When $b = 1$, this is an encryption of K_{fs}. B then draws a random bit a, and knowing K_{fs}, can build the challenge (C, s) and (C', s') as an encryption of F^a. It replaces the encrypted file system key in C, s, C' and s' with the c_{fs} and returns the result to A. If A outputs a, A wins and B outputs 1 to the IND-CPA game. If A loses, B outputs 0. For $b = 0$, this was a perfect simulation of the IND-naCFA′ game. B has success probability $\Pr[a \not\leftarrow A \mid b = 0] = 1 - p'$. For $b = 1$, this was a perfect simulation of the IND-naCFA game. B has success probability $\Pr[a \leftarrow A \mid b = 1] = p = p' + d$. Together, B has success probability $\Pr[b \leftarrow B] = \frac{1}{2}(1-p') + \frac{1}{2}(p'+d) = \frac{1}{2} + \frac{d}{2}$. Since d is non-negligible, B has a non-negligible advantage in the IND-CPA game which is a contradiction.

Now, assume towards another contradiction that A' is a successful attacker on IND-naCFA′. We transform A' into a successful attacker B' on IND-CPA security of $\mathcal{E}_2$: The game draws K_{fs} and a random bit b. B' draws $K_{\text{master}} \leftarrow \mathsf{Gen}_1(1^k)$. When A' uses Init, B' generates a new K'_{fs}, encrypts it with K_{master}, and creates an empty ciphertext file system. Knowing K_{master}, the Update_j oracle can be implemented easily.

Upon receiving challenges F^0 and F^1 from A', B' first generates an empty file system and encrypts it to (C, s) using the encryption oracle and prepending $c'_{\text{fs}} = \mathsf{Enc}_1(K_{\text{master}}, 0)$. Then, B' updates it with F^0 and F^1 respectively, and uses the LR-oracle provided by IND-CPA successively for each pair of blocks in $\mathsf{Repr}(F^0)$ and $\mathsf{Repr}(F^1)$. This is possible, since we require $(F^0, F^1) \in R_d$ (i. e. both have the same number of blocks), Repr can be implemented to choose the same block ids for F^0 and F^1 and all blocks are of the same size. B' remembers all encrypted blocks returned by the oracle, prepends c'_{fs} to get C' and passes it to A' together with a generated file system state s' in which all block ids in have version number 1.

This is a correct simulation of the IND-naCFA′ game. When A' submits a guess for b, B' forwards it and thus inherits its success probability. This is a contradiction to the assumption that $\mathcal{E}_2$ is IND-CPA-secure. ■

Theorem 16 shows that CryFS is also adaptively secure according to Definition 32. Since block IDs are public and CryFS only re-encrypts blocks for which the plaintext changed (for performance reasons), we set R_d to restrict both challenge file systems add, delete or modify blocks with the same block IDs. Theorem 17 shows that CryFS exhibits integrity according to Definition 33.

Theorem 16 (Adaptive Security of CryFS)**.**
$\mathsf{CryFS}^{\mathcal{E}_1,\mathcal{E}_2} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{Update}, \mathsf{Dec})$ *is* IND-aCFA *secure, if*
$\mathcal{E}_1 = (\mathsf{Gen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ *and* $\mathcal{E}_2 = (\mathsf{Gen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$ *are* IND-CPA *secure encryption schemes.*

Proof. Consider the following modification to IND-naCFA: When the adversary queries Init or the Update_j oracles or expects output (C, s), instead of getting $\mathsf{Enc}_1(K_{\text{master}}, K_{\text{fs}})$ they instead get $\mathsf{Enc}_1(K_{\text{master}}, 0)$. Now, assume towards a contradiction an adversary A with a success probability of p against IND-aCFA and a success probability of p' against IND-aCFA′, where $p = p' + d$ for a positive non-negligible d. This adversary can be used to construct an adversary B with a non-negligible advantage of $\frac{d}{2}$ which breaks the IND-CPA security of $\mathcal{E}_1$. The game draws $K_{\text{master}} \leftarrow \mathsf{Gen}_1(1^k)$ and a random bit b. When A uses the Init oracle, B generates a new file system key $K_{\text{fs}} \leftarrow \mathsf{Gen}_2(1^k)$ and uses the LR oracle of the IND-CPA game to get c_{fs} as either an encryption of 0 or of K_{fs}, depending on the value of b. Then it generates a new empty file system (C_j, s_j) but replaces the encryption of K_{fs} with c_{fs}. A expects access to an Update_j oracle which can be built by using K_{fs} to decrypt and encrypt blocks. Again, B replaces all encryptions of K_{fs} with c_{fs}. When the adversary outputs j, F^0, F^1, B draws a random bit a. It uses Update_j to build the challenge (C', s') as an encryption of F^a. If A outputs a (A wins), B outputs 1. If A loses, B outputs 0. For $b = 0$, this was a perfect simulation of the IND-aCFA′ game. B has success probability $\Pr[a \not\leftarrow A \mid b = 0] = 1 - p'$. For $b = 1$, this was a perfect simulation of the IND-aCFA game. B has success probability $\Pr[a \leftarrow A \mid b = 1] = p = p' + d$. Together, B has success probability $\Pr[b \leftarrow B] = \frac{1}{2}(1-p') + \frac{1}{2}(p'+d) = \frac{1}{2} + \frac{d}{2}$. Since d is non-negligible, B has a non-negligible advantage against IND-CPA.

Now, assume towards another contradiction that A' is a successful attacker on IND-aCFA′. We transform A' into a successful attacker B' on the IND-CPA security of $\mathcal{E}_2$. Intuitively, B' selects a random file system created by A' and uses A' to break its security. Since the number of file systems is a fixed constant, this only reduces the success probability by a constant amount. The reduction works as follows. The game draws K_{fs} and a random bit b. B' draws $\mathsf{K}_{\text{master}} \leftarrow \mathsf{Gen}_1(1^k)$ and draws a random $j^* \leftarrow \{1, \ldots, q_{\mathsf{Init}}\}$. When A' uses Init for the j-th time and $j \neq j^*$, B' generates a new K'_{fs}, encrypts it with K_{master}, and creates an empty ciphertext file system. Knowing K_{master}, the Update_j oracle can easily be implemented. In every output, $\mathsf{Enc}_1(K_{\text{master}}, K_{\text{fs}})$ is replaced with an encryption of 0. When A' uses Init for the j^*-th time, B' generates a new empty file system by using the encryption oracle of the IND-CPA experiment to encrypt all blocks. Again, B' prepends $\mathsf{Enc}_1(K_{\text{master}}, 0)$. B' also saves the current plaintext file system F_j (which is empty). If A' uses their access to the Update_j-oracle, B' updates the saved plaintext according to the input to the oracle. It uses the encryption oracle to encrypt added or modified blocks and exchanges them in the saved ciphertext. B' updates the saved file system F_j and the state s_j. Upon receiving challenge j, F^0 and F^1 from A', B' updates the corresponding plaintext F_j for both F^0 and F^1 respectively and passes the added and modified blocks of $\mathsf{Repr}(F^0)$ and $\mathsf{Repr}(F^1)$ (when compared to $\mathsf{Repr}(F_j)$) to the LR oracle of the IND-CPA experiment. It now has an encryption of either

the modified blocks in F^0 or in F^1. Since it is required that $(F_j, F^0, F^1) \in R_d$ (i. e. they add, remove, and modify blocks with the same ID), B' knows which ciphertext blocks it has to add, remove and replace with their new versions in order to generate the correct ciphertext file system, even though it does not know which change was selected by the experiment. B' prepends $\mathsf{Enc}_1(K_{\text{master}}, 0)$ to the generated ciphertext and passes it to A' along with the updated state. This is a correct simulation of the IND-aCFA′ game. When A' submits a guess for b, B' forwards it to the game and thus inherits its success probability. This is a contradiction to the assumption that $\mathcal{E}_2$ is IND-CPA secure.

■

Theorem 17 (Integrity of CryFS)**.** $\mathsf{CryFS}^{\mathcal{E}_1,\mathcal{E}_2} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{Update}, \mathsf{Dec})$ *is* INT-FS *secure, if* $\mathcal{E}_1$ *is* IND-CPA *and* $\mathcal{E}_2$ *is* INT-CTXT *secure.*

Proof. Again, we first change INT-FS to INT-FS′ by replacing $\mathsf{Enc}_1(K_{\text{master}}, K_{\text{fs}})$ with $\mathsf{Enc}_1(K_{\text{master}}, 0)$ in the output of all oracles. Assume towards a contradiction that an adversary A with success probability of p against INT-FS and success probability of p' against INT-FS′ exists (where $p = p' + d$ for a positive non-negligible d). This adversary can be used to construct an adversary B with an advantage of $\frac{d}{2}$ against the IND-CPA security of $\mathcal{E}_1$ by using the following reduction: When A uses Init, B generates $K_{\text{fs}} \leftarrow \mathsf{Gen}_2(1^k)$ and uses the LR oracle of the IND-CPA game to get c_{fs} as either an encryption of 0 or of K_{fs}. It generates (C_j, s_j) using K_{fs} but replaces the encrypted file system key with c_{fs}. B builds the Update_j and Dec_j oracles using K_{fs} to decrypt and encrypt blocks. Each output contains c_{fs} instead of the encrypted file system key. When Dec_j is used, B checks whether decryption was successful for $C \neq C'$, i. e. whether A was successful. If A was successful, B outputs 1, otherwise it outputs 0. If $b = 0$, this was a perfect simulation of the INT-FS′ game. B has success probability $\Pr[0 \leftarrow B \mid b = 0] = 1 - p'$. If $b = 1$, this was a perfect simulation of the INT-FS game. B has success probability $\Pr[1 \leftarrow B \mid b = 1] = p = p' + d$ Together, B has success probability $\Pr[b \leftarrow B] = \frac{1}{2}(1 - p') + \frac{1}{2}(p' + d) = \frac{1}{2} + \frac{d}{2}$. Since d is non-negligible, this is a non-negligible advantage for B against IND-CPA.

Now, assume towards another contradiction that A' is a successful attacker on INT-FS′. We give a reduction which transforms A' into a successful attacker B' on INT-CTXT. The game draws $K_{\text{fs}} \leftarrow \mathsf{Gen}_2(1^k)$ and B' draws $K_{\text{master}} \leftarrow \mathsf{Gen}_1(1^k)$. B' draws a random $j^* \leftarrow \{1, \ldots, q_{\mathsf{Init}}\}$. When A' uses Init for the j-th time with $j \neq j^*$, B' generates a new independent K'_{fs} and creates a new ciphertext file system with this key. Knowing K'_{fs}, implementing Update_j and Dec_j oracles is straightforward. In every output, $\mathsf{Enc}_1(K_{\text{master}}, K_{\text{fs}})$ gets replaced by $\mathsf{Enc}_1(K_{\text{master}}, 0)$. When A' uses Init for the j^*-th time, B' creates a new empty file system but uses the encryption oracle provided by INT-CTXT to encrypt all blocks. It also builds Update_j

and Dec_j but uses the decryption and encryption oracles of the INT-CTXT game to decrypt and encrypt. Instead of prepending $\mathsf{Enc}_1(K_{\text{master}}, K_{\text{fs}})$, which B' does not know, it prepends $\mathsf{Enc}_1(K_{\text{master}}, 0)$.

Since A' is successful, there is an oracle query $\mathsf{Dec}_j(K, C', s_j)$ which decrypts successfully with $C_j \neq C'$. With non-negligible probability $\frac{1}{q_{\mathsf{Init}}}$, this happens for $j = j^*$, where B' implemented Init using the INT-CTXT experiment. C_j and C' have the same set of block IDs, otherwise $\mathsf{Dec}_j(K_{\text{master}}, C', s_j) = \perp$. So there has to be a block in C' which is different from the corresponding block in C_j, i. e. $\exists i, c_i, c_i' : (i, c_i) \in C_j, (i, c_i') \in C', c_i \neq c_i'$. This block c_i' was input to the decryption oracle of the IND-CTXT game when decrypting C'. We argue that c_i' wins the INT-CTXT game. First note that INT-FS′ decrypts with $c_{\text{fs}} = \mathsf{Enc}_1(K_{\text{master}}, K_{\text{fs}})$ from the state, not with the $c_{\text{fs}}' = \mathsf{Enc}_1(K_{\text{master}}, 0)$ passed to the adversary. Therefore c_i' decrypts successfully with the key from the INT-CTXT experiment. We now have to argue that c_i' was never output by the INT-CTXT encryption oracle. Recall that this oracle is only used for encrypting the output of the j-th query of the Init oracle and for the outputs of the Update_j oracle. Since C' decrypts successfully, we know that the plaintext $((i', b_i'), v_i') := \mathsf{DecBlock}(K, (i, c_i'))$ has ID $i = i'$ and a version number $v_i' \geq v_i^s$ where v_i^s is the version number in the state. All previous Update'_j oracle queries for this block ID encrypted a block with version number $v_i \leq v_i^s$, and $v_i = v_i^s$ only for c_i where we know $c_i' \neq c_i$. So we know c_i' was not output of the Update_j oracle. If (i, c_i') was in the j-th output of the Init oracle, then $v_i' = 0$. In this case, either block i was never modified, which is a contradiction to $c_i \neq c_i'$, or block i was modified, which means $v_i^s > 0$ and therefore is a contradiction to successful decryption. Taking everything into account, we know that c_i' was never output by the INT-CTXT encryption oracle and thus wins the game. This is a contradiction to the assumed security of $\mathcal{E}_2$.

■

Lastly, we show that CryFS can also be secure against chosen ciphertext attacks.

Theorem 18 (Chosen Ciphertext Attacks)**.**
$\mathsf{CryFS}^{\mathcal{E}_1,\mathcal{E}_2} = (\mathsf{Gen}, \mathsf{Init}, \mathsf{Update}, \mathsf{Dec})$ *is* IND-naCCFA *and* IND-aCCFA *secure, if* $\mathcal{E}_1 = (\mathsf{Gen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ *is an* IND-CPA *and* $\mathcal{E}_2 = (\mathsf{Gen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$ *an* IND-CPA *and* INT-CTXT *secure encryption scheme.*

Proof. This follows directly from Theorem 15, Theorem 17 and Lemma 3.

■

Chapter 4

Modeling Computer Networks

The following chapter is based on joint work with Dirk Achenbach and Jörn Müller-Quade. Parts of the included content have already been presented in the following works:

- Dirk Achenbach, Jörn Müller-Quade, Jochen Rill: *Universally Composable Firewall Architectures Using Trusted Hardware.* BalkanCryptSec 2014 [3].
- Dirk Achenbach, Jörn Müller-Quade, Jochen Rill: *Synchronous Universally Composable Computer Networks.* BalkanCryptSec 2015 [2].
- Dirk Achenbach: *On Provable Security for Complex Systems.* PhD Thesis 2016 [1]

4.1 Introduction

Information Technology (IT) systems are at the heart of most automated systems today. Not only cars and airplanes rely on networked computer systems, but also factories, water supply plants, and nuclear facilities. At the same time, IT systems have never faced more serious threats—national and foreign secret services, criminal organizations, and even corporate hacker groups threaten the integrity and availability of services society relies on.

Especially the protection of computer networks against attackers from the Internet is a crucial component of any security strategy. Network firewalls in particular seem to be attractive targets for attackers. Documents that were leaked by Edward Snowden in 2013, reveal that the National Security Agency has the capability to install backdoors in a number of commercial firewalls: JETPLOW, HALLUXWATER, FEEDTROUGH, GOURMETTROUGH, and SOUFFLETROUGH [70]. Likewise, the Chinese government is alleged to force Huawai to install backdoors in routers and other network gear [17].

As is the case with data outsourcing schemes (as presented in Chapter 3), however, formal cryptographic security models are rarely used when designing

such networks. Instead, such systems are empirically tested for known security weaknesses.

In this work, we therefore address the research question whether established formal security models can be used to analyze real computer networks. Our goal is to provide a "recipe" for modeling and analyzing networks. We attempt to achieve this in two steps. First, we start by using the basic UC framework to model specific and very basic network structures, namely architectures using multiple firewalls. Using this model, we investigate on a conceptual level whether the threat of a compromised firewall can be mitigated by using a combination of multiple firewalls in combination with trusted hardware. Since the basic UC framework lacks the ability to model time, availability guarantees can not be expressed within the resulting model. Therefore, as a second step, we use the extension by Katz et al. [75] to make this possible. The adapted model also allows the expression of general network structures.

4.2 Related Work

To the best of our knowledge, we are first to explicitly model computer networks in the UC framework.

Network Firewalls The purpose, aim and function of network firewalls is widely understood and agreed upon, e.g. [15, 90, 68]. The informational RFC 2979 [50] defines characteristics of firewalls. Since there is no globally agreed-on standard for what constitutes good and bad network packets however, there is also no complete specification of the function of a firewall.

The security of firewalls or systems of firewalls has mainly been studied under two aspects. One concern is verifying the correctness and soundness of rule sets. Gouda et al. [60] develop a formal model for verification of distributed rule sets based on trees. They are able to check whether the firewall system accepts or denies a specific class of packets. Ingols et al. [69] check for conflicting or otherwise problematic rules with the aid of Binary Decision Diagrams.

We are not aware of any works that consider the firewall as being malicious.

Formal Analysis of Computer Networks While network security is considered a practical field, formal methods have also been applied to computer networks. Research generally concentrates on modeling attacks and vulnerabilities [69] and on generating verification policies [65, 80]. While such approaches help in configuring specific network components and in mitigating threats, they do not have the advantages of cryptographic security models.

UC Proofs for Practical Protocols The UC framework is the quasi state-of-the-art framework for proving the security of cryptographic *building block* protocols like Commitments [22] or Oblivious Transfer [86]. Because it has a composition theorem, it is argued that more complex protocols can then be composed of these components. However, the UC framework has also been used to prove the security of more complex schemes, such as TLS [51], OAuth [28], disk encryption [37], and robust combinations of network firewalls [3]. Our contribution falls in line with this work. We investigate composing large computer networks.

Secure Hardware Katz [73] uses tamper-proof hardware to realize universally composable multi-party computation. He assumes tamper-proof tokens that can be programmed with an arbitrary program. Such a programmed token is then handed to another party in the protocol, which may then interact with the token. Goldwasser et al. [57] introduce the computational paradigm of one-time programs, i.e. programs that can only be run once, on one input. Of course, such programs cannot exist purely in software, as software can be copied indefinitely. Goldwasser et al. introduce "one-time-memory devices" to create a compiler for one-time programs.

Robust Combiners The idea of mistrusting the implementation of a secure functionality has been studied in the scope of *robust combiners*. A (k, n)-robust combiner combines n candidate implementations of the secure functionality $\mathcal{P}$ in a way that the overall security still holds if at least k implementations are secure [63].

The notion of a robust combiner is not suited for our purposes. The very definition of robust combiners requires a specific and fixed functionality $\mathcal{P}$. However, in the case of firewalls, it is unclear what this functionality precisely is. Informally speaking, the functionality of a network firewall is "filtering all malicious packets". It is not possible to formalize this functionality in a protocol or a program, since, in general, it is not possible to decide whether an arbitrary packet is malicious or not.

Byzantine Fault Tolerance The problem of handling malicious actors is reminiscent of byzantine fault tolerance. However, we use a very different communication structure. In the original Byzantine Generals Problem [81], every party can communicate with every other party. This leads to specific bounds concerning the number of trusted parties needed to achieve fault tolerance. Even when signing messages is possible, in order to allow for m corrupted parties, one still needs at least $(2m + 1)$ trusted parties and $(m + 1)$ rounds of communication. In our case, we do not allow the parties to communicate freely, but only according to the specific structure of the network—we do not allow firewalls to exchange messages with each other.

Thus, the results which byzantine fault tolerance research provides are not applicable to our scenario.

4.3 Modeling Firewall Architectures

We assume a packet-switched local-area network (LAN) in which there are only uncompromised hosts. They are connected through a single uplink to the Internet, in which are potentially compromised hosts. To facilitate an easier discussion, we call machines in the LAN being "inside" and machines on the Internet being "outside". The "inside" is only connected to the "outside" through a firewall (network), whose job is to protect machines "inside" from machines "outside". For ease of exposition, we model communication in networks in one direction only (cf. Section 4.3).

The output of a firewall then depends on the packet $p \in P$ it gets as input (where P is the set of all possible packets) and its internal state $s \in S$.

After processing this information, the firewall then outputs a packet p' to one or multiple connected devices and updates its internal state (e.g. outputs a new internal state s'). The functionality of a firewall is defined formally in Definition 35.

Definition 35 (The functionality of an ideal firewall j $\mathrm{F}_{\mathrm{fw}_j}$)**.**

$$\mathrm{F}_{\mathrm{fw}_j} : P \times S \to (P \cup \bot) \times S$$

$$\mathrm{F}_{\mathrm{fw}_j}(p, s) = \begin{cases} (p', s') & \text{if packet is accepted,} \\ (\bot, s') & \text{otherwise.} \end{cases}$$

We stress that our definition of a firewall functionality is universal. Because it is stateful—it receives its previous state as input, may use it for its computation and outputs an updated state—a firewall may base its output on an arbitrarily long history of incoming and outgoing packets. It may, for example, reconstruct a TCP session. Further, we do not restrict how its output depends on its input. A firewall might for example receive a packet, store it, transform it, and output it much later. Because the functionality processes whole packets including their payload, our definition covers the whole network protocol stack (e.g. Ethernet, IP, TCP, HTTP, HTML).

To simplify the exposition, we only discuss unidirectional networks. The easiest approach for extending the model to bidirectional communication would be using an independent instance of $\mathcal{F}_{\mathsf{ideal}}$ for each direction and deducing the security of the composed system by using the Composition Theorem. However, this approach would require the protocols for each direction to be independent of each other and not have a joint state. Actual firewall solutions base their decisions on all observed packets (not only those in one direction), however. Thus, the security of the bidirectional extensions of the architectures we discuss has to be proven manually.

We only discuss the security of a single atomic building block for complex firewall architectures. The Composition Theorem of the UC framework provides us with a strong guarantee for networks composed of several building blocks.

4.3.1 Adversarial Model

We assume an outside adversary who can statically corrupt exactly one firewall in the network. He gains full control over this firewall and can send and receive messages in its name (via a GSM link, for example). Because our constructions are symmetric, our corruption model is equivalent to an adaptive model.

