Compliant Cryptologic Protocols

by

Viswanathan Kapaleeswaran

Bachelor of Engineering, First Class
(Electronics, *Bangalore University*, India) 1995
Graduate Diploma in Information Technology, Distinction
(*QUT*, Australia) 1997

Thesis submitted in accordance with the regulations for
Degree of Doctor of Philosophy

Information Security Research Centre
Faculty of Information Technology
Queensland University of Technology

February 2001

QUT

QUEENSLAND UNIVERSITY OF TECHNOLOGY

DOCTOR OF PHILOSOPHY THESIS EXAMINATION

CANDIDATE NAME:	*KapaleeswaranViswanathan*
RESEARCH CENTRE:	*Information Security*
PRINCIPAL SUPERVISOR:	*Dr Colin Boyd*
ASSOCIATE SUPERVISOR(S):	*Professor William Caelli* *Professor Ed Dawson*
THESIS TITLE:	*Compliant Cryptologic Protocols*

Under the requirements of PhD regulation 16.8, the above candidate presented a Final Seminar that was open to the public. A Faculty Panel of three academics attended and reported on the readiness of the thesis for external examination. The members of the panel recommended that the thesis be forwarded to the appointed Committee for examination.

Name: A/Prof Colin Boyd
***Panel Chairperson** (Principal Supervisor)*

Name: Prof Ed Dawson
Panel Member

Name: Dr Ernest Foo
Panel Member

Under the requirements of PhD regulations, Section 16, it is hereby certified that the thesis of the above-named candidate has been examined. I recommend on behalf of the Examination Committee that the thesis be accepted in fulfillment of the conditions for the award of the degree of Doctor of Philosophy.

Name: Professor Ed Dawson *Date:* 19/7/2001
***Chair of Examiners (Head of School or nominee)** (Examination Committee)*

FORM B

Keywords

Compliant cryptologic protocols, compliant cryptosystems, compliance, enforceability, fundamental services, atomic services, confidentiality, integrity, integrity verification technique, key recovery, hybrid key recovery, anonymous token systems, secure selection protocols, electronic cash, peer review protocol, electronic auction, electronic voting.

Abstract

Literally, the word *compliance* suggests conformity in fulfilling official requirements. The thesis presents the results of the analysis and design of a class of protocols called compliant cryptologic protocols (CCP). The thesis presents a notion for compliance in cryptosystems that is conducive as a cryptologic goal. CCP are employed in security systems used by at least two mutually mistrusting sets of entities. The individuals in the sets of entities only trust the design of the security system and any trusted third party the security system may include. Such a security system can be thought of as a broker between the mistrusting sets of entities.

In order to provide confidence in operation for the mistrusting sets of entities, CCP must provide compliance verification mechanisms. These mechanisms are employed either by all the entities or a set of authorised entities in the system to *verify* the compliance of the behaviour of various participating entities with the rules of the system.

It is often stated that confidentiality, integrity and authentication are the primary interests of cryptology. It is evident from the literature that authentication mechanisms employ confidentiality and integrity services to achieve their goal. Therefore, the fundamental services that any cryptographic algorithm may provide are confidentiality and integrity only.

Since controlling the behaviour of the entities is not a feasible cryptologic goal, the verification of the confidentiality of any data is a futile cryptologic exercise. For example, there exists no cryptologic mechanism that would prevent an entity from willingly or unwillingly exposing its private key corresponding to a certified public key. The confidentiality of the data can only be assumed. Therefore, any verification in cryptologic protocols must take the form of integrity verification mechanisms.

Thus, compliance verification must take the form of integrity verification in cryptologic protocols. A definition of compliance that is conducive as a cryptologic goal is presented as a guarantee on the confidentiality and integrity services. The definitions are employed to provide a classification mechanism for various message formats in a cryptologic protocol. The classification assists in the characterisation of protocols, which assists in providing a focus for the goals of the research. The resulting concrete goal of the research is the study of those protocols that employ message formats to provide restricted confidentiality and universal integrity services to selected data.

The thesis proposes an informal technique to understand, analyse and synthesise the integrity goals of a protocol system. The thesis contains a study of key recovery, electronic cash, peer-review, electronic auction, and electronic voting protocols. All these protocols contain message formats that provide restricted confidentiality and universal integrity services to selected data.

The study of key recovery systems aims to achieve robust key recovery relying only on the certification procedure and without the need for tamper-resistant system modules. The result of this study is a new technique for the design of key recovery systems called *hybrid key escrow*.

The thesis identifies a class of compliant cryptologic protocols called *secure selection protocols* (SSP). The uniqueness of this class of protocols is the similarity in the goals of the member protocols, namely peer-review, electronic auction and electronic voting. The problem statement describing the goals of these protocols contain a tuple, (I, D), where I usually refers to an identity of a participant and D usually refers to the data selected by the participant. SSP are interested in providing confidentiality service to the tuple for hiding the relationship between I and D, and integrity service to the tuple after its formation to prevent the modification of the tuple. The thesis provides a schema to solve the instances of SSP by employing the electronic cash technology. The thesis makes a distinction between electronic cash technology and electronic payment technology. It will treat electronic cash technology to be a certification mechanism that allows the participants to obtain a certificate on their public key, without revealing the certificate or the public key to the certifier. The thesis abstracts the certificate and the public key as the data structure called *anonymous token*. It proposes design schemes for the peer-review, e-auction and e-voting protocols by employing the schema with the anonymous token abstraction.

The thesis concludes by providing a variety of problem statements for future research that would further enrich the literature.

Contents

List of Figures

List of Tables

Declaration

The work contained in this thesis has not been previously submitted for a degree or diploma at any higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Date: 9/8/2001

Acknowledgements

I wish to thank my principal supervisor, Associate Professor Colin Boyd, for his tireless guidance, enthusiastic support and unconditional encouragement. I wish to express my heartfelt gratitude for my associate supervisor, Professor Ed Dawson, for choosing me for this research and, for his timely research directions, support and encouragement. I sincerely thank my associate supervisor Professor Bill Caelli, for supporting my research by providing policy information, especially in the area of key recovery and digital signatures. The information was useful in the conceptualisation of the research problem. The cumulative guidance provided me with the best platform for research that one would wish for. I thank the Australian Research Council for supporting this project.

I wish to personally thank Andrew Clark for his friendship and technical support that are invaluable. I wish to personally thank Juan Manuel González and Ernest Foo for their friendship, collaboration and support. My sincere thanks to Greg Maitland for his support, illuminating tutorials in higher mathematics and the fruitful discussions on many research topics. My sincere thanks to all the researchers, students and staff for their friendly interactions and support. They are: Giulio Faini, DongGook Park, Carsten Rudolf, Gary Gaskell, Gary Carter, Paul Ashley, Riza Aditya, Jason Reid, Jason Smith, Bill Millan, Lauren Nielsen, Manyi Lu, Susie Hlaing, Richard Au, Ricco Lee, Yvonne Hitchcock, Joanne Fuller, Agung Prasetijo, Leoni Simpson, Linda Burnett, Mark Looi, Adrian McCullagh and Jeremy Zellers. I thank all the administrative staff at the research centre and the school for their support and administrative services, they are: Christine Orme, Betty Hansford, Anne Hamburger, Gina Farrington, Averil James and June Escobar.

I wish to express my deepest, evergreen gratitude to my mother, father and family. Their love, support, and unshakable belief in me provides an unfathomable strength to succeed in the goals in life. I wish to express my respect for all the teachers and gratitude to those who participated in my life.

I extend my thanks to all the anonymous individuals, mathematicians, technicians and other knowledge-seekers whose selfless works have made the authoring of this thesis possible.

Chapter 1

Introduction

What compliances will remove dissension?
- JONATHAN SWIFT (1667 - 1745)

Democracy is a small hard core of common agreement,
surrounded by a rich variety of individual differences.
- JAMES B. CONANT (1893 - 1978)

Human interactions are rife with contradictions. The *perception* of opposing interests, mistrust and related properties causes such contradictions. Irrespective of its application, every technology must inevitably face such contradictions in some manner, which generally employs an *acceptable solution*, which in turn would be the result of an acceptable compromise. The presence of contradictions demands compliance mechanisms for the interactions.

Modern computing systems have greatly eased many monotonous and routine jobs, and have created many new modes of comfort, recreation and freedom. Albeit important, they are one of the many technologies used to assist human interactions and, therefore, are not immune to such perceptions, interests and events. Although it will be difficult to eliminate the contradictions altogether, it is possible to design interactions that could be acceptable to *all* the involved sets of entities. The study of compliant cryptologic protocols will be very useful for such goals. Compliant cryptologic protocols possess verification procedures that allow validation of the behaviour of various participants against the system rules. At the same time, the verification equations must not adversely affect other security functionalities of the system. Usually, compliant cryptologic systems provide a revocation service that can be employed when compliance verifications fail.

Key recovery and electronic cash systems are specialised forms of cryptologic systems that have evolved as a response to contradictory requirements in secure communication systems. Key recovery systems (KRS) focus on the provision of the confidentiality service in secure communication systems. The sets of participants in KRS are the users, the escrow agents and the law enforcement agents (LEA). There exists an inherent mistrust between the users and the LEA. The users are interested in the setting up of robust confidentiality channels between themselves and the LEA is interested in revoking the confidentiality service from the channels, in order to eavesdrop the communications. Both the LEA and the users must place a prescribed amount of trust on the escrow agents to achieve their respective goals. Similarly, electronic cash systems provide compliance mechanisms for the contradictory requirements held by the authorities and the users. The authorities require strong authentication for all valid *participating* users and the *participating* users require services that would allow them to remain anonymous within the system. Under some circumstances, the authorities may additionally require the revocation of anonymity service (tracing) from the *participating* users. In order to implement the tracing functionality the users and the authorities must trust a set of trustees.

Electronic auction systems consist of a set of bidders, a set of auctioneers, and a set of trustees. The bidders require the provision of confidentiality service to their bids until the closing of the bidding period and the auctioneers require the provision of the integrity service to the bids. After the closing of the bidding period, the auctioneers require the revocation of the confidentiality service for the bid and the bidders require the provision of the integrity service to all the bids. Additionally, anonymity for the losing bidders and global verification of the fairness of the bidding process may also be required. It is evident that the bids require *restricted confidentiality* service, until the closing of the bidding period, and *universal integrity* service. Contradiction occurs when the bidders are not trusted to reveal the bid or when anonymity service must be provided to all participants except the winning bidder. The contradictions can be solved when the bidders and the auctioneers trust the set of trustees to either revoke the confidentiality of the bid or the anonymity for the bidders. Note that revocation of anonymity is, fundamentally, revocation of confidentiality.

Electronic voting systems are among the most complicated and politically sensitive applications of compliant cryptologic protocols. In such systems, the voters require the confidentiality service for their votes and the authorities require verification of the correctness of the votes, before they are tallied. If a vote is incorrect, the voting authorities must not learn that a particular voter had cast an invalid vote. Otherwise, the privacy of the voters would be in jeopardy. If anonymity is provided to the voters then conflicts occur because the authorities require every valid vote to have been cast by authenticated voters. The authorities must also be confident that every voter can vote only once.

The concentration of this thesis is on cryptologic systems that require suitable forms of compliance mechanisms to engender trust in an otherwise mistrusting sets of entities. Inevitably, the mechanisms require acceptable forms of compromise solutions that require *all* the sets of entities to forgo certain capabilities, in return for other capabilities. The methodology of research for this thesis strives to be as apolitical as possible to provide solutions for various problems that are acceptable to *all the participating sets of entities*. This approach results in a predominantly technical treatment of various issues with minimal policy analysis. The advantage of this approach is a technique that consists of *independent* layers of research – namely technical, policy, and management research. The technical research provides feasibility, analyses, design and evaluation for known problem statements from a technical stand-point that the other research concentrations can employ. The problem statements for the technical research are provided by policy, management and technical research.

1.1 Goals and Contributions

The abstract goal of the thesis is to present the similarities in compliant cryptologic systems in an organised manner so that future research can use the data to construct design or analysis frameworks for such systems. Due to the vastness of topics in the abstract goal, the thesis concentrated on a single category of compliant cryptologic systems that provide *restricted confidentiality* and *universal integrity* services. Examples of cryptosystems that possess this pattern are key recovery systems [88] and fair electronic cash systems [38].

The thesis presents a streamlined approach for the visualisation and organisation of cryptologic systems. It visualises the presence of basic services and the finitely enumerable combinations in which they may be employed. The analysis and design proposals that result from such an approach are simple and easy to comprehend. Due to the simplicity in design, the security analyses of the proposals are considerably abstracted.

The concepts proposed in this thesis have the ability to classify seemingly different protocols under a single category, so that advancements in solutions for one protocol can be easily applied to other protocols in the category. The concepts abstract the effects of contradictory requirements in cryptologic systems. This goal was achieved by analysing and enumerating the basic services that all cryptologic systems will use, and analysing the effects of contradictory requirements on these services.

The four related goals in the thesis are as follows:

1. **The development of an informal framework, consisting of the basic services, for the analysis of compliant cryptologic systems**. This goal is to identify the services common to all cryptologic protocols. This information along with the manner in which these services are employed will be sufficient to characterise all cryptographic operations in a protocol. The identification of such similarities in the goals of protocols could be used to group protocols so that a solution for one protocol in a group can be applied to all the other protocols. Thus a classification of cryptologic protocols based on the cryptographic operations employed will be possible.

2. **The development of an informal, conceptual tool for the verification of integrity service**. Since the primary aim of this thesis is the study of cryptologic protocols that provide restricted confidentiality and universal integrity services, a tool for the study of integrity verification equations is essential. Since there exist very few results in the literature for the study of integrity verification equations, such as that of Simmons [85], there is a need to develop suitable techniques to understand the achievement of various integrity verification equations in a protocol. Techniques for the study of confidentiality in systems are popular in the literature, such as that of Abadi and Rogaway [1].

3. **The analysis and design of key recovery systems**. Key recovery systems possess the most basic and straightforward form of compliance statement of the form: restricted confidentiality and universal integrity services. Thus, key re-

covery protocols are the *simplest* and most general class of protocols that can be categorised under this compliance statement.

4. **The analysis, design and usage of anonymous token systems.** Anonymous token systems (ATS) provide restricted or universal confidentiality service to the identity of registered users. Such systems can be used as sub-protocols to solve more complex protocols goals, such as those of peer-review, electronic auction and electronic voting. A potentially conflicting pair of requirements in such systems is the need for robust authentication of system participants and the need for maintaining their privacy. The protocol specific requirements are achieved by designing additional protocols that employ the anonymous authorisation information form the ATS.

The contributions of the thesis are represented by the following list.

1. **A simplified view of cryptologic systems.** The goals of cryptologic protocols are expressed in terms of the basic services, which facilitate simple analysis and design techniques. The simplified view assisted in a clear conceptualisation of various protocol goals, which was very useful in the development of various protocols presented in this thesis.

   When formalised as a syntax containing the representation for confidentiality and integrity services, the view will allow the designer to view protocol goals as integrity services, confidentiality services, and as a combination of these services. This technique would greatly simplify the analysis and design of complex protocols.

2. **Development of informal technique for studying the achievements of various integrity verification equations.** This technique was employed to design an alternative proposal for the Cramer-Shoup cryptosystem [26], which was proved to be secure against adaptive chosen ciphertext attack – the strongest form of attack on any encryption mechanism. This application demonstrated the usefulness of the integrity verification technique (IVT) for the design of new protocols. The IVT was also employed for the analysis of an electronic cash proposal and a key recovery proposal, which resulted in the identification of fundamental de-

sign flaws that were not reported in the literature. This application demonstrated the application of IVT for analysing protocols for design flaws.

3. **Analysis and design of key recovery protocols.** Key recovery protocols are the simplest class of protocols belonging to the compliance category of interest for this thesis. A new property essential for key recovery systems operating over an untrusted, open network was identified. This property was called enforceability. The effect of the absence of this property in private-key and session-key recovery systems was demonstrated. A new paradigm for software based key recovery system that emulated all the properties noticeable in the Clipper proposal [88] was presented. This paradigm was called hybrid key recovery.

4. **Abstraction of anonymous token systems** (ATS). The electronic cash system was analysed using the abstraction. A generic schema that employs the ATS to solve a class of protocols called secure selection protocols (SSP) was conceived. Peer-review protocols, electronic auction and electronic cash systems were identified to be instances of SSP. The schema conceived as result of the previous contribution was employed to solve these instances. The similar solution provided evidence for the possibility for the collective design of seemingly disparate protocols. The solutions also provide evidence for the capabilities of the first contribution.

1.2 Published Material

All publications were co-authored with Prof. Colin Boyd and Prof. Ed Dawson. The list of the papers in a reverse chronological order is as follows:

1. A Three Phased Schema for Sealed Bid Auction System Design. In *Australasian Conference for Information Security and Privacy, ACISP'2000*, 412-426. Lecture Notes in Computer Science, Springer-Verlag.

2. Secure Selection Protocols. In *International Conference on Information Security and Cryptology, ICISC'99*, 130-146. Lecture Notes in Computer Science, Springer-Verlag.

3. Signature Scheme for Controlled Environments. In *International Conference on Information and Communication Security, ICICS'99*, 119-134. Lecture Notes in Computer Science, Springer-Verlag.

4. Strong Binding for Software Key Escrow. In *Proceedings of the 1999 ICPP Workshops, ICPP'99, Japan*, 134-139. IEEE Press, 1999.

5. Publicly verifiable key escrow with limited time span. In *Australasian Conference for Information Security and Privacy, ACISP'99*, 36-50. Lecture Notes in Computer Science, Springer-Verlag.

1.3 Organisation

The main matter of this thesis can be classified into the following groups:

1. Chapters 2 and 3 will present information and tools for the analysis and design of compliant cryptologic protocols;

2. Chapter 4 will deal with the problem of providing restricted confidentiality service. Material from Papers 3, 4 and 5 listed in Section 1.2 is included in this chapter; and,

3. Chapter 5 will discuss the mechanisms for the provision of confidentiality service for an identity of the participants (anonymity service) and present design tools that employ the mechanisms for providing anonymity service to achieve a category of protocols called *secure selection protocols*. Materials from Papers 1 and 2 listed in Section 1.2 are included in this chapter.

The first group of chapters present generic information that will be useful for visualising the problem statements in the other two groups. The second group focuses on key recovery systems and the third group on systems that could employ the anonymity service.

There are three appendices in this thesis. Appendix B presents the third-party protocols employed by mechanisms in this thesis. Appendix C discusses the relevant key recovery proposals available in the literature. Appendix D details the proposal for a fair electronic cash proposal and presents an abstraction of the proposal.

Chapter 2

Compliances in Cryptosystems

The secret of getting ahead is getting started. The secret of getting started is breaking your complex overwhelming tasks into small manageable tasks, and then starting on the first one.
- MARK TWAIN

A complex system that works is invariably found to have evolved from a simple system that worked.
- JOHN GALL

The protocol logic of modern cryptographic systems is becoming more complex with every successful proposal. The complexity hinders precise reasoning, which results in unclear protocol goals for some applications; this makes analysis and design of these protocols more difficult.

This state of affairs is acute, especially, in the analysis and design of compliant systems because a particular service to one group may require the revocation of related services from another. Thereby, the very act of providing a service needs clear understanding. Of the many areas of interest, the different kinds of services and the manner in which they are provided commands significant attention.

The aim of this chapter is to analyse cryptosystems from a basic and simple view point, in order to establish a *common ground* for the analysis of protocols. The aim is accomplished by identifying the atomic (or fundamental) services that any cryptologic protocol would provide and developing an understanding of cryptologic systems (cryptosystems) based on the atomic services. The result of this aim is to present a precise statement of purpose of this research. Also, as a consequence of the aim, different properties of cryptosystems, namely, verifiable encryption, compliance, and enforceability are identified and explained.

2.1 Introduction: A View of Cryptosystems

A simple[1] and robust view of cryptosystems allows for a simple characterisation of cryptosystems, which renders subsequent threads of reasoning about various cryptosystems simple and easy to understand. Traditionally, the art and science of cryptography has been interested in technologies for two basic services, confidentiality and integrity, that can be employed in suitable combinations to realise more powerful constructs. Thereby, confidentiality and integrity can be considered to be the basic (or atomic) services present in all cryptologic protocols. Rueppel [78] presented a similar treatment of cryptosystems, but from the perspective of computer security. A similar view of cryptosystems can be realised from the perspective of cryptologic protocol analysis and design.

The basic services can be viewed as follows: keys provide a service (confidentiality or integrity) with respect to messages. Note the conspicuous absence of the terms entity and trust in the view. The concept of "entities" is external to cryptology — it can be "believed" or trusted that some keys are "held" by certain entities, but this is an extraneous assumption. The importance of entities (like Alice or Bob) is deliberately avoided in subsequent definitions and analyses, in order to facilitate a *key-centric view* of cryptosystems[2]. The reasoning for such an approach follows naturally from the importance of keys in modern cryptosystems.

A cryptosystem can be viewed to be a composition of integrity and confidentiality services. Although the confidentiality service is essential for the provision of the integrity service, they can be considered, with loss of generality and for the sake of simplicity, to be independent. Thereby, cryptosystems can be decomposed into an integrity component and a confidentiality component. This decomposition when represented in a suitable fashion will result in a simple characterisation of the goals of the cryptosystem – that is the integrity goal and the confidentiality goal. If a cryptosystem possesses deficiencies in either of these goals, then it will possess deficiencies as a whole.

[1]"There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult." - C.A.R. Hoare

[2]This is in contrast to an *entity-centric view*, such as that of the BAN logic [14].

2.1.1 Informal Definitions for the Basic Services

A suitable high level definition for confidentiality and integrity will be useful in the subsequent discussions. At the same time, necessary precautions must be in place that would guarantee the definition to be broad enough to encompass all currently available technologies (and those that could be available in the future). Since keys are central to all cryptosystems, the definitions for integrity and confidentiality will be based on keys, messages and ciphertexts.

Definition 2.1 Confidentiality *is the basic service that grants* access *to a message, given the ciphertext* and *the corresponding key.*

It may be useful to abstract confidentiality as a proposition: **if** the key is **known then** the message is **known**. Note that this abstraction does not answer the following question with certainty: if the message is known, can the key be known? The truth table for the implies ($\Rightarrow$) boolean operator suggests that the key can either be known or unknown, when the message is known. The truth-table does not unambiguously answer the question. Let K be a boolean value denoting the knowledge of the key and M be a boolean value denoting the knowledge of the message, then the following equation represents confidentiality:

$$K \Rightarrow M$$

Note that the ciphertext represents the confidentiality service, therefore the above equation is a logical representation of any ciphertext. The confidentiality mechanism is an atomic service that could be employed as a *logical access control* node. In other words, the ciphertext controls the access to a message using the corresponding key(s). Since, in most practical confidentiality systems, a message can also be a key, a one-way function can be viewed as an access control mechanism with the same value for the message and the key. For example, a symmetric key cipher such as DES or AES will be a one-way function with the key string equal to the message string. As Diffie and Hellman [31] noted, every confidentiality rendering system tends to possess the one-way property, irrespective of whether it is a public-key or a private-key system.

Definition 2.2 Integrity *is the basic service that determines the* immutability *of a message, given the message, the ciphertext* and *the corresponding key.*

Since the integrity service is usually modeled as a verification equation, the integrity process takes three inputs (the key, the ciphertext and the message) and produces a single, binary output to signal mutability or immutability. When there exists a bijection between the message and the ciphertext, the message input to the integrity process may be the information describing the bijection, rather than the message itself. Since, most of the currently used technologies base their behaviour on the bijective behaviour of the underlying mathematical structure, this approach is widely used. For example:

1. in the discrete log settings, given a public key $y = g^x \bmod p$, where p is a suitable prime, $x \in \mathbb{Z}_p^*$ is the private key and $g \in \mathbb{Z}_p^*$ is an element (which could be a generator): there exists a bijection between y (the ciphertext) and x (the plaintext), or more precisely the equivalence class $X = \{x \mid y = g^x \bmod p\}$. Thus given y the verifier can draw logical conclusions about X;

2. in the case of symmetric key encryption algorithms, the bijection between the plaintext and ciphertext is crucial for proper decryption behaviour.

The analysis will regard confidentiality and integrity to be independent services, and will aim to *decompose* the cryptosystem to obtain two views – representing views for the integrity and the confidentiality services. These views can be analysed to gain better understanding of the protocol, which will be useful for the analysis and optimisation procedures.

Menezes, von Oorschot and Vanstone [62] additionally list authentication and non-repudiation as basic goals of cryptosystems. The omission of these services, here, *will not affect* the goals because authentication and non-repudiation employ confidentiality and integrity services, in turn, to achieve their results. For example, a signature system provides *confidentiality* service for the private key of the signer[3], and *integrity* service for the verifier, for a message – by employing the message, the set of public keys, and the information about the bijection between the set of private keys and the set of public keys. The random numbers that some signature systems may use can be modeled either into the message to be signed or the public-private key pair. Such approaches will either result in a signature on a randomised message or, a signature employing a randomised short-term key pair derived from the certified long-term key pair.

[3]No entity may compute the private key given the value of the public key ($\mathit{PrivateKey} \Rightarrow \mathit{PublicKey}$) or the signature ciphertexts ($\mathit{PrivateKey} \Rightarrow \mathit{SignatureCiphertexts}$)

The sources of random numbers are crucial for many modern protocols. Hence, it may be argued that random number generation is a basic service. This argument has its merits and demerits. A demerit is that it tends to complicate the representation of protocols (cryptosystems), which needs simplification. So, this aspect must be accommodated in the representation at a higher level. In order to accomplish such a goal, a precise understanding of the role of random numbers will be very useful. The importance of random numbers (or pseudo-random numbers) in contemporary cryptography is a direct consequence of the basic nature of cryptosystems. A random number is unpredictable, thereby its confidentiality is guaranteed until its generation (or until the termination of the generation process): no entity, including the generator, should be able to predict the number. Thus, random number generators are *tools*, like encryption-decryption algorithms, for implementing confidentiality, which is a basic service. Abadi and Rogaway [1] discuss the modelling of the confidentiality service from a positional calculus and complexity point of view. Their discussion suggest a strong similarity between the two approaches.

2.1.2 Composition of Cryptosystems

Observing the definitions for confidentiality (Definition 2.1) and integrity (Definition 2.2), it is evident that:

1. confidentiality is a proposition represented by ciphertexts; and,

2. integrity is a test (or diagnosis) on the relationship between a ciphertext and message.

Confidentiality, by itself, is not a test, rather it is a proposition about the access to a message given the knowledge of a key, which is usually intended to be private. Thus confidentiality represents the private view of the cryptosystem (the view that is available only when a key is available). Note that the relationship between the ciphertext and the message cannot be determined because a single ciphertext can provide access to different messages (views) when different keys (accesses) are employed. For example, suppose a message m_1 is encrypted employing the key k_1 to obtain a ciphertext c_1. If the ciphertext c_1 is decrypted using another key k_2 that is unrelated to k_1, the result will potentially be a random message m_2 that would be unrelated to m_1. The confidentiality view represents the *trusting environment*, where all the entities participating in the protocol know and trust the relevant keys, and their associations with corresponding entities. In fact, there would be no other choice because this view presents only propositions and no diagnoses or tests that can be verified.

Integrity is a diagnostic tool that can be used to *verify* the relationship between a message, a ciphertext and a key. It provides a binary answer to signal relationship or lack thereof. Thus, integrity can be viewed to be the mutability of the relationship between the message and the ciphertext based upon the assumptions on mutability of the key. Unlike the confidentiality service, the relationship between a ciphertext and a message can be precisely determined given the knowledge of a key. If the key is public then the view is public, else it is private. This view represents the *mistrusting environment* where, by Definition 2.2, every entity needs to check the relationship between the message and the ciphertext. Although the entities can assign propositions to various diagnoses, this view contains diagnoses alone and no propositions. This is a crucial observation that is important for protocol designers who use the protocol constructs of other designers. A construct may robustly provide a diagnosis but it is up to the protocol designer to meaningfully interpret, or assign correct proposition or propositions to the diagnosis.

The complete cryptosystem is a combination of confidentiality and integrity services such that there are propositions and diagnoses. Propositions and diagnoses can be unrelated or related. Usually, though, related tuples of propositions and diagnoses are more interesting from the perspective of a protocol design. The related tuples can be thought of to be the *glue* that binds different confidentiality constructs (propositions) and integrity constructs (diagnoses). In most cases of deficient protocols, the failure can be traced to an unrelated tuple of propositions and diagnoses that was misinterpreted as related. That is, incompatible or incorrect propositions were assigned to some of the diagnoses.

Thus a cryptosystem has a propositional view and a diagnostic view. The propositional view, inherently, cannot be tested and the diagnostic view is solely for testing. This decomposition of cryptosystems allows us to view a cryptosystem:

1. purely as a propositional system;

2. purely as a diagnostic system; and,

3. as a system with a combination of propositions and diagnoses.

The first two views are useful to perform simple, first-hand analyses of the (propositional or diagnostic) achievements of the cryptosystem and the third view is useful to analyse the correctness of the synthesis of these achievements. If there is a deficiency in either of the first two views, then the third view will have a deficiency, *but not necessarily the other way around.*

In the case of public key cryptosystems we can, usually, treat the propositional view as a private view and the diagnostic view as a public view, based on the knowledge of a key.

2.1.3 A Characterisation of Cryptosystems

In this section, various terms to be used in the definition of cryptosystems will be clarified, followed by the definition itself.

- **Security Object:** is a collection of functionally related ciphertexts that provide the confidentiality service to a set of messages using a set of keys. A system may have many security objects. Examples are public keys (identity), session keys, electronic coins and ticket-granting tickets in authentication schemes like Kerberos.

- **Node:** is a collection of entities. While in the sending mode the node is interested in the confidentiality, or preserving the confidentiality, service of a set of security objects. While in the receiving mode the node is interested in the integrity service of a set of security objects and/or *accessing* the messages in some (or all) of the ciphertexts.

- **Source:** of a security object is the node that created the security object in its entirety – that is the source must form all the ciphertexts corresponding to that security object.

- **Sink:** of a security object is the node that is the intended target entity and has *access* to the message in some or all of the ciphertexts in the security object.

- **Message Format:** is a logical container that contains security objects and related ciphertexts for the verification of the integrity of the message format.

Definition 2.3 *A cryptosystem contains at least one security object and a source.*

There must be at least a single security object and a source (to form the security object) to *cause* any subsequent events, if any, to occur. Note the conspicuous absence of a requirement for a sink in the definition. This is to encompass those systems that may lock information forever, which are valid cryptosystems.

2.2 Verifiable Encryption

As stated in the previous section, the confidentiality view presents only propositions and no diagnoses that can provide proofs for the propositions. There are many applications that require the verification of the message format by a node other than the source or the sink. For example, publicly verifiable secret sharing schemes [86, 3] may require a monitor (a node) to verify the ciphertexts sent by the dealer of the secret (the source) to a shareholder (the sink), in order to ascertain that the shareholder will obtain a valid share. Similar examples are available in key-recovery [90], e-cash [12], e-voting and e-auction [91] systems.

Verifiable encryption techniques are primarily concerned with the formation of specialised message formats that contain security objects produced by some confidentiality system (such as an encryption algorithm) and diagnostic data for some integrity verification system. Stadler [86] proposed a *form* of verifiable encryption to achieve a publicly verifiable secret sharing scheme, which provided a public diagnosis about a proposition regarding the encryption of individual secret shares. The goal of the verifiable encryption scheme [86] was the design of a message format that contained the following components:

Confidentiality: Security object of the i^{th} secret share $c_{1i} = Enc(key_i, share_i)$ and $c_{2i} = f(share_i)$, where f is a one-way function, and the ciphertext protecting the secret, $c_2 = f(secret)$; and,

Integrity: The ciphertexts of the message formats contained data for the following diagnosis (integrity checks);

1. The message component of c_{1i} and c_{2i} are equal; and,
2. The message component of c_{2i} satisfies a relationship with c_2, and all the other ciphertexts $c_{2j} = \mathit{OneWayFunction}(\mathit{share}_j \mid i, j \in \mathcal{A})$, where $\mathcal{A}$ is a pre-defined access-control structure.

Note that the confidentiality and integrity views are not disjoint – as both the views contain the ciphertexts c_{1i}, c_{2i} and c_2: so the compositional view is the union of the independent views. Many other proposals for verifiable encryption have been proposed, with or without explicitly identifying them with this terminology. This section will present different forms of verifiable encryption.

Verifiable encryption can be considered to be a method for associating a message with a key, and thereby the ciphertext, without revealing the message. A brief classification of verifiable encryption techniques will be useful for the analysis and synthesis of verifiable encryption schemes. The next sub-section will present a classification for publicly verifiable encryption schemes.

2.2.1 A Classification of Publicly Verifiable Encryption Schemes

Publicly verifiable encryption is a technique to allow the prover to encrypt a message, m, usually, under the public key, y, of a receiver to obtain a ciphertext, $\mathcal{E}$, and prove to any verifier that m in $\mathcal{E}$ has a particular property, **without revealing** additional information (as defined by the primitive) about m.

The classes of publicly verifiable encryption that can be listed are:

Class 0: (Commitment for encrypted message) Given the one-way image of a message and the encryption of the message, prove that the pre-image of the one-way image is *equal* to the decryption of the encrypted message. That is:

$$\langle PROOFEQ(\mathcal{O}(m) = \mathcal{O}(Dec(Enc(m))))\rangle,\ \mathcal{O}(m),\ Enc(m)$$

where $\mathcal{O}$ is a one-way function (or a suitable commitment function) and Enc is a public key encryption function such that Dec is the decryption function, which can be efficiently computed only with the knowledge of a corresponding private key. $PROOFEQ$ is a proof of equality. It can be observed that:

1. $\mathcal{O}(m),\ Enc(m)$ are the propositions; and,

2. $\langle PROOFEQ(\mathcal{O}(m) = \mathcal{O}(Dec(Enc(m)))\rangle$ is the diagnosis.

Given a ciphertext $Enc(m)$ and the one-way image of the message, $\mathcal{O}(m)$, mechanisms in this class allows for the proof of equality of the pre-image of $\mathcal{O}(m)$ and the decryption of the ciphertext, if that is the case. Stadler [86] and, Asokan, Shoup and Waidner [3] employed this class to design a publicly verifiable secret sharing protocol and a fair exchange protocol, respectively.

Class 1: (Equality of encrypted message) Given two ciphertexts[4] under different keys, prove that they encrypt the same message. That is:

$$\langle PROOFEQ(Dec_1(Enc_1(m)) = Dec_2(Enc_2(m)))\rangle,\ Enc_1(m),\ Enc_2(m)$$

where Dec_i is the decryption function corresponding to the encryption function Enc_i such that Dec_i can be computed only by entity i and, $PROOFEQ$ is the transcript of a proof system for equality relationship. It is important to note that Dec_1 and Dec_2 may use similar or different encryption algorithms.

1. $Enc_1(m),\ Enc_2(m)$ are the propositions; and,
2. $\langle PROOFEQ(Dec_1(Enc_1(m)) = Dec_2(Enc_2(m)))\rangle$ is the diagnosis.

Proposals belonging to this class provide mechanisms to allow a *single* message to be encrypted for two (or more) parties and prove this. Note that the definition for this class and Class 0 mechanisms are identical if we replace the one-way function, $\mathcal{O}$, with an encryption function – or in other words, a commitment scheme is a generic encryption function such that *nobody* knows the corresponding decryption function. Frankel and Yung [38] and, Verheul and van Tilborg [89] employed this class to design a fair off-line e-cash system and a binding ElGamal proposal, respectively. Note that Enc_1-Dec_1 and Enc_2-Dec_2 need not be similar cryptosystems. In fact, the decryption mechanism (of one of the cryptosystems) need not even exist.

Class 2: (Membership of message) Given a ciphertext prove that the encrypted message is a member of a pre-defined set. That is:

$$\langle PROOFIN(Dec(Enc(m)) \in \mathcal{M})\rangle,\ Enc(m),\ \mathcal{M}$$

[4]Extension to more than two ciphertexts can be easily derived from the basic form.

where $\mathcal{M}$ is a finite set of messages and $PROOFIN$ is the transcript of the proof system for the membership relationship (of $m \in \mathcal{M}$). Here:

1. $Enc(m)$ is the proposition; and,
2. $\langle PROOFIN(Dec(Enc(m)) \in \mathcal{M})\rangle$ is the diagnosis.

Proposals in this class might use a witness indistinguishable proof along with a probabilistic encryption scheme. Cramer, Gennaro and Damgård [22] employed this class in the design of a voting scheme.

Class 3: (Knowledge of structure of the encrypted message) Given the encryption of the one-way image[5] of a message, prove knowledge of the message. That is:

$$\langle PROOFKNOW_m(Dec(Enc(\mathcal{O}(m))))\rangle,\ Enc(\mathcal{O}(m))$$

where $PROOFKNOW_m$ is the transcript of the proof of knowledge of m. It could be noted that:

1. $Enc(\mathcal{O}(m))$ is the proposition; and,
2. $\langle PROOFKNOW_m(Dec(Enc(\mathcal{O}(m))))\rangle$ is the diagnosis.

For example, the proof system may convince the verifier that the user knows the discrete logarithm of the encrypted message. Discussion on this form of proof is not yet popular in the research literature. Any break-through in the quest for concrete solutions for this class of algorithm will provide improved alternatives to many known protocol suites such as e-cash and e-auctions. For example, $\mathcal{O}(m)$ can be the public key of the sender and thereby m the private key. This approach could yield many interesting solutions for some applications.

There may well be additional classes of publicly verifiable encryption. Irrespective of the class, all the algorithms will contain a set of publicly accessible propositions (security objects) and a set of diagnoses to prove some aspect of the encrypted message (ciphertexts in the message format). Most cryptosystems that require restricted confidentiality service in a highly untrusted environment employ some form of verifiable encryption.

[5]or any other appropriate structure.

2.3 Types of Compliance

After the examination of verifiable encryption, a natural query would be: why is verifiable encryption important for compliant protocol design? The answer lies in the nature of the class of compliant cryptosystems that is of interest for this research. Verifiable encryption is the most natural tool that can be used to achieve restricted confidentiality service with an universal integrity service. Many compliant systems operate for at least two sets of users, with users of one set having a fragile trust relationship[6] with the users of the other set. Also, in most cases, the service of interest for one set may contradict the service of interest for the other set. Prominent examples are:

1. in a key-recovery system the set of users are interested in the establishment of confidentiality channels between themselves while the set of law-enforcement users are interested in the controlled revocation of confidentiality from these channels;

2. in electronic coin systems the set of users wish to realise anonymous funds transfer to the set of merchants and the set of trustees may wish to revoke the anonymity in the case of double-spending, black-mailing, or under some special conditions;

3. in electronic auction systems the set of bidders are interested in confidentiality service for the value of their bid and the set of auctioneers are interested in the revocation of the service for the winning bid, if not for all the bids;

4. in electronic voting systems the set of voters are interested in the confidentiality of their vote while the set of officials are interested in the authenticity and correctness[7] of the vote.

If we consider anonymity to be confidentiality service for an identity, then all the examples require some form of verifiable confidentiality service.

1. Key-recovery systems require the set of users to prove recoverability service, for the set of law-enforcement users, in order to avail the required service;

[6]A relationship where two parties do not trust each other but engage in some form of interaction to obtain some service from a common system.

[7]Only authenticated voters can vote once. If the tallying procedure does not verify the correctness of the vote, then the voting procedure must check if the vote is a valid choice.

2. Electronic cash systems require the set of customers to prove the revocability of their identity;

3. Electronic auction systems require the set of bidders to prove that the confidentiality of the bid can be revoked;

4. Electronic voting systems require the voters to prove that a valid vote has been encrypted and that the voter has not already voted in the election.

Most of these systems possess a fine balance between provision and revocation of services for a set of mutually mistrusting users. A straightforward manner would be to require the set of users, who avail some service from the other set of users, to prove compliance to the rules of the system. The services in such systems can be analysed by considering the following aspects:

1. the provision of service (or functionality); and,

2. the logic of the provision.

Functionally, cryptosystems provide two types of service, which are confidentiality and integrity, as discussed in Section 2.1.1. Thus, the first aspect has already been dealt with. The remainder of this section will discuss the second aspect and its relationship with the first aspect. The provision of services can either be restricted or universal.

Definition 2.4 *A cryptosystem may provide a service of interest until an event occurs. The occurrence of the event may be probabilistic in nature or deliberately triggered. Such a type of guarantee for a service is called* restricted.

For example:

1. In key recovery schemes, such as [88, 63], the confidentiality service (for a message or session key) is guaranteed until a set of trustees participate in the key recovery protocol.

2. In electronic cash schemes, such as the proposal by Brands [12], the confidentiality service (for an identity embedded in a coin) is guaranteed until the coin is double spent;

3. In fair electronic cash schemes, such as the proposal by Frankel, Tsiounis and Yung [38], the confidentiality service is guaranteed until a set of trustees participate in a tracing protocol;

Definition 2.5 *A cryptosystem may provide a service of interest without any conditions. Such a type of guarantee for a service is called* universal.

For example:

1. In electronic cash schemes, an universal integrity service must be provided to the structure of the coin;
2. In key recovery systems the universal integrity service is essential for LEAF like components [88] in order to avoid integrity oriented attacks [10].
3. Universal integrity and confidentiality services are essential for key agreement protocols; and,
4. Universal integrity and confidentiality services for the private key of the root certification authority (if present) is essential for the proper functioning of the public-key infrastructure.

Cryptographic services are provided on a per-message basis and cryptosystems may have many messages, therefore a cryptosystem may provide a variety of services. For example, a cryptosystem may provide restricted confidentiality service for some messages and universal confidentiality for some other messages.

Since confidentiality and integrity are the basic services, the entities participating in the system are interested in the manner in which these services are provided. Compliance is a property that is global to the cryptosystem.

Definition 2.6 Compliance *is a* guarantee *on the confidentiality and integrity services by the cryptosystem to the participating entities.*

For example:

1. every message communicated in a message format (such as an electronic coin) by every source and sink in a key escrow system must be *accessible* to the law enforcement agency.

2. every fair electronic coin (message format) in an e-cash system must contain a valid signature by the bank and the (hidden) identity of the customer owning the coin. The source of the coin (security object) is the node containing the bank or the customer, and the sink is the node containing the bank or the merchant.

Definition 2.6 presents a very broad view of compliance. It is possible to categorise cryptosystems based on the definition and the type of service provided by cryptosystems. Since there are two basic services (confidentiality and integrity) and two types of service (restricted and universal), there are four types of compliance guarantee. They are:

Compliance Category 0: aims to guarantee *universal confidentiality and integrity* services for all security objects and/or message formats. Most security objects in traditional cryptosystems such as identification systems, signature systems, key establishment systems and entity authentication systems can be classified under this category. In all these systems confidentiality and integrity services are guaranteed universally. For example, a signature system provides *universal confidentiality* service for the private key of the signer (source) and, *universal integrity* service for the verifier (sink), for a message – by employing the message, public key, and, the information about the bijection between the set of private keys and the set of public keys.

Compliance Category 1: aims to guarantee *universal integrity* service for all message formats and *restricted confidentiality* service for selected security objects. Most key recovery systems and e-cash systems can be classified under this category. For example, the e-coins (security objects) proposed by Brands [12] and Frankel, Tsiounis and Yung [38] provide *restricted confidentiality* service for the identity of the customer (part of the source), and *universal integrity* service for the identity, for every other participant – the merchant and the bank (sink). The primary interest of this thesis is to focus on this category of compliance.

Compliance Category 2: aims to guarantee *universal confidentiality* service for all security objects and *restricted integrity* service for selected message formats. Prospective cryptosystems that may employ the *deniable encryption* concept proposed by Canetti *et al* [16] could be an example for this category.

