CERIAS Tech Report 2003-36

EFFICIENT PRIMITIVES FOR ENSURING SECURITY IN E-COMMERCE TRANSACTIONS

by Jung Min Park

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

EFFICIENT PRIMITIVES FOR ENSURING SECURITY IN E-COMMERCE TRANSACTIONS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Jung Min Park

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2003

To my parents and my brother,

for their love and support

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisors, Professor Edwin K. P. Chong and Professor Howard Jay Siegel, for their encouragement and support throughout the years of my doctoral study. Their guidance and nurturing not only sowed the seeds of my curiosity and interest in research, but also helped me reap the results of my research efforts that made this thesis possible. Their dedication to research and teaching has made a profound impression on me, and I will try to follow suit as I take the first steps of my career after graduation. I would also like to express my gratitude to Professor Ness Shroff, whose help and support were invaluable to me. In addition, I would also like to thank the other members of my committee, Professor Arif Ghafoor and Professor Kihong Park, for their helpful suggestions.

During my stay at Colorado State University as a visiting research associate, I was fortunate enough to work with Professor Indrajit Ray. Our collaboration was very valuable to my research, and I am grateful to him for his help and support.

This work could not have been possible without the support and love of my family. I will always be grateful to my father, Young Ju Park, and my mother, Myoung Hee Lee, for their love and encouragement that has afforded me the desire and determination to pursue my goals, how lofty they might be. I am also grateful to my wife, Yoo-Young Chung, and my younger brother, Jung Hoon Park, for their boundless support. They will always have a special place in my heart.

I would like to thank my friends and officemates for making my years at Purdue enjoyable and memorable. Special thanks go to Hunsoo Choo, Jeongsoon Park, Jang Won Lee, Heon Huh, Woopyo Jeong, Jeongjoon Lee, Jong-Hyeok Jeon, Gang Wu, Xin Liu, Jong-Kook Kim, and Jeff Herdtner for their friendship.

This dissertation was supported by the Purdue Center for Education and Research in Information Assurance and Security (CERIAS), by the Colorado State University

George T. Abell Endowment, by NSF under grants ANI-0099137, ECS-0098089, and ANI-0207892, and by DARPA/ISO under contracts DABT63-99-C0012 and DABT63-99-C0010.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Park, Jung Min. Ph.D., Purdue University, August, 2003. Efficient Primitives for Ensuring Security in E-Commerce Transactions. Major Professors: Edwin K. P. Chong and Howard Jay Siegel.

Fueled by the exponential growth in the number of people with access to the Internet, electronic-commerce (e-commerce) transactions via the Internet have become a major part of our economy. For a wider range of e-commerce applications to take advantage of the untapped business potential of the Internet, some challenging and interesting security problems need to be solved. In this thesis, we study two such problems, and provide efficient solutions for both.

In the foreseeable future, some e-commerce vendors will generate revenue by providing digital streaming applications such as information broadcasts (e.g., stock quotes). For the first issue, we investigate the problem of authenticating packet streams in multicast or broadcast networks. Our approach is to encode the hash values and digital signatures with Rabin's Information Dispersal Algorithm (IDA) to construct an authentication scheme that amortizes a single signature operation over multiple packets. This strategy is especially efficient in terms of space overhead because just the essential elements needed for authentication (i.e., one hash per packet and one signature per group of packets) are used in conjunction with an erasure code that is space optimal. We evaluate the performance of our scheme using both analytical and empirical results.

Applications such as e-commerce payment protocols using electronic money require that fair exchange be assured. For the second issue, we investigate the problem of constructing fair-exchange protocols. Our approach uses a novel signature paradigm—the gradational signature scheme—to construct protocols that are efficient and scalable. Unlike previous approaches, our scheme does not employ any

costly zero-knowledge proof systems in the exchange protocol. Use of zero-knowledge proofs is needed only in the protocol setup phase—this is a one-time cost. The resulting exchange protocol is more efficient than the previous solutions in terms of computation and communication overhead.

1. INTRODUCTION

1.1 Overview

The Web and the ever-increasing number of its users are revolutionizing the business world. By offering products and services on the Web, businesses are generating huge revenues without having to invest in costly brick-and-mortar storefronts, or pay the overhead costs required to maintain them. Besides its obvious competitive advantages for businesses, electronic commerce (e-commerce) provides customers with the convenience of being able to purchase just about any merchandise with just a few mouse clicks. These factors have made the e-commerce arena the ideal place to sell and buy products and services. In the near future, a wider breadth of items and services will be provided via e-commerce applications. For the successful deployment of these applications, varying degrees of security need to be ensured for all parties involved in the transactions. In this thesis, we investigate two security problems that are crucial to this topic.

The first problem deals with authenticating packet streams in multicast (or broadcast) networks. In the foreseeable future, some e-commerce vendors will generate revenue by providing digital streaming applications such as information broadcasts (e.g., stock quotes) and multiparty videoconferencing. These applications can be most efficiently handled by using multicast instead of unicast protocols. In multicast, a single copy of packets is sent by the sender and routed to every receiver within the multicast group via multicast-enabled routers. For most of these applications, authentication of the transmitted packets is needed; that is, receivers of the multicast group should be able to verify that the packets came from the stated source. A naive way of providing authentication is for the sender to digitally sign each individual packet. However, the computation cost of conventional signature schemes is too high to make this practical

for most streaming applications, and hence a more efficient scheme must be employed. One way of overcoming this obstacle is to amortize a single signature operation over multiple packets. In this thesis, we propose an efficient method of amortizing a single signature operation over multiple packets that is highly robust against packet loss. Our scheme employs Rabin's *Information Dispersal Algorithm* (*IDA*) [Rab89] to encode the hash values and digital signatures (i.e., the authentication information), and hence is appropriately named *Signature Amortization using IDA* (*SAIDA*). According to our empirical results, this strategy is very effective in authenticating packet streams in very high loss environments. In addition to simulation experiments, we develop analytical results concerning the performance of SAIDA. From this, we are able to deduce some very interesting facts about the relationship between the authentication probability and the space overhead required for the authentication information. Here, authentication probability is the conditional probability of being able to verify a given packet given that it has been received by the receiver. We also discuss techniques for making SAIDA robust against a certain type of denial-of-service (DoS) attacks.

The second problem deals with the construction of practical fair-exchange protocols for e-commerce. In a fair exchange, either each player gets the other player's item, or neither player does. In the foreseeable future, data with great intrinsic value, such as financial data, medical data, and electronic forms of money (e.g., electronic checks and electronic cash), will be exchanged regularly over the Internet. In such instances, ensuring fairness is critical if the participants are to be protected from fraud. As more business is conducted over the Internet, the fair-exchange problem is gaining greater importance. In this thesis, we propose a novel technique for constructing fair-exchange protocols that is efficient and scalable. For this purpose, we employ a novel signature paradigm called the *gradational signature* paradigm. The intrinsic features of our gradational signature paradigm enable us to construct protocols that do not require the use of zero-knowledge proof systems in the exchange protocol. This significantly simplifies the exchange protocol, and thus improves efficiency significantly. Zero-knowledge proof systems are used in the protocol setup phase, the

registration protocol, but this is only a one-time cost. Once the registration protocol has been successfully executed, it can support any number of exchanges. We also analyze the security of our fair-exchange protocol employing RSA-based gradational signatures. In our proof, we define a new RSA-related computational problem—the Additive Split Key RSA Single Target Inversion (ASK-RSA-STI) problem. This is an extension of the well-known RSA Single Target Inversion (RSA-STI) problem. We also discuss the security issues involved in splitting the RSA private key.

1.2 Contributions

In this thesis, we investigate two problems that are crucial to the security of certain e-commerce applications. The first problem involves authenticating packet streams in multicast (or broadcast) networks, and the second problem deals with the construction of fair-exchange protocols for e-commerce. Our focus is on finding practical solutions to these problems, and therefore performance and efficiency are the predominant factors that are considered.

Authenticating packet streams in multicast or broadcast networks is difficult primarily because of the following reasons: (i) authenticating a stream by digitally signing each packet is not practical because conventional digital signature algorithms are computationally too expensive; (ii) in general, packet loss rates in multicast sessions are much higher than unicast sessions. Our solution is to encode the hash values and the signatures with Rabin's Information Dispersal Algorithm (IDA) to construct an authentication scheme that amortizes a single signature operation over multiple packets. This strategy is very efficient in terms of computation and space overhead because just the essential elements needed for authentication (i.e., one hash per packet and one signature per group of packets) are used in conjunction with an erasure code that is space optimal.

According to our empirical results, our scheme—Signature Amortization using IDA (SAIDA)—achieves very high verification rates with minimal space overhead

required for the authentication information. Here, verification rate is defined as the number of verifiable packets of a given stream divided by the number of received packets of the same stream. Our simulation results also show that SAIDA is very robust against packet loss. That is, even when the loss rates are very high, SAIDA is able to maintain relatively high verification rates. This is not the case for the previously proposed stream authentication schemes that were examined.

In addition to empirical results, we develop analytical results on the performance of SAIDA. Because actual packet losses incurred in the Internet are highly bursty in nature, analyzing SAIDA under an independent packet loss assumption would not yield meaningful results. Therefore, we chose two bursty loss models in our analyses. Because the two loss models can be represented as irreducible, positive recurrent Markov chains, we are able to apply results from *renewal theory* in our analyses. The resulting analytical results reveal a surprising fact. It suggested that the authentication probability can be improved without increasing the space overhead required for the authentication information, if the block size of the IDA- encoding process is increased. Here, authentication probability refers to the conditional probability of being able to verify a packet given that it is received, and the block size refers to the number of packets processed by one instance of the IDA-encoding process.

It is well known that designing a fair-exchange protocol without employing a Trusted Third Party (TTP) is possible, but is too inefficient for practical uses. One can employ a secure server, executing the tasks of the TTP, to directly handle every exchange, but this approach suffers from fault-tolerance and scalability problems. For such reasons, many exchange protocols are devised with the goal of minimizing the role of the TTP. Protocols known as "optimistic protocols" achieve this goal by involving the TTP only when one of the exchanging parties behaves unfairly or aborts the protocol prematurely; otherwise, the TTP is never involved. Despite the many advantages of optimistic protocols, many (e-commerce) applications do not adopt such protocols because optimistic protocols often entail intricate cryptographic techniques that incur considerable computation overhead. The vast majority of optimistic

protocols use zero-knowledge proof systems that are, most often, the most expensive part of the exchange procedure. These proof systems require the prover and the verifier to compute multiple modular exponentiations of very large integers, and these operations are quite costly. Our approach achieves significant improvement over previous optimistic protocols by using a novel signature paradigm called the *gradational signature* paradigm. The intrinsic features of this signature paradigm enable us to construct fair-exchange protocols without resorting to zero-knowledge proof systems. The resulting protocols are very simple and efficient. In fact, they are among the most efficient optimistic protocols known to date. We do use zero-knowledge proof systems in the protocol setup phase, the registration protocol, but this is only a one-time cost. Once the registration protocol has been successfully executed, it can support any number of exchanges. In previous optimistic protocols, the zero-knowledge proof systems are directly employed in the exchange procedure itself. Therefore, the entities of the exchange had to generate and verify costly zero-knowledge proofs for every exchange.

As a practical application of optimistic fair-exchange protocols, we present a certified e-mail protocol. Using RSA-based gradational signatures, we are able to construct a "lightweight" certified e-mail protocol. This protocol requires less computation and communication overhead compared to previous protocols, which makes it especially suitable for the wireless mobile setting. The efficiency of our protocol directly translates to less power consumption and prolonged battery life for the mobile devices (e.g., cellular phones and PDAs) used in exchanging the e-mail messages.

1.3 Thesis Organization

In the next chapter, we investigate the problem of authenticating packet streams in multicast (or broadcast) networks. In the first section, we give some introductory information on the subject. Specifically, we define the problem, and discuss the technical challenges involved in authenticating multicast packet streams. In Section 2.2,

we briefly discuss related work, and give an overview of erasure codes. The rationale for our approach, along with the detailed authentication/verification procedure, is given in Section 2.3. It is widely known that actual packet losses in the Internet are correlated and bursty in nature, and thus the simple assumption of independent packet losses is not appropriate for the analytical evaluation of our authentication scheme. In Section 2.4, we define two bursty loss models—the 2-state Markov Chain (2-MC) loss model and the Biased Coin Toss (BCT) loss model. We characterize the asymptotic authentication probability of our scheme using the two loss models. A lower bound of the authentication probability is also derived for the BCT loss model. In Section 2.5, we evaluate the performance of our technique, comparing it with four other previously proposed schemes. In the comparison, we use the following criteria to evaluate the schemes: sender delay, receiver delay, computation overhead, communication overhead, and verification rate. Our basic authentication scheme can be vulnerable to a certain type of Denial-of-Service (DoS) attacks. Techniques for thwarting such DoS attacks are given in Section 2.6. Finally, concluding remarks for Chapter 2 are given in Section 2.7.

In Chapter 3, we study the problem of designing fair-exchange protocols for e-commerce. In the first section, we define the problem, and discuss the fundamental issues related to the fair-exchange problem. A brief overview of our approach is also given. In Section 3.2, we give a brief overview of optimistic fair-exchange protocols, and discuss related work. In addition, we discuss a zero-knowledge proof system for demonstrating the possession of discrete logarithms. This zero-knowledge proof system is employed in our fair-exchange protocols. The details of the gradational signature paradigm is explained in Section 3.3. The concept and features of the gradational signature paradigm are discussed. In the last subsection, we also describe how the paradigm can be applied to construct optimistic fair-exchange protocols. In Section 3.4, we give details on how to construct gradational signature schemes based on four well-known conventional signature algorithms. For two of the signature algorithms, their security is based on the intractability of factoring large integers,

and the other two's security is based on the intractability of the discrete logarithm problem. In Section 3.5, we construct optimistic fair-exchange protocols using the gradational signature paradigm. The efficiency of our protocol is compared with previously proposed protocols in Section 3.6. We analyze the security of one of our protocols in Section 3.7. In the same section, we also discuss the security issues involved in splitting RSA private keys. The concluding remarks for the chapter are given in Section 3.8.

As a practical application of optimistic fair-exchange protocols, we present a "lightweight" certified e-mail protocol in Chapter 4. In the first section, we discuss the concept of certified e-mail, and examine some of the challenges in designing efficient certified e-mail protocols. In Section 4.2, a brief overview of related work is given. The required properties of certified e-mail protocols are discussed in Section 4.3. The special properties that would be needed for certified e-mail protocols used in wireless mobile environments is also examined in the same section. In Section 4.4, our certified e-mail protocol is described in detail. We compare the efficiency of our protocol with previously proposed schemes in Section 4.5. We conclude the chapter in Section 4.6.

Final concluding remarks are given in Chapter 5. We summarize the thesis and discuss avenues for future research. Some interesting areas for future research include fault-tolerant data depositories and distributed trusted computing entities (e.g., certification authorities and trusted third parties for exchange protocols).