4.3.2 Trusted Hardware

Firewalls are inherently complex computer systems, which, as we have discussed previously, should not be universally trusted. However, secure systems cannot be build without any trust. Similar to Trusted Platform Modules, which are being used as hardware-based security anchors for the boot process of modern computers, we envision a hardware-based security anchor for networks. Such a device would have to realize two very simple functionalities depending on the direction of the packet flow. In one direction its job is to compare packets that come in from different sources and decide whether to let them pass. In the other direction its job is to split incoming packets and distribute them to several firewalls. Because such a "packet comparator" offers only limited functionality (especially in comparison to a firewall), they could be manufactured easily, maybe even by the network owner himself. Also, it would be very hard to hide any backdoors or undocumented functionality in the device. Thirdly, because of its simple functionality, the device need not be able to download updates or even be freely programmable. We envision such a device to be realized as an Application-Specific Integrated Circuit (ASIC). In our security analysis, we assume that the specialized hardware we use cannot be compromised, i.e. is *trusted hardware*. In the following we will refer to the hardware as *trusted component*, in short tc.

The specific functionality of such a device depends on the particular network architecture, but it would always involve some kind of packet comparison. We express the notion of "packet equivalence" with a relation $\equiv$ that we assume to be defined appropriately.

A firewall may change the order of the packets it delivers. Some packets might need to be inspected more closely (Deep Packet Inspection), while others would just be waved through—take for example packages from a voice-over-IP (VoIP) connection. Therefore, it is not sufficient for the trusted hardware to compare packets one-by-one in the order they arrive.

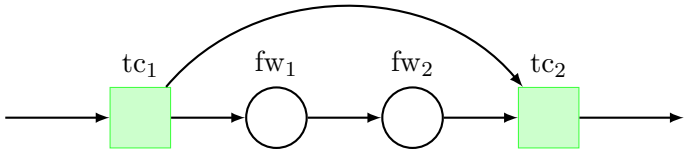


Figure 4.1: The serial concatenation of firewalls using secure hardware to compare packets. tc_2 compares whether "what goes in, comes out". tc_1 forwards the packet to the first firewall and second trusted hardware. The connecting arrows represent network cables in a "real" network.

4.4 Serial Concatenation of Two Firewalls

The first idea that comes to mind is to concatenate two firewalls and compare whether packets that exit the network originally were sent from the outside. This way, no firewall can "make up" packets. his concatenation of firewalls is not secure however, as we will show.

Figure 4.1 shows a graphical representation of the network architecture of the serial concatenation. fw_1, fw_2, tc_1 and tc_2 will be the parties in the corresponding UC protocol.

In order to proof the insecurity of this architecture, we have to model it within the UC framework. To this end, we first provide the protocol of the serial architecture and define the ideal network function.

Definition 36 (The protocol of the serial firewall architecture $\pi_{\mathsf{serial\text{-}2fw}}$)**.** The protocol the parties are following is defined as follows:

- tc_1: Upon receiving p from $\mathcal{Z}$: Call $\mathcal{F}_{\mathsf{net,serial\text{-}2fw}}(p)$.
- fw_k: Upon receiving p via $\mathcal{F}_{\mathsf{net,serial\text{-}2fw}}$: Calculate $\mathrm{F}_{\mathrm{fw}_k}(p, s) = (p', s')$. If $p' \neq \perp$, call $\mathcal{F}_{\mathsf{net,serial\text{-}2fw}}(p')$. Save the new internal state s'.
- tc_2: Upon receiving p from tc_1 via $\mathcal{F}_{\mathsf{net,serial\text{-}2fw}}$, store p in a local storage. Upon receiving p from fw_2 via $\mathcal{F}_{\mathsf{net,serial\text{-}2fw}}$, check whether there is an entry q in the local storage (with $p \equiv q$). If so, write p to the output tape and delete the entry.

Packets from outside the network always arrive at tc_1 first. Parties cannot communicate directly. Instead, we provide them with an ideal functionality for communication. This functionality ensures that parties can only communicate in a way that is fixed by the structure of the network. This is justified, since in a "real" network, components can also only communicate along the network cables.

We omit session IDs from all descriptions of functionalities and protocols. Different instances behave independently. We use the notion of "public

The ideal network function $\mathcal{F}_{\mathsf{net,serial\text{-}2fw}}$

Initialize two empty queues Q_1 and Q_2

- Upon receiving p from tc_1:
  - Push p into the Q_1 and Q_2.
  - Give p to the adversary.
- Upon instruction from the adversary, remove p from Q_1 and give it to fw_1.
- Upon instruction from the adversary, remove p from Q_2 and give it to tc_2.
- Upon receiving p from fw_1: Provide a public delayed output of p to fw_2.
- Upon receiving p from fw_2: Provide a public delayed output of p to tc_2.

Figure 4.2: The ideal network function representing the serial concatenation of firewalls with special hardware.

The ideal functionality of two firewalls $\mathcal{F}_{\text{ideal-2fw}}$

Let fw_k be the uncorrupted firewall.

- Initialize $s := \perp$.
- Upon receiving p from tc_1, compute $\text{F}_{\text{fw}_k}(p, s) = (p', s')$. If $p' \neq \perp$ make a public delayed output of p' to tc_2.

Figure 4.3: The ideal functionality of two firewalls.

delayed output", introduced by Canetti [21]. This means that a message is given to the adversary prior to delivery. The adversary then decides when (or whether) it is delivered.

The main idea for the ideal functionality is that any firewall architecture, regardless of the amount of different firewalls or their specific rule set, should behave as if the corrupted firewall was not there (see Figure 4.3).

Theorem 19. $\pi_{\text{serial-2fw}}$ *does* not *UC realize* $\mathcal{F}_{\text{ideal-2fw}}$ *in the* $\mathcal{F}_{\text{net,serial-2fw}}$*-hybrid model.*

The idea is that if fw_2 is corrupted, it could output a malicious packet just at the same time this packet arrives at tc_1 (sent by the environment). This would force tc_2 to output the packet, even though it was blocked by fw_1.

Proof. Let fw_2 be corrupted and fw_1 be honest. Let p be a packet that is blocked by fw_1. The environment inputs p to tc_1. This will cause tc_1 to call $\mathcal{F}_{\text{net,serial-2fw}}(p)$ where p will be saved in Q_2. Next, the adversary instructs $\mathcal{F}_{\text{net,serial-2fw}}$ to deliver p to tc_2 and uses fw_2 (which he controls) to call $\mathcal{F}_{\text{net,serial-2fw}}(p)$ and allows the public delayed output of p to be delivered to tc_2. tc_2 will now have two identical packets (one from tc_1 and one from fw_2) in its storage and will output p, even though p has been blocked by fw_1.

There is no simulator which can simulate this attack in the ideal model, since fw_1 will always block the packet in the ideal model and the output of fw_2 will not be considered. ■

4.5 Parallel Composition of Two Firewalls

The serial composition of two firewalls is not secure with regard to our envisioned ideal functionality. Better results can be achieved using parallel composition. The idea is that the trusted hardware only accepts a packet

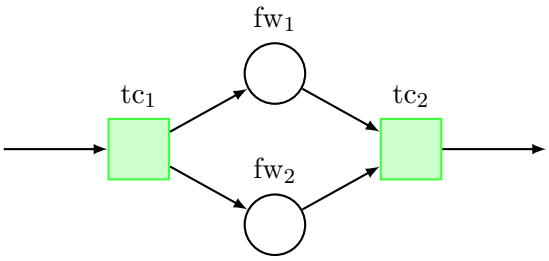


Figure 4.4: The parallel composition of two firewalls with trusted hardware. tc_2 only accepts packets that are output by both firewalls.

if both firewalls accept it. Figure 4.4 shows this composition. We will now discuss the security of this architecture.

The protocol of the parallel architecture is defined in Definition 37.

Definition 37 (Protocol of the parallel architecture $\pi_{\mathsf{parallel\text{-}2fw}}$)**.**

- tc_1: Upon receiving p from $\mathcal{Z}$: Call $\mathcal{F}_{\mathsf{net,parallel\text{-}2fw}}(p)$.
- fw_k: Upon receiving p via $\mathcal{F}_{\mathsf{net,parallel\text{-}2fw}}$: Compute $\mathrm{F}_{\mathrm{fw}_k}(p, s) = (p', s')$. If $p' \neq \perp$, call $\mathcal{F}_{\mathsf{net,parallel\text{-}2fw}}(p')$. Save the new internal state s'.
- tc_2: Upon receiving p from fw_k via $\mathcal{F}_{\mathsf{net,parallel\text{-}2fw}}$, check if there is an entry (j, q) with $k \neq j$ and $p \equiv q$ in the internal storage. If so, output p and remove both entries. Else, save (k, p).

The functionality describing the network structure is depicted in Figure 4.5.

We will compare the protocol from Definition 37 with an ideal functionality. The ideal functionality is the same as in the serial case, since the natural approach of defining ideal functionalities only uses the uncorrupted firewall, which again leads to the functionality in Figure 4.3. However, as in the serial case, the parallel architecture does not realize this functionality.

Theorem 20. $\pi_{\mathsf{parallel\text{-}2fw}}$ *does not UC realize* $\mathcal{F}_{\mathsf{ideal\text{-}2fw}}$ *in the* $\mathcal{F}_{\mathsf{net,parallel\text{-}2fw}}$*-hybrid model.*

We prove this by describing an attack which cannot be simulated.

Proof. Let, w.l.o.g., fw_1 be honest and fw_2 be corrupted. Also, let p_1 and p_2 be packets that are accepted by fw_1. The environment sends packets p_1 and p_2 to tc_1. The adversary triggers the delivery of both packets to fw_1. Both packets are accepted by fw_1 and forwarded to tc_2. Then, the adversary sends packets p_2 and p_1 from fw_2. Since both packets have been accepted and were sent to tc_2 previously (but in reverse order), tc_2 will output p_2

The ideal network function $\mathcal{F}_{\mathsf{net,parallel\text{-}2fw}}$

Initialise two empty queues Q_1 and Q_2 for fw_1 and fw_2 respectively.

- Upon receiving p from tc_1: Push p into Q_1 and Q_2
- Upon receiving instruction from the adversary: Fetch the next packet p Q_1 and deliver it to fw_1
- Upon receiving instruction from the adversary: Fetch the next packet p Q_2 and deliver it to fw_2
- Upon receiving p from fw_k: Provide a public delayed output of p to tc_2.

Figure 4.5: The ideal network function representing the parallel concatenation of firewalls with trusted hardware.

and p_1—in this order. Thus, the adversary was able to reverse the order of packets. Since the adversary is not allowed to influence the order of packets in the ideal model, there exists no simulator which can simulate this attack. ■

The Internet Protocol explicitly does not give any guarantees about the ordering of packets, since the correct order is encoded in the packet. The packet itself, however, can not be altered by the adversary. Thus, we modify our ideal functionality and explicitly grant the attacker the ability to reorder the outgoing packet stream. The new ideal functionality is described in Figure 4.6.

Theorem 21. $\pi_{\mathsf{parallel\text{-}2fw}}$ *UC realizes* $\mathcal{F}_{\mathsf{ideal\text{-}2fw}}^{\mathsf{reorder}}$ *in the* $\mathcal{F}_{\mathsf{net,parallel\text{-}2fw}}$*-hybrid model.*

Proof. To prove the statement, we will give the description of a simulator and show that this simulator can simulate every adversary, so that no environment can distinguish between the real and ideal model. Let w.l.o.g. fw_1 be corrupted and fw_2 be honest. Let $\mathcal{S}$ be a simulator with the following functionality:

> Upon activation, or when given a packet p, simulate the real model and observe its output. If the output of the real model is a packet p', advise the ideal functionality to deliver p'. (The case that p' is not present in the internal memory of the ideal functionality need not be covered, as is proven below.)

The relaxed ideal functionality of two firewalls $\mathcal{F}_{\text{ideal-2fw}}^{\text{reorder}}$

Let fw_k be the uncorrupted firewall.

- Initialize $s := \perp$ and a set of outgoing packets $P := \emptyset$.
- Upon receiving p from tc_1, compute $\text{F}_{\text{fw}_k}(p, s) = (p', s')$. If $p \neq \perp$, add p to P. Set $s := s'$ and give p to the adversary.
- Upon receiving p from the adversary, if $p \in P$, output p to tc_2. Remove p from P.

Figure 4.6: The relaxed ideal functionality of two firewalls. The adversary is explicitly allowed to reorder outgoing packets.

Note that the simulator receives exactly the same input as the adversary in the real model—it can perfectly simulate the communication between the adversary and the environment. Thus, the environment can only distinguish the models based on their output streams. We argue that the output of the real and ideal model are identical. Assume towards a contradiction that they are not.

Let $\{fw_2(S)\}$ denote the set of all packets fw_2 outputs when given the input stream S. There are two possibilities which would cause a difference in output streams:

Case 1. The adversary in the real model suppressed a packet which did not get suppressed in the ideal model. This is impossible however, since the simulator only advises the ideal functionality to deliver a packet if it observes it being output in its simulation of the real model.

Case 2. The real model outputs a packet which is not output in the ideal world.
Assume that this was the case and let p be that packet. The following conditions have to hold: p has to be in $\{fw_2(S)\}$ and p has to be output by $\mathcal{A}$ (using fw_1). This is true because the trusted hardware will ensure that a packet is only output when both firewalls accept it. For a packet *not* to be output in the *ideal model*, one of the following conditions have to hold:

- p is not in $\{fw_2(S)\}$. This way, p will not be in the internal memory of the ideal functionality. Thus, the simulator can not advise the delivery of that packet. This is a contradiction, since we assumed that p was output in the real model, which in turn implies that $p \in \{fw_2(S)\}$.
- $p \in \{fw_2(S)\}$ *and* the simulator did not advise the functionality to

deliver p. This is also a contradiction, since we assumed that p was output in the real model. This would cause the simulator to advise the output of p by definition.

We now have shown that the assumption that the environment can observe a difference in packet output stream in the real and the ideal world leads to a contradiction in all cases. This, in combination with the ability of the simulator to fully simulate the adversary, proves the indistinguishability of the models.

■

4.6 Parallel Composition of Three Firewalls

The parallel approach to compose firewalls described above does indeed improve security compared to one single and potentially malicious firewall. However, there is a large class of attacks that become possible when the adversary can selectively suppress packets. Because the parallel architecture with two firewalls cannot prevent this attack, we extend the architecture to a quorum of three firewalls.

In the following section, we assume that all uncorrupted firewalls in this architecture will have the same behavior. However, we allow them to disagree on the order of packets.

There is a non-trivial attack on this architecture. When both uncorrupted firewalls both output the same packet p, the adversary can use clever timing to output p from the corrupted firewall directly after the first uncorrupted firewall. The trusted hardware would then observe two p packets on different interfaces and output p. However, a third p packet would arrive from the second uncorrupted firewall. Then, the adversary could output p again. This would cause tc_2 to output p again and thus duplicate the packet. The natural extension of $\mathcal{F}_{\mathsf{ideal\text{-}2fw}}^{\mathsf{reorder}}$ to the case of three firewalls is vulnerable to the attack. When fw_1 and fw_2 both output the same packet, both will be saved in P. The adversary can now output both packets. This functionality is depicted in Figure 4.7.

The other protocols and functionalities $\pi_{\mathsf{parallel\text{-}3fw}}$ and $\mathcal{F}_{\mathsf{net,parallel\text{-}3fw}}$ can easily be extended to the case of three firewalls by adding the third firewall as an additional party. We will omit their descriptions here.

Theorem 22. *$\pi_{\mathsf{parallel\text{-}3fw}}$ UC realizes $\mathcal{F}_{\mathsf{ideal\text{-}3fw}}^{\mathsf{reorder}}$ in the $\mathcal{F}_{\mathsf{net,parallel\text{-}3fw}}$-hybrid model.*

The proof is very similar to the proof of Theorem 21. We omit it here.

It is not acceptable to give an attacker the ability to duplicate packets. We alter the functionality of our trusted hardware slightly to prevent the attack. The idea is that the moment the hardware outputs a packet, exactly two firewalls must have output this packet before. Then, the hardware can

The relaxed ideal functionality of three firewalls $\mathcal{F}_{\text{ideal-3fw}}^{\text{reorder}}$

Let w.l.o.g. fw_1 and fw_2 be the non-corrupted parties

- Initialize $s_1 := \bot$, $s_2 := \bot$ and a set of outgoing packets $P := \emptyset$.
- Upon receiving p from tc_1, compute $\text{F}_{\text{fw}_1}(p, s_1) = (p', s')$ and $\text{F}_{\text{fw}_2}(p, s_2) = (p'', s'')$. If $p' \neq \bot$, add p' to P and if $p'' \neq \bot$ also add p'' to P. Set $s_1 := s'$, $s_2 := s''$ and give p to the adversary.
- Upon receiving p from the adversary, if $p \in P$, output p to tc_2. Remove p from P.

Figure 4.7: The relaxed ideal functionality of three firewalls. The adversary is explicitly allowed to reorder outgoing packets.

mark this packet as missing from the third firewall. If it arrives eventually, this mark will be removed and no further action will be taken.

The definition of the resulting protocol can be seen in Definition 38.

Definition 38 (Protocol of tc_2 with packet deduplication)**.**

Initialize three empty lists Q_1, Q_2 and Q_3 for each of the three firewalls. Upon receiving packet p from firewall fw_k

- Check if Q_k contains an entry $-q$ with $p \equiv q$. If so, delete $-q$ and halt.
- Check if $\exists j \neq k$ such that $\exists q \in Q_j$ and $p \equiv q$:
  - Remove q from Q_j,
  - output p,
  - add an entry $-p$ to all other Q_i with $i \neq j$ and $i \neq k$.
- Otherwise, store p in Q_k.

Using this definition, we replace the functionality of tc_2 from $\pi_{\text{parallel-3fw}}$ and call the new protocol $\pi_{\text{parallel-3fw}}^{\text{deduplication}}$.

The corresponding ideal functionality is depicted in Figure 4.8. It now continuously checks whether the amount of identical packets being given to hw matches the amount of identical packets which either one of the uncorrupted firewalls sent. As previously however, we allow the reordering of packets.

Theorem 23. $\pi_{\text{parallel-3fw}}^{\text{deduplication}}$ *UC realizes* $\mathcal{F}_{\text{ideal-3fw}}^{\text{reorder,deduplication}}$ *in the* $\mathcal{F}_{\text{net,parallel-3fw}}$*-hybrid model.*

The ideal functionality of three firewalls without packet duplication $\mathcal{F}_{\text{ideal-3fw}}^{\text{reorder,deduplication}}$

Let w.l.o.g. fw_1 and fw_2 be the non-corrupted parties.

- Initialise three lists Q_1, Q_2 and Q_{out}. Set $s_1 := \bot$ and $s_2 := \bot$.
- Upon receiving p from tc_1: compute $\text{F}_{\text{fw}_1}(p, s_1) = (p', s')$ and $\text{F}_{\text{fw}_2}(p, s_2) = (p'', s'')$. Set $s_1 := s'$ and $s_2 := s''$. Save p' in Q_1 and p'' in Q_2. Give p to the adversary.
- Upon receiving $(\mathsf{deliver}, p''', k)$ ($k \in \{1, 2\}$) from the adversary: If Q_k contains a valid packet p''':
  - Check how many times that packet (or an equivalent packet) is in Q_{out}. Let that number be n.
  - Check if either Q_1 or Q_2 (or both) contain that packet at least $n + 1$ times.
  - If so, give p''' to tc_2 and save it in Q_{out}.

Figure 4.8: The ideal functionality of three firewalls without packet duplication. For every packet, at least one of the firewalls must have sent this packet at least as often as it got passed to tc_2.

Proof. The proof is similar to the proof of Theorem 21. We argue that the simulator behaves identically to the adversary and that the output of the ideal network is identical to the output of the real network. Let $\mathcal{S}$ be a simulator with the following functionality:

> Upon activation, or when given a packet p, simulate the real model and observe its output. If the output of the real model is a packet p', calculate (for the ideal functionality) the index of the memory structure in which p' is saved as well as its position within the memory. Advise the functionality to deliver the packet on that index. (The case that p' is not found in the internal memory structure of the ideal functionality need not be covered, as is proven below.)

The argument that $\mathcal{S}$ is always able to suppress a packet in the ideal model, which is suppressed in the real world, is identical to Case 1 in the proof of Theorem 21. We need to argue Case 2: $\mathcal{S}$ is able to schedule every packet it observes in the output of its internal simulation of the real network. Assume towards a contradiction, that p is such a packet that, after the input stream S is processed, is output by tc_2 in the real model but not output by the ideal functionality.

Let Q_1, Q_2 and Q_3 be the lists the trusted hardware uses in the real protocol for storing the packets output by the firewalls and marking the "negative" packets. Let Q_{tc} be the list of all packets it has ever output. Let Q'_1, Q'_2, Q'_{out} be the lists the ideal functionality uses for keeping track of the packets. Let $\|Q\|_p$ denote the number of packets p the list Q contains. We then define $|Q|_p := \|Q\|_p - \|Q\|_{-p}$.

First, observe that $\mathcal{S}$ only schedules packets it observes in its simulation of the real model. Hence, by the description of tc_2 $|Q_1|_p = |Q'_1|_p - |Q_{\mathrm{tc}}|_p$ and $|Q_2|_p = |Q'_2|_p - |Q_{\mathrm{tc}}|_p$. We know from the argument for Case 1 that the ideal functionality will never output packets, which were not output in the real model ($\forall p : |Q'_{out}|_p \leq |Q_{\mathrm{tc}}|_p$). We thus have:

$$|Q_1|_p \leq |Q'_1|_p - |Q'_{out}|_p \tag{4.1}$$
$$|Q_2|_p \leq |Q'_2|_p - |Q'_{out}|_p \tag{4.2}$$

For p to be output in the real model, one of the following conditions has to hold:

$$|Q_1|_p > 0 \text{ and } |Q_2|_p > 0 \tag{4.3}$$
$$|Q_1|_p > 0 \text{ and } |Q_3|_p > 0 \tag{4.4}$$
$$|Q_2|_p > 0 \text{ and } |Q_3|_p > 0 \tag{4.5}$$

This is true because the trusted hardware will only forward packets which are in at least two of the packet lists. The functionality of tc_2 can be restated in the following way: For every packet p which is output, insert a packet $-p$

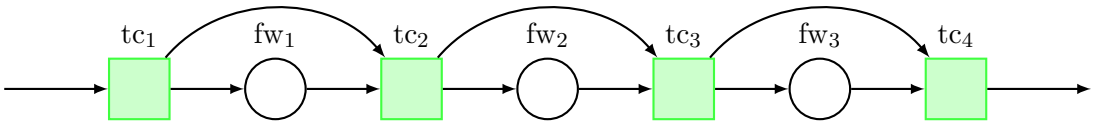


Figure 4.9: The serial composition of three firewalls.

into the lists of the three firewalls. If there are two packets p and $-p$ in the same list, both cancel each other out.