Compliance Category 3: aims to guarantee *restricted integrity and confidentiality* services for selected security objects and/or message formats. Oblivious transfer (OT) protocols, introduced by Rabin [75], are good candidates for this category. The source has two messages and avails confidentiality service for both. One of these services is restricted. It engages in the OT protocol with the sink. At the end of the OT protocol, the confidentiality service for *one* of the messages is revoked by the sink, which, in-turn, avails the confidentiality service for that message, say m_i. Additionally, the sink avails the integrity service for the message, m_i. It cannot avail the integrity service for the other message. That is the integrity service for the messages are restricted based on the choice of the sink.

A cryptosystem's compliance guarantees can be uniquely specified by tuples containing:

1. the message format specification;
2. the compliance category (or categories) of security objects in the message format;
3. the level of enforcement, to be discussed in the next section, for the compliance.

This is a very useful way to categorise compliance and present the information to the design, analysis, implementation, deployment and maintenance phases of a project. Note that additional information may be required by some phases, but these provide some of the essential information that the design phase is interested in communicating.

2.3.1 Enforcement of Compliance

Recalling the characterisation of cryptosystems from Section 2.1.3, we observe that the message formats contain diagnostic information on some security objects. As this research focuses on security objects belonging to compliance category 1 (see Section 2.3), universal integrity service (diagnosis) is of interest. So, immutable message formats are important for this analysis. Popular examples of message formats will be:

1. the LEAF structure in the Clipper proposal [88];
2. the electronic coin abstraction proposed by Brands [12]; and,

3. the bind data in the binding ElGamal scheme for a fraud detectable proposal for key recovery by Verheul and van Tilborg [89].

The integrity service for all these message formats is critically important for the proper operation of the system. If we study the example of certification-based key-recovery schemes employing the public-key technology such as the binding ElGamal proposal by Verheul and van Tilborg [89], we can easily realise the intent of these proposals: if the public-key infrastructure is employed for legal communication, then key-recovery is mandatory. Such propositions appear to work well on paper, but it may be technically difficult to *implement* the proposition. Thus, hundred percent compliance may not be possible to implement. In such situations, it must be possible to grade the level of compliance.

Definition 2.7 *The degree of the compliance guarantee provided determines the* enforcement level *of the cryptosystem. This is the* enforceability *property of the cryptosystem.*

In key recovery systems, for example, co-operating source and sink can always bypass escrow (trivially by employing super-encryption procedures). Thus, key recovery systems may operate assuming source rogue-user, sink rogue-user or source-sink rogue-user models presented by Denning [29]. The resulting enforceability depends on such design assumptions. The following broad categorisation of enforceability is possible:

Enforceability Level 0: cryptosystems employ an on-line monitor to guarantee compliance.

Enforceability Level 1: cryptosystems employ an off-line monitor to guarantee detection of failure of compliance.

Each enforcement level may encompass different degrees of compliance that depend on varying factors such as implementation details, available technology, and so on. For a fine-grained calibration of the level of enforceability into various degrees, standards such as the US Federal Information Processing Standard FIPS:140-1 [67] may be employed. The remainder of this section will concentrate only on the level of enforceability mentioned above, rather than delving into the degree of enforceability that may be possible. This is because the degree of enforceability is an implementation issue and need not necessarily be a design issue, so as to avoid complicated design and analysis techniques.

The compliance guarantee is achieved by engineering *on-line* or *off-line* monitors (resulting in enforceability level 0 or level 1 systems respectively) that verify the message formats being communicated within the system. We refer to Section 2.1.3 for a discussion on the terminologies that will be used in the following definition.

Definition 2.8 *A monitor is a node that is responsible for the integrity verification of the message formats and their conformance to the compliance guarantee.*

The following observations about the nature of monitor are important to understand the enforceability mechanisms:

1. In the case of *on-line monitors*, the security objects must be received by the monitor before the sink can access the message and/or avail the integrity service. In this case, the cryptosystem achieves the compliance guarantee due to the on-line nature of the monitor. The on-line nature of the monitor can be realised by requiring the monitor to be physically on-line, like key recovery systems [88], or logically on-line, like electronic cash systems with observers [18, 12]. In the case where the monitor is logically on-line, the sink may have to forward the message format to the monitor before the required service can be availed, instead of the source sending the message format directly to the monitor. The tamper-resistant hardware that was employed by the Clipper proposal [88] is an example for on-line monitors.

2. In the case of *off-line monitors*, the security objects need not be received by the monitor before the sink can avail the required service. The sink may be expected to forward the message format to the monitor or the monitor may intercept (or wire-tap) the message format while it is in transit. In this case the cryptosystem can only guarantee the detection of failure of compliance. For example, the electronic cash scheme as proposed by Frankel and Yung [38] can only detect double-spending of e-coins and not prevent it because the monitor (bank) is off-line. Similarly, the fraud detectable key-recovery scheme proposed by Verheul

and van Tilborg [89] can, at best, detect malicious communicating parties employing legal message formats to by-pass key recovery and, cannot prevent such activities.

Most cryptosystems require strict formats for the messages being transmitted during each transmission phase. The source of the communication creates the message formats to obtain confidentiality service and the sink may verify the message formats to determine the integrity service for the messages contained in the format. Note that the verification process is only interested in determining the integrity of the message formats. This is a subtle, but important, observation that is crucial for understanding the role of *enforcement* in some cryptosystems to be discussed later in this thesis. There are two options for the verification process:

1. only the target entity (or set of target entities) can assume the role of sink, which results in *restricted verifiability* of the format; and,
2. any entity can assume the role of sink, which results in the *global verifiability* of the format.

Both forms of verifiability are useful depending on the purpose for the verification. For example, some key escrow systems may require any entity to act as a monitor and other systems that provide restricted anonymity may place restrictions on the monitors that can obtain the integrity service.

2.4 Summary

This chapter presented a broad introduction to compliant cryptosystems. The primary interest of the research is the study of cryptologic systems that employ message formats belonging to compliance Category 1 – universal integrity service for all message formats and restricted confidentiality service for select security objects.

A cryptosystem is primarily concerned with the manner of provision of the basic services. The manner of provision of services is encompassed by the definition for the term compliance. Compliance is a guarantee and requires constant auditing of the system by suitable entities. Since confidentiality by itself does not provide audit (diagnosis) tools and integrity by itself is not sufficient to achieve the goals of most systems, a combination of confidentiality and integrity services is essential. A straightforward combination of confidentiality and integrity services results in the (publicly) verifiable encryption proposals, which invariably are employed by all compliant cryptosystems.

Message formats are important for the audit of cryptosystems. Diagnosis of message formats can be achieved by designing on-line or off-line monitors. The nature of the monitors decides the enforcement level of the cryptosystem. On-line monitors provide more effective enforcement than that of off-line monitors.

The next chapter will propose an integrity verification technique that would be useful to help protocol designers to understand the implications of various verification equations. Understanding the verification equations is important because they are the audit (diagnostic) tools for monitoring the compliance of message formats. This is an important issue in a highly mistrusting environment.

Chapter 3

Integrity Verification Technique

Subtlety may deceive you; integrity never will.
- OLIVER CROMWELL
There are things known, and there are things unknown.
And in between are the doors.
- JIM MORRISON

As was stated in Chapter 2, universal integrity service, for all participants, is essential for all message formats belonging to the compliance Category 1. An understanding of the achievements of various verification equations must be achieved. The achievements determine the effectiveness of the compliance statements of various message formats employed by the cryptosystems.

Since there is no known formal integrity verification methodology available in the literature, an informal methodology was developed. The technique employs a graphical representation for the integrity service, as defined in Chapter 2. This representation results in a chain of propositions that relates the integrity of various keys to other keys, which may in turn be messages or ciphertexts. The assumptions (such as beliefs) and trust conditions (such as certification) for the keys, messages and ciphertexts are not represented. This results in a syntax that deals entirely in the domain of cryptology, which contains only messages, keys and ciphertexts.

This chapter will explain the methodology developed for the verification of integrity services. The methodology is then employed to analyse an encryption algorithm that is secure against the adaptive chosen ciphertext attack, an electronic cash system, and a key-recovery system. The usefulness of this technique is further demonstrated by designing a more efficient alternative to the encryption algorithm and identifying deficiencies in the proposals for the electronic cash system and the key-recovery system.

3.1 Introduction

The integrity verification technique (IVT) for the study of the integrity services of a proposal focuses on the verification equations of the system of interest. This approach is useful in abstracting the *unpredictable behaviour* of the signer.

This section presents a characterisation of the Schnorr signature scheme [81] and its variants in Section 3.1.1. Section 3.1.2 contains a discussion on Schnorr-type blind signature schemes [18, 12] and outlines the subtleties that protocol designers must be aware of.

The notations employed in this chapter are as follows:

- $\langle X \rangle$ represents a set of values named X, which may denote the public-key, message or ciphertexts;
- $[x_1, x_2, \cdots]$ represents a tuple; and,
- $\Big(\cdots\Big)$ is the delimiter for separating individual verfication equations.

3.1.1 Characterising Signature Schemes

The servicing of a message, M, by a key, K, by employing a ciphertext, C, can be represented as follows:

$$K \xrightarrow{SERVICE,C} M$$

where, $SERVICE \in \{\mathcal{C}, \mathcal{I}\}$ is the type of service, $\mathcal{C}$ is the keyword for the confidentiality service and $\mathcal{I}$ is the keyword for the integrity service. Confidentiality is the private view of participants and integrity is the public view. Note that the terms private and public are relative, and depend on the assumptions about the ownership of various keys. Since this chapter is concerned with the characterisation of the integrity service, $SERVICE = \mathcal{I}$. So, the $SERVICE$ component of the expression will not be explicitly depicted.

A signature scheme, from the perspective of the integrity goal, can be visualised to be a mechanism that *transfers* the integrity service from a key to a message. The following representation results from the technique:

$$\langle PublicKey \rangle \xrightarrow{\langle Ciphertexts \rangle} \langle Message \rangle$$

The term $\langle \mathit{Ciphertexts} \rangle$ represents the result of any cryptographic operation, including the encryption and signature operations. For example, if $y = g^x \bmod p$ for a suitable value of p, then y is a ciphertext. Usually, the signature process is computationally expensive and the messages are arbitrarily long. Therefore, suitable message digest (symmetric key) techniques are employed. This gives raise to two techniques.

The first technique is to directly sign the message digest. Suppose that an RSA key pair [77], $[e, n]$, is employed to sign a message, m, employing a secure hash function, $\mathcal{H}$, to generate the following verification equations:

$$\begin{aligned} c &\stackrel{?}{=} \mathcal{H}(m, \cdots) \\ r &\stackrel{?}{=} c^e \bmod n \end{aligned}$$

then $[c, r]$ is a signature tuple. This technique is represented as follows:

$$\begin{aligned} (\langle \mathit{SymmetricKey} \rangle &\xrightarrow{\langle \mathit{MessageDigest} \rangle} \langle \mathit{Message} \rangle) \wedge \\ (\langle \mathit{PublicKey} \rangle &\xrightarrow{\langle \mathit{SignatureCiphertexts} \rangle} \langle \mathit{MessageDigest} \rangle) \end{aligned}$$

where:

1. $\langle \mathit{SymmetricKey} \rangle$ is *Null*, since $\mathcal{H}$ is usually an unkeyed hash function;

2. $\langle \mathit{MessageDigest} \rangle = c$;

3. $\langle \mathit{Message} \rangle = [m, \cdots]$;

4. $\langle \mathit{PublicKey} \rangle = [e, n]$; and,

5. $\langle \mathit{SignatureCiphertexts} \rangle = r$.

Henceforth, the logical and operation will be represented by the $\wedge$ symbol. This operator means that individual verification equations must output `true` for the verification system to output `true`. Note that the *Null* key represents the no key scenario and is known globally to all participants. Also note the myriad of protocol design possibilities when *SymmetricKey* is not equal to the *Null* key.

The second technique is to sign a symmetric key that would provide the integrity service to the message. The technique proposed by Fiat and Shamir [37], and adopted by Schnorr [81] is a good example. The following representation results from our proposal:

$$(\langle PublicKey\rangle \xrightarrow{\langle SignatureCiphertext\rangle} \langle SymmetricKey\rangle \xrightarrow{\langle MessageDigest\rangle} \langle Message\rangle)$$

The symmetric key, in this case, cannot be the *Null* key. Note that the representation, by itself, does not suggest that the signature ciphertext provides non-repudiation service to the message, rather it suggests integrity service for the symmetric key, which in turn provides integrity service to the message. This is because the representation deals with a lower level view to trace the flow of integrity service, which is more fundamental than the non-repudiation service. A one-to-one relationship between the symmetric key and the message is essential to extend the non-repudiation service to the message, which in the Schnorr signature scheme is achieved by a one-to-one relationship between the signature ciphertext and the message digest. The rest of this section will explain this form of representation in detail.

A tuple $[r,\ c]$ is a valid Schnorr signature on a set of messages m by the public key $[g, y, p]$ (henceforth the symbol p, representing the prime number, will be omitted from the public key whenever it can be implicitly understood), if the following equation holds:

$$c \stackrel{?}{=} \mathcal{H}(m, A)$$

where, $\mathcal{H}$ is a secure hash function, c is the message digest and $A = y^c g^r$ is the symmetric key. The integrity goal of the Schnorr signature scheme can be expressed as follows:

$$\left([g, y] \xrightarrow{[c,r]} A \xrightarrow{c} m\right) \tag{3.1}$$

That is a *trusted* public key, $[g, y]$, provides integrity service to a symmetric key, A, by employing the ciphertext, $[c, r]$. The symmetric key, A, in turn provides integrity service to the message, m, by employing the ciphertext c. The same value of the ciphertext, c is employed by the public key and the symmetric key. This is an important requirement to prevent the generation of multiple signature transcripts from a single Schnorr signature.

The proof of equality of discrete logarithms employed by Chaum and van Antwerpen [17] resembles the Schnorr signature. It proves that $\log_g y = \log_v u$ for some u

and v. Note that $[g, y]$ **or** $[u, v]$ must be *trusted* or *certified*. The verification equation for such a scheme is as follows:

$$c \overset{?}{=} \mathcal{H}(m, A, B)$$

where,

1. c is the message digest;
2. $\mathcal{H}$ is a secure hash;
3. m is the set of messages;
4. $[c, r]$ is the signature ciphertext; and
5. $A = y^c g^r$ and $B = u^c v^r$ are the symmetric keys.

The integrity goal of this scheme can be expressed as follows:

$$\Big((([g, y] \overset{[c,r]}{\rightarrow} A) \wedge ([v, u] \overset{[c,r]}{\rightarrow} B)) \overset{c}{\rightarrow} m\Big) \tag{3.2}$$

The symmetric keys A **and** B provide integrity service to m. It is crucially important to note that $[g, y]$ or $[v, u]$ *must be certified* (using some private or public certification scheme) before any integrity deductions can be made. The protocol *associates* the integrity of $[g, y]$ (or $[v, u]$) with the integrity of $[v, u]$ (or $[g, y]$). Once this association is made and the *absolute* integrity of at least one of the key tuples is deduced, then the integrity of the symmetric keys $[A, B]$, and thereby the message m, can be deduced. Without certification of any of the keys, no meaningful deductions on the integrity service can be made. Note that this requirement is *inherited* from the Schnorr signature scheme represented in Equation 3.1.

3.1.2 Characterising Schnorr-Type Blind Signature Schemes

The blind signature technique [28] allows an entity to obtain a signature tuple on a message from a signer without revealing either the signature tuple or the message. This allows the entity to prove to any other entity that it was authorised by the signer without revealing its identity – the entity is anonymous.

A well known method to obtain a blind signature requires the signer to engage in a honest-verifier zero-knowledge identification protocol with the receiver (of the signature), who would play the role of a *skewed honest-verifier* to obtain the blind signature. Chaum and Pedersen [18] demonstrated the technique to obtain a blind Schnorr signature, which was later modified by Brands [12] to obtain a specialised version called restrictive blind signature. A goal of this chapter is to characterise both these schemes in order to highlight their subtle and important properties, which are ignored by some protocol designers. This oversight introduces many deficiencies in the integrity goal of the resulting cryptosystem as will be presented in Section 3.3.1.

A Schnorr-type blind signature was first proposed by Chaum and Pedersen [18]. The signature tuple is the same as that of Schnorr signature scheme (see Section 3.1.1) and has the same signature verification equation. The only difference is that the signer *cannot know* the message that is being signed, which in the case of Schnorr signature is the symmetric key and *not the message itself.* This is a subtle point that should actually mean that the signer is authorising the symmetric key only and *does not necessarily* authorise the message that the symmetric key may provide integrity to – as was the case in the original Schnorr signature scheme. Interestingly, this problem has a counterpart in the key recovery research (and cryptologic research as a whole), where it is a difficult problem to *restrict the use of certified keys* [15, 54, 11].

Since the verification equation for a blind Schnorr signature is the same as the Schnorr signature scheme, the subtlety is introduced in the integrity goal by employing a *modifier*. This is because the blinding process provides the *confidentiality service* and the syntax presented in this chapter deals only with the *integrity service*. Since the blinding process does not alter the integrity goal of the protocol, any alteration of the representation of the integrity goal for the Schnorr signature, to introduce the subtlety, must be purely a convention. The best way to accomplish this requirement would be to introduce a modifier. In Equation 3.1, the message that is signed, m, is represented employing a modifier as $\overline{m}$. Syntactically, Equation 3.1 is otherwise unchanged. The integrity goal is represented as follows:

$$\left([g, y] \xrightarrow{\quad [c,r] \quad} A \overset{c}{\rightarrow} \overline{m}\right) \tag{3.3}$$

Note that the signature generation procedure may or may not be blinded[1], so the modi-

[1]In the case of an e-cash system the customer could engage in a normal Schnorr signature protocol

fier is intended only for the interpretation of a potential weakness in argument. In other words, the modifier is *a statement of intent* and *not of a fact.* In the previous equation, the modifier suggests that the signer *may have no control over* the message, m.

The restrictive blind signature by Brands [12] is similar to the blind Schnorr signature scheme [18], with an additional property that the signer guarantees the *structure* of the symmetric key, A. In the original proposal [12], the signer employs the Schnorr variant (by Chaum and van Antwerpen, see Section 3.1.1) represented by Equation 3.2 and guaranteed the representation (structure) of one of the symmetric keys with respect to the bases $[g_1, g_2]$. The verification equations employed by the merchant (during the spending phase) and the bank (during the deposit phase) in Brands' scheme are as follows:

$$\begin{aligned} c &= \mathcal{H}(A, B, z, a, b) \\ a &\overset{?}{=} g^r y^{-c} \\ b &\overset{?}{=} A^r z^{-c} \\ d &= \mathcal{H}_0(A, B, \cdots) \\ B &\overset{?}{=} g_1^{r_1} g_2^{r_2} A^{-d} \end{aligned}$$

where:

1. c and d are message digests;
2. $\mathcal{H}$ and $\mathcal{H}_0$ are secure hash functions;
3. $[A, z]$ is a temporary key pair;
4. B is a message;
5. $[a, b]$ is the symmetric key tuple blindly *authorised* by the bank; and,
6. $[g, g_1, g_2, y, y_1, y_2]$ is the public key of the bank such that $y = g^{x_B}$, $y_1 = g_1^{x_B}$ and $y_2 = g_2^{x_B}$, where x_B is the banks private key;
7. $[r, c]$ is the signature tuple by the bank; and,
8. $[r_1, r_2]$ is the signature tuple on B employing the key$[g_1, g_2, A]$.

with the bank, and the merchant cannot discern this fact.

The integrity goal of this scheme is represented as follows:

$$\Big(\big(([g,y] \xrightarrow{[c,r]} a) \wedge ([A,z] \xrightarrow{[c,r]} b)\big) \overset{c}{\to} \overline{B}\Big) \wedge$$
$$\Big([g_1,g_2,A] \xrightarrow{[r_1,r_2,d]} B \overset{d}{\to} [A,\cdots]\Big) \tag{3.4}$$

It can be read as: the bank authorises the symmetric keys $[a,b]$ using its public key $[g,y]$ and, $[A,z]$ by its *association* with $[g,y]$. The symmetric keys provide the integrity service to B. This is the joint statement of the first verification equation. This is a blinded operation. The second verification equation provides the integrity service to B by employing the public key $[g_1,g_2,A]$ and the ciphertexts $[r_1,r_2,d]$. B, in turn, provides the integrity service to a predetermined set of messages and A. This is not a blinded operation. The implicit assumption for the goal of this proposal is the *association* of the bases $[g_1,g_2]$ with the key A, which was a part of the key $[A,z]$ which was *associated* with $[g,y]$ by the blind signature process. Thereby, whoever possessed the signature (the first verification equation) must also possess the knowledge of the representation of A with respect to the base $[g_1,g_2]$ (just as the Schnorr signature scheme required the signer to possess the representation of the public key y with respect to the base g). This additional check allowed the bank (which took part in the signature generation process) to gain another implicit confidence: the blind signature transcript contains a valid, *hidden* identity that is a representation of the bases $[g_1,g_2]$. In the case of electronic cash systems employing blind signature, the merchant, without trusting the bank, *cannot* gain this knowledge as it can make no logical deductions about the withdrawal protocol (signature generation process).

3.2 The Cramer-Shoup Public Key System

The technique proposed in Section 3.1 will be useful for the analysis of any protocol that requires some form of integrity verification. In order to highlight this aspect, the Cramer-Shoup Public Key Encryption System [26], which was proposed to provide security against the adaptive chosen ciphertext attack, will be analysed. Security against this attack is related to the non-malleability property for public-key cryptosystems [35]. In a nutshell, if the owner (who knows the private key corresponding to the public key) can verify the *immutability* of the ciphertext after its formation, then the owner could defend from attacks that required decryption of modified ciphertexts, such as the adaptively chosen ciphertext attack.

The basic scheme of the Cramer-Shoup cryptosystem [26] operates on a group G of order a large prime q, such that the Diffie-Hellman decision problem is intractable. Suitable generators $g_1, g_2 \in_R G$ are also chosen. The public key of the receiver will be of the form:

$$h = g_1^z, c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}$$

where $z, x_1, x_2, y_1, y_2 \in_R \mathbb{Z}_q$ are the private keys. The ciphertext 4-tuple is of the form:

$$u_1 = g_1^r, u_2 = g_2^r, e = mh^r, v = c^r d^{r\alpha}$$

where $r \in_R \mathbb{Z}_q$ is a random number, $\alpha = \mathcal{H}(u_1, u_2, e)$, and $\mathcal{H}$ is a universal one-way function.

The receiver responds to only those ciphertext tuples that conform with the following verification equations:

$$\begin{aligned} \alpha &= \mathcal{H}(e, u_1, u_2) \\ v &= u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} \end{aligned}$$

If the above equations are satisfied, the receiver decrypts the ciphertext as $m = e/u_1^z$, which is the ElGamal decryption algorithm, to obtain the message. Employing the methodology proposed in Section 3.1, the integrity goal of the verification equations can be represented as follows:

$$\left([x_1, y_1, x_2, y_2] \overset{[\alpha,v]}{\rightarrow} [u_1, u_2] \overset{\alpha}{\rightarrow} e\right) \tag{3.5}$$

That is the symmetric keys $[u_1, u_2]$ provide integrity to e employing the ciphertext α and the (private) keys $[x_1, x_2, y_1, y_2]$ provide integrity to the symmetric keys $[u_1, u_2]$ employing the ciphertexts $[v, \alpha]$. It is interesting to observe the *structural* similarity of this representation with that of Schnorr signature tuples in Equation 3.1. That is both the Cramer-Shoup verification algorithm and the Schnorr signature verification algorithm employ the same verification structure, which is:

$$Key1 \overset{[a,b]}{\rightarrow} Key2 \overset{a}{\rightarrow} message$$

The identification of such similarities is a useful feature of this representation. Since the Cramer-Shoup algorithm employs a similar structure to the Schnorr signature algorithm for the provision of integrity service, it would be worthwhile to investigate if an encryption scheme secure against adaptive chosen-ciphertext attack can be devised by employing the Schnorr signature scheme.

3.2.1 A New Scheme

An alternate encryption system to the Cramer-Shoup cryptosystem [26], with robust integrity verification mechanism and similar verification structure, exists. The remainder of this section will present such a proposal.

System Settings A cyclic group G of prime order q is chosen such that the Schnorr signature algorithm and ElGamal encryption algorithm are secure. The certified public key of the receiver contains $y = g^x$, $y_1 = g^{x_1}$ such that $x \neq x_1$ (and so $y \neq y_1$) for some $x, x_1 \in_R \mathbb{Z}_q$ and $g \in_R G$, so that $y, y_1 \in G$.

Encryption Algorithm The sender performs the following calculations to securely send a message m:

1. choose at random, $r, r_1 \in_R \mathbb{Z}_q$, and compute $u = g^r$, $e = my^r$ and $u_1 = y_1^{r_1}$;
2. compute $\alpha = \mathcal{H}(e, u_1, u)$, by employing a secure hash algorithm $\mathcal{H}$; and,
3. compute $v = r - r_1\alpha \bmod q$ and send (u_1, e, α, v) to the receiver.

Integrity Verification and Decryption Algorithm The decryption algorithm performs the following calculations when the ciphertext tuple (u_1, e, α, v) is available:

1. compute $u = u_1^{\alpha x_1^{-1}} g^v$;
2. verify, $\alpha \stackrel{?}{=} \mathcal{H}(e, u_1, u)$;
3. compute message, $m = e/u^x$;

If the verification is successful, the receiver responds to the sender, else the receiver rejects the tuple.

Completeness and Efficiency Analysis The verification equation holds because:

$$u_1^{\alpha x_1^{-1}} g^v = g^{x_1 r_1 \alpha x_1^{-1} + v} = g^{\alpha r_1 + v} = g^r = u$$

The above equation explains the reason for having unequal private keys, $x \neq x_1$. Otherwise, the value u^x, the shared Diffie-Hellman key, can be computed by employing the public values. Thus, y can be considered to be a unique public-key meant for confidentiality services and y_1 to be a unique public-key meant for integrity services.

The representation of the verification equation in our proposal would be similar to Equation 3.1, that is:

$$\left(([g, u_1, x_1] \overset{[\alpha,v]}{\rightarrow} u \ \ \overset{\alpha}{\rightarrow} \ \ [e, u_1]\right) \tag{3.6}$$

Clearly, the ciphertext can be verified only if the private-key x_1 is available. Part of the ElGamal ciphertext, u, which is used as the symmetric key for a Schnorr-type signature scheme, can be computed only with the knowledge of x_1. The inclusion of private-keys, in Equations 3.5 and 3.6, in the starting node of keys (the left-most set of elements) abstracts this notion.

The performance comparison between the basic scheme of the Cramer-Shoup system and the alternate proposal is presented in Table 3.1. In the table, $t = \log_2 |G|$ is the number of bits required for the representation of elements in G, $l = \log_2 |\mathbb{Z}_q|$ is the number of bits required for representation of elements in $\mathbb{Z}_q$ and L be the information in bits required for the representation of the respective group information. It terms of key size, the alternate proposal requires smaller public and private keys. In term of performance, the alternate proposal achieves less number of exponentiations for the sender as suggest in Table 3.1.

Parameter	Cramer-Shoup System	Alternate Proposal
No. of sender exponentiations	5	3
No. of receiver exponentiations	3	3
Size of private key	$5l$	$2l$
Size of public key	$5t + L$	$3t + L$
Size of ciphertext	$4t$	$2t + 2l$

Table 3.1: Table of Comparison

Security Analysis

The Cramer-Shoup cryptosystem and the alternate proposal achieve the same integrity goal (observe integrity representations in Equations 3.5 and 3.6): the sender proves the knowledge of the shared Diffie-Hellman key y^r in some fashion, and thereby the knowledge of the message m. Thus, the security against chosen ciphertext attack has led to non-malleable encryption [35] (due to the use of non-interactive proof of knowledge systems and the hash function) and plaintext-awareness [6]. An advantage of the Cramer-Shoup cryptosystem is the existence of a formal proof for its achievement of the non-malleability property. A prospective security proof for the alternate proposal could provide insight into its achievement, which may be plain-text awareness. Note that plain-text awareness is a stronger notion of security than non-malleability.

The ciphertext of the alternate cryptosystem is a 4-tuple, $X = (e = my^r, u_1 = y_1^{r_1}, \alpha = \mathcal{H}(e, u_1, u), v = r - r_1\alpha \bmod q)$. There are three independent security mechanisms employed by the alternate proposal:

The supplementary encryption scheme (SES): A message $w = g^{r_1}$ is encrypted under a public key $y_1 = g^{x_1}$ as $u_1 = y_1^{r_1}$ and the corresponding decryption being $u_1^{x_1^{-1}} = w$;

The Schnorr signature scheme (SSS): employs $w = g^{r_1}$ as the public key to achieve the non-malleability (immutability) property; and,

The ElGamal encryption scheme (EES): Given the values for e, $u = g^r$ and $y = g^x$, find an m such that $m = e/u^x$.

The ciphertext 4-tuple can be considered to be an *extended* ElGamal ciphertext $(e = my^r, u = g^r)$, where a part of the ciphertext is encrypted, $(u_1, \alpha, v) = \mathit{Enc}_{y_1}(u)$, where *Enc* employs SES and SSS as subroutines. Note that the ElGamal ciphertext $u = g^r$ is *not* sent in the clear, only e is sent in the clear. Intuitively, this double encryption must provide increased security than the original ElGamal proposal, which was employed without any modification in the Cramer-Shoup cryptosystem.

A security proof for the scheme proposed is not *currently* available. At the present juncture, this is the only disadvantage, compared with the Cramer-Shoup cryptosystem. The primary purpose for the inclusion of the proposal in this section is the illustration of the reasoning capabilities offered by the integrity verification technique.

3.3 Analysis of Protocols with Weak Integrity Service

The previous section provided a study for the design of new protocols by studying the integrity goals of existing protocols. This section will present the analysis of the integrity goals of the proposals for an efficient e-cash proposal in Section 3.3.1 and a fraud detectable key-recovery system in Section 3.3.2. The aim of this section is to highlight the usefulness of this technique in identifying protocol deficiencies.

3.3.1 Analysis of an Efficient E-Cash Proposal

Radu, Govaerts and Vandewalle [76] observed that Brands' restrictive blind signature [12] was computationally expensive due to the number of exponentiations it required, compared to the blind Schnorr signature [18]. Their proposal required less frequent computation of the restrictive blind signatures [12] and frequent computation of blind Schnorr signatures [18] to achieve a more efficient cash system. The proposal was a three-phased withdrawal mechanism briefly described as follows:

1. *get_pseudonym* protocol between the user and the bank to obtain a restrictive blind signature on a pseudonym, π, by employing the Brands withdrawal protocol (see Equation 3.4). This allows the bank to guarantee that the pseudonym π is derived from a registered identity π_0.

2. *withdraw_big_coin* protocol between the user and the bank allows the user to obtain a blind Schnorr signature (see Equation 3.3) on a `big coin` that associates a pseudonym, β with a valid long-term pseudonym π; and,

3. *exchange_big_coin* protocol between the user and the bank that allows the user to *anonymously* withdraw many `small coins` after providing the bank with a valid `big coin` and the corresponding long term pseudonym π.

The user can spend the `small coins` with any merchant. Radu *et al.* proposed the use of a smart-card during the spending protocol that will act as an observer [18] to prevent double spending of `small coins`. The certified public keys of the bank is represented by the tuple, $[g, P, P_1]$ such that the bank possesses the representation of P and P_1 to the base g.

The verification equations that the bank employs to verify the long-term pseudonym during the *exchange_big_coin* phase are as follows:

$$\begin{aligned} c &= \mathcal{H}(\pi, z, A, B) \\ A &\stackrel{?}{=} g^r P^c \\ B &\stackrel{?}{=} \pi^r z^c \\ d &= \mathcal{H}(\beta, \alpha) \\ \alpha &\stackrel{?}{=} g_1^{r_1} g_2^{r_2} \pi^d \end{aligned} \tag{3.7}$$

These are the verification equations of Brands' restrictive blind signature scheme discussed in Section 3.1.2. Note that the first three equations are the results of user participation in the *get_pseudonym* protocol and the last two equations are formed by the bank with the assistance from the user (and the smart-card) during the *exchange_big_coin* phase, which is *anonymously* executed by the user.

The verification equation that the bank employs to verify the `big coin` during the *exchange_big_coin* phase are as follows:

$$\begin{aligned} e &= \mathcal{H}(\beta, \pi, D) \\ D &\stackrel{?}{=} g^{r_3} P_1^e \end{aligned} \tag{3.8}$$

This is a blind Schnorr signature explained in Section 3.1.2 by Equation 3.3.

A `big coin` is considered valid if and only if it satisfies Equations 3.7 and 3.8. If the `big coin` is valid then the bank issues many `small coins`, which do not use Brands' scheme.

Protocol Deficiency: As stated previously in Section 3.1.2, a blind signature must be considered as an authorisation for a symmetric key and not for the message that could be serviced by the symmetric key. Radu *et al.* did not observe this caution in their proposal for an efficient e-cash. As will be shown, this oversight results in a weakness in their proposal that allowed unaccounted transfer of funds between accounts, that is the property of non-transferability is not achieved.

The verification equations (see Equation 3.7) for the long-term pseudonym employs Brands' restrictive blind signature scheme discussed in Section 3.1.2. These equations can be represented as follows:

$$\left((([g,P] \xrightarrow{[c,r]} A) \wedge ([\pi,z] \xrightarrow{[c,r]} B)) \xrightarrow{c} \overline{[\pi,z]}\right) \wedge$$
$$\left([g_1,g_2,\pi] \xrightarrow{[r_1,r_2,d]} \alpha \xrightarrow{d} [\beta]\right) \tag{3.9}$$

In the Brands' scheme, the symmetric key α (B in Equation 3.4) was serviced by c, which restricted the use of α to only one servicing – otherwise the private key of the user would be revealed (a deficiency of Schnorr-type signature schemes). Whereas, in the scheme proposed by Radu *et al.*, the symmetric key α was not serviced by c. Thereby the value for α can be changed to allow for multiple servicing of values of β by π.

The verification equations (see Equations 3.8) for the `big coin` employs the Chaum-Pedersen blind signature scheme (see Section 3.1.2, Equation 3.3). These equations can be represented as follows:

$$\left([g,P_1] \xrightarrow{[e,r_3]} D \xrightarrow{e} \overline{[\beta,\pi]}\right) \tag{3.10}$$

Note that the claimed association between a long term pseudonym, π, and the short term pseudonym, β, happens during this protocol. Radu *et al.*, analysed $[e,r_3]$ as a blind signature on $[\beta,\pi]$ by the key pair $[g,P_1]$, the certified public key of the bank. Hence they argued that association was authorised by the bank. The flaw in this argument is: $[e,r_3]$ is a blind signature on D and *not on* $[\beta,\pi]$. Referring to equation 3.10, clearly the integrity check relies on the *use of the key*, D, which was authorised by the bank, to associate the tuple $[\beta,\pi]$ and this problem is similar to the generic situation explained in Section 3.1.2. That is, the bank is *trusting* the user to correctly associate one of his/her long-term pseudonyms, π, with a short-term pseudonym, β. This allows the user to associate the π value of another user with the β value that resulted from his/her withdrawal. In effect, this would allow *unaccounted* money transfer between users, which may result in perfect black-mailing and/or money-laundering [92]. Although Radu *et al.* did not comment about the property of *non-transferability*[2] in their

[2]The property which is essential to prevent unaccounted transfer of funds.

paper, many practical monetary systems require this property for their proper functioning. Clearly, their scheme lacks the *non-transferability* property because of lack of integrity checks.

In order to visualise this problem let the long-term pseudonym of a black-mailer be π, which was derived from his/her long term identity π_0 using the *get_pseudonym* protocol. The black-mailer can perform the following actions to achieve a perfect-blackmail;

1. Allow the victim user to participate in the mutual-authentication protocol that takes place before the *withdraw_big_coin* transaction;
2. Logically or physically hijack the withdrawal terminal from the victim user to prevent him/her from registering the value of π; and,
3. Perform the *withdraw_big_coin* transaction with the bank as prescribed by the protocol, employing π as the pseudonym.

This attack would enable the black-mailer to possess an anonymous coin that identified the victim or the bank, unless the anonymity for all the customers of the bank is revoked.

3.3.2 Analysis of the Binding ElGamal Proposal

Verheul and van Tilborg [89] employed the proof of equality of discrete logarithms (see Equation 3.2, Section 3.1.1) in a novel manner to realise a key recovery system. A brief summary of this proposal is presented as follows.

Suppose that Ronald (borrowing the nomenclature for the entities from their proposal) in the USA wishes to use a key-recovery enabled public-key infrastructure to securely communicate with Margaret in the UK. Let A be the escrow authority in the USA and B be the escrow authority in the UK. The system operates with a common, prime-order multiplicative subgroup $G_q < \mathbb{Z}_p^*$ (where p and q are suitable prime numbers), with a common generator $g \in G_q$ chosen securely by trusted officials. Let the certified public key (elements of G_q) of Margaret be y_M, of A be y_A and, of B be y_B, such that the respective entities possess the representation of the public key with respect to the base g. In order to send a message, M, Ronald is required to perform the following operations:

1. Choose a (random) symmetric session key, S, and encrypt the message as $E = enc(S, M)$, where enc is a suitable encryption algorithm;

2. Choose a random value $k \in Z_q$ and encrypt S under the public keys y_M, y_A and y_B as: $C = g^k$, $R_M = Sy_M^k$, $R_A = Sy_A^k$ and $R_B = Sy_B^k$, which is the multi-ElGamal encryption ciphertext for three parties; and,

3. Form proof of equality of discrete logarithms to prove that:

$$\log_g C = \log_{y_A/y_M}(R_A/R_M) = \log_{y_B/y_M}(R_B/R_M)$$

   The transcripts of this proof system become the verification equations for the system that can be verified by any monitor.

Ronald sends the ciphertext and the proof in the clear (using a standard message format) to Margaret, who can decrypt (ElGamal decryption) the session key, S, from R_M and C and, decrypt the ciphertext E using S to obtain M. The verification equations, which were proposed to be checked by an off-line global-monitor, for their proposal are:

$$\begin{aligned}
c &= \mathcal{H}(E, C, R_A, R_B, R_M, D, F, I, \cdots) \\
D &\stackrel{?}{=} g^c C^r \\
F &\stackrel{?}{=} (y_A/y_M)^c (R_A/R_M)^r \\
I &\stackrel{?}{=} (y_B/y_M)^c (R_A/R_M)^r
\end{aligned} \tag{3.11}$$

where: $\mathcal{H}$ is a secure hash function, $[c, r]$ is a Schnorr signature tuple.

The representation for the verification equations of the key recovery scheme, using the notation presented in Equation 3.2, is as follows:

$$\begin{aligned}
&\Big((([g, C] \xrightarrow{[c,r]} D) \wedge \\
&([y_A/y_M, R_A/R_M] \xrightarrow{[c,r]} F) \wedge \\
&([y_B/y_M, R_B/R_M] \xrightarrow{[c,r]} I)) \xrightarrow{c} [E, C, R_A, R_B, R_M, \cdots]\Big)
\end{aligned} \tag{3.12}$$

It is evident that this representation is similar to the representation provided in Equations 3.1 and 3.2. Comparing the above representation with Equations 3.1 and 3.2, the following observations can be made:

1. none of the key pairs ($[g, C]$, $[y_A/y_M, R_A/R_M]$ and $[y_B/y_M, R_B/R_M]$) can be trusted because they are uniformly chosen by the sender (who is not trusted for certification procedures);

2. ratios of keys provide the integrity service to the symmetric keys F and I, which is not a *standard* assumption of Schnorr-type signatures.

These observations suggest a deficiency in the system that allows the sender to manipulate the keys, which were meant to be the *starting point* of the integrity service – that is if the starting point is corrupted then the integrity service that it transfers is also corrupted. This deficiecy can be used to attack the protocol.

Protocol Deficiency: Prior to discussing an attack on a key recovery system, the meaning of a non-trivial attack must be understood. A key recovery protocol is deficient if successful adversaries abide with the message formats suggested by the protocol and *procure legitimate services from the key recovery infrastructure* to ensure secure communication. For example, if a public-key based key recovery system provides robust certification mechanism, such as robust public key infrastructures, and requires key recovery enablement before the certification can be employed, then an adversary is successful when certified public keys are employed and key recovery is avoided. The attack on the proposal, by Verheul and van Tilborg [89], by Pfitzmann and Waidner [73] need *not necessarily* be an attack on the protocol proposed by Verheul and van Tilborg, rather it is an attack on all session-key recovery systems without any form of private-key recovery. It outlines the *generic* concealed-encryption attack[3] on key recovery protocols and *fails* to explain the manner in which the *concealed key* may be established. Although our attack exploits the property of concealed-encryption attack, it is not a generic attack on all session-key recovery protocols, rather it is a specialised attack on the proposal [89], *which resulted from an oversight in the protocol design.* Moreover, the manner in which an illegal session key can be established using *the key recovery infrastructure* will be explained. This distinction is important for protocol designers, who may employ the proposed fraud detection mechanism [89] for

[3]There is no technique available to check if a claimed key was used during the encryption process — verifiable encryption for symmetric key systems is not currently available.

a different application that may not have properties similar to that of key-recovery applications. For example, Abe [2] successfully employed a similar integrity verification mechanism for a mix network proposal.

Suppose that the sender and a hidden receiver ($\tilde{M}$) would like to communicate using the actual receiver (M) as the decoy. The sender can accomplish this by employing the following steps:

1. choose a random session key, $\tilde{S}$;
2. encrypt the message with $\tilde{S}$ to obtain the ciphertext, E;
3. obtain the public keys of the hidden receiver, y_H, the decoy, y_M and the authorities (y_A, y_B);
4. choose a random value for k;
5. compute a decoy session key, $S = \tilde{S}y_H^k/y_M^k$;
6. encrypt the decoy session key for the decoy and the authorities, $R_M = Sy_M^k = \tilde{S}y_H^k$, $R_A = Sy_A^k$, $R_B = Sy_B^k$ and $C = g^k$;
7. form the verification equation as suggested by the representation in Equation 3.12;
8. send the ciphertexts and verification parameters to decoy.

The hidden receiver performs the following steps:

1. wiretap the communication to decoy to obtain E, R_M and C;
2. obtain session key, $\tilde{S} = R_M/C^{x_H}$, where x_H is the private key of the hidden receiver; and,
3. decrypt E using $\tilde{S}$ to obtain the message.

The monitor will verify the equations properly, the decoy receiver and the authorities will retrieve the decoy session key, S, from the respective ciphertexts employing the respective private keys, and the decoy session key, S, will not decrypt E correctly. Also note that it will be difficult to find the hidden receiver, y_H, or the actual session key, $\tilde{S}$ (finding the hidden receiver would imply breaking the multi-ElGamal cryptosystem [89]).

3.4 Summary

A novel technique to represent the integrity goal of a system was presented by accounting for all the verification equations and ignoring the unnecessary protocol complexities that produced the equations. Also, the usefulness of this abstraction in encompassing the *unpredictability* of the protocol participants was demonstrated. The use of the technique was demonstrated by the identification of *similar* protocol deficiencies in *seemingly* different scenarios, by the application of the technique.