2. AUTHENTICATION OF PACKET STREAMS IN MULTICAST NETWORKS

2.1 Introduction

With the growing demand for novel types of group communications, multicast has received a lot of attention in recent years. In multicast, a single copy of packets is sent by the sender and routed to every receiver within the multicast group via multicast-enabled routers. For a wide range of applications, multicast is an efficient and natural way of communicating information. Some examples include information broadcasts (e.g., news feeds, weather updates, and stock quotes), multiparty video-conferencing, and software updates. For successful implementation, many of these applications will require varying degrees of security requirements (i.e., confidentiality and authentication).

Confidentiality for multicast transmissions can be provided using techniques that utilize symmetric (secret) key cryptography. Confidentiality would be provided by encrypting the message with the secret key being shared by the sender and the receivers of the multicast group before transmission. Consequently, off-the-shelf solutions such as the Advanced Encryption Standard (AES) can be readily employed for this purpose. For confidentiality, the main concern is the complexity involved in key management (e.g., key distribution, revocation, and group updates).

The solution to the authentication problem for unicast transmission is well known. For example, efficient hash-based message authentication codes (MAC) known as HMACs can be employed. However, this solution is inadequate for the multicast setting. The difficulty of the problem lies in the fact that preventing the forgery of packets by a colluding group of receivers precludes the use of any MAC-based scheme, if small communication (space) overhead is required, and time synchroniza-

tion between the sender and the receiver is difficult to maintain. The rationale for this argument is given in Subsection 2.2.1. Without using symmetric key cryptosystems, the most obvious way of providing authentication is to sign each individual packet using the sender's digital signature. However, the computation overhead of current signature schemes is too high to make this practical. According to [WoL98], a Pentium II 300 MHz machine devoting all of its processor time can only generate 80 512-bit RSA signatures per second. Clearly, this approach is not practical.

Packet loss is another important issue that needs to be considered. While this may not be a problem for applications using reliable transport protocols (e.g., Transmission Control Protocol (TCP)/ Internet Protocol (IP)), it is a serious issue for multicast applications that use the User Datagram Protocol (UDP) over IP-Multicast. Because UDP only provides best-effort service, packet loss can be high. Therefore, while the content being broadcast might be able to bear packet losses, the authenticity of a non-negligible percentage of the received packets might not be verifiable by the receiver, if the authentication scheme is not robust against loss.

Techniques for reliable multicast exist, such as the Scalable Reliable Multicast (SRM) [FlJ97] and the Reliable Adaptive Multicast Protocol (RAMP) [KoZ96], but they are complex and not yet standardized. Within the Internet Engineering Task Force (IETF), the Reliable Multicast Transport (RMT) working group is chartered to standardize reliable one-to-many transport protocols for bulk data. Because it is difficult for a single protocol to meet the requirements of all multicast applications, the RMT working group is standardizing three protocol instantiations, one from each of the following three categories: (i) a NACK-based protocol, (ii) a tree-based ACK protocol, and (iii) an asynchronous layered coding protocol that uses forward error correction (FEC). The Digital Fountain model [ByL98] is a scheme that belongs to the third category [LuV02], and provides reliable one-to-many transport of bulk data using Tornado codes [LuM97]. The standardization of reliable multicast protocols by the RMT working group is not yet complete. Moreover, dissemination of information

via broadcast (opposed to multicast) protocols incurs loss, and hence packet loss remains as an important issue in the authentication of packet streams.

Our approach to multicast message authentication is based on the technique of signature amortization, that is, amortizing a single signing operation over multiple packets. Our technique is especially efficient in terms of space overhead because just the essential elements needed for authentication (i.e., one hash per packet and one signature per block of packets) are used in conjunction with an erasure code that is space optimal. Note that our approach is similar, in concept, to the RMT working group's third category of protocol instantiations, but with the difference that we are using FEC techniques to encode authentication information and not the data itself.

We also show how our approach can readily be extended to combat denial-of-service (DoS) attacks, where the attacker inserts bogus packets into the message stream. This problem is raised in [LuV02] within the context of reliable multicast protocols. One solution is to implement additional functionality into the multicast router or the receiver's computer so that the source IP address can be screened against a list of authentic transmitter addresses. This solution can be easily defeated, however, if the attacker has the capability to spoof source addresses. The techniques discussed in Section 2.6 avoid this weakness.

In the next section, we briefly discuss related work and give an overview of erasure codes. The rationale for our approach, along with the detailed authentication and verification procedure, is given in Section 2.3. In Section 2.4, we derive the asymptotic authentication probability of our scheme using two different bursty loss models. A lower bound of the authentication probability is also derived for one of the loss models. In Section 2.5, we evaluate the performance of our technique, comparing it with four other previously proposed schemes. Techniques for thwarting DoS attacks are given in Section 2.6. The concluding remarks for this chapter are given in Section 2.7. Portions of material from this chapter were published in [PaC02, PaC03a].

2.2 Technical Background

2.2.1 Related Work

Multicast authentication schemes can be divided into two major classes—unconditionally secure authentication and computationally secure authentication. Pioneering research on unconditionally secure authentication was done by Simmons [Sim84], and later extended to the multicast setting by Desmedt et al. [DeF92]. As the name suggests, unconditionally secure authentication provides very strong security guarantees, but is less practical than computationally secure techniques. Our focus is on computationally secure methods.

In computationally secure multicast authentication, two approaches can be taken. One approach is to use Message Authentication Codes (MAC), and the other is to utilize digital signatures. Schemes based on MACs do not require any computationally intensive computations. The *asymmetric MAC* scheme proposed by Canetti et al. [CaG99] and the *Timed Efficient Stream Loss-tolerant Authentication* (*TESLA*) proposed by Perrig et al. fall into this category. The asymmetric MAC scheme is vulnerable to collusion attacks and is not scalable because the size of the authentication information increases as the number of receivers increases. Using TESLA, packet streams can be authenticated with a small communication overhead, but this method requires time synchronization between the sender and the receivers, which might not be feasible in large multicast groups. In their recent work, Boneh et al. [BoD01] showed that one cannot build an efficient (in terms of communication overhead) collusion resistant multicast authentication scheme solely based on MACs. Even though TESLA is resistant to collusion attacks and its communication overhead is small, the result of Boneh et al. is still valid because TESLA uses a digital signature to "bootstrap" the first packet of an authentication chain.

If MAC-based techniques are not employed, the obvious alternative is to use digital signatures. The biggest challenge in using digital signatures for authentication is the computationally intensive nature of the asymmetric-key-based signatures. For

this reason, previous authentication schemes approached this problem in two directions: designing faster signature techniques and amortizing a signature operation over multiple packets.

In general, making the signature scheme faster comes at the cost of increased space overhead. In [Roh99], Rohatgi proposes using a combination of k-time signatures and certificates for the k-time public keys (created with a regular signature scheme) to authenticate packets. Despite its improvement over the one-time signature scheme, this method still requires a space overhead on the order of several hundred bytes per packet.

Perrig [Per01] proposes a one-time signature scheme called BiBa, which has two important advantages over regular signature schemes: its signature generation and verification times are significantly faster. This implies that each packet can be individually signed (via a BiBa signature) at a rate that is fast enough to match the packet transmission rate. By individually signing each packet, there is no need to buffer packets for authentication or verification, and thus this reduces delay. Hence, for applications like the broadcast of stock quotes, which requires the authentication of real-time data, the broadcast authentication protocol based on the BiBa signature scheme might be appropriate. However, BiBa also has limitations. Its security is dependent on the time synchronization maintained between the sender and the receivers, and the maximum allowable time synchronization error between the sender and the receivers, which we denote as τ, limits the authentication rate. Because τ has to be set at a feasible value, there is a practical limitation on the authentication rate. For applications that require the sender to authenticate packets that have relatively fast transmission rates (e.g., video streams), this can be a problem. Another point to consider is the size of the communication overhead created by the BiBa authentication protocol. With reasonable parameter values, each BiBa signature size is 100–200 bytes, and a new set of public keys, whose size is on the order of 10 KB, needs to be transmitted at regular intervals.

Many schemes based on digital signatures attempt to reduce the computation and communication overhead by amortizing a single signature operation over multiple packets. For example, Wong and Lam [WoL98] employ Merkle's *authentication trees* [Mer89] to reduce the size of the authentication information, and sign multicast packets. However, because each packet is individually verifiable, every packet contains a copy of a digital signature along with other authentication information (e.g., hash values), which requires a large communication (or space) overhead.

If individual packet verification is relaxed so that the verification of a packet is dependent on other packets, then the communication overhead can be reduced substantially. In this approach, verification of each packet is not guaranteed and instead is assured with a certain probability. The scheme proposed by Perrig et al., called *Efficient Multi-chained Stream Signature* (*EMSS*) [PeC00], takes this approach, and uses a combination of hash functions and digital signatures to authenticate packets. EMSS is an extension of Gennaro and Rohatgi's *stream signing* technique [GeR97]. The basic idea is as follows: A hash of packet P_1 is appended to packet P_2, whose hash is in turn appended to P_3. If a *signature packet*, containing the hash of the final data packet (i.e., P_3) along with a signature of the hash, is sent after P_3, then nonrepudiation is achieved for all three packets. In essence, hash values act as chains (or directed edges) between the packets so that they form a single string that can be signed by one digital signature. To reduce the verification delay, a stream of packets is divided into blocks, and the above process is repeated for every block. EMSS achieves relatively high verification rates at the cost of increased space overhead and delayed verification. Throughout the thesis, we will use the term *verification rate* to represent the number of verifiable packets of a given stream divided by the number of received packets of the same stream.

Golle and Modadugu [GoM01] propose a scheme similar to EMSS called the *augmented chain* technique. They propose a systematic method of inserting hashes in strategic locations so that the chain of packets formed by the hashes will be resistant to a burst loss. The chain of packets constructed using this method is parameterized

by the integer variables a and p; a chain of packets can sustain a burst loss of up to $p(a-1)$ packets in length. To reduce the verification delay, a stream is divided into blocks of packets, and each block is constructed using the augmented chain technique with only the last packet in the block being signed.

In [MiS01], Miner and Staddon propose a similar authentication scheme, based on hash chaining techniques, specifically designed to resist multiple bursts. In the construction of their scheme, called the *piggybacking* scheme, n packets are partitioned into r equal-sized priority classes. The signature packet is the first packet in the highest priority class. It is assumed that the packets in the highest priority class are spaced evenly throughout the block so that two consecutive packets of the highest priority class are located exactly r packets apart. By constructing the hash chains using the piggybacking scheme, packets in the i-th priority class can tolerate x_i bursts of size at most $b_i = k_i r$, where k_i is a parameter dependent on the configuration of the hash chains.

In their recent work, Pannetrat and Molva [PaM02] propose a stream authentication scheme that is similar to our approach (initially published in [PaC02]). They propose to authenticate real-time data streams by piggybacking the current block's encoded authentication information (via an erasure code) onto the next block. Using this method, no packets need to be buffered by the sender, but, of course, additional buffering is needed at the receiver. The same technique can be readily applied to our authentication scheme.

2.2.2 Erasure Codes

When a stream of packets is sent via the Internet, a fraction of the packets are lost during transit. A standard solution to this problem is to retransmit data that is not received. In some applications, this solution is not practical. An alternative solution is to use forward error correction (FEC) techniques. Among FEC codes, *erasure codes* are of particular interest to our application. We briefly review the basic

characteristics of two well-known erasure codes—*Information Dispersal Algorithm* (*IDA*) [Rab89] and *Tornado codes* [LuM97].

IDA was originally developed as an algorithm for providing reliable storage or transmission of information in distributed systems. The basic idea of IDA is to process the source, say a file F, by introducing some amount of redundancy, and splitting the result into n pieces, which are then transmitted. Reconstruction of F is possible with any combination of m pieces, where $m \leq n$. Each distributed piece is of size $|F|/m$, which clearly shows that the scheme is space optimal. The space overhead can be controlled by adjusting the parameters n and m—the ratio n/m determines the space overhead incurred by the encoding process.

Unlike IDA, Tornado codes can encode and decode data in linear time. The number of segments needed for decoding is slightly larger than the number of preencoded segments, and thus they are sub-optimal in terms of space overhead. Specifically, for a set of n segments, encoding with Tornado codes increase the number to

$$\frac{n}{1-p(1+\epsilon)},$$

where $0 < p < 1$ and $0 < \epsilon < 1$. If the receiver acquires more than $1-p$ fraction of the encoded segments, then the original data segments can be reconstructed with high probability in time proportional to $n \ln(1/\epsilon)$.

2.3 Our Scheme for Stream Authentication in Multicast Networks

2.3.1 The Rationale for Our Approach

In EMSS, there are three factors that affect the verification rate—number of signature packets, number of hashes contained in the signature packet, and number of hashes contained in the data packet. The number of hashes in the signature and data packets is always greater than one, improving the robustness to packet loss at the cost of increased communication overhead. This tradeoff is unavoidable, but can be done

more efficiently using erasure codes. This is best illustrated using a simple example in the following paragraph.

The sender transmits the hash of a packet appended to k other packets for increased resistance to loss. We assume that a block consists of n packets, and packet loss is independent. The probability that at least one out of the k packets will reach the destination is $1 - q^k$, where q is the loss probability. The space overhead would be $k \cdot h$, where h is the size of the hash. Using the same overhead, one can encode the hash using IDA and append the n encoded segments to n packets. The minimum number of encoded segments needed for reconstruction of the hash is only $m = \lceil n/k \rceil$, where $\lceil x \rceil$ denotes the smallest integer not less than x. The probability that the hash can be reconstructed successfully at the receiver is given by

$$1 - \sum_{i=0}^{m-1} \left(\begin{array}{c} n \\ i \end{array} \right) (1-q)^i q^{n-i}. \tag{2.1}$$

Instead of using IDA, the sender could also use probabilistic codes such as Tornado codes. In this case, the minimum number of segments needed for decoding is

$$m = n(1-p) = \left\lceil \frac{n}{k} \left(1 + \frac{\epsilon(k-1)}{\epsilon+1} \right) \right\rceil, \tag{2.2}$$

where p and ϵ are the performance parameters of the Tornado code (see Subsection 2.2.2). The probability of successful reconstruction of the hash is given by (2.1) using the value of m specified by (2.2). It is obvious that the probability given by (2.1) is higher than $1 - q^k$, and the probability for IDA is higher than that for the Tornado code.

The above example suggests that using an erasure code to encode the hash values would be more efficient than simply appending duplicate hashes to the packets. As an extended version of EMSS, Perrig et al. [PeC00] suggest using universal hash functions or IDA to split the packet's hash value into multiple pieces before appending them onto other packets. This certainly produces a more loss-resistant scheme with the same amount of communication overhead. However, it introduces complexities—the time-consuming process of encoding and decoding must be performed for each hash.

This can be a computational bottleneck, especially when multiple hashes are used per packet. We suggest a simple method of avoiding this problem at the cost of sender delay. Instead of encoding individual hashes, we suggest concatenating all the hashes within the block to form a single piece before encoding. This strategy requires only one encoding/decoding per block.