For p *not* to be written to any of the lists in the ideal model, the following condition has to hold:

$$|Q'_{out}|_p \geq |Q'_1|_p \text{ and } |Q'_{out}|_p \geq |Q'_2|_p \tag{4.6}$$
$$\Leftrightarrow |Q'_1|_p - |Q'_{out}|_p \leq 0 \text{ and } |Q'_2|_p - |Q'_{out}|_p \leq 0 \tag{4.7}$$

This again describes the difference between the amount of packages p each individual firewall has output and the amount of packages p which got output in total after processing S.

Concluding the argument, conditions (4.1) to (4.5) give us $|Q'_1|_p - |Q'_{out}|_p > 0$ and $|Q'_2|_p - |Q'_{out}|_p > 0$, which contradict condition (4.7). ■

4.7 Serial Composition of Three or More Firewalls

Even though the serial combination of two firewalls is insecure, we can now take inspiration from the parallel architecture with three firewalls to build a (semi-)secure serial architecture using three firewalls. The main idea is to place one trusted hardware component between each firewall. These components have forward each incoming packet to the next firewall as well as to the next trusted hardware. This ensures that the decision of each firewall is counted. Figure 4.9 shows this architecture and Definition 39 depicts the new functionality of the trusted hardware components. The new network functionality is shown in Figure 4.10. We use the same protocol for the firewalls, as in the previous chapters—that is, they evaluate their specific firewall functionality and update their state. We call the protocols for tc and fw together the *protocol of the serial architecture* $\pi_{\mathsf{serial\text{-}3fw}}$

Definition 39 (Protocol of the trusted components tc_i)**.**

Let n be the total number of firewall members in the architecture. Let n be an odd number.

- tc_1:
  - Upon receiving p from $\mathcal{Z}$: Call $\mathcal{F}_{\mathsf{net,serial\text{-}3fw}}(p,(p,0))$.
- tc_k with $k \in [2,n[$:

The network function of the serial architecture with three firewalls. $\mathcal{F}_{\mathsf{net,serial\text{-}3fw}}$

- Initialize three empty first-in-first-out queues: P_1, P_2, P_3.
- Upon receiving (p, ctr_p) from tc_k: Add (p, ctr_p, k) to P_1 and (p, k) to P_2.
- Upon receiving p' from fw_k: Add (p', k) to P_3.
- Upon instruction from the adversary, fetch (p, ctr_p, k) from P_1 and deliver (p, ctr_p) to tc_{k+1}
- Upon instruction from the adversary, fetch (p, k) from P_2 and deliver p to fw_k.
- Upon instruction from the adversary, fetch (p', k) from P_3 and deliver p' to tc_{k+1}.

Figure 4.10: Network function of the serial architecture with three firewalls.

  - Upon receiving (p, ctr_p) from tc_{k-1} via $\mathcal{F}_{\mathsf{net,serial\text{-}3fw}}$:
    If there is a saved packet p' with $p \equiv p'$, set $\mathrm{ctr}_p := \mathrm{ctr}_p + 1$. Call $\mathcal{F}_{\mathsf{net,serial\text{-}3fw}}(p, \mathrm{ctr}_p)$.
    Else, if there is no saved packet, save p and ctr_p.
  - Upon receiving p from fw_{k-1} via $\mathcal{F}_{\mathsf{net,serial\text{-}3fw}}$:
    If there is a saved packet $p' \equiv p$ and a saved counter $\mathrm{ctr}_{p'}$, set $\mathrm{ctr}_p := \mathrm{ctr}_{p'} + 1$. Call $\mathcal{F}_{\mathsf{net,serial\text{-}3fw}}(p, \mathrm{ctr}_p)$.
    Else, if there is no saved packet, save p.

- tc_n:
  - Upon receiving (p, ctr_p) from tc_{n-1} via $\mathcal{F}_{\mathsf{net,serial\text{-}3fw}}$:
    - If $\mathrm{ctr}_p \geq \frac{n+1}{2}$: output p to $\mathcal{Z}$.
    - If $\mathrm{ctr}_p = \frac{n-1}{2}$: If there is a saved packet p' and $p \equiv p'$, output p to $\mathcal{Z}$. If there is no saved packet, save p.
  - Upon receiving p from fw_{n-1} via $\mathcal{F}_{\mathsf{net,serial\text{-}3fw}}$: If there is a saved packet p' and $p \equiv p'$, output p to $\mathcal{Z}$

We will now show that this architecture is secure—but only if we restrict the firewall's functionality in a way that each firewall performs the same, stateless function. First, observe that this architecture does not work if

The ideal functionality of the serial architecture with three firewalls. $\mathcal{F}^{\text{reorder}}_{\text{ideal-serial-3fw}}$

- Initialize $s := \bot$ and a set of outgoing packets $P := \emptyset$.
- Upon receiving p from tc_1, compute $\text{F}_{\text{fw}}(p, s) = (p, s')$ If $p \neq \bot$, add p to P. Set $s := s'$ and give p to the adversary.
- Upon receiving p from the adversary, if there is a $p \in P$, output p to tc_4. Remove p from P.

Figure 4.11: Ideal functionality of the serial architecture with three firewalls. If any of the uncorrupted firewalls accepts a packet, it is stored in a packet list from which it can then be delivered by the adversary.

firewalls change packets. If a packet p enters the network, it is directly forwarded to tc_2, which will wait for the same packet to be output by fw_1. If fw_1 now changes that packet to p', tc_2 will never find a matching packet for p and the architecture will drop the packet.

To make the architecture work at all, we have to restrict the firewall functionality to accept packets as they are, or drop them completely. Also, remember that we assume that uncorrupted firewalls will always agree on whether a particular packet should be dropped or accepted. Thus, it is no longer necessary to model a firewall functionality for each firewall, since all uncorrupted firewalls will always agree on the same output anyway. Adhering to our past notation, we call this global firewall functionality F_{fw}.

Definition 40 shows the resulting protocol and Figure 4.11 the resulting ideal functionality, which is an adapted version of $\mathcal{F}^{\text{reorder}}_{\text{ideal-3fw}}$.

Definition 40 (Protocol of the firewall fw_i)**.**

- Initialize $s := \bot$
- Upon receiving p from tc_i: compute $\text{F}_{\text{fw}}(p, s) = (p, s')$. If $p \neq \bot$, call $\mathcal{F}_{\text{net,serial-3fw}}(p)$ and set $s := s'$.

We can now state our theorem.

Theorem 24. *$\pi_{\text{serial-3fw}}$ does not realize $\mathcal{F}^{\text{reorder}}_{\text{ideal-serial-3fw}}$ in the $\mathcal{F}_{\text{net,serial-3fw}}$-hybrid model.*

Proof. To proof our claim, we give an attack in the real model which cannot be simulated in the ideal world by any simulator.

Assume a stateful firewall functionality $\mathrm{F}_{\mathrm{fw}_2}$ which drops every second packet. In the real model, the adversary corrupts fw_1. He then switches the order of the first two packets, which are processed by the architecture. More concretely, assume two packets p and p' are input into the network and get delivered to fw_1 in this order. The adversary now reverses the order, which will cause fw_2 to drop p, which in turn will cause the whole architecture to drop p and accept p'.

In the ideal world, the simulator cannot change the order of incoming packets and fw_2 will drop p' instead. Thus, the simulator will not be able to instruct the delivery of p', even though it was output in the real model.

■

We now state our main result that the serial architecture with three firewalls is secure for any (stateless) firewall functionality (i.e. s is always set to $\perp$).

Theorem 25. *$\pi_{\mathsf{serial\text{-}3fw}}$ realizes $\mathcal{F}^{\mathsf{reorder}}_{\mathsf{ideal\text{-}serial\text{-}3fw}}$ in the $\mathcal{F}_{\mathsf{net,serial\text{-}3fw}}$-hybrid model for n firewall members and any stateless firewall functionality F_{fw}, if there are at most $\frac{n-1}{2}$ corrupted firewall members.*

Proof. The proof is provided by giving a description of a simulator $\mathcal{S}$ and showing that its behavior is such that no environment $\mathcal{Z}$ can distinguish the execution of $\mathcal{F}^{\mathsf{reorder}}_{\mathsf{ideal\text{-}serial\text{-}3fw}}$ from an execution of the real protocol using tc_i and fw_i. The simulator $\mathcal{S}$ is notified by $\mathcal{F}^{\mathsf{reorder}}_{\mathsf{ideal\text{-}serial\text{-}3fw}}$ of each packet coming into the network. He maintains an internal simulation of the real network and for each packet p in the real network's output, instructs $\mathcal{F}^{\mathsf{reorder}}_{\mathsf{ideal\text{-}serial\text{-}3fw}}$ to deliver p as well. We argue that the outputs of $\mathcal{F}^{\mathsf{reorder}}_{\mathsf{ideal\text{-}serial\text{-}3fw}}$ and the real protocol are identical.

Let n be the number of firewall members in the architecture. The core of our argument is that the firewall functionality is stateless, i.e. $\mathrm{F}_{\mathrm{fw}}(p, s)$ ignores its second argument. For simplicity we write $\mathrm{F}_{\mathrm{fw}}(p)$. For the sake of clarity we denote $\mathcal{F}^{\mathsf{reorder}}_{\mathsf{ideal\text{-}serial\text{-}3fw}}$'s firewall functionality by $\mathcal{F}_{\mathrm{fw-ideal}}$ and that of fw_i by $\mathcal{F}_{\mathrm{fw-real}}$.

Note that $\mathcal{S}$ is made aware of each packet arriving at the network (in their correct order) and thus can simulate the behavior of the network perfectly. Also, because $\mathcal{F}^{\mathsf{reorder}}_{\mathsf{ideal\text{-}serial\text{-}3fw}}$ maintains a set of outbound packets from which the simulator can choose packets to deliver, $\mathcal{S}$ can change the order of packets and also drop them at will. Let I be the set of packets in $\mathcal{F}^{\mathsf{reorder}}_{\mathsf{ideal\text{-}serial\text{-}3fw}}$'s output and let R be the set of packets in the output of the real network. We show that there is no packet in R that is not also in I, i.e. $I \supseteq R$. Assume towards a contradiction that $I \not\supseteq R$. Thus $\exists p \in R : p \notin I$ which must have been output by tc_n. By design of the architecture, tc_n collects the decisions of all firewall members. Consequently, there must have been at least $\frac{n+1}{2}$ firewall members for which there exists an incoming packet o such that $\mathcal{F}_{\mathrm{fw-ideal}}(o) = \perp \neq \mathcal{F}_{\mathrm{fw-real}}(o) = p$. We now observe that $\mathcal{F}^{\mathsf{reorder}}_{\mathsf{ideal\text{-}serial\text{-}3fw}}$

evaluates the same firewall functionality as uncorrupted firewall members in the real protocol: $\mathcal{F}_{\text{fw}-\text{ideal}} = \mathcal{F}_{\text{fw}-\text{real}}$. Thus, only corrupted firewall members can cause a difference in outputs. This is a contradiction to the assumption that there are at most $\frac{n-1}{2}$ corrupted firewall members. ■

We point out, that the above argument does indeed not work for stateful firewall functionalities as there might exist states s, s' such that $\mathcal{F}_{\text{fw}-\text{ideal}}(o, s) \neq \mathcal{F}_{\text{fw}-\text{real}}(o, s') = p$.

4.8 Improving the Model: Availability and Bigger Networks

In the previous sections, we have modeled and analyzed several firewall architectures using the basic UC framework. As discussed previously, one of the major drawbacks of the basic framework is that it does not allow the expression of availability guarantees. For computer networks however, availability is an important property. Fortunately, Katz et al. designed an extension to the UC framework, which allows to model availability [75]. Unfortunately, this extension introduces a number of modeling artifacts, which makes it extremely difficult to use. We address some of these problems, by introducing new formalisms. Finally, while it is easily possible to write a custom ideal functionality for each network structure, when there are only a few participants, this does not scale well for bigger networks. We therefore design a generic network functionality, which takes the graph of the network as input and automatically takes care of all the required formalisms of the UC framework.

4.8.1 The Basic Tools

Ideally, modeling and analyzing a network would require four steps: 1) Specify what the wanted functionality of the network is, 2) draw a graph of the network layout, 3) specify the protocol the machines in the network adhere to, and 4) prove that the protocol does achieve what the wanted functionality does.

We designed tools that capture various technical details of the UC framework and allow to use it in a way that is close to the intuitive approach. Specifically,

1. By defining $\mathcal{F}_{\mathsf{wrap}}$, we simplify the specification of an ideal network functionality.

2. We provide an ideal network functionality $\mathcal{F}_{\mathsf{net}}^{G}$ that routes messages according to a given network topology induced by a network graph G.

3. We propose a 5-phase paradigm which allows for an easy and structured modeling of the behavior of machines in the network.

The Ideal Network Functionality We model the network as a directed graph $G = (V, E)$, while V is the set of machines in the network and $E \subseteq V^2$ is a relation on V. (We model the network as a directed graph to account for unidirectional links [72].) To model bidirectional links, one requires that $(v, v') \in E$ iff $(v', v) \in E$. There is a delivery queue for each edge in the graph. Nodes can send messages for each outgoing edge and can poll incoming messages from each incoming edge. To send a packet, a party *src* can call the network functionality $\mathcal{F}_{\mathsf{net}}^{G}$ with a (finite) set of packets with corresponding recipients $\{(dest_1, msg_1), (dest_2, msg_2), \dots\}$. Each packet in the set will then be appended to the queue associated with the edge between nodes *src* and $dest_i$, if it exists. Further, modeling Katz et al.'s bounded delay channels [75], we associate two counters with each edge in the graph—one for the total delay and one for the accumulated delay of the channel. The adversary can increase the delay of a channel up to a fixed maximum amount. When a machine polls a queue the delay counter is decreased. When the delay has reached 0, a message is taken from the queue and handed to the machine. This allows for explicit modeling of different network latencies across different communication channels and allows the adversary to take advantage of that. This functionality makes it easy to define the communication channels for a network since one provides a graph of the network and the corresponding channel topology for the UC framework is generated automatically. We point out that we implicitly use Katz et al.'s "multisend" functionality where parties send multiple packets in one atomic call to the network. Because we do not consider adaptive corruption, the adversary cannot preempt parties during a send operation.

The 5-Phase Paradigm We propose a 5-phase paradigm for modeling network protocols. We require each honest party to follow this paradigm. An honest party will need exactly five explicit activations by the environment machine to finish its round. During its first activation ("input phase"), the party will accept input by the environment. Upon the second activation ("fetch phase"), it will issue a fetch request to the network to get its input which it will process and possibly send to other parties in one single call during the third activation ("send phase"). The fourth activation ("output phase") is the only activation in which a party will produce output to the environment. The fifth activation is used to signal "$\mathsf{RoundOK}$" to $\mathcal{F}_{\mathsf{clock}}$: all work is done for this round.

Upon further activations the party will wait for the next round to begin. We stress that an honest party will poll the network exactly once per round while a compromised party might poll the network more often. We assume that

The ideal parameterized network function $\mathcal{F}_{\mathsf{net}}^{G,\delta}$

Interpret $G = (V, E)$ with $E \subseteq V^2$ as a directed graph. For each edge $e \in E$, initialize a queue Q_e and two variables d_e and d'_e which represent the current and the accumulated delay for the queue.

- Upon receiving a message (send, M) with $M = \{(dest_1, msg_1), (dest_2, msg_2), \dots\}$ from party src, for each tuple $(dest, msg) \in M$ do:
  - Check if $src, dest \in V$ and $(src, dest) \in E$. If so, continue. Else, ignore this tuple and start processing the next message.
  - Append msg to queue $Q_{(src,dest)}$. Hand msg to the adversary.
- Upon receiving message (delay, e, T) from $\mathcal{A}$: Let (d_e, d'_e) be the delay variables for the queue of edge e. If $d'_e + T \leq \delta$, set $d_e = d_e + T$ and $d'_e = d'_e + T$ and return $(\mathsf{delay\text{-}set})$ to the adversary. Else halt.
- Upon receiving message (fetch, Q) from party P and if $Q \subseteq V$:
  - Initialize a set of responses $r := \emptyset$ and for every party $P' \in Q \subseteq V$ with $(P', P) \in E$:
    - Let $(d_{(P',P)}, d'_{(P',P)})$ be the delay variables for edge (P', P).
    - Set $d_{(P',P)} = d_{(P',P)} - 1$. If $d_{(P',P)} = 0$, remove the first message msg from $Q_{(P',P)}$, set $d'_{(P',P)} = 0$, and set $r = r \cup (msg, (P', P))$.
  - If $r \neq \emptyset$, send r to P. Else halt.

Figure 4.12: The generalized ideal network function. It is parameterized with a graph that features protocol participants as nodes and expresses links as edges. We model the network as a directed graph to accommodate for specialized connection types as for example optical fibers or data diodes [72]. We also implemented Katz et al.'s bounded delay-channel [75] to model links with a delay.

every party will initialize and update a round counter and further maintain state for the number of activations per round and whether (RoundOK) has already been signaled. This requires sending $\mathcal{F}_{\mathsf{clock}}$ a (RequestRound) request on activation and accordingly updating state information, but imposes no limitations for the party. This paradigm simplifies the modeling of network protocols by structuring the behavior of network parties but it also allows for a more intuitive description of the ideal functionality the network performs, as we will discuss in the next paragraph.

Note that this model (like the one of Katz et al. [75]) requires the environment to explicitly schedule the network. However, intentional starvation of one network component is not a suitable strategy for distinguishing the real from the ideal model, as the ideal functionality will be aware of round changes and can ensure that there is no observable progress in both models.

The Wrapper Functionality To simplify the definition of ideal functionalities, we introduce an ideal "wrapper" functionality $\mathcal{F}_{\mathsf{wrap}}$ (see Figure 4.13 (a)). It "wraps around" the ideal functionality and moderates its communication with the dummy parties in the ideal world. Its main task is to count activations of dummy parties. Since honest parties adhere to the 5-phase paradigm, it will only notify the ideal functionality if the environment gives input to a party (during the first activation), if a party could create output in the real model (during its fourth activation) and when a round is complete. It also ensures that the adversary is activated at least as often as in the real model.

Specifying Ideal Functionalities The tools introduced above allow for a natural description of ideal functionalities. $\mathcal{F}_{\mathsf{wrap}}$ will send a notification for important events (e.g. inputs, outputs and round changes) and the ideal functionality reacts to them appropriately. Specifically, the ideal functionality will not be required to count activations itself or activate the adversary sufficiently often. Since the network functionality provides a bound for the maximum delay a channel can have, it is also easily possible to model availability. The ideal functionality only has to maintain a counter corresponding to the delay δ of the channel for each packet and reduce this counter by one every time a round is complete. When the counter reaches zero, the packet can be output immediately when output is requested by $\mathcal{F}_{\mathsf{wrap}}$. Since all honest parties will poll for new packets once per round the adversary can delay a packet delivery for a maximum of δ rounds per channel.

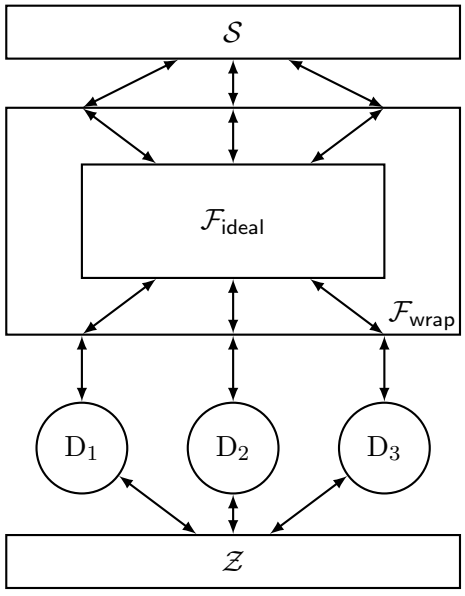
Note that we only specify the behavior for input by honest parties. We implicitly assume that messages from the adversary to corrupted parties or vice versa are delivered immediately.

The wrapping function for ideal functionalities $\mathcal{F}_{\mathsf{wrap}}$

Maintain an activation counter c_p for each of the honest dummy parties. Relay all communication from $\mathcal{F}_{\mathsf{ideal}}$ directly to the environment. Upon activation by the environment, i.e. upon receiving input m through a dummy party p:

- If $c_p < 5$ increase the activation counter of the party.
- If $c_p = 1$ send message (input, m, p) to $\mathcal{F}_{\mathsf{ideal}}$.
- If $c_p = 2$ or $c_p = 3$, send message $(\mathsf{activated}, \mathsf{p})$ to the adversary.
- If $c_p = 4$ send message (output, p) to $\mathcal{F}_{\mathsf{ideal}}$.
- If $\forall p' : c_{p'} = 5$ reset all activation counters and send $(\mathsf{RoundComplete})$ to $\mathcal{F}_{\mathsf{ideal}}$.

(a)



(b)

Figure 4.13: The ideal "wrapper" functionality $\mathcal{F}_{\mathsf{wrap}}$ acts as a relay between the dummy parties and the ideal functionality. It counts activations of parties and notifies the ideal functionality of important events like round changes, thus simplifying the formulation of ideal functionalities.

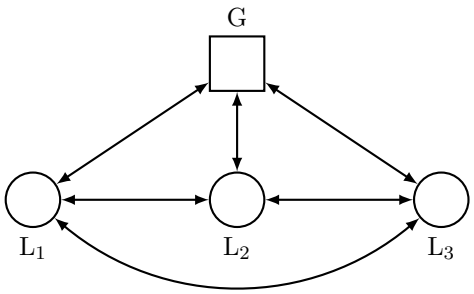


Figure 4.14: The network graph $\mathsf{byz} = (V, E)$ for the Byzantine Generals problem with $V = \{G, L_1, L_2, L_3\}$ and $E = V^2$. It is fully connected—each party can communicate with every other party.