Many proposals for *compliant* systems tend to ignore the importance of the integrity service, while in pursuit of the confidentiality service. Blaze [10] formulated an attack on the integrity service in the Clipper proposal [88], which was predominantly focused on the confidentiality service. Unfortunately, many protocols in various fields of cryptologic application still succumb to attacks similar to those detailed in Sections 3.3.1 and 3.3.2, namely attacks exploiting weaknesses in integrity services. Integrity and confidentiality services must be given equal footing for the design of robust protocols. The proposed technique will assist protocol developers to identify and solve problems relating to the integrity service.

The use of the new technique is not just for the analysis of protocol deficiencies (as in Section 3.3), but also for the synthesis of protocols (as in Section 3.2). The prospective development of a *uniform syntax* for protocol constructs will greatly assist in the analysis *and* design of modern cryptologic systems.

Chapter 4

Key Recovery Systems

Secrecy is as indispensable to human beings as fire, and as greatly feared.
- SISSELA BOK
Swedish philosopher, "Secrets," 1983.

The concern of key recovery, or escrow, systems is the access to a confidential message. The three sets of players in the system are the users, the escrow agents and the law-enforcement agents. The users communicate confidential messages between themselves (or with outside users) and when a formal request is made the escrow agents provide access to the messages for the law-enforcement agents. Clearly, the requirements of the users and the law-enforcement agents are contradictory. The state is further complicated by the apparent mistrusting relationship shared by the users and the law-enforcement agents.

Key recovery systems nicely fit into compliance Category 1 due to their restricted confidentiality service and universal integrity service, essential to achieve the perceived requirements. This chapter will detail requirements of key recovery systems, analyse and propose problems, propose a new paradigm for key recovery systems, and present a scheme conforming with the paradigm.

4.1 Introduction

A key recovery system is a compliant cryptosystem, with the conflicting requirements being confidentiality service for the set of users and revocation of confidentiality service (controlled wiretapping) for the set of law enforcement agencies. The escrow agents assume the role of judges or trusted third parties, and the role of monitors is achieved using appropriate compliance checking mechanisms.

Key recovery was initially propounded as a mechanism for wiretapping confidentiality channels, in order to bridge two contradictory requirements, namely privacy for system users and ability to break this privacy by an *authorised* and *well defined* set of entities. The thesis identified a twofold argument for broader research into key recovery systems. Firstly, key recovery mechanisms [65] will be useful in other application scenarios as well – the most prominent of them being cryptographic key management. Secondly, researching key recovery techniques will facilitate better understanding of mechanisms for *revocation* of cryptographic services. The latter argument is more relevant for this thesis due to its interest in restricted confidentiality. Revocation is a logical tool to achieve restricted confidentiality, which, as discussed in Chapters 2 and 3, is essential for many other applications of cryptologic protocols.

In trying to bridge the contradictory requirements many proposals infringed on the security assumptions of cryptographic services that were unrelated (in a broad sense) to key recovery systems. For example, a fundamental assumption of public key cryptography is that only the owner of the public key should know the corresponding private key and many (unrelated) public key based protocols rely heavily on this assumption for their security. If one of these protocols contradicts the fundamental assumption, then all the (unrelated) public key protocols will fail to guarantee their security arguments. Advantageously, the goal of many public key protocols is *confidentiality* of some data or key information. The goal of key recovery protocols, on the other hand, is to *break* this confidentiality in a *controlled manner*. Traditionally, *breakable* services were not the aim of cryptographic protocols, in fact the protocols battled hard against activities that attempt to break a service. This deliberation resulted in many cryptographic systems that were unyielding to the concept of *revocation*, which is an essential component of many pragmatic systems — the most prominent of such systems being a public key infrastructure that supports revocation of certificates.

Notation: Some common notations used in this chapter are as follows.

$\mathcal{H}$: A cryptographically secure hash function.

h: A cryptographically secure keyed hash function.

$k \in_R \mathbb{Z}_p$: Choose k randomly from the congruence class *modulo* p.

$||$: String concatenation operator.

$\mathbb{Z}_p^*$: Group of non-zero integers modulo p.

$C = enc(K, M)$: Symmetric key encryption of the message M with the key K.

$M = dec(K, C)$: Symmetric key decryption of the ciphertext C with the key K.

$E = Enc_y(M)$: Encryption of the message M with the public key y.

$M = Dec_x(E)$: Decryption of the ciphertext E with the private key x.

4.2 Properties of Key Recovery Systems

The essential properties for the design of key recovery system are:

Compliance: All messages communicated between the set of users, in a confidential manner by employing the legal services and message formats, must be accessible to the law enforcement agents with the assistance from the recovery agents. There may be additional compliance guarantees, but this is the fundamental guarantee that all key recovery schemes strive to achieve.

Enforceability: Only the *intended receiving* party can access the confidential message: and this is possible if and only if the law-enforcement agents, with the assistance from the escrow agents, can access the same confidential message.

Traceability: The law-enforcement agents must be able to determine the destination (and optionally the source) of the message format without ambiguity. This requirement is a logical antecedent for the enforceability and compliance properties of this application domain. The nature of the property of a key recovery system will effectively decide the level of anonymity that the users may obtain. If the system is designed to employ a global monitor then there will be no anonymity for the source and the sink of the message.

In order to avoid complicated design, this thesis designed a monitor that employs global verification techniques. The resulting monitor can be redesigned to employ a restricted verification technique, so as to provide *trust-based anonymity*[1] for the users by

[1]The users need to trust the monitor for the anonymity service (confidentiality service for their identity).

designing suitable confidentiality services for the diagnostic elements – the elements that provide integrity service to the security objects contained in the message format.

Key recovery systems can be classified based on the nature of the keys that are recovered. There are three types of key recovery systems:

1. Long-term key recovery systems: recover the long-term secret keys such as a private key corresponding to a certified public key;
2. Short-term key recovery systems: recover the short-term or ephemeral keys such as the session keys established using certified long-term keys;
3. Hybrid-key recovery systems: fully recover the short-term keys and partially recover the long-term keys such that the owners of the long-term key have an *exclusive* knowledge of a part of the long-term key and an authority has an *exclusive* knowledge of the other part of the key. Both the owners and the authority must collaborate to obtain a desired service. This category was identified by this research.

4.3 Private Key Recovery

Private key recovery is a long-term key recovery system, where the private key corresponding to the certified public key is recovered. That is the users are expected to surrender their private keys to the escrow agents in return for the certification of the corresponding public key. A number of proposals for software key escrow [53, 63, 94] opted to use public key certification as the compliance mechanism by escrowing the private key of users before their registration. The general idea is that, to benefit from the public key infrastructure provided by the system, the users must have their public keys certified, and this certification is only available if the corresponding private key is escrowed. Such systems do not require large amounts of storage space, as would a session key recovery system, and are relatively easy to implement.

However, escrowing private keys has many disadvantages. The most prominent drawback is that once the private key is recovered the users can have no control over the period of key-recovery capability for law-enforcement: past, present and future communications of the user can potentially be tapped unless the long-term key is changed regularly. Also, the security of the database used to store the private keys of users will be critical for confidentiality, identification and escrow services. Since the private keys are stored in the database only for escrow services, the security of the database must not affect the confidentiality and identification services. Moreover, users have to place considerable trust in the authority or escrow agents that will be disproportionate to the service they obtain from the system. In fact, such systems *cannot* provide robust key escrow while at the same time protecting essential user rights.

4.3.1 A Time-limiting Key Recovery Proposal

As stated previously, the recovery of private keys allows the escrow agents to access confidential messages without any restraint on time. The fundamental concept behind key escrow proposals is to protect confidentiality of the honest citizen and *revoke* it from the dishonest citizen. While many schemes can be devised to grant or revoke the confidentiality service for selected users (citizens), the judgment of whether a citizen is honest or dishonest can only be reached with human involvement. This seems to be one of the weak links in any escrow system. A person in the government might be honest when the government is in control, but when another government takes over (possibly by a coup) the same person may be viewed as dishonest. This observation is applicable for all citizens, even for government officials who might *control* the escrow system. For any escrow system to be complete it should address this problem.

The main problem related to this phenomenon is decryption (using the escrow mechanism) of ciphertexts that were intercepted in the *past*. The Clipper proposal [88], detailed in Section C.2, acutely suffered from this weakness. In the proposal, when the law enforcement agency (LEA) obtains a *single* court order it can decrypt past, present and future communications from/to the target without any form of *restraint*.

Limiting escrow activity in time is essential for escrow systems [57, 48]. Many proposals relied on tamper-resistant hardware (or software) to accomplish this requirement. Reliance on tamper-resistance, especially in software, is difficult and will affect scalability of the implementation. Many proposals [11] relied only on certification procedures to accomplish the goal of the protocol. The discussion in this section is on such schemes.

Burmester, Desmedt and Seberry [13] proposed a multi-party protocol that required the citizen, LEA and *all* the trustees to be available during the set-up phase. The proposal will henceforth be referred to as the BDS scheme. A brief summary of the BDS scheme is presented in Appendix C.1. A novel scheme will be devised, such that the trustees need not be on-line during the registration phase, by employing publicly verifiable encryption. This approach results in a more robust system in which trust on the trustees is minimal. This approach is in-line with the philosophy for design presented in Section 4.2, which required all integrity protecting mechanisms to be publicly verifiable.

The BDS scheme consisted of a key escrow system that was claimed to limit the time span of wiretapping. The driving argument in the paper was that *the trustees could be compromised* at some point in time. It was assumed that *at least a minimum number* of the trustees will be honest in *erasing* the old share of the private key after computing the new share from the old share. The argument that the trustees could be compromised, may result in severe repercussions on the trust model of the system. The actual duties for which the trustees are trusted was not clearly mentioned in their paper. These reasons directly contribute to an attack on the system when a citizen (possibly an influential government officer) conspires with *at least a minimum number* of the trustees, to avoid escrow and still get his/her public key certified. In the l-out-of-l model that was detailed in BDS scheme, the minimum number is one.

According to their scheme, citizens can periodically update the private keys and at the same instance the trustees can simultaneously update the respective shares. Also, if at least one of the trustees erases its old share, then it will be difficult to compute the old private key from the existing shares. Only the new private key can be reconstructed. This property is achieved using a homomorphic, one-way function. In the BDS scheme squaring in a composite modulus was used for the design of such a homomorphic one-way function. Proof of the Diffie-Hellman relationship, $DH(g^{s_1}, g^{s_2}) = g^{s_1 s_2}$, was used in the BDS scheme to generate proofs for correctness of the shares generated by the citizens. This proof can be converted into a non-interactive protocol by employing suitable hash algorithms, as suggested by Fiat and Shamir [37].

A Potential Pitfall

The underlying assumption for the development of the BDS scheme was that the trustees could be compromised at some instance of time, but the protocols for the three phases assumed complete trust in the trustees. These contradictory assumptions in the design of the system are serious flaws. Moreover, it is very difficult to place complete trust in any entity in practice. Secure systems should place minimal trust in necessary parties in a protocol and explicitly mention the assumptions on trust relationships. Consider the forms of attack that allow a citizen to by-pass escrow by conspiring with some of the trustees, and still *use* the system in such a way that the identity of the conspiring trustees cannot be found. There are three potential break-points in the system that could be the foci of such an attack. They are;

1. In the set-up (or registration) phase the LEA *has to unconditionally trust* the trustees to report fraud against the user when they do not receive the discrete logarithm of $\{z_i | i = 1, \cdots, l\}$, the user published in the bulletin board. An attack could allow the user to give a *wrong share* to the trustee and still get his/her public key certified. No mechanism was proposed that would allow any neutral party to detect this fault.

2. In the up-date phase there is no publicly verifiable proof that the trustee will update the shares as prescribed by the protocol. The protocol relied on an implicit trust in the trustee for this update. We note that the only trust on the trustees that was explicitly mentioned in the BDS scheme was the deletion of old shares after computing new shares.

3. In the key recovery phase there is no publicly verifiable proof that will guarantee that the trustee will use the correct value of its share $\{s_i | i = 1, \cdots, l\}$. Some of the trustees could use a wrong value of the share that will prevent legal access to the plaintext and be unidentified.

An Improved Proposal

Publicly verifiable proofs can be employed so that any number of neutral entities (monitors) can check the correctness of the registration phase and detect malicious parties.

The proposal of Asokan *et al.* [3] is the only encryption algorithm independent and efficient publicly verifiable encryption which is known. The pseudocode to achieve an off-line version of their proposal is presented in Appendix B.2. Due to the verifiable encryption mechanism the improved scheme does not require the existence of secure channels between citizens and trustees as was the case in the BDS scheme. Since the improved scheme is an extension of the BDS scheme, it inherits all its security advantages.

System Settings System settings are essentially similar to that of the BDS scheme, except for certain additions to the existing parameters. The LEA is trusted to execute the prescribed protocols faithfully. The LEA sets up a public key infrastructure that can be used for secure communications with the trustees. The public keys of the trustees $\{y_i | i = 1, \cdots, l\}$ (corresponding to the private keys $\{x_i | i = 1, \cdots, l\}$) are certified and registered in a public directory. At least a minimum number of the trustees are trusted to change their public-private key pair periodically, publish the new public key and erase the previous private key. This is essential to avoid decryption of the encrypted shares sent to the trustees using their public keys at an arbitrary point of time.

Set-Up Phase Citizen j generates a large prime number p_j such that $p_j = 2p_{j1}p_{j2}+1$, where p_{j1} and p_{j2} are large primes. Also, $p_{j1} \equiv p_{j2} \equiv 3 \bmod 4$, so that -1 is a quadratic non-residue in the fields $\mathcal{Z}_{p_{j1}}$ and $\mathcal{Z}_{p_{j2}}$. Let $n_j = p_{j1}p_{j2}$. The citizen then chooses $g_j \in \mathcal{Z}_{p_j}$ that is a generator of $\mathcal{Z}_{p_j}$ and its private key $x_j \in_R \mathcal{Z}_{n_j}$. He/she then computes the public key as $y_j = g_j^{x_j} \bmod p_j$. The public data will be $\{g_j, y_j, p_j\}$. The private data will be $\{x_j, p_{j1}, p_{j2}\}$. The citizen and the LEA engage in a protocol that has the following steps;

1. Citizen: Computes the shares for the trustees as $x_j = \Pi_{i=1}^{l} s_i \pmod{p_j - 1}$ and performs verifiable encryption of the shares as, {VerEnc with input (s_i, g, p, y_i) and output$(c_i, D_i, P_{1_i}, \cdots, P_{80_i})$ $|i = 1, \cdots, l\}$. Sends $\{c_i, D_i, P_{1_i}, \cdots, P_{80_i} | i = 1, \cdots, l\}$ to the LEA.

2. LEA: Checks the proofs of verifiable encryption as {CheckVerEnc with input $(c_i, p, g, D_i, P_{1_i}, \cdots, P_{80_i})$ and output$(\mathit{check}_i) | i = 1, \cdots, l\}$. If any of the check_i is $FAIL$ then signals error message to the citizen and terminates the protocol.

3. Citizen: Sends $z_{1,\cdots,i} = g^{s_1,\cdots,s_i}$ and the proofs of $z_{1,\cdots,i} = DH(z_{1,\cdots,i-1}, D_i)$ for $i = 2, \cdots, l$ to the LEA.

4. LEA: If proofs for all the Diffie-Hellman relationships are correctly verified, certifies $y_j = z_{1,\cdots,l}$ as the citizen's public key in the system; forwards the verifiable encryption $\{c_i, D_i, P_{1_i}, \cdots, P_{80_i}\}$ to trustee i, who can decrypt it with the knowledge of x_i as DecryptVerEnc with inputs $(x_i, c_i, P_{1_i}, \cdots, P_{80_i})$ and output(s_i), which is the share of the citizen's secret key. The LEA stores the value of D_i against trustee i's identity along with citizen j's identity.

In the above protocol the LEA need not trust any other entity to check the correctness of the proofs. Moreover, any other neutral entity can verify the correctness of this protocol due to the presence of publicly verifiable proofs.

Update Phase

1. Citizen: Computes $x_{j_{new}} = x_j^2 \pmod{p_j - 1}$, computes $y_{j_{new}} = h_j^{x_{jnew}} \bmod p_j$ and proves the relationship $y_{j_{new}} = DH(y_j, y_j)$ to the LEA.

2. LEA: Temporarily stores $y_{j_{new}}$ in local directory along with the citizen's identity.

3. Trustees: Compute $\{s_{i_{new}} = s_i^2 \pmod{p_j - 1} | i = 1, \cdots, l\}$, compute $\{D_{i_{new}} = g_j^{s_{inew}} \bmod p_j | i = 1, \cdots, l\}$ and prove the relationship $\{D_{i_{new}} = DH(D_i, D_i) | i = 1, \cdots, l\}$ to the LEA.

4. LEA: Certifies the new public key of the citizen, replaces the old value of the public key with the new value in the public directory of the citizen, and updates the local directory by replacing the value of D_i with the value of $D_{i_{new}}$.

5. Trustees: Delete and *forget* the old shares.

The modified update protocol enforces the correct and synchronised update of shares when public key is updated; this enforcement was absent in the BDS scheme. The trustees have to be trusted to perform step 5 correctly, as there are no known techniques to provide such guarantees.

Key Recovery Phase The users are expected to use the ElGamal cryptosystem to securely communicate using certified public keys. The ciphertext in this system will then be of the form $(g^k, My_a^k) = (A, B)$ for the public key (y_a, g, p). When the LEA obtains a court order to wire-tap the communication of a user, it intercepts the ciphertexts sent to the user. The ciphertext component A along with the court order are sent to the trustees. The trustees then engage in a multi-party protocol to compute y_a^k from A using their respective shares by computing $C = A^{\prod_{j=1}^{l} a_i}$, where a_i is the share held by the trustees at that time. When the LEA is given C it can compute the message as $M = B/C$.

The LEA intercepts the ciphertext pair $(A, B) = (g_j^k, My_j^k)$ sent to citizen j, obtains a court order to wiretap the citizen's communication and presents A along with the court order to the LEA. The LEA then engages in the key recovery protocol, explained in the previous paragraph, with the trustees. If decryption fails after this protocol, then each trustee proves that it used the correct value of its share using the proof for equality of discrete logarithm. This proves that $\log_{g_j} D_i = \log_{E_{i-1}} E_i \bmod p_j$, which ensures that the trustees have used the discrete logarithm of D_i (the share s_i) to compute E_i from E_{i-1}. The LEA and the trustees engage in the following protocol;

1. LEA: Sends A to trustee 1.

2. Trustees: Compute $\{E_i = E_{i-1}^{s_i} \bmod p_j | i = 1, \cdots, l\}$, where $E_0 = A$, and compute proof of equality of discrete log as {LogEq with input $(s_i, g_j, E_{i-1}, D_i, E_i, p_j)$ and output $(d_i, e_i) | i = 1, \cdots, l\}$. Send $\{E_i, d_i, e_i\}$ to the LEA.

3. LEA: Computes {CheckLogEq with input $(d_i, e_i, g_j, E_{i-1}, D_i, E_i, p_j)$ with output $(\mathit{check}_i) | i = 1, \cdots, l\}$. If check_i is $FAIL$ register fraud against trustee i.

4. LEA: Computes $M = B/E_i \bmod p_j$.

This protocol guarantees message recovery or identification of malfunctioning trustee which ever the case may be.

Security Analysis The security of the improved proposal relies on the security of publicly verifiable encryption [3] and the BDS scheme.

Theorem 4.1 *Nobody except the corresponding trustee can obtain information about the private key of the user from the verifiably encrypted ciphertexts if the verifiable encryption of Asokan et al [3] is secure.*

Proof: The users send the shares of the private-key verifiably encrypted (as proposed by Asokan, Shoup and Waidner [3]) under the public-keys of the corresponding trustees in the first step of the set-up phase. The users are not required to reveal their private-keys in any other manner. □

Theorem 4.2 *No citizen can obtain a valid certificate without legal escrow of the private key, even by colluding with a minimum number of trustees.*

Proof: In order to avoid key escrow and at the same time obtain a valid certificate, the citizen must be able to perform any one of the following:

1. Generate wrong proof that will pass the verification procedure of verifiable encryption, so that a wrong pre-image of the commitment (g^{s_i}) for the encryption can be sent to the authorities. Since the verifiable encryption technique in [3] is assumed to be secure, this will not possible.

2. Generate wrong proof that will pass the verification procedure to prove Diffie-Hellman relationship [13], so that wrong value of shares can be encrypted for the escrow agent. Since the proof in the BDS scheme is assumed to be secure, this will not be possible. □

Theorem 4.3 *No trustee can use wrong value of the share during key recovery phase and be unidentified, due to publicly verifiable proof of knowledge.*

Proof: If the trustee uses a different value in the key recovery phase, it must be able to generate wrong proofs for the proof of equality of discrete logarithms to avoid identification. Since the proof of equality of discrete logarithm is assumed to be secure, malicious trustees cannot remain unidentified. □

Computational Requirements The robustness of the improved proposal is a result of extra computations. The inclusion of publicly verifiable encryption in the set-up phase is the major source of the computational overhead. Since the set-up protocol is performed only once per user, it is not considerable when the robustness of the protocol is taken into account. The computational requirement for the publicly verifiable encryption is;

Prover: The major computations that the prover has to perform are $2N + 1$ hash computations and N exponentiations, where N is the security parameter of the protocol, which is 80 in the improved proposal. Asokan *et al* [3] suggest that each party can do this using under 2000 modular multiplications.

Verifier: The verifier has to perform $2N + x + 1$ hash computations and N exponentiations, where x is the number of $1'$s in the challenge c.

The extra computational overhead in the update phase in the improved proposal as compared to the BDS scheme, is due to the Diffie-Hellman relationship proof that has to be performed once by each trustee and l times by the LEA. In the key recovery phase, if message recovery is successful then computational overhead will be zero.

Observations

The time-limiting escrow mechanism by Burmester, Desmedt and Seberry achieved (assuming some trust relationships) the much required time-limiting property in private key recovery systems. At the same time, both the original scheme and the improved scheme will succumb to the problems detailed in the next section. This is because both the schemes are fundamentally private key recovery schemes and, as will be explained in the next section, the concept of private key recovery may fail due to various reasons.

Comparing the original scheme and the improved scheme, valuable information about key recovery systems (and the systems belonging to compliance Category 1, in general) can be obtained concerning public verifiability of the integrity service. Public verification seems to be fundamentally essential in environments where very weak trust relationships are noticeable. Public verification (or any form of verification for that matter) is an expensive procedure. Thus, a pragmatic design may require a suitable trade-off between the amount of trust and efficiency of operation.

4.3.2 Private key recovery and digital signatures

A fundamental principle of escrow systems is that only keys for confidentiality should be escrowed, while keys used for signing and authentication should not be. This is because authorities have no need to access such keys and, moreover, it puts in doubt the integrity of any signed message purporting to come from the key owner. This extends to a loss of the non-repudiation service, the lynchpin of electronic commerce services. Therefore, if the private keys of users were to be escrowed then the same public key infrastructure cannot be used for verification of digital signatures. There will be two entities (the user and the escrow agent) possessing the secret key corresponding to a public key, which contradicts the assumption for the existence of digital signatures.

A logical solution would be to set up a separate unescrowed, public key infrastructure for digital signatures. However, this solution would allow the users to effectively bypass the escrow procedure by using the unescrowed public key infrastructure (intended for digital signatures) to communicate. This is because all the well known public key signature schemes have public keys for which corresponding encryption algorithms are known. Recently, Young and Yung [95] suggested a possible methodology for the design of public keys that can be employed only in signature systems. Even in the case that no encryption algorithms were known for such public keys, signatures alone are sufficient to set up keys for confidentiality using well known protocols such as STS [32].

Open question: Can robust private key escrow systems and digital signature systems co-exist?

In a key escrow system that adequately protects the rights of individuals it is not reasonable that the private key is known to the authorities. Argument for the opposite case also exists, namely that for robust key escrow it is not possible for private keys to be known to individuals for reasons to be explained in the following section.

4.3.3 Key conversion attack

Micali [63], Kilian and Leighton [53], Young and Yung [94], and others have proposed escrow systems that rely only on mechanisms escrowing the private key of entities.

This mechanism by itself will not guarantee key escrow. Strong traceability and enforceability mechanisms are also essential constituents of escrow systems. A generic attack on private key escrow systems without any mechanism for traceability and enforceability called the *key conversion attack* will be detailed in this section. Users employing this attack can convert to a new pair of keys from the certified (escrowed) keys to realise secure communications without key escrow. Since this conversion will be based on the certification procedure, the communicating parties avail the certification service offered by the escrow system. The basis for the attack is the existence of secure *proxy public keys*.

A proxy signature or cryptosystem allows the owner of a certified public key to delegate to a proxy, the power to sign or decrypt documents. Mambo *et al.* [60, 59] present a detailed discussion on this topic, in which proxy keys for both RSA and discrete logarithm based algorithms are constructed.

The semantics for the key conversion attack are essentially similar to proxy cryptography. A new key pair is derived from the certified public key for encryption services and the owner of the public key assumes the role of the proxy, which is the case of self-proxy. Henceforth, all calculations will take place in the group $\mathbb{Z}_p$, unless stated explicitly.

Let y_{old} be the public key corresponding to the escrowed private key x_{old}, which is the discrete logarithm of y_{old}. The equations for a key conversion attack may require the following equations;

$$y_{new} = f_1(y_{old}, r),$$
$$x_{new} = f_2(x_{old}, f_3(r))$$

where r is a random number and f_1, f_2 and f_3 are publicly known functions.

A mechanism for a key conversion attack based on the proxy public key system [60] is presented. In this discussion g is a generator of the group $\mathbb{Z}_p^*$. To convert to a new key pair the following operations are performed by the owner of the public key and collaborating entities;

Owner: Select $k \in_R \mathbb{Z}_p$.

Compute $K = g^k$.

Compute the new secret key, $x_{new} = x_{old} + kK \bmod (p-1)$.

Broadcast K.

Sender: Obtain y_{old} from a registry and compute $y_{new} = y_{old}K^K$.

Encrypt the message, M, using the new public key $E = E_{y_{new}}(M)$.

Send encrypted message E to the owner.

Owner: Decrypt the message using the new private key, $M = D_{x_{new}}(E)$

Theorem 4.4 *If there exists a secure algorithm for the generation of proxy key-pair for a public-key cryptosystem employing a random value, then the key conversion attack can bypass any private key escrow scheme that employs the public key cryptosystem.*

Proof: Assume that the participant and the escrow agents know the private key corresponding to the certified public key of the participant.

If there exists a secure algorithm for the generation of proxy keys for a public key cryptosystem employing a random value, then the participant can generate a secure proxy key-pair. The participant can send, in clear-text, the proxy public-key to its peer. Since the escrow agents, with a overwhelming probability, cannot guess the random value, they cannot compute the private-key corresponding to the proxy public-key. The authenticity of the proxy public-key can be validated by employing the certificate for the original public key. Therefore, with a high probability, the participant can circumvent the private key escrow mechanism. □

Since it is believed that the proxy keys of Mambo *et al.* [60] are indeed secure, the claim holds true in the situation that discrete logarithm based public keys are used. It is evident that users may perform the key conversion attack by publishing a K value of their own choosing. Notice that it is an assumption that users may not have any other means to authenticate the K value as belonging to the correct user. However, the property of the proxy key ensures that whatever value of K is used in encryption, messages may be decrypted only with knowledge of the certified private key. Therefore, confidentiality of the message is ensured, although reception by the intended receiver is not.

It is clear that any party, in particular a malicious escrow agent, can publish a K value claiming to be from the user. In the general case this means the encrypted message can never be decrypted by any party. In the case when the escrow agent publishes K, it alone will be able to decrypt the message. Now since the escrow agent always has the power to decrypt, and can also be assumed to control communications to the user, this actually achieves nothing that cannot be done without use of a proxy key. However, these arguments lead to the conclusion that an escrow agent who wishes to prevent the attack must prevent all communications from taking place between legitimate parties.

Analysis of a Private Key Recovery System

An analysis of the auto-recoverable, auto-certifiable cryptosystems by Young and Yung [94] for private key escrow in software systems will be presented. In order to highlight the broad applicability of the key conversion attack, a drawback of the scheme by Young and Yung, henceforth termed as YYS, will be detailed. The drawback will allow rogue users to avail cryptologic services from the certification infrastructure *and* by-pass key recovery.

System Settings: The system set-up for YYS consists of a common prime modulus r, such that $q = 2r + 1$ is a prime and $p = 2q + 1$ is a prime. The value of r is chosen to render the discrete logarithm problem intractable in the groups $\mathbb{Z}_{2q}^*$ and $\mathbb{Z}_p$. Suitable generators $g \in \mathbb{Z}_p$ and $g_1 \in \mathbb{Z}_{2q}^*$ are determined and published as system parameters. Each escrow authority, EA_i, chooses $z_i \in_R \mathbb{Z}_{2r}$ and computes $Y_i = g_1^{z_i} \bmod 2q$. The common public key of all the escrow authorities is calculated as $Y = \prod_{i=1}^{m} Y_i$, where m is the number of authorities, and the certificate for the tuple $(Y, g_1, 2q)$ is published securely. Note that the private key corresponding to Y is $z = \sum_{i=1}^{m} z_i \bmod 2r$.

Key Generation: The key generation algorithm encrypts the private key under the public key of the authorities, Y, using the so-called "double decker" encryption. Each participant chooses $k \in_R \mathbb{Z}_{2r}$ and computes $C = g_1^k \bmod 2q$. The private key, x, is calculated to satisfy the equation $Y^k x = g_1^k \bmod 2q$. The public key, y, is calculated as $y = g^x \bmod p$.

The participant then calculates part of a certificate of revocation, $v = g^{Y^{-k}} \bmod p$. Non-interactive proof of knowledge transcripts, P_1, P_2 and P_3 are computed *simultaneously*, so that:

1. P_1 proves the knowledge of k in C;
2. P_2 proves the knowledge of k in v; and,
3. P_3 proves the knowledge of k in v^C.

The participant publishes *securely* the auto-certified public key tuple to be (C, v, P_1, P_2, P_3) and (y, g, p). Note that (C, v, P_1, P_2, P_3) is the certificate for the public key, (y, g, p), of the participant.

Secure Communication: The sender obtains securely the certificate of the participant and verifies the proof transcripts (P_1, P_2, P_3) by employing the public key of the authority, $(Y, g_1, 2q)$, to check the certification of the public keys and the enablement of key recovery.

The public key tuple of the receiver, (y, g, p), is employed along with a suitable public key encryption algorithm to securely send a message to the participant. When the ElGamal encryption algorithm is employed, a message c is encrypted as $(b = cy^r \bmod p, a = g^r \bmod p)$, where r is a random value.

Key Recovery: When provided with suitable warrants and the ciphertexts, (b, a), wire-tapped during the secure communication phase, the authorities can decrypt the private key, x, corresponding to the certified public key tuple (C, v, P_1, P_2, P_3) (y, g, p) and then decrypt (b, a) to obtain c, the message communicated during the secure communication phase.

The m authorities calculate the private key, x, of the participant by computing the following algorithm in a distributed manner:

```
Set $s_0 = a$
For $i$ in 1 to $m$ and $j$ in 0 to $m - 1$ do
    $s_i = s_j^{C^{z_i}} \bmod p$
EndFor
Decrypt $c = b/(s_m^C) \bmod p$
Output $c$
```

Note that during the first iteration of the for loop, $j = 0$ and $i = 1$ therefore $s_j = s_0 = a$ and $s_i = s_1$. That is, authority i must wait for the output, s_{i-1}, from the previous authority to compute s_i and the first authority employs a to compute s_1.

The algorithm works because $Y^k = \prod_{i=1}^{m} s_i \bmod 2q$ and the private key can be decrypted as $x = CY^{-k} \bmod 2q$ by using the value C available as a part of the public key of the participant. Instead of revealing the private key, x, the algorithm performs an ElGamal decryption and reveals the message, c, which is not meant to be a long term secret.

Key Conversion Attack The participant chooses $r_1 \in_R \mathbb{Z}_q$, computes $R = g^{r_1} \bmod p$, and publishes R in its personal directory that can be accessed by the sender. The sender obtains securely the public key of the participant from the authorities as (C, v, P_1, P_2, P_3) and (y, g, p), obtains the value of R from the personal directory of the participant, computes the new public key as $y_{new} = yR^R \bmod p$, encrypts the message c as $(b = cy_{new}^r \bmod p, a = g^r \bmod p)$, and sends (b, a) to the participant. Upon receiving (b, a), the participant calculates the new private key as $x_{new} = x + r_1 R \bmod 2q$ and decrypts $c = b/(a^{x_{new}}) \bmod p$.

Since the authorities only know the discrete logarithm of y and, with a good probability, will not know the discrete logarithm of R, they cannot calculate the discrete logarithm of y_{new}, if the proxy cryptosystem of Mambo *et al.* [60, 59] is secure. Therefore, they cannot decrypt the ciphertext (b, a) encrypted using the public key y_{new}.

4.3.4 Inference

Although private key escrow presents itself as an efficient and easy mechanism for escrow enablement, the notion suffers fundamentally from many problems. Privacy of participants, lack of enforcement mechanisms to prevent attacks such as key conversion explained in Section 4.3.3, and the paradoxical state of escrow enforcement and the existence digital signature systems explained in Section 4.3.2 are the important issues.

4.4 Session Key Recovery

A session key recovery system is a short-term key recovery system. In such recovery systems certified long-term keys are employed to establish a session key which is then securely communicated to the escrow agents by employing the respective certified long-term keys. The advantage with this approach is that the user has complete control over the key recovery process. It is also inherently time-limiting, which was not the case with private key recovery systems. The disadvantage with this approach is the overhead in communications and difficulty in enforcing the compliance guarantee (because the key recovery service requires the discretion of the user). The latter observation is a security problem in many situations. A solution for this problem would be constant monitoring of user activities and their conformance to the system rules. As discussed in Section 2.3.1, enforcement of compliance can be achieved by designing an on-line or off-line monitor in order to realise enforceability Level 0 or enforceability Level 1, respectively.

Verheul and van Tilborg [89] proposed a message format for the compliance verification procedure, meant to be employed by an off-line monitor. Thereby their proposal was able to accomplish enforceability Level 1 only. As was discussed in Section 3.3.2, the proposed message format is not suitable as a integrity verification method. But, it may be possible to use the message format in a system that employs enforceability Level 0, as will be explained in Section 4.5.

Apart from Verheul and van Tilborg's proposal there has been no significant proposal meant for software implementation without any reliance on tamper resistance. Desmedt [30] presented a proposal for achieving traceability of ciphertexts in software implementations. But the viability of this proposal was seriously questioned by Knudsen and Pedersen [54], who demonstrated their concerns with practical attacks. The protocol failure in the proposals by Desmedt [30], and Verheul and van Tilborg [89] can be traced to inadequate or flawed integrity verification mechanisms which resulted in systems with very weak enforceability property. Reasonably secure integrity verification mechanisms are fundamental components for robust traceability of message formats. Most of the *practical* session key recovery proposals such as the Clipper proposal [88] and that of Gennaro *et al.* [42] (which has been incorporated into IBM's

SecureWay [50] product line) critically depend on some form of tamper resistance and strong trust relationships to achieve this goal.

Session key-recovery, by itself, is not sufficient for a robust key recovery system as it would be very difficult to enforce compliance and achieve robust traceability. This research concludes that to achieve robust key-recovery systems, a robust traceability architecture, a reasonably secure integrity verification mechanism and a strong enforcing system are essential. The next section will present an architecture that employs this philosophy.

4.5 Hybrid Key Escrow – A New Paradigm

It is evident that a key recovery system must provide traceability, enforceability and compliance checking mechanisms to preserve its properties in an open environment [40, 29]. In the Clipper proposal [88] (see Section C.2), the LEAF component of the message format traced source and destination of messages. A 16-bit checksum was intended to be the compliance checking mechanism to ensure legal formation of the LEAF component. The rule programmed into the Clipper chip that would disallow the decryption of ciphertexts with illegal LEAF components[2] provided an enforcement mechanism. The attacks reported by Blaze [10] applied to the algorithms used for implementation of the compliance checking mechanism and not necessarily to the enforcement mechanism.

An easily understandable and implementable escrow system with robust enforceability properties can be realized when on-line authorities (servers) are used in the design. Such systems can be made scalable by partitioning the network into domains with each domain having its authority – which is similar to the architecture employed by the WWW (world-wide web) service and the TCP/IP v4 communication protocol suite [36]. This would result in a better design than a monolithic system presented in the Clipper proposal.

A new paradigm for the design of key recovery is proposed in Section 4.5.2 that uses *partial* private key escrow for enforceability and session key recovery for plaintext recoverability. This distinction is essential for engendering user trust in the sys-

[2]Along with the secrecy of the SKIPJACK algorithm, which was used by the Clipper chip.

tem. The new paradigm possesses traceability and compliance checking properties. The proposal employs the the compliance checking mechanism of Verheul and van Tilborg [89].

4.5.1 The Binding ElGamal Proposal

This scheme was proposed by Verheul and van Tilborg [89]. It employs session key recovery and a multi-party extension of the ElGamal encryption scheme with a mechanism for checking the compliance of message formats. It provides an effective mechanism for compliance checking of message formats in the system. The Clipper [88] proposal possessed a compliance checking mechanism that checked for the legal formation of the LEAF component of the message format. The message was not allowed to be decrypted if the LEAF was ill-formed. This policy mechanism was programmed in the chip providing an effective way to *enforce* compliance. Such an *enforcement* mechanism is absent in the binding ElGamal proposal, which directly contributed to the attacks on the system, as explained in Section 3.3.2.

4.5.2 Hybrid Key Escrow with Strong Binding

The general idea of this scheme is to allow for the *partial* escrow of the private key of users and the recovery of entire session key. Thus, no entity will know the private key corresponding to a certified public key in the system. The public key infrastructure will consist of independent domains, with an authority controlling each domain. The private key corresponding to the certified public key is made up of two secret shares. One share is held by the user and the other by the authority of the domain in which the user is registered. The user cannot sign or decrypt messages employing the certified key pair without the participation of the authority, and vice versa. This property along with rules for formation of message formats can be used to realise robust enforceability mechanism. The scheme uses the binding ElGamal scheme of Verheul and van Tilborg [89] for checking compliance of message formats in the system.

Traceability Architecture

The traceability architecture, proposed by Boyd [11], allows a user and the authority of the system to share the secret key corresponding to a certified public key. The system settings are as follows.

- The system operates in the group $\mathbb{Z}_p^*$, where $p = 2q + 1$ and p and q are large primes.
- $g \in \mathbb{Z}_p^*$ generates the group.

All operations will take place in the congruence class modulo p, unless stated otherwise. At the end of this protocol the user's public key will be certified in the system with the user and the authority possessing shares of the secret key corresponding to the certified public key.

1. A user a chooses a secret value x_1, with $(x_1, p-1) = 1$.
2. The authority calculates $x_2 = h(K_T, Id_a)$, where Id_a is the identity number of user a in the system and K_T is the authority's master secret key.
3. The authority calculates $w = g^{x_2}$ and sends w to user a.
4. User a checks that $w^2 \neq 1$ and $w^{x_1} \neq 1$, calculates $v = g^{x_1}$ and sends v to the authority.
5. The authority certifies $y_a = g^{x_1 x_2}$ as the public key of user a, for use in the system. The certificate will be P_a.
6. The authority sends the public key and its certificate to user a.
7. User a checks that $y_a \overset{?}{=} w^{x_1}$ and the validity of the certificate. If both checks are positive, then user a accepts the public key and certificate.

The advantage of this architecture is that it is independent of the wiretapping mechanism. Wiretapping can be introduced in many ways. We propose an architecture in which the law enforcement agency is a *special user* in the system. The law enforcement agency (LEA) registers as the first user by following the user registration protocol and obtains the public key as $y_{lea} = g^{x_{lea} b_e}$, where x_{lea} is the secret key of the LEA and b_e is the authority's secret key corresponding to y_{lea}. The public key of the LEA can either be contained as a part of the user's certificate or the authority's certificate. The authority in the system is trusted to the level that it will follow the relevant procedures before assisting the LEA to wiretap communications. Since, the LEA is abstracted as a user in the system, the normal users of the system do not have to trust the LEA.

An alternative architecture could use a public key corresponding to a secret key shared between a number of escrow agents, who are in turn responsible for recovery of session keys. In this scenario the authority's responsibility with regard to compliance checking is simply to check that the session key is encrypted with the escrow agents' public key.

Function Definitions

The essential operations in the system are presented as functions to improve clarity of discussion. The inputs to the function will be placed inside the brackets that follow the name of the operation and the result set (if any) will be on the left hand side of the equation.

$\mathbf{R} = \mathrm{Enc}_{\{\mathbf{y_a},\mathbf{y_{lea}}\}}\ (\mathbf{S},\mathbf{k})$

```
        Compute $R_a = Sy_a^k$
        Compute $R_{lea} = Sy_{lea}^k$
        Compute $C = g^k$
        Assign $R \leftarrow \{R_a, R_{lea}, C\}$
```

This function takes a message S and performs a multi-ElGamal encryption using the public keys y_a and y_{lea}. The outputs of the function are exponentiation C of the base g to the random value k and the ciphertexts R_a and R_{lea} corresponding to the respective public keys.

$\mathbf{B} = \mathrm{formBind}(\mathbf{E},\mathbf{R},\mathbf{y_a},\mathbf{y_{lea}},\mathbf{k})$

```
        Choose $j \in_R \mathbb{Z}_p$
        Compute $D = g^j$
        Compute $F = (y_{lea}/y_a)^j$
        Compute $v = \mathcal{H}(F, D, C, R_a, R_{lea}, E)$
        Compute $z = vk + j \bmod q$
        Assign $B \leftarrow \{E, R, D, F, z\}$
```

This function forms the binding parameters for the ciphertext available in its input. $E = \text{enc}(S, M)$ is the symmetric encryption of the message M with the session key S. The output of this function will contain a non-interactive proof (F, D, z), that can be used to check if the same message S is encrypted for the receiver and LEA using their respective public keys and random number k, which is the discrete logarithm of C.

```
checkBind($\mathbf{B}, \mathbf{y_a}, \mathbf{y_{lea}}$)
    Compute $v = \mathcal{H}(F, D, C, R_a, R_{lea}, E)$
    Check $g^z \overset{?}{=} C^v D$
    Check $(y_{lea}/y_a) \overset{?}{=} (R_{lea}/R_a)^v F$
```

This function checks the non-interactive proof generated by formBind(...) using the public keys of the relevant parties.

```
$\mathbf{T} = \text{PartDec}_{\mathbf{x_1}}(\mathbf{B})$
    Compute $C' = C^{x_1}$
    Assign $T \leftarrow \{E, R_a, C'\}$
```

This function performs a partial decryption of the ElGamal ciphertext using one of the secret shares corresponding to the public key with which the encryption was performed. The output of this function will allow the owner of the other secret share to decrypt the message.

```
$\mathbf{S} = \text{Dec}_{\mathbf{x_i}}(\mathbf{T})$
    $S = R_a/C'^{x_i}$
```

This function performs ElGamal decryption. Its output is the message S.