The main advantage of EMSS is that there is no sender delay incurred by the authentication process—a given packet can be transmitted immediately after its hash value is computed without the need to buffer other packets. This can be an advantage in situations where data is generated in real time, and immediate dissemination is crucial. However, for some multicast applications, the sender has a priori knowledge of at least a portion of the data (e.g., prerecorded news footage), and some sender delay is tolerated. In fact, most authentication schemes incur some degree of sender delay (see [GoM01, MiS01, WoL98]).

If a certain amount of sender delay is allowed, then a more significant problem can be addressed. It is obvious that the delivery of the signature packets is crucial for any authentication scheme. In most of the previous work [PeC00, GoM01, MiS01], performance results (both analytical and empirical) are based on the assumption that the signature packets are received. The authors suggest accomplishing this task by empowering the receivers to request retransmissions of the lost signature packets or sending multiple copies of the signature packet. However, the retransmission of signature packets can put considerable strain on the resources of the sender and the network, especially in large multicast networks. In [YaK96], Yajnik et al. observe packet loss characteristics of actual multicast sessions, and show that considerable amounts of the packets would need to be retransmitted, if reliable multicast services are to be provided through retransmissions. In one particular data set, 62.6% of the packets is lost by at least one receiver. This implies that retransmission would have been necessary for 62.6% of the packets.

Sending multiple copies of the signature packet can be an alternate solution, but this also has drawbacks. Signature packets are generally large (e.g., 128 bytes, if

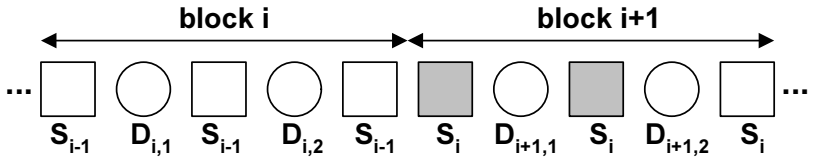

Fig. 2.1. Receiver delay caused by sending multiple copies of the signature packet.

1024-bit RSA is used), and sending these packets several times can increase the communication overhead noticeably. Moreover, because actual losses in the Internet are highly correlated and bursty, each copy of the signature packet would have to be interspersed uniformly among the packets to ensure maximum effectiveness. If copies of the signature packets are distributed in the current block, then this would cause sender delays in schemes that utilize hash-chaining techniques with edges directed backwards (i.e., hash of a packet is appended to the packets following it)—schemes like EMSS. The sender delay is incurred because data packets in the block cannot be transmitted before the replicas of the signature packet are interspersed among the data packets.

The obvious alternative is to distribute the copies in the next block to avoid the sender delay. However, this can cause increased delay on the receiver side—a receiver might have to buffer a maximum of $2n$ data packets before verifying a given packet, where n is the number of data packets per block. This case is illustrated as a simple example in Figure 2.1. In the figure, circles and squares represent data packets and signature packets, respectively. The first two signature packets in the $(i+1)$-th block are assumed to be lost and are represented as darkened squares. The receiver needs to buffer $D_{i,1}$, $D_{i,2}$, $D_{i+1,1}$, and $D_{i+1,2}$ before verifying the data packets of the i-th block (i.e., $D_{i,1}$ and $D_{i,2}$) using S_i, which is the signature packet of the i-th block. Considering these problems, the obvious alternative is to apply FEC techniques to the signature packets. We can easily make the signature packets robust against packet loss by using erasure codes and appending each encoded piece to the data packets. For our authentication scheme, we employ IDA instead of probabilistic codes such

as Tornado codes. Tornado codes can encode/decode data very rapidly (i.e., linear time), but do so with a high probability only when the number of segments being encoded is large. For this reason, Tornado codes are appropriate when a large number (i.e., hundreds to thousands) of segments are being encoded [WeW01]. We use IDA because the encoding involves a fairly small number of segments, and IDA has the added advantage of being space optimal. It should be noted that our authentication scheme is independent of the type of the erasure code, and other erasure codes (e.g., Tornado codes) that have other attractive properties can be employed.

2.3.2 The Authentication Procedure

To reduce the computation burden of signing each packet, two approaches can be taken: designing faster signature techniques, and amortizing a single signature operation over multiple packets. Our main focus is reducing the size of the authentication overhead, and therefore we took the second approach, which offers better space efficiency. We propose a scheme that employs IDA to amortize a single digital signature over a block of packets. Our scheme, appropriately named *Signature Amortization using IDA* (*SAIDA*), is designed to provide space-efficient authentication even in high packet-loss environments. The following steps describe the authentication procedure in detail:

1. Let $\|$ denote concatenation. A stream of packets is first divided into groups (or blocks). We represent a stream as

$$\Gamma = G_1 \parallel G_2 \parallel \cdots ,$$

   where each group G_i is a concatenated string of n packets (i.e., $G_i = P_{(i-1)n+1} \parallel \cdots \parallel P_{i \cdot n}$), and each packet $P_i \in \{0,1\}^\kappa$ for some constant κ. The same operations are performed on every group, so we will only focus on the first group.

2. A *packet hash* $H(P_i), i = 1, \ldots, n$ for each packet is computed using a hash function H.

3. The packet hashes are concatenated to form the string

$$F^1 = H(P_1) \parallel \cdots \parallel H(P_n)$$

of size N (i.e., F^1 consists of N characters). Let c_i represent the i-th character in F^1. In practice, c_i may be considered as an eight-bit byte, and all calculations are done in $\mathbb{Z}_{257}$ or $GF(2^8)$. One copy of F^1 is stored in a buffer, while another copy is divided into blocks of length m as follows:

$$F^1 = (c_1, \ldots, c_m), (c_{m+1}, \ldots, c_{2m}), \ldots, (c_{N-m+1}, \ldots, c_N).$$

To simplify the following discussion, let $\boldsymbol{S}_i = (c_{(i-1)m+1}, \ldots, c_{im}), 1 \leq i \leq N/m$.

4. Choose n vectors $\boldsymbol{a}_i = (a_{i1}, \ldots, a_{im}), 1 \leq i \leq n$, such that every subset of m different vectors are linearly independent, as specified in [Rab89].

5. Using vectors $\boldsymbol{a}_i = (a_{i1}, \ldots, a_{im}), 1 \leq i \leq n$, F^1 is processed and divided into n segments as

$$F_i^1 = (\boldsymbol{a}_i \cdot \boldsymbol{S}_1, \boldsymbol{a}_i \cdot \boldsymbol{S}_2, \ldots, \boldsymbol{a}_i \cdot \boldsymbol{S}_{N/m}), i = 1, \ldots, n,$$

where $\boldsymbol{a}_i \cdot \boldsymbol{S}_k = a_{i1} \cdot c_{(k-1)m+1} + \cdots + a_{im} \cdot c_{km}$. It follows that $|F_i^1| = |F^1|/m$.

6. The *group hash* is computed by taking the hash of the other copy of F^1, that is,

$$H_G(G_1) = H(F^1) = H(H(P_1) \parallel \cdots \parallel H(P_n)),$$

where $H_G(G_1)$ denotes the group hash of the first group of packets.

7. The group hash is signed by a digital signature scheme using the sender's private key K_r, and denoted as $\sigma(K_r, H_G(G_1))$. This value is IDA encoded using the same set of vectors to yield

$$\sigma_1(K_r, H_G(G_1)), \ldots, \sigma_n(K_r, H_G(G_1)).$$

Although this procedure was explained as a separate step for clarity, the signature can be concatenated with F^1 before applying IDA, so that only one encoding procedure per group is necessary.

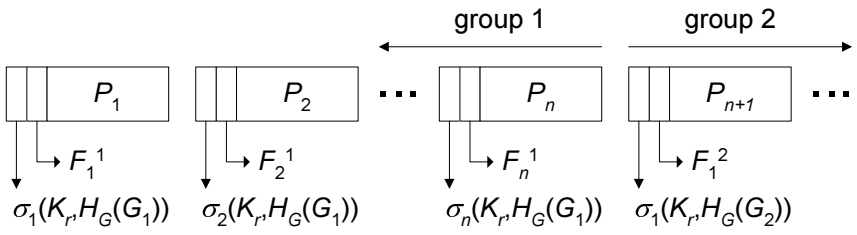

Fig. 2.2. Authenticated packet stream.

8. Each signature segment (created in the seventh step) and hash segment (created in the fifth step) are concatenated with the corresponding packet to form an authenticated packet. A group of n authenticated packets combine to form an authenticated group, which is expressed as

$$\sigma_1(K_r, H_G(G_1)) \parallel F_1^1 \parallel P_1, \ldots, \sigma_n(K_r, H_G(G_1)) \parallel F_n^1 \parallel P_n.$$

An instance of an authenticated packet stream is illustrated in Figure 2.2. The verification of the stream is straightforward. Assuming that at least m packets are received, the receiver can successfully reconstruct F^1 and $\sigma(K_r, H_G(G_1))$ from any combination of m packets as follows:

1. Assume that segments $F_1^1, \ldots, F_m^1$ are received. Using the m pieces, it is readily seen that

$$\mathbf{A} \cdot \begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix} = \begin{pmatrix} \boldsymbol{a}_1 \cdot \boldsymbol{S}_1 \\ \vdots \\ \boldsymbol{a}_m \cdot \boldsymbol{S}_1 \end{pmatrix},$$

where $\mathbf{A} = (a_{ij})_{1 \le i,j \le m}$ is an $m \times m$ matrix whose i-th row is $\boldsymbol{a}_i$.

2. Because $\mathbf{A}$ is invertible, $\boldsymbol{S}_1$ can be obtained from

$$\boldsymbol{S}_1 = \begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix} = \mathbf{A}^{-1} \cdot \begin{pmatrix} \boldsymbol{a}_1 \cdot \boldsymbol{S}_1 \\ \vdots \\ \boldsymbol{a}_m \cdot \boldsymbol{S}_1 \end{pmatrix}.$$

3. Using the same procedure, $\boldsymbol{S}_2, \ldots, \boldsymbol{S}_{N/m}$ can be obtained, and F^1 is reconstructed by concatenating these values.

4. The same techniques are applied to reconstruct $\sigma(K_r, H_G(G_1))$.

5. All the packets in G_1 can be verified using F^1 and $\sigma(K_r, H_G(G_1))$.

For SAIDA, the trade-off between verification rate and communication overhead can be readily governed by changing the parameters n (number of encoded segments) and m (minimum number of encoded segments needed for decoding). Note that the space overhead, which is determined by n/m, only applies to the authentication information (i.e., hashes and digital signatures) and not to the data itself. Specifically, $A(n/m)$ represents the space overhead incurred after the IDA encoding process, where A is the size of the authentication information.

2.4 The Authentication Probability

2.4.1 Loss Models

It is well known that actual packet losses in networks are bursty rather than independent in nature. In this section, using asymptotic techniques, we derive the *authentication probability* of SAIDA using two different bursty loss models. We define the authentication probability as $\Pr\{P_i \text{ is verifiable} | P_i \text{ is received}\}$, where P_i represents the i-th packet of a block. This definition is adopted from [MiS01]. Note that this is different from the *verification rate* referred to in Subsection 2.2.1. We use the term verification rate to represent the number of verifiable packets of a given stream divided by the number of received packets of the same stream. In the simulation experiments of Section 2.5, we use the verification rate as the performance measure because its value can be directly calculated from the simulation data.

The authentication probability is directly affected by the loss behavior of the network. In [YaM99], it is shown that the 2-state Markov chain can accurately model bursty loss patterns in certain cases, and hence we adopt this model as one of our loss

models. Throughout the thesis, we denote this model as the *2-state Markov Chain* (*2-MC*) loss model.

In [MiS01], the authors derive an authentication probability lower bound (for their piggybacking scheme) based on a bursty loss model, which is motivated by ideas in the theory of error-correction codes. We call this model the *Biased Coin Toss* (*BCT*) loss model. We choose this model for our analyses in Subsection 2.4.3 and Subsection 2.4.4 because it facilitates the direct comparison of the authentication probability lower bound between SAIDA and the piggybacking scheme.

For fixed n and m, finding the analytical expression for the authentication probability seems to be extremely difficult, and hence we use asymptotic techniques in our analyses. In the analyses, results from *renewal theory* are applied.

2.4.2 The Asymptotic Authentication Probability Under the 2-State Markov Chain Model

In this subsection, we derive the asymptotic authentication probability of SAIDA under the 2-state Markov (2-MC) loss model. First, we need to define the 2-MC loss model.

Definition 2.4.1 2-MC loss model. *The loss process is modeled as a discrete-time Markov chain with two states—*0 *and* 1*—representing no loss and loss, respectively. It is defined by the four transition probabilities* (*i.e.,* p_{00}*,* p_{11}*,* p_{01}*, and* p_{10})*. The stationary probabilities* (*i.e., the long-run proportion of transitions that are into a given state*) *are denoted as* π_0 *and* $\pi_1 = 1 - \pi_0$*. The expected burst-loss length* β*, and probability of loss* q *can be expressed using the four parameters of the Markov chain.*

We represent this loss process as a discrete-time binary time series $\{S_i\}_{i=1}^{i=\infty}$ taking values in the set $\{0, 1\}$. Before deriving the authentication probability, we need the following lemmas. To express our main analytical result (i.e., Proposition 2.4.1), it is convenient to represent the four transition probabilities in terms of the stationary probabilities and β.

Lemma 2.4.1 *The four transition probabilities can be expressed using β, π_0, and π_1 as follows:*

$$p_{10} = \frac{1}{\beta},\ p_{01} = \frac{\pi_1}{\beta\pi_0},\ p_{11} = 1 - \frac{1}{\beta},\ p_{00} = 1 - \frac{1}{\beta}\left(\frac{1}{\pi_0} - 1\right).$$

Proof Let $\boldsymbol{B}$ be a random variable representing the length of a consecutive string of losses in steady state. Then the expected burst length is

$$\beta = E[\boldsymbol{B}] = \frac{\mu_{00} - 1}{p_{01}},$$

where μ_{ij} denotes the expected number of transitions until the chain enters state j given that it is presently in state i. The value of μ_{00} can be written as an infinite sum of kf_{00}^k, where f_{ij}^n denotes the probability that, starting in i, the first transition into j occurs at time n. Hence,

$$\begin{aligned}
\mu_{00} &= \sum_{k=1}^{\infty} k f_{00}^k \\
&= p_{00} + p_{01}p_{10}\sum_{k=0}^{\infty} p_{11}^k(k+2) \\
&= 1 + \frac{p_{01}}{p_{10}}.
\end{aligned}$$

Substituting this value for μ_{00} in the above expression for β, we get $\beta = 1/p_{10}$. It follows that $p_{10} = 1/\beta$. Now, using the relation between stationary probabilities π_0 and π_1, we obtain

$$\begin{aligned}
\pi_0 &= \pi_0 p_{00} + \pi_1 p_{10} \\
&= \pi_0(1 - p_{01}) + \pi_1 p_{10}.
\end{aligned}$$

Substituting the value $1/\beta$ for p_{10}, and solving for p_{01}, we obtain $p_{01} = \pi_1/(\beta\pi_0)$. The remaining transition probabilities p_{00} and p_{11} can be obtained from the relation $p_{00} = 1 - p_{01}$ and $p_{11} = 1 - p_{10}$, respectively. ■

The results of the following lemma is needed to prove Lemma 2.4.3.