4.8.2 Example: Byzantine Generals

As an example, we will use the presented methodology to model a popular example from the literature: the Byzantine Generals problem. We will then restate a popular result concerning this problem by giving a proof in our framework.

The Byzantine Generals Problem The Byzantine Generals problem was first introduced by Lamport, Shostak, and Pease [81]. The motivation is as follows: suppose that a commanding general wants to give orders (for the sake of simplicity he will only use "attack" or "retreat") to his lieutenants but he does not know which of them are trustworthy. Also, the lieutenants do not know whether the general himself is trustworthy. Now suppose that each of the participants can communicate with each other participant via "oral" messages. The Byzantine Generals problem is to find an algorithm that, given a number of parties n (one of them is the general), ensures that:

1. All loyal lieutenants obey the same order and

2. If the general is loyal, then every loyal lieutenant obeys the order he sends.

Note that a disloyal (corrupted) lieutenant can arbitrarily lie about messages he received and try to deceive other lieutenants. He can also refuse to send any messages. However, it is assumed that loyal parties will notice when messages are missing. Lamport et al. [81] show that there can not be a generic solution to the problem for three parties, but there is a solution for four parties. We will now model the Byzantine Generals problem with four parties according to our methodology and give a formal security proof for a specific solution to the problem.

Modeling the Byzantine Generals Problem The network in this example is fully connected. Every party can transmit messages to every other party. There is a maximum latency of 2δ until a packet is output by one of the parties: a possible delay of δ from the general to the lieutenants and another possible delay of δ for a packet from one lieutenant to reach the others.

The Byzantine Generals problem statement implies that a party notices if it will not receive messages from another party anymore, so that it will not wait indefinitely. In reality this is usually realized by timeouts—we will use the same mechanism here.

Figure 4.15 shows the protocol which implements a solution to the generals problem. Figure 4.16 shows the corresponding ideal functionality. This functionality fulfills the requirements for a solution to the Generals problem given earlier.

We will now show that this protocol realizes the ideal functionality.

Theorem 26. *π_{byz} realizes $\mathcal{F}_{\mathsf{byz\text{-}ideal}}$ in the $\mathcal{F}_{\mathsf{net}}^{\mathsf{byz},\delta}$-hybrid model.*

Proof. We prove the theorem by giving a stepwise transformation from the real model to the ideal model. We argue that the individual transformation steps are indistinguishable for the environment, and thus, by the transitivity of indistinguishability, the real model is indistinguishable from the ideal model. Start with the real protocol.

Regroup all parties into a new machine $\mathcal{S}$. The adversary simulator $\mathcal{S}$ will simulate the real network in all transformation steps. Introduce dummy parties D_{G}, D_{L_1}, D_{L_2}, D_{L_3} for all protocol parties and relay messages from and to $\mathcal{Z}$ appropriately. Introduce a new machine $\mathcal{F}_{\mathsf{byz\text{-}ideal}}$. Route all communication from the dummies to $\mathcal{S}$ and vice versa through $\mathcal{F}_{\mathsf{byz\text{-}ideal}}$. The regrouping of parties is indistinguishable for the environment. In the upcoming transformation steps, we will gradually expand $\mathcal{F}_{\mathsf{byz\text{-}ideal}}$'s functionality:

1. Initialize variables m_{L_1}, m_{L_2}, and m_{L_3}. When receiving a message m from dummy party G, set $m_{L_1} := m$, $m_{L_2} := m$ and $m_{L_3} := m$. Also initialize and save a round counter $d := 2\delta$. This modification is indistinguishable, since it only stores information and does not alter the communication.

2. If G is corrupted, accept a message $(\mathsf{set}, m_1, m_2, m_3)$ from $\mathcal{S}$. Check if there are $i \neq j$ such that $m_i = m_j$. If so, set $m_{L_1}, m_{L_2}, m_{L_3}$ to m_i. Else set $m_{L_1} = m_1, m_{L_2} = m_2, m_{L_3} = m_3$. This modification again only stores information.

3. When $\mathcal{S}$ attempts to pass output m from an uncorrupted party p in the simulation back to the dummy party, only allow it to pass through $\mathcal{F}_{\mathsf{byz\text{-}ideal}}$ if either

A solution to the Byzantine Generals problem with four parties π_{byz}

Each party maintains a local round counter r.

- Party G:
  - "Input": Upon first activation this round and input m by $\mathcal{Z}$, save m and ignore further inputs.
  - "Send": Upon third activation, call $\mathcal{F}_{\mathsf{net}}^{\mathsf{byz}}(\mathsf{send}, (\mathrm{L}_1, m), (\mathrm{L}_2, m), (\mathrm{L}_3, m))$ if m was saved.
  - "RoundOK": Upon fifth activation, send $(\mathsf{RoundOK})$ to $\mathcal{F}_{\mathsf{clock}}$.
- Party L_n:
  - "Fetch": Upon second activation,
    - call $\mathcal{F}_{\mathsf{net}}^{\mathsf{byz}}(\mathsf{fetch}, \{\mathrm{G}, \mathrm{L}_k, \mathrm{L}_j\})$ for $k \neq j \neq n$. If the call was successful, save the messages for the corresponding parties.
  - "Send": Upon third activation,
    - if there is a message m by party G which has not been broadcast yet, broadcast it: call $\mathcal{F}_{\mathsf{net}}^{\mathsf{byz}}(\mathsf{send}, (\mathrm{L}_k, m), (\mathrm{L}_j, m))$ with $k, j \neq n$.
  - "Output": Upon fourth activation,
    - if $r < 2\delta$ and there are two identical messages m from two different parties (other than G), output m. If there are three different messages from the different parties, output the message from party 1;
    - if $r = 2\delta$ output $\mathsf{retreat}$.
  - "RoundOK": Upon fifth activation, send $(\mathsf{RoundOK})$ to $\mathcal{F}_{\mathsf{clock}}$.

Figure 4.15: The protocol for the Byzantine Generals problem with four parties. The ideal network functionality allows for a maximum delay of δ for each message and messages have to be sent from the general first and from the lieutenants afterwards. Thus a party will assume a timeout after 2δ rounds.

The ideal functionality of the Byzantine Generals problem with four parties $\mathcal{F}^{\delta}_{\mathsf{byz\text{-}ideal}}$.

Upon initialization store a delay value $d := (2\delta)$ and initialize three variables $m_{L_1} := \perp, m_{L_2} := \perp, m_{L_3} := \perp$.

- Upon receiving message $(\mathsf{input}, m, \mathrm{G})$ from $\mathcal{F}_{\mathsf{wrap}}$ and if G is honest: store $m_{L_p} := m$ for $p \in \{1, 2, 3\}$ and send $(\mathsf{input}, m, \mathrm{G})$ to the adversary.
- Upon receiving message $(\mathsf{set}, m_1, m_2, m_3)$ from the adversary and if G is corrupted: if $m_{L_1} = \perp, m_{L_2} = \perp, m_{L_3} = \perp$, and there are two identical messages m_i, m_j with $i \neq j$, set $m_{L_1}, m_{L_2}, m_{L_3} := m_i$, else set $m_{L_1}, m_{L_2}, m_{L_3} := m_j$ where j is the smallest index for which $m_j \neq \perp$.
- Upon receiving message $(\mathsf{output}, p_1, p_2, p_3)$ from the adversary: mark messages m_{p_1},m_{p_2},m_{p_3} as ready for output.
- Upon receiving message (output, p) from $\mathcal{F}_{\mathsf{wrap}}$:
  - If $d = 0$: output $\mathsf{retreat}$ to p.
  - if $d \neq 0$ and if m_p is marked as ready for output, output m_p to p.
- Upon receiving message $(\mathsf{RoundComplete})$ from $\mathcal{F}_{\mathsf{wrap}}$, decrease d by 1 and send $(\mathsf{RoundComplete})$ to the adversary.

Figure 4.16: The ideal functionality of the three generals problem. If the general is honest, all honest parties will obey his order. If he is corrupted, all parties will obey the same order. As in the real protocol the adversary can not delay the output for more than 2δ rounds.

(a) m has been stored as m_p in $\mathcal{F}_{\mathsf{byz\text{-}ideal}}$, or

(b) the message is retreat.

We have to argue the indistinguishability of this modification. A real protocol party will only output a message other than retreat when it has received two identical messages. This will only happen if

(a) G is honest—then, m will have been provided by $\mathcal{Z}$ through dummy party G and thus saved for every party in the ideal functionality, or

(b) G is corrupted and sent two identical messages. In this case, $\mathcal{S}$ will have used the set-message to provide these messages and they will also have been saved for every party.

4. Introduce $\mathcal{F}_{\mathsf{wrap}}$ as a wrapper around $\mathcal{F}_{\mathsf{byz\text{-}ideal}}$. For each notification that a round is complete from $\mathcal{F}_{\mathsf{wrap}}$ decrease the delay value d and notify $\mathcal{S}$ that the round is complete. $\mathcal{F}_{\mathsf{wrap}}$ will not notify $\mathcal{S}$ about activations in phase 4 ("output"), but $\mathcal{F}_{\mathsf{byz\text{-}ideal}}$ instead. The simulator is thus not able to accurately simulate the exact order of outputs. However, the simulator is still able to determine the set of messages to output for each party in each round: he still is notified about the input to the protocol, when a party sends a message, and when a round is complete. We alter the strategy of $\mathcal{S}$ to make the modification indistinguishable: in each round, observe which parties will output a message and notify the ideal functionality that these parties are ready for output. Now, when $\mathcal{Z}$ activates a party and expects output, the ideal functionality will output possible messages for that specific party. This allows for all messages other than retreat to be output correctly. So, if $d = 0$ after the fourth activation of a party, $\mathcal{F}_{\mathsf{byz\text{-}ideal}}$ just outputs retreat, mimicking the behaviour in the real model. $\mathcal{F}_{\mathsf{byz\text{-}ideal}}$ and $\mathcal{S}$ now behave as specified in the ideal model, perfectly emulating the real model.

This concludes the proof. ■

4.8.3 Firewalls Revisited

In this section, we will apply our improved model to the previously discussed example of the three firewall architecture. In addition to being easier to work with, this new model also allows us to express the availability properties of the architecture.

Definition 41 (The functionality of an ideal firewall $\mathrm{F}_{\mathrm{fw}_j}$)**.**

$$\mathrm{F}_{\mathrm{fw}_j} : P \times V \times S \to (P \cup \perp) \times (V \cup \perp) \times S$$

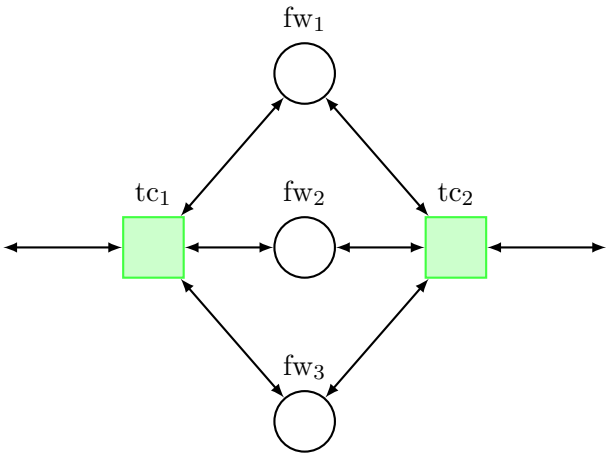


Figure 4.17: The three firewall network. The graph directly serves as the network model for $\mathcal{F}_{\mathsf{net}}^{G}$: $G = (V, E)$ with $V = \{\mathrm{tc}_1, \mathrm{tc}_2, \mathrm{fw}_1, \mathrm{fw}_2, \mathrm{fw}_3\}$ and $E' = \{(\mathrm{tc}_1, \mathrm{fw}_1), (\mathrm{tc}_1, \mathrm{fw}_2), (\mathrm{tc}_1, \mathrm{fw}_3), (\mathrm{tc}_2, \mathrm{fw}_1), (\mathrm{tc}_2, \mathrm{fw}_2), (\mathrm{tc}_2, \mathrm{fw}_3)\}, E = E' \cup \{(v, u) \mid (u, v) \in E'\}$.

$$F_{\mathrm{fw}_j}(p, v, s) = \begin{cases} (p', v', s') & \text{if output is generated,} \\ (\bot, \bot, s') & \text{else.} \end{cases}$$

Definition 41 provides a modified definition of the firewall function F_{fw}, adapted to work with the new graph based network model (Figure 4.12). The function accepts a packet p from the set of all packets P, a node from the network graph $v \in V$ and a state $s \in S$ and outputs another packet, another node (the receiver of that packet) and a new state.

Definition 42 shows the protocol of the three firewall solution as expressed using the improved model. Figure 4.18 show the corresponding ideal functionality.

Definition 42 (The protocol of the three firewall architecture π_{fw})**.**
party tc_k:

- "Input": Upon first activation by message (input, m) from $\mathcal{Z}$, save m.
- "Fetch": Upon second activation by message (output) from $\mathcal{Z}$,
  - call $\mathcal{F}_{\mathsf{fw\text{-}net}}(\mathsf{fetch}, \{\mathrm{fw}_1, \mathrm{fw}_2, \mathrm{fw}_3\})$, save the message m corresponding to fw_i as (m, i);
  - if there are two entries (m, i) and $(-m, i)$ on the tape, delete both.
- "Send": Upon third activation by message (output) from $\mathcal{Z}$, call $\mathcal{F}_{\mathsf{fw\text{-}net}}(\mathsf{send}, (\mathrm{fw}_1, m), (\mathrm{fw}_2, m), (\mathrm{fw}_3, m))$ if m was saved previously. Delete m.

- "Output": Upon fourth activation by message (output) from $\mathcal{Z}$, if there are two saved entries (m, i) and (m', i') with $m \equiv m'$ and $i \neq i'$: delete both messages and output m. If $i, i' \neq 1$, save $(-m, 1)$, else if $i, i' \neq 2$, save $(-m, 2)$, else if $i, i' \neq 3$, save $(-m, 3)$.
- "RoundOK": Upon fifth activation by message (output) from $\mathcal{Z}$, send (RoundOK) to $\mathcal{F}_{\mathsf{clock}}$.

party fw_k:

- "Fetch": Upon second activation by message (output) from $\mathcal{Z}$,
  - call $\mathcal{F}_{\mathsf{fw\text{-}net}}(\mathsf{fetch}, \mathrm{tc}_1, \mathrm{tc}_2)$ and save the message m corresponding to tc_i as (m, i);
  - for all saved messages (m, i): compute $F_{\mathrm{fw}_k}(m, i, s) = (m', i', s')$ and replace that (m, i) with (m', i').
- "Output": Upon fourth activation by message (output) from $\mathcal{Z}$, if there are two messages (m, i) and (m', i'), call $\mathcal{F}_{\mathsf{fw\text{-}net}}(\mathsf{send}, (\mathrm{tc}_i, m), (\mathrm{tc}_{i'}, m'))$.
- "RoundOK": Upon fifth activation, send (RoundOK) to $\mathcal{F}_{\mathsf{clock}}$.

Theorem 27. *π_{parallel} realizes $\mathcal{F}_{\mathsf{fw\text{-}ideal}}$ in the $\mathcal{F}_{\mathsf{net}}^{\mathsf{fw},\delta}$-hybrid model.*

Proof. We prove the lemma via game hopping, starting from the real model. In each step we will modify the ideal functionality and argue that the modification is indistinguishable. We will w.l.o.g. assume that fw_3 is corrupted. Encapsulate the network in a new machine $\mathcal{S}$, introduce dummies for all fw_i and hw_i, and construct a new machine $\mathcal{F}_{\mathsf{fw\text{-}ideal}}$ which connects the dummy machines with their counterparts in the (now simulated) real network. Modify $\mathcal{F}_{\mathsf{fw\text{-}ideal}}$ step-wise:

1. Introduce variables to keep state for the firewalls. When receiving (input, m) through hw_k, evaluate the firewall functionalities $\mathrm{F}_{\mathrm{fw}_1}$ and $\mathrm{F}_{\mathrm{fw}_2}$, update the respective firewall states and save the output packets p_1 and p_2 in a list Q_k as $(\mathsf{in}, 1, p_1, 2\delta)$ and $(\mathsf{in}, 2, p_2, 2\delta)$. This modification stores additional information but does not alter the communication and is thus indistinguishable.
2. When being advised to output a message p for a party hw_k by the simulator, only do so if there is an entry (in, i, p, d) in Q_k and delete that entry. Every message scheduled by the simulator in this manner was output by one of the firewalls in its simulation. Consequently, this message is also stored in Q_k. The real protocol party fw_k will internally delete all messages it outputs. Thus, this modification is indistinguishable.

The ideal functionality of the firewall architecture $\mathcal{F}^{\delta}_{\mathsf{fw\text{-}ideal}}$

Maintain a list of scheduled packets for each direction: $\mathsf{Q}_1, \mathsf{Q}_2$. Let w.l.o.g fw_3 be the corrupted party. In each case, if there are multiple entries to choose from, pick the first.

- Upon receiving $(\mathsf{input}, m, \mathrm{tc}_k)$ from $\mathcal{F}_{\mathsf{wrap}}$: Compute the firewall functions and update the internal states. Let the outputs of fw_1 and fw_2 be p' and p''. Store $(\mathsf{in}, 1, p', 2\delta)$ and $(\mathsf{in}, 2, p'', 2\delta)$ in Q_k if there is no entry $(\mathsf{missing}, 1, p', 0)$ or $(\mathsf{missing}, 2, p'', 0)$ respectively. Send $(\mathsf{input}, m, \mathrm{tc}_k)$ to the adversary.
- Upon receiving $(\mathsf{output}, \mathrm{tc}_k)$ from $\mathcal{F}_{\mathsf{wrap}}$:
  - If there are two entries $(\mathsf{in}, 1, p', 0)$ and $(\mathsf{in}, 2, p', 0)$ in Q_k, erase the corresponding entries from the queue and output p' to tc_k.
  - Else: if there is an entry $(\mathsf{deliver}, i, p, d)$ in Q_k remove it. Check if there is another entry (in, i', p, d') in Q_k with $i \neq i'$. If so, remove that entry too, if not, add an entry $(\mathsf{missing}, |i-3|, p, 0)$ to Q_k.
- Upon receiving $(\mathsf{RoundComplete})$ from $\mathcal{F}_{\mathsf{wrap}}$: Replace each entry (in, i, p, d) (or $\mathsf{deliver}, i, p, d$) with $d > 0$ in Q with $(\mathsf{in}, i, p, d-1)$ (or $(\mathsf{deliver}, i, p, d)$ and send $(\mathsf{RoundComplete})$ to the adversary.
- Upon receiving $(\mathsf{output}, \mathsf{p}, \mathrm{tc}_k)$ from the adversary: if there is an entry (in, i, p, d) in Q_k, replace it by $(\mathsf{deliver}, i, p, d)$.

Figure 4.18: The ideal functionality of the three firewall architecture expressed using the improved model.

3. When a packet p is output based on any entry $(\ldots, i, p, d)$ in Q_k, check if there is another entry $(\ldots, j, p, d)$ with $i \neq j$. If so, delete that entry as well. If not, add an entry $(\mathsf{missing}, |i-3|, p, d)$ to Q_k. Further, when receiving (input, m) through hw_k and evaluating the firewall functionalities, before saving the resulting packets p_1 and p_2, check if there is an entry $(\mathsf{missing}, 1, p_1, 2\delta)$ or $(\mathsf{missing}, 2, p_2, 2\delta)$ in Q_k. If there is, remove that entry and do not save the resulting packet. This modification is indistinguishable as $\mathcal{F}_{\mathsf{fw\text{-}ideal}}$ now implements the exact behaviour of hw_1 and hw_2.

4. Add $\mathcal{F}_{\mathsf{wrap}}$ as a wrapper around $\mathcal{F}_{\mathsf{fw\text{-}ideal}}$. When receiving $(\mathsf{RoundComplete})$ from $\mathcal{F}_{\mathsf{wrap}}$, decrease the delay value d of each entry in Q_1 and Q_2 by 1. Send $(\mathsf{RoundComplete})$ to the simulator. When being advised to output a packet p for party hw_k by the simulator, instead of outputting the packet immediately, replace the corresponding entry in Q_k by $(\mathsf{deliver}, i, p, d)$. When being asked to provide output for party hw_j by $\mathcal{F}_{\mathsf{wrap}}$, check if there is an entry in Q_j with $d = 0$. If so, output that packet. If not, check if there is an entry marked for delivery. If so, output the corresponding packet. Always perform the output according to the mechanism described in Step 3.

The simulator's simulation of the real network is not perfect after transformation step 4. Concretely, $\mathcal{S}$ is not notified of the fourth activation ("output") of honest protocol parties. However, as we argued in the proof of Theorem 26, the output *decision* is made during prior activations. Hence, by $\mathcal{S}$ announcing output early to $\mathcal{F}_{\mathsf{fw\text{-}ideal}}$, $\mathcal{S}$ and $\mathcal{F}_{\mathsf{fw\text{-}ideal}}$ perfectly emulate the real protocol. ($\mathcal{F}_{\mathsf{wrap}}$ delivers output after the fourth activation only.) ■

Chapter 5

Modeling Electronic Payment

The following chapter is based on joint work with Dirk Achenbach, Timon Hackenjos, Alexander Koch, Bernhard Löwe, Jeremias Mechler and Jörn Müller-Quade. Parts of the included content have already been presented in the following work:

- Dirk Achenbach, Roland Gröll, Timon Hackenjos, Alexander Koch, Bernhard Löwe, Jeremias Mechler, Jörn Müller-Quade, Jochen Rill: *Your Money or Your Life—Modeling and Analyzing the Security of Electronic Payment in the UC framework.* Financial Cryptography 2019 [7]
- Alexander Koch: *Cryptographic protocols from physical assumptions.* PhD Thesis 2019 [77]

5.1 Introduction

"Your money, or your life!"—surrender your belongings or face death. This threat was used by bandits in England until the 19th century [85]. As people often needed to carry all their valuables with them when traveling, banditry was a lucrative (albeit dangerous) endeavor. Today, electronic money transfer (EMT) systems alleviate the need to have one's valuables at hand, but introduce new threats as well. Instead of resorting to violence, modern thieves may compromise their victim's bank account. Once they are widely deployed, insecure EMT systems are notoriously difficult to transition away from—magnetic stripes are still in use today. The current state-of-the-art payment standard EMV (short for *Europay International, MasterCard and VISA*, also known as "Chip and PIN") improves on this, but falls short of providing a secure solution to payment (or money withdrawal), as shown by its many weaknesses described in literature.