The Scheme

When a sender needs to send a confidential message to user A, he/she encrypts the message M using a well chosen session key S by employing a standard symmetric key encryption algorithm to obtain E. The sender encrypts the session key under the public keys of the receiver and the LEA to obtain R_a and R_{lea} respectively. (Here we assume that only one LEA is involved, but two or more could be accommodated.) The sender then binds E with C, R_a and R_{lea} to obtain the binding parameters. Id_a, E, C, R_a, R_{lea} and the binding parameters are sent to the authority. The steps followed by the sender can be summarised as:

- Choose S
- Encrypt message: $E = enc(S, M)$
- Choose $k \in_R \mathbb{Z}_p$
- Encrypt session key: $R = Enc_{\{y_a, y_{lea}\}}(S, k)$
- Form binding parameters:
  $B = \text{formBind}(E, R, y_a, y_{lea}, k)$
- Send to Authority: $\{B, Id_a\}$

The authority checks for the legal formation of the binding parameters and, if they are correct, alters C using the secret share x_2 corresponding to the user's public key to obtain C' so that A can decrypt the session key using her secret key. Note that the correct value $x_2 = h(K_T, Id_a)$ can only be obtained by the authority if the correct identity Id_a is sent with the message. The steps followed by the authority can be summarised as:

- Check binding parameters:
  $\text{checkBind}(B, y_a, y_{lea})$
- Compute $x_2 = h(K_T, Id_a)$
- Partial decryption: $T = PartDec_{x_2}(B)$
- Send to Receiver A: $\{T\}$

On receipt of the partially decrypted message, A performs a standard ElGamal decryption to obtain the session key S and decrypts the ciphertext E using the session key to retrieve the message M. The steps followed by A are:

- Decrypt session key: $S = Dec_{x_1}(T)$
- Decrypt message: $M = dec(S, E)$

It is evident that the authority cannot obtain the message M with any of the information it possesses. The message can be decrypted only by the receiver, and law enforcement agency in the domains of the sender and receiver. To wiretap any communication the LEA has to;

1. approach the authority of their domain,
2. produce a court order for wiretapping the communications to a user,
3. request the authority to alter the message component C using the authority's secret key corresponding to the LEA's public key to get C_{lea},
4. perform standard ElGamal decryption of R_{lea} using C_{lea} and LEA's secret key to obtain the session key S, and,
5. perform symmetric key decryption on E using S to retrieve M.

Compliance Guarantee Specification

As noted in Section 2.3, the compliance specification for the scheme presented in the previous section can be presented as follows:

Message Format: $\{E, R_a, R_{lea}, C, D, F, z\}$ such that the tuple is a verifiable encryption of the form:

$$\langle PROOFEQ(Dec_a(R_a, C) = Dec_{lea}(R_{lea}, C)\rangle,\ R_a,\ R_{lea},\ C$$

This type was classified as Class 1, in Section 2.2.1.

Compliance Category 1: Restricted confidentiality and universal integrity for the message encrypted in the ciphertext (R_a, C) is the goal of this message format.

Enforceability Level 1: An on-line authority was essential for the communication between users; the security analysis in the next section will provide analysis.

The Clipper proposal [88] has a similar specification:

Message Format: The LEAF (law enforcement access field) component contained the ciphertexts of the session key and the proof transcripts for integrity verification. The proof transcript was not globally verifiable as the knowledge of the session key was essential for the verification procedure.

Compliance Category 1: Restricted confidentiality and universal integrity for the session key is the goal of the message format.

Enforceability Level 1: Due to the secrecy of the algorithm, it was assumed that only Clipper systems can access message formats formed by other Clipper systems. Due to the tamper-resistant nature of the systems and the secrecy of the algorithm, the *use* of the system during every communication was essential. That is the Clipper system had to be on-line during every communication phase involving another Clipper system.

Security

The scheme achieves the properties essential for robust key recovery systems. First note that the key conversion attack is not possible. This is the case since no user knows the complete information of the private key corresponding to their public key, so any proxy key that relies on knowledge of the discrete logarithm of the public key cannot be obtained by any user. The attack by Pfitzmann and Waidner [73] and the attack presented in Section 3.3.2 on the binding ElGamal [89] proposal will not affect the proposal presented in Section 4.5.2. If a sender transmits a ciphertext that was not encrypted with the session key S, which was encrypted under the certified public keys of the user and the LEA, then the recipient (see Section 3.3.2) will not be able to decrypt it since he/she will not know the actual session key. The individual security requirements can be determined as follows:

Confidentiality If a sender uses the prescribed protocol then the properties of multi-ElGamal, proven by Verheul and van Tilborg [89, Theorem 2.3] guarantee the plaintext is only available to the chosen recipient and the LEA.

Escrow enforcement If the authority checks that the binding is correct then any message encrypted with any user's certified public key must be potentially decryptable by the LEA. This follows from the properties of binding ElGamal. It is proven by Verheul and van Tilborg [89] that the protocol ensures, under reasonable assumptions, that both parts of the multi-ElGamal encryption contain the same plaintext. The *important extra point* is that only those ciphertexts for which a corresponding binding exist will be partially decrypted by the authority. Due to this property the attack detailed in Section 3.3.2 will not be successful. This is because the authority, who is trusted in this regard, will not assist the hidden receiver (see Section 3.3.2) to decrypt the message format.

Compliance If any encryption algorithm is used which requires the discrete logarithm of any public key to be known for decryption then the ciphertext must pass through the authority for decryption to take place. This follows immediately because no user knows the complete private key corresponding to any public key.

Note that, in this architecture, although the users can perform a variant of the Diffie-Hellman key exchange protocol employing their certified public keys to avoid escrow, the central authority has a guarantee for accountability from these users. This was not possible in any of the previous key escrow proposals. The property of compliance that we can prove is not as strong as we might like. Ideally we would like to show that any encryption algorithm that uses a certified public key (and no other secret information) in any way whatsoever must be subject to escrow enforcement. This seems too difficult to achieve and so we cannot claim unconditionally that our architecture enforces compliance even though it avoids all the known attacks on other schemes.

4.5.3 Source Traceability

The message format (LEAF component) of the Clipper proposal [88] had the ability to robustly trace the source and the destination based on various assumptions. The binding ElGamal proposal by Verheul and van Tilborg [89], which the proposal for hybrid key-recovery system in Section 4.5.2 employed, cannot robustly trace the source of the message format. In order to provide this capability to the hybrid key-recovery system, the sender of the message can be expected to sign some parts of the message format that can be universally verified. In situations where privacy is essential, forms of designated verifier proof or group signature system can be employed. For reasons stated in Section 4.2, the emphasis in this chapter will be on publicly verifiable message formats.

For this purpose, if an unescrowed signing key is issued for all participants in the system then the hybrid system will also be susceptible to the deficiencies stated in Section 4.3.2. Thereby, a signature system for the hybrid key-recovery paradigm is essential. The design for a suitable signature algorithm for this purpose will result in the following message transactions and procedures:

1. the private key corresponding to the certified public key of the sender will be shared by the sender and its authority as suggested in Section 4.5.2;

2. the sender can form valid signature tuples, which can be universally verified, only by interacting with its authority;

3. the sender's authority will assist the sender to form valid signature tuples only after it verifies that the binding ElGamal data is correctly verified;

4. the receiver's authority will assist the receiver to decrypt message formats only if it correctly verifies the accompanying signature tuples and the binding ElGamal data.

When there exists some implicit trust relationship between the sender's authority and the receiver's authority, the receiver's authority may just rely on the signature generated by the sender and the sender's authority, without verifying the binding ElGamal data. Many other alternatives are possible depending on how message formats are verified by the authorities and the trust relationship they share.

This section will propose a joint signature scheme that uses the traceability architecture proposed by Boyd [11], the Schnorr signature scheme in designate verifier mode [81] and the signature scheme by Horster, Michels and Petersen [49] (HMP) for the generation of digital signatures. Joint signature scheme is essential for source traceability in the hybrid key recovery system due to the reasons stated in Section 4.3.2.

System Entities

The joint signature system consists of a group of signers, labelled as $1, \cdots, l$ for an integer l. This analysis concentrates on the case when $l = 2$. This is because of the design assumption for a single authority corresponding to a particular user: that is the user and authority perform the joint signature procedure. The joint signature scheme can be easily extended to include more than two signers, but the increase would also result in a change in the verification equation and an increase in the size of the signature tuple. The group of l signers will be labelled as $S = \{s_1, \cdots, s_l\}$. Any global verifier can know the identity of S from the public key certificate and cannot obtain any information about the individual signers, if the information is not certified. A certificate authority, CA, maintains the public key certificate repository. The system settings and description is essentially the same as the hybrid key recovery systems explained in Section 4.5.2.

Signature Generation and Verification Process

The signature process is distributed and consists of the following algorithms:

1. partSign with inputs (m, x_i, w, q) and outputs (c, d): generates a partial signature tuple (c, d) on the message m using the secret key share x_i. It employs the Schnorr signature generation algorithm in the designated verifier mode.

2. partCheck with inputs (m, c, d, x_j, y, g, p) and outputs (1 **or** 0): checks the correct formation of the tuples (c, d) on the message m using the other secret share $(x_i \mid i \neq j)$ and, outputs 1 for success or 0 for failure. Note that the 3-tuple (m, c, d) can be checked only with the knowledge of $(x_j \mid i \neq j)$. It employs the Schnorr signature verification algorithm in the designated verifier mode.

3. completeSign with inputs (d, x_j, g, q) and outputs (d', r): completes the partial signature so that the 4-tuple (m, c, d', r) can be successfully checked by a universal verifier, if the 3-tuple (m, c, d) was successfully checked using the function desigCheck. It uses the HMP signature generation algorithm.

4. completeCheck with inputs (m, c, d', r, y, g, p) and outputs (1 **or** 0): checks if the 3-tuple (c, d', r) is a valid signature on m by the certificate holders of the public key y. It uses a combination of the Schnorr (in designated verifier mode) and HMP signature verification algorithms.

Figure 4.1 shows a graphical representation of the message dynamics in the joint signature system. The protocol and algorithms are shown in Table 4.1. The signature scheme is valid because:

Signer 1 (Sender)	**Signer 2 (Sender Authority)**	**Verifier**
partSign:		
$k_1 \in_R \mathcal{Z}_q^*$		
$c = \mathcal{H}(m\|\|w^{k_1})$		
$d = k_1 - cx_1 \bmod q$		
$\xrightarrow{c,d,m}$		
	partCheck:	
	$c \overset{?}{=} \mathcal{H}(m\|\|y^c g^{x_2 d})$	
	completeSign:	
	$k_2 \in_R \mathcal{Z}_q^*$	
	$r = g^{k_2}$	
	$d' = dx_2 - k_2 r \bmod q$	
	$\xrightarrow{c,d',m,r}$	
		completeCheck:
		$c \overset{?}{=} \mathcal{H}(m\|\|y^c g^{d'} r^r)$

Table 4.1: Joint signature scheme

$$\begin{aligned} w^{k_1} &= g^{x_2 k_1} && (4.1)\\ g^{x_2 k_1} &= g^{x_2(cx_1+d)} && (4.2)\\ g^{x_2 x_1 c + x_2 d} &= y^c g^{x_2 d} \end{aligned}$$

Thus, $w^{k_1} = y^c g^{x_2 d}$ is used in partCheck. Now,

$$\begin{aligned} y^c g^{x_2 d} &= y^c g^{d'+k_2 r} && (4.3)\\ y^c g^{d'+k_2 r} &= y^c g^{d'} r^r \end{aligned}$$

Thus, $w^{k_1} = y^c g^{d'} r^r$ is used in the function completeCheck.

Security analysis

Formal security analysis for signature schemes has been very difficult, which is the reason that many signature systems believed to be secure do not have a concrete proof

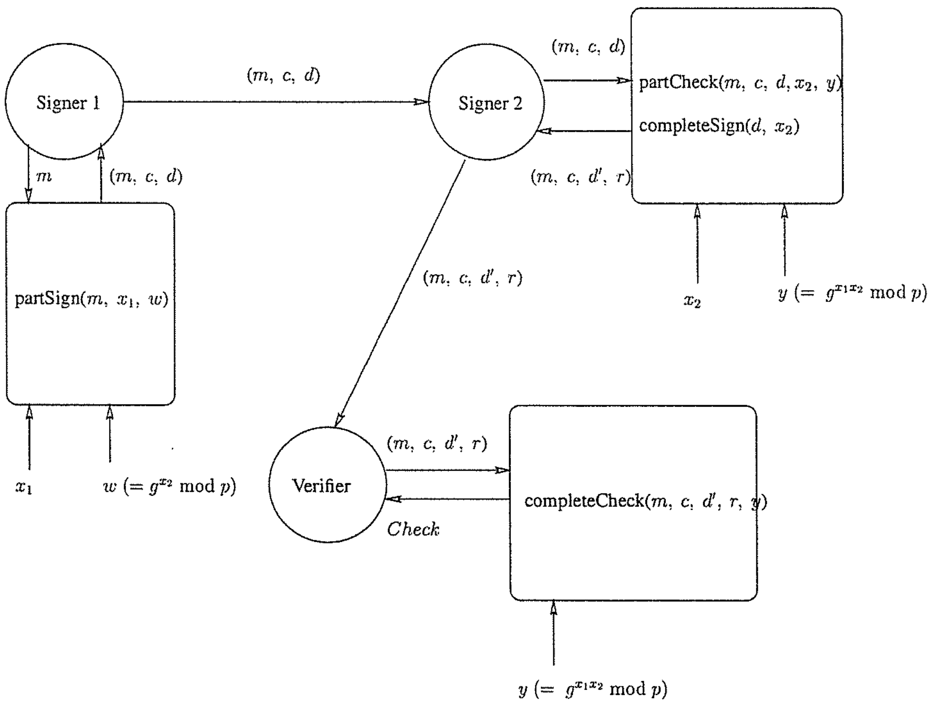

Figure 4.1: Message dynamics in the joint signature system

of security but only in specialised models of security [74] working under strict assumptions [5]. We would ideally like the proof of security to prove the equivalence of "breaking" a signature scheme to solving a known hard problem. We present security arguments of our scheme that facilitate a better understanding of the system, so that attempts to prove its security could be efficient and successful. This section will discuss the desirable properties of joint signature schemes in general followed by an analysis on a model of the joint signature scheme. In order to improve clarity, the analysis will assume only two signers sharing a public key.

Security properties The desirable security properties of the joint signature scheme (see figure 4.1) are stated in this section.

Security Property 4.1 *Only signer 1 can compute the function partSign to obtain valid signature tuples that can be verified by signer 2.*

Security Property 4.2 *Only signer 2 can compute the function partCheck to check the validity of partial signatures when signer 1 computes the function partSign.*

Security Property 4.3 *Function completeCheck can be universally checked and outputs 1 when valid outputs of the completeSign are given.*

Security Property 4.4 *Signer 2 cannot form valid universally verifiable signatures using the function completeSign without the outputs of the function partSign as computed by signer 1.*

Security Property 4.5 *Only signer 2 can compute the function completeSign to obtain valid joint signature tuples when the outputs of the function partSign as calculated by signer 1 are given.*

Theorem 4.5 *The proposal for the joint signature system satisfies:*

1. *security property 4.1, if the Schnorr signature is not forgeable,*
2. *security properties 4.2, if the decisional Diffie-Hellman problem is intractable, and,*
3. *security property 4.3.*

Proof:

1. The algorithm for the function partSign is a direct application of the Schnorr signature scheme with the base for generating the challenge as $w = g^{x_2}$ (for signer 1) instead of g, as is the case in the Schnorr signature scheme.
2. Following the previous arguments, to check the validity of the Schnorr signature triplets the verifier should possess the knowledge of the discrete logarithm $\log_g w \bmod p$. This is because the values x_1 and x_2 are chosen at random and no universal verifier can correlate the relation between w and y due to the decisional Diffie-Hellman problem [26].
3. The algorithm for the function completeCheck does not require any private data as its input. Correctness follows from equation 4.3. □

It is interesting to note that in the case of standard Schnorr signature $w = g \bmod p$, so that everyone knows a solution for the discrete logarithm $\log_g w \bmod p$.

Security property 4.4 states that universally verifiable signatures can be formed only when both the signers co-operate. Informal analysis suggests that the proposal satisfies this property. We conjecture that the proposal relies on the security of the Schnorr signature scheme against existential forgery to achieve this property.

Security property 4.4 aims at providing security for a signer when the other signer is the attacker. Security property 4.5 provides a stronger notion of security, in that it provides security against an external attacker. We conjecture that the proposal satisfies this property when the Schnorr signature scheme is secure against existential forgery.

The next section will discuss an analysis of a model of our signature system.

Analysis of a model The modes of attack on any signature scheme [83] are:

- **Total Break:** Given a set of well formed signatures, it will be possible to compute the secret key.
- **Universal Forgery:** Finding an algorithm that will form a valid signature on a message without the knowledge of the secret key corresponding to the public key.
- **Selective Forgery:** Forge a signature for a particular chosen message, with or without interacting with the original signer.
- **Existential Forgery:** Forge at least one message without the knowledge of the private key. Note that the message value could be random without any meaning [83].

One approach for proving the security of a signature scheme could be to show that existential forgery is not possible which, being the weakest form of attack, will provide the strongest notion of security. Unfortunately it seems difficult to achieve this proof. This section provides a security analysis of the proposal against a special form of existential forgery on the Schnorr signature system and its derivatives called $(m,\ c)$-forgery (see Definitions 4.2 and 4.5.3).

In the rest of this security analysis, all arithmetic operations are assumed to be performed in the group of integers, $\mathbb{Z}_p$, unless stated otherwise. The Schnorr signature verification process can be defined as:

Definition 4.1 *The tuple $(c,\ d)$ is said to be a valid Schnorr signature on a message m by the entity owning the public key y, if the following equation is verified:*

$$c \overset{?}{=} \mathcal{H}(m\|y^c g^d) \tag{4.4}$$

Let S denote the set of algorithms that can be used to implement $(m,\ c)$-forgery attacks on the Schnorr signature scheme. Then the attack algorithms in set S can be defined as:

Definition 4.2 *The algorithms in set S can be modeled using a probabilistic polynomial time Turing machine that on input d will return, with a high probability, m and c such that the tuple $(c,\ d)$ is a valid Schnorr signature on m, verifiable using the public key y.*

The verification process for the joint signature scheme can be defined as:

Definition 4.3 *The tuple $(c,\ d',\ r)$ is said to be a valid joint Schnorr signature on a message m by the entity owning the public key y, if the following equation is verified:*

$$c \overset{?}{=} \mathcal{H}(m\|y^c g^{d'} r^r) \tag{4.5}$$

Let J denote the set of algorithms that can be used to implement $(m,\ c)$-forgery attacks on the joint signature scheme. Then the attack algorithms in set J can be defined as:

Definition 4.4 *The algorithms in set J can be modeled using a probabilistic polynomial time Turing machine that on inputting $(d',\ r)$ will return, with a high probability, m and c such that the tuple $(c,\ d',\ r)$ is a valid Schnorr signature on m, verifiable using the public key y.*

Note that algorithms in the sets S and J appear to be related. This is because a common strategy to implement the algorithms in sets S and J could be to attack the algorithm used for the hash function. That is given the partial sequence of the input, d or $(d',\ r)$, find the remaining input, m, and the corresponding output, c.

The $(m,\ c)$ attack concentrates on the structure of the universal verification equation, which is an essential part of any signature scheme, and tries to form valid signature n-tuples on (random) messages that could be verified using the public key *without* the knowledge of the corresponding private key. Note that this attack is stronger than existential forgery, thereby proof of security against $(m,\ c)$-attack provides a weaker notion of security than a proof of security against existential forgery. So, if there exists an algorithm to perform existential forgery on a signature scheme, an algorithm to implement the $(m,\ c)$-attack may also exist, but not necessarily the other way around.

The security analysis of the HMP signature scheme assumes the existance of an efficient algorithm to perform existential forgery on the ElGamal signature system [20] and its variants, as suggested by the following lemma.

Lemma 4.1 *The HMP signature system is existentially forgeable.*

Proof: The HMP signature tuple (r, s) on a message m and public key y must satisfy the equation $g^s = y^m r^r \bmod p$, where the private key $x = \log_g y$. To arrange this an attacker:

1. Chooses arbitrary integers b and c.

2. Computes $r = g^b y^c \bmod p$, $s = br \pmod{p-1}$ and $m = -cr \pmod{p-1}$.

The tuple (r, s) will then be a valid signature on the message m verifiable using the public key y. $\square$

The joint signature system can be viewed as a cascade of the Schnorr and HMP signature schemes as shown in Figure 4.2. Signer 1 performs the Schnorr signature and Signer 2 performs the HMP signature, as suggested in the protocol depicted in Table 4.1. Thus, the signature tuple corresponding to the message, m, and the public key, $g^{x_1 x_2}$, is $(d',\ r,\ e)$.

Theorem 4.6 *If the Schnorr signature scheme can be $(m,\ c)$-forged employing algorithms from the set S, then the joint signature scheme can be $(m,\ c)$-forged employing algorithms from the set J.*

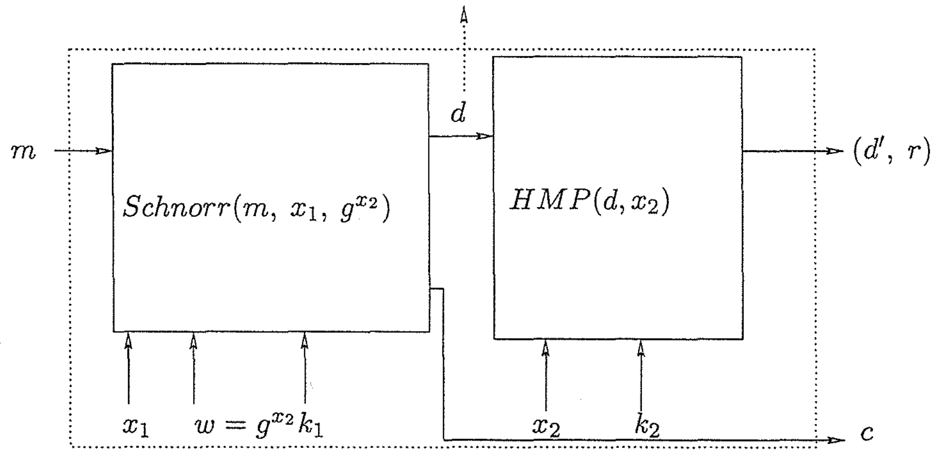

Figure 4.2: A Visualisation of the Joint Signature Scheme

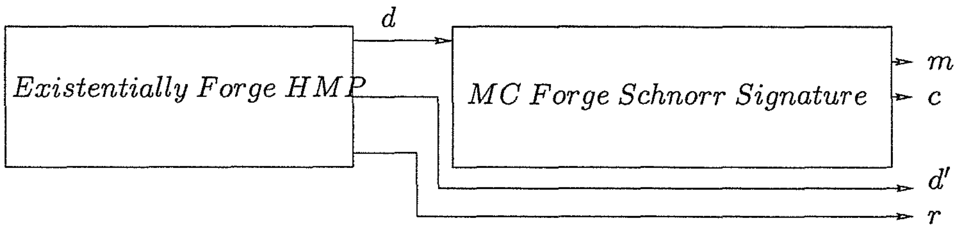

Figure 4.3: Attack Scenario for the Joint Signature Scheme

Proof: Figure 4.3 illustrates the approach for this proof. Suppose that signer 2 is mounting an attack on signer 1. Let $w = g^{x_1}$, so that $\log_{g^{x_2}} y = \log_g w$. Assume that the Schnorr signature scheme is $(m,\ c)$-forgeable. Since the HMP signature scheme is existentially forgeable, the attacker can obtain a valid signature tuple $(d',\ r)$ on some message value d that can be verified using w as the public key. Since the Schnorr signature is assumed to be $(m,\ c)$-forgeable, the tuple $(m,\ c)$ can be formed when given d, so that the 3-tuple $(m,\ c,\ d)$ satisfies the equation $c \overset{?}{=} \mathcal{H}(m||y^c w^d)$. But the HMP signature on d is $(d',\ r)$ so that $w^d = g^{d'} r^r$. Thus $c \overset{?}{=} \mathcal{H}(m||y^c g^{d'} r^r)$, which is the signature verification scheme for the joint signature scheme. Thus the 4-tuple $(m,\ c,\ d',\ r)$ is a $(m,\ c)$-forged signature on the joint signature scheme. □

Theorem 4.7 *If the joint signature scheme can be $(m,\ c)$-forged employing algorithms from the set J, then the Schnorr signature scheme can be $(m,\ c)$-forged employing algorithms from the set S.*

Proof: Figure 4.4 illustrates the approach for this proof:

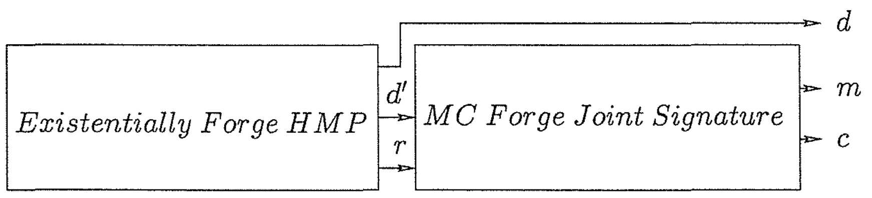

Figure 4.4: Attack Scenario for the Schnorr Signature Scheme

Suppose that signer 2 is mounting an attack on signer 1. Let $w = g^{x_1}$, so that $\log_{g^{x_2}} y = \log_g w$. Assume that joint signature scheme is $(m,\ c)$-forgeable. There exists an existential forgery on the HMP signature scheme. Find existentially forged signatures $(d',\ r)$ on d such that $w^d = g^{d'} r^r$. Since joint signature scheme is assumed to be $(m,\ c)$-forgeable, find $(m,\ c)$ so that $(m,\ c,\ d',\ r)$ is a valid joint signature satisfying the equation $c \overset{?}{=} \mathcal{H}(m||y^c g^{d'} r^r)$. Thus $c \overset{?}{=} \mathcal{H}(m||y^c w^d)$, which is a $(m,\ c)$-forged Schnorr signature tuple $(d,\ c)$ on m. □

Corollary 4.1 *Joint signature scheme is $(m,\ c)$-forgeable if and only if Schnorr signature scheme is $(m,\ c)$-forgeable.*

Proof: The proof follows from the proofs for Theorems 4.6 and 4.7. □

Lemma 4.2 *If signer 2 can form signature tuples without the help from signer 1, then so can a universal attacker without the help of signer 1 and signer 2.*

Proof: The public key is of the form $y = g^{x_1 x_2} = g^x$. Therefore, $x'_2 = 1$ is a valid share because $y = g^x = g^{x x'_2}$.

By symmetry of arguments, if signer 2 can form valid signature tuples that can be verified employing the public key y with the knowledge of x_2 and without the knowledge of x_1, then a universal attacker can form valid signature tuples that can be verified employing the public key y with the knowledge of x'_2 and without the knowledge of x.

□

The above lemma is important because it suggests that the signers have no additional advantage compared with an universal attacker to forge joint signature tuples without the assistance from its peer. This lemma provides additional understanding about the proposals achievement of security Properties 4.1, 4.4, and 4.5.

Maurer and Massey [58] presented a folk theorem that suggested that cascaded ciphers will be at least as difficult to break as its component ciphers. Similarly, the following observation on the joint signature scheme, which is a cascaded signature system, can be made as:

Observation 4.1 *A joint signature scheme's security against* $(m,\ c)$*-forgery will not be any more secure than the most secure (against* $(m,\ c)$*-forgery) signature scheme in the cascade.*

Evidence for Observation 4.1 can be found in Theorems 4.6 and 4.7, and in Corollary 4.1. The joint signature scheme was a cascade of the HMP signature scheme and the Schnorr signature scheme. Lemma 4.1 indicates that the HMP signature scheme is existentially forgeable. The $(m,\ c)$-forgery is a stronger attack than existential forgery. The security for the joint signature scheme primarily relies on the the security of the Schnorr signature scheme against $(m,\ c)$-forgery.

4.6 Summary

Three forms of key recovery techniques, namely private key, session key and hybrid key recovery schemes were discussed. The inherent problem with the private key and session key recovery systems for software implementations were discussed. A new key recovery paradigm called hybrid key recovery was presented. It was shown that hybrid key recovery achieves, relying only on the certification procedure and an online authority, every security aspect that the Clipper proposal achieved by relying on tamper-resistant hardware and secrecy of the confidentiality algorithm. The hybrid key recovery proposal seems to be the only practical, open, certification-based, robust (as is possible), software key recovery proposal currently available in the open literature. The following comparison between the Clipper proposal [88] and the scheme proposed in Section 4.5.2 can be made:

Compliance: The LEAF component, which was essential for the verification of compliance, is replaced by the publicly verifiable message format proposed by Verheul and van Tilborg [89];

Enforceability: the reliance on tamper resistance and secrecy of algorithms is replaced by trusting the on-line authority and by the traceability architecture proposed by Boyd [11].

A key-recovery system is perfect if it can guarantee a proposition of the following form: secure communications to/from a user is possible if and only if the escrow authority (and the law enforcement agency) can have access to the confidential message being communicated. Perfect key recovery is an unsolved issue because any two conspiring rogue users can employ a secure key agreement protocol to effectively by-pass key recovery. In fact, the findings of this research suggests the impossibility of perfect key recovery systems. This result may be traced back to the modeling of cryptosystems by Shannon[84] used widely in the design of cryptologic systems, which assumes an *insecure* physical communication channel and a secure cryptographic algorithm as the only requirements for secure logical communication channel. The original work on public-key cryptosystems by Diffie and Hellman [33] adopted this strategy. Most key recovery systems tend to provide an authenticated channel that is unique for a pair of users, otherwise unique key recovery will not be possible – conceptually, the participants can utilise the lack of the uniqueness property to by-pass key recovery. On the other hand, the provision of such a channel along with the modeling of cryptosystems, provides a clear advantage for the conspiring users of the system in by-passing key recovery.

The proposal presented in Section 4.5 provides an auditing tool that can be employed to hold the conspiring users accountable in the case of an illegal usage. This seems to be the best possible solution for key recovery systems. The requirement for key-recovery can be summarised as follows:

Service Phase: The escrow agency provides the users with a service, which the users cannot achieve *without the assistance* of the agency. The Clipper proposal intended to provide an *infrastructure* for secure communications based on a *secret algorithm*. In the hybrid key-recovery proposal, the agency provides a *robust* *certification* infrastructure. Thereby, this phase symbolises a transfer of service from a powerful agency to normal users;

Compliance Phase: The users, in response to the service, grant some *privileges* to the agency. This phase provides “plain-text access” to the agency.

The literature review of this research found that almost all commercial, public-domain proposals (excluding the Clipper proposal) did not achieve robustness in the second phase. The proposed hybrid-key recovery systems seems to be the only mechanism available for certification-based, software key-recovery system that encompasses all the properties evident in the Clipper proposal, in a much better fashion.

Chapter 5

Anonymous Token Systems

Dazzling achievements are possible, which can make a man's name live for thousands of years. But above this level, far above, separated by an abyss, is the level where the highest things are achieved. These things are essentially anonymous.

- SIMONE WEIL

(1909 - 1943) French philosopher

"La Table Ronde," "Human Personality," 1950.

The confidentiality service is provided to data by employing a key, which results in a ciphertext. An interesting scenario occurs when the data is a representation of the identity of participants in the system. The result of providing the confidentiality service to the identity of participants is the anonymity service. A collection of ciphertexts that are essential for the provision of the anonymity service is called a token. Two methods for achieving the anonymity service are:

1. the token is a function of a random string, so that there exists no relationship between the token and the identity of the participants; and,

2. the token is a function of a random string and the identity of the participant, such that the relationship between the identity and the token is confidential and is known only to the participant, and optionally to a trustee.

The second approach is more comprehensive and can be employed to model the first approach. Such a model, for example, may provide all the participants in the system with the same identity. A class of systems, which provides confidentiality service to an identity of a participant, called the *anonymous token systems* (ATS) is the interest of this chapter. The word token denotes the security objects (ciphertext or ciphertexts) that provides the confidentiality service to the identity. The token will belong to compliance Category 1, as discussed in Section 2.3, if the ATS accommodates mechanisms for tracing[1]. It will belong to compliance Category 0, as discussed in Section 2.3, if the ATS does not provide mechanisms for tracing. The ATS employed in Sections 5.3 and 5.4 belong to the former category and the ATS which could be used in Section 5.5 would belong to the latter category.

ATS requires compliant cryptologic protocols because of the following potentially conflicting requirements for:

1. the authorities, who require every anonymous participant to be authorised and the authorisation procedure will require a suitable form of authentication; and,

2. the anonymous participants, who require the anonymity service.

Additionally, the authorities may require the revocation of anonymity service, which would contradict fundamentally the requirements of the participants. The only known approach to solve such fundamental conflicts requires the authorities and the participants to trust a set of revocation authorities, who can perform the revocation service. Such systems require potentially additional compliance tests because every participant must prove the ability of the revocation authorities to perform the revocation. Although the above discussion may provide a picture where compliance testing and anonymity service are at the opposite ends of a spectrum, it is possible to achieve both the requirements with suitable assumptions. Such systems provide compliance verification equations for environments with the anonymity service.

Electronic cash (e-cash) provides anonymity for the sink (or receiver) of the token (signature ciphertexts). The blind signature technology has been the only *known* efficient technique for the design of anonymous token systems (e-cash systems). Group signature schemes, on the other hand, provide anonymity for the source (or signer or sender) of the token. The signer is anonymous and the receiver of the signature does not obtain anonymity service. The primary interest of this chapter is to employ techniques that provide the anonymity service for the sink of the token, such as the electronic cash technology, to design an ATS.

The ATS can be treated as a black-box that provides the integrity service to a relationship between an identity and the data, and the confidentiality service to the relationship. Secure selection protocols (SSP) allow robust linkage of tuples of the form

[1]When tracing of the identity is required, the anonymity service must be restricted. Thereby, only restricted confidentiality service can be provided for the identity of participants.

(I, D), where I is an identity and D is a data and provide confidentiality to the tuples. The word *selection* suggests that the data D may be the choice (selection) of an entity with identity, I, and the word *secure* implies the provision of the confidentiality and the integrity services for this selection.

Three possible approaches for providing the confidentiality service to the relationship would be to provide the confidentiality service for I, for D, or for both I and D. This chapter is interested in the first or third approach, namely confidentiality for the identity I, depending on the application. The peer-review protocol presented in Section 5.3 adopts the first approach and the proposals for electronic auction and electronic voting systems adopt the third approach. These approaches result in the design of compliance verification equations that can operate in an anonymous environment.

A concrete proposal to achieve anonymity in electronic systems was first proposed by Chaum using blind signatures [19, 28]. Since then, research for the provision of the anonymity service, especially for e-cash systems, has been extensive [70, 92, 12, 55]. Like security systems, the effective anonymity provided by such systems critically depends on the *weakest* link in the communication infrastructure – in this scenario, the word *weakest* refers to the ease of tracing transactions. If a layered anonymity system is assumed, comprising of a logical anonymity channel operating over a physical anonymity channel, then the effective anonymity would be the weakest anonymity service provided by one of the two layers. That is, if a perfectly anonymous logical channel is layered over a physical channel providing weak anonymity service, then the total system will provide a weak anonymity service. Both the physical and logical layers must provide sufficient anonymity service, in order to achieve the required level of effective anonymity. There has been significant advancement in the research for the provision of the anonymity service in physical channels by employing mix-networks [51, 2].

The subsequent discussions will assume the presence of an anonymous physical channel and concentrate on the dynamics of the anonymous logical channels. This separation of concerns yields an efficient analysis and design approach.

This chapter will present an analysis of anonymous token systems, the electronic cash technology and then employ the concepts to propose a generic schema for the design of secure selection protocols. The schema will then be employed to design a peer-review proposal and an auction system. A conceptual design for a basic electronic voting system employing the schema will also be presented.

5.1 Overview of Anonymous Token Systems

The aim of anonymous systems is to hide the identities of *registered users*. Thus, anonymity can be modeled as the confidentiality service for an identity. Encryption algorithms provide the confidentiality service to a message by employing a key. The confidentiality service is guaranteed as long as the encrypted message is suitably protected and not transmitted over insecure channels. On a similar note, anonymity systems provide confidentiality for an identity as long as the identity is suitably protected. Cryptologic services assist in the process for the maintenance of confidentiality and do not assist in the creation process, which is an issue external to cryptography.

An anonymous token system (ATS) is a suite of protocols that can be used for anonymous transfer of credentials – that is the identity of the source of the credential is hidden from the destination and all other parties. The protocols in the suite are token issuing, token utilisation, token submission and tracing. The tracing protocol is relevant only when restricted anonymity is a requirement.

The ATS is a specialised authentication system, generically represented as in Figure 5.1. In the figure, the interactions are represented by the lines connecting the

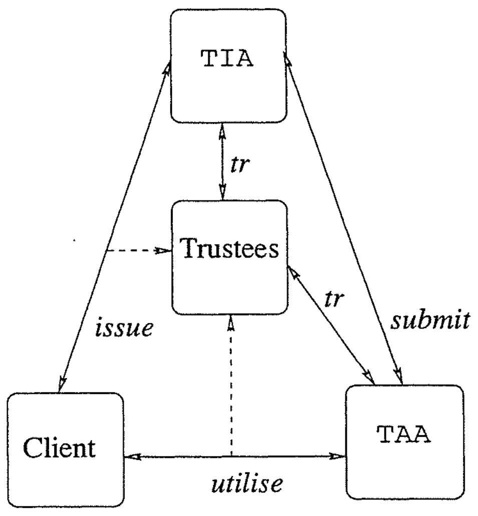

Figure 5.1: Basic Anonymous Token System

respective entities as follows:

1. Token issuing protocol performed by the token issuing authority (`TIA`) and the client. This interaction does not provide immediate anonymity service. Let $\mathcal{I}$ be a set of tuples that contain all the legal tuples that describe possible conversations of this protocol. Every instance of a legal conversation could then be represented by the tuple *issue* $\in \mathcal{I}$, as shown in Figure 5.1. When the blind signature technique is employed `TIA` assumes the role of a signer and the client the role of a *honest* verifier;

2. Token utilisation protocol performed by the client and the token accepting authority (`TAA`). The client can remain anonymous during this interaction. Let $\mathcal{U}$ be a set of tuples that contains all the legal tuples that describe possible conversations of this protocol. Every instance of a legal conversation could then be represented by the tuple *utilise* $\in \mathcal{U}$, as shown in Figure 5.1. This protocol allows the client to submit the anonymous token, obtained as a result of the token issuing protocol, to the `TAA` without identifying itself, in return for a specified security service;

3. Token submission protocol between the `TAA` and the `TIA` allows the `TAA` to submit the tokens it has accepted during the token utilisation protocol. Let $\mathcal{S}$ be a set of tuples that contains all the legal tuples that describe possible conversations of this protocol. Every instance of a legal conversation could then be represented by the tuple *submit* $\in \mathcal{S}$, as shown in Figure 5.1. In the case of electronic cash systems, there exists a bijection between the set of conversations for the submission protocol, $\mathcal{S}$, and the set of conversations for the utilisation protocol, $\mathcal{U}$. Therefore, $\mathcal{U}$ uniquely and unambiguously describes $\mathcal{S}$. In order to simplify the representation, it will be assumed that $\mathcal{S} = \mathcal{U}$ and *submit* $=$ *utilise*, which is the case in the popular proposals for the electronic cash technology [12, 38]. Although this research has not identified a scenario where *submit* $\neq$ *utilise*, it may have useful applications. Therefore, the thesis must accomodate such a scenario, which future applications may employ.

4. Tracing protocol between the set of trustees and any authorised entity provides a mechanism for determining the tuple *utilise* $\in \mathcal{U}$ given *issue* $\in \mathcal{I}$, or *issue* $\in \mathcal{I}$

given *utilise* $\in \mathcal{U}$. Let $\mathcal{T}$ be a set of tuples that contains all the legal tuples that describe possible conversations of this protocol. Every instance of a legal conversation could then be represented by the tuple $tr \in \mathcal{T}$, as shown in Figure 5.1.

In the simplest form, the token issuing protocol allows the client to authenticate to the TIA and obtain a certificate on a pseudonym. The token utilisation protocol allows the client to *prove* to the TAA that it has authenticated to the TIA, without identifying itself or the public tuple (*issue* $\in \mathcal{I}$) of the token issuing protocol. The properties of the transactions in ATS that were identified as important are as follows:

FP0 : Valid token issuing tuples, *issue* $\in \mathcal{I}$, can be formed only with the assistance of the TIA.

FP1 : *issue* $\in \mathcal{I}$ and *utilise* $\in \mathcal{U}$ must possess a one-to-one relationship;

FP2 : for every entity, excepting the client, it must be intractable (or difficult) to compute *utilise* given *issue*;

FP3 : for every entity, excepting the client, it must be intractable (or difficult) to compute *issue* given *utilise*.

The issues in ATS that were identified as important are as follows:

Authorisation: all the tuples in the set $\mathcal{I}$ can be formed only after an interaction with the TIA, and given a legal tuple $issue_1 \in \mathcal{I}$ it must be intractable to compute another tuple $issue_2 \in \mathcal{I}$, without interacting with the TIA. An intuitive approach to achieve this property is to include secure signature ciphertexts as a part of the legal tuples in the set $\mathcal{I}$. Property **FP0** is important for this issue.

Anonymity : given *issue* it must be infeasible to determine *utilise*. And, given *utilise* it must be infeasible to determine *issue*. Properties **FP2** and **FP3** are central to this issue.

Reusability : the number of successful token utilisation protocols must be uniquely determined by the number of successful token issuing protocols. In the simplest case, there must exist a bijection between the set $\mathcal{U}$ and the set $\mathcal{I}$. **FP1** is an essential property in this regard.

Traceability : optionally it may be necessary to *uniquely* determine:

- the tuple *utilise* $\in \mathcal{U}$ given the tuple *issue* $\in \mathcal{I}$. This issue is called *token tracing*, which is known as coin tracing in electronic cash systems;
- the tuple *issue* $\in \mathcal{I}$ given the tuple *utilise* $\in \mathcal{U}$. This issue is called *client tracing* or client identity tracing, which is known as owner tracing in electronic cash systems.

FP1 is fundamentally important for all solutions for tracing.

The requirements for anonymity and traceability are contradictory requirements, but they can be achieved by providing:

restricted confidentiality service for the identity of the client, which achieves a trust-based anonymity service and a traceability service, to achieve Properties **FP2** and **FP3**; and,

universal integrity service for the tuples *issue* $\in \mathcal{I}$ and *utilise* $\in \mathcal{U}$ to achieve Property **FP1**: in order design compliance checking mechanism for the verification of traceability.

Therefore, there will exist message formats, in ATS with support for anonymity revocation, that can be classified under compliance Category 1, as discussed in Section 2.3.

5.2 Examples and Applications of ATS

The previous section provided the properties of ATS. The concrete solutions that achieve these properties are presented in Section 5.2.1. Section 5.2.2 will present an explanation for the protocol failure in the proposal for a payment system that employed an ATS, without explicitly identifying with this terminology. Section 5.2.3 will propose a generic schema for the design of a class of protocols called secure selection protocols by employing the ATS. The schema will present a design heuristic to avoid design flaws identified in Section 5.2.2.

5.2.1 Electronic Cash Technology Based on the Discrete-Log Problem

The electronic cash technology is a natural candidate for an ATS. The aim of the electronic cash technology is to hide the relationship between the set of withdrawal transcripts and the spending transcripts. The withdrawal protocol, which allows the customers to create e-coins with the assistance of the bank, generates the withdrawal transcript. The spending protocol, which allows the customers to prove ownership of the so-formed e-coins to the merchant, generates the spending transcripts. Therefore, the withdrawal protocol is an ideal candidate for the token issuing protocol and the spending protocol can be the token utilisation protocol.

Popular approaches for the design of electronic cash systems are based on techniques that facilitate the creation of specialised certificates for keys. The certification procedure prevents any entity, other than an optional trusted third party, from determining the identity of the owner of the key from the information contained in the certificate and the key. The certification mechanism may employ a *suitable form* of verifiably encrypted signature tuples to achieve the goals. Such a specialised certification procedure, invariably, achieves the properties required for an ATS.