Lemma 2.4.2 *Let $\boldsymbol{T}$ denote the number of transitions between successive visits to state* 1*. Then the following holds:*

$$E[\boldsymbol{T}^2] = \frac{2\pi_0}{\pi_1}\left(\frac{1}{p_{01}}\right) + \frac{1}{\pi_1}.$$

Proof The 2-MC loss model is an irreducible, positive recurrent Markov chain, and hence visits to a given state constitute a *renewal process*. Hence, visits to state 1 is a (renewal) event, and a new cycle begins with each visit to state 1. By the theory of renewal reward processes, the long-run average reward per unit time is equal to the expected reward earned during a cycle divided by the expected time of a cycle. We can form a renewal reward process by imagining that a reward is given at every time instant that is equal to the number of transitions from that time onward until the next visit to state 1. Then the expected reward earned during a cycle divided by the expected time of a cycle is given by

$$\begin{aligned}\frac{E[\boldsymbol{T} + (\boldsymbol{T}-1) + (\boldsymbol{T}-2) + \cdots + 1]}{E[\boldsymbol{T}]} &= \frac{E[\boldsymbol{T}^2 + \boldsymbol{T}]}{2E[\boldsymbol{T}]} \\ &= \frac{E[\boldsymbol{T}^2]}{2E[\boldsymbol{T}]} + \frac{1}{2}.\end{aligned}$$

Because the long-run average reward per unit time is the same as the average number of transitions it takes to transition into state 1, it follows that

$$\frac{E[\boldsymbol{T}^2]}{2E[\boldsymbol{T}]} + \frac{1}{2} = \sum_{i=0}^{1} \pi_i \mu_{i1}. \tag{2.3}$$

Solving for $E[\boldsymbol{T}^2]$, and using the fact that $E[\boldsymbol{T}] = \mu_{11} = 1/\pi_1$, we obtain

$$E[\boldsymbol{T}^2] = \frac{2\pi_0\mu_{01} + 1}{\pi_1}. \tag{2.4}$$

By conditioning on the next state visited, we obtain $\mu_{01} = 1 + p_{00}\mu_{01}$, which can be solved to obtain $\mu_{01} = 1/p_{01}$. Substituting this result into (2.4) gives the desired result. ■

Now, we are ready to prove Lemma 2.4.3.

Lemma 2.4.3 *Define $N(n)$ as the number of visits to state 1 by time n. Then for all k*

$$\Pr\left\{\frac{N(n)-\eta}{\sigma} < k\right\} \to \Phi(k) \quad \text{as } n \to \infty,$$

where $\eta = n\pi_1$, $\sigma^2 = \pi_1\pi_0 n(2\beta\pi_0 - 1)$, and $\Phi(k)$ is the standard normal distribution function.

Proof Visits to state 1 constitute a renewal event, and hence $N(n)$ is a delayed (general) renewal process. By Theorem 3.3.5 of [Ros96], the following holds:

$$\Pr\left\{\frac{N(n)-\eta}{\sigma} < k\right\} \to \Phi(k) \quad \text{as } n \to \infty,$$

where $\eta = n/E[\boldsymbol{T}]$, and $\sigma^2 = n\text{Var}(\boldsymbol{T})/(E[\boldsymbol{T}])^3$. Using the relation $E[\boldsymbol{T}] = \mu_{11} = 1/\pi_1$, we obtain $\eta = n\pi_1$. The variance of $\boldsymbol{T}$ is obtained by using the results of Lemma 2.4.2:

$$\begin{aligned}\text{Var}(\boldsymbol{T}) &= E[\boldsymbol{T}^2] - \frac{1}{\pi_1^2} \\ &= \frac{2\pi_0\pi_1 + p_{01}(\pi_1 - 1)}{p_{01}\pi_1^2}.\end{aligned}$$

Using $\text{Var}(\boldsymbol{T})$ (given above) and $E[\boldsymbol{T}] = 1/\pi_1$, we obtain

$$\sigma^2 = \pi_1 n \frac{2\pi_0\pi_1 + p_{01}(\pi_1 - 1)}{p_{01}}.$$

Substituting p_{01} with the value derived in Lemma 2.4.1, we obtain

$$\sigma^2 = \pi_1\pi_0 n(2\beta\pi_0 - 1),$$

and the desired result follows. ■

Note that because σ^2 is nonnegative, we conclude that $\beta \geq 1/(2\pi_0)$. In the following corollary, we generalize Lemma 2.4.3 to include the case when k (on the left-hand side of the equation) is a function of n.

Corollary 2.4.1 *Define*

$$F_{N(n)}(x) = \Pr\{(N(n) - \eta)/\sigma < x\}.$$

Let $g(n)$ denote some function of n, and define k such that $\lim_{n\to\infty} g(n) = k$. Then, the following holds:

$$F_{N(n)}(g(n)) \to \Phi(k) \ \ as\ n \to \infty.$$

Proof Fix $\epsilon > 0$. Then there exists an N such that for all $n \geq N$, $|g(n) - k| < \epsilon$, that is, $k - \epsilon < g(n) < k + \epsilon$. Hence, for all $n \geq N$,

$$F_{N(n)}(k - \epsilon) \leq F_{N(n)}(g(n)) \leq F_{N(n)}(k + \epsilon)$$

because $F_{N(n)}$ is a monotonically increasing function. Applying Lemma 2.4.3 to the above inequalities, we obtain

$$\Phi(k - \epsilon) \leq \liminf_{n\to\infty} F_{N(n)}(g(n)) \leq \limsup_{n\to\infty} F_{N(n)}(g(n)) \leq \Phi(k + \epsilon).$$

These inequalities hold for arbitrarily small ϵ, so by the continuity of Φ, we obtain the desired result. ■

Now we state our main result: derivation of the asymptotic authentication probability for SAIDA assuming the 2-MC loss model. As before, it is assumed that a block (or group) consists of n packets, and the minimum number of encoded segments required for decoding is m. The result stated by Proposition 2.4.1 holds for $n \to \infty$. To obtain a nontrivial result, we let m increase as n becomes large, but in such a way that it grows more slowly than n. Specifically, we let $n \to \infty$, and $m = n\pi_0 - \gamma\sqrt{n}$ for some fixed constant γ.

Proposition 2.4.1 *The authentication probability of the i-th packet in the block is given by the following expression when the minimum number of encoded segments required for decoding is $m = n\pi_0 - \gamma\sqrt{n}$:*

$$\Pr\{P_i \text{ is verifiable} | P_i \text{ is received}\} \to \Phi(k) \ \text{ as } n \to \infty,$$

where

$$k = \frac{\gamma}{\sqrt{\pi_1\pi_0(2\beta\pi_0 - 1)}}.$$

Proof Define the renewal process $\{N(0), N(1), \ldots\}$ as follows: $N(0) = 0$, and $N(n-i)$ is the number of packet losses between P_{i+1} and P_n, where P_n represents the last packet in a block (or group).

We can see that $\Pr\{P_i \text{ is verifiable}|P_i \text{ is received}\}$ is lower bounded by the probability of the number of packet losses between P_{i+1} and P_n being less than $n - m - (i-1) + 1$. This is because having at most $n - m - (i-1)$ losses after P_i guarantees that we can verify P_i regardless of what happened before P_i. Hence,

$$\Pr\{P_i \text{ is verifiable}|P_i \text{ is received}\} \geq \Pr\{N(n-i) < n - m - (i-1) + 1\}.$$

Note that if $i \geq n - m + 2$, then the above inequality holds trivially. Now, let

$$y = \frac{\gamma\sqrt{n} + i(\pi_1 - 1) + 2}{\sqrt{\pi_1\pi_0(n-i)(2\beta\pi_0 - 1)}}.$$

Then,

$$\Pr\left\{\frac{N(n-i) - \pi_1(n-i)}{\sqrt{\pi_1\pi_0(n-i)(2\beta\pi_0 - 1)}} < y\right\} = \Pr\{N(n-i) < n - m - (i-1) + 1\}. \quad (2.5)$$

Now, using a similar approach, we find the upper bound. The authentication probability is upper bounded by the probability of the number of packet losses between P_{i+1} and P_n being less than $n - m + 1$. This is because the verification of P_i implies that at most $n - m$ packet losses can be tolerated after P_i. Hence,

$$\Pr\{P_i \text{ is verifiable}|P_i \text{ is received}\} \leq \Pr\{N(n-i) < n - m + 1\}.$$

Note that if $i \geq m$, then the above inequality holds trivially. Now, let

$$z = \frac{\gamma\sqrt{n} + i\pi_1 + 1}{\sqrt{\pi_1\pi_0(n-i)(2\beta\pi_0 - 1)}}.$$

Then,

$$\Pr\left\{\frac{N(n-i) - \pi_1(n-i)}{\sqrt{\pi_1\pi_0(n-i)(2\beta\pi_0 - 1)}} < z\right\} = \Pr\{N(n-i) < n - m + 1\}. \quad (2.6)$$

Hence, the authentication probability is lower bounded by (2.5) and upper bounded by (2.6). Define

$$k = \frac{\gamma}{\sqrt{\pi_1\pi_0(2\beta\pi_0 - 1)}}.$$

Comparing the values of y, z, and k for fixed i and $n \to \infty$, it follows that

$$\lim_{n\to\infty} y = \lim_{n\to\infty} z = k.$$

Therefore, the lower bound and the upper bound for the authentication probability are asymptotically the same (for fixed i and $n \to \infty$), and the following holds by Corollary 2.4.1:

$$\Pr\{P_i \text{ is verifiable} | P_i \text{ is received}\} \to \Phi(k) \text{ as } n \to \infty.$$

■

Proposition 2.4.1 reveals an interesting relationship between n, n/m, and the authentication probability. According to the proposition, if π_0, β, and γ are constant, then the asymptotic authentication probability is also constant. Suppose that $\gamma > 0$, which is equivalent to $\pi_0 > m/n$. This corresponds to the case when the asymptotic authentication probability is greater than 0.5. Because $m = n\pi_0 - \gamma\sqrt{n}$, increasing n (while keeping γ constant) increases m in such a way that the value of n/m decreases. This behavior can be observed in Figure 2.3, where n/m versus n is plotted for $\gamma = 1$ and $\pi_0 = 0.8$. The value of n/m has significance because it determines the space overhead caused by the IDA encoding process—a decrease in n/m results in a decrease in space overhead. To elaborate, this means that for $\gamma > 0$, by increasing n, a constant (asymptotic) authentication probability can be maintained while decreasing n/m. This suggests that the authentication probability can be improved, while a constant space overhead is maintained, by increasing the number of packets per block. This hypothesis was confirmed using simulations. Figure 2.4 shows the change in authentication probability as n is increased (while keeping $n/m = 1.5$, $\pi_0 = 0.8$, and $\beta = 8$ constant, and $\pi_0 > m/n$). The solid curve represents the authentication probability obtained from simulations, and the dotted curve represents the value of $\Phi(k)$ given in Proposition 2.4.1. As hypothesized, the authentication probabilities of the two curves increase as n is increased, although n/m is kept constant.

From the above result, we can conclude that it would be advantageous to make the block size large, if relatively large verification delays (at the receiver) are tolerated.

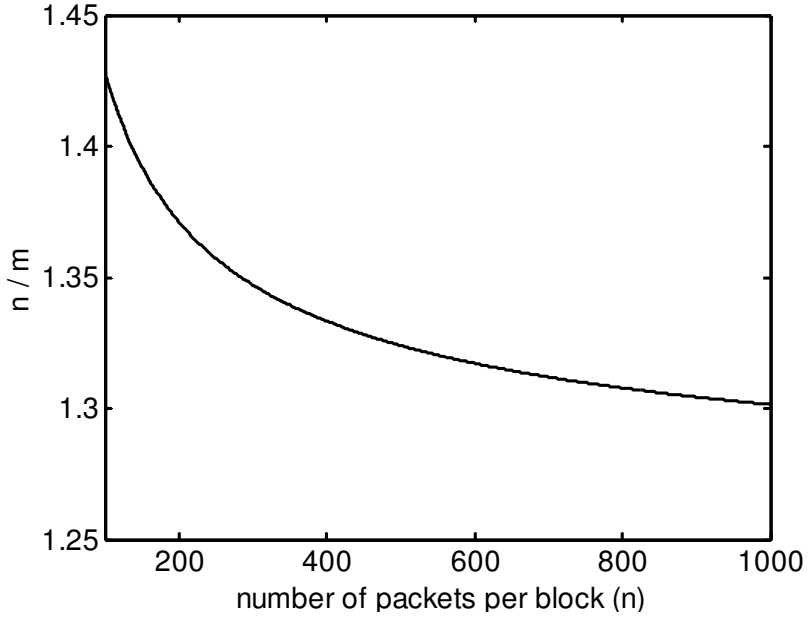

Fig. 2.3. Plot of n/m versus n for $\pi_0 = 0.8$ and $\gamma = 1$.

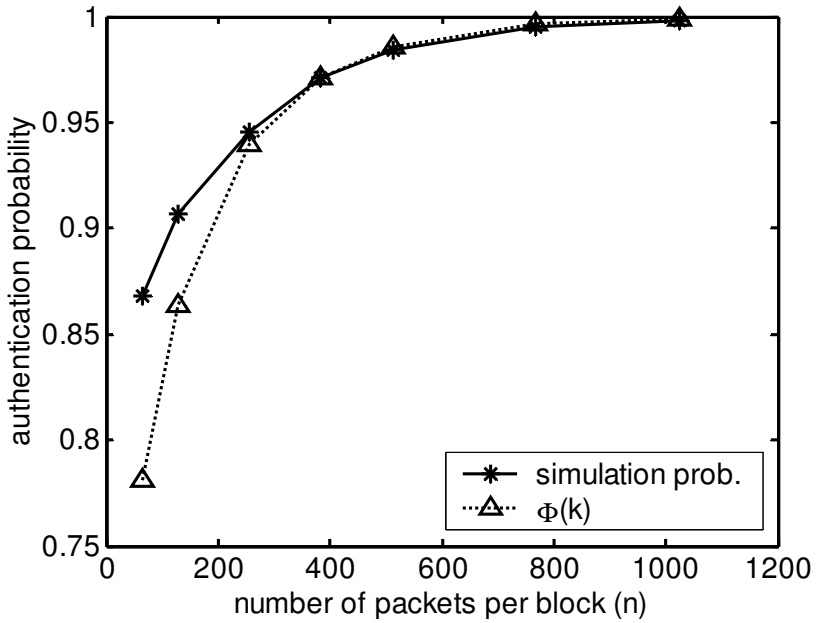

Fig. 2.4. Authentication probability versus n.

By making the block size large, the sender can decrease the space overhead incurred by the authentication process without affecting performance (i.e., the authentication probability).

2.4.3 The Asymptotic Authentication Probability Under the Biased Coin Toss Model

In this subsection, we derive the asymptotic authentication probability of SAIDA under the Biased Coin Toss (BCT) loss model. First, we need to define the BCT loss model.