Among these are practical attacks, such as 1. "cloning" chip cards by pre-computing transaction messages (so-called "pre-play attacks") [18], 2. dis-

abling the personal identification number (PIN) verification of stolen cards by intercepting the communication between chip card and POS device [84], 3. tricking an innocent customer into accepting fraudulent transactions by relaying transaction data from a different POS (so-called "relay attacks") [43].

Upon close examination of these attacks one finds that these issues mainly stem from *two major false assumptions* which are baked into the design of the EMV protocol: 1. that the communication between all protocol participants (e.g. between the chip card and the POS) cannot be intercepted and 2. that the POS (or the ATM) itself is trustworthy. Even though these assumptions are critical for the security of EMV, they are not explicitly stated in the standardization documents [45, 46, 47]. We suggest that this is mainly because EMV has been created by a functionality-focused engineering process in which problems are fixed as they occur and features are added when necessary, rather than a design process that uses formal models and techniques. Modern cryptographic protocols in contrast are designed by first providing a formal description of the protocol, explicitly stating all necessary assumptions and then giving a proof of security. This does not make cryptographic protocols unbreakable, but it does make their potential breaking points explicit. Therefore, we argue that it is necessary to start developing electronic payment protocols by using the same methodology of rigorous formal modeling as has already been established in cryptography.

5.2 Related Work

Secure Human-Server Communication Basin et al. give an enumeration of minimal topologies of channels between a human (restricted in its abilities), a trusted server, a possibly corrupted intermediary and a trusted device, that realize an authenticated channel between the human and the server. There, the authors model the setting of four entities, namely a human (restricted in its abilities), a trusted server, a possibly corrupted intermediate platform and an additional trusted device, such as a user's smartphone, to which the human may have direct secure channels. They give a complete enumeration of the minimal channel topologies between these entities that achieve an authenticated channel between the human and the server. Our work differs in two main aspects: Their model uses either fully secure or untrusted channels only and cannot account for just authenticated or just confidential communication, which is important in our setting due to the presence of CCTV cameras or shoulder-surfing. For example, we assume that everything displayed at the ATM or a user's smartphone is not confidential, while entering a PIN at the PIN pad can be done in a confidential way, by suitably covering the pad in the process. Second, our model is based on the UC framework, which gives stronger guarantees and composability, as well as security for concurrent and interleaved execution, compared to the

stand-alone setting they consider.

Alternative Hardware Assumptions As we will see later in Section 5.3.2, the *confirmation of payment information* by the user is an important sub-problem we aim to solve for achieving secure payment. A possible solution is "Display TAN" [19] providing a smartcard with a display to show the transaction data. However, wide-spread adoption is still not in reach anytime soon. Smart-Guard [40] uses such smartcards with a display together with an encrypting keyboard fixed to the card to achieve a functionality which may be used for payment. These strong hardware assumptions allow for flexible trust assumptions, accounting for several combinations of trusted/hacked status of the involved devices. Their protocol comes with a formally verified security proof, albeit not in the UC framework. For our construction we do not propose a new kind of hardware device, but rely on the user's smartphone.

Ecash and Cryptocurrencies Besides human-server payment protocols, there is also electronic cash, first invented by [30], and modern decentralized cryptocurrencies, such as Bitcoin [52], which can be used to transfer money. In general, these have very different design goals, as they care to establish an electronic money system with certain anonymity/pseudonymity properties, without the possibility to double-spend and in particular, without a trusted bank. In contrast, we are concerned with the authenticated transmission of the transaction data from a human user to the bank. To the best of our knowledge, there is no UC-based model of electronic payment as presented in our work.

There are many different forms of electronic payment with differing goals, such as electronic cash schemes and cryptocurrencies. It is important to note that, beside the correct and extensive formal modeling of payment, our work focuses on the secure establishment of what can be described as an authenticated channel of a human user to the bank via certain devices and channels (plus the extra needed to account for money transfer), to overcome the many problems of EMV-based payment. In particular, we do not model the amount of money, or the money being electronically transferred to other users. For electronic cash systems with formal security proofs, see e.g. [30, 13, 16, 10, 61]. Trolin [96] models ecash with an ideal UC functionality, which may issue/spend coins, and check for double-spending.

EMV. EMV is not only a single payment protocol, but a complete protocol suite for electronic payment (cf. [45, 46, 47]). With more than 700 pages for the basic specification and 4000 additional pages for the VISA specification alone, EMV is sometimes criticized as too complex [8]. Protocols that are EMV-compliant might just implement the EMV interface while using another secure protocol. This means that, while there are multiple attacks against

the EMV *payment protocol*, not every protocol with EMV in its name is automatically insecure. In addition to the attacks mentioned previously, there are other attacks as described by Chothia et al. [32] and Emms et al. [44].

Anderson et al. [9] discuss whether EMV is a monolithic system, even reducing the possibilities for innovation. Since we use the UC framework for our model, we inherently support non-monolithic, modular systems. Sub-protocols that UC-realize each other can be exchanged for one another. Furthermore, [9] explore the possibility to use smartcards (as used by EMV) for other applications. Following a similar goal, we give a formalization of signature cards within our model and show limits to using such cards, see Section 5.4.2.

Degabriele et al. [39] investigate the joint security of encryption and signatures in EMV using the same key-pair. A scheme based on elliptic curves (as it is used in EMV) is proven secure in their model. However, as they conclude, their proof does not eliminate certain kinds of protocol-level attacks. Cortier et al. [35] present an EMV-compliant protocol using trusted enclaves and prove the security of their protocol using TAMARIN [95]. Both approaches lack the modularity, composability and security for parallel execution provided by the UC framework.

5.3 A Formal Model for Electronic Payment

As a basis for our model, observe the process of withdrawing cash at an automated teller machine (ATM). First, there is the bank and its customer, Alice. Second, there is the money dispensing unit inside the ATM. Assuming authenticated communication from Alice to the bank and from the bank to the money dispensing unit, secure payment is easy: Alice communicates the amount of cash she needs and the identity of the money dispensing unit she expects to receive the cash from. The bank then instructs the money dispensing unit to dispense the money. However, Alice is a human and therefore cannot perform cryptographic operations required for a classical channel establishment protocol. Thus, Alice needs another party which offers a user interface to her and communicates with the bank, namely an ATM.

This does not only apply to cash withdrawal but can be extended to electronic money transfer (EMT) in general. To this end, think of Alice as the *initiator* of a transaction and the money dispensing unit as the *receiver*. The process of money withdrawal can now be framed as a payment of money from Alice's account to the account of the money dispensing unit (which, upon receiving money, promptly outputs cash) using the ATM as an (input) *device*. The same works for the point of sale: here, the device's owner (e.g. the supermarket) is the receiver. The model must not be restricted to one initiator and one receiver however, but instead generally must allow for multiple initiators and receivers. This is important to be able to capture

attacks where an adversary attempts to relay a transaction to a different receiver than originally intended. A corrupted ATM could, for example, try to relay a message intended for one money dispensing unit to a different one (a different receiver) in another ATM and thus let a different person collect Alice's money.

Regarding our adversarial model, as discussed earlier, we make no assumption about the trustworthiness of the ATM whatsoever and do assume that the adversary has control over all communication. We do make certain assumptions regarding the trustworthiness of different protocol participants. First, we assume the money dispensing unit (or receiver in general) to be trusted. If it is under adversarial control, the adversary could simply dispense money at will. Second, since our work focuses on the challenges that arise from the interaction of humans with untrustworthy devices over insecure communication, we do not model the bank's book-keeping and therefore assume the bank to be incorruptible. Third, for reasons of simplicity, our model only considers a single bank, even though in practice most transactions involve at least two banks. This is justified, however, as banks in general can communicate securely with each other.

5.3.1 Modeling Electronic Payment in the UC framework

In the following, to simplify the model, we consider the case of *static corruption*, where parties may only be corrupted prior to protocol execution. Extending our work to adaptive corruption is left for future work.

We denote the set of initiators as S_{I}, the set of receivers as S_{R}, the set of devices as S_{D} and the bank as B. We also define a mapping $D\colon S_{\mathrm{R}} \to S_{\mathrm{D}}$ of receivers to single devices ($D(R)$) to explicitly name which device belongs to which receiver.

In order to model the adversary's probability of successfully attacking credentials like PINs, we introduce a parametrized distribution $\mathcal{D}$. Let X denote the event of a successful attack. Then $\mathcal{D} : A \to F_X$ maps a value d (e.g. the amount) from a domain A (e.g. $\mathbb{Q}$) to a probability mass function $f_{d,X} \in F_X$ over $\{\texttt{confirm}, \texttt{reject}\}$. An adversary's success probability of correctly guessing a four-digit PIN chosen uniformly at random with one try could be modeled as follows: $\mathcal{D}(m_\$) = f_X$ for all $m_\$ \in \mathbb{Q}$ with $f_X(\texttt{confirm}) = \frac{1}{10000}$, $f_X(\texttt{reject}) = \frac{9999}{10000}$. $\mathcal{D}$ could also map different $d \in A$ to different $f_{X,d}$, modeling that transactions with small amounts require less protection than ones with bigger amounts. $\mathcal{F}_{\mathcal{D}}$ is the ideal functionality $\mathcal{F}$ parametrized with $\mathcal{D}$. Ideal functionalities may have additional parameters, either implicit or explicit ones passed as arguments, e.g. $\mathcal{F}_{\mathcal{D}}(A, B)$.

In the best possible scenario, ideal payment would work as follows: the initiator submits his desired transaction data to an ideal functionality, which then notifies the bank and the receiver about who paid which amount of money to whom without involvement of the adversary whatsoever. In our

adversarial model, no payment protocol realizes this strong ideal functionality: an attacker who controls all communication will at least be able to observe that a transaction takes place, even if he cannot see or change its contents. What is more, such a strict security definition would ignore the fact that in all payment protocols which rely on the initiator being protected by a short secret (like a PIN), an attacker always has a small chance of success by guessing the secret correctly.

Our ideal functionality for electronic payment is thus designed with regards to the following principles: 1. The adversary always gains access to all transaction data. An electronic payment operation can be secure (that is all participants of the transaction get notified about the correct and non-manipulated transaction data) without the transaction data being secret. 2. The adversary can always successfully change the transaction data at will with a small probability (e.g. if he guesses the PIN correctly). 3. The payment operation occurs in three stages. In the first stage, the initiator inputs his intended transaction data which the adversary can change at will. This models that a corrupted input device will always be able to change the human initiator's transaction data, even if it will be detected at a later stage. In the second and third stage, the receiver and the bank are notified about the transaction data. The resulting functionality is depicted in Figure 5.1.

5.3.2 Confirmation is Key

Since the human initiator of a transaction can never be sure that an input device correctly processes his transaction data, he needs a way of confirming the transaction data with the bank before the transaction is processed. We formalize this confirmation mechanism within the ideal functionality $\mathcal{F}_{\text{CONF}}$ (specified in Figure 5.2). $\mathcal{F}_{\text{CONF}}$ is a two-party functionality which allows a sender to transmit a message and the receiver of the message to *confirm* or *reject* it. As with the ideal payment functionality, the adversary gets the chance to force a confirmation with a certain probability, modeling the insecurity inherent to real-world protocols which use short secrets. Note that he can always force the confirmation to be rejected.

To realize $\mathcal{F}_{\text{PAY}}$, we need authenticated communication from the bank to the receiver, so that the receiver can be notified of the transaction. For most real-world payment protocols, this authenticated communication is easy to establish, since receivers are electronic devices and not humans. In the case of cash withdrawal, the bank owns the money dispensing unit and can pre-distribute cryptographic keys to establish authenticated communication.

Using $\mathcal{F}_{\text{CONF}}$ and $\mathcal{F}_{\text{AUTH}}$ [21], we propose a protocol π_{PAY} which realizes $\mathcal{F}_{\text{PAY}}$. This protocol is informally depicted in Figure 5.3.

Theorem 28. *Let I, B, R, and $D(R)$ ITMs, where I is human, and B and R are honest. Then, π_{PAY}, informally depicted in Figure 5.3, UC-*

The Ideal Functionality for Electronic Payment $\mathcal{F}_{\mathrm{PAY},\mathcal{D}}(I, B, R)$.

Parametrized by a set of receivers S_{R}, a designated receiver $R \in S_{\mathrm{R}}$, a set of initiators S_{I}, an initiator $I \in S_{\mathrm{I}}$, the bank B and a parametrized distribution $\mathcal{D}$.
Initialize $I' = I$, $R' = R$, *attacked* = `no`.
Assertion: At any time, $I, I' \in S_{\mathrm{I}}$ and $R, R' \in S_{\mathrm{R}}$. If the assertion is violated, halt.

Phase 1: Collecting Information

1. Upon receipt of message $(\texttt{transfer}, sid, R, m_{\$})$ from I: Send $(sid, I, R, m_{\$})$ to the adversary, receive $(sid, I', R', m'_{\$})$ and output $(\texttt{input-received}, sid, I', R', m'_{\$})$ to B.

Phase 2: Confirmation and Execution

2. Resume upon instruction by the adversary.
3. If I' is honest, $(I', R', m'_{\$}) \neq (I, R, m_{\$})$ and *attacked* = `no`, halt.
4. Make a public delayed output of $(\texttt{received}, sid, I', m'_{\$})$ to R'.

Phase 3: Ensuring Consistency

5. Resume upon instruction by the adversary and make a public delayed output of $(\texttt{processed}, sid, I', R', m'_{\$})$ to B. Halt upon confirmation by the adversary.

Attack

- Upon receiving an input $(\texttt{attack}, sid)$ in Phase 2 from the adversary, sample an element $b \in \{\texttt{confirm}, \texttt{reject}\}$ according to $\mathcal{D}(m'_{\$})$. If $b = \texttt{confirm}$, set *attacked* = `yes`, otherwise set *attacked* = `no`. Return $(\texttt{attack}, sid, attacked)$ to the adversary. Ignore all further `attack` queries.

Figure 5.1: The ideal functionality $\mathcal{F}_{\mathrm{PAY}}$ for electronic payment.

The Ideal Functionality for Confirmation $\mathcal{F}_{\text{CONF},\mathcal{D}}(S, C)$

Parameters: The message sender S, the respective confirmer C and a parametrized distribution $\mathcal{D}$.
Initialize *attacked* = `no`, *initiated* = `no`, *completed* = `no`.

- Upon receiving $(\texttt{initiate}, sid, C, m)$ from ITI S, make a public delayed output of $(\texttt{initiate}, sid, S, m)$ to C and set *initiated* = `yes`. Ignore all subsequent `initiate` messages.
- Upon receiving $(\texttt{reply}, sid, S, b)$ from ITI C when *initiated* = `yes`, *completed* = `no` and $b \in \{\texttt{confirm}, \texttt{reject}\}$: Make a public delayed output of $(\texttt{answer}, sid, C, b)$ to S. Upon confirmation from the adversary, set *completed* = `yes` and halt.
- Upon receiving $(\texttt{force-confirm}, sid)$ from the adversary, assert that C is honest, *initiated* = `yes`, *completed* = `no` and *attacked* = `no`. If this holds, set *attacked* = `yes` and sample an element $b \in \{\texttt{confirm}, \texttt{reject}\}$ according to $\mathcal{D}(m)$. If b = `confirm`, set *completed* = `yes` and make a public delayed output of $(\texttt{answer}, sid, C, \texttt{confirm})$ to S and halt upon confirmation by the adversary. Otherwise, return $(\texttt{fail}, sid)$ to the adversary.
- Upon receiving $(\texttt{force-reject}, sid)$ from the adversary, assert that C is honest, *initiated* = `yes`, *completed* = `no` and *attacked* = `no`. If this holds, set *attacked* = `yes` and *completed* = `yes`, make a public delayed output of $(\texttt{answer}, sid, C, \texttt{reject})$ to S and halt upon confirmation by the adversary. Otherwise, return $(\texttt{fail}, sid)$ to the adversary.

Figure 5.2: The ideal functionality for confirmation of messages.

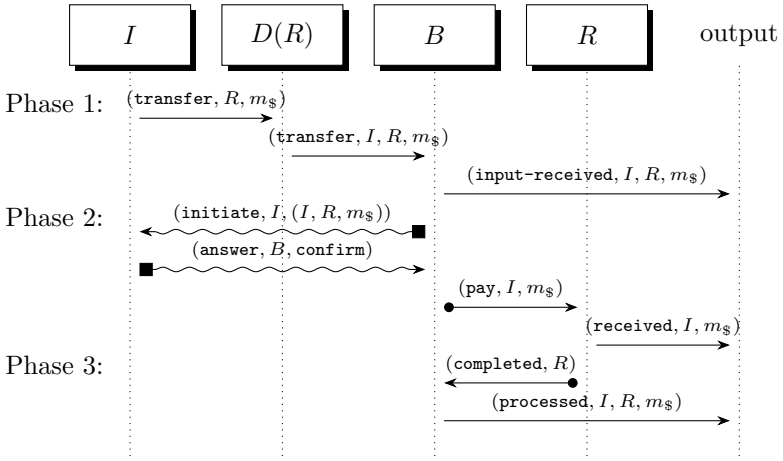


Figure 5.3: The protocol π_{PAY} realizing $\mathcal{F}_{\text{PAY},\mathcal{D}}(I, B, R)$ using $\mathcal{F}_{\text{CONF},\mathcal{D}}(B, I)$, $\mathcal{F}_{\text{AUTH}}(B, R)$ and $\mathcal{F}_{\text{AUTH}}(R, B)$, the latter two depicted as $\bullet\!\!\rightarrow$. The use of an imperfect $\mathcal{F}_{\text{CONF},\mathcal{D}}$ is depicted via $\blacksquare\!\!\rightsquigarrow$. The protocol is between the human initiator I, the ATM $D(R)$, the bank B and the money dispenser R. The protocol proceeds in three phases, namely (1) the information collection phase, (2) the confirmation and execution phase and (3) the phase which ensures a consistent view on what happened.

realizes $\mathcal{F}_{\text{PAY},\mathcal{D}}(I, B, R)$ *in the* $\mathcal{F}_{\text{AUTH}}(B, R), \mathcal{F}_{\text{AUTH}}(R, B), \mathcal{F}_{\text{CONF},\mathcal{D}}(B, I)$*-hybrid model.*

Proof. To proof our statement, we consider a series of hybrid experiments H_i, starting with the real execution between the environment $\mathcal{Z}$, the real-world (dummy) adversary $\mathcal{A}$ and the real protocol parties in H_0. We gradually change the execution until we end with the execution of the ideal protocol for $\mathcal{F}_{\text{PAY}}$ between the environment $\mathcal{Z}$, the simulator $\mathcal{S}$ and dummy parties for $\mathcal{F}_{\text{PAY}}$ in H_5. We give a specific simulator for each of the steps, such that each step is indistinguishable (even perfectly so) from the previous one. Due to the transitivity of indistinguishability, it then follows that the real and ideal execution are perfectly indistinguishable.

Hybrid H_0. This denotes the real execution of π_{PAY} between $\mathcal{Z}$, the adversary $\mathcal{A}$ and real protocol parties.

Hybrid H_1. For H_1 we make the following changes. First, we encapsulate the parties and ideal functionalities from the real protocol within a new Interactive Turing Machine (ITM) which we call $\mathcal{S}_1$ (the simulator). Second, we add a dummy party for each party from the real protocol. These dummy parties interact with an ideal functionality $\mathcal{F}_1$ that relays all inputs to the

simulator and receives back outputs. $\mathcal{S}_1$ just simulates the behavior of the real protocol and performs outputs via $\mathcal{F}_1$ as necessary.

We argue that H_1 and H_2 are perfectly indistinguishable. As the ideal functionality $\mathcal{F}_1$ immediately reports inputs to $\mathcal{S}_1$ and allows it to make arbitrary outputs in the name of all parties, $\mathcal{S}_1$ is able to perfectly simulate the real parties' protocol.

Hybrid H_2. H_2 is identical to H_1, except that $\mathcal{F}_1$ is replaced with the ideal functionality $\mathcal{F}_2$ that works as follows:

- Behave like $\mathcal{F}_{\text{PAY}}$ for Phase 1 (Collecting Information).
- Do not allow direct outputs of `input-received`-messages from $\mathcal{S}_2$ in the name of B.

$\mathcal{S}_2$ is the same as $\mathcal{S}_1$ with the exception that, if I is corrupted, $\mathcal{S}_2$ inputs the (transfer)-message himself upon instruction of $\mathcal{Z}$

We argue that H_2 and H_1 are perfectly indistinguishable. With the restrictions imposed on the simulator by the new ideal functionality, two deviations become possible: either B outputs a different value in H_2 than in H_1 or, when I is corrupted, it does not send the message to start a transfer when instructed. The second case can be handled by $\mathcal{S}_2$. It remains to argue that B's output remains unchanged.

If I inputs a (transfer)-message, $\mathcal{F}_2$ sends $(sid, I, R, m_\$)$ to the simulator which contains all of the original information and allows him to continue his perfect simulation of the real protocol. With this knowledge, $\mathcal{S}_2$ can advise $\mathcal{F}_2$ to output the `input-received`-message to B at the right time.

Hybrid H_3. In H_3 we make the following changes to the ideal functionality:

- Behave like $\mathcal{F}_{\text{PAY}}$ for Phase 2 (Confirmation and Execution).
- Add the attack interface of $\mathcal{F}_{\text{PAY}}$.
- Do not allow direct outputs of `received`-messages in the name of R.

$\mathcal{S}_3$ works exactly as $\mathcal{S}_2$.

We argue that $\mathcal{H}_3$ and $\mathcal{H}_2$ are perfectly indistinguishable. The main difference between the two hybrid models is that the simulator can no longer directly influence the output of the receiver. If the adversary is instructed to perform an attack on the confirmation channel in the real protocol, the simulator can use the attack interface to simulate the attack. This, however, is no problem, since the execution of Phase 2 only resumes upon instruction of the simulator and he can thus instruct an attack beforehand and cause the ideal functionality to accept changed transaction details from Phase 1.

In all other cases, the simulator can simply instruct the ideal functionality to continue, based on the output of the receiver in the simulation of the real protocol. If any changes to I, R or $m_\$$ are necessary, the simulator can do so via the ideal functionality in Phase 1.