Chaum and Pedersen [18] presented an electronic cash scheme based on the discrete logarithm problem by employing a blind Schnorr signature scheme [81]. This proposal has been widely researched and employed in many subsequent proposals, including the proposal for a restrictive blind signature scheme by Brands [12] and its enhanced version supporting anonymity revocation by Frankel, Tsiounis, and Yung [38]. This section will explain the dynamics of the e-cash scheme by Frankel, Tsiounis, and Yung.

The system consists of the mint (or bank) acting as the TIA, the customer (client), the merchant acting as the TAA and a trustee. The bank and the customer employ the token issuing protocol (withdrawal protocol) to compute the tuple, *issue* $\in \mathcal{I}$, and, a tuple S_c known only to the client, and another tuple S_{tia} known only to the TIA. The tuple *issue* $\frown S_c \frown S_{tia}$, where $\frown$ is the tuple concatenation operator, must be unique and, usually, a function of the long term private keys of the TIA and the client. The steps involved in the token issuing protocol (withdrawal protocol) are as follows:

1. The mint commits to two inputs, $a' = h_1(w)$ and $b' = h_2(I, w)$ such that h_1 and

h_2 are one-way functions, and w is a random, secret value;

2. The customer employs the commitments to calculate secret, integrity keys $a = h_3(a', u, v)$ and $b = h_4(b', u, v, s)$, such that h_3 and h_4 are one-way functions and u, v and s are random, secret values;

3. The customer computes the checksum of the parameters to be signed by employing the secret, integrity keys computed in the previous step as: $c = \mathcal{H}(\cdots, a, b)$;

4. The customer encrypts the checksum, c, by employing a probabilistic encryption scheme as $c' = f_1(c, u)$ and sends c' to the mint;

5. The mint calculates the signature of c' by employing its private key X_B as $r' = S(c', w, X_B)$ and returns the signature r' to the customer;

6. The customer encrypts the signature r' by employing a probabilistic encryption scheme as $r = f_2(r', u, v)$.

At the end of this process the mint and the customer would have the knowledge of *issue* $= (a', b', c', r')$, the withdrawal transcript, the mint's secret values are $S_c = (w)$ and the secret values of the customer are $S_c = (a, b, c, r, u, v, s)$. Note that the tuple *partspend* $= (a, b, c, r)$ would be a subsequence of the spending transcript, *utilise* $\in \mathcal{U}$.

Suppose the verification equation for the signature system is represented by $V(r, c, y_B) := o$, where $o \in \{0, 1\}$ and 1 denotes successful verification only when $r = S(c, w, X_B)$. It should be true that:

$$V(r', c', y_B) = V(S(c', w, X_B), c', y_B) = 1 \tag{5.1}$$

if the bank did indeed sign c' (step 5). That is, the message c', which is known to the mint, and the mint's signature must be successfully verified.

In order to successfully verify the *blinded* version of the signature tuples, (c, r), the following equation must be valid:

$$V(r, c, y_B) = V(f_2(r', u, v), c, y_B) = 1 \tag{5.2}$$

Substituting for r' and c' in terms of r and c the following equation can be deduced:

$$V(r, c, y_B) = V(f_2(S(f_1(c, u), w, X_B), u, v), c, y_B) = 1 \tag{5.3}$$

which is the universal verification equation employed by the merchant during the spending protocol (token utilisation protocol) to determine the bank's signature. The composition of functions denoted by $V(f_2(S(\cdots), \cdots))$ represents signature, unblinding (a probabilistic encryption process) and verification operations. Thus, the scheme provides a mechanism for the verification of a blinded (encrypted[2]) signature tuple.

Comparing Equations 5.1 and 5.3, there must exist a bijection between the sets of tuples $\mathcal{I} = \{(S(c', w, X_B), c')\}$ and $\mathcal{U} = \{(f_2(S(f_1(c, u), w, X_B), u, v), c)\}$ to achieve the properties discussed in Section 5.1. Function f_2 provides confidentiality for the first term in the tuples and function f_1 provides the confidentiality service for the second in the tuples. The signature function S guarantees the bijective property between the two sets. In the scheme by Chaum and Pedersen [18] and its variants [12] and [38], the functions $S(c', w, X_B) = w + c'X_B$, $\mathcal{H}$, $f_1(c, u) = c/u$ and $f_2(r', u, v) = r'u + v$ achieved Properties **FP0, FP1, FP2** and **FP3.**

In order to enable universal tracing, the clients must encrypt their identity for the trustees in a ciphertext, e, by employing a probabilistic encryption algorithm such as the ElGamal encryption algorithm. The clients must then prove to the TAA that the identity encrypted in e is the same as the identity embedded in the certified integrity key, b, (Steps 2 and 3 of the token issuing protocol) which is available as a part of the tuple $\mathit{partspend}$ - a subsequence of the spending transcript $\mathit{utilise}$, in minimal knowledge [38]. Since, b is also a ciphertext that provides confidentiality service to the identity of the customer, the spending protocol employs a publicly verifiable encryption algorithm of type Class 1, as discussed in Section 2.2.1. The tuple describing the resulting conversation of this proof is called $\mathit{traceproof}$. The spending transcript would then be, $\mathit{utilise} = \mathit{partspend} \frown e \frown \mathit{traceproof}$, where $\frown$ is the tuple concatenation operator. The spending transcript provides the proof of participation in the token issuing protocol due to the $\mathit{partspend}$ tuple and the proof of traceability of the customer due to the tuple $e \frown \mathit{traceproof}$.

The proposals for electronic cash proposal by Brands [12] and Frankel, Tsiounis and Yung [38] (FTY scheme) are ideal candidates for ATS. The FTY scheme is identical to the Brands scheme, with the exception of the universal tracing service for the

[2]Blinding may be treated as a form of encryption. The primary goal for both these terminologies is confidentiality. Section 6.2 presents a future research direction that employs this interpretation for the design of ATS.

set of trustees. Appendix D presents a detailed description of the FTY scheme.

5.2.2 Analysis of a System that used ATS

The restrictive blind signature scheme proposed by Brands [12] is a good example for an ATS. It achieves all the properties of the ATS (see Section 5.1) as discussed in the previous section.

Radu, Govaerts and Vandewalle [76] proposed an electronic payment system (RGV proposal) that *used* the proposal by Brands as an ATS. The flaw in the RGV proposal outlined in Section 3.3.1 suggests that a system that employs a secure ATS may still be insecure. Therefore, systems that employ ATS must be carefully designed. A primary problem with the RGV proposal was its effort to correlate the anonymity service provided by independent systems. The resulting deficiency of the protocol can be described in terms of the properties of an ATS.

The withdrawal phase consisted of three phases: *get_pseudonym*, *withdraw_big_coin* and *exchange_big_coin*. Let PS_i be the set of legal transcript tuples of the *get_pseudonym* protocol between participant i and the TIA. Similarly , let BG_i be the set of legal transcript tuples of *withdraw_big_coin* and XBG_i be the legal transcript tuples of *exchange_big_coin*. Note that $ps_i \frown bg_i \frown xbg_i \in \mathcal{I}$, where $\mathcal{I}$ is the set of legal tuples of the token issuing transcripts, as discussed in Section 5.1, $\frown$ is the tuple concatenation operator, $ps_i \in PS_i$, $bg_i \in BG_i$, and $xbg_i \in XBG_i$.

In order to provide robust traceability in the scheme, there must have been a bijection between PS_i, BG_i and XBG_i. Although, there existed a bijection between the sets BG_i and XBG_i, the flaw outlined in Section 3.3.1 proved the non-existence of such a relationship between PS_i and BG_i. This flaw allowed the customers to link a tuple from PS_i with a tuple from BG_j, which allowed participant i and participant j to transfer funds between themselves without the knowledge of any authorities. Since the *exchange_big_coin* protocol did not provide mechanisms for trustees to trace universally the tuples from the set XBG_j, participant j could perform a perfect crime [92].

The proposal does not achieve Properties **FP0** and **FP1** because the participants can collude to obtain *unauthorised* valid tuples and avoid tracing.

5.2.3 A Generic Schema for the Design of SSP

ATS can be employed as a sub-system to provide the anonymity service. A protocol sub-system, which achieves the requirements that are specific to the application instance of SSP, can be *securely* interfaced with the ATS. The word "securely" is stressed to highlight the potential pit-falls of such an interfacing, namely the deficiency in the RGV proposal described in the previous section.

SSP deal with the provision of the confidentiality and the integrity service to a tuple of the form (I, D), where I represents the identity of a registered user and D represents the choice of the user, or simply a data that is to be associated with the user. The confidentiality service for the tuple is essential to prevent unauthorised entities from associating the value of D with an identity I. The integrity service is essential to prevent any entity, including I, from altering the value of D in an unauthorised manner. There are many instance applications of SSP, namely peer-review systems, electronic auction systems, electronic voting systems, and payment systems that may be more complex than a simple e-cash system.

In this section, a generic design schema for SSP is presented. The schema employs the ATS as a sub-system, by interfacing it with an application specific protocol sub-system. The ATS is employed as a specialised certification system that is used by the protocol sub-system. The public-keys certified by the ATS are employed in a suitable manner to provide confidentiality and integrity services to various messages. The protocol sub-systems will not generate certificates for newly generated public-keys that are not related to the public-keys certified by the ATS. This approach guarantees the prevention of deficiencies such as the RGV proposal. The three phases of the schema are as follows:

Token Issuing Phase: The participants in the system *authenticate* to the token issuing authority (TIA) and obtain a certificate, which is the *anonymous token*, for their pseudonym – the pseudonym is known only to the participant during this phase;

Service Registration Phase: The participants *anonymously* contact the token accepting authority (TAA), present the anonymous token and prove ownership of the token, and submit the data (the choice) to the TAA in a suitable form (such as a plaintext, verifiably encrypted ciphertext for a trustee, a commitment, a signature and so on);

Service Delivery Phase: the participants, *anonymously* either enable the service they had registered for or obtain the results of their choice.

The cumulative result of the three phases is the confidentiality service for the identity of the participants and the integrity service for the choice of the participant. The protocol sub-system may additionally provide restricted confidentiality service for the data, if required.

A crucial aspect of the schema is the interfacing of the two sub-systems, namely the ATS and the protocol sub-system. A prudent practice for the design of the protocol sub-system is to avoid the design of protocol goals that will fundamentally contradict the services of the ATS. The ATS provides the anonymity service and optionally the anonymity revocation service and non-transferability services. The design of the protocol sub-system must not duplicate these services. An example of a duplication of service that would potentially undermine the services of the ATS would be the independent provision of anonymity service by the protocol sub-system. Such a provision will undermine the traceability and the anonymity revocation services of the ATS, as demonstrated in Section 5.2.2.

In general, the interfacing can be visualised as a *transfer of service* from the providing protocol sub-system to the client protocol sub-system. Once such a transfer happens the client protocol sub-system must *preserve* the services. For example if a key-agreement sub-system is interfaced with a client protocol sub-system, then the client protocol sub-system must preserve the confidentiality and integrity properties of the session-key provided by the key-agreement sub-system.

The schema will be employed to design a peer-review system in Section 5.3, an electronic auction system 5.4 and to discuss the possibility of an electronic voting system employing the schema in Section 5.5. The ATS used in these applications are the e-cash techniques discussed in Section 5.2.1. Future developments in the design of ATS, as, for example, discussed in Section 6.2, can be easily incorporated in the schema without affecting the goals of the individual applications.

5.3 Analysis and Design of a Peer Review System

This section will propose a solution for an instance of SSP called the peer review problem. The peer review problem consists of a set of participants called peers, having two roles in the system, namely that of the reviewer and the candidate to be reviewed. No participant should review itself: a solution to the peer review problem is a permutation of a set of participants *with no fixed points*. The properties of the peer review protocol are:

1. The solution must define a permutation without any fixed points.
2. Every reviewer is also a candidate.
3. The solution must provide one-way anonymity service for the reviewers. That is the reviewers know the identity of the candidate, but the candidate does not know the identity of the reviewer.

The proposal will employ the three phased protocol schema, detailed in Section 5.2.3, to solve the problem.

Any form of peer review system must contain at least *four* participants and at least *three* of them must be honest, in order to provide minimal confidentiality service for the identity of the reviewer. Otherwise, in the case of anonymous peer-review systems, the system cannot provide anonymity. Suppose that A, B and C are the participants, and the set of ordered pairs containing the reviewer and the candidate is $\{(A, B), (B, C), (C, A)\}$. A will know that C is its reviewer because it is reviewing B and if B is reviewing A then C has to review itself, which is not allowed. Suppose that there are four participants with D being the fourth participant, and C and D are the two dishonest participants – without loss of generality. C and D can collude by revealing their choices to each other, which would effectively reduce the four participant system to a three participant system that does not provide the required confidentiality services. The reasoning for the case when $n = 2$ is trivial.

A challenging (and interesting) problem that is inherent in the problem statement is that when two participants collude they will be able to obtain some information that could *weaken* the anonymity of honest participants. The information that colluding participants obtain is inversely proportional to the total number of participants in the system and directly proportional to the number of colluding participants. Overcoming this predicament could be difficult without weakening the security services for honest participants. A solution for this situation remains an open-problem, which seems to be similar (but not same) to the receipt-freeness problem in electronic voting systems [69].

The compliance requirements of this problem can be stated as follows:

1. the identity of the reviewer must be confidential;
2. every reviewer must be an authenticated candidate;
3. the relationship between the reviewer and the candidate cannot be changed after successful completion of the peer review protocol – that is integrity service for the tuples of the form (reviewer ID, candidate ID) must be universal; and,
4. it must be possible to revoke the confidentiality service provided to the identity of the reviewer by a set of trustees.

When the schema is employed to solve this problem, the anonymous token, AT, must provide confidentiality service to the identity of the reviewer and the peer review protocol must provide universal integrity service to the selection of the reviewer or the candidate, depending on the approach taken by the peer review protocol sub-system.

5.3.1 Basic solution

A simple solution to solve the peer review problem may consist of three steps.

Step 1 Every participant wishing to participate in the protocol signs a random message and publishes the signature and message in a publicly readable bulletin board, B_1. Let the number of signatures in B_1 be n, which is the number of participants.

Step 2 Each participant generates a random pseudonym and *anonymously* publishes its pseudonym in a publicly readable bulletin board B_2. The step completes when n pseudonyms are published. Let the set of pseudonyms be represented by PS.

Step 3 Each participant in turn chooses a pseudonym from B_2, such that it does not select the pseudonym it submitted. Let this choice be ps. The participant then generates the proof for its knowledge of the secret corresponding to one of the pseudonyms in the set $PS \backslash \{ps\}$, without revealing its pseudonym. It signs its identity, choice and the proof, and submits the signature along with the message to a bulletin board B_3. It also removes its choice from B_1, so that nobody else can make the same choice. This phase completes when n valid messages along their signatures are present in the bulletin board B_3. Anyone may check if every public key used for verifying the signatures in B_1 is also used in B_3.

Drawbacks and solution: The protocol proposed assumes honest participants, which may not be very desirable. The protocol has the following drawbacks:

P1 Two participants, say i and j, can reveal their pseudonyms as u_i and u_j to each other, so that they can select each other.

P2 Two participants, say i and j, can generate the transcripts in Step 3 for each other, so that they can select themselves.

P3 Since v_i is only a short term secret, participant i can reveal this value to j, so that j can select twice. This would allow j to select itself.

P4 The system does not provide anonymity revocation, which may be required in common applications.

P5 An attacker can mount a denial of service attack on the system and be unidentified, because Step 2 does not guarantee that only the participants involved in Step 1 can submit *only one* pseudonym.

It seems difficult to overcome problem P1. Moreover, P1 does not *adversely* affect the goals of the protocol when the number of honest participants are in majority. But P2 and P3 do adversely affect the goals of the protocol. These problems can be solved if the participants are forced to use their long term secret values, namely the private key corresponding to their certified public key, to generate the transcripts in Step 3. The assumption is that the participants would not, in their own interest, reveal their long term private key to anyone — the private key corresponding to the certified long term public key may provide access to the participant's bank account or health care system records or some crucial information repository or source. P4 can be solved by linking Step 2 to Step 1, so that the link can be computed if necessary. P5 can be solved by issuing *only one* anonymous token to every participant who registered in Step 1 and accepting *only one* pseudonym for every anonymous token in Step 2. The next section will present a method for solving some of the abovementioned problems.

It is interesting to note that problems with similar traits as P2 and P3 are observed in other protocol applications as well. Non-transferability of electronic cash [72], receipt-free electronic voting [69] and prevention of purchase of votes [66] are some examples.

5.3.2 The Protocol Schema

It will be assumed that all participants possess certified public keys that support digital signature and authentication schemes. The necessary entities in the system are a token issuing authority (or token issuer) TIA, whose public key y_t is available to all the participants through a secure channel and a token accepting authority (or supervisor) TAA whose role is to act as a monitor of the system. There need be no explicit trust placed on TAA due to the use of publicly verifiable proof systems. Let the system have n (such that $n > 3$) participants. The three phases of the schema are:

Phase 1 (Token Issuing Phase): Participant i generates a message C_i (for commitment), signs this message using its public key, say y_i, sends the message and the signature, say D_i, to TIA and obtains an anonymous token (which is also a certificate), AT_i, such that only participant i knows the ordered pair, (y_i, AT_i). All participants must participate in this phase before proceeding to the next phase. The participation can be checked when n unique, valid signature tuples, (C_i, D_i), are submitted and n tokens are withdrawn from TIA.

Phase 2 (Service Registration Phase): Participant i (anonymously) submits AT_i to TAA, proves ownership of AT_i, submits its pseudonym, $u_i = \mathit{secret}(v_i)$, where *secret* could be a one way function, and keeps v_i as its secret. After verifying the proof transcripts, TAA publishes $(u_i,\ AT_i)$ in a publicly accessible directory, along with the proof transcripts. All participants must participate in this phase before proceeding to the next phase. The participation can be checked when n tokens are submitted to TAA. In order to create a strong link between Phase 1 and this phase, the value of u_i must be a function (or part) of the anonymous token AT_i. In other words, it cannot be randomly generated during this phase.

Phase 3 (Service Delivery Phase): Participant i chooses its reviewer to be the owner of the pseudonym u_j, such that $j \neq i$, generates transcripts to prove that it knows the secret value corresponding to *one* of the $n-1$ public values in the set $\{u_l \mid l \neq j\}$ and commits to the choice by signing the choice and the transcripts of the proof. If TAA successfully verifies the proof transcripts and the signature, it publishes the tuple $(y_i,\ u_j)$ along with the proof and signature transcripts in a public directory. Participant j can query the public directory (or, to achieve maximum anonymity, download the entire database to a secure storage area that it controls and query the local copy of the database) to know the identity of its candidate, y_i. If n participants complete this phase and the public key used for verifying D_i was used to verify the signature of the commitment to the choice then, TAA announces the protocol to be complete. If participant n cannot prove that it knows the secret corresponding to one of the $n-1$ public values in the set $\{u_l \mid l \neq j\}$, then u_j must be its pseudonym. This event results in a deadlock[3]. In which case, TAA announces the protocol to be incomplete and all the participants must start the protocol anew from Phase 1.

Since the technology used to generate AT_i provides computational anonymity, the resulting system will provide *fair* peer review. If AT_j can be linked to y_j in Phase 1, then y_j can be linked to y_i, by linking the tuple $(y_j,\ AT_j)$ with the tuple $(u_j,\ y_i)$, which would be publicly available from Phase 2 of the protocol.

[3]The analysis of the probability of deadlock occurrence as a function of the number of participants is presented in Appendix E. The preliminary analysis suggests the probability to have an upper bound of $1/(1+n)$, where n is the total number of participants.

5.3.3 The Protocol

The cryptographic tools to be used in this section are proof of equality of discrete logarithm (PEDL) (see Appendix B.1), partial proof of knowledge of discrete logarithm (PPEDL) (see Appendix B.1.2) and the electronic cash technology as proposed by Frankel, Tsiounis and Yung (see Appendix D).

System setup The supervisor of the system, TAA, selects a large prime p such that computing discrete logarithms in $\mathbb{Z}_p$ is intractable. TAA also selects a generator g, of the group $\mathbb{Z}_p^*$. Henceforth, all arithmetic will be performed in the congruence class modulo p, unless stated otherwise. The token issuer, TIA, possesses a public key y_t of the form $y_t = g^{x_t}$, where $x_t \in_R \mathbb{Z}_p^*$ is the private key corresponding to y_t. The tuple $(g,\ p,\ y_t)$ is published as the public parameters for the selection system. The supervisor maintains two bulletin boards with read permission for everyone and edit permission only for the supervisor. Let the two bulletin boards be labelled $\mathcal{A}$ and $\mathcal{B}$. Bulletin board $\mathcal{A}$ will contain unselected pseudonyms and bulletin board $\mathcal{B}$ will contain the selected pseudonyms.

Let there be n participants in the system, such that $n \geq 4$. The public key of participant i, $y_i(= g^{x_i} \mid x_i \in_R \mathbb{Z}_p^*)$, is published in a certified public directory with x_i as the corresponding private key. Every participant in the system possesses a certified public key.

Additional system parameters required for the anonymous token system (see Appendix D) are also published.

Phase 1 Participant i generates and signs a message to obtain a message-signature tuple (C_i, D_i) and, sends the tuple to TIA (who verifies the signature using i's public key). The token will be a blind signature on a message by TIA that can be verified using its public key y_t. Participant i chooses a random value $v_i \in_R \mathbb{Z}_p^*$ and computes $u_i = g^{v_i}$. The participant then lets u_i be the message to be blindly signed by TIA and obtains an anonymous token AT_i by executing the *IssueToken* protocol of the anonymous token system (see Appendix D) with TIA. Thus, $AT_i := \langle u_i,\ Cert_{u_i}\rangle_i := IssueToken(i, \mathtt{TIA}, \{v_i\}_i, \{x_t\}_{\mathtt{TIA}})$.

Phase 2 The following steps are performed by individual participants and the TAA:

Step 2.1 Participant i anonymously contacts TAA, presents the tuple $(AT_i,\ u_i)$ to TAA, engages in the *UtiliseToken* protocol with TAA. This step is represented by the equation:

$$\langle Proof_{u_i} \rangle := UtiliseToken(u_i, Cert_{u_i}, \texttt{TAA}, y_t, f_2, \{v_i, x_i\}_i)$$

as explained in Section D.2. Note that the tuple (v_i, x_i) are the private keys corresponding to the pseudonym and the long term public key, respectively, of Participant i. Additionally, f_2 is the public key of the trustee, if one exists, who can revoke the anonymity service from Participant i (refer to Appendix D, Section D.2 for details). TAA checks if AT_i is a valid token issued by TIA on u_i.

Step 2.2 If TAA successfully verified the transcripts then it publishes the tuple $(AT_i,\ u_i,\ Proof_{u_i})$ in a public directory.

Step 2.3 TAA enters u_i into $\mathcal{A}$.

All participants must complete this phase before the protocol can proceed to the next phase.

Phase 3 The following steps are performed by individual participants and TAA:

Step 3.1 Participant i authenticates to TAA using its public key $y_i(= g^{x_i})$.

Step 3.2 Participant i chooses a pseudonym u_j such that $j \neq i$ from $\mathcal{A}$.

Step 3.3 Participant i presents u_j to TAA along with the output of the algorithm for partial proof of discrete logarithm, PPEDLGen (see Section B.1.2), with input $(\{u_l \mid l \neq j\}, u_i, v_i, x_i)$ and output $(d_i, \ , c_i,\ \{c_{il} \ \mid l \neq j\})$. $\{u_l \mid l \neq j\}$ is the set of pseudonyms of all the participants of the system *excepting* u_j, which is the choice of Participant i.

Step 3.4 TAA verifies the output of the algorithm sent by Participant i using the partial proof of discrete logarithm algorithm, PPEDLVer (see Section B.1.2), with input $(\{u_l \mid l \neq j\},\ y_i,\ d_i,\ ,c_i,\ \{c_{il} \mid l \neq j\})$ and output in $\{0,1\}$ ($\{$SUCCESS, FAILURE$\}$). If it successfully verifies the transcripts using the public key y_i, it removes u_j from $\mathcal{A}$, adds it to $\mathcal{B}$ and publishes the tuple $(u_j,\ y_i)$ along with the transcript $(\{u_l \mid l \neq j\},\ y_i,\ d,\ ,c,\ \{c_l \mid l \neq j\})$ in a public directory.

Step 3.5 Participant j can consult the public directory (in a secure manner – to achieve maximum anonymity) to find y_i as its candidate to be reviewed. Participant j keeps this knowledge as its secret.

When Participant n (the last participant), with public key $y_n(= g^{x_n})$, engages in the protocol for Phase 3, there will be only one entry in $\mathcal{A}$. If the last entry is u_n (the pseudonym of Participant n), then there will be deadlock. In the case of a deadlock, Participant n cannot generate valid transcripts in Step 3.3, as it will not possess the knowledge of discrete logarithm for any of the elements in the set $\{u_l \mid l \neq n\}$. Participant n must then prove that it knows the discrete logarithm of u_n by sending the output of the algorithm for PPEDLGen with input $(\{u_n\},\ u_n,\ v_n,\ x_n)$ and output $(d_n,\ c_n,\ c_n)$ to the TAA[4]. Observe that the algorithm PPEDLGen with the input set containing only one element ($\{u_n\}$) will be similar to the Schnorr signature algorithm, which proves the knowledge of discrete logarithm of a given value — in this scenario the transcripts prove the knowledge of discrete logarithm of u_n *and* y_n simultaneously. If TAA successfully verifies the PPEDLVer with inputs $(u_n, y_n, d_n, c_n, c_n)$ and output in $\{0,\ 1\}$, then it publishes the tuple $(u_n, y_n, d_n, c_n, c_n)$ in a public database and announces the protocol to be incomplete. In this case all participants must restart the protocol from Phase 1. If no deadlock occurs then the protocol iteration is announced to be complete. This can be detected when the last participant successfully completes Step 3.4, in Phase 3.

Anonymity revocation: The ATS employed in this proposal supports anonymity revocation. Let the tuple *utilise*$_i \in \mathcal{U}$ describe the token utilisation conversation corresponding to the token AT_i. Anonymity revocation is achieved by determining the tuple *issue*$_i \in \mathcal{I}$, describing the token issuing conversation, corresponding to *utilise*$_i$.

[4]c_n is twice because the set $\{u_n\}$ in the input contains only one element.

As discussed in Section 5.1, the trustee has the power to determine $issue_i$, which will contain the identity of the customer owning the token AT_i. Appendix D presents the equations for the mechanisms.

Thus, the TAA, TIA or any other authorised entity can engage in the *Trace* protocol, explained in Appendix D.2, with the trustee to obtain the tuple $(y_i,\ AT_i)$, which can link y_i to u_i when the public information $(AT_i,\ u_i)$ is produced.

5.3.4 Security Analysis

This section will present an analysis of the phases to elucidate its achievement of the desired properties.

Property 1 (Permutation without fixed points): In Phase 1, when Participant i authenticates to TIA using its public key y_i, it receives only one AT_i. If more than one token was issued to Participant i using y_i, then TIA can be held responsible (all transcripts are publicly verifiable and signed by individual entities). Phase 2 allows only one pseudonym to be submitted for every AT_i. Phase 3 requires Participant i to prove its knowledge for at least one pseudonym in the set of pseudonyms that does not contain its choice. In order to pass this phase, Participant i cannot choose itself. Thereby, the protocol is a permutation without fixed points.

Property 2 (Bijection between the sets of reviewers and candidates): Since every user is allowed to submit only one pseudonym and selects a different pseudonym (from the set of submitted pseudonyms), every reviewer is also a candidate.

Property 3 (Anonymity service for the reviewers): Reviewers are anonymous from the candidate and the candidate is not anonymous from the reviewer. Since every user chooses the pseudonym of its reviewer *after* authentication (using the public key, say y_i), this choice is public and the reviewer (say u_j) can know the identity of the candidate. From the publicly known tuples $(AT_j,\ u_j)$ and $(u_j,\ y_i)$ (in Phases 1 and 2), candidate i cannot know the identity of reviewer j (y_j), if the technique used for generating anonymous tokens does provide anonymity. Candidate i cannot obtain the tuple $(y_j,\ u_j)$ by observing the protocol runs in

Phase 3, if the proof system used is witness indistinguishable. Proposition 5.1 provides the proof for this property.

Theorem 5.1 *Assuming that the participants do not collude and the electronic cash technology prevents any entity other than participant i to compute the tuple $(y_i,\ AT_i)$, the system provides the anonymity service to the reviewers.*

Proof: TAA, by itself or in collusion, cannot correlate the values y_i and u_i, using the public knowledge $(AT_i,\ u_i)$. Since the functions PPEDLVer and PPEDLGen provide a proof that is witness indistinguishable (see [23]), TAA, by itself or in collusion, cannot correlate the value y_i with u_i using the outputs of the function PPEDLGen, as computed by Participant i. □

If the proof systems used for the anonymous token technology and partial proof of knowledge protocol construct are *publicly verifiable*, then the trust level on the token issuer, TIA, and the supervisor, TAA, can be considerably reduced. The advantage of this approach is that it does not make any assumptions on the possible inclusion of anonymity revocation mechanism. This is an advantage of abstracting anonymous token, AT_i, to provide this service. Anonymity revocation mechanisms can be built into the token technology without affecting other core functionality of the protocol (permutation without fixed points).

5.4 Analysis and Design of Sealed-Bid Auction System

The fundamental goal of auction systems is the distribution of scarce resources among, potentially, many bidders based on well devised rules to determine the winning strategy [64]. A common approach to protect the interests of individual bidders, from conspiring bidders and auctioneers, is the sealed bid auction system. A *seal* is employed to provide secrecy for a bid, until a pre-defined event. In the physical world, the sealed bid may simply be a sealed envelope that encloses a paper containing the value of the bid, along with optional non-repudiation information from the bidder. The sealing process guarantees a fair auction procedure for *honest bidders*. At the same time, there must be mechanisms to open the seal, after the occurrence of the specified event, to reveal the winning bidder in order to avoid disavowal after participation. A requirement for some systems, *but not necessarily for the sealing method*, is to protect the secrecy of the losing bids. This requirement provides restricted privacy for the losing bidders. The word "restricted" is important because once the identity of the bidders is known and the identity of the winning bidder and the corresponding bid value are published, automatically some information about the bid values of the losing bidders is revealed. The only approach to provide complete privacy for losing bidders would be to refrain from publishing the identity of all the bidders.

In order to electronically implement the sealed bid auction procedure, the first step is to design a suitable *sealing process*. Towards this end the requirements *specific* to the sealing process must be identified. Once such a sealing process is devised, this abstraction can be used along with other techniques to achieve a complete auction system.

5.4.1 Literature Review

Confidentiality of the bid has been of paramount importance for the design of electronic auction systems. To achieve confidentiality of bid some proposals [61, 52] used secret-sharing primitives to distribute the value of the bid among many trustees. If at least a threshold of the trustees are honest, they will not assist in opening the bid *before* the closing period. This approach generally results in inefficient systems, when public verifiability is required. This is because there exists no efficient protocol construct for publicly verifiable encryption [86] which is an essential building block for publicly verifiable secret sharing schemes. The other approach to publicly verifiable secret sharing is that of Schoenmakers [82], which is more efficient than the scheme by Stadler [86]. However, its application to the auction scheme will remain inefficient, as compared with the scheme to be proposed in the subsequent sections. An estimation for the number of exponentiations required for this approach will be provided in Section 5.4.4.

Sakurai and Miyazaki [80] proposed an elegant auction system where the confidentiality of bid is controlled only by the bidder. For non-repudiation of the bid, Sakurai and Miyazaki used the undeniable signature scheme. Unfortunately, the computational and communicational complexity of the scheme [80] is dependent on the number of participants, thereby rendering their system inefficient for large scale auction systems. Moreover, the scheme requires every bidder to be on-line, which may not be a desirable property in large scale auctions over open networks.

Sako [79] attempted to modify their proposal [80] using group encryption (for a group of trusted auctioneers), instead of the undeniable signature scheme, for sealing the bid. Due to this approach, the proposal by Sako lost the primary advantage realised by the proposal by Sakurai and Miyazaki, which is user-controlled confidentiality for the bid.

Harkavy *et al* [47] proposed an auction scheme based on secure distributed computing primitives. Although they claim the system to be moderately efficient, the security arguments for their scheme remain unclear.

The following are the properties important for the design of sealed bid auction systems:

Confidentiality of bid: Only the bidder must know the bidding strategy until the closing period.

Non-repudiation of bid: The winning bidder must not be able to repudiate or change the bidding strategy.

Publicly verifiable auction: Any monitor must be able to verify the validity of the auction procedure.

Anonymity of bidder: The bidder-bid relationship must be known only to the bidder, unless the bid conforms with the winning strategy.

Independence of auction rules: The security protocols for auction rules must be independent of the auction rules.

5.4.2 The Approach

The system consists of two sub-systems, an anonymity sub-system that provides anonymity to all its users and an auction sub-system that allows the users to participate in the auction procedure. Thus the system, in effect, provides an anonymous auction service. The auction sub-system can be explained in terms of the following physical world entities. The auction system consists of a "magic seal," that will allow only the entity that sealed the bid, to open it. The bidders place their bid values inside an envelope and apply the "magic seal" to it. In order to register in a particular auction protocol, the bidders send the envelope to the auctioneer using a *registered post* service, which guarantees that the sealed bid will reach the auctioneer, who will not repudiate the receipt. When the actual auction procedure starts the bidders assist the auctioneer to break the "magic seal."

5.4.3 An Abstraction of the Sealed Bid

This section will propose a mechanism for sealing the bid, which is central to the notion of the sealed bid auction procedure. The sealing process can be defined as follows:

Definition 5.1 *The sealing process is represented by:*

$$\langle Proofs_S \rangle \;\; := \;\; Seal(b, r, I)$$

where $\langle Proofs_S \rangle$ contains the sealed bid values along with the transcripts for proof of knowledge of the bid value, b, a randomiser, r, and the identity (or public key) of the sealer (or bidder), I. Given $\langle Proofs_S \rangle$ the following must be true:

Hiding: *It must be intractable to determine the values of b or r.*

Binding: *It must be intractable to determine distinct tuples (b, r) and (b', r') such that,*

$$((\langle Proofs_S \rangle := Seal(b, r, I)) \quad AND \quad ((\langle Proofs_S \rangle := Seal(b', r', I))$$

Non-repudiation: *It must be intractable to determine (b, r, I) and (b', r', I') such that,*

$$((\langle Proofs_S \rangle := Seal(b, r, I)) \quad AND \quad ((\langle Proofs_S \rangle := Seal(b', r', I'))$$

unless $(b, r, I) = (b', r', I')$.

The requirements for non-repudiation encompass the requirements for binding.

There may be many approaches to realise the sealed bid. The most prominent of them would be:

1. the signed commitment approach. Here the bidder can use a suitable commitment scheme [71, 27] to commit to the bid and then sign the commitment value.

2. the signed encryption approach. Since semantically secure encryption schemes [45] can be idealised to be a commitment scheme, an encryption scheme can be used instead.

The first approach can be used when universal confidentiality is a requirement for the sealed bid. If *revocation of the confidentiality service*[5] from the sealed bid without the participation of the bidder is required, then the latter approach along with suitable *key recovery* techniques can be employed. In which case, the bidder can be expected to encrypt the value of the bid under the public key of a trusted entity. The proposal in this thesis will employ the first approach to provide universal confidentiality service for the sealed bid.

A Concrete Proposal for the Sealed Bid

Based on the Definition 5.1, a three-pass, Schnorr type [81, 27] protocol is designed to accomplish a sealed bid.

System Settings A prime order subgroup G of $\mathbb{Z}_p^*$ is chosen to be of order q such that, $p = 2q + 1$ for sufficiently large prime p, so as to render the discrete logarithm problem intractable. Two generators, g and g_1, for the group are published such that nobody knows[6] $\log_g g_1$. All operations are carried out in either $\mathbb{Z}_p^*$ or $\mathbb{Z}_q$ depending on the group being operated upon. The public key of the sealer (bidder) is *certified* to be $y_1 = g^{x_1}$ and $y_2 = g_1^{x_2}$.

Sealing Protocol An interactive protocol between the sealer and the receiver (of the seal) is as shown in Table 5.1. The sealer wishes to commit to the bid value $b \in \mathbb{Z}_q$ and identify himself/herself using the public keys y_1 and y_2. The sealer forms the commitment S to the bid value b and another commitment B for purpose of identification, and sends the two commitment values to the receiver. The receiver picks a random challenge c and returns challenge. The sealer then forms the response (s_1, s_2) with

[5]Note that revocation of the confidentiality service for the sealed bid is different from the revocation of the confidentiality service for the identity – anonymity revocation.

[6]When g_1 is the public key of a trusted entity or a Diffie-Hellman value of the public key of the trustee and the bidder, a similar approach can be used to design the signed encryption approach mentioned in the earlier section.

Sealer		Receiver
$a, d_1, d_2 \in_R \mathbb{Z}_q^*$		
$S = g^b g_1^a,\ B = g^{d_1} g_1^{d_2}$	$\xrightarrow{S,B}$	
	$\xleftarrow{c}$	$c \in_R \mathbb{Z}_q$
$s_1 = d_1 - cx_1,\ s_2 = d_2 - cx_2$		
$t_1 = s_1 - bc,\ t_2 = s_2 - ac$	$\xrightarrow{t_1,t_2}$	
		$B \overset{?}{=} (Sy_1y_2)^c g^{t_1} g_1^{t_2}$

Table 5.1: The Sealing Protocol

respect to the public key (y_1, y_2) and the commitment B. The sealer now uses (s_1, s_2) to respond to the commitment S as t_1 and t_2. The idea behind this concept is that, the tuple (S, B, c, s_1, s_2) is unique (with an overwhelming probability) in every protocol run and could not have occurred previously if the sealer or the verifier is honest. Therefore the responses t_1 and t_2 are unique in every protocol run. And so, the tuple (S, B, c, t_1, t_2) is also unique in every protocol run, with an overwhelming probability.

The following theorems for the proposal will assist in understanding its accomplishments and security.

Theorem 5.2 *The proposed protocol belongs to the class of honest verifier zero-knowledge protocols.*

Proof: The protocol belongs to the three pass, honest verifier class because the protocol follows the commitment-challenge-response model (see Appendix A), and the prover cannot verify the randomness of the challenge, c, chosen by the verifier. The protocol transcripts can be easily simulated by calculating $B = (Sy_1y_2)^c g^{t_1} g_1^{t_2}$ after choosing S, c, t_1 and t_2. □

Theorem 5.3 *If the values in the tuple (S, B, c, t_1, t_2) cannot be altered, then the protocol possesses the properties required for binding to the value of b.*

Proof: It is assumed that the system setup guarantees that nobody knows the discrete logarithm $\log_g g_1$, and computing discrete logarithm is intractable. To open the seal, $S = g^b g_1^a$ for $a, b \in_R \mathbb{Z}_q$, the sealer must reveal the tuple (b, a) and the verifier will check the equation $S = g^b g_1^a$. Given S and knowing a and b, assume that it is possible to open the commitment as $b' \neq b$. For this to happen the sealer must be able to reveal the tuple (b', a') such that $S = g^{b'} g_1^{a'}$. If the sealer can calculate the tuple (b', a') after calculating the commitment from the tuple (b, a) then it can compute the discrete logarithm $\log_g g_1 = (b' - b)/(a - a')$. □

The above proof is an adaptation of the proof for a theorem presented by Pedersen [71, Theorem 3.1].

Corollary 5.1 *Given the tuple (S, B, c, t_1, t_2), it will be infeasible to determine the value of b. Thereby, the protocol* hides *the value of b.*

Proof: The value of S is uniformly distributed in G if the value a is uniformly distributed in $\mathbb{Z}_q$. Thus, by itself S hides the value of b as discussed in the proof by Pedersen [71, Theorem 3.1].

Theorem 5.2 proved that the tuple (S, B, c, t_1, t_2) can be formed without knowing the tuple (b, a) or interacting with the sealer. Therefore the tuple hides the value of b according to the honest-verifier zero-knowledge proof technique (see Appendix A).□

Theorem 5.4 *When the sealer does not know the private keys corresponding to the public keys y_1 and y_2, and the discrete logarithm problem is hard, the sealer convinces the receiver with a probability of $1/2^{|q|}$, where $|q|$ is the size of q in bits.*

Proof: The sealer can cheat the receiver by guessing the challenge correctly in advance without knowing the private keys corresponding to the discrete logarithm problem. Then by Theorem 5.2 the sealer can form correct transcripts. If $|q| = \log_2 q$, then the number of legal challenges will be of the form $2^{|q|}$. When the receiver chooses the challenges at random, as prescribed by the protocol, the probability that the sealer will correctly guess the challenge is $1/2^{|q|}$. □

The Non-Interactive Version

The interactive protocol suggested in Table 5.1 can be converted into a non-interactive version using the Fiat-Shamir heuristic [37]. For this purpose, a collision intractable hash function $\mathcal{H} : \{0,1\}^* \mapsto \mathbb{Z}_q$ will be employed. The sealer performs the following function with the bid b, his/her private key (x_1, x_2) and the commitment value b as the inputs to obtain the output as (S, t_1, t_2, c).

```
Function Sealer
        with input: $(x_1, x_2, b, a, g, g_1, p, q)$
        and output: $(S, t_1, t_2, c)$
        $d_1, d_2 \in_R \mathbb{Z}_q^*$
        Compute:
        $S = g^b g_1^a \bmod p,\ B = g^{d_1} g_1^{d_2} \bmod p$
        $c := \mathcal{H}(y_1, y_2, S, B)$
        $s_1 = d_1 - cx_1 \bmod q,\ s_2 = d_2 - cx_2 \bmod q$
        $t_1 = s - bc \bmod q,\ t_2 = s - ac \bmod q$
End Function Sealer
```

The outputs of the sealing function can be verified by employing the following function:

```
Function VerifySeal
        with input: $(S, t_1, t_2, c, y_1, y_2, g, g_1, p)$
        output: $(Result)$
        If $c \overset{?}{=} \mathcal{H}(y_1, y_2, S, (Sy_1y_2)^c g^{t_1} g_1^{t_2} \bmod p)$, then
            $Result \leftarrow Pass$
        Else
            $Result \leftarrow Fail$
        EndIf
End Function VerifySeal
```

In this function the verifier checks the sealing transcripts against the public key of the sealer.

To open the seal the sealer can release the tuples (b, a). The values can be checked against the seal as follows:

```
Function VerifyOpenedSeal
        with input: $(a, b, S, t_1, t_2, c, y_1, y_2, g, g_1, p, q)$
```

```
                    and output: $(Result)$
                    If $S \stackrel{?}{=} g^b g_1^a$, then
                      $s_1 = t_1 + ac \bmod q$, $s_2 = t_2 + bc \bmod q$
                    Else
                      $Result \leftarrow Fail$
                      GoTo End Function
                    EndIf
                    If $c \stackrel{?}{=} \mathcal{H}(y_1, y_2, S, (y_1y_2)^c g^{s_1} g_1^{s_2} \bmod p)$, then
                      $Result \leftarrow Pass$
                    Else
                      $Result \leftarrow Fail$
                    EndIf
End Function VerifyOpenedSeal
```

In this function the verifier checks the tuples (b, a) against the commitment value S. If they are correctly verified the actual signature value (s_1, s_2) is computed from t_1 and t_2. The value of (s_1, s_2) is then checked for proper signature. Note that this is optional, because if the seal tuples pass the VerifySeal function and the tuple (b, a) are correctly verified against S, then (s_1, s_2) will be a legal signature tuple on S.

5.4.4 The Complete Auction System

A three phased auction system design that employs an anonymous token system (see Appendix D for nomenclature) and the process for sealing the bid proposed in Section 5.4.3 will be presented in this section.