Definition 2.4.2 BCT loss model. *Let $0 < q < 1$, and let $b \geq 1$ be an integer. For all i, a burst of length b packets begins with packet P_i (i.e., loss includes P_i) with probability q.*

For $b = 1$, this model is equivalent to the 2-MC loss model with $p_{01} = p_{11} = q$ and $p_{10} = p_{00} = 1 - q$ (and hence $\pi_1 = q$). This has the effect of removing the dependence of S_{i+1} on S_i (for all i), and hence, the loss (or no loss) of packets is determined by independent tosses of a q-biased coin. For $b > 1$, this model produces bursty loss patterns, whereas for $b = 1$, it produces independent packet losses.

We can use the techniques applied in Subsection 2.4.2 to derive the asymptotic authentication probability of SAIDA under the BCT loss model. The same techniques are also applicable in this case because the BCT loss process can be modeled as a discrete-time Markov chain with $b + 1$ states: $0, 1, \ldots, b$. The BCT loss model is represented as a Markov chain in Figure 2.5. In this Markov chain, state 0 represents no loss, and states 1 through b denote packet loss. The transitions into state 0 are renewal events. Recall that in the derivations of Subsection 2.4.2, visits to state 1 represented renewal events.

To derive the asymptotic authentication probability under the BCT model, we need to find the variance of $\boldsymbol{T}_0$, where $\boldsymbol{T}_0$ denotes the number of transitions between successive visits to state 0. The variance of $\boldsymbol{T}_0$ is given by the following lemma.

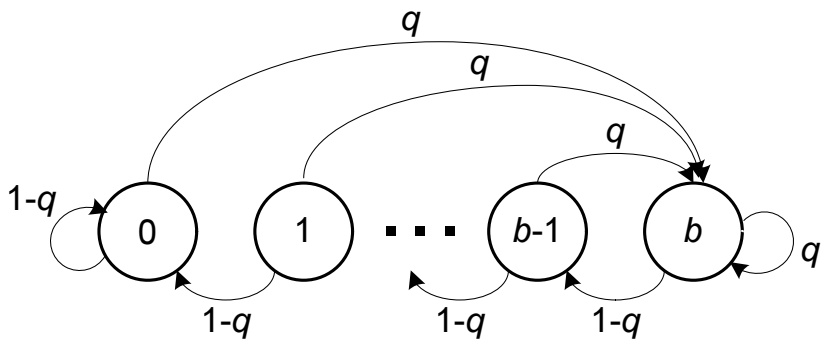

Fig. 2.5. The BCT loss process modeled as a Markov chain.

Lemma 2.4.4 *The variance of* $\boldsymbol{T}_0$ *is given by:*

$$\mathrm{Var}(\boldsymbol{T}_0) = \frac{2}{(1-q)^b}\sum_{i=0}^{b}\pi_i\mu_{i0} - \frac{(1-q)^b+1}{(1-q)^{2b}},$$

where

$$\mu_{i0} = \begin{cases} \frac{1}{(1-q)^b} & i = 0 \text{ or } 1 \\ \frac{2-q}{(1-q)^b}\sum_{i=0}^{\lfloor (i-1)/2\rfloor}(1-q)^{2i} & i \geq 2 \text{ and even} \\ \frac{1}{(1-q)^b}\left(1+(2-q)\sum_{i=0}^{\lfloor i/2\rfloor -1}(1-q)^{2i+1}\right) & i \geq 3 \text{ and odd,} \end{cases}$$

and

$$\pi_i = \begin{cases} (1-q)^b & i = 0 \\ q(1-q)^{b-i} & 1 \leq i \leq b. \end{cases}$$

Here, $\lfloor x \rfloor$ *denotes the largest integer not greater than* x*.*

Proof Using the same argument used in the proof of Lemma 2.4.2 (see (2.3)), we obtain:

$$E[\boldsymbol{T}_0^2] = \frac{2}{\pi_0}\sum_{i=0}^{b}\pi_i\mu_{i0} - \frac{1}{\pi_0}. \tag{2.7}$$

By using the relations among the stationary probabilities, we obtain the following:

$$\pi_i = \sum_{j=0}^{b}\pi_j P_{ji} \quad 0 \leq i \leq b.$$

The values of π_i stated in the lemma can be obtained by using the above set of equations and the relation

$$\pi_0 + \pi_1 + \cdots + \pi_b = 1.$$

Conditioning on the next state visited, the following set of equations can be obtained:

$$\mu_{i0} = 1 + \sum_{j=1}^{b} P_{ij}\mu_{j0} \quad 0 \leq i \leq b.$$

Solving for the values of μ_{i0} (using the above set of equations), we obtain the values of μ_{i0} stated in the lemma.

Now, using (2.7) and the relation $E[\boldsymbol{T}_0] = 1/\pi_0$, the variance of $\boldsymbol{T}_0$ is given by

$$\mathrm{Var}(\boldsymbol{T}_0) = \frac{2}{\pi_0}\sum_{i=0}^{b} \pi_i\mu_{i0} - \frac{1}{\pi_0} - \frac{1}{\pi_0^2}.$$

Using the relation $\pi_0 = (1-q)^b$ in the above equation, we obtain the desired result. ■

Now, we state the asymptotic authentication probability of our authentication scheme, SAIDA, under the BCT model. The same techniques used in the proof of Proposition 2.4.1 are applied here.

Proposition 2.4.2 *The authentication probability of the i-th packet in the block is given by the following expression when the minimum number of encoded segments required for decoding is* $m = n\pi_0 + \gamma\sqrt{n}$*:*

$$\Pr\{P_i \text{ is verifiable} | P_i \text{ is received}\} \to 1 - \Phi(k) \text{ as } n \to \infty,$$

where

$$k = \frac{\gamma}{\sqrt{(1-q)^{3b}\mathrm{Var}(\boldsymbol{T}_0)}},$$

and $\mathrm{Var}(\boldsymbol{T}_0)$ *is given in Lemma 2.4.4.*

Proof Define the renewal process $\{M(0), M(1), \ldots\}$ as follows: $M(0) = 0$, and $M(n-i)$ is the number of packets received among $P_{i+1}, \ldots, P_n$. We can see that

the authentication probability is lower bounded by the probability of the number of packets received among $P_{i+1}, \ldots, P_n$ not being less than $m-1$. Hence,

$$\begin{aligned} \Pr\{P_i \text{ is verifiable} | P_i \text{ is received}\} &\geq \Pr\{M(n-i) \geq m-1\} \\ &= 1 - \Pr\{M(n-i) < m-1\}. \end{aligned}$$

Now, let

$$v = \frac{\gamma\sqrt{n} + i\pi_0 - 1}{\sqrt{\frac{(n-i)\mathrm{Var}(\boldsymbol{T}_0)}{(E[\boldsymbol{T}_0])^3}}}.$$

Then, the following holds:

$$1 - \Pr\left\{ \frac{M(n-i) - \pi_0(n-i)}{\sqrt{\frac{(n-i)\mathrm{Var}(\boldsymbol{T}_0)}{(E[\boldsymbol{T}_0])^3}}} < v \right\} = 1 - \Pr\{M(n-i) < m-1\}. \tag{2.8}$$

Similarly, the authentication probability is upper bounded by the probability of the number of packets received among $P_{i+1}, \ldots, P_n$ not being less than $m-i$. Hence,

$$\begin{aligned} \Pr\{P_i \text{ is verifiable} | P_i \text{ is received}\} &\leq \Pr\{M(n-i) \geq m-i\} \\ &= 1 - \Pr\{M(n-i) < m-i\}. \end{aligned}$$

Now, let

$$w = \frac{\gamma\sqrt{n} + i\pi_0 - i}{\sqrt{\frac{(n-i)\mathrm{Var}(\boldsymbol{T}_0)}{(E[\boldsymbol{T}_0])^3}}}.$$

Then, the following holds:

$$1 - \Pr\left\{ \frac{M(n-i) - \pi_0(n-i)}{\sqrt{\frac{(n-i)\mathrm{Var}(\boldsymbol{T}_0)}{(E[\boldsymbol{T}_0])^3}}} < w \right\} = 1 - \Pr\{M(n-i) < m-i\}. \tag{2.9}$$

Hence, the authentication probability is lower bounded by (2.8) and upper bounded by (2.9).

Define

$$k = \frac{\gamma}{\sqrt{\frac{\mathrm{Var}(\boldsymbol{T}_0)}{(E[\boldsymbol{T}_0])^3}}}.$$

Comparing the values of v, w, and k for fixed i and $n \to \infty$, it follows that

$$\lim_{n\to\infty} v = \lim_{n\to\infty} w = k.$$

Therefore, the lower bound and the upper bound are asymptotically the same (for fixed i and $n \to \infty$), and the following holds by Corollary 2.4.1:

$$\Pr\{P_i \text{ is verifiable} | P_i \text{ is received}\} \to 1 - \Phi(k) \text{ as } n \to \infty.$$

Substituting the value of $1/(1-q)^b$ for $E[\boldsymbol{T}_0]$ in k, we obtain the desired result. ■

Not surprisingly, Proposition 2.4.2 reveals the same relationship between n, n/m, and the authentication probability—by increasing n, the authentication probability can be improved while a constant value of n/m is maintained (see Figure 2.4). This means that there is a way to improve the authentication probability without increasing the space overhead (required for the authentication information), if increased delay is tolerated. Of course, this is done by increasing the value of n.

2.4.4 The Lower Bound of the Authentication Probability

In this subsection, using the BCT loss model, we derive a lower bound of the authentication probability for SAIDA. According to the loss model, the maximum number of places where a burst error can occur (and still allow the guaranteed authentication of P_i) is given by

$$z = \left\lfloor \frac{n-m}{b} \right\rfloor.$$

Recall that n, m, and b denote the total number of encoded segments, the minimum number of encoded segments needed for decoding, and the minimum burst length of the BCT loss model, respectively. Because the loss of a packet is determined by the flip of a q-biased coin, the probability that z or fewer coin tosses result in losses lower bounds the authentication probability. Hence, the authentication probability is bounded as follows:

$$\Pr\{P_i \text{ is verifiable} | P_i \text{ is received}\} \geq \begin{cases} \sum_{j=0}^{z} \binom{n-b}{j} q^j (1-q)^{n-b-j} & \text{if } i > b-1 \\ \sum_{j=0}^{z} \binom{n-i}{j} q^j (1-q)^{n-i-j} & \text{if } i \leq b-1. \end{cases}$$

The above derivation takes into account the fact that none of the z coin tosses can occur in the $b-1$ packets immediately preceding P_i because it is assumed that the

packet is received. For $i \leq b-1$, this would be $i-1$ packets immediately preceding P_i.

In [MiS01], the authors derive a lower bound of the authentication probability (for the piggybacking scheme) based on the BCT loss model. The lower bound is given by the following expression (see [MiS01] for derivation):

$$\Pr\{P_i \text{ is verifiable} | P_i \text{ is received}\} \begin{cases} \geq \sum_{j=0}^{z'} \binom{i-1-b}{j} q^j (1-q)^{i-1-b-j} & \text{if } i > b+1+z' \\ = 1 & \text{if } i \leq b+1+z', \end{cases}$$

where $z' = \min(x_i, \lfloor b_i/b \rfloor)$. Parameters x_i and b_i denote the maximum number and the maximum size of the bursts that can be tolerated, respectively. Note that this lower bound was derived assuming that the signature packet was received, whereas the lower bound for SAIDA was derived without this assumption.

In Figure 2.6, we plot the lower bounds of the two schemes as the communication overhead (per packet) is increased. Because the lower bound changes with the position of the packet, the average lower bound (among n packets) is used. Note that the lower bound of SAIDA resembles a distorted staircase function. This is because of the floor function within the expression for the lower bound of SAIDA. The following parameters are used:

- signature size = 128 bytes;
- hash size = 16 bytes;
- $n = 128$, $b = 4$, $q = 0.2$, $b_i = 48$;
- parameter x_i is increased from one to twelve in increments of one;
- values for m : $\{67, 45, 34, 28, 23, 20, 17, 16, 14, 13, 12, 11\}$.

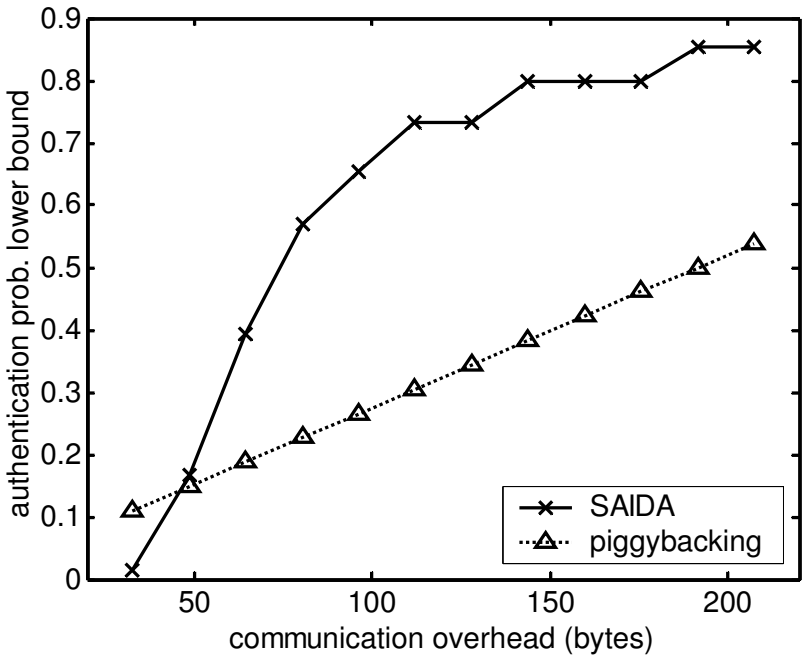

Fig. 2.6. Authentication probability lower bounds.

2.5 Performance Evaluations

2.5.1 Overhead Comparisons

We compare our solution with four previously proposed schemes: the authentication tree, EMSS, the augmented chain, and the piggybacking scheme. We only consider schemes that amortize a signing operation over multiple packets. In the comparison, the following assumptions are made:

- All five schemes employ a block size of n packets.
- Communication overhead of the authentication tree is obtained for a tree of degree two, and assuming a signature size of σ and a hash size of h.
- The augmented chain is parameterized by the integer variables a and p, where $p < n$.

- For SAIDA, n is the number of encoded segments, and m is the minimum number of encoded segments needed for decoding.

The five stream authentication schemes are summarized in Table 2.1 based on the following performance criteria:

- **sender delay:** delay on the sender side (in number of data packets) before the first packet in the block can be transmitted.
- **receiver delay:** delay on the receiver side (in number of data packets) before verification of the first packet in the block is possible.
- **computation overhead:** number of hashes and signatures computed by the sender per block.
- **communication overhead:** size of the authentication information (i.e., hashes and digital signatures) required for each packet.
- **verification rate:** number of verifiable packets of a given stream divided by the number of received packets of the same stream.