The output of the receiver is thus the same in H_2 and H_3.

Hybrid H_4. Hybrid H_4 is identical to H_3, except that $\mathcal{F}_3$ is replaced with $\mathcal{F}_{\text{PAY},\mathcal{D}}$. This means, that the simulator can no longer make outputs of processed-message in the name of B.

We argue that $\mathcal{H}_4$ and $\mathcal{H}_3$ are perfectly indistinguishable. This is trivially the case, since the simulator will only instruct the ideal functionality to deliver the processed-message to B if he sees this message in his simulation of the real protocol. If any changes to I, R or $m_\$$ have been necessary, the simulator would have performed them in the previous stages.

■

Even though this might seem unsurprising at first, this allows to break down the complexity of realizing $\mathcal{F}_{\text{PAY}}$ into two easier problems: realizing a confirmation mechanism between the initiator and the bank and realizing authenticated communication between the receiver and the bank.

5.3.3 How Our Model Captures Existing Attacks

One of our main motivations for establishing a new formal model for electronic payment is to make trust assumptions explicit in order to detect unrealistic ones which enable practical attacks like [18], [84] and [43]. Thus, our model needs to be able to capture these kinds of attacks. Protocols analyzed within our framework must be insecure if they allow for these attacks. In the following, we explain how this is achieved.

Changing Transaction Data. The adversary controlling all communication can easily change transaction data. Protocols which allow this unconditionally are insecure in our model, since $\mathcal{F}_{\text{PAY}}$ only allows to change the transaction data successfully if the adversary mounts a successful attack (i.e. guesses the initiator's PIN in the real world) or the (possibly changed) initiator is corrupted.

Relay Attacks. The aim of a relay attack [43] is to get Alice to authorize an unintended transaction, which benefits the attacker, by relaying legitimate protocol messages between the point of sale (POS) device she uses to pay for goods to another POS device which Alice uses at the same time. If Alice's input device is corrupted, she cannot know with certainty which transaction data she authorizes. Depending on the point of view, this amounts to either changing the *receiver* of a transaction initiated by Alice or changing the

initiator of a transaction initiated by a third party Carol. Thus, in our model, this attack is just a special case of *changing transaction data*.

Pre-Play Attacks. Pre-play attacks [18] basically rely on two facts: 1. once unlocked, smartcards, as used in the EMV protocol, can be coerced into generating message authentication codes (MACs) for arbitrary transaction messages and 2. that even honest ATMs use predictable "unpredictable numbers". Cards interacting with a corrupted ATM can be used to easily generate additional MAC tags. This attack can be modeled by using a global smartcard functionality (which we present in Section 5.4.2) within the Generalized Universal Composability (GUC) extension of the basic Universal Composability (UC) framework. In the GUC framework, the environment (and thus indirectly the adversary) can even access the smartcard in the name of *honest* parties in protocol sessions different from the challenge session. Thus, a payment protocol that GUC-realizes $\mathcal{F}_{\text{PAY}}$ must in particular be secure against all kinds of attacks that result from injecting pre-calculated (sensitive) data into other sessions. Protocols which do not prevent these kinds of attacks (e.g. by enforcing some sort of freshness on the protocol messages) cannot be secure in our model.

5.4 Towards Realizing Secure Electronic Payment

The core challenge when realizing $\mathcal{F}_{\text{PAY}}$ is the authenticated transmission of transaction data from the (human) initiator to the bank. This can also be captured formally: the functionality $\mathcal{F}_{\text{PAY}}$ can be used to implement the ideal authenticated communication functionality $\mathcal{F}_{\text{AUTH}}$ between initiator and bank (up to the attack success probability captured by the distribution $\mathcal{D}$) by encoding the message as an amount to be transmitted. We use this insight to establish several guidelines for the design of secure payment protocols: First, we state a *necessary* condition for protocols that realize $\mathcal{F}_{\text{PAY}}$: they must use setup assumptions that are strong enough to realize authenticated communication between the (human) initiator and the bank. Protocol designers can use this condition as an easily checkable criterion for the *insecurity* of payment protocols. Second, we state several setups that are *sufficient* for realizing $\mathcal{F}_{\text{PAY}}$.

Analogous to $\mathcal{F}_{\text{CONF}}$ as shown in Figure 5.2, we define an ideal functionality $\mathcal{F}_{\text{AUTH},\mathcal{D}}$ that allows the adversary to change the message transmitted or force the transmission of messages according to some parametrized probability distribution $\mathcal{D}$.

For the sake of an easier exposition, we consider ideal functionalities like $\mathcal{F}_{\text{AUTH},\mathcal{D}}$ that model the transmission of *only one* message. If multiple messages have to be transmitted over the same "channel", this model does not adequately capture reality, as an adversary would be able to attack *each*

Ideal functionality $\mathcal{F}_{\text{AUTH},\mathcal{D}}(S, R)$

Parametrized by a sender S, a receiver R and a parametrized probability distribution $\mathcal{D}$. Initially, set *initiated* $= 0$, *attacked* $= 0$.

- Upon receiving an input $(\texttt{Send}, S, R, sid, m)$ from ITI S, set *initiated* $= 1$ and generate a public delayed output $(\texttt{Sent}, S, R, sid, m)$ to R and halt.
- Upon receiving $(\texttt{force-send}, sid, m', v)$ from the adversary: If *initiated* $= 1$ and the $(\texttt{Sent}, S, R, sid, m)$ output has already been delivered to R or if *attacked* $= 1$, do nothing. Otherwise, sample $r \leftarrow \mathcal{D}(m')$ and set *attacked* $= 1$. If $r = v$, output $(\texttt{Sent}, S, R, sid, m')$ to R and halt. Otherwise, output $(\texttt{fail}, sid)$ to the adversary.

Figure 5.4: Ideal functionality $\mathcal{F}_{\text{AUTH},\mathcal{D}}$.

transmission independently. In this case, ideal functionalities for channels like $\mathcal{F}_{\text{SC}}$ (cf. [23]) can be adapted the same way.

5.4.1 Requirements for Secure Electronic Payment

In this section, we establish necessary and sufficient criteria for secure electronic payment. Let $\mathcal{F}_{\text{AUTH},\mathcal{D}}(I, R)$ denote the imperfect ideal authenticated communication functionality between parties I and R, and $\mathcal{F}_{\text{SMT},\mathcal{D}}(I, R)$ the corresponding ideal secure message transfer functionality (where correct guessing according to $\mathcal{D}$ results in loss of secrecy and authenticity).

Throughout this section, let I, B, R be ITMs, where I is human[1], B is honest and $\mathcal{D}$ a parametrized distribution.

First, we can show that $\mathcal{F}_{\text{PAY}}$ can be used to realize $\mathcal{F}_{\text{AUTH},\mathcal{D}}$ and vice-versa.

Theorem 29. *There exists a protocol that UC-realizes $\mathcal{F}_{\text{AUTH},\mathcal{D}}(I, R)$ in the $\mathcal{F}_{\text{PAY},\mathcal{D}}(I, \star, R)$-hybrid model, where $\star$ is an arbitrary protocol party.*

Proof (Sketch). First, we outline the description of π. Upon receiving input $(\texttt{Send}, S, R, sid, m)$, S starts an interaction with $\mathcal{F}_{\text{PAY}}$ where S acts as initiator, receiver and device while R acts as the bank. S sends $(\texttt{transfer}, sid', S, m)$ to $\mathcal{F}_{\text{PAY}}$. When receiving $(\texttt{processed}, sid', S, R, m)$, R outputs $(\texttt{Sent}, I, sid, m')$.

We now consider how to simulate when $\mathcal{Z}$ has changed either the sender or the amount (corresponding to the message). In case $\mathcal{F}_{\text{AUTH},\mathcal{D}}$ asks $\mathcal{S}$

[1]Note that our results hold for arbitrary I.

for confirmation about the delayed output, do nothing. When receiving $(\texttt{attack}, v)$ from $\mathcal{Z}$, $\mathcal{S}$ sends $(\texttt{force-send}, sid, amount', v)$ to $\mathcal{F}_{\text{AUTH},\mathcal{D}}$. If the output is $(\texttt{fail}, sid)$, $\mathcal{S}$ reports $(\texttt{fail}, sid)$ to $\mathcal{Z}$. If the attack is successful, B outputs the correct value in the interaction with $\mathcal{F}_{\text{PAY}}$.

■

Note that an authenticated channel between R and S (corresponding to bank and receiver) is not necessary as π executes $\mathcal{F}_{\text{PAY}}$ only until the first message to R (corresponding to the bank) is sent.

Theorem 30. *There exists a protocol π in the $\mathcal{F}_{\text{AUTH},\mathcal{D}}(I, B), \mathcal{F}_{\text{AUTH}}(B, R)$-hybrid model that UC-realizes $\mathcal{F}_{\text{PAY}}(I, R, B, \mathcal{D})$ for uncorrupted B.*

Proof (Sketch). We can use a modified version of π_{PAY} as π. We describe the necessary changes. Starting with π_{PAY} as shown in Figure 5.3, we first replace Phase I with a message $(sid, P_i, P_k, m_{\$})$ over $\mathcal{F}_{\text{AUTH},\mathcal{D}}$ from P_i to B. We also omit Steps 1-2 in Phase II and abide to the protocol description for the remaining protocol. ■

In particular, Theorem 29 implies that protocols without any authenticated communication or only between the bank and the receiver cannot realize $\mathcal{F}_{\text{PAY}}$:

Corollary 2. *Let π be a protocol that is in the $\mathcal{F}_{\text{AUTH}}(B, R)$, $\mathcal{F}_{\text{AUTH}}(R, B)$-hybrid model only (in particular, there is no authenticated communication between I and B). Then there is no protocol ρ in the bare model such that ρ^{π} UC-realizes $\mathcal{F}_{\text{PAY},\mathcal{D}}(I, B, R)$ if $\mathcal{D}$ admits the adversary at least a non-negligible successful attack probability.*

This insight can be generalized and gives a *necessary condition*: A protocol π that realizes $\mathcal{F}_{\text{PAY},\mathcal{D}}(I, B, R)$ must use setups that can be used to realize $\mathcal{F}_{\text{AUTH},\mathcal{D}}(I, B)$.

Theorem 31 (Necessary Requirements)**.** *Let F be a set of ideal functionalities. Let Π be the set of all subroutine-respecting protocols with the set of protocol parties $P \subseteq \{I, R, B\}$ that use only ideal functionalities in F. If there is no protocol $\pi \in \Pi$ such that π^F realizes $\mathcal{F}_{\text{AUTH},\mathcal{D}}(I, B)$, then there is no protocol $\rho \in \Pi$ such that ρ^F realizes $\mathcal{F}_{\text{PAY},\mathcal{D}}(I, B, R)$.*

Theorem 31 can be easily shown using Theorem 29 and the UC composition theorem:

Proof. Let $F' \subseteq F$. Suppose for the sake of contradiction that there exists a protocol $\rho \in \Pi$ such that $\rho^{F'} \geq \mathcal{F}_{\text{PAY}}(I, R, B, \mathcal{D})$ and for all protocols $\pi \in \Pi$, it holds that $\pi^{F'} \not\geq \mathcal{F}_{\text{AUTH},\mathcal{D}}(I, B)$. By Theorem 29, there exists a protocol $\tau \in \Pi$ that UC-realizes $\mathcal{F}_{\text{AUTH},\mathcal{D}}(I, B)$ in the $\mathcal{F}_{\text{PAY}}(I, R, B, \mathcal{D})$-hybrid model. By the UC composition theorem, it follows that $\tau^{\rho^{F'}} \geq \mathcal{F}_{\text{AUTH},\mathcal{D}}(I, R)$. By

setting $\pi := \tau^{\rho}$, we obtain a contradiction that there is no protocol $\pi \in \Pi$ such that $\pi^{F'}$ UC-realizes $\mathcal{F}_{\text{AUTH},\mathcal{D}}(I, B)$. ■

Conversely, it is easy to see that $\mathcal{F}_{\text{PAY},\mathcal{D}}$ can be realized by (also) using e.g. $\mathcal{F}_{\text{AUTH},\mathcal{D}}(I, B)$. We state several sufficient requirements in the following theorem:

Theorem 32 (Sufficient Requirements)**.** *Let π be a protocol that UC-realizes*

1. $\mathcal{F}_{\text{AUTH},\mathcal{D}}(I, B)$, or
2. $\mathcal{F}_{\text{SMT},\mathcal{D}}(B, I)$, or
3. $\mathcal{F}_{\text{CONF},\mathcal{D}}(B, I)$.

Then, there exists a protocol ρ such that ρ^{π} UC-realizes $\mathcal{F}_{\text{PAY},\mathcal{D}}(I, B, R)$ in the $\mathcal{F}_{\text{AUTH}}(B, R)$, $\mathcal{F}_{\text{AUTH}}(R, B)$-hybrid model.

Proof (Sketch). (1) holds because $\mathcal{F}_{\text{AUTH},\mathcal{D}}(I, B)$ can be used instead of $\mathcal{F}_{\text{CONF},\mathcal{D}}(B, I)$ in π_{PAY}. (2) holds because $\mathcal{F}_{\text{SMT},\mathcal{D}}(I, B)$ can be used to realize $\mathcal{F}_{\text{AUTH},\mathcal{D}}(I, B)$. (3) follows from Theorem 28. ■

5.4.2 No Authentication Using Smartcards Without Additional Trust

By default, EMV uses smartcards containing shared secrets with the bank in order to authenticate transactions. However, this only works if the input device which accesses the smartcard (e.g. the automated teller machine (ATM)) can be trusted. Otherwise, after the initiator enters his personal identification number (PIN) to authorize a seemingly legitimate transaction, the input device can present false (transaction) data to the smartcard (cf. e.g. [18]).

In the following we prove the intuition that smartcards are not sufficient for realizing $\mathcal{F}_{\text{PAY}}$. In Figure 5.5, we give a global signature card functionality $\overline{\mathcal{G}}_{\text{SigCard}}$ in the Generalized Universal Composability (GUC) framework, closely modeled after $\overline{\mathcal{G}}_{\text{cert}}$ as defined by Hofheinz, Müller-Quade, and Unruh [64]. In complete analogy to $\mathcal{F}_{\text{cert-auth}}$ [64], we also define an ideal functionality $\mathcal{F}_{\text{sig-auth}}$ which uses $\overline{\mathcal{G}}_{\text{SigCard}}$ instead of $\overline{\mathcal{G}}_{\text{cert}}$ as shared functionality. We omit its formal description here, as it is very similar to the original version.

We can now prove that $\overline{\mathcal{G}}_{\text{SigCard}}$ cannot be used to realize any authenticated communication, if one participant is human.

Theorem 33. *In the setting of human-server communication, there is no protocol π that GUC-realizes $\mathcal{F}_{\text{sig-auth}}$ in the $\overline{\mathcal{G}}_{\text{SigCard}}$-hybrid model.*

Proof (Sketch). Consider a protocol $\phi_{\text{sig-auth}}$ which realizes $\mathcal{F}_{\text{sig-auth}}$ using $\overline{\mathcal{G}}_{\text{SigCard}}$ In order to authentically send a message m to the receiver, the

Global Signature Card Functionality $\overline{\mathcal{G}}_{\text{SigCard}}$.

Implicitly parametrized by a security parameter λ and an EUF-CMA-secure signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Vfy})$. Let $seized, vk, sk$ denote arrays with default value $\perp$.

Conventions Whenever receiving input of the form $(\star, sid, \star)$ from a party P with identity (PID, SID), check that $SID = sid$. In case of a mismatch, ignore the query. Inputs from sub-parties are handled as being from the respective main party.

Seize and Release On input $(\texttt{seize}, sid)$ from P_i, set $seized[P_i] = sid$ if $seized[P_i] = \perp$ and return $(\texttt{seized}, sid)$ to P_i. Otherwise, ignore the request. On input $(\texttt{release}, sid)$ from P_i, set $seized[P_i] = \perp$ if $seized[P_i] = sid$ and return $(\texttt{released}, sid)$ to P_i. Otherwise, ignore the request.

Initialization On input $(\texttt{KeyGen}, sid)$ from P_i, check if $seized[P_i] = sid$. Otherwise, abort. If $vk[P_i] \neq \perp$, also abort. Generate $(sk_i, vk_i) \leftarrow \mathsf{Gen}(1^\lambda)$ and set $vk[P_i] = vk_i$ and $sk[P_i] = sk_i$.

Public Key Retrieval On input $(\texttt{PublicKey}, sid, P_j)$ from P_i, make a public delayed output of $(\texttt{PublicKey}, sid, P_j, vk[P_j])$.

Signature Generation On input $(\texttt{Sign}, m, sid)$ from party P_i: If $vk[P_i] = \perp$ or $seized[P_i] \neq sid$, return $\perp$. Otherwise, continue. Compute $\sigma \leftarrow \mathsf{Sig}(sk[P_i], m)$. Output $(\texttt{Signature}, sid, m, \sigma)$.

Figure 5.5: The ideal global signature card functionality $\overline{\mathcal{G}}_{\text{SigCard}}$.

initiator I has to obtain a signature on m from $\overline{\mathcal{G}}_{\text{SigCard}}$. In the setting of human-server interaction, the initiator cannot do this himself must use an immediate party instead, as the human cannot interface with the signature card directly. However, unless the interface device $D(I)$ is trusted, $D(I)$ can change the received m to an arbitrary m'. Thus, while $\phi_{\text{sig-auth}}$ can be used for authenticated communication between $D(I)$ and a recipient R, it cannot be used for authenticated communication between I and R, unless $D(I)$ is trusted. ■

Theorem 34. *There exists no protocol π in the $\overline{\mathcal{G}}_{\text{SigCard}}, \mathcal{F}_{\text{AUTH}}(B,R), \mathcal{F}_{\text{AUTH}}(R,B)$-hybrid model that GUC-realizes $\mathcal{F}_{\text{PAY}}(I,B,R)$ if I is human.*

Proof. In Theorem 33 we established that $\overline{\mathcal{G}}_{\text{SigCard}}$ cannot be used for authenticated communication, if one participant is human. From Theorem 31, we know that establishing authenticated communication between I and B is necessary for any protocol that wants to realize $\mathcal{F}_{\text{PAY}}$. The claim follows directly from these observations. ■

5.4.3 Realistic Assumptions

Protocols build on assumptions to achieve security. However, there often is a huge discrepancy regarding to how realistic these assumptions are. EMV relies on the security of the ATM which is often publicly accessible and offers a large attack surface. Unpatched operating systems and exposed Universal Serial Bus interfaces are only two examples for vulnerabilities that have been exploited successfully. As explained in Section 5.3.2, a secure protocol can be constructed by establishing a confirmation mechanism. However, if the input device is corrupted, an additional device is required.

Such additional devices could for example be transaction authentication number (TAN) generators or smartphones. In principle these allow for the creation of protocols that are secure in our model. However, smartphones, which are increasingly used to replace smartcards, regularly call attention because of vulnerabilities. They are complex systems connected to the Internet and are thus more vulnerable to attacks—especially if they are operated by people without expertise in IT security. However, this dilemma can be resolved by requiring trust in only *one of the two devices*. We call this property 1-of-2 (*one-out-of-two*) security (which is, in the case of authentication, also known as multi-factor authentication). This means that a protocol is still secure if one of the two devices is corrupted, no matter which one of them. We argue that, in addition to realizing $\mathcal{F}_{\text{PAY}}$, payment protocols should support this property in order to further reduce the attack surface.

5.5 On the Security of Current Payment Protocols

In this chapter, we use our acquired insights to analyze current protocols for withdrawing cash, paying at the point of sale (POS), and online banking. Table 5.1 summarizes our findings. Our model allows for a structured and fast categorization of payment protocols on a conceptual level, even without a detailed protocol description. Even though EMV is the most widely used standard for payments, we do not elaborate on its security in this chapter. As mentioned before, its design incorporates at least two assumptions that do not hold, as several attacks have been demonstrated. Current payment protocols such as Google Pay, Apple Pay, Samsung Pay, Microsoft Pay and Garmin Pay provide an app that uses the EMV contactless standard to communicate with existing POS devices via near-field communication [93, 97]. Since they rely on Consumer Device Cardholder Verification Method, the user is authenticated by the mobile device exclusively. Currently, these apps use a personal identification number (PIN), a fingerprint or face recognition and thus do not incorporate a second device such as the POS device for authentication. Therefore the security of the protocol is solely based on the mobile device.

The protocols discussed in this section make additional implicit assumptions, which we believe to be plausible, but want to make explicit. These include the following:

1. An additional trusted device beside the input device. This is a plausible assumption if the device is simple, less so if it is a smartphone. However, using an additional device could enable protocols to be 1-of-2-secure.

2. Authenticated communication between the initiator of a transaction and an additional personal device. This is a realistic assumption, since the initiator owns the device. Likewise the initiator can authenticate themselves to the device, e.g. by unlocking the screen of a mobile device.

3. Confidential communication from the initiator to the automated teller machine (ATM), which can be realized by covering the PIN pad with one's hand if the ATM is not compromised.

4. Confidential communication from the ATM to the bank. This can be realized using public-key cryptography.

In the following, we examine multiple protocols for cash withdrawal and online banking.

Cardless Cash. Cardless Cash [34] is an app-based protocol for cash withdrawal offered by numerous banks in Australia. In its most simple variant, it works as follows: After registration, the app can be used to create a "cash code" by entering the desired amount and a phone number. The

phone number is used to send a PIN via SMS and allows to permit someone else to withdraw cash. To dispense the cash, the PIN has to be entered at the ATM alongside the cash code. The security of the protocol is solely based on the ATM, since all relevant information is entered there and no additional confirmation mechanism is established.

VR-mobileCash. VR-mobileCash [98] is another app-based protocol for cash withdrawal offered by Volks- und Raiffeisenbanken, a German association of banks. Upon registration, the user receives the mobile personal identification number (mPIN), which has to be entered on the ATM later on to confirm a transaction. To withdraw cash, the user has to enter the desired amount into the app. After selecting mobile payment at the ATM, the ATM shows a mobile transaction identification number (mTIN) which has to be entered into the app. The ATM then shows the requested amount and asks the user to enter the mPIN. If the mPIN is correct the ATM dispenses the requested amount of cash.