System Settings: The system consists of a set of bidders $\mathcal{B}$, a mint, $\mathcal{M}$, for issuing electronic coins, a registrar, $\mathcal{R}$, an Auctioneer, $\mathcal{A}$, and a trustee $\mathcal{T}$.

A suitable prime-order subgroup, G of $\mathbb{Z}_p^*$, of order q, is chosen such that $p = 2q + 1$ is a large prime and the discrete logarithm problem is intractable. Suitable generators, g and g_1, are chosen such that $\log_g g_1$ is not known to any entity. The arithmetic operations are performed in the relevant groups. A suitable hash function $\mathcal{H} : \{0, 1\}^* \mapsto \mathbb{Z}_q^*$ is chosen. Additional system setting requirements for the ATS (See Appendix D) are published along with tuple $(p, q, g, g_1, \mathcal{H})$.

The public key of the following entities are published:

1. The public key of each bidder, $I = g^{u_1}$, where u_1 is the corresponding private key.
2. The public key of $\mathcal{R}$, $y_r = g^{x_r}$, where x_r is the corresponding private key.
3. The public key of $\mathcal{A}$, $y_a = g^{x_a}$, where x_a is the corresponding private key.
4. The public keys of the mint, $\mathcal{M}$, $y_M = g^{x_M}$ and the trustee $f_T = g^{x_T}$.

The Three Phases

The pictorial representation of the model for an auction system is presented in Figure 5.2. The three phases in the auction system are:

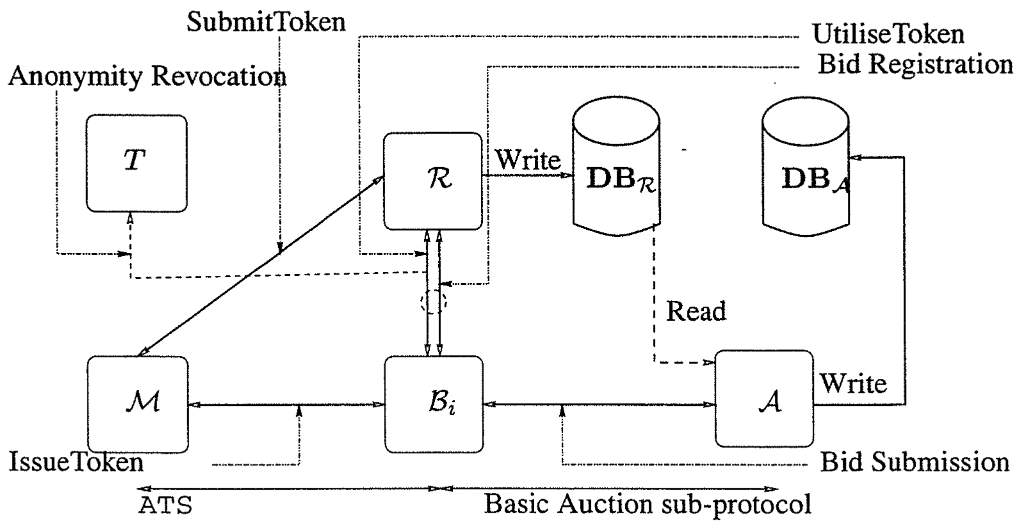

Figure 5.2: System Dynamics of the Auction System

Phase 1 (Token Issuing Phase): consists of the IssueToken function in the ATS. Bidder, $\mathcal{B}_i$ owning the public key I wishing to participate in individual auction activities, engages in the IssueToken protocol with the mint, $\mathcal{M}$, to obtain $\langle A_i, Cert_{A_i}\rangle$:

$$\langle A_i, Cert_{A_i}\rangle_I \quad := \quad IssueToken(I, M, \{s\}_I, \{X_M\}_M)$$

The interpretation for the token system is available in Appendix D.

Phase 2 (Service Registration Phase): consists of the UtiliseToken function of the ATS system and, the Sealer and VerifySeal functions presented in Section 5.4.3. The bidder, $\mathcal{B}_i$, performs the following tasks:

1. presents the tuple $(A_i, Cert_{A_i})$ to $\mathcal{R}$;
2. engages in the UtiliseToken protocol to convince its ownership of the tuple without revealing its identity. The protocol will be of the form:

$$\langle Proof_{A_i}\rangle \; := \; UtiliseToken(A_i, Cert_{A_i}, R, y_b, f_T, \{s, u_1\}_{A_i})$$

   If the UtiliseToken function is successfully executed, then $\langle Proof_{A_i}\rangle$ will contain $A_{1_i} = g^{u_1 s}$ and $A_{2_i} = g_1^s$ (see Appendix D), such that $A_i = A_{1_i}A_{2_i}$.
3. chooses its bid value, $b \in \mathbb{Z}_q^*$;
4. seals the bid using the sealing function explained in Section 5.4.3. It chooses $a \in_R \mathbb{Z}_q^*$ and computes the following:

$$(S, s_1, s_2, c) \; := \; Sealer(u_1 s, s, b, a, g, g_1, p, q)$$

   Here (S, s_1, s_2, c) are the outputs of the sealing function and $(u_1 s, b, a, p, q)$ are the inputs (see Section 5.4.3). If $\mathcal{R}$ verifies the sealing function correctly as:

$$Pass \; \stackrel{?}{=} \; VerifySeal(S, s_1, s_2, c, A_{1_i}, A_{2_i}, g, g_1, p)$$

   where $(S, s_1, s_2, c, A_{1_i}, A_{2_i}, g, h, p)$ are the inputs to the function and the output is either pass for successful verification or fail for unsuccessful verification.

If $\mathcal{R}$ is satisfied with all the proofs in this section it signs the tuple as:

$$\sigma_{R_i} \; := \; Sign(x_r, S, s_1, s_2, c, A_{1_i}, A_{2_i}, g, g_1, \langle Proof_{A_i}\rangle)$$

Where $Sign$ is a suitable signature algorithm that signs all the inputs

$$(S, s_1, s_2, c, g, g_1, A_{1_i}, A_{2_i}, \langle Proof_{A_i}\rangle)$$

using the private key x_r. $\mathcal{R}$ then stores:

$$(S, s_1, s_2, c, g, g_1, A_{1_i}, A_{2_i}, \langle Proof_{A_i}\rangle)$$

along with $\langle Proof_L \rangle$ and σ_{R_i} in a publicly readable directory $DB_{\mathcal{R}}$, indexed by A_{1_i}.

$\mathcal{R}$ *refrains* from accepting registration *after* a specified time, end of bid registration time (EBRT).

Phase 3 (Service Delivery Phase): bid submission phase, consists of the **VerifyOpened-Seal** sub-protocol. This phase starts *after* the EBRT and finishes when the end of bid submission time (EBST) is reached, that is $\mathcal{A}$ accepts bids during the period between EBRT and EBST. This is the actual bidding phase and only those bids received in this phase are counted[7]. In this phase, $\mathcal{B}_i$ contacts the auctioneer, $\mathcal{A}$ and authenticates using its pseudonym A_{1_i}. $\mathcal{B}_i$ opens the commitment by sending (b, a) to $\mathcal{A}$. On receiving the tuple, $\mathcal{A}$:

1. obtains the registration transcripts from $DB_{\mathcal{R}}$ using A_{1_i} as the index into the database;
2. verifies the signature on the transcript by $\mathcal{R}$;
3. obtains the seal values, (S, s_1, s_2, c), from the transcript;
4. verifies the opened commitments as;

$$Pass \overset{?}{=} VerifyOpenedSeal(a, b, S, s_1, s_2, c, A_{1_i}, A_{2_i}, g, g_1, p, q)$$

   and aborts the submission process when the result is not $Pass$;
5. signs the bid tuple (b, a) along with the the seal values (S, s_1, s_2, c) as:

$$\sigma_{\mathcal{A}_i} := Sign(x_a, S, s_1, s_2, c, A_{1_i}, A_{2_i})$$

   where $Sign$ is a suitable signature function, x_a the private key and $(S, s_1, s_2, c, A_{1_i}, A_{2_i})$ that is being signed to result in the signature tuple $\sigma_{\mathcal{A}_i}$;
6. returns the signature tuple $\sigma_{\mathcal{A}_i}$ to $\mathcal{B}_i$ as a receipt of the bids;
7. stores the tuples (b, a) along with $\sigma_{\mathcal{A}_i}$ and (A_{1_i}, A_{2_i}) in a publicly readable directory $DB_{\mathcal{A}}$, indexed by b.

[7] $\mathcal{R}$ is trusted not to accept sealed bids after EBRT and $\mathcal{A}$ not to accepts opened bids after EBST. This assumption is valid because $DB_{\mathcal{R}}$ and $DB_{\mathcal{A}}$ are publicly readable and can be suitably monitored for potential breach of trust.

Announcement of Results: When the auction is terminated, the highest bid, b, is chosen from the database (which is publicly readable, thereby providing public verifiability) and $\mathcal{B}_i$ (the owner of the bid b) is announced as the winner. $\mathcal{B}_i$ identifies with $\mathcal{A}$ using the pseudonym, A_{1_i}, which is available in $DB_{\mathcal{A}}$ and avails the auctioned goods. Note that the anonymity of the winning bidder *need not* be revoked, but can be if necessary.

Anomalies: There can be two cases of anomalies that could occur:

1. the winning bidder does not claim the goods; or,

2. the auctioned goods are denied to the winning bidder.

In the first case the winning bidder does not claim the goods and thereby does not pay for the goods. In which case, $\mathcal{A}$ or any other entity can approach the trustee $\mathcal{T}$ and engage in the tracing protocol to compute the identity, I, of $\mathcal{B}_i$, possessing the pseudonym, A_{1_i}. This is computed using the tracing protocol described in Appendix D as:

$$\langle I, Proof_T \rangle \quad := \quad Trace(X, T, \langle A_i, Cert_{A_i} \rangle, \langle Proof_{A_i} \rangle, \{X_T\}_T)$$

Note that all the information required for tracing are present in $DB_{\mathcal{A}}$ and DB_R.

In the second case when the auction goods are denied to the winning bidder (due to software glitch or some other error), $\mathcal{B}_i$ can approach $\mathcal{R}$ with the receipt, $\sigma_{\mathcal{A}_i}$, that it received during the bid submission phase, identify itself using the pseudonym A_{1_i} and avail the goods or other compensations.

Analysis

The accomplishments of the protocol against the requirements stated in Section 5.4.1 will be verified in this section.

Confidentiality of bid: The confidentiality of the bid is provided by the hiding property of the sealing function, until the bid submission phase. Since, with an overwhelming probability, only the bidder can open the commitment values correctly, the scheme provides user-controlled confidentiality for the bid.

Non-repudiation of bid: This property is provided by the *non-transferability* property of the electronic cash scheme and the non-repudiation property of the sealing function. The non-transferability property of the e-cash system is important because if the bidder transfers the power to spend the coin to another entity it would have to reveal the values of $u_1 s$ and s to that entity, and u_1 is a long term secret key of the bidder's account with the mint.

Publicly verifiable auction: Since all the proof transcripts in the system are publicly verifiable, the proposed auction system possesses this property.

Anonymity of bidder: Restricted anonymity is provided to all the bidders using the e-cash system as an anonymous token issuer. Note that the anonymity of the winning bidder is also preserved.

Independence of auction rules: All the bid values, b, will reside in $DB_{\mathcal{A}}$ in *cleartext*. Any suitable auction rules can be employed to determine the winning bidder.

Comparison Based on Efficiency

This section will compare the computational requirements of the scheme proposed in Section 5.4.4, the proposals using publicly verifiable secret sharing (PVSS) [82] schemes, such as that of Franklin and Reiter [61], and the auction scheme proposed by Sakurai and Miyazaki [80]. The number of modular exponentiations by each entity for achieving confidentiality of bid were counted. The results are presented in Table 5.2.

An estimate (based on the publicly verifiable secret sharing scheme proposed by Schoenmakers [82]) of the number of modular exponentiations by each entity for achieving confidentiality of bid in schemes employing PVSS, such as that of Franklin and Reiter [61], is presented. The use of a t out of n scheme with $t = 2$ and $n = 2$, which is the simplest mode, will be assumed. The estimates are presented in Table 5.2.

The protocol proposed by Sakurai and Miyazaki [80] accomplishes anonymity of losing bidders and user controlled anonymity using undeniable signatures. The estimate assumes the following variables: L is number of bids, $J \in \{0, \cdots, L-1\}$ is the index of winning bid value, J is the index of the winning bid and B the number of winning bidders. The assumed values are: $L = 10$ and $N = 100$. The estimates are presented in Table 5.2. In the table, the best case condition occurs when $J = 0$ and the worst case condition occurs when $J = 9$.

	Bidder	**Auctioneer**	**Trustee**
The proposal[a]	20	$5N/2N$	$15N$
PVSS schemes[b]	$4n + 2t$	$4nN$	$4(n + t)N$
Sakurai *et al.*[c]	$10J + 3$ (Losing Bidder) $10J + 13$ (Winning Bidder)	$6JN + 6B$	

Table 5.2: Computational Comparison of Proposals

[a]Anonymity for winning and losing bidders.
[b]No anonymity for winning and losing bidders.
[c]Anonymity for losing bidders.

The following observations are made on Table 5.2:

1. the bidders need to perform a constant number of exponentiations in the scheme proposed in this thesis;
2. the number of exponentiations that the auctioneer needs to perform:
   (a) is linear with the number of bidders, in the scheme proposed in this thesis;
   (b) is directly proportional to the number of bidders *and* the number of trustees in PVSS schemes; and,
   (c) is directly proportional to the *product* of the number of winning bidders and number of bids and, to the number of bidders, in [80].

The proposal possesses a superior performance, in comparison with the approach based on the publicly verifiable secret sharing approach (without anonymity for losing bidders). In comparison with the scheme by Sakurai and Miyazaki, the proposal achieves the properties in a much more efficient manner with a constant number of exponentiations for the bidders.

Comparison Based on the Characteristics

This section will present the characteristics of the proposal, in comparison with the other schemes, to demonstrate its achievements. The characteristics of the proposal are:

1. the number of exponentiations is a linear function of the number of bidders;
2. it is possible to provide anonymity even to the winning bidder;
3. any type of auction rule can be employed, like highest price, lowest price or Vickery (second highest price).
4. provides user controlled anonymity;
5. any anonymity providing mechanism or sealing mechanism can be used, as long as they guarantee the required properties.
6. phases 2 and 3 permit stateless operations. That is every bidder need not have continuous connections with the auction centre. This is very useful for implementations over stateless protocols like the HTTP protocol in the WWW applications on the Internet [36]. Suitable anonymous token issuing facility can be employed to have a stateless Phase 1.

The characteristics of the schemes [61] that use publicly verifiable secret sharing are:

1. users cannot control confidentiality of their bid during the bidding process;
2. generally inefficient; and,
3. independent of the auction rules.

The characteristics of the scheme proposed by Sakurai and Miyazaki [80] are:

1. it provides user controlled confidentiality for the bid values and the bid value of the losing bidders is not revealed.
2. it requires reliable real time networks and, therefore, may not be suitable for use over the Internet;
3. it can only operate with either the highest price or the lowest price auction rules, in order to provide anonymity for losing bidders;
4. since, bidders must choose a value of the bid from a fixed set of bid values, it may not be suitable for all scenarios of auction;

5. the auction system depends critically on the state of the proceedings; and,

6. if the connection of *any single* bidder to the network is disconnected, due to some reason (accidentally or maliciously), the entire auction system will be stalled. Thereby, it is less robust.

5.4.5 Discussion

The anonymous token employed in the proposal provides restricted confidentiality and universal integrity services for the identity of the bidder. Thereby the anonymous token, AT_i, belongs to compliance Category 1, as discussed in Section 2.3.

The proposal provides universal confidentiality and integrity services to the bid so that only the bidder can reveal the choice of the bid. Therefore the bid ciphertext, (S, s_1, s_2, c), belongs to compliance Category 0, as discussed in Section 2.3. Due to the modular nature of the schema, future extensions to auction sub-protocol can accommodate restricted confidentiality service for the bid without adversely affecting the anonymity service for the bidders. The modification could allow a set of trusted third parties to open the bid without the involvement of the bidder. Such a modification would result in a bid ciphertext that belongs to compliance Category 1. The scope for customisation and refinement of the proposed auction system is broad. The proposed auction system can potentially provide anonymity service for the winning bidder, even after the end of the bidding process. Currently, a practical limitation that may be applicable is its requirement for an anonymous physical channel.

Although the problem of timing the bidding periods has been studied in the literature [87], it may not be a cryptologic problem. The solution to the problem would rather be trust-based involving a stable, accurate and trusted source for time. The cause for this reasoning can be found in the characterisation of cryptosystems in Chapter 2. The definitions for the basic services, namely confidentiality and integrity, in Section 2.1.1 accounts only for keys, messages and ciphertexts, and is independent of time.

The proposal by Sakurai and Miyazaki [80], and Sako [79] opted to provide confidentiality service only for the data. The identity of the bidder was public. A characteristic, which is not necessarily a disadvantage, of both the proposals is their inability to provide anonymity service for the winning bidder after the termination of the bidding process. The proposals required complicated interactions to determine the winning bid in a specialised manner because the bid ciphertexts must be compared before the winning bid ciphertext can be decrypted[8]. Since ciphertexts are designed to appear random, such comparisons must be carefully designed and, usually, are not extensible to many modes of auction rules. The proposal in Section 5.4.4 facilitated comparison of bids in *plaintext*. This property allows the auction system to be independent of the auction rules, such as highest bid, lowest bid and Vickrey auction. The advantage of both the proposals [80, 79] is that they do not require an anonymous physical channel.

5.5 Analysis of Electronic Voting Systems

The design of electronic voting system has been a long standing problem in cryptographic research [21, 25, 22]. The technique presented in this thesis allows a better analysis of voting systems by identifying the services (confidentiality and integrity) required for various data and the manner in which the services are provided (restricted or universal). The schema presented in Section 5.2.3 will be a useful tool for the design of an electronic voting system that requires confidentiality service for the ballot.

The primary aim of this section is to enumerate the possible approaches for the design of a complete voting system. The advantages of a voting system that employs the anonymous token paradigm presented in Section 5.2.3 will be highlighted. Finally, a conceptual sketch for the design of an electronic voting system that utilises the schema presented in Section 5.2.3 will be outlined.

5.5.1 Major Entities in a Voting System

The major entities in the voting scheme are:

1. **Voter:** This entity requires confidentiality service for its ballot as the minimum requirement from the system. It sends confidential electronic information (the ballot) to the system that must be tallied in a prescribed manner.

2. **Teller:** This entity collects and stores the ballots from the voters. It must not know the value of the vote (in the ballot) or the voter-vote relationship.

[8]The losing bid ciphertexts must not be decrypted, in order to provide confidentiality service for the bids of the losing bidders.

3. **Tallier:** This entity counts all the valid votes collected by the teller and publishes the result in a publicly readable bulletin board.

4. **Monitor:** This is an optional entity that can monitor and register any fraudulent activities in the voting system. It could be an active participant or a passive observer in the system that monitors every communication and computational results of all the participants. Usually, when all the communications are accompanied by universal verifiable proof transcripts, explicit modeling of this entity in the system will not be essential.

5.5.2 Requirement Analysis

The basic properties of any voting system that are identified in the literature [68, 8] are as follows:

Authorisation (BP1): Only authorised voter may vote.

Uniqueness (BP2): No entity may vote more than once.

Confidentiality (BP3): No entity may be able to determine the voting strategy of other voters.

Integrity (BP4): Nobody may be able to duplicate or modify votes of other voters.

Receipt-freeness (BP5): Voters may not be able to accurately prove their voting strategy after their participation in the election.

Secure e-voting systems [68] can potentially provide additional properties, which are not available in the contemporary manual voting systems. These advanced properties are as follows:

Computerisation (AP1): The voting process may take place over a computer network.

Verification of Tally (AP2): Every voter may be able to make sure that his/her vote has been taken into account.

Change of Ballot (AP3): Voters may change their ballot (change the voting strategy) within a given period of time.

Revalidation of Individual Ballot (AP4): If the voter finds that his/her vote has been misplaced, he/she may be able to prove this to the voting authority without jeopardizing ballot secrecy.

The primary concerns of this thesis are the basic properties of voting systems. At the same time, prospective extensions to accommodate the advanced properties are possible.

The data-structure that is central to the security of e-voting systems is the ballot. A ballot is a collection of votes, which correspond to individual candidates participating in the elections. The votes contain the choice or preferences of the voters. In order to guarantee properties **BP1** to **BP4**, the system may maintain a database of tuples of the form $integrity(confidentiality(I_i, B_i, K_c), K_i)$, where:

1. the function *integrity* represents the integrity service that employs a key K_i;
2. the function *confidentiality* represents the confidentiality service that employs a key K_c;
3. I_i represents the unique identity of voter i; and,
4. B_i represents the ballot formed by voter i.

There exist many possibilities to share the knowledge of the keys, K_i and K_c. The integrity service guarantees that the values of the ballot and the identity cannot be changed by any entity (possibly, excepting voter i, if the advanced properties are to be achieved). The confidentiality service guarantees secrecy of the tuple, (I_i, B_i), for voter i. Since the secrecy of the tuple can be maintained by providing universal confidentiality service to I_i or B_i, there exists two approaches to achieve this goal. The first approach provides *universal confidentiality* service only for B_i, and the second approach provides *universal confidentiality* service for I_i. The term *universal confidentiality* suggests that the value will remain secret *forever*. If such a security service is not achievable with current technology [34], then the properties of the keys used for the confidentiality service may be appropriately chosen [56] to provide secrecy for a sufficient period. For example, the period of secrecy may be comparable with the average life span of the population of voters.

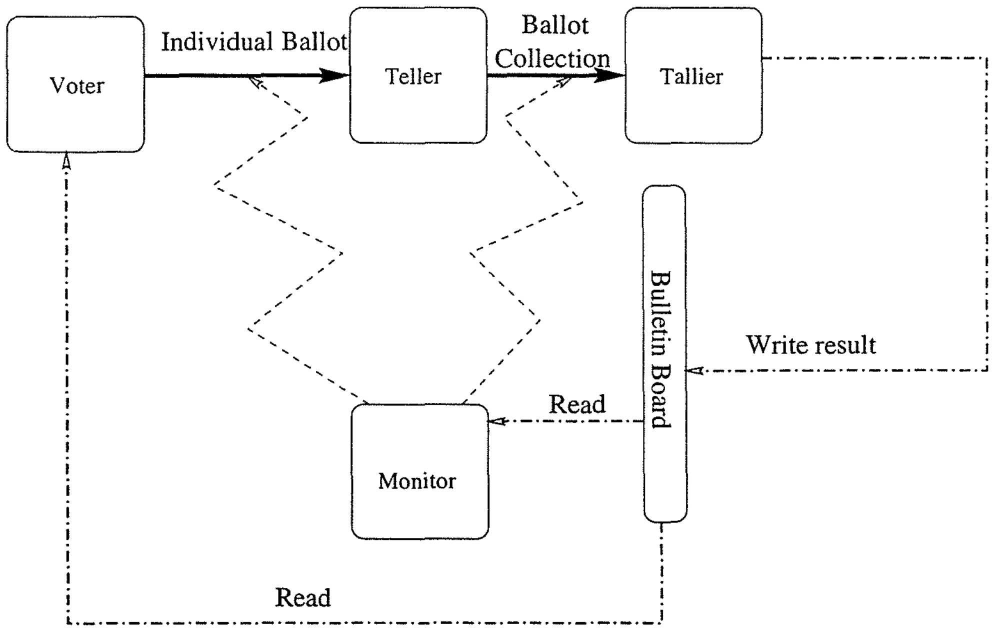

Figure 5.3: Dynamics in a Generic Electronic Voting System

The general message dynamics are shown in Figure 5.3. In the figure, the voter prepares the individual ballot in a prescribed manner and send it to the teller. The teller forwards the collection of ballots it accepted to the tallier. The tallier verifies the ballots and tallies the ballots the conform with the election rules. The result of the tally are published in a publicly readable bulletin board. An, optional, monitor may eavesdrop on the communications between the voters, the teller, the tallier and the bulletin board to verify the adherence to the election rules. Subsequent discussion in this section will discuss the advantages of designing an electronic voting system based on the anonymous token paradigm explained in Section 5.2.3, in comparison with the other approaches.

5.5.3 Techniques for Privacy of Votes

Confidentiality for the tuple of the form $(I,\ D)$, where I may represent the identity of a participant and D may represent the data formed or chosen by I, can be provided by either;

1. providing confidentiality service for D alone;

2. providing confidentiality service for I alone; or,

3. providing confidentiality service for I and D.

In the case of voting systems I is the identity of the voter and D is the vote or the ballot, depending on the design of ballots. The aim is to provide confidentiality service to the tuple, $(I,\ D)$, so that no entity other than the voter can know the value of the tuple.

Universal Confidentiality Service for the Ballot

The first approach, namely confidentiality service for the ballot, has been very popular in the literature. The PhD thesis by Benaloh [9] is a good source of information for the design of voting systems adopting this approach. In this approach *universal* confidentiality service for the ballot is essential for the proper functioning of the system. Each voter encrypts the ballot for a trusted tallier and submits the resulting ciphertext. The encryption mechanism or the election procedure may prevent the decryption of individual ciphertexts. Otherwise, the encryption mechanism would result in the revocation of confidentiality service from the tuples, which would effectively contradict the aim of the voting system.

Many schemes [21, 25, 22] have been devised that allow the tallier to tally the individual ballot *ciphertexts* and decrypt the resulting ciphertext to obtain the tally of the individual ballot plaintexts. Benaloh [9] proposed a technique called homomorphic encryption for the design of voting systems. The homomorphic encryption technique uses an encryption function f_1 and decryption function f_2 such that:

$$f_1(a) \oplus f_1(b) = f_1(a \otimes b)$$

and;

$$f_2(f_1(a \otimes b)) = c$$

such that $c = a \otimes b$, where $\oplus$ and $\otimes$ are suitable operators, a and b are individual ballot plaintexts, and c is the tally of ballot plaintexts. In order to provide confidentiality service for the voter-ballot relationship, no entity, other than voter, must be able to determine the value of a corresponding to the ciphertext $f_1(a)$. This requirement is important because the teller in the election system may register information of the form $\{(I_a, f_1(a)), (I_b, f_1(b)), \cdots\}$, where I_x is the identity of the voter corresponding to the ballot ciphertext $f_1(x)$, to achieve Requirements **BP1** and **BP2**.

Generally, there may be many candidates participating in an election and every *valid ballot* that selects the winning candidate must conform with an authorised format. For example, suppose that there are two candidates (generalisation to more than two candidates is straight forward) and every valid ballot must elect *only one* of the candidates. In order to design such a rule, a popular approach is the abstraction of the ballot of the form: $\mathcal{B}_a = (f_1(a_1), f_1(a_2))$, where a_i is the vote for the i^{th} candidate. That is two vote ciphertexts per voter, corresponding to each candidate, must be allowed. For simplicity, suppose that $a_i \in \{0, 1\}$, where 0 represents the rejection vote and 1 represents the acceptance vote. Then, in order to prove that $\mathcal{B}_a$ is a valid ballot, voter I_a must be able to prove two statements:

1. every a_i is either 0 or 1; *and,*

2. only one of the values of a_i is 1 (or 0) without revealing the index i corresponding to the choice.

Most proposals for ballots of this form, such as that of Cramer *et al.* [25, 22], do not achieve the second proof. They focus only on the first proof. At this stage there seems to be no proposal to achieve simultaneously both the proofs. Moreover, proposals for ballots of this form become tediously complicated when there are more candidates or when the ballot is more complicated, such as those ballots that must accommodate preference information.

Any successful proposal for providing universal confidentiality service for the ballots may be required to model a *null* ballot that will allow voters to reject all the candidates. This is essential for those elections where participation is mandatory, such as in Australia, and the voters do not wish to vote for any of the candidates.

This approach may not be a viable technique for large scale voting systems because the size of the ballot increases rapidly with the number of candidates and the complexity of valid preference statements.

Restricted Confidentiality Service for the Ballot

This section presents an alternate approach for the provision of the universal confidentiality service for the voter-ballot relationship. In this approach universal confidentiality service is provided to the identity of the voter (instead of the ballot). Therefore, this approach can accommodate restricted confidentiality service for individual ballots, which would result in a validation process for individual ballot *plaintexts*. Note that this was not the case with the approach outlined in the previous section, where validation of ballot *ciphertexts* was required. Therefore, this approach is more suitable for electronic systems that could replace existing large-scale, manual voting systems.

All proposals that adopt this approach will employ a suitable form of anonymous token system and a schema similar to that presented in Section 5.2.3. Such proposals will, invariably, provide restricted confidentiality service for the ballot and universal confidentiality service for the identity of the voter. The anonymous token, which provides universal confidentiality and integrity services to the identity of the voter, would belong to compliance Category 0. The ballot ciphertext would provide restricted confidentiality and universal integrity services. Therefore, it will belong to compliance Category 1.

The following procedure can be noticed in popular manual voting systems:

VStep 1 voters *identify* (authenticate) themselves to a voting authority in a suitable fashion;

VStep 2 the voting authority verifies the identity of the voters against a ledger containing the official list of voters. If the entry corresponding to the voter is not already marked, then the authority marks the entry, else it prevents the voter from proceeding to the next step;

VStep 3 the voting authority issues a *single, unmarked, uniform* ballot[9] to the voter;

VStep 4 the voter registers the choice in the ballot in a voting area that provides *physical confidentiality* service for the choice;

[9]The ballot must not, in any manner, uniquely identify the voter. This step provides universal confidentiality service for the identity of the voter during the tallying phase of the election.

VStep 5 the voter submits the ballot to a ballot-box, which intrinsically *mixes* all the votes;

VStep 6 the ballot-box provides *physical integrity* to all the ballots until the event of its opening;

VStep 7 the tallier(s) *revoke* the confidentiality service from the ballots by opening the ballot-box and *verify* the validity of the ballot; and,

VStep 8 the tallier(s) create a tally of the votes in the valid ballots and publish the results in an appropriate manner.

The identification procedure in **VSteps 1** and **2** guarantee Properties **BP1** and **BP2**. The anonymity service provided in **VStep 3** provide *universal confidentiality* service for the identity of the voter, in that even an honest voter[10] cannot identify the ballot containing his/her choice after this step. This property of manual voting systems is popularly known as receipt freeness. **VSteps 4** through **8** achieve the remaining properties of the voting system.

The schema presented in Section 5.2.3 provides a tool for ballots that offer restricted confidentiality service for the vote. This aim can be achieved by providing universal confidentiality service[11] for the identity of the voter.

5.5.4 A Conceptual Design for a Basic E-Voting System

The first step in the automation of large scale elections would be to automate individual phases of the elections. The conceptual design to be presented in this section introduces a mechanism that would allow the users to electronically register their participation, say over the Internet. The voting phase must take place within a polling booth to achieve the property of receipt-freeness. It will employ the schema presented in Section 5.2.3 along with a suitable anonymous token system (ATS, see Section 5.1).

Choice for ATS

The ATS must provide universal confidentiality service for the identity of the voters (participants). Therefore, the the ATS *must not* provide the trace functionality, which

[10] A voter who does not place identification marks on the ballot.
[11] The ATS employed must provide universal confidentiality service for the identity.

provides the anonymity revocation service. A suitable choice could be the Brands e-cash proposal [12] that employs the concept of wallets with observers [18]. This scheme is an ideal choice because of the following properties of the Brands proposal:

Non-transferability : This property assumes the existence of a private-key, corresponding to the certified long-term public-key, of the owner (voter) such that the owner (voter), in its own interest, will not risk the exposure of the private-key. This property prevents an owner (voter) from obtaining a e-coin (token) and transferring it to another entity (proxy). If the owner (voter) does transfer the secret corresponding to the e-coin (token) then it risks the exposure of its private key, which would be against its interests.

Observer paradigm : There exists an observer, which is a physical device, corresponding to each owner that *prevents* the use of the token more than once (double-spending). At the same time, the paradigm guarantees that the observer cannot, even in collusion with the bank (TIA) that issued the e-coin (token), undermine the privacy (anonymity) of the owner (voter).

The Brands proposal provides an anonymity revocation (restricted confidentiality service for the identity of the voter) mechanism when the e-coin (token) is double-spent. Since the observer paradigm prevents double-spending this threat to voter privacy is avoided.

Since the proposal by Frankel, Tsiounis and Yung, (FTY scheme) detailed in Appendix D, is exactly the same as that of the Brands proposal *excluding the tracing operations*, the discussions in the rest of this section will employ the function definitions explained in Section 5.1 and Appendix D. It is important to note that the *Trace* function explained in Appendix D.2 will not exist, and the owners (voters) will not enable tracing while employing the *UtiliseToken* function.

System Settings

The system consists of sets of Voters $\mathcal{V}$, authorisation authorities $\mathcal{A}$, tellers $\mathcal{B}$, and talliers $\mathcal{T}$. $\mathcal{A}$ chooses primes p and q such that $p - 1 = \delta + k$ for a specified constant δ, and $p = \gamma q + 1$ for a small integer γ. A unique subgroup $\mathcal{G}_q$ of prime order q of the multiplicative group $\mathcal{Z}_p$ and generators g, g_1, g_2 of $\mathcal{G}_q$ are defined. The secret key of $\mathcal{A}$, $X_A \in_R \mathbb{Z}_q$ is created. $\mathcal{T}$ chooses a private-key as $X_T \in_R \mathbb{Z}_q$ and computes the public key $y_T = g_2^{X_T}$. $\mathcal{B}$ chooses a private key $x_B \in_R \mathbb{Z}_q$ and computes its public keys as $h_B = g^{X_B}$, $h_{B1} = g_1^{X_B}$, $h_{B2} = g_2^{X_B}$. Henceforth the set of keys for $\mathcal{A}$ will be represented as $y_A = \{h, h_1, h_2\}$, and that of be $\mathcal{B}$ as $y_B = \{h_B, h_{B1}, h_{B2}\}$. $\mathcal{A}$ securely publishes p, q, g, g_1, g_2, a secure hash function $\mathcal{H}$, its public keys y_A, the public keys of $\mathcal{B}$, y_B, and the public key of $\mathcal{T}$, y_T.

$\mathcal{A}$ associates every voter $v_i \in \mathcal{V}$ with the identity $I_i = g_1^{x_i}$, where $x_i \in \mathcal{G}_q$ must be securely generated by the voter such that $g_1^{x_i} g_2 \neq 1$. The secret-key of the voter, x_i, must satisfy the assumption for non-transferability mentioned in the previous section. This could be guaranteed if the private key x_i provides access to the health information of the voter or is a part of the voter's electronic passport. The voter must prove the knowledge of discrete logarithm of I with respect to g_1.

$\mathcal{A}$ will assume the role of the TIA, $\mathcal{B}$ will assume the role of the TAA and $\mathcal{V}$ will play the assume of the clients, as in the schema in Section 5.2.3. The system dynamics for the resulting voting system will be as shown in Figure 5.4. In the figure protocols belonging to the ATS are: *IssueToken, UtiliseToken* and *SubmitToken.* The protocols belonging to the basic voting system are *SubmitConfBallot*, which allows the voters to encrypt their ballot for $\mathcal{T}$, and *SendConfBallot*, which allows $\mathcal{B}$ to forward the ballots it collected to $\mathcal{T}$ for tallying.

Trust Assumptions

The following assumption about the voting authority, $\mathcal{A}$, is essential:

Authorisation and Uniqueness: $\mathcal{A}$ will issue a single token to every authorised voter.

The assumptions about the teller, $\mathcal{B}$, are:

Fairness: $\mathcal{B}$ will present the available choices of candidates to the voters during the ballot submission phase;

Enforcement: $\mathcal{B}$ will prevent the voter's electronic agent from communicating with other entities during the ballot submission phase;

Communication: $\mathcal{B}$ will send the ballots it collected only to $\mathcal{T}$.

The following assumptions about the talliers, $\mathcal{T}$, are essential:

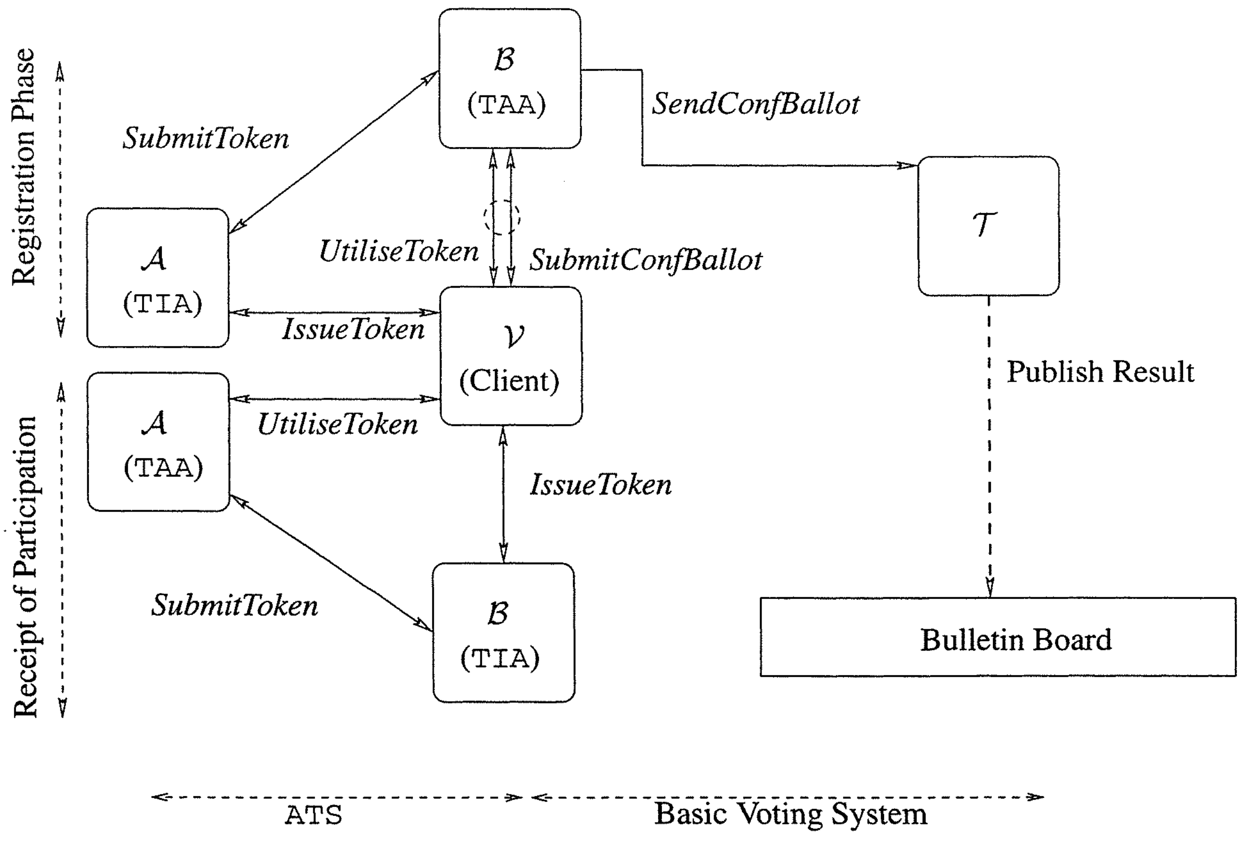

Figure 5.4: System Dynamics of a Basic Voting System

Receipt-freeness: $\mathcal{T}$ will not assist any entity to verify the integrity of individual ballots;

Fairness: $\mathcal{T}$ will not alter or modify the ballots it receives, will tally every valid ballot and publish the result of the tally.

Although these trust assumption do not significantly depart from the trust assumptions of the manual voting systems, future research may provide avenues to improve the situation in electronic voting systems. Such improvements should result in more flexible trust assumptions.

Registration Phase

This phase can be executed over an open network or in a registration booth some days before the start of the election. This phase consists of the following steps:

1. voter $v_i \in \mathcal{V}$ with a public-key $I_i = g_1^{x_i}$ contacts $\mathcal{A}$ and authenticates by employing its public key, I_i;

2. $\mathcal{A}$ consults with an electronic ledger to check if v_i has already participated in this phase. If v_i has not participated in this phase it proceeds to the next step;

3. v_i engages in the token issuing protocol (see Section 5.2.3 and Appendix D.2) with $\mathcal{A}$, to obtain an anonymous token AT_i, as follows:

$$\langle AT_i, Cert_{AT_i} \rangle_{I_i} \quad := \quad IssueToken(I_i, \mathcal{A}, \{s\}_{I_i}, \{X_A\}_{\mathcal{A}})$$

   The anonymous token AT_i contains two pseudonyms of the form $g_1^{x_1 s}$ and g_2^s (see Appendix D.2), where $s \in_R \mathbb{Z}_q$ is an output of the token issuing protocol known only to the voter;

4. $\mathcal{A}$ records the participation of v_i in the electronic ledger.

At the end of this phase, v_i must possess an anonymous token AT_i that will grant access to an electronic ballot and y_T the public key of $\mathcal{T}$, which can be obtained from $\mathcal{A}$ in a secure fashion.

This phase is represented as *IssueToken* in Figure 5.4. It achieves VSteps 1 and 2 of the election protocol, as explained in Section 5.5.3.

Ballot Submission Phase

Restricted confidentiality and universal integrity service for the ballot is essential. The confidentiality of the ballot is essential because only $\mathcal{T}$ must be able to verify the validity of the ballot. This is also essential to prevent the announcement of election results before the completion of the elections in all the election booths. The universal integrity service ensures that the ballot communicated by $\mathcal{V}$ to $\mathcal{T}$ through $\mathcal{B}$ is not altered.

In order to participate during this phase the voter, v_i, is expected to visit a polling booth along with the anonymous token AT_i (say, in a smart card) it obtained during the registration phase. The polling booth must contain a polling machine representing the teller, $\mathcal{B}$. The voter must possess a hand held device to perform the cryptologic computations. The hand held device may be provided by authorised the election officials before entering the polling booth. It can also be a third generation mobile telephone that is owned by the voter, and therefore trusted. If this is the case, the officials must make sure that the telephone can communicate only with $\mathcal{B}$ until the voter completes this phase. v_i must not authenticate to $\mathcal{B}$ with its original public-key, I_i.

This phase consists of the following steps:

1. v_i submits its anonymous token AT_i with $\mathcal{B}$ and engages in the token utilisation protocol (see Section 5.2.3 and Appendix D.2), as follows:

   $$\langle Proof_{AT_i} \rangle \quad := \quad UtiliseToken(AT_i,\, Cert_{AT_i}, \mathcal{B}, y_A, null, \{s, u_1\}_{AT_i})$$

   where $null$ symbolises the absence of trustees for tracing purposes and y_A is the set of the public keys of $\mathcal{A}$, which is required to verify the certificate, $Cert_{AT_i}$, for the anonymous token, AT_i. If the token utilisation protocol is successfully executed, $\mathcal{B}$ stores $Proof_{AT_i}$ in a private database and allows v_i to proceed to the next step;

2. $\mathcal{B}$ sends the choice of entities to v_i;

3. v_i computes the ballot $b_i \in \mathcal{G}_q$ in a prescribed fashion to represent its voting strategy;

4. v_i chooses $k \in_R \mathcal{G}_q$ and encrypts the ballot for $\mathcal{T}$ as ($e_i = b_i y_T^k$, $f_i = g_2^k$, $c_i = \mathcal{H}(e_i, f_i)$, $r_i = k - c_i s \bmod q$), where g_2^s is the authorised pseudonym of the voter available in AT_i;

5. v_i sends the tuple (e_i, c_i, r_i) to $\mathcal{B}$.