According to Table 2.1, the verification rate for EMSS, augmented chain, piggybacking, and SAIDA is not constant and actually depends on the communication overhead. The authentication tree technique has the favorable property of guaranteeing the verification of every received packet, but at the cost of a larger communication overhead—an overhead on the order of several hundred bytes would be required for practical block sizes. Note that the receiver delay for SAIDA is not fixed; it can be anywhere between m and n (i.e., in the interval $[m, n]$). We emphasize that SAIDA's advantage over the other schemes is the ability to obtain high verification rates with minimal communication overhead (see Subsection 2.5.2). By strategy, our scheme trades off sender and receiver delay for improvements in the verification rate. Note that the technique of Pannetrat et al. [PaM02] can be applied to SAIDA to remove the sender delay at the cost of increased receiver delay and vice versa.

Table 2.1
Comparison of the stream authentication schemes.

	authentication tree	EMSS	augmented chain	piggy-backing	SAIDA
sender delay	n	1	p	n	n
receiver delay	1	n	n	1	$[m, n]$
computation overhead	$2n - 1, 1$	$n + 1, 1$	$n + 1, 1$	$n + 1, 1$	$n + 1, 1$
communication overhead	$\sigma + 1 + (h + 1) \lg n$	variable	variable	variable	variable
verification rate	1.0	variable	variable	variable	variable

2.5.2 The Trade-off Between Verification Rate and Communication Overhead

As mentioned earlier, if the requirement on individual packet verification is relaxed, then the communication overhead can be reduced substantially. In this approach, verification of each packet is not guaranteed, and instead is assured with a certain probability. EMSS, augmented chain, piggybacking, and SAIDA fall into this category, and as expected, there is a trade-off between verification rate and communication overhead for these schemes.

For the augmented chain method, the number of hash chains per packet is not a variable parameter. However, multiple copies of the signed packet can be transmitted to increase the verification rate. The same technique can be applied to the piggybacking scheme to improve performance. For simulations of the piggybacking scheme, we assume that there is only one priority class (i.e., priority class 0) with $x_0 = 2$ and $b_0 = 29$. Recall that x_i and b_i denote the maximum number of bursts, and the maximum size of a burst that can be tolerated in the i-th priority class, respectively. In SAIDA, higher verification rates can be achieved by increasing the value of

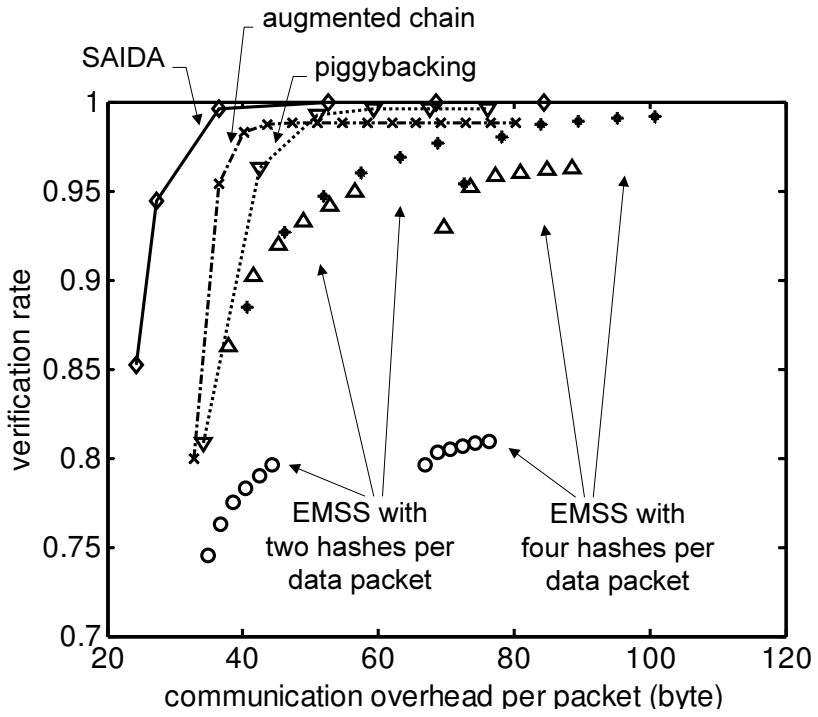

Fig. 2.7. Verification rate versus communication overhead.

n/m. The tradeoff between performance and communication overhead in EMSS was already discussed in Subsection 2.3.1

Figure 2.7 shows the verification rate for the four probabilistic authentication schemes: the augmented chain, EMSS, piggybacking, and SAIDA. To simulate bursty loss patterns, we used the 2-MC loss model defined in Subsection 2.4.2 with a packet loss probability of 0.2. The choice of 0.2 as the loss probability is motivated by the fact that, in general, the receiver loss rates are greater for multicast transmissions compared to unicast transmissions. In [YaM99], the authors, using actual network measurements, showed that the loss rates for multicast sessions are much higher (more than twice) compared to their corresponding unicast sessions. Some multicast sessions were observed to have loss rates exceeding 20% [YaK96, YaM99]. The simulation parameters for the curves of Figure 2.7 are given in Table 2.2.

Table 2.2
Simulation parameters for Figure 2.7.

scheme	simulation parameters
general parameters	loss prob. = 0.2, avg. burst length = 8, block size = 128, size of hash = 16 bytes, size of signature = 128 bytes
EMSS	length of hash chain is uniformly distributed over $[1, 127]$
augmented chain	$p = 6$ and $a = 15$
piggybacking	$x_0 = 2$ and $b_0 = 29$
SAIDA	$n = 128$, $m = \{90, 80, 60, 42, 32, 26\}$

For EMSS, verification rates are obtained by simulating numerous combinations of the three factors discussed in Subsection 2.3.1. The two clusters of markers represent the simulation results for EMSS. The left cluster represents EMSS implemented with two hashes appended per data packet, and the right cluster represents EMSS implemented with four. Each cluster is composed of three types of markers—circle markers represent implementations with a single signature packet per block, while the triangle and asterisk markers represent implementations with two and three signature packets per block, respectively. Each type of marker is used several times to represent the different number of hashes appended in the signature packet. The number of hashes appended in the signature packet is varied from 15 to 90 in increments of fifteen.

When multicast packets (via UDP) are sent across networks with heavy congestion or route failures, packet loss can be high. Furthermore, conditions for the network can change abruptly during a relatively short time period. Even if the verification rate of the packets is satisfactory at the start of reception, it could deteriorate rapidly as the loss rate increases in the network. We performed experiments to examine the effect of increased packet loss on the robustness of the stream authentication scheme. Figure 2.8 shows the change in verification rate as the authentication space overhead is kept constant, and the packet loss probability is increased. The authentication

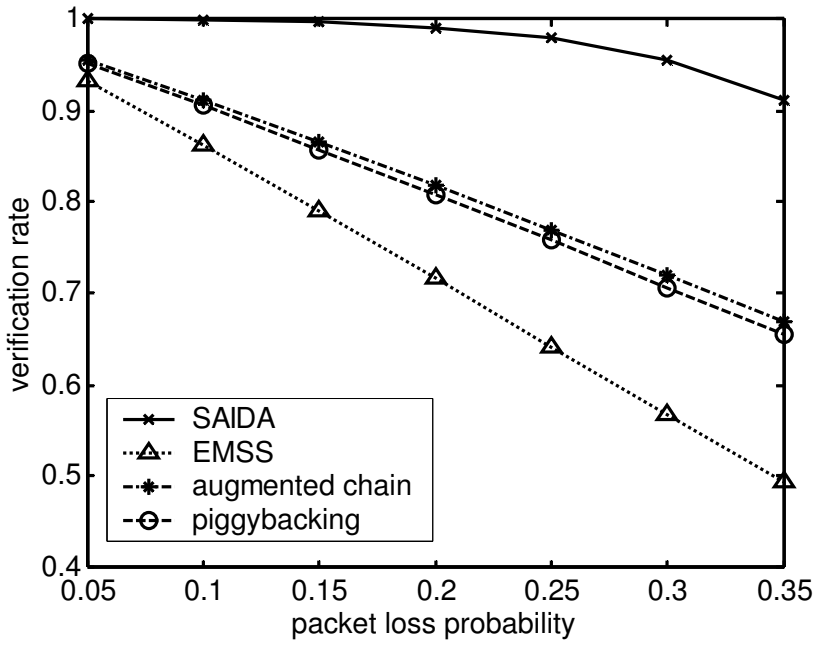

Fig. 2.8. Verification rate versus packet loss probability.

space overhead per packet is fixed at 34 bytes for all four schemes. Again, the 2-MC loss model is used. The simulation parameters are given in Table 2.3. For SAIDA, we can see that the verification rate drops very gradually as the packet loss probability is increased. The verification rates of the other schemes showed a much sharper decline.

2.6 Making SAIDA Robust Against Denial-of-Service Attacks

Consider the following denial-of-service (DoS) attack: an attacker inserts bogus packets in the packet stream to interfere with the reconstruction process of SAIDA. Recall that IDA is an erasure code that is robust against missing segments of information, and is not robust against modified segments (either intentional or unintentional). If one or more of the IDA encoded segments (which are used in the reconstruction) are modified during transit, the receiver has no way of detecting this, and the reconstruction of the original message fails. Although UDP, which is used

Table 2.3
Simulation parameters for Figure 2.8.

scheme	simulation parameters
general parameters	avg. burst length = 8, block size = 128, size of hash = 16 bytes, size of signature = 128 bytes
EMSS	length of hash chain is uniformly distributed over $[1, 64]$, number of hashes per signature packet = 5, number of hashes per data packet = 2, number of signature packets per block = 1
augmented chain	number of signature packets per block = 1, $p = 6$ and $a = 15$
piggybacking	number of signature packets per block = 1, $x_0 = 2$ and $b_0 = 29$
SAIDA	$n = 128$, $m = 65$

by multicast applications, provides a simple way of detecting errors within a packet via the UDP *checksum* (see p. 182, [Com95]), this does not protect against intentional alteration of the packet. By inserting bogus packets (with valid checksums) within the packet stream, an attacker can successfully interrupt multicast application services. In [Rab89], Rabin proposes a solution to combat this type of an attack by cryptographically *fingerprinting* the IDA encoded segments—this enables the receiver to verify which encoded segments were modified and which were not. Although this solution is effective against attackers that are outside the multicast group, it does not prevent attacks carried out by malicious users who are members of the multicast group (see [Rab89] for details).

In [Kra93], an improved solution, which uses a new cryptographic tool called *distributed fingerprints*, is proposed that does not suffer from the drawback mentioned above. Krawczyk's approach is to use IDA in combination with an error-correcting code, which is robust against information modification as well as loss. As expected, error-correcting codes add more redundancy in the encoding process compared to erasure codes. The amount of redundancy is commonly measured by a parameter

known as the *blowup factor*, which is defined as the ratio between the total size of the encoded information and the size of the original information. For IDA, the blowup factor is

$$\frac{n}{n-t},$$

where t is the maximum allowable number of missing encoded segments, and n is the total number of encoded segments. According to Lemma 1 given in [Kra93], an error-correcting code that can recover the original information in the presence of up to α altered segments and t missing segments has a blowup factor lower bounded by

$$\frac{n}{n-t-2\alpha}.$$

Codes such as Reed–Solomon codes achieve this bound.

In terms of computational complexity, error-correcting codes, using straightforward implementations, can encode and decode in time $O(n^2)$. Using more sophisticated techniques, encoding and decoding times of $O(n \log n)$ and $O(n \log^2 n)$ are possible, respectively [Bla84]. For values of n appropriate to our problem, it is generally viewed that encoding and decoding can be done in real time.

Using Krawczyk's approach, we can modify SAIDA so that it is robust against DoS attacks of the type discussed above. The following steps describe the modified authentication procedure:

1. Apply Steps 1 through 7 (of Subsection 2.3.2) to create IDA encoded segments of F^1 and $\sigma(K_r, H_G(G_1))$.

2. Concatenate σ_i and F_i^1 to form $\sigma_i \parallel F_i^1$ for $i = 1, \ldots, n$. Here, σ_i denotes $\sigma_i(K_r, H_G(G_1))$.

3. For $\sigma_i \parallel F_i^1, i = 1, \ldots, n$, compute its *fingerprint* $H(\sigma_i \parallel F_i^1)$ using a collision-resistant hash function H, and then concatenate these values to form a single string

$$\Omega = H(\sigma_1 \parallel F_1^1) \parallel \cdots \parallel H(\sigma_n \parallel F_n^1).$$

4. Encode Ω with an error-correcting code (e.g., Reed-Solomon code) to obtain

$$\omega_1, \ldots, \omega_n.$$

5. Concatenate each signature segment σ_i, hash segment F_i^1, and ω_i with the corresponding packet to form an authenticated packet. That is, create the following:

$$\sigma_i(K_r, H_G(G_1)) \parallel F_i^1 \parallel \omega_i \parallel P_i, \quad \text{for } i = 1, \ldots, n.$$

The resulting authenticated packets are transmitted.

The verification procedure is described in the following steps. We assume that t, the number of missing segments, and α, the number of altered segments, are small enough so that IDA and the error-correcting code are able to reconstruct the original information.

1. Using the decoding algorithm of the error-correcting code, reconstruct Ω. Let the following string represent the reconstructed string Ω:

$$H(\sigma_1 \parallel F_1^1) \parallel \cdots \parallel H(\sigma_n \parallel F_n^1).$$

2. For each authenticated packet that is received, extract its signature segment and hash segment, concatenate the two values, and hash it, which we denote as $H(\tilde{\sigma}_i \parallel \tilde{F}_i^1)$. Compare this value with the corresponding part of Ω (i.e., $H(\sigma_i \parallel F_i^1)$). If $H(\sigma_i \parallel F_i^1) = H(\tilde{\sigma}_i \parallel \tilde{F}_i^1)$, then the signature segment $\tilde{\sigma}_i$ and hash segment $\tilde{F}_i^1$ are considered to be legitimate (i.e., unmodified). Otherwise, the signature and hash segments are considered to be modified.

3. Employing the decoding algorithm of IDA (see Subsection 2.3.2), reconstruct F^1 and $\sigma(K_r, H_G(G_1))$ using only legitimate segments.

In [Kra93], it is shown that the distributed fingerprint scheme is asymptotically (in the size of the original message) space optimal. If the distributed fingerprint scheme

with Reed-Solomon encoding is applied to SAIDA, the resulting authentication space (or communication) overhead per block is given by the following:

$$\frac{n(nh+s)}{n-t-\alpha} + \frac{n^2 h}{n-t-2\alpha},$$

where h denotes the size of the hash and s denotes the size of the signature. Here, we assumed that the same hash function is used to compute the packet hashes and the fingerprints. The resulting authentication scheme can reconstruct the original authentication information in the presence of up to α modified packets and t lost packets.