Although not stated explicitly in the public documentation, the mobile device has to be online during the transaction, as the ATM is informed about the transaction data. If the mobile device is corrupted but the ATM is honest, a user can detect an attack because he has to confirm the transaction by entering the mPIN at the ATM and thus verifies the location of the ATM. However, a *corrupted ATM* can employ a relay attack by displaying the mTIN of another corrupted ATM and forwarding the entered mPIN to it thus allowing the second corrupted ATM to dispense the cash. This could be fixed by adding a serial number imprinted on the ATM which is also displayed in the app after entering the mTIN. Thereby VR-mobileCash could potentially realize $\mathcal{F}_{\text{PAY}}$ and even be 1-of-2-secure.

chipTAN comfort. ChipTAN comfort [88] is a protocol for online banking widely used in Germany. Here, the initiator uses a computer as an input device and possesses two additional personal devices: a transaction authentication number (TAN) generator and a smartcard. The TAN generator is used to confirm transactions and thus realizes a confirmation mechanism. This works as follows: First, a transaction has to be requested in the browser. Then, the banking website shows a flickering code. The user puts the smartcard into the TAN generator and scans the flickering code. After reviewing the transaction data presented on the personal device, he presses a button which reveals a TAN that has to be entered into the website.

This protocol satisfies all requirements for a secure realization of $\mathcal{F}_{\text{PAY}}$ by establishing a confirmation channel that allows a user to detect tampering of the transaction data. What is more, the protocol potentially provides a form of 1-of-2 security, since as long as either the input device or alternatively the TAN generator together with the smartcard are uncorrupted, there exists a

Table 5.1: Comparison of different payment protocols. A protocol is marked as offline, if the additional device does not require an Internet connection during the payment process. The security of a protocol is put in parentheses if it meets our requirements for a secure protocol but has not been proven secure.

Protocol	Offline	Secure	Applicable for
Cardless Cash	✓	×	Withdrawal
VR-mobileCash	×	×	Withdrawal
chipTAN comfort	✓	(✓: 1-of-2)	Online banking
photoTAN	✓	(✓: 1-of-2)	Online banking
L-Pay (our scheme)	✓	✓: 1-of-2	Withdrawal, PoS

confirmation mechanism from the bank to the initiator. This is only true for single transactions, however (see [89] for details). Collective bank transfers using chipTAN comfort have been shown to be insecure [89], because the TAN generator only displays summarized information about the transactions. ChipTAN comfort with collective bank transfers also does not realize $\mathcal{F}_{\text{CONF}}$, because the transaction data submitted over the confirmation channel are incomplete.

photoTAN. PhotoTAN (or QR-TAN) is a variant of chipTAN comfort, where the code to transmit data to the TAN generator is encrypted by the bank. Furthermore, a smartphone can be used as an alternative to a special-purpose TAN generator. In our model, this encryption does not have an impact on security, since the transaction data is not confidential and is displayed on the smartphone nonetheless. However, some banking apps for photoTAN [41, 33] show the TAN immediately after scanning the code and before the transaction data have been confirmed by the user. Thus, in the scenario of cash withdrawal, an attacker that corrupted an ATM and deploys a camera monitoring the ATM could change the submitted transaction data at the ATM, read the TAN from the victim's display and confirm the transaction without the initiator's consent.

5.6 Realizing Secure Electronic Payment

In Section 5.3.2, we gave a protocol π_{PAY} that realizes $\mathcal{F}_{\text{PAY},\mathcal{D}}(I, R, B)$ in the $\mathcal{F}_{\text{AUTH}}(B, R)$, $\mathcal{F}_{\text{AUTH}}(R, B)$, $\mathcal{F}_{\text{CONF},\mathcal{D}}(B, I)$-hybrid model. While realizing $\mathcal{F}_{\text{AUTH}}$ between the bank and the receiver is simple, realizing $\mathcal{F}_{\text{CONF},\mathcal{D}}(B, I)$ in a way suitable for humans is a challenge under realistic trust assumptions.

The protocols in Section 5.5 use one or more additional devices, such as smartphones, smartcards or optical code readers to give the initiator a

confirmation capability. Yet all cash withdrawal protocols still need a trusted automated teller machine (ATM).

In the following, we improve on this by presenting a simple offline protocol called L-Conf (informally described by $\pi_{\text{L-Conf}}$ in Figure 5.6). It is inspired by chipTAN and photoTAN which use similar mechanisms. Our protocol is secure even if either the additional device A, such as the initiator's smartphone, or the input device is compromised. We call this property *one-out-of-two security*, formally defined as follows:

Definition 43 (One-out-of-two security)**.** Let X_1, X_2 be Boolean variables, π a protocol and $\mathcal{F}$ an ideal functionality. We say that π *UC-realizes* $\mathcal{F}$ *with one-out-of-two security relative to* X_1 *and* X_2, if $X_1 \vee X_2$ implies that π UC-realizes $\mathcal{F}$.

$\pi_{\text{L-Conf}}$ can be used with π_{PAY} to realize $\mathcal{F}_{\text{PAY}}$. We call the resulting protocol L-Pay. The protocol starts with a setup phase: The bank B and the initiator I agree on a personal identification number (PIN) and the initiator's smartphone shares keys with the bank for an authenticated secret-key encryption scheme.

The main part, depicted in Figure 5.6, consists of the execution of two protocols π_1 and π_2, each realizing $\mathcal{F}_{\text{CONF}}(B, I)$ under different assumptions. By combining their results, the composed protocol $\pi_{\text{L-Conf}}$ realizes $\mathcal{F}_{\text{CONF}}(B, I)$ even if either the input device or the additional device is compromised.

In π_1, the bank first encrypts the transaction data together with a fresh one-time transaction authentication number (TAN). The ciphertext is then transmitted to the initiator's input device, displayed appropriately, transferred to the smartphone (e.g. by scanning a QR code) and is decrypted. The TAN is only shown after the transaction data has been checked and *explicitly confirmed* by the initiator. Afterwards, the initiator enters the TAN into the input device.

In order to achieve security even if the initiator's smartphone is corrupted, π_2 requires the initiator to also check and confirm the transaction by entering his PIN into the input device (confidentially over $\mathcal{F}_{\text{Confid}}$), which is then sent to the bank confidentially. Only if the bank receives both the correct TAN and PIN, it considers the transaction to be confirmed. Now, if only the initiator's smartphone is corrupted, the adversary is able to present false transaction data to them or even to perform the confirmation himself. However, this would be noticed immediately, since the transaction data shown on the input device would be wrong and the initiator would not enter his PIN. Conversely, if only the input device is malicious and displays wrong transaction data, the initiator will notice this using their smartphone.

Theorem 35. *Let I, B, $D(R)$ and A be ITMs, where I is human. Let S be the domain of $\mathcal{D}_1, \mathcal{D}_2$, let π_1 UC-realize $\mathcal{F}_{\text{CONF},\mathcal{D}_1}(B, I)$ if A is honest*

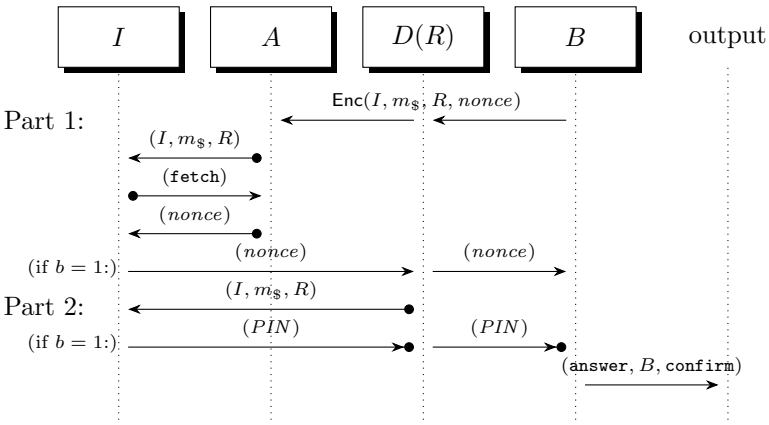


Figure 5.6: Main phase of $\pi_{\text{L-Conf}}$ realizing $\mathcal{F}_{\text{CONF},\mathcal{D}}(B, I)$ using authenticated and confidential channels drawn as $\bullet\!\!\longrightarrow$ and $\longrightarrow\!\!\bullet$, resp. The protocol is between the human initiator I, his personal device A, the ATM $D(R)$ and the bank B. The bit $b \in \{0, 1\}$ indicates, whether I wants to confirm, hence $(nonce)$ and (PIN) are only sent in this case.

and let π_2 *UC-realize* $\mathcal{F}_{\text{CONF},\mathcal{D}_2}(B, I)$ *if* $D(R)$ *is honest. Then,* $\pi_{\textit{L-Conf}}$ *UC-realizes* $\mathcal{F}_{\text{CONF},\mathcal{D}_3}(B, I)$ *in the* $\mathcal{F}_{\text{AUTH}}(A, I)$, $\mathcal{F}_{\text{AUTH}}(I, A)$, $\mathcal{F}_{\text{AUTH}}(D(R), I)$, $\mathcal{F}_{\text{Confid}}(I, D(R))$, $\mathcal{F}_{\text{Confid}}(D(R), B)$*-hybrid model where for all* $x \in S$*:*

$$\mathcal{D}_3(m_\$)(x) := \begin{cases} \max\left(\mathcal{D}_1(m_\$)(\texttt{confirm}), \mathcal{D}_2(m_\$)(\texttt{confirm})\right) & x = \texttt{accept} \\ 1 - \max(\mathcal{D}_1(m_\$)(\texttt{confirm}), \mathcal{D}_2(m_\$)(\texttt{confirm})) & x = \texttt{reject} \end{cases}$$

Proof (Sketch). The protocol $\pi_{\text{L-Conf}}$ (Figure 5.6) can be interpreted as the sequential composition of two confirmation protocols π_1 (Part 1) and π_2 (Part 2). It holds that π_1 realizes $\mathcal{F}_{\text{CONF}}(B, I)$ if A is honest, and that π_2 realizes $\mathcal{F}_{\text{CONF},\mathcal{D}}(B, I)$ if $D(R)$ is honest (omitting the unnecessary message from B to $D(R)$ to initiate Part 2). Let $b \in \{\texttt{confirm}, \texttt{reject}\}$ denote the initiator's input and let $b_1, b_2 \in \{\texttt{confirm}, \texttt{reject}\}$ denote the outputs of π_1 and π_2 as received by B, respectively. After having received b_1 and b_2, B outputs b', which is `confirm` if $b_1 = b_2 = \texttt{confirm}$, and `reject` otherwise. By definition, $b' = \texttt{confirm}$ while $b = \texttt{reject}$ holds with probability upper-bounded by $\max\left(\mathcal{D}_1(m_\$)(\texttt{confirm}), \mathcal{D}_2(m_\$)(\texttt{confirm})\right)$. Thus, $\pi_{\text{L-Conf}}$ UC-realizes $\mathcal{F}_{\text{CONF},\mathcal{D}_3}(B, I)$ with one-out-of-two security relative to the assumptions that A or $D(R)$ is honest, respectively. ■

Chapter 6

Conclusion

In this thesis, we investigated the question, if and how the cryptographic methodology for the development of secure protocols can be applied to real-world systems. To this end, we considered three different real-world use cases: computer networks, data outsourcing and electronic payment.

Computer Networks Computer networks can be elegantly described within the basic Universal Composability (UC) framework [22]. Because of its composition theorem, which allows to break down complex network structures into smaller components which can be analyzed separately, the UC framework seems especially well-suited to this task.

In Chapter 4 we gave hybrid functionalities to model the network structure and showed how to construct ideal functionalities to describe the intended security properties of the network. We validated these tools by formally analyzing the security of networks consisting of multiple firewalls and specially designed trusted hardware components in the presence of one or more actively malicious firewalls. We found that, the naïve serial concatenation of two firewalls is insecure, whereas the serial concatenation of three firewalls is secure, when firewalls are stateless. The parallel combination of three firewalls is the most secure solution.

If availability guarantees need to be expressed, the extension to the basic UC framework by Katz et al. [75] must be used. This posed a particular challenge, since the extension introduced a lot of modeling artifacts which need to be incorporated. We showed how this can be somewhat mitigated by creating a wrapping layer around ideal functionalities. We gave a generic hybrid functionality to model any network structure as well as a guideline to using Katz et al.'s framework for modeling computer networks. The model was validated by (re-)analyzing a solution to the Byzantine Generals Problem and proving that the parallel combination of three firewalls is available.

In summary, modeling and analyzing real-world computer networks using the UC framework does seem feasible—if one does not need to express

availability. The basic UC framework is expressive enough to capture all required properties of computer networks and easy enough to use (at least for experts). Modeling availability however, is extremely complicated and error-prone (even for experts). A lot more research has to be done in this area, before it becomes really feasible to apply to real-world systems.

Data Outsourcing Even though data outsourcing has been investigated by cryptographic research in the past, there is no common consensus which security model or notion should be used to express protocols in this field. Instead there exist a plethora of security models and notions from which to choose, each with wildly different security goals.

Thus, we first presented a game-based framework that aims to unify the security notions of outsourcing schemes. To this end we precisely defined outsourcing schemes with queries and introduced the security notions *Data Privacy*, *Query Privacy*, and *Result Privacy*. While Data Privacy and Query Privacy capture independent objectives, we showed that both Data Privacy and Query Privacy are necessary for keeping the *results* of a query to an outsourced data set private. We defined generalized versions of these notions to capture constructions with weaker security properties. For validation, we showed how security notions for existing outsourcing schemes fit into our framework.

However, when applied to the cryptographic cloud file system CryFS, the new model turned out to be insufficient. This was mainly because, a particular security goal, which traditionally has not been considered for outsourcing schemes like searchable encryption, and was thus also missing from our unified model, is very important for file systems: integrity. So, we introduced a novel model for the security and integrity of cloud file systems Using this model, we were able to prove the security and integrity of CryFS.

In summary, whether modeling and analyzing real-world outsourcing schemes is feasible or not, mostly depends on the specific use case. Our unified framework is easy to use but can only capture two security goals: data privacy and query privacy. This is enough for most outsourcing schemes from cryptographic research (like searchable encryption) but fails when applied to use cases in which other security properties are important (such as integrity). Our model for security and integrity of cloud file systems, which we developed as a consequence, is also easy to use, generic and designed to be applicable for a wide range of file systems. However, it is easy to imagine that there exist real-world outsourcing schemes for which completely different security properties are important and which would require the development of yet another model.

Electronic Payment As with computer networks, the UC framework is also well-suited for the use for electronic payment, especially because of its

composition theorem.

Payment protocols typically involve a human user who is not capable of performing cryptographic operations and therefore needs an intermediate device (e.g. an automated teller machine (ATM)) to interface with the protocol, which can not always be trusted. In this work we introduced a formal model for the security of such protocols. In particular, we did not assume all intermediate devices as trusted and we specifically considered that case that the initiator of a transaction is human. We gave an ideal functionality for secure electronic payment which can be used as a guideline for the development of future payment protocols. With our model, we then developed a set of basic requirements for electronic payment protocols without which no protocol can be considered secure. We were also able to formalize the intuition that smartcards (such as bank cards) are not sufficient for establishing authenticated channels using the Generalized Universal Composability (GUC) framework [24].

To validate these results, we discussed different current payment protocols and find that most do not realize these requirements. We then specified a protocol called L-Pay (based upon chipTAN and photoTAN), which uses an additional smartphone and which can be proven secure in our model if either the ATM or the smartphone is honest.

In summary, analyzing real-world electronic payment protocols does seem partly feasible. We established a set of requirements for secure electronic payments in general, which can directly be applied to any such protocol as a kind of litmus test for (in-)security. We can use these requirements to reason why EMV in particular is not a secure electronic payment protocol according to our definition. However, modeling EMV directly in our framework seems impossible, since even the basic specification of EMV is over thousand pages long. Simpler electronic payment protocols (such as L-Pay) can directly be modeled within our framework, however.

Directions for Future Research Having an experienced cryptographer in every development team does not seem to be realistic in the foreseeable future. Thus, whether or not cryptographic methodology will be used for real-world protocols in the future will mostly depend on improvements in usability. While having a software supported process is surely the ultimate goal, there are several intermediate steps necessary to get there.

One is to continue research into easy-to-use security models and notions for specific use cases with clear guidelines on how to apply them. Modeling availability guarantees is particularly difficult. While this work has established some tools to make this task easier in one particular use case, more research into simpler models for time-based security guarantees is required before this methodology can really be applied to real-world systems. This could also be used to extend our model for electronic payment with the ability to

model time-based security measures (such as timeouts). Using Katz et al.'s extension for this task does not seem feasible. How to proceed with security models for outsourcing schemes is also an open question. One the one hand, it seems plausible to try to unify our model for data and query privacy with our model for security and integrity of file systems. This would make it easier for protocol designers to chose the right security model. One the other hand, trying to cover too many use cases within one security model could make its application more complex. Thus, another direction for research could be to identify more classes of outsourcing schemes (such as searchable encryption and file systems) and specifically tailor a security model to their needs.

Author's Publications

1. Ingmar Baumgart, Matthias Börsig, Niklas Goerke, Timon Hackenjos, Jochen Rill, Marek Wehmer: *Who Controls Your Energy? On the (In)Security of Residential Battery Energy Storage Systems.* IEEE SmartGridComm 2019 [12].

2. Dirk Achenbach, Roland Gröll, Timon Hackenjos, Alexander Koch, Bernhard Löwe, Jeremias Mechler, Jörn Müller-Quade, Jochen Rill: *Your Money or Your Life—Modeling and Analyzing the Security of Electronic Payment in the UC framework.* Financial Cryptograpy 2019 [7]

3. Sebastian Messmer, Jochen Rill, Dirk Achenbach, Jörn Müller-Quade: *A Novel Cryptographic Framework for Cloud File Systems and CryFS, a Provably-Secure Construction.* DBSec 2017 [83]

4. Dirk Achenbach, Anne Borcherding, Bernhard Löwe, Jörn Müller-Quade, Jochen Rill: *Towards Realising Oblivious Voting* ICETE (Selected Papers) 2016 [6]

5. Dirk Achenbach, Bernhard Löwe, Jörn Müller-Quade, Jochen Rill: *Oblivious Voting: Hiding Votes from the Voting Machine in Bingo Voting.* SECRYPT 2016 [5]

6. Dirk Achenbach, Matthias Huber, Jörn Müller-Quade, Jochen Rill: *Closing the Gap: A Universal Privacy Framework for Outsourced Data.* BalkanCryptSec 2015 [4].

7. Dirk Achenbach, Jörn Müller-Quade, Jochen Rill: *Synchronous Universally Composable Computer Networks.* BalkanCryptSec 2015 [2].

8. Dirk Achenbach, Jörn Müller-Quade, Jochen Rill: *Universally Composable Firewall Architectures Using Trusted Hardware.* BalkanCryptSec 2014 [3].

9. Rolf Haynberg, Jochen Rill, Dirk Achenbach, Jörn Müller-Quade: *Symmetric Searchable Encryption for Exact Pattern Matching using Directed Acyclic Word Graphs.* SECRYPT 2013 [62]

References

[1] Dirk Achenbach. *On provable security for complex systems*. Karlsruhe, 2016. URL: http://dx.doi.org/10.5445/IR/1000052204%20;%20http://nbn-resolving.de/urn:nbn:de:swb:90-522047%20;%20http://d-nb.info/1084112426/34%20;%20http://digbib.ubka.uni-karlsruhe.de/volltexte/1000052204.

[2] Dirk Achenbach, Jörn Müller-Quade, and Jochen Rill. "Synchronous Universally Composable Computer Networks". In: *Cryptography and Information Security in the Balkans - Second International Conference, BalkanCryptSec 2015, Koper, Slovenia, September 3-4, 2015, Revised Selected Papers*. 2015, pp. 95–111. DOI: 10.1007/978-3-319-29172-7_7. URL: http://dx.doi.org/10.1007/978-3-319-29172-7_7.

[3] Dirk Achenbach, Jörn Müller-Quade, and Jochen Rill. "Universally Composable Firewall Architectures using Trusted Hardware". In: *Cryptography and Information Security in the Balkans*. Ed. by Berna Ors and Bart Preneel. Vol. 9024. Lecture Notes in Computer Science. To appear, preprint version online at http://eprint.iacr.org/2015/099.pdf. Springer Berlin Heidelberg, 2015. ISBN: 978-3-319-21355-2.

[4] Dirk Achenbach et al. "Closing the Gap: A Universal Privacy Framework for Outsourced Data". In: *Cryptography and Information Security in the Balkans - Second International Conference, BalkanCryptSec 2015, Koper, Slovenia, September 3-4, 2015, Revised Selected Papers*. 2015, pp. 134–151. DOI: 10.1007/978-3-319-29172-7_9.

[5] Dirk Achenbach et al. "Oblivious Voting: Hiding Votes from the Voting Machine in Bingo Voting". In: *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT, Lisbon, Portugal, July 26-28, 2016*. 2016, pp. 85–96. DOI: 10.5220/0005964300850096. URL: https://doi.org/10.5220/0005964300850096.

[6] Dirk Achenbach et al. "Towards Realising Oblivious Voting". In: *E-Business and Telecommunications - 13th International Joint Conference, ICETE 2016, Lisbon, Portugal, July 26-28, 2016, Revised Selected*

Papers. 2016, pp. 216–240. DOI: 10.1007/978-3-319-67876-4_11. URL: https://doi.org/10.1007/978-3-319-67876-4%5C_11.

[7] Dirk Achenbach et al. "Your Money or Your Life - Modeling and Analyzing the Security of Electronic Payment in the UC Framework". In: *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*. Ed. by Ian Goldberg and Tyler Moore. Vol. 11598. Lecture Notes in Computer Science. Springer, 2019, pp. 243–261. DOI: 10.1007/978-3-030-32101-7_16. URL: https://doi.org/10.1007/978-3-030-32101-7%5C_16.

[8] Ross J. Anderson. *Security engineering – a guide to building dependable distributed systems*. 2nd ed. Wiley, 2008, pp. I–XL, 1–1040. ISBN: 978-0-470-06852-6.

[9] Ross J. Anderson et al. "Might Financial Cryptography Kill Financial Innovation? – The Curious Case of EMV". In: *Financial Cryptography and Data Security, FC 2011*. Ed. by George Danezis. Vol. 7035. LNCS. Springer, 2011, pp. 220–234. ISBN: 978-3-642-27575-3. DOI: 10.1007/978-3-642-27576-0_18.