6. $\mathcal{B}$ sends (AT_i, e_i, c_i, r_i) to $\mathcal{T}$ in a confidential channel.

7. $\mathcal{T}$ stores the tuples in a private, secure database.

If proof of participation in this phase is essential (such as in Australia where participation in elections is mandatory), then v_i engages in the token issuing protocol with $\mathcal{B}$ to obtain an anonymous token as a receipt of participation. The interactions are as follows:

$$\langle RT_i,\, Cert_{RT_i} \rangle_{g_1^{u_i s}} \quad := \quad IssueToken(g_1^{u_i s}, \mathcal{B}, \{s_r\}_{g_1^{u_i s}}, \{X_B\}_{\mathcal{B}})$$

In above protocol, $\mathcal{B}$ performs a restrictive blind signature [12] (see Appendix D for details) on the pseudonym tuple of the form $(g_1^{u_i s s_r}, g_2^{s_r})$, where $s_r \in_R \mathbb{Z}_q$ is a randomly chosen value by v_i. The interpretation of the syntax is available in Appendix D.2, which is the same as explained in the registration phase. In order to prove to $\mathcal{A}$ about its participation in this phase, v_i authenticates to $\mathcal{A}$ and performs the token utilisation protocol by employing RT_i as follows:

$$\langle Proof_{RT_i} \rangle \;:=\; UtiliseToken(RT_i, Cert_{RT_i}, \mathcal{A}, y_B, null, \{s_r, su_1\}_{RT_i})$$

where $null$ symbolises the absence of trustees for tracing purposes and y_B is the set of the public keys of $\mathcal{B}$, which is required to verify the certificate, $Cert_{RT_i}$, for the anonymous token, RT_i. This use of ATS is possible because:

1. the ATS in the registration phase provides confidentiality to tuples of the form (I_i, AT_i) as explained in Section 5.1 by Properties **FP2** and **FP3**; and,

2. the ATS in this phase provides confidentiality to tuples of the form (AT_i, RT_i) as explained in Section 5.1 by Properties **FP2** and **FP3**.

At the end of the elections $\mathcal{A}$ will have a list of information of the form (I_i, RT_i) and $\mathcal{B}$ will have the list of all AT_is it had accepted. If the ATS achieves properties **FP1** an **FP2**, the values (I_i, RT_i) and AT_i cannot be correlated when there are many voters who wish to maintain their privacy.

This phase achieves **VSteps** 3, 4 and 6 described in Section 5.5.3. It does not achieve **VStep** 5 because the individual ballots are not mixed. This is because each ballot tuple,$(AT_i,\ e_i,\ c_i,\ r_i)$, is different, with a high probability — the anonymous tokens AT_i must be unique in order to achieve Property **FP0** and **FP1** described in Section 5.1. It achieves the essential goal (privacy of vote) of **VStep** 5 by arguing the intractability to determine the identity of a voter, I_i, who owns the token AT_i. Moreover, it provides an end-to-end integrity service between the voters and $\mathcal{T}$, which was not the case in manual voting systems. In manual voting systems, every link of communication must be physically secure to guarantee the integrity of the ballots.

Tallying Phase

The ballot tuples sent by $\mathcal{B}$ to $\mathcal{T}$ during the ballot submission phase will be of the form, $(AT_i,\ e_i,\ c_i,\ r_i)$. For each ballot tuple, $\mathcal{T}$ performs the following steps:

1. retrieves the pseudonyms, $(g_1^{u_i s}, g_2^s)$ from AT_i;
2. calculates, $f_i = g_2^{r_i}(g_2^s)^{c_i}$;
3. checks if, $c_i \stackrel{?}{=} \mathcal{H}(e_i, f_i)$;
4. decrypts ballot as, $b_i = e_i / f_i^{X_T}$, where X_T is the private key of $\mathcal{T}$ corresponding to the public key y_T;
5. decodes b_i and determines the individual votes and the validity of the ballot;
6. tallies the ballot, if it is valid.

On completion of the tallying process, $\mathcal{T}$ publishes the result of the tally in a publicly readable bulletin board. Note that the ballot tuples, (AT_i, e_i, c_i, r_i), must not be published to achieve receipt-freeness. This phase achieves **VSteps** 7 and 8.

Security Analysis

The achievements of the above proposal is presented by comparing them with the basic services mentioned in Section 5.5.2, as follows:

Authorisation (BP1): If $\mathcal{A}$ is trusted to issue valid anonymous tokens only to authorised voters and the electronic cash system presented by Brands (see Appendix D) prevents entities from forging the signature of the bank ($\mathcal{A}$ in our proposal) then only authorised voters can participate in the ballot submission phase. That is the ATS used possesses the authorisation property described in Section 5.1.

Uniqueness (BP2): If the proposal by Brands prevents double-spending of e-coins then the proposal prevents voters from voting more than once and remain undetected. That is the ATS used possesses the authorisation and reusability properties described in Section 5.1. The system must only allow a single use of the token. The reusability property can be *enforced* by the use of the observer paradigm described in Section 5.1.

Confidentiality (BP3): If the ATS (Brands e-cash proposal) used possesses the anonymity property presented in Section 5.1 and the voters use suitable anonymous physical channels then the proposal preserves the confidentiality service for the voter-vote (or voter-ballot) relationship.

Integrity (BP4): If the Schnorr signature scheme [81] employed in the *SubmitConf-Ballot* protocol and the tallying phase is secure (unforgeable) then the ballot submitted by the voter cannot be altered without the knowledge of private key, (u_i, s), of the voter, v_i.

Receipt-freeness (BP5): If the tallier, $\mathcal{T}$, is trusted not to publish the ballot-token tuples sent by $\mathcal{B}$, $\mathcal{B}$ does not forward the tuples to any entity other than $\mathcal{T}$ and the ATS possesses the non-transferability property presented in Section 5.1, then the proposed protocol prevents the voters from proving the content of their ballot to other entities. Note that such trust assumptions were generically categorised as "untappable channels" and "strong physical security" by Okamoto [69].

The directed, simple trust assumptions, modular nature of the framework and the proposed design will be ideally suited for future design and implementation developments. The security analysis of the resulting systems will also be simple to comprehend as shown above.

5.5.5 Comments on Electronic Voting Systems

The design of electronic voting systems is a specialised topic that must deal with many technical and sociological considerations. The literature contains only partial voting systems that do not address all the important requirements. Most proposals address the properties for specialised voting systems that cannot be widely used.

This section presented a basic electronic voting system to automate the important aspects of modern elections. In its present form it is not suitable for deployment over open networks if achievement of receipt-freeness is essential.

A concept called *deniable encryption* was proposed by Canetti, Dwork, Naor and Ostrovsky [16]. This concept allows an entity to encrypt a message m in the ciphertext c under the public key key of a receiver. The important property of this scheme is the ability of the sender to prove the encrypted message to be m or m', depending on its choice. Canetti *et al.* have already identified the usefulness of such an encryption scheme in voting systems to achieve receipt-freeness. This is because the scheme allows the voters to exercise their choice during the ballot submission phase and still pretend that they have cast the vote as agreed with some entity, who could possibly be coercive.

Another important aspect of electronic voting systems would be key sizes. Since the privacy service provided by election systems must, at least, range over the average life expectancy of the citizens of a country, the choice for the size of the public keys of $\mathcal{T}$, especially, is critical. Although there exists no scientific method to determine the size of the keys as a function of the period of privacy requirement, there do exist empirical estimates by Lenstra and Verheul [56].

Real life voting systems demand solutions to many important issues, such as receipt-freeness [69]. Receipt-freeness is the property by which the voting officials can be convinced that the voters cannot prove their choice to other entities. This is essential to prevent vote-buying. **VSteps** 3 and 5 of the contemporary voting system perfectly achieves this property assuming honest voters. It robustly achieves this property assuming dishonest voters and honest election officials, if the ballots are physically separated from individual votes. Currently, it seems difficult to achieve receipt-freeness in electronic voting systems that could operate over unprotected, open networks, such as the Internet.

A more important aspect is that the manual voting systems provide universal confidentiality service *forever*. This is because the physical security plays a major role in such a guarantee. Since the confidentiality service provided by every known cryptographic algorithm *decays* with time [34] and with every improvement in the cryptanalytic technology, electronic voting systems employing contemporary cryptographic algorithms *cannot* provide universal confidentiality *forever*. This aspect of the technology may allow a powerful adversary, such as government owned intelligence services, to record various ciphertexts during the election period and engage in cryptanalytic techniques that may take some years. This problem can be solved by devising algorithms for *information theoretic* security services, as opposed to *complexity theoretic* security services. Such algorithms become fundamentally essential when electronic voting systems are employed for nation-wide elections. Almost all known public-key technologies provide only complexity theoretic security services.

The requirements for electronic voting systems have their epicenter in sociological considerations and any technological solution must be able to address them in an uncompromising manner. This is a challenging form of compliant cryptologic system that requires a robust solution to bridge the requirements for the identification of voters (to achieve **BP1** and **BP2**) and, confidentiality and integrity service for individual ballots (to achieve **BP3** and **BP4**). Any robust solution for such systems must strive to provide information theoretic security that should withstand decades of cryptanalytic attacks. The number of years of security provided by such algorithms must *at least* be greater than the average-life span of the population that uses the voting system.

5.6 Summary

The analysis of cryptologic systems in terms of the basic services, namely confidentiality and integrity, and the manner of provision of these services, restricted and universal, assisted in a simple analysis of a class of protocol systems called secure selection protocols. A generic schema was presented that facilitated identical proposals for two seemingly different protocol applications, namely peer-review system and electronic auction system.

The conflicts in the interest of various participants were interestingly similar when expressed in terms of the cryptologic objectives. When dealing with systems that provide restricted confidentiality and universal integrity services to tuple of the form (I, D), it was realised that the approach for providing confidentiality service to the identity, I, rather than to the data, D, resulted in a more comprehensible system. The restricted nature of the confidentiality service was modeled into the ATS, which allowed the selection protocol to be independent in providing the integrity service. Therefore, the anonymous tokens issued using the ATS belonged either to compliance Category 1 when restricted anonymity (confidentiality) service was provided and compliance Category 0 when universal anonymity service was of interest.

Further relationship between the selection subsystem and other compliant system was evident especially in the electronic auction system in Section 5.4. The proposal in this section provided user controlled confidentiality service for the bidders. The corresponding ciphertexts ($S = g^b g_1^a$) provided universal confidentiality service to the bid. Therefore these ciphertexts belonged to compliance Category 0. It may be possible to extend the scheme to provide a trustee controlled confidentiality service by incorporating a suitable key recovery technique (such as those in Chapter 4). The resulting ciphertexts would then belong to compliance Category 1, as they would provide restricted confidentiality service. It is important to realise that the mode of provision of the confidentiality service (for the data or selection) in this model does not affect the mode of provision of the anonymity service (confidentiality service for the identity).

The approach facilitated a modular design technique for the synthesis of protocols and reuse of existing protocols. When perfectly designed, the coupling of the sub-systems could be such that individual sub-systems may be replaced by functionally identical systems. For example, an efficient technology for the ATS is available, it may be possible to *patch* the security software by replacing the code for the older ATS with the code for the latest ATS. Modularity in protocol design is a relatively new field of study and the schema in Section 5.2.3 may provide useful information for research in that direction.

Chapter 6

Summary and Research Directions

How nature loves the incomplete. She knows: if she drew a conclusion it would finish her.
- CHRISTOPHER FRY, 1950

Talent jogs to conclusions to which Genius takes giant leaps.
- EDWIN PERCY WHIPPLE (1819 - 1886)

The fundamental nature of confidentiality and integrity in cryptologic protocols was established. The resulting visualisation of cryptologic systems provides simple conceptualisation, analysis and design techniques. These results facilitate a simple, *cryptologic definition* for the term *compliance*, which has been previously stated only in a non-cryptologic language for specialised situations such as legal wiretapping of encrypted communications [40]. The thesis identifies compliance to be a broader cryptologic phenomenon, encompassing the entire class of cryptologic protocols. This identification allows for the correlation of seemingly disparate protocols such as fair electronic cash [38] and fraud detectable key recovery [89].

Various protocols that exhibit a particular form of compliance guarantee were analysed and designed. This was achieved by using the threads of reasoning established earlier in the thesis. The thesis concentrated on key recovery protocols and on a class of protocols called secure selection protocols (SSP). A new paradigm called hybrid key recovery proposed to achieve robust key recovery. A design schema for SSP was proposed employing a concept called anonymous token systems ATS. The analysis of protocols in this thesis is explained in terms of the basic services, namely the confientiality and the integrity services. Such an explanation highlights the new analysis and design philosophy that the thesis propounds.

Section 6.1 will present a chapter-wise summary of the thesis, in order to highlight its contributions. Section 6.2 will present potential research directions that were identified during the course of the research and Section 6.3 will conclude the section.

6.1 Summary of the Chapters

In Chapter 2, the thesis presented a simple and intuitive representation of cryptologic systems, and employed the technique to analyse and design the class of protocols called compliant cryptologic protocols. It was realised that the definition of compliance in cryptosystems is broad and covered many areas of cryptologic protocols. Thereby, a classification of message formats used in protocols was used to characterise cryptologic systems. The classification was employed to restrict the scope of the thesis to those cryptologic protocols that employ message formats that provide *restricted* confidentiality and *universal* integrity services to various messages.

The term compliance was defined as a guarantee by the system to its participants. The guarantee was either restricted or universal service for specified messages. The elements in the fundamental set of services that encompasses all possible services required for cryptologic protocol construction was enumerated as confidentiality and integrity. Although, without loss of generality, these services were assumed to be independent of each other, there exists a fine relationship between them. For example, in order to achieve robust integrity services, protocols must establish confidential keys. Suppose Alice and Bob require to communicate confidently without Carol being able to modify the messages. Carol can perform every logical operation that Alice and Bob can perform, if she can modify the messages without the knowledge of Alice and Bob. Since cryptography is modeled on the secrecy of keys, Alice and Bob can gain reasonable confidence about the inability of Carol if they possess a secret that Carol does not.

The importance of publicly verifiable encryption techniques, in particular, for the design of restricted confidentiality and universal integrity services was highlighted. Various classes of publicly verifiable encryption techniques were enumerated.

In Chapter 3, a technique for the analysis of the integrity goals of various protocols was presented. The foundation for the technique was based on the content established previously by the thesis. An informal syntax for the technique was created and employed in the analysis of an encryption scheme meant to be secure against the adaptive chosen ciphertext attack, an efficient electronic cash system and a fraud detecting key recovery proposal. The analysis assisted in the development of precise understanding of the goals for these systems, which resulted in either the development of an alternate proposal or the identification of drawbacks.

In Chapter 4, the popular types of key recovery were analysed, which resulted in the proposal of a new paradigm for the design of key recovery systems. The paradigm was called hybrid key recovery. The analyses in this chapter employed heavily the concepts in Chapter 2. The chapter highlighted the inherent problems associated with private key recovery and session key recovery systems. Furthermore, it explained the reason for the robustness of the hybrid key recovery systems against the problems faced by session key and private key recovery systems. In order to provide source traceability in the hybrid key recovery proposal, a new signature scheme called the joint signature scheme was developed. The joint signature scheme provided the control for an authority over the use of a public key in signature processes by the participants. It was argued the resulting system with hybrid key recovery and source traceability emulated closely the properties of the Clipper proposal in a better fashion, and was the only proposal suitable for robust software implementation that is available currently.

In Chapter 5, a specialised form of confidentiality service, called anonymity service, was studied. The concepts developed in Chapter 2 were employed to study and analyse the properties of existing electronic cash proposals. An abstraction called anonymous token system (ATS) was detailed and the usefulness of the electronic cash proposals for the design of ATS was demonstrated. The abstraction was employed to propose a generic schema for the design of a class of protocols called secure selection protocols (SSP). SSP deals with tuples of the form (I, D), where I represents the identity of the participant and D the data preferred by the participant, which would be its selection. The goal of SSP was the provision of confidentiality service for the relationship between I and D, and integrity service for the selection, (I, D). The schema was employed to design a peer-review system and an electronic auction system. The modularity of the schema was evident in the resulting system designs, which provided a functional independence for the sub-system that provides the confidentiality service for the relationship and the sub-system that provides the integrity service to the tuple. The concepts developed in Chapter 2 were used to analyse the requirements for the design of electronic voting systems. It was argued that the simplest and most effective approach for the design of a large scale, electronic voting system would be the use of the proposed schema for the design of SSP. The deficiency in current cryptographic knowledge available in the open literature for the design of electronic voting system was also detailed.

6.2 Research Directions

Detailed classification of cryptologic protocols: A precise classification of cryptologic protocols is possible based on the mode of compliance employed. Such a classification would readily yield an understanding of the manner in which various cryptologic services to the messages employing keys interact to achieve a complex goal. This goal seems to be an ideal first extension to this thesis.

Formal syntax for the representation of cryptologic protocol goals: The integrity verification technique presented in Chapter 3 employed a rather informal syntax, which by its potential was extremely successful in accomplishing its goals. Ideas from this chapter will assist greatly in the development of a formal syntax for the representation of cryptologic goals. Although, research for the representation of the confidentiality goal has commenced [1], the research for the representation of the integrity goal requires more input.

Joint signature scheme: The joint signature scheme that was proposed in Section 4.5.3 achieved adequately and robustly the requirements for source traceability in the hybrid key recovery system. The variation of the global signature verification equation with variable number of participants in the signature generation protocol may be a potential drawback in some applications. It may be useful to research for joint signature schemes that do not have this property.

The relation between joint signature schemes and proxy signature schemes [60] may be a productive area for research.

Alternate proposal for the design of anonymous token systems: Currently blind signature schemes [18] are the only known technique for the design of anonymous token systems (ATS, see Chapter 5). Intuitively, the goals of an ATS can be achieved using suitable verifiable encryption of signature tuples. Although there are results in the literature dealing with verifiable encryption of signatures [41, 4], they are rudimentary and do not yield to the goals of anonymity service because these schemes provide confidentiality service only to parts of the signature tuple. In order to design robust anonymity services, it must be possible to encrypt the entire signature tuple, namely as explained in Section 5.2.1.

An approach may be to employ a probabilistic encryption algorithm [43] along with a robust signature scheme. The steps involved during the token issuing protocol would be:

1. the client commits to a randomiser r as $C = commit(r)$, where *commit* is a suitable commitment scheme;

2. the TIA could produce the token by performing the following signature, $\sigma = Sign(f(C))$, where f is a suitable randomising function that could additionally embed the identity of the customer for tracing purposes, and returns σ to the client;

3. the client could generate the anonymous token from σ by performing the encryption function as, $AT = ProbEnc(\sigma, r)$.

During the token utilisation protocol, the client would have to prove its knowledge of the signature of TIA, σ, and and the randomiser, r, to the TAA in minimal or zero knowledge by revealing the value of AT. Note that *ProbEnc* must completely hide σ and at the same time allow the TAA to verify its structure. That is TAA must verify robustly that the hidden σ is indeed a signature of the TIA.

Concrete proposal for electronic voting system: A concrete proposal for the design of a robust electronic voting system can be made by employing the generic schema presented in Section 5.2.3. Due to the highly sensitive requirements of large-scale, practical election systems, it may be useful to design an ATS and a basic tallying subsystem that is information theoretically secure. Such a system will emulate closely the existing manual voting systems.

6.3 Conclusion

The thesis analysed and designed many cryptologic protocols by viewing them to be a composition of confidentiality and integrity services. It is possible to categorise message formats in the cryptologic systems based on the manner in which the basic services are rendered, namely restricted or universal service.

The notion of compliance in cryptologic systems was formalised and a classification of message formats in such systems was proposed. This classification will be very useful in the characterisation, analysis and design of many cryptologic protocols.

The development of a successful, informal syntax for the representation of the integrity goal suggests the existence of a formal syntax. It may be possible to develop a common syntax for the representation of cryptologic protocols that specifies the confidentiality and integrity services provided to various messages being communicated within the cryptologic system.

The hybrid key recovery system demonstrated the existence of a robust software key recovery system that could emulate closely the achievements of the Clipper proposal in a more efficient, scalable and secure fashion.

The approach to solve many protocols belonging to the class of secure selection protocols based on a common framework provides evidence for the existence of strong relationships between protocols that are seemingly different.

An effort to render various designs of cryptologic protocols to be modular was successful. It may be possible to design cryptologic protocols in a modular manner, which will emulate closely the developments in the techniques for computer software design. A formal syntax for the representation of cryptologic protocols would be the first step in this direction.

Appendix A

Honest Verifier Zero-Knowledge Proof

There are many ways of proving the knowledge of a secret. In the traditional password-based systems, this was achieved by revealing the secret to the verifier. Such a system would be efficient and useful only in trusted environments, such as in military operations with a well established chain of command. If such a trust relationship does not exist then alternate mechanisms are essential. Zero-knowledge proof (ZPK) technique is one such mechanism. The concept was first proposed by Goldwasser, Micali and Rackoff [44] and its application in identification protocols was demostrated by Fiat and Shamir [37].

ZPK, as the acronym may suggest, allows a prover to prove its knowledge of a secret without revealing it. Such an approach allows the verifier to gain confidence about the assertion of the prover ("I [prover] know a secret.") and no *additional knowledge*. The technique is highly suited for many applications such as identification and entity authentication in untrusted environments.

Prover		**Verifier**
$a \in_R \mathbb{Z}_q$		$c \in_R \mathbb{Z}_q$
$w = g^a \bmod p$	$\xrightarrow{w}$	
	$\xleftarrow{c}$	
$r = a - cx \bmod q$	$\xrightarrow{r}$	
		$w \overset{?}{=} y^c g^r \bmod p$

Table A.1: The Schnorr Identification Protocol

In order to explain the properties of ZPK, an identification scheme will be analysed. Schnorr [81] proposed an identification scheme that assumes the intractability of the discrete logarithm problem. Let $p = 2q+1$ be a large prime such that q is also a prime. The value of p must render the discrete log problem to be intractable. Let the secret of the prover be $x \in_R \mathbb{Z}_q$. The public image of the secret would be $y = g^x \bmod p$, where $g \in \mathcal{G}_q$ is a generator and $\mathcal{G}_q \subset \mathbb{Z}_p^*$ is the prime order sub-group. The prover and the verifier engage in a protocol described in Table A.1. This protocol requires three communication runs between the prover and the verifier. The runs are named commitment, challenge and response, in that order. Therefore, it is called a *three-pass* protocol.

Observing the transcript of the protocol run, (w, c, r), it *appears* that the verifier cannot obtain any additional knowledge, other than the assertion of the prover. This is because, there exists a simulator that the verifier can employ to generate such transcripts, without interacting with the prover. The simulator performs the following computations:

1. choose $c, r \in_R \mathbb{Z}_q$;
2. compute $w = y^c g^r \bmod p$;
3. output (w, c, r).

If both parties act correctly then (w, c, r) is indistinguishable from the simulated runs. Due to the existence of such a simulator, the protocol possesses some properties of ZKP. Since the verifier could have generated the tuples without interacting with the prover, the verifier would not gain any additional information.

The above analysis of the protocol made a crucial assumption, which being the verifier will choose c in a random fashion. A *dishonest verifier* can, for example, compute the challenge as $c = \mathcal{H}(w)$, where $\mathcal{H}$ is a secure hash function, instead of choosing it randomly. This may allow the dishonest verifier to obtain additional information other than the assertion of the prover because there exists no simulator for a protocol run of the form $(w, c = \mathcal{H}(w), r)$ that cannot compute the pre-image of $\mathcal{H}$. The deficiency can be overcome by requiring the verifier to commit to the challenge, before the prover could send its commitment. The resulting protocol is as shown in Table A.2.

Assuming that the verifier is *honest*, he randomly selects the challenge as prescribed by the protocol in Table A.1, the identification protocol would possess the

Prover		**Verifier**
$a \in_R \mathbb{Z}_q$		$c \in_R \mathbb{Z}_q$
	$\xleftarrow{C}$	$C = g^c \bmod p$
$w = g^a \bmod p$	$\xrightarrow{w}$	
	$\xleftarrow{c}$	
$C \overset{?}{=} g^c \bmod p$		
$r = a - cx \bmod q$	$\xrightarrow{r}$	
		$w \overset{?}{=} y^c g^r \bmod p$

Table A.2: The Perfect-ZKP Schnorr Identification Protocol

zero-knowledge property. For the above mentioned reasons the protocol in Table A.1 is said to belong to the *honest-verifier zero-knowledge protocol* (HVZKP). Considering the number of communication runs required to complete the protocol, it is called a three-pass HVZKP protocol. Three-pass HVZKP protocol constructs are very useful in the synthesis of non-interactive protocols such as signature protocols, when used along with secure hash algorithms, which assumes the role of the honest-verifier. This technique is commonly known as the Fiat-Shamir heuristic [37], the security of which is currently provable only under the random oracle model [74].

Appendix B

Essential Protocol Constructs

B.1 Proof of Knowledge of Discrete Logarithm (PEDL)

The proof of knowledge introduced by Schnorr [81] in the non-interactive mode is presented in this section. Here the prover P has to prove the he knows the discrete logarithm of a public value u, where $u = g^v \bmod p$ and g is a publicly known generator of the group $\mathcal{Z}_p^*$. The prover performs the following function:

```
Begin Function PEDLGen
with input $(u,\ v)$ and output $(c,\ d,\ r)$
        Choose at random   $k \in_R \mathcal{Z}_p$
        Compute            $r = g^k$ and $c = \mathcal{H}(u,\ r)$
                           $d = cv + k \pmod{p-1}$
                           output $\leftarrow (c,\ d,\ r)$
End Function PEDLGen
```

The verifier performs the following function:

```
Begin Function PEDLVer
with input $(c,\ d,\ r,\ g,\ u)$ and output $\in \{0,1\}$
        Check        $g^d \overset{?}{=} u^c r$
                     $c \overset{?}{=} \mathcal{H}(u,\ r)$
        If SUCCESS   output $\leftarrow 1$
        Else         output $\leftarrow 0$
End Function PEDLVer
```

If $\mathcal{H}$ is a cryptographically secure hash function, the verifier can be convinced that the prover knows $\log_g u \bmod p$ when the function PEDLVer outputs 1.

B.1.1 Proof of Equality of Discrete Logarithm (PEQDL)

The proof presented in the previous section can be extended to prove the equality of discrete logarithm of two values to different bases as suggested by Chaum and van Antwerpen [17]. Let a prover P have the knowledge of the discrete logarithm v of $u = g^v \bmod p$ and $u_1 = g_1^v \bmod p$, where g and g_1 are a publicly known generators of the group $\mathcal{Z}_p^*$. The prover performs the following function:

```
Begin Function PEQDLGen
with input $(u,\ u_1,\ v)$ and output $(c,\ d,\ r\ r_1)$
        Choose at random   $k \in_R \mathcal{Z}_p$
        Compute            $r = g^k, r_1 = g_1^k$ and $c = \mathcal{H}(u,\ u_1\ r,\ r_1)$
                           $d = cv + k \pmod{p-1}$
                           output $\leftarrow (c,\ d,\ r,\ r_1)$
End Function PEQDLGen
```

The verifier performs the following function:

```
Begin Function PEQDLVer
with input $(c,\ d,\ r,\ r_1,\ g,\ g_1,\ u,\ u_1)$ and output $\in \{0,1\}$
        Check          $g^d \overset{?}{=} u^c r$
                       Check $g_1^d \overset{?}{=} u_1^c r_1$
                       $c \overset{?}{=} \mathcal{H}(u,\ u_1,\ r,\ r_1)$
        If SUCCESS     output $\leftarrow 1$
        Else           output $\leftarrow 0$
End Function PEQDLVer
```

If $\mathcal{H}$ is a cryptographically secure hash function, the verifier can be convinced that the prover knows $\log_g u \bmod p = \log_{g_1} u_1 \bmod p$ when the function PEQDLVer outputs 1.

B.1.2 Partial Proof of Knowledge of Discrete Logarithm (PPEDL)

Cramer *et al.* [24, 23] proposed a scheme to transform an interactive proof system into a proof system that will convince a verifier that the prover knows some secret, using a suitable secret sharing scheme with an appropriate access structure. In this section we propose a modification to the witness indistinguishable variant of the Schnorr identification protocol [81] proposed in [23] to obtain a more computationally efficient protocol construct that can be used for the proof of knowledge of discrete logarithm. The proposal presented in this section transforms their interactive proof system into a non-interactive proof system and applies the screening technique used in batch verification methods [93, 7] to the protocol proposed in [23]. The soundness and completeness properties of the protocol in [23] are not affected by the changes when a cryptographically secure hash function is used. This is due to the use of standard hashing technique [37] for the transformation. The proposal also integrates the Schnorr signature scheme [81], so that the prover will provide the verifier with transcripts for the proof that also contains his/her signature.

Suppose that a set of values $\mathcal{U} = \{u_i = g^{v_i} \mid i = 1, \cdots, n\}$ are publicly known and a prover, possessing the public key y_j ($y_j = g^{x_j}$), wishes to prove to a verifier that he/she knows the discrete logarithm of *at least one* of the public values. For this to happen the verifier must allow the prover to *simulate* (or cheat) at most $n-1$ proofs. Assume that the prover knows v_j, which is the secret value corresponding to u_j for some $j \in \{1, \cdots, n\}$. The prover performs the following function:

Begin Function PPEDLGen
with input $(\mathcal{U},\ u_j \in \mathcal{U},\ v_j, x_j)$ and output $(d,\ , c,\ \{c_i \mid i = 1, \cdots, n\})$

Choose at random $k_j \in_R \mathcal{Z}_p$, $\{c_l, d_l \in_R \mathcal{Z}_p \mid l \neq j\}$

Compute
$$r_j = g^{k_j} \tag{B.1}$$
$$\{r_l = g^{d_l} u_l^{c_l} \mid l \neq j\} \tag{B.2}$$
$$r = \prod_{i=i}^{n} r_i \bmod p \tag{B.3}$$
$$c = \mathcal{H}(u_1, \cdots, u_n, r) \tag{B.4}$$
$$c_j = c - \sum_{l \neq j} c_l \tag{B.5}$$
$$d = k_j - v_j c_j - x_j c + \sum_{l \neq j} d_l \pmod{p-1} \tag{B.6}$$

```
        output ← $(d,\ ,c,\ \{c_i \mid i = 1, \cdots, n\})$
End Function PPEDLGen
```

The verifier performs the following function:

```
Begin Function PPEDLVer
with input $(\mathcal{U},\ y_j,\ d,\ ,c,\ \{c_i \mid i = 1, \cdots, n\})$ and output $\in \{0, 1\}$
    Check          $c \overset{?}{=} \mathcal{H}(u_1, \cdots, u_n, g^d y_j^c \prod_{i=1}^{n} u_i^{c_i})$    (B.7)
                   $c \overset{?}{=} \sum_{i=i}^{n} c_i$    (B.8)
    If SUCCESS     output ← 1
    Else           output ← 0
End Function PPEDLVer
```

If the function PPEDLVer outputs 1 when the transcripts from the prover are provided as inputs, then the verifier can, with a very high probability, decide that the prover knows the discrete logarithm of at least one of the n public values.

Computational requirements: The function PPEDLGen requires $2n - 1$ modular exponentiations and the function PPEDLVer requires $n + 2$ modular exponentiations.

Analysis

We shall assume that the hash function, $\mathcal{H}$, used for the proof of partial knowledge, is cryptographically secure.

When $|\mathcal{U}| = 1$, $i = j$ and the proof is the usual proof for knowledge of discrete logarithm. So the verification equation will be of the form,

$$c \overset{?}{=} \mathcal{H}(u_1, \cdots, u_n, g^d y_j^c u_j^c) \tag{B.9}$$

which is a standard Schnorr signature with the signature inputs to the hash function of the form $g^d (y_j u_j)^c$. It is evident that this transcript can be formed without the knowledge of discrete logarithm of *both y_j and u_j*, if and only if the Schnorr signature can be forged. Thus, based on the assumption that the Schnorr signature is unforgeable, the above equation is sound, in that the prover cannot cheat the verifier.

When $y_j = 1$, Equation A.7 will be of the form,

$$c = \mathcal{H}(u_1, \cdots, u_n, g^d \prod_{i=1}^{n} u_i^{c_i}) \quad \text{(B.10)}$$

which is the verification equation for the non-interactive version of the protocol proposed in [23]. If the prover can form this equation without the knowledge of discrete logarithms for any of the u_i's then the protocol construct proposed by Cramer *et al.* [23] is flawed *or* the standard hashing technique proposed by Fiat and Shamir [37] is flawed. Thus on the assumption that *both* the techniques [37, 23] are not flawed, the above equation is sound and complete, in that the prover cannot cheat the verifier and the honest prover will generate transcripts that will be accepted by the verifier.

Observe that the Schnorr signature scheme [81] and the partial proof of knowledge protocol [23] are derivatives of the Schnorr identification protocol [81]. The Schnorr identification scheme is a three move protocol, the moves being commitment, challenge and response. The protocol presented in the previous section constrains the prover to use the same commitment and challenge to generate (two) different responses, namely Schnorr signature and partial proof of knowledge, that can be independently interpreted by the verifier. Thus, the proposed protocol constrains valid transcripts to contain a message tuple $(u_1, \cdots, u_n)$, commitment (r) , challenge (c), and the response (d) that is interpreted as a Schnorr signature on the message tuple *and* proof of knowledge of at least one discrete logarithm in the set of public values $(u_1, \cdots, u_n)$.

B.2 Publicly Verifiable Encryption Scheme

The proposal for a public verifiable encryption scheme by Asokan, Shoup and Waidner [3] will be presented in this section. The pseudocodes (to be presented) for the functions can be used to realise an off-line version of their proposal.

B.2.1 System Settings

Let $C = \text{Enc}(t, s_1, y)$ be a public key encryption function that encrypts the message s_1 of length k_s under the public key y using the random string t of length k_t bits and $s_1 = \text{Dec}(C, x)$ be the public key decryption function that decrypts the ciphertext C using the private key x corresponding the public key y. The OAE encryption function of Bellare and Rogaway [6], which is based on the RSA problem, is recommended. The one way function is realised by modular exponentiation as $\mathcal{O}(m) = g^m \bmod p$, where $g \in \mathcal{Z}_p^*$ is a generator. A set of hash functions are chosen such that $\mathcal{H}_1 : \{0,1\}^{160} \to \{0,1\}^{k_s+k_t}$, $\mathcal{H}_2 : \{0,1\}^* \to \{0,1\}^{160}$, $\mathcal{H}_3 : \{0,1\}^* \to \{0,1\}^{160}$ and $\mathcal{H}_4 : \{0,1\}^* \to \{0,1\}^{80}$. To verifiably encrypt a message $s_1 \in \mathcal{Z}_p^*$ under the public key y, *the sender / prover* use the functions in the following sub-sections.

B.2.2 Function VerEnc

1. Select random number r and compute the hash of the value as (t, s_2), which are the higher and lower order bits of the result, respectively. Use t as the randomiser to encrypt s_2 under the public key y. Compute the commitment to the encrypted message as g^{s_2}, where $g \in \mathcal{Z}_p^*$ is the generator of the multiplicative group. Compute the hash value of the ciphertexts and the commitments as h.

2. In a challenge-response mode h is sent to the verifier who chooses a challenge $c \in \{0,1\}$. In our applications we make use of the standard hashing approach to realise an off-line version of the protocol. The user generates many hash values $\{h_j | j = 1, \cdots, n\}$, where n is the security parameter which is 80 in our case. The prover then computes the hash value of all h_j to obtain the challenge c. The prover then uses the bits of c as the challenge.

3. If the j^{th} bit of the challenge c is 0, the prover *opens* the encryption and does not send message information. If the bit is 1 the prover *does not* open the encryption but sends message information that can be decrypted using the private key corresponding to the public key used for encryption.

The algorithm can be described by the function VerEnc as follows:

```
Function VerEnc with input $(s_1, g, p, y)$
and output $(c, D, P_1, \cdots, P_{80})$ is
        Compute $D = g^{s_1} \bmod p$;
        For each $j = 1, \cdots, 80$ do
                Select at random: $r_j \in \{0,1\}^{160}$;
                Compute: $(t_j, s_{2_j}) = \mathcal{H}_1(r)$;
                Encrypt : $(A_j, B_j) = (\text{Enc}(t_j, s_{2_j}, y), g^{s_{2_j}} \bmod p)$;
                Compute: $h_j = \mathcal{H}_2(A_j, B_j)$;
        done;
        Compute: $c = \mathcal{H}_4(h_1, \cdots, h_{80})$;
        For each $j = 1, \cdots, 80$ do
                If $c_j$ is 0 then
                /* $c_j$ is the $j^{th}$ bit of $c$*/
                    Assign: $P_j \leftarrow \{r\}$;
                Else
                    Compute: $s_{3_j} = s_1 + s_{2_j} \bmod p$;
                    Assign: $P_j \leftarrow \{A_j, s_{3_j}\}$;
                EndIf;
        done;
End Function VerEnc;
```

B.2.3 Function CheckVerEnc

1. *The verifier* recomputes the hash value h_j in two different ways. If j^{th} bit of c is 0, the encryption can be recomputed with the value of r and hence the value of h_j. If the bit is 1 the verifier recomputes the hash value h_j using the ciphertext and the commitment.

2. *The verifier* then checks if the challenge was generated properly by recomputing the value of c from the values of h_j. If the verifier is able to check this correctly then the proof is accepted.

The algorithm can be described by the function CheckVerEnc as follows:

```
Function CheckVerEnc with input $(c, p, g, D, P_1, \cdots, P_{80})$
and output $(check)$ is
        For each $j = 1, \cdots, 80$ do
                If $c_j$ is 0 then
                    Assign: $\{r_j\} \leftarrow P_j$;
                    Compute: $(t_j, s_{2_j}) = \mathcal{H}_1(r_j)$;
                Encrypt : $(A_j, B_j) = (\text{Enc}(t_j, s_{2_j}, y), g^{s_{2j}} \bmod p)$;
                Compute: $h_j = \mathcal{H}_2(A_j, B_j)$;
            Else
                Assign: $\{A_j, s_{3_j}\} \leftarrow P_j$;
                Compute: $h_j = \mathcal{H}_2(A_j, g^{s_{3j}}/D)$;
            EndIf;
    done;
    If $c \stackrel{?}{=} \mathcal{H}_4(h_1, \cdots, h_{80})$ then
            Assign: $check \leftarrow PASS$;
    Else
            Assign: $check \leftarrow FAIL$;
    EndIf;
End Function CheckVerEnc;
```

B.2.4 Function DecryptVerEnc

1. *The receiver* locates the ciphertext with message information by locating the bit position in the challenge c that has a value 1. Note that during decryption j has to be selected at random in order to avoid fraud.

2. *The receiver* can then decrypt the ciphertext using its private key to obtain the message.

The algorithm can be described by the function DecryptVerEnc as follows:

```
Function DecryptVerEnc with input $(x, c, P_1, \cdots, P_{80})$
and output $(s_1)$ is
    Forever do
            Select at random: $j \in \{1, \cdots, 80\}$
            if $c_j$ is 1 then
                Assign: $\{A, s_3\} \leftarrow P_j$;
                Break For loop;
            EndIf;
    done;
    Decrypt: $s_2 = \text{Dec}(x, A)$;
```

```
        Compute: $s_1 = s_3 - s_2 \bmod p$;
End Function DecryptVerEnc;
```

Appendix C

Key Escrow Proposals

C.1 Key Escrow with Limited Time Span

The proposal by Burmester, Desmedt and Seberry [13] is briefly outlined in this section.

C.1.1 System Settings

The user generates a large prime p such that $p - 1$ has two large prime factors p_1 and p_2, such that $p_1 \equiv p_2 \equiv 3 \bmod 4$, so that -1 is a quadratic non-residue in the fields $\mathcal{Z}_{p_1}$ and $\mathcal{Z}_{p_2}$. The user then publishes p and $g \in \mathcal{Z}_p$, such that the order of the element g is $p_1 p_2$.

C.1.2 Set-up Phase

The user chooses a secure private key $a \in_R \mathcal{Z}_{p-1}^*$ and computes l shares $\{s_i | i = 1, \cdots, l\}$ of a, such that $\Pi s_i = a \pmod{p-1}$. The user then computes the public key $y_a = g^a \pmod{p}$. The user publishes y_a along with p and g. He/she then engages in a *multi-party* protocol with the law enforcement agency (LEA) and the l trustees in order to obtain a certificate. The multi-party protocol essentially consists of the following steps;

1. The user securely communicates the respective shares to the corresponding trustees and publishes $z_i = g^{s_i}$ for $i = 1, \cdots, l$ in a bulletin board.

2. Each trustee checks if it has received the discrete logarithm of the respective z_i published in the bulletin board. If not the trustee registers a fraud message against the user with the LEA.

3. The user sends $z_{1,2,\cdots,k} = g^{s_1 \cdots s_k}$ for $k = 2, \cdots, l$ to the LEA along with the proofs for $z_{1,2,\cdots,k} = DH(z_{1,2,\cdots,k-1}, z_k)$ for $k = 2, \cdots, l$. It could be noted $z_{1,2,\cdots,l} = y_a$.

4. If no fraud was registered against the user by any of the trustees, the LEA checks the proofs for $z_{1,2,\cdots,k}$ for $k = 2, \cdots, l$ by reading the individual values of z_k from the bulletin board.

5. If the LEA checks the proofs successfully, then it certifies $y_a = z_{1,2,\cdots,l}$ as the users public key in the system.

C.1.3 Update Phase

The homomorphic property of the squaring operator on the private key is used in this phase. The use r computes the new private key as $a_{new} = a^2$ and trustee i updates the i^{th} share as $s_{i_{new}} = s_i^2$. After computing the new share, at least a threshold of the trustees are *trusted* to *erase and forget* the old shares. The user then proves to the LEA that $y_{a_{new}} = g^{a_{new}}$ is the Diffie-Hellman of the old public key y_a, that is $y_{a_{new}} = DH(\ y_a, y_a\ )$, to obtain a certificate for the new public key.

C.1.4 Key Recovery Phase

The users are expected to use the ElGamal cryptosystem to securely communicate using certified public keys. The ciphertext in this system will then be of the form $(g^k, My_a^k) = (A, B)$ for the public key (y_a, g, p). When the LEA obtains a court order to wire-tap the communication of a user, it intercepts the ciphertexts sent to the user. The ciphertext component A along with the court order are sent to the trustees. The trustees then engage in a multi-party protocol to compute y_a^k from A using their respective shares by computing $C = A^{\Pi_{j=1}^{l} a_i}$, where a_i is the share held by the trustees at that time. When the LEA is given C it can compute the message as, $M = B/C$.

C.2 The Escrowed Encryption Standard

The Clipper proposal [88] was announced by the U.S. government in early 1993. It is a symmetric key based solution that is referred to as the escrowed encryption standard (EES). Its primary purpose was to provide strong encryption services for U.S. citizens and, at the same time, provide mechanisms for legal wire-tapping activities by the law enforcement agencies. The scheme envisaged a tamper-resistant hardware device that could be manufactured only by U.S. government approved organisations.

The proposal aimed at developing a hierarchy of secret keys, called unit keys, that could be known only to the hardware device – even the citizen owning the device could not know the unit key corresponding to the device. The only exception to this aim was the existence of two independent escrow agents who have shares of the unit key. The escrow agents can collaborate and compute the unit key.

The design of the scheme also required *all* interoperable clipper chips to possess the knowledge of a *common* key, called the family key. Like the unit keys, the family key is not know to the citizen. The role of the family key is to protect the integrity of the message format and secrecy of the session keys.

The rest of this chapter will explain the chip programming, communication, and wiretapping phases of the proposal.