2.7 Conclusions

Through our results, we showed that SAIDA is an efficient method for multicast stream authentication that is highly robust against packet loss. For the same amount of authentication space (or communication) overhead, it achieved the highest verification rates among all the probabilistic schemes that were examined. Table 2.1 suggests that there is no single scheme that is superior in all aspects. Depending on the delay, computation, and communication-overhead requirements, different schemes are appropriate for different applications. We expect that our scheme's high tolerance for packet loss and low communication-overhead requirement will make it useful in many multicast or broadcast applications. As already mentioned, SAIDA might not be appropriate in situations where the data to be sent is generated in real time, and immediate broadcast of it is crucial. Our scheme will be most useful in cases where the sender has a priori knowledge of at least a portion of the data to be broadcast (e.g., broadcast of prerecorded material).

3. CONSTRUCTION OF FAIR-EXCHANGE PROTOCOLS FOR E-COMMERCE

3.1 Introduction

Fueled by the exponential growth in the number of people with access to the Internet, e-commerce transactions via the Internet have become a major part of our economy. According to some analysts, consumer sales over the Internet have risen from $3.9 billion in 1998 to $108 billion in 2003. During the same time span, business-to-business commerce over the Internet jumped from $48 billion to $1.5 trillion. To date, the majority of e-commerce transactions involve exchanging the customer's credit-card number for the merchant's guarantee of merchandise delivery (i.e., an electronic receipt). In such transactions, items being exchanged have no intrinsic value, and thus the importance of ensuring a "fair exchange" has not received widespread attention. In a fair exchange, either each player gets the other player's item, or neither player does. In the foreseeable future, data with significant intrinsic value, such as financial data, medical data, software, and electronic forms of money (e.g., Mondex[1] and InternetCash[2]) will be exchanged regularly over the Internet. In such instances, ensuring fairness is critical if the participants are to be protected from fraud. In person-to-person transactions, fairness is provided by physical cues. The customer can minimize her risk by using her visual and tactile senses to verify the merchandise, and even make judgments about the merchant or the store. The merchant, on the other hand, can physically check the received payment for its validity before handing over the merchandise. On the Internet, where such physical cues cannot be used, fairness of the exchange must be guaranteed by the use of a fair-exchange pro-

[1]For details, see `http://www.mondexusa.com/`.

[2]For details, see `http://www.internetcash.com/`.

tocol that employs sound cryptographic techniques. For instance, applications such as payment protocols via electronic money [BoF98, CoT95], electronic contract signing [AsS00,EvG85], and certified e-mail delivery [AbG02,AtN02,BaT94] use protocols where fairness is ensured via cryptographic primitives. As more business is conducted over the Internet, the fair-exchange problem is gaining greater importance.

Significant effort has been devoted to the study of the fair-exchange problem. For an overview of fair-exchange protocols, we refer the reader to [RaR02]. Fair-exchange protocols can be broadly categorized into three types: (i) gradual exchange protocols, (ii) protocols requiring an on-line trusted third party (TTP), and (iii) protocols requiring an off-line TTP. There are some protocols [CoT95, ZhG96] (not included in the three categories above) that do not ensure fairness, but provide a weaker guarantee: the protocol gathers evidence during execution so that if one party obtains the other's item without sending his, the dishonest party can be prosecuted using the evidence.

In gradual exchange protocols [Blu83, EvG85], the probability of fair exchange is gradually increased over several rounds of message exchanges. These protocols are impractical because extensive amounts of communication are needed. Furthermore, proofs of their security rely on both parties having equal computational power, which is unrealistic for most applications.

In fair-exchange protocols with an on-line TTP [BaT94, CoT95], a TTP is directly involved in every exchange, and the TTP must be available for the entire duration of the exchange. The protocol itself is relatively simple and computationally efficient. However, maintaining a highly fault-tolerant TTP that needs to be involved in every exchange can be expensive. Moreover, the TTP can become a bottleneck, and pose scalability problems. Protocols with an off-line TTP avoid the problems faced by on-line protocols. In off-line TTP protocols, the TTP is involved in the protocol only if one of the parties behaves unfairly or aborts the protocol prematurely; otherwise, the TTP is never involved. Hence, these protocols are also known as "optimistic" fair-exchange protocols. Despite the many appealing properties of optimistic proto-

cols, many applications adopt an on-line TTP primarily because optimistic protocols entail intricate cryptographic techniques that incur considerable overhead. The vast majority of optimistic protocols use zero-knowledge proof systems. Informally, a zero-knowledge proof system allows a prover to demonstrate knowledge of a secret while revealing no useful information to the verifier in conveying this demonstration of knowledge to others. Generally, these proofs require the prover and the verifier to compute multiple modular exponentiations of very large integers, which are costly to compute. In Subsection 3.2.1, we discuss some noteworthy optimistic fair-exchange protocols that have been proposed previously.

In this chapter, we present a novel approach for constructing optimistic fair-exchange protocols. Although our protocol uses an off-line TTP, it is very efficient in terms of computation and communication overhead. In fact, it requires very little overhead beyond what is typically required in on-line TTP protocols, and to the best of our knowledge, it is one of the most computationally efficient optimistic protocols known to date. The noticeable improvement in efficiency (over previously proposed optimistic protocols) is possible because we employed a novel digital signature paradigm that we call the *gradational signature* paradigm. Note that in exchange protocols for e-commerce, the typical item offered by one of the players (i.e., the customer or merchant) is essentially a digital signature or an extension of it (e.g., digital checks and digital receipts). The intrinsic features of the gradational signature paradigm make it possible to devise simple and efficient optimistic fair-exchange protocols.

In the next section, we give a brief overview of optimistic fair-exchange protocols, and discuss related work. We also discuss a zero-knowledge proof system that is employed in our fair-exchange protocols. In Section 3.3, the gradational signature paradigm is discussed in detail. We discuss how to construct gradational signature schemes based on four well-known conventional signature algorithms in Section 3.4. In Section 3.5, we construct optimistic fair-exchange protocols using the gradational signature paradigm. The efficiency of our protocol is compared with previously pro-

posed protocols in Section 3.6. We consider the security issues involved in splitting RSA private keys, and analyze the security of one of our protocols in Section 3.7. The concluding remarks for this chapter are given in Section 3.8. A preliminary version of portions of material from this chapter was published in [PaC03b].

3.2 Technical Background

3.2.1 Optimistic Fair-Exchange Protocols

Most, if not all, optimistic fair-exchange protocols in the literature are largely based on two different types of protocol frameworks—one is the framework proposed by Asokan et al. [AsS98, AsS00], and the other is the framework used by Bao et al. [BaD98]. However, each previously proposed protocol employs a different technique for constructing the cryptographic primitive that ensures fairness—this primitive is the cornerstone of the fair-exchange protocol, and its design poses the greatest technical challenge. To clarify how such a *fairness primitive* is used in an exchange protocol, we present an example of a simple optimistic fair-exchange protocol. The protocol is essentially the same as what is proposed in [BaD98] excluding the specific fairness primitive used in the protocol. Let Alice and Bob be two players trying to exchange their respective digital signatures σ_A and σ_B on a message M (known a priori to both parties), and let Charlie represent a TTP. In the first step of the protocol, one of the items that Alice sends to Bob is her *commitment* to the transaction, which we denote as c_A. This value c_A has no intrinsic value, but serves as Alice's commitment to the exchange. Along with c_A, Alice has to send a *voucher* V_C. This voucher is a signed statement from Charlie that guarantees the following: (i) there exists a tamperproof one-to-one link between c_A and Alice's signature σ_A, and (ii) Charlie can compute σ_A using c_A if needed. We call c_A and V_C collectively as the fairness primitive. The fair-exchange protocol is executed as follows:

EXCHANGE.

1. Alice sends Bob c_A and its associated voucher V_C.

2. Bob verifies V_C and c_A, and, if valid, sends his signature σ_B to Alice. Otherwise, he stops the protocol.

3. If σ_B is valid, Alice sends σ_A to Bob, otherwise she stops the protocol.

4. If σ_A is valid, Bob ends the exchange protocol. Otherwise, Bob initiates the dispute resolution protocol.

DISPUTE RESOLUTION.

1. If σ_A is invalid (in Step 4), or if Bob fails to receive anything from Alice (in Step 3), he initiates a dispute resolution protocol with Charlie, and sends c_A, V_C, and σ_B to him.

2. Charlie verifies whether c_A, V_C, and σ_B are valid. If they are valid, Charlie computes σ_A (which he computes using c_A), and sends it to Bob. He also forwards σ_B to Alice.

Note that, in the last step of the dispute resolution protocol, Bob's signature σ_B needs to be forwarded to Alice in case Bob is dishonest. This is necessary to prevent the scenario where Bob attempts to obtain σ_A via the dispute resolution protocol after intentionally aborting the exchange protocol after Step 1.

Although the example illustrates a signature-exchange protocol, this basic model can readily be adapted to almost any exchange protocol for e-commerce. For instance, in an e-commerce payment protocol, the signature of Alice (acting as a customer) would correspond to her electronic check, and the signature of Bob (acting as a merchant) would be replaced by some digital merchandise (e.g., MP3 music files, MPEG-4 media files, or e-books). In such a scenario, message M would include information such as Alice's unique identity, Bob's unique account number, price of merchandise, description of merchandise, date of transaction, and a serial number. The serial number is needed to combat replay attacks. In a replay attack, an adversary

records a communication session, and replays the entire session, or a portion of it, at some later point in time.

Observe that the above protocol framework does not ensure a *timely termination*, and, as a result, does not ensure fairness when time-sensitive items are exchanged. Consider the following scenario: after Step 1 of the exchange protocol, Bob aborts the exchange, and initiates a dispute resolution protocol after a long delay. Unaware of Bob's intentions, Charlie sends σ_A to Bob, and forwards σ_B to Alice. If the intentional delay (caused by Bob) is sufficiently long and σ_B is time-sensitive, then Alice suffers a loss. For example, if σ_B represents a digital airline ticket with an expiration date, then the ticket is useless to Alice after that date. To prevent such cases, a *timestamp* can be attached to c_A that specifies an expiration time. After this expiration time, Charlie would refuse to execute the dispute resolution protocol with Bob. Unlike the protocol framework used by Bao et al. [BaD98], the framework proposed by Asokan et al. [AsS98, AsS00] ensures timely termination without the use of timestamps. However, it is more complicated and requires the TTP to store state information. Although our techniques can be applied to *both* frameworks, we choose the framework of Bao et al. to illustrate how our techniques are applied to construct efficient optimistic protocols. We emphasize that our main contribution is the construction of a novel fairness primitive, and is *not* the proposal of a new protocol framework.

3.2.2 Related Work

We discuss some of the previous work on optimistic protocols, concentrating on the fairness primitive used in each scheme. Optimistic fair-exchange protocols can be categorized into four types depending on what cryptographic techniques are used to ensure fairness: (i) protocols using *verifiable escrow*, (ii) protocols using *verifiable encryption*, (iii) protocols using *convertible undeniable signatures*, and (iv) protocols using *off-line coupons*.

In [AsS98, AsS00], Asokan et al. propose an optimistic protocol that uses a cryptographic primitive called *verifiable escrow* to produce the fairness primitive. The verifiable escrow scheme is an extension of the ordinary escrow scheme. In an ordinary escrow scheme, a player sends his item (e.g., a digital signature) encrypted under the TTP's public key so that the recipient can have it decrypted by the TTP if necessary. A verifiable escrow scheme has the additional feature that enables the recipient of the escrow to verify that it is indeed the escrow of the desired item. In the scheme of Asokan et al., a verifiable escrow of a signature is created as follows: first, the signer reduces the "promise" of a signature to a "promise" of a particular homomorphic pre-image (of the signature); then, a cut-and-choose interactive zero-knowledge proof, in combination with an ordinary escrow scheme, is used to verifiably escrow the homomorphic pre-image. The reduction process is constructed such that it guarantees that the promised signature can be recovered from the pre-image. The above technique can be applied to almost any signature scheme as long as there is a way to reduce the signature into a homomorphic pre-image. Its drawback is that it requires the two exchanging parties to execute extensive amounts of computations during the interactive zero-knowledge proof. Furthermore, communication overhead is relatively high—the amount of data transmitted (by both parties) is on the order of several thousand bytes. The approach of Asokan et al. is later generalized by Camenisch and Damgard [CaD00], but the computational burden remains.

Fair-exchange protocols using *verifiable encryption* was proposed by Bao et al. [BaD98] and Ateniese [Ate99] independently. These protocols apply ad-hoc techniques to create an encryption of a signature that conforms to the signature type. For instance, a verifiable encryption of an RSA signature [RiS78] is created by encrypting it via the ElGamal encryption scheme [Elg85] using the TTP's public key. To prove that the verifiable encryption was correctly generated without revealing the signature itself, the signer uses a zero-knowledge proof. Note that the verifiable encryption corresponds to the signer's commitment, and the zero-knowledge proof corresponds to the voucher. To create a verifiable encryption of discrete-logarithm-based signa-

tures (e.g., ElGamal or Schnorr [Sch91]), ElGamal encryption cannot be used, and instead, cryptosystems based on suitable trap-door functions such as the Naccache-Stern cryptosystem [NaS98] or the Okamoto-Uchiyama cryptosystem [OkU98] must be employed. A verifiable encryption scheme using the Okamoto-Uchiyama cryptosystem is proposed in [Ate99, Bao98]. The primary advantage of protocols based on verifiable encryption is its efficiency. Ateniese [Ate99] showed that his protocol based on verifiable encryption requires considerably less computation and communication overhead compared to protocols based on verifiable escrow. However, because the verifiable encryption approach is based on ad-hoc techniques, it can be vulnerable to security flaws—the verifiable encryption of Guillou-Quisquater signatures proposed in [BaD98] was shown to be insecure in [BoF98] and [Ate99].

In [Che98, BoF98], the authors construct fair-exchange protocols using cryptographic primitives known as *convertible undeniable signatures*. The concept of undeniable signatures, which was first introduced by Chaum and van Antwerpen [ChV90], is different from the conventional notion of digital signatures in that the assistance of the signer is necessary in the verification process. The convertible undeniable signature [BoC90, DaP96, MiS97] is an extension of the undeniable signature. Like undeniable signatures, they require the signer's assistance for verification, but have the additional feature that allows the signature to be converted into a "universally verifiable" signature. That is, the original signature can be converted into a signature that can be verified by anyone without the assistance of the signer. When used in optimistic fair-exchange protocols, a convertible undeniable signature of a player acts as her commitment, and is given to the other player in one of the initial steps of the exchange. Its validity is shown via an interactive zero-knowledge proof performed by the two players involved in the exchange. If a dispute arises, the player holding the convertible undeniable signature contacts the TTP. The TTP converts the convertible undeniable signature into a universally verifiable signature, and returns it to the player who initiated the dispute resolution protocol. One drawback of this approach is that most converted signatures are not *compatible* with standard signatures. That

is, the converted signature does not have the same form as a standard signature, and hence needs a verification algorithm that is different from the algorithm used to verify standard signatures. If the fair-exchange feature is to be implemented on top of existing e-commerce systems, it is preferable to have digital signatures used in exchange protocols to be compatible with existing standard signatures. The convertible undeniable signature scheme proposed by Camenisch and Michels [CaM00] has the compatibility property, but this scheme is essentially the same as the verifiable encryption scheme. It should be noted that the convertible undeniable signatures used in [BoF98] is compatible with standard RSA signatures [RiS78]. However, their approach is limited to the RSA signature scheme.