[10] Foteini Baldimtsi et al. "Anonymous Transferable E-Cash". In: *PKC 2015*. Ed. by Jonathan Katz. Vol. 9020. LNCS. Springer, 2015, pp. 101–124. ISBN: 978-3-662-46446-5. DOI: 10.1007/978-3-662-46447-2_5.

[11] David A. Basin, Sasa Radomirovic, and Michael Schläpfer. "A Complete Characterization of Secure Human-Server Communication". In: *IEEE 28th Computer Security Foundations Symposium, CSF 2015*. Ed. by Cédric Fournet, Michael W. Hicks, and Luca Viganò. IEEE Computer Society, 2015, pp. 199–213. ISBN: 978-1-4673-7538-2. DOI: 10.1109/CSF.2015.21.

[12] I. Baumgart et al. "Who Controls Your Energy? On the (In)Security of Residential Battery Energy Storage Systems". In: *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. 2019, pp. 1–6.

[13] Mira Belenkiy et al. "Compact E-Cash and Simulatable VRFs Revisited". In: *Pairing-Based Cryptography – Pairing 2009*. Ed. by Hovav Shacham and Brent Waters. Vol. 5671. LNCS. Springer, 2009, pp. 114–131. ISBN: 978-3-642-03297-4. DOI: 10.1007/978-3-642-03298-1_9.

[14] Mihir Bellare and Chanathip Namprempre. "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm". In: *Journal of Cryptology* 21.4 (2008), pp. 469–491.

[15] S.M. Bellovin and W.R. Cheswick. "Network firewalls". In: *Communications Magazine, IEEE* 32.9 (1994), pp. 50–57. ISSN: 0163-6804. DOI: 10.1109/35.312843.

[16] Eli Ben-Sasson et al. “Zerocash: Decentralized Anonymous Payments from Bitcoin”. In: 2014, pp. 459–474. DOI: 10.1109/SP.2014.36.

[17] *Bloomberg alleges Huawei routers and network gear are backdoored*. Ars Technica. https://arstechnica.com/gadgets/2019/04/bloomberg-claims-vodafone-found-backdoors-in-huawei-equipment-vodafone-disagrees. Apr. 2019.

[18] Mike Bond et al. “Chip and Skim: Cloning EMV Cards with the Pre-play Attack”. In: 2014, pp. 49–64. DOI: 10.1109/SP.2014.11.

[19] Borchert IT-Sicherheit UG. *Display-TAN Mobile Banking: Secure and Mobile*. 2018. URL: http://www.display-tan.com/ (visited on 09/18/2018).

[20] Christian Cachin, Silvio Micali, and Markus Stadler. “Computationally Private Information Retrieval with Polylogarithmic Communication”. English. In: *Advances in Cryptology — EUROCRYPT ’99*. Ed. by Jacques Stern. Vol. 1592. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1999, pp. 402–414. ISBN: 978-3-540-65889-4. DOI: 10.1007/3-540-48910-X_28. URL: http://dx.doi.org/10.1007/3-540-48910-X_28.

[21] Ran Canetti. “Universally composable security: a new paradigm for cryptographic protocols”. In: *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*. Oct. 2001.

[22] Ran Canetti and Marc Fischlin. “Universally Composable Commitments”. English. In: *Advances in Cryptology – CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2001, pp. 19–40. ISBN: 978-3-540-42456-7. DOI: 10.1007/3-540-44647-8_2. URL: http://dx.doi.org/10.1007/3-540-44647-8_2.

[23] Ran Canetti and Hugo Krawczyk. “Universally Composable Notions of Key Exchange and Secure Channels”. In: *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*. Ed. by Lars R. Knudsen. Vol. 2332. Lecture Notes in Computer Science. Springer, 2002, pp. 337–351. ISBN: 3-540-43553-0. DOI: 10.1007/3-540-46035-7_22. URL: https://doi.org/10.1007/3-540-46035-7%5C_22.

[24] Ran Canetti et al. “Universally Composable Security with Global Setup”. In: *4th Theory of Cryptography Conference, TCC 2007*. Ed. by Salil P. Vadhan. Vol. 4392. LNCS. Springer, 2007, pp. 61–85. ISBN: 3-540-70935-5. DOI: 10.1007/978-3-540-70936-7_4.

[25] David Cash, Alptekin Küpçü, and Daniel Wichs. "Dynamic Proofs of Retrievability Via Oblivious RAM". In: *J. Cryptol.* 30.1 (Jan. 2017), pp. 22–57. ISSN: 0933-2790. DOI: 10.1007/s00145-015-9216-2.

[26] David Cash et al. "Dynamic searchable encryption in very-large databases: Data structures and implementation". In: *Network and Distributed System Security Symposium, NDSS*. Vol. 14. 2014.

[27] David Cash et al. "Highly-scalable searchable symmetric encryption with support for boolean queries". In: *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 353–373.

[28] Suresh Chari, Charanjit S Jutla, and Arnab Roy. "Universally Composable Security Analysis of OAuth v2. 0." In: *IACR Cryptology ePrint Archive* 2011 (2011), p. 526.

[29] Melissa Chase and Emily Shen. *Substring-Searchable Symmetric Encryption*. Cryptology ePrint Archive, Report 2014/638. http://eprint.iacr.org/2014/638. 2014.

[30] David Chaum, Amos Fiat, and Moni Naor. "Untraceable Electronic Cash". In: *CRYPTO '88*. Ed. by Shafi Goldwasser. Vol. 403. LNCS. Springer, 1988, pp. 319–327. ISBN: 3-540-97196-3. DOI: 10.1007/0-387-34799-2_25.

[31] Benny Chor et al. "Private Information Retrieval". In: *J. ACM* 45.6 (Nov. 1998), pp. 965–981. ISSN: 0004-5411. DOI: 10.1145/293347.293350. URL: http://doi.acm.org/10.1145/293347.293350.

[32] Tom Chothia et al. "Relay Cost Bounding for Contactless EMV Payments". In: *Financial Cryptography and Data Security, FC 2015*. Ed. by Rainer Böhme and Tatsuaki Okamoto. Vol. 8975. LNCS. Springer, 2015, pp. 189–206. ISBN: 978-3-662-47853-0. DOI: 10.1007/978-3-662-47854-7_11.

[33] Commerzbank. *Das photoTAN-Lesegerät*. URL: https://www.commerzbank.de/portal/media/a-30-sonstige-medien/pdf/themen/sicherheit-1/Flyer_Lesegeraet.pdf (visited on 12/13/2018).

[34] Commonwealth Bank of Australia. *Cardless Cash*. 2018. URL: https://www.commbank.com.au/digital-banking/cardless-cash.html (visited on 09/25/2018).

[35] Véronique Cortier et al. "Designing and Proving an EMV-Compliant Payment Protocol for Mobile Devices". In: *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017*. IEEE, 2017, pp. 467–480. ISBN: 978-1-5090-5762-7. DOI: 10.1109/EuroSP.2017.19. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7961955.

[36] Reza Curtmola et al. “Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions”. In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. CCS '06. Full version available at https://eprint.iacr.org/2006/210. Alexandria, Virginia, USA: ACM, 2006, pp. 79–88. ISBN: 1-59593-518-5. DOI: 10.1145/1180405.1180417.

[37] Ivan Damgård and Kasper Dupont. *Universally Composable Disk Encryption Schemes*. Cryptology ePrint Archive, Report 2005/333. http://eprint.iacr.org/. 2005.

[38] Ivan Damgård, Sigurd Meldgaard, and Jesper Buus Nielsen. “Perfectly secure oblivious RAM without random oracles”. In: *Theory of Cryptography*. Springer, 2011, pp. 144–163.

[39] Jean Paul Degabriele et al. “On the Joint Security of Encryption and Signature in EMV”. In: *CT-RSA 2012*. Ed. by Orr Dunkelman. Vol. 7178. LNCS. Springer, 2012, pp. 116–135. ISBN: 978-3-642-27953-9. DOI: 10.1007/978-3-642-27954-6_8.

[40] Michael Denzel, Alessandro Bruni, and Mark Dermot Ryan. “SmartGuard: Defending User Input from Malware”. In: *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. IEEE Computer Society, 2016, pp. 502–509. ISBN: 978-1-5090-2771-2. DOI: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0089. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7814735.

[41] Deutsche Bank. *photoTAN – schnell und einfach aktiviert*. URL: https://www.deutsche-bank.de/pfb/data/docs/Photo_TAN_Smartphone_2.pdf (visited on 12/13/2018).

[42] Victor J. Dielissen and Anne Kaldewaij. “A simple, efficient, and flexible implementation of flexible arrays”. In: *Mathematics of Program Construction: Third International Conference, MPC '95 Kloster Irsee, Germany, July 17–21, 1995 Proceedings*. Ed. by Bernhard Möller. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 232–241. ISBN: 978-3-540-49445-4. DOI: 10.1007/3-540-60117-1_13.

[43] Saar Drimer and Steven J. Murdoch. “Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks”. In: *Proceedings of the 16th USENIX Security Symposium 2007*. Ed. by Niels Provos. USENIX Association, 2007. URL: https://www.usenix.org/conference/16th-usenix-security-symposium/keep-your-enemies-close-distance-bounding-against.

[44] Martin Emms et al. “Harvesting High Value Foreign Currency Transactions from EMV Contactless Credit Cards Without the PIN”. In: *2014 ACM SIGSAC Conference on Computer and Communications Security*. Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. ACM, 2014, pp. 716–726. ISBN: 978-1-4503-2957-6. DOI: 10.1145/2660267.2660312. URL: http://dl.acm.org/citation.cfm?id=2660267.

[45] EMV. *Integrated Circuit Card Specifications for Payment Systems: Book 1. Application Independent ICC to Terminal Interface Requirements, Version 4.3*. 2011.

[46] EMV. *Integrated Circuit Card Specifications for Payment Systems: Book 2. Security and Key Management, Version 4.3*. 2011.

[47] EMV. *Integrated Circuit Card Specifications for Payment Systems: Book 3. Application Specification, Version 4.3*. 2011.

[48] Chris Erway et al. “Dynamic Provable Data Possession”. In: *Proceedings of the 16th ACM Conference on Computer and Communications Security*. CCS ’09. Chicago, Illinois, USA: ACM, 2009, pp. 213–222. ISBN: 978-1-60558-894-0. DOI: 10.1145/1653662.1653688.

[49] Sergei Evdokimov, Matthias Fischmann, and Oliver Gunther. “Provable security for outsourcing database operations”. In: *Data Engineering, 2006. ICDE’06. Proceedings of the 22nd International Conference on*. IEEE. 2006, pp. 117–117.

[50] Ned Freed. “Behavior of and requirements for Internet firewalls”. In: *RFC 2979* (2000).

[51] Sebastian Gajek et al. “Universally composable security analysis of TLS”. In: *Provable Security*. Springer, 2008, pp. 313–327.

[52] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. “The Bitcoin Backbone Protocol: Analysis and Applications”. In: *EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, 2015, pp. 281–310. ISBN: 978-3-662-46802-9. DOI: 10.1007/978-3-662-46803-6_10.

[53] Craig Gentry and Zulfikar Ramzan. “Single-Database Private Information Retrieval with Constant Communication Rate”. English. In: *Automata, Languages and Programming*. Ed. by Luís Caires et al. Vol. 3580. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, pp. 803–815. ISBN: 978-3-540-27580-0. DOI: 10.1007/11523468_65. URL: http://dx.doi.org/10.1007/11523468_65.

[54] Kristian Gjøsteen. “Computer Security – ESORICS 2005: 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005. Proceedings”. In: ed. by Sabrina de Capitani di Vimercati, Paul Syverson, and Dieter Gollmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. Chap. Security Notions for

Disk Encryption, pp. 455–474. ISBN: 978-3-540-31981-8. DOI: 10.1007/11555827_26.

[55] Eu-Jin Goh. "Secure Indexes." In: *IACR Cryptology ePrint Archive* 2003 (2003). https://eprint.iacr.org/2003/216/, p. 216.

[56] Oded Goldreich and Rafail Ostrovsky. "Software Protection and Simulation on Oblivious RAMs". In: *J. ACM* 43.3 (May 1996), pp. 431–473. ISSN: 0004-5411. DOI: 10.1145/233551.233553. URL: http://doi.acm.org/10.1145/233551.233553.

[57] Shafi Goldwasser, YaelTauman Kalai, and GuyN. Rothblum. "One-Time Programs". In: *Advances in Cryptology – CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 39–56. ISBN: 978-3-540-85173-8. DOI: 10.1007/978-3-540-85174-5_3. URL: http://dx.doi.org/10.1007/978-3-540-85174-5_3.

[58] Shafi Goldwasser and Silvio Micali. "Probabilistic encryption". In: *Journal of computer and system sciences* 28.2 (1984), pp. 270–299.

[59] Michael T Goodrich et al. "Oblivious RAM simulation with efficient worst-case access overhead". In: *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. ACM. 2011, pp. 95–100.

[60] Mohamed G Gouda, Alex X Liu, and Mansoor Jafry. "Verification of distributed firewalls". In: *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*. IEEE. 2008, pp. 1–5.

[61] Gunnar Hartung et al. "BBA+: Improving the Security and Applicability of Privacy-Preserving Point Collection". In: *CCS 2017*. Ed. by Bhavani M. Thuraisingham et al. ACM, 2017, pp. 1925–1942. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134071.

[62] Rolf Haynberg et al. "Symmetric Searchable Encryption for Exact Pattern Matching using Directed Acyclic Word Graphs." In: *SECRYPT*. 2013, pp. 403–410.

[63] Amir Herzberg. "Folklore, practice and theory of robust combiners". In: *Journal of Computer Security* 17.2 (2009), pp. 159–189.

[64] Dennis Hofheinz, Jörn Müller-Quade, and Dominique Unruh. "Universally composable zero-knowledge arguments and commitments from signature cards". In: *5th Central European Conference on Cryptology*. 2005.

[65] Hejiao Huang and H. Kirchner. "Formal Specification and Verification of Modular Security Policy Based on Colored Petri Nets". In: *Dependable and Secure Computing, IEEE Transactions on* 8.6 (Nov. 2011), pp. 852–865. ISSN: 1545-5971. DOI: 10.1109/TDSC.2010.43.

[66] Matthias Christoph Huber. *Provable and practical security for database outsourcing*. Karlsruhe, 2016. URL: http://dx.doi.org/10.5445/IR/1000058652%20;%20http://nbn-resolving.de/urn:nbn:de:swb:90-586522%20;%20http://d-nb.info/1113109289/34%20;%20http://digbib.ubka.uni-karlsruhe.de/volltexte/1000058652.

[67] Matthias Huber et al. "Cumulus4j: A Provably Secure Database Abstraction Layer". In: *CD-ARES Workshops*. 2013, pp. 180–193.

[68] Kenneth Ingham and Stephanie Forrest. "A history and survey of network firewalls". In: *University of New Mexico, Tech. Rep* (2002).

[69] Kyle Ingols et al. "Modeling modern network attacks and countermeasures using attack graphs". In: *Computer Security Applications Conference, 2009. ACSAC'09. Annual*. IEEE. 2009, pp. 117–126.

[70] *Interactive Graphic: The NSA's Spy Catalog*. Spiegel Online International. http://www.spiegel.de/international/world/a-941262.html. Dec. 2013.

[71] Seny Kamara and Charalampos Papamanthou. "Parallel and dynamic searchable symmetric encryption". In: *Financial Cryptography and Data Security*. Springer, 2013, pp. 258–274.

[72] Myong H Kang, Ira S Moskowitz, and Stanley Chincheck. "The pump: A decade of covert fun". In: *Computer Security Applications Conference, 21st Annual*. IEEE. 2005, 7–pp.

[73] Jonathan Katz. "Universally Composable Multi-party Computation Using Tamper-Proof Hardware". In: *Advances in Cryptology – EUROCRYPT 2007*. Ed. by Moni Naor. Vol. 4515. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 115–128. ISBN: 978-3-540-72539-8. DOI: 10.1007/978-3-540-72540-4_7. URL: http://dx.doi.org/10.1007/978-3-540-72540-4_7.

[74] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC cryptography and network security. 2008, pp. xviii + 534. ISBN: 1-58488-551-3.

[75] Jonathan Katz et al. "Universally Composable Synchronous Computation". In: *Theory of Cryptography*. Ed. by Amit Sahai. Vol. 7785. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 477–498. ISBN: 978-3-642-36593-5. DOI: 10.1007/978-3-642-36594-2_27. URL: http://dx.doi.org/10.1007/978-3-642-36594-2_27.

[76] Louiza Khati, Nicky Mouha, and Damien Vergnaud. "Full Disk Encryption: Bridging Theory and Practice". In: *Topics in Cryptology – CT-RSA 2017: The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14–17, 2017, Proceedings*. Ed. by Helena Handschuh. Cham: Springer International Publishing, 2017, pp. 241–257. ISBN: 978-3-319-52153-4. DOI: 10.1007/978-3-319-52153-4_14.

[77] Alexander Koch. *Cryptographic protocols from physical assumptions*. Karlsruhe, 2019. URL: http://dx.doi.org/10.5445/IR/1000097756.

[78] Kaoru Kurosawa and Yasuhiro Ohtaki. *How to Construct UC-Secure Searchable Symmetric Encryption Scheme*. Cryptology ePrint Archive, Report 2015/251. http://eprint.iacr.org/2015/251. 2015.

[79] Eyal Kushilevitz and Rafail Ostrovsky. "Replication is not needed: Single database, computationally-private information retrieval". In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society. 1997, pp. 364–364.

[80] Romain Laborde et al. "Network security management: A formal evaluation tool based on RBAC policies". In: *Network Control and Engineering for QoS, Security and Mobility, III*. Springer, 2005, pp. 69–80.

[81] Leslie Lamport, Robert Shostak, and Marshall Pease. "The Byzantine Generals Problem". In: *ACM Trans. Program. Lang. Syst.* 4.3 (July 1982), pp. 382–401. ISSN: 0164-0925. DOI: 10.1145/357172.357176. URL: http://doi.acm.org/10.1145/357172.357176.

[82] Ueli Maurer. "Constructive cryptography – A new paradigm for security definitions and proofs". In: *Theory of Security and Applications (TOSCA 2011)*. Ed. by S. Moedersheim and C. Palamidessi. Vol. 6993. Lecture Notes in Computer Science. Springer-Verlag, Apr. 2011, pp. 33–56.

[83] Sebastian Messmer et al. "A Novel Cryptographic Framework for Cloud File Systems and CryFS, a Provably-Secure Construction". In: *Data and Applications Security and Privacy XXXI*. Ed. by Giovanni Livraga and Sencun Zhu. Cham: Springer International Publishing, 2017, pp. 409–429. ISBN: 978-3-319-61176-1.

[84] Steven J. Murdoch et al. "Chip and PIN is Broken". In: *31st IEEE Symposium on Security and Privacy, S&P 2010*. IEEE Computer Society, 2010, pp. 433–446. ISBN: 978-0-7695-4035-1. DOI: 10.1109/SP.2010.33. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5504620.

[85] Old Bailey Proceedings Online, ed. *Trial of JOHN BUCKLEY, THOMAS SHENTON*. Version 8.0. Sept. 1781. URL: https://www.oldbaileyonline.org/browse.jsp?div=t17810912-37 (visited on 09/22/2018).

[86] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. "A Framework for Efficient and Composable Oblivious Transfer". English. In: *Advances in Cryptology—CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 554–571. ISBN: 978-3-540-85173-8. DOI: 10.1007/978-3-540-85174-5_31. URL: http://dx.doi.org/10.1007/978-3-540-85174-5_31.

[87] Benny Pinkas and Tzachy Reinman. "Oblivious RAM revisited". In: *Advances in Cryptology–CRYPTO 2010*. Springer, 2010, pp. 502–519.

[88] Postbank. *Postbank chipTAN comfort*. 2018. URL: https://www.postbank.de/privatkunden/chiptan-comfort.html (visited on 09/25/2018).

[89] RedTeam Pentesting GmbH. *Man-in-the-Middle Attacks against the chipTAN comfort Online Banking System*. 2009. URL: https://www.redteam-pentesting.de/publications/2009-11-23-MitM-chipTAN-comfort_RedTeam-Pentesting_EN.pdf (visited on 09/25/2018).

[90] Christoph L Schuba and Eugene H Spafford. "A reference model for firewall technology". In: *Computer Security Applications Conference, 1997. Proceedings., 13th Annual*. IEEE. 1997, pp. 133–145.

[91] Elaine Shi et al. "Oblivious RAM with O ((logN) 3) worst-case cost". In: *Advances in Cryptology–ASIACRYPT 2011*. Springer, 2011, pp. 197–214.

[92] Radu Sion and Bogdan Carbunar. "On the computational practicality of private information retrieval". In: *In Proceedings of the Network and Distributed Systems Security Symposium*. 2007.

[93] Smart Card Alliance. *Contactless EMV Payments: Benefits for Consumers, Merchants and Issuers*. URL: http://www.emv-connection.com/downloads/2016/06/Contactless-2-0-WP-FINAL-June-2016.pdf (visited on 12/17/2018).

[94] Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. *Practical Dynamic Searchable Encryption with Small Leakage*. Cryptology ePrint Archive, Report 2013/832. http://eprint.iacr.org/2013/832. 2013.

[95] Tamarin. *Tamarin prover*. 2018. URL: https://tamarin-prover.github.io/ (visited on 12/19/2018).

[96] Mårten Trolin. "A Universally Composable Scheme for Electronic Cash". In: *INDOCRYPT 2005*. Ed. by Subhamoy Maitra, C. E. Veni Madhavan, and Ramarathnam Venkatesan. Vol. 3797. LNCS. Springer, 2005, pp. 347–360. ISBN: 3-540-30805-9. DOI: 10.1007/11596219_28.

[97] Visa. *Visa Token Service*. URL: https://usa.visa.com/partner-with-us/payment-technology/visa-token-service.html (visited on 12/17/2018).

[98] Volksbank Mittelhessen eG. *VR-mobileCash: Geld abheben ohne Karte*. URL: https://www.vb-mittelhessen.de/privatkunden/girokonto-kreditkarten/infos-banking/geld-abheben-ohne-karte.html (visited on 09/25/2018).

[99] Yihua Zhang and Marina Blanton. "Efficient Dynamic Provable Possession of Remote Data via Update Trees". In: *Trans. Storage* 12.2 (Feb. 2016), 9:1–9:45. ISSN: 1553-3077. DOI: 10.1145/2747877.