C.2.1 Chip Programming Phase

Individual integrated circuits (chips) implement the Clipper proposal in a tamper-resistant environment and employ a *declassified* algorithm called Skipjack, which is a 64 bit block cipher with a key size of 80 bits. Each chip contains a unique unit key and share a common family key with all the interoperable chips.

Let the unique unit key be UK_A, where A is the entity owning the chip. This key is generated and shared by the escrow agents such that $UK_A = K_1 \oplus K_2$, where $\oplus$ is the XOR operation and K_i is the secret share of the unit key for the i^{th} escrow agent. This step requires each escrow agent to maintain a *secure* database that can store K_i against the identity of the user, A, and maintain the confidentiality of the share.The family key FK of the domain of operation is also stored in the chip.

The encryption algorithm and algorithm for the verification of integrity rules are wired into the chip.

C.2.2 Communication Phase

When user A, owning $chip_A$, needs to securely communicate with user B, owning $chip_B$, it performs the following operations:

1. a session key, k, is calculated by A and B by employing a suitable, external procedure;

2. A provides the message, m, to be communicated and the session key, k, to $chip_A$ computes and outputs the following values:

    (a) $E(k, m)$, where E is the encryption algorithm of the Skipjack cipher; and,

    (b) $LEAF(A, k) = E(FK, D(A, k))$, where $LEAF$ is an algorithm in all clipper chips and $D(A, K) = (ID_A, E(UK_A, k), f(A, k, IV))$, where ID_A is the identity of user A available in the chip and f is an algorithm providing integrity to (A, k) by employing the IV as the initialisation vector.

3. A sends $(E(k, m), LEAF(A, k))$ to B using an insecure channel;

4. B provides $(E(k, m), LEAF(A, k))$ along with k and A to $chip_B$, which performs the following operations;

    (a) the LEAF component is decrypted by employing FK to obtain $D(A, k)$;

    (b) the integrity of the decrypted value is verified by recomputing $f(A, k, IV)$ and checking its equality with the checksum present in $D(A, k)$;

    (c) if the previous checksum fails, the chip aborts the process without decrypting the message;

    (d) the session key k is employed to decrypt the message m from $E(k, m)$, which is available in $D(A, k)$, and outputs m.

   Observe that although B knows k and $E(k, m)$, it cannot determine the value of m if it does not know the Skipjack algorithm, E. Thus, the secrecy of the encryption algorithm was an essential enforceability mechanism.

C.2.3 Wiretapping Phase

The procedure for wiretapping consists of the following steps:

1. the law enforcement agency (LEA) must obtain a legal authorisation for wiretapping communications between A and B;

2. the LEA presents the authorisation to the escrow agents who would then collude to calculate the unit key as $UK_A = K_1 \oplus K_2$, and provide UK_A to the LEA;

3. the LEA wiretaps insecure channel used in the third step of the communication phase to obtain $(E(k, m), LEAF(A, k))$, decrypts the session key k from the LEAF component by employing UK_A and FK, and decrypts the message m by employing k and $E(k, m)$.

The flaw in the proposal is in the second step of the wiretapping phase, where the escrow agents reveal the unit key, UK_A, which is supposed to be the *long-term, secret key* of user A. This flaw allows the LEA to obtain a single legal authorisation, and decrypt the past present and future communications of A. The proposal by He and Dawson [48] improved this scenario, but still depended on tamper-resistance and secrecy of the encryption algorithm.

Appendix D

Electronic Cash System

This chapter will provide a brief overview of the fair off-line cash scheme, proposed by Frankel, Tsiounis, and Yung [38, 39].

D.1 Introduction

An electronic cash system is an anonymous token system (discussed in Chapter 5) that is for the design of a payment system. In such systems:

1. the tokens are treated as electronic coins;
2. the token issual authority (`TIA`) assumes the role of the mint (or bank);
3. the client would be the customer of the mint (or the bank);
4. the token accepting authority (`TAA`) assumes the role of the merchant;
5. the token issual protocol is employed as the withdrawal protocol;
6. the token utilisation protocol is employed as the spending protocol;
7. the token submission protocol is employed as the deposit protocol;
8. the tracing protocol is employed as a owner-tracing protocol, if it exists.

The first e-cash system that employed a Schnorr-like signature scheme for the withdrawal phase was proposed by Chaum and Pedersen [18]. Brands [12] extended the proposal by Chaum and Pedersen to provide a restrictive blind signature scheme, which provided mechanisms for the merchant to embed the identity of the customer in every coin. This mechanism required the use of the long-term private key that was employed in the withdrawal protocol to be used in the spending protocol, as well. As an effect, the Brands' proposal:

1. achieved the non-transferability property of electronic coins, on the assumption that the customer will not divulge the private key to another entity;
2. allowed for tracing customers who double-spend coins by compromising their private keys.

Frankel, Tsiounis, and Yung [38] exploited the mechanism by designing a new spending transcript that employs a publicly verifiable encryption proof. The publicly verifiable encryption mechanism was a minimal-knowledge proof system that allowed the customer to prove the equality of identity embedded in the coin and the identity encrypted for the trusted third party. Therefore, this system provides *restricted confidentiality* for the identity of the customer – anonymity.

D.2 System Dynamics

System settings: The mint, $\mathcal{B}$, chooses primes p and q such that $p - 1 = \delta + k$ for a specified constant δ, and $p = \gamma q + 1$ for a small integer γ. A unique subgroup $\mathcal{G}_q$ of prime order q of the multiplicative group $\mathbb{Z}_q$ and generators g, g_1, g_2 of $\mathcal{G}_q$ are defined. The mint's secret key $X_B \in_R \mathbb{Z}_q$ is created. Hash functions $\mathcal{H}, \mathcal{H}_0, \mathcal{H}_1, \cdots$, from a family of correlation-free one way hash functions are defined. The mint publishes $p, q, g, g_1, g_2, (\mathcal{H}, \mathcal{H}_0, \mathcal{H}_1, \cdots)$ and its public keys $h = g^{X_B}$, $h_1 = g_1^{X_B}$, $h_2 = g_2^{X_B}$. The public key of the trustee, $\mathcal{T}$, of the form $f_2 = g_2^{X_T}$ is also published, where $X_T \in_R \mathbb{Z}_q$. Note that $\mathcal{T}$ should be a distributed entity to reduce the level of trust placed on it.

The mint associates the user with the identity $I = g_1^{u_1}$, where $u_1 \in_R \mathcal{G}_q$ is generated by the user such that $g_1^{u_1} g_2 \neq 1$. The user is expected to prove the knowledge of discrete logarithm of I *w.r.t.* g_1. The user computes $z' = h_1^{u_1} h_2 = (I g_2)^{X_B}$.

Function IssueToken: The *user* identifies himself/herself to the *mint* (or the mint) and then engages in this protocol to obtain a restrictive blind signature on a *pseudonym* of the form $A = (I g_1)^s$, where g_1 is a base and s a secret, random value. The restrictive blind signature, restricts the structure of A to be of this form. The value of A is never revealed to the mint. We shall express this phase as, $\langle A, Cert_A\rangle_I :=$ $IssueToken(I, \mathcal{B}, \{s\}_I, \{X_B\}_B)$, which should be read as, "I engages in the token issual protocol with $\mathcal{B}$ using a (random) value s (known only to I) to obtain a certificate $Cert_A$ for A, which are known only to I, signed by the mint using its private key X_B."

This protocol creates a restrictive blind signature on I, so that at the completion of the protocol the user obtains a valid signature of the mint on $(Ig_2)^s$ for a random secret value s known only to the user. The signature verification equation $sig(A, B, \mathbf{u_i}) =$ (z, a, b, r) satisfies the following equation:

$$g^r = h^{\mathcal{H}(\mathbf{u_i},A,B,z,a,b)}a \quad \text{and} \quad A^r = z^{\mathcal{H}(\mathbf{u_i},A,B,z,a,b)}b \tag{D.1}$$

In our proposal the anonymous token AT_i would be the tuple $(\mathbf{u_i}, A, B, z, a, b)$, h the public key of the token issuer TIA and $\mathbf{u_i}$ the pseudonym of participant i whose identity in the system is I. The withdrawal protocol is of the form:

User		**Mint**
		$w \in_R \mathbb{Z}_q$
$s, \mathbf{v_i} \in_R \mathbb{Z}_q$	$\xleftarrow{a',b'}$	$a' = g^w, b' = (Ig_2)^w$
$A = (Ig_2)^s$ and $\mathbf{u_i} = \mathbf{g}^{\mathbf{v_i}}$		
$z = z'^s$		
$x_1, x_2, u, v \in_R \mathbb{Z}_q$		
$B_1 = g_1^{x_1}$		
$B_2 = g_2^{x_2}$		
$B = [B_1, B_2]$		
$a = (a')^u g^v$		
$b = (b')^{su} A^v$		
$c = \mathcal{H}(\mathbf{u_i}, A, B, z, a, b)$		
$c' = c/u$	$\xrightarrow{c'}$	$r' = c'X_B + w \bmod q$
$r = r'u + v \bmod q$	$\xleftarrow{r'}$	

The user verifies if $g^{r'} \overset{?}{=} h^{c'}a'$, $(Ig_2)^{r'} \overset{?}{=} z'^{c'}b'$.

Function UtiliseToken: The *user* derives two pseudonyms of the form $A_1 = g^{u_1 s}$ and $A_2 = g_1^s$ from A, such that $A = A_1 A_2$. The user then proves to the *merchant* its knowledge of the pre-images of A_1 and A_2 with respect to the reference bases g and g_1 respectively (thereby proving the knowledge of representation of A). Additionally, he/she proves that his/her identity, which is *hidden* in A_1, has been encrypted for a trustee under the public key $f_2 = g^{X_T}$ resulting in the ciphertext $D = (D_1, D_2)$. The user never reveals his/her identity, I, to the merchant. We shall express this phase as, $\langle Proof_A \rangle := UtiliseToken(A, Cert_A, \mathcal{M}, y_b, f_2, \{s, u_1\}_A)$, which should be read as, "A engages in the utilise token protocol with $\mathcal{M}$ using the certificate, $Cert_A$, $\mathcal{B}$'s public key y_B and the private data (s, u_1) to generate the transcripts for a proof system $\langle Proof_A \rangle$, which contains an encryption of the identity of the user under the public key f_2." $\langle Proof_A \rangle$ contains the following tuples, $(A_1, A_2, Encryption_{f_2}(I))$ *along with* the corresponding proof transcripts. Here $Encryption_{f_2}(I)$ is the encryption of the user identity I under the public key f_2.

This protocol is performed in an anonymous channel. The following table provides the sketch for the protocol. Note that, in the protocol, the tuple (D_1, D_2) is an ElGamal encryption of the identity of the user I under the public key of $\mathcal{T}$. Also note how the identity of the shop $\mathcal{I}_S$ is bound to the coin when the challenge d is generated by the shop.

User		**Shop**
$A_1 = g_1^{u_1 s},\ A_2 = g_2^{u_1 s}$		
$m \in_R \mathbb{Z}_q$		
$D_1 = I g^{X_T m},\ D_2 = g_2^m$	$\xrightarrow{D_1, D_2}$	$D_2 \overset{?}{\neq} 1$
	$\xrightarrow{\mathbf{u_i}, A_1, A_2, A, B, (z,a,b,r)}$	$A \overset{?}{=} A_1 A_2,\ A \overset{?}{\neq} 1$
		$sig(A, B, \mathbf{u_i}) \overset{?}{=} (z, a, b, r)$
		$d = \mathcal{H}_1(A_1, B_1, A_2, B_2, \mathcal{I}_S, \text{date/time})$
		$s_0, s_1, s_2 \in_R \mathbb{Z}_q$
		$D' = D_1^{s_0} g_2^{s_1} D_2^{s_2}$
$V = \mathcal{H}_1((D')^s / (f_2')^{ms})$	$\xleftarrow{d, D', f_2'}$	$f_2' = f_2^{s_0} g_2^{s_2}$
$r_1 = d(u_1 s) + x_1$		
$r_2 = ds + x_2$	$\xrightarrow{r_1, r_2, V}$	$V \overset{?}{=} \mathcal{H}_1(A_1^{s_0} A_2^{s_1})$
		$g_1^{r_1} \overset{?}{=} A_1^d B_1,\ g_2^{r_2} \overset{?}{=} A_2^d B_2$

Function SubmitToken: The *merchant* submits the proofs of knowledge, which it received in the Protocol UtiliseToken, to the *mint* and avails credit. The mint can check if it has already received the tuple (A_1, A_2) to detect double spent transcripts. We shall express this phase as, $SubmitToken(\mathcal{M}, \mathcal{B}, A, Cert_A, \langle Proof_A \rangle)$, which should be read as, "$\mathcal{M}$ engages in the submit token protocol with $\mathcal{B}$ to submit the values $\langle A, Cert_A \rangle$ and $\langle Proof_A \rangle$."

The shop deposits the payment transcripts to the mint, which checks the transcripts using the same checking equations that the shop employed during the payment protocol. If the equations hold then the mint deposits a suitable amount into the shop's account.

Function Trace: The *mint*, or any other authorised entity, traces the identity of the user who spent a particular transcript, by contacting the *trustee*. When the trustee is provided with the transcript $\langle Proof_A \rangle$, it can retrieve the ciphertext $Encryption_{f_2}(I)$ and decrypt it using its private key to obtain the identity. We are only interested in the "owner tracing" aspect and not in the "coin tracing" aspect of fair e-cash. We shall express this phase as, $\langle I, Proof_T \rangle := Trace(\mathcal{X}, \mathcal{T}, \langle A, Cert_A \rangle, \langle Proof_A \rangle, \{X_T\}_T)$, which should be read as, "$\mathcal{X}$ engages in the tracing protocol with $\mathcal{T}$ using the values $\langle A, Cert_A \rangle$ and $\langle Proof_A \rangle$, to obtain the identity I and an *optional*[1] proof $Proof_T$, for proof of correct decryption of the ciphertext. The trustee uses its private key X_T for this purpose."

[1]It is surprising that many proposals have not explicitly concentrated on this aspect, namely proof of correct revocation.

Appendix E

Probability of Deadlock and Derangement

Section 5.3.2 presented a peer review protocol that could potentially result in a deadlock situation. It will be useful to derive an equation that provides the probability of deadlock occurrence as a function of the number of participants, n. Every participant in the protocol is always presented with a set of choices, which contains the pseudonyms of available reviewers. The set of choices for the last participant contains a single element. That is, the last participant will have no choice for its reviewer. A deadlock is said to have occurred if the pseudonym (choice) represents the last participant. This is a deadlock because no participant in the protocol is allowed to choose itself. This appendix presents an analysis of the problem using the combinatorial problem for counting derangement.

Every run of the peer review protocol results in a permutation of the set containing the pseudonyms of all reviewers. For simplicity, consider a protocol with only three participants (reviewers or peers). Let A, B and C be the pseudonyms of the three participants. Without loss of generality, let A exercise the first choice, B the second choice, and finally C. A tree representing all possible permutations and probabilities for $n = 3$ is provided in Figure E.1. Level A depicts all the sets of choices that may be available for A during the protocol. Levels B and C have the same connotation. $\{\overline{A}, B, C\}$ represents the set of choices for A, given the restriction that A cannot choose itself. Similar sets of choices label the respective nodes (rectangles). Consider the leftmost branch of the tree: if A chooses C, then B must choose A, since the protocol allows only C to choose itself. It is evident from the graph that the permutation (C, A, B) occurs with a probability of $1/2$ and the other two permutations (B, C, A)

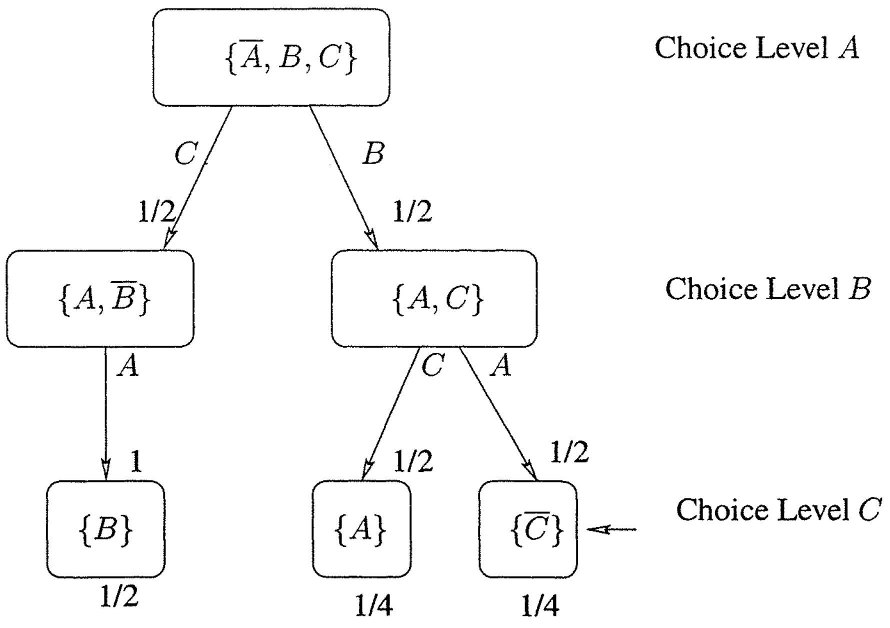

Figure E.1: Tree of Selection for $n = 3$

and (B, A, C) occur with a probability of $1/4$. Thus the deadlock situation, where C must choose itself occurs with a probability of $1/4$ when $n = 3$. The probability that this is not the case is $1/2 + 1/4 = 3/4$.

Notice that in Figure E.1, the tree of selection for $n = 3$, the leftmost sub-tree (containing the permutation (C, A, B)) does not represent any deadlock permutation. Such a sub-tree, where there will be no deadlock permutations, will exist for all values of $n \geq 3$. For example when $n = 4$, the first participant (say A) would have the choice $\{\overline{A}, B, C, D\}$. Suppose A selects uniformly from this set of choices. Then A will select D with a probability of $1/3$. The sub-tree representing this choice will not represent any deadlock permutations.

An interesting pattern can be observed by examining trees of selection for three, four and five participants. Probability of deadlock for a given number of participants is:

$$p_D = \sum_{k=1}^{l} \prod_{i=1}^{n-1} p_i^{(k)}$$

where, l is the number of deadlock permutations, n is the number of participants and $p_i^{(k)}$ is the probability of selection of the i^{th} participant in the k^{th} deadlock permutation.

It is the case that the contribution to the deadlock probability by the first participant is $p_1^{(k)} = p_1^{(j)} = 1/(n-1)$ for all k and j such that $k \neq j$. That is the probability contribution by the first participant is a constant. It is also the case that the probability contribution by the penultimate participant, $i = n-1$, is also a constant in the above expression. The penultimate participant, in a deadlocked permutation, is always provided with a set of choices of cardinality two. This is because if the penultimate participant is provided with a set of choices of cardinality one, it will be forced to choose the last participant, if the last participant is not already chosen. Thereby, effectively avoiding the deadlock situation. For the above reasons, $p_{n-1}^{(k)} = p_{n-1}^{(j)} = 1/2$ for all k and j such that $k \neq j$. The equation for probability for deadlock could then be reduced as follows:

$$p_D = \frac{1}{2(n-1)} \sum_{k=1}^{l} \prod_{i=2}^{n-2} p_i^{(k)}$$

It will be interesting to investigate for a recursive expression, or any form of expression that can be computed without having to construct the whole tree of selection, to replace the series representing the summation of products. Figure E.2 provides a plot for the first term in the above expression for p_D, which is $1/2(n-1)$.

E.1 Derangement

The problem of calculating the probability of deadlock occurrence is partly related to the combinatorial problem called derangement [46, Chapter 8]. This area of combinatorics is interested in counting the number of permutations of n objects such that none of the objects are in their original position. This problem is also known as the sub-factorial problem[1]. A well known formula for counting the derangements [46] is as follows:

$$Derangement(n) = n! \sum_{k=0}^{n} (-1)^k/(k!)$$

where $n!$ represents the factorial of n. Note that $\sum_{k=0}^{n}(-1)^k/(k!)$ is the Macluarin series that converges to e^{-1} in the limit as $n \to \infty$, where e is the base of natural logarithm.

[1]Thanks to Mr. Greg Maitland for introducing the term.

In the peer review protocol, an upper bound for the probability of deadlock occurrence can be computed by considering the hypothetical case where all the permutations are equally likely. Let N_D be the number of possible permutations that result in a deadlock and N_G be the number of possible permutations that are good (that do not result in deadlock).

The proposed peer review protocol allows a fixed point to occur only in one place: that of the last object (or participant). Therefore, the number of possible deadlock permutations, when there are n participants, can be computed as follows:

$$N_D = Derangement(n-1)$$

That is, the position of the last object is fixed and the remaining objects are deranged. The number of good permutations is:

$$N_G = Derangement(n)$$

The probability of deadlock occurrence, p'_D, assuming equally likely permutations will be:

$$p'_D = N_D/(N_D + N_G)$$

After the substitution of the values for N_D and N_B, cancellation of the relevant terms, and the approximation of the Maclaurin series to $1/e$, the following approximation can be obtained:

$$p'_D \approx \frac{1}{1+n(1+\frac{(-1)^n e}{n!})}$$

For large values of n, the above equation can be further approximated as follows:

$$p'_D \approx \frac{1}{1+n}$$

The graph that plots the exact value of the probability of deadlock and the approximation, p'_D, presented in this section, is shown in Figure E.2. The algorithm that was employed to compute the exact value of the probability of deadlock used the following logic. To solve the problem, it would be fair to assume that P_i will choose P_j uniformly from the set $\mathcal{S}\backslash\{P_i\}$, where $\mathcal{S}$ is the set of choices available for P_i, so that the probability that P_i will choose P_j will be $p_i = 1/|\mathcal{S}\backslash\{P_i\}|$. The first participant

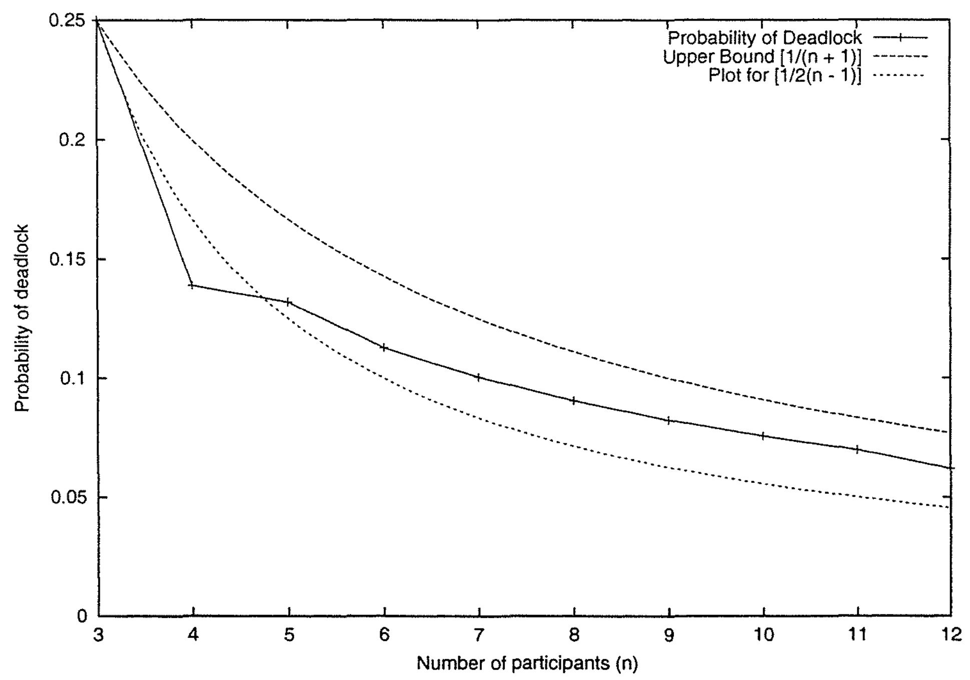

Figure E.2: A Graph for the Probability of DeadLock Occurrence

will always have $n - 1$ possible choices for reviewers. Therefore, $p_1 = 1/(n-1)$ (for $n = 3, p_1 = 1/2$). Assuming that the different participants choose independently, the probability of individual deadlock permutations can be computed as a product of the probability of every selection in the deadlock permutation. The total probability of deadlock can be calculated as the sum of the probability of occurrence of individual deadlock permutations. That is,

$$p_D = \sum_{k=1}^{l} \prod_{i=1}^{n} p_i^{(k)}$$

where, l is the number of deadlock permutations, n is the number of participants and $p_i^{(k)}$ is the probability of selection of the i^{th} participant in the k^{th} deadlock permutation.

This study of deadlock permutations suggests that the probability of deadlock occurrence decreases when the number of participants increase. Although an equation to estimate the probability of deadlock was provided, an equation for the precise calculation of deadlock occurrence is an interesting open problem.

E.2 Source Code

The source code implementation in the C language for computing the probability of deadlock occurrence given the number of participants is presented in this section. The program takes parameters representing the number of participants and computes the corresponding probability values for the occurrence of deadlock. The program works satisfactorily until $n = 12$. Currently, when provided with values of $n \geq 13$, the program requires significant memory and computation time on a Pentium II processor with 64 MB RAM.

```
#include <stdio.h>
#include <stdlib.h>

/*Algorithm for obtaining the next permutation in a
lexicographical order. Returns the lexicographical
position of the resulting permutation contained in
intarray */

int getNextHighPerm(int * pi, int n)
{

  int i, j, count; /* primary indices */

  int r, s, temp; /*indices for swapping purposes */

  int lastPerm = 0;


  /* Find the rightmost place where pi[i] > pi[i + 1] */

  for(i = n - 2; pi[i] > pi[i+1];i -=1);

  /* Find pi[j], the smallest element to the right
     of pi[j] and greater than it*/

  for(j = n - 1; pi[i] > pi[j];j -= 1);

  /* Interchange and then reverse pi[i + 1],...,pi[n - 1] */
  /* Interchanging */
  temp = pi[i];
  pi[i] = pi[j];
  pi[j] = temp;

  /*Reversing*/

  r = n - 1;
  s = i + 1;
  while( r > s){

      /*Swaping pi[r] and pi[s]*/
      if(s >= 0 & s < n & r>=0 & r <n){
        temp = pi[r];
        pi[r] = pi[s];
        pi[s] = temp;
      }
      r -= 1;
      s += 1;
  }

  for(i = n - 2; pi[i] > pi[i+1];i -=1);
  if(i < 0) lastPerm = 1;


  return lastPerm;
}


/*Algorithm to check if there are any fixed points in
  the given permutation contained in pi*/

int noFixedPoint(int* pi, int n){
  /*Let us start with true and set the flag to false
    when there is a fixed point*/
  int nfp = 1;

  int count;

  for(count = 0; (count < n); count++){
    if(pi[count] == count){
      nfp = 0;
      break;
    }
  }

  return nfp;
}


/*Algorithm for parsing the permutation in the array
 pi, and calculating the probability weight.
*/

int getDLProb(int* pi, int n){
  int dlProb= 1;
  int i, j;
  int search, found;

  for(i = 0, j = n - 1; i < n - 1 & j >= 1; i++, j--){
    found = 0; /*not found*/
    for(search = 0; search < i; search ++){
      if(pi[search] == i){
        found = 1;
        break;
      }
    }
    if(found == 0) dlProb*= j;
    else dlProb *= (j + 1);
  }
  return dlProb;
}


/* The startup function that contains the workhorse and
   performes memory management for the program.*/

int main(int argc, char* argv[]){
  int untiln, n, lastPerm;
  int * pi;
  int i, temp;
  int probCount;
  float prob;

  if(argc < 2){
    printf("Usage: progname <upperlimit> \n");
    printf("Number must be less than 13 \n");
    printf("OR: progname <upperlimit> <lowerlimit> \n");
    exit(0);
  }

  untiln = atoi(argv[1]);

  if(argc > 2){
    /*User has given lower limit*/
    n = atoi(argv[2]);
  }else{/*Use the default lower limit*/
    n = 3;
  }

  /*Some cosmetics */
  printf("n      \t      Probability\n");
  for(i =0; i <80;i++) printf("-");
  printf("\n");

  /*Now lets start the workhorse: Do permutations
   from n to untiln */

  for(;n <= untiln ; n++){ /*Count prob for 3 to untiln */

      prob = 0.0; /*dl probability is zero to start with*/

      pi = (int *) malloc(n * sizeof(int));

      if(pi == NULL){
        printf("Unable to allocate memory. Damn! \n");
        exit(0);
      }

      /*Initialise the pi array with 0, 1, 2, 3,...,n */
      for(i = 0; i < n; i++) pi[i] = i;


      /*Starting! This is not the last permutation.*/
      lastPerm = 0;

      /*Do until the last permutation.*/
      do{
        lastPerm = getNextHighPerm(pi, n-1);
        if((noFixedPoint(pi, n-1)) == 1){

          /* pi contains a permutation without any fixed points.*/
          probCount = getDLProb(pi, n);
          prob += 1.0/probCount;
        }
      }while(lastPerm == 0);
      printf("%d \t\t %f \n",n , prob);

      /*Time to get rid of unwanted memory.*/
      free(pi);

  }/*Done until the number untiln.*/
  return 0;
}
```

Bibliography

[1] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000), Sendai, Japan*, 2000. To appear.

[2] Masayuki Abe. Mix-networks on permutations networks. In K. Lam, E. Okamoto, and C. Xing, editors, *Advances in Cryptology – ASIACRYPT'99*, volume 1716 of *LNCS*, pages 258–273. Springer-Verlag, 1999.

[3] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. In *Advances in Cryptology - EUROCRYPT'98*, pages 591–606. Springer-Verlag, 1998.

[4] Giuseppe Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In *6^{th} ACM Conference on Computer and Communication Security*. ACM Press, 1999.

[5] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[6] M. Bellare and P. Rogaway. Optimal Asymetric Encryption. In Alfredo De Santis, editor, *Advances in Cryptology – CRYPTO'94*, volume 950 of *LNCS*, pages 92–111. Springer-Verlag, 1994.

[7] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital sigantures. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *LNCS*, pages 236–250. Springer-Verlag, 1998.

[8] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Proceedings of STOC'94*, pages 544–553, 1994.

[9] Josh Daniel Cohen Benaloh. *Verifiable Secret-Ballot Elections.* PhD thesis, Faculty of the Graduate School, Yale University, December 1996.

[10] Matt Blaze. Protocol failure in the escrowed encryption standard. In *The 2nd ACM Conference on Computer and Communications Security*, November 1994.

[11] Colin Boyd. Enforcing traceability in software. In Y. Han, T. Okamoto, and S. Quing, editors, *International Conference on Information and Communication Security, ICICS'97*, volume 1334 of *LNCS*, pages 398–408. Springer-Verlag, 1997.

[12] Stefan Brands. Untraceable Off-line Cash in Wallet with Observers. In Tor Helleseth, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *LNCS*, pages 344–359. Springer-Verlag, 1993.

[13] Mike Burmester, Yvo Desmedt, and Jennifer Seberry. Equitable key escrow with limited time span. In K. Ohta and D. Pei, editors, *Advances in Cryptology – ASIACRYPT'98*, volume 1514 of *LNCS*, pages 380 – 391. Springer-Verlag, 1998.

[14] M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. In *Proceedings of the Royal Society of London*, volume 426, pages 233–271, 1989.

[15] W.J. Caelli and D. Longley. Implementation of key escrow with key vectors to minimise potential misuse of key. In *20th National Information Systems Conference, Baltimore*. NIST, October 1997.

[16] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In Burt Kaliski, editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *LNCS*, pages 90–104. Springer-Verlag, 1997.

[17] D. Chaum and H. van Antwerpen. Undeniable signatures. In G. Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *LNCS*, pages 212–216. Springer-Verlag, 1989.

[18] David Chaum and T. Pedersen. Wallet Databases with Observers. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer-Verlag, 1992.

[19] David L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[20] C.J.Mitchell, F.Piper, and P.Wild. *Contemporary Cryptology: The Science of Information Integrity*, chapter Digital Signatures, pages 325–378. IEEE Press, 1992.

[21] Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme. In *IEEE Symp. on Foundations of Computer Science*, pages 372–382. IEEE press, 1985.

[22] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election shceme. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *LNCS*, pages 103–118. Springer-Verlag, 1997.

[23] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. Technical Report CS-R9413, Computer Science/Department of Algorithmics and Architecture, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, 1994.

[24] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo G. Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer-Verlag, 1994.

[25] Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. In Ueli Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *LNCS*, pages 72–83. Springer-Verlag, 1996.

[26] Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In H. Krawczyk, editor,

Advances in Cryptology – CRYPTO'98, volume 1462 of *LNCS*, pages 13–25. Springer-Verlag, 1998.

[27] Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer-Verlag, 1998.

[28] David Chaum. Blind Signatures for Untraceable Payments. In Sherman A.T. Chaum D., Rivest R.L., editor, *Advances in Cryptology – CRYPTO'82*, pages 199–203. Plenum Press, 1983.

[29] D.E. Denning and D.K. Branstad. A taxonomy for key escrow encryption systems. *Communications of ACM*, pages 34–40, March 1996.

[30] Yvo Desmedt. Securing traceability of ciphertexts — towards a secure software key escrow system. In L. C. Guillou and J.-J. Quisquater, editor, *Advances in Cryptology – EUROCRYPT'95*, volume 921 of *LNCS*, pages 147–157. Springer-Verlag, 1995.

[31] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[32] W. Diffie, P.C. van Oorschot, and M.J. Weiner. Authentication and authenticated key exchanges. In *Designs, Codes and Cryptography*, volume 2, pages 107–125, 1992.

[33] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[34] Whitfield Diffie and Susan Landau. *Privacy on the Line: The Politics of Wiretapping and Encryption.* The MIT Press, Cambridge, Massachusetts, 1998. ISBN 0-262-04167-7.

[35] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proceedings of the Twenty Third Annual Symposium on the Theory of Computing, ACM*, pages 542–552, 1991.

[36] Douglas E. Comer. *Internetworking with TCP/IP*, volume 1. Prentice Hall, 1995.

[37] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer-Verlag, 1986.

[38] Yair Frankel, Yiannis Tsiounis, and Moti Yung. Indirect Discourse Proofs: Achieving Efficient Fair Off-Line E-Cash. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT'96*, volume 1163 of *LNCS*, pages 286–300. Springer-Verlag, 1996.

[39] Yair Frankel, Yiannis Tsiounis, and Moti Yung. Fair Off-Line e-cash Made Easy. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology – ASIACRYPT'98*, volume 1514 of *LNCS*, pages 257–270. Springer-Verlag, 1998.

[40] Yair Frankel and Moti Yung. Escrow encryption systems visited: Attacks, analysis and designs. In D. Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *LNCS*, pages 222–235. Springer-Verlag, 1995.

[41] Matthew K. Franklin and Michael K. Reiter. Verifiable signature sharing. In L. Guilloy and J-J. Quisquater, editors, *Advances in Cryptology – EUROCRYPT'95*, volume 921 of *LNCS*, pages 50–63. Springer-Verlag, 1995.

[42] Rossario Gennaro, Paul Karger, Stephen Matyas, Mohammad Peyravian, Allen Roginsky, David Safford, Michael Willet, and Nev Zunic. Two-phase cryptographic key recovery system. *Computers and Security*, 16:481–506, 1997. Available at: http://www.ibm.com/security/library/wp_key.html.

[43] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and Ssytem Sciences*, 28:270–299, 1984.

[44] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J.Computing*, 18:186–208, 1985.

[45] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[46] Ralph P. Grimaldi. *Discrete And Combinatorial Mathematics, An Applied Introduction*. Addison-Wesley, second edition, 1989.

[47] Michael Harkavy, Doug Tyger, and Hiroaki Kikuchi. Electronic auctions with private bids. In *Third USENIX Workshop on Electronic Commerce*, 1998.

[48] Jingmin He and Ed Dawson. A new key escrow cryptosystem. In Josef Pieprzyk and Jennifer Seberry, editors, *Information Security and Privacy, ACISP'96*, volume 1172 of *LNCS*, pages 105–114. Springer-Verlag, 1996.

[49] Patrick Horster, Markus Michels, and Holger Petersen. Generalized ElGamal signatures for one message block. Technical Report TR-94-3, Department of Computer Science, University of Technology Chemnitz-Zwickau, May 1994.

[50] IBM. IBM SecureWay key recovery technology, a techonology review. ftp://service2.boulder.ibm.com/software/icserver/doc/keyrec.pdf, 1997.

[51] Markus Jakobsson. A Practical Mix. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *LNCS*, pages 448–462. Springer-Verlag, 1998.

[52] Hiroaki Kikuchi, Michael Harkavy, and Doug Tygar. Multi-Round Anonymous Auction Protocols. *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems*, 1999.

[53] Joe Kilian and Tom Leighton. Fair cryptosystem, revisited. In D. Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *LNCS*, pages 208–221. Springer-Verlag, 1995.

[54] Lars R. Knudsen and Torben P. Pedersen. On the difficulty of software key escrow. In U. M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *LNCS*, pages 237–244. Springer-Verlag, 1996.

[55] Jan L.Camenisch, Jean Marc Piveteau, and Markus A. Stadler. Blind Signatures Based on the Discrete Logarithm Problem. In Alfredo DeSantis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *LNCS*, pages 428–432. Springer-Verlag, 1994.

[56] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography, PKC'2000*, volume 1751 of *LNCS*, pages 446–465. Springer-Verlag, 2000.

[57] Arjen K. Lenstra, Peter Winkler, and Yacov Yacobi. A key escrow system with warrant bounds. In D. Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *LNCS*, pages 197 – 207. Springer-Verlag, 1995.

[58] Ueli M. Maurer and James L. Massey. Cascade ciphers: The importance of being first. *Journal of Cryptology*, 6(4):55–61, 1993.

[59] Masahiro Mambo and Eiji Okamoto. Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. In *IEICE Trans. Fundamentals*, volume E80-A, January 1997.

[60] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy signatures: Delegation of the power to sign messages. In *IEICE Trans. Fundamentals*, volume E79-A, September 1996.

[61] Matthew K. Franklin and Michael K. Reiter. The design and implementation of a secure auction service. *IEEE Transaction on Software Engineering*, 22(5):302–312, May 1996.

[62] Alfred J. Menezes, Paul C. von Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[63] S. Micali. Fair public-key cryptosystem. Technical Report Technical Number 578, MIT lab for Computer Science, September 1993.

[64] Paul Milgrom. Auctions and Bidding: A Primer. *Journal of Economic Perspectives*, 3(3):3–22, 1989.

[65] National Institute of Standards and Technology, Govt. of U.S.A. Requirements for key recovery products. Available at http://csrc.nist.gov/keyrecovery/, November 1998. Report of the Technical Advisory Commitee to Develop A Federal Information Processing Standard for the Federal Key Management Infrastructure.

[66] Valtteri Niemi and Ari Renvall. How to prevent buying of votes in computer elections. In Josef Pieprzyk and Reihanah Safavi-Naini, editors, *Advances in Cryptology - ASIACRYPT'94*, volume 917 of *LNCS*, pages 164–170. Springer-Verlag, 1994.

[67] NIST, National Institute of Standards and Technology, Gov. of the USA. *Security Requirements for Cryptgraphic Modules, FIPS 140-1*, January 1994.

[68] Hannu Nurmi, Arto Salomaa, and Lila Santean. Secret ballot elections in computer networks. *Computers & Security*, 10:553–560, 1991.

[69] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In Bruce Christianson, Bruno Crispo, Mark Lomas, and Michael Roe, editors, *Security Protocol, 5^{th} International Workshop*, volume 1361 of *LNCS*, pages 25–36. Springer-Verlag, 1997.

[70] Tatsuaki Okamoto and Kazuo Ohta. Universal Electronic Cash. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *LNCS*, pages 324–337. Springer-Verlag, 1991.

[71] T. Pedersen. Non-interactive and information theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer-Verlag, 1991.

[72] Holger Petersen and Guillaume Poupard. Efficient Scalable Fair Cash with Off-line Extortion Prevention. In *Information and Communication Security, ICICS'97*, pages 463–473, November 1997.

[73] Birgit Pfitzmann and Michael Waidner. How to break fraud-detectable key recovery. *Operating Systems Review, ACM press*, 32(1):23–28, January 1998.

[74] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli Muarer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer-Verlag, 1996.

[75] M. Rabin. How to exchange secrets by oblivious transfer. Technical Report Technical Memo, TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[76] Cristian Radu, René Govaerts, and Joos Vandewalle. Efficient electronic cash with restricted privacy. In Rafael Hirschfeld, editor, *Financial Cryptography, FC'97*, volume 1318 of *LNCS*, pages 24–28. Springer-Verlag, 1997.

[77] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[78] Rainer A. Rueppel. A formal approach to security architectures. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *LNCS*, pages 387–398. Springer-Verlag, 1991.

[79] Kazue Sako. An Auction Scheme which Hides the Bids of Losers. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography, PKC'2000*, number 1751 in LNCS, pages 422–432. Springer-Verlag, 2000.

[80] K. Sakurai and S. Miyazaki. A Bulletin-Board Based Digital Auction Scheme with Bidding Down Strategy. In *Proceedings of CrypTEC'99*, pages 180–187. City University of Hong Kong Press, July 1999.

[81] C.P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4:161–174, 1991.

[82] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In M. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *LNCS*, pages 148–164. Springer-Verlag, 1999.

[83] S.Goldwasser, S.Micali, and R.Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM journal of computing*, 17(2):281–308, April 1998.

[84] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.

[85] Gustavus J. Simmons. A natural taxonomy for digital information authentication schemes. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, LNCS, pages 269–288. Springer-Verlag, 1987.

[86] Markus Stadler. Publicly verifiable secret sharing. In Ueli Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *LNCS*, pages 190 – 199. Springer-Verlag, 1996.

[87] S.G. Stubblebine and P.F. Syverson. Fair On-line Auctions Without Special Trusted Parties. In Matthew Franklin, editor, *Financial Cryptography, FC'99*, volume 1648 of *LNCS*. Springer-Verlag, 1999.

[88] U.S. DEPARTMENT OF COMMERCE / National Institute of Standards and Technology. *Federal Information Processing Standard 185—Escrowed Encryption Standard*, February 1994.

[89] Eric R. Verheul and Henk C.A. van Tilborg. Binding ElGamal: A fraud-detectable alternative to key-escrow proposals. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *LNCS*, pages 119–133. Springer-Verlag, 1997.

[90] Kapali Viswanathan, Colin Boyd, and Ed Dawson. Publicly verifiable key escrow with limited time span. In Pieprzyk J., Safavi-Naini R., and Seberry J., editors, *Information Security and Privacy, ACISP'99*, volume 1587 of *LNCS*, pages 36–50. Springer-Verlag, 1999.

[91] Kapali Viswanathan, Colin Boyd, and Ed Dawson. A three phased schema for sealed bid auction system design. In Ed Dawson, Andrew Clark, and Colin Boyd, editors, *Information Security and Privacy, ACISP'2000*, volume 1841 of *LNCS*, pages 412–426. Springer-Verlag, 2000.

[92] B. von Solms and D. Naccache. On Blind Signatures and perfect crimes. *Computers and Security*, pages 581–583, October 1992.

[93] S. Yen and C. Laih. Improved digital signature suitable for batch verification. *IEEE Transactions on Computers*, 44(7):957–959, July 1995.

[94] Adam Young and Moti Yung. Auto-recoverable and auto-certifiable cryptosystems. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *LNCS*, pages 17–31. Springer-Verlag, 1998.

[95] Adam Young and Moti Yung. Towards signature-only signature schemes. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT'2000*, volume 1976 of *LNCS*, pages 97–115. Springer-Verlag, 2000.