Another method of constructing optimistic protocols is to use *off-line coupons*. In this approach, a player's signature is verifiably escrowed by a coupon, which is acquired from a TTP in advance of the exchange protocol. For security reasons, each coupon can be used only once. In practice, the coupons would be acquired from the TTP in bulk, prior to initiating the transactions, to increase efficiency. Obviously, it is necessary to contact the TTP for new coupons once all the previously issued coupons have been exhausted. This means that intermittent contact with the TTP is necessary. This approach is much more efficient than any of the other approaches discussed above. However, it has some drawbacks. For some signature algorithms, the off-line coupons approach proposed by Asokan et al. [AsS00] requires the use of a zero-knowledge proof system—this noticeably increases the computation overhead of the scheme (see Section IX of [AsS00]). A similar scheme proposed by Ateniese (called *tokens*) [Ate99] does not suffer from this drawback, but is only applicable to signatures based on the discrete-logarithm problem. The only type of optimistic fair-exchange protocol that rivals our scheme in terms of computation and communication efficiency is the off-line coupons approach. Our approach, however, does not suffer from the drawbacks mentioned above.

It should be noted that the difference between the techniques categorized above is, in some cases, vague, and certain schemes can belong to more than one category. For

instance, verifiable encryption of signatures can be considered as a type of convertible undeniable signatures.

3.2.3 A Zero-Knowledge Proof for Demonstrating the Possession of Discrete Logarithms

Informally, a zero-knowledge proof system allows a prover to demonstrate knowledge of a secret while revealing no useful information to the verifier in conveying this demonstration of knowledge to others. See [MeV96] for a detailed discussion on this topic. In the following, we describe an interactive zero-knowledge proof for proving the possession of discrete logarithms. This protocol is a modified extension of the protocol proposed in [ChE87]—it is modified to work with a composite modulus (see [GeK97]).

In the protocol, a prover P proves to a verifier V that an integer $\Omega = \omega^x \mod N$ is a power of $\omega \in \mathbb{Z}_N^*$ without revealing x. Here, $\mathbb{Z}_N^*$ denotes the multiplicative group of integers modulo N. The values Ω, N, and ω are known to both parties. The prover P randomly chooses a value $r \in \{1, \ldots, \phi(N)\}$, and sends to V the value $\omega' = \omega^r \mod N$, where $\phi(\cdot)$ denotes the Euler's totient function[3]. The verifier V answers with a random bit b. If $b = 0$, P sends back r, otherwise she sends back the value $\lambda = r + x \mod \phi(N)$. In the first case, V checks whether $\omega' = \omega^r \mod N$, and in the second case, he checks whether $\omega^\lambda \equiv \omega'\Omega \pmod N$. If Ω is not a power of ω, the probability that P passes this test is $1/2$. By repeating the protocol k times, this probability reduces to 2^{-k}.

We need to use the above zero-knowledge proof system in the registration phase of our fair-exchange protocol with RSA-based gradational signatures (see Subsection 3.5.1). To improve efficiency, we choose the non-interactive version of it, a la

[3] $\phi(N)$ denotes the number of integers in the interval $[1, N]$ that are relatively prime to N.

the Fiat-Shamir heuristic [FiS86]. The non-interactive version is executed as follows. The prover P randomly picks k integers $r_i \in \{1, \ldots, \phi(N)\}$, $1 \leq i \leq k$, and computes

$$\omega_i' = \omega^{r_i} \bmod N, \quad 1 \leq i \leq k.$$

Now, P computes

$$H(\omega \parallel \Omega \parallel \omega_1' \parallel \cdots \parallel \omega_k'),$$

where H is a suitable one-way hash function (see [FiS86] for details on the appropriate choice of H), and $\parallel$ denotes concatenation. Next, P takes the first k bits of the hash output, and assigns them as $b_1, \ldots, b_k$. Using these values, λ_i, $1 \leq i \leq k$, are computed as follows:

$$\lambda_i = \begin{cases} r_i & \text{if } b_i = 0 \\ r_i + x \bmod \phi(N) & \text{if } b_i = 1. \end{cases}$$

The prover P sends to V the values $(b_1, \ldots, b_k)$ and $(\lambda_1, \ldots, \lambda_k)$. After receiving these values, V computes

$$z_i = \begin{cases} \omega^{\lambda_i} \bmod N & \text{if } b_i = 0 \\ \omega^{\lambda_i} \cdot \Omega^{-1} \bmod N & \text{if } b_i = 1 \end{cases}$$

for $1 \leq i \leq k$. Next, V computes $H(\omega \parallel \Omega \parallel z_1 \parallel \cdots \parallel z_k)$, and compares its first k bits with $(b_1, \ldots, b_k)$. If they are identical, then V accepts P's claim that Ω is a power of ω.

3.3 The Gradational Signature Paradigm

3.3.1 The Basic Concept

We formalize and implement a novel signature paradigm, which we denote as *gradational signatures*. Using gradational signatures, we construct optimistic fair-exchange protocols that are very simple and efficient. The intrinsic features of the gradational signature paradigm enable us to construct optimistic fair-exchange protocols without resorting to costly cryptographic operations in the exchange procedure.

Although gradational signatures have many traits in common with *multisignatures*, they possess unique characteristics that set them apart. We briefly review the notion of multisignatures before describing gradational signatures.

If multiple signers need to sign a single message, the naive approach would be to let the signers create their own signature and concatenate the individual signatures. However, this causes an expansion in the size of the signature. A multisignature scheme allows multiple signers to sign a single message in an efficient manner such that the resulting multisignature has little or no difference in size with an individual signature. Since its introduction by Itakura and Nakamura [ItN83], several multisignature schemes based on different (single-signer) signature schemes have been proposed [Boy89, Har94b, MiO01]. The players in a typical multisignature scheme are $n \geq 2$ signers and a verifier. For each signer, there is an *individual private key* sk_i. Each signer uses its individual private key to compute its *individual signature* σ_i. Depending on the multisignature scheme, an *individual public key* pk_i, which is used to verify the individual signatures, might also be assigned to each signer. These individual signatures are combined to form the multisignature σ. In essence, the individual private keys conjointly form a *joint private key* sk that is required to create a multisignature. The joint private key has an explicit algebraic relation with the partial private keys, that is,

$$sk = sk_1 \oplus \cdots \oplus sk_n,$$

where $\oplus$ denotes some binary operation (e.g., addition or multiplication). If the keys are created correctly, it is infeasible to create a multisignature with a proper subset of $\{sk_1, \ldots, sk_n\}$. This differs from *threshold signatures* [DeF89, Har94a] in which a sufficiently large subgroup of signers can sign messages. To verify a multisignature, the verifier uses a *joint public key* pk, which is the counterpart of the joint private key.

In a gradational signature scheme, there are three entities: a *primary signer*, a *cosigner*, and a verifier.[4] The cosigner is an entity that is trusted by both the primary signer and the verifier. The primary signer possesses the *partial private keys*, sk_1 and sk_2, and the cosigner has knowledge of sk_2 only. The primary signer plays a primary role, whereas the cosigner plays an auxiliary role in generating a signature. Note that in multisignature schemes, every signer has an equal hand in the multisignature generation process. As the name implies, in a gradational signature scheme, a digital signature is generated in two phases. In the first phase, the primary signer generates a *partial signature* σ_1 using sk_1. The partial signature is verified using the *partial public key* pk_1. A partial signature is different from the common notion of a digital signature in that it does not provide non-repudiation—in fact, its verification key pk_1 is *not certified* by the certification authority (CA). For a signature to provide non-repudiation, there has to be a tamperproof link between the public key and the signer's identity, and this link is provided by the public key's certificate issued by the CA. Thus, σ_1 has no value as a signature, but serves as a token of the primary signer's commitment to the transaction in which the partial signature is used.

In the second phase, the desired signature σ, which we call the *full signature*, is computed using σ_1 and the *supplemental component* $\hat{\sigma}_2$. Hence, the full signature is comprised of two parts: the partial signature and the supplemental component. The partial private key sk_2 is required to generate $\hat{\sigma}_2$. Note that either the primary signer or the cosigner can compute $\hat{\sigma}_2$, whereas only the primary signer can compute σ_1. Unlike the partial signature, the supplemental component cannot be verified for correctness of construction; in fact, it does not have a verification key (i.e., a partial public key). In the gradational signature paradigm, the supplemental component functions only in a subsidiary capacity, that is, it is an auxiliary component of the full signature, and has no functionality of its own. Its main purpose is to allocate the cosigner a role in the full signature generation process. Because of these properties,

[4]We associate the primary signer with the female gender, and associate the cosigner and verifier with the male gender. The notations associated with the primary signer and cosigner have subscripts of one and two, respectively.

the relation between σ_1 and $\hat{\sigma}_2$ is different from the relation among the individual signatures of a multisignature scheme.[5]

At the end of the second phase, a full signature is generated. The full signature is identical to a typical digital signature in form and functionality. Hence, its verification algorithm is identical to the algorithm used to verify standard signatures. Its validity is checked using the *full public key* pk, which is certified by the CA.

In essence, the partial private keys, sk_1 and sk_2, conjointly form a *full private key* sk that is required to create a full signature. This means that both sk_1 and sk_2 are required to generate a full signature. Note that this "splitting" of sk into sk_1 and sk_2 is similar to how the joint private key is split into individual private keys in a multisignature scheme. Observe that the primary signer can compute the full signature on her own, but the cosigner cannot (because he does not have sk_1). However, the cosigner can compute the full signature σ if he acquires the partial signature σ_1 from another entity. In the following, we define the components that constitute the gradational signature paradigm.

- *Key generator* KG: A polynomial-time probabilistic algorithm, which on input of a security parameter k, returns keys (sk, pk), (sk_1, pk_1) and sk_2. The first pair denotes the full private-public key pair, the second pair denotes the primary signer's partial private-public key pair, and sk_2 denotes the cosigner's partial private key.

- *Partial signature generator* Sig_p: A polynomial-time probabilistic or deterministic algorithm, which on input of message $M \in \{0,1\}^*$, public/secret system parameters ρ, and the partial private key sk_1, outputs a partial signature σ_1. (Here, $\{0,1\}^*$ denotes a binary string of arbitrary finite length.)

- *Supplemental component generator* Sig_s: A polynomial-time probabilistic or deterministic algorithm, which on input of M, ρ, σ_1, and the partial private key sk_2, outputs a supplemental component $\hat{\sigma}_2$.

[5]The "hat" notation is used in $\hat{\sigma}_2$ to signify its functional difference with σ_1

- *Full signature generator Sig*: A polynomial-time probabilistic or deterministic algorithm, which on input of M, ρ, sk_1, and sk_2, outputs a full signature σ.

- *Partial signature verification algorithm Ver_p*: A polynomial-time deterministic algorithm, which takes as inputs M, ρ, alleged partial signature σ_1, and partial public key pk_1. The output is one or zero depending on whether the partial signature is "valid" or "invalid," respectively. That is,

$$Ver_p(M, \rho, \sigma_1, pk_1) = \begin{cases} 1 & \text{if } \sigma_1 \in \{Sig_p(M, \rho, sk_1)\} \\ 0 & \text{otherwise,} \end{cases}$$

  where $\{X(u)\}$ denotes the set of all possible output values of algorithm X with input u.

- *Full signature verification algorithm Ver*: A polynomial-time deterministic algorithm, which takes as inputs M, ρ, alleged full signature σ, and full public key pk. The output is one or zero depending on whether the full signature is "valid" or "invalid," respectively. That is,

$$Ver(M, \rho, \sigma, pk) = \begin{cases} 1 & \text{if } \sigma \in \{Sig(M, \rho, sk_1, sk_2)\} \\ 0 & \text{otherwise.} \end{cases}$$

Note that a full signature σ can be computed without any knowledge of sk_1 or sk_2, if the corresponding partial signature σ_1 and supplemental component $\hat{\sigma}_2$ are known. The schematics of the gradational signature paradigm and the (two-party) multisignature paradigm are shown in Figure 3.1 and Figure 3.2, respectively. In certain multisignature schemes, each individual signature is not generated in parallel as depicted in the figure, but instead computed in tandem (see [Oka88]).

3.3.2 Features of the Gradational Signature Paradigm

The following lists the intrinsic features of the gradational signature paradigm.

- *Key splitting*: By splitting the full private key into two partial private keys, the gradational signature paradigm enables the primary signer and the cosigner

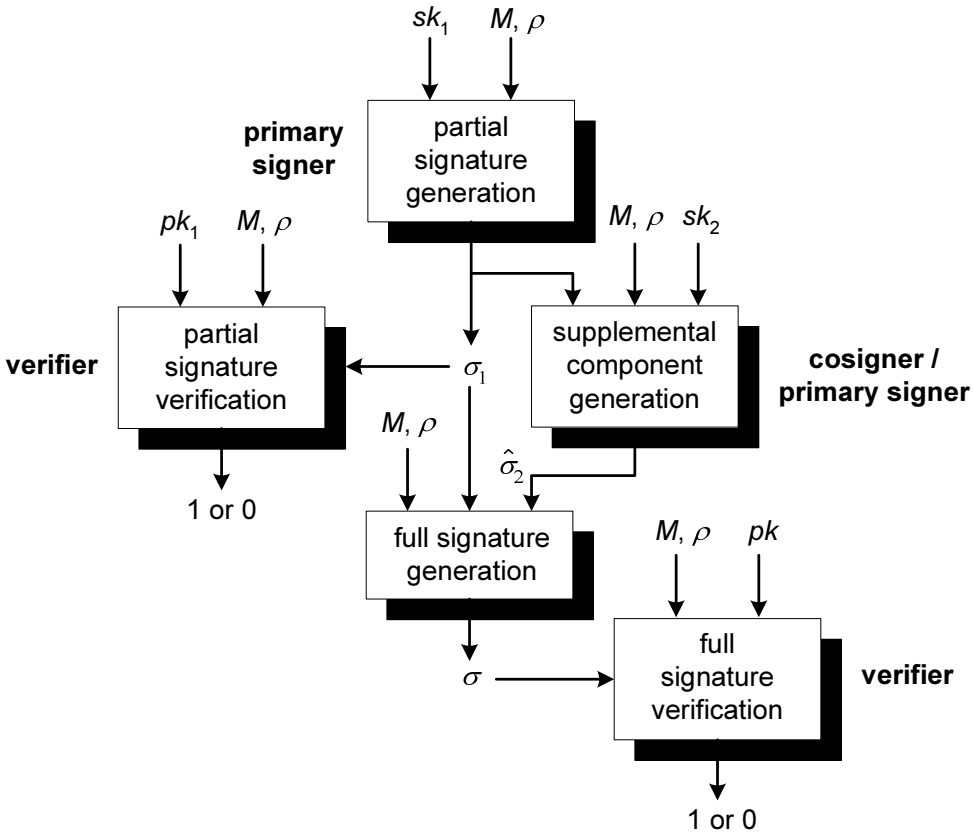

Fig. 3.1. The gradational signature paradigm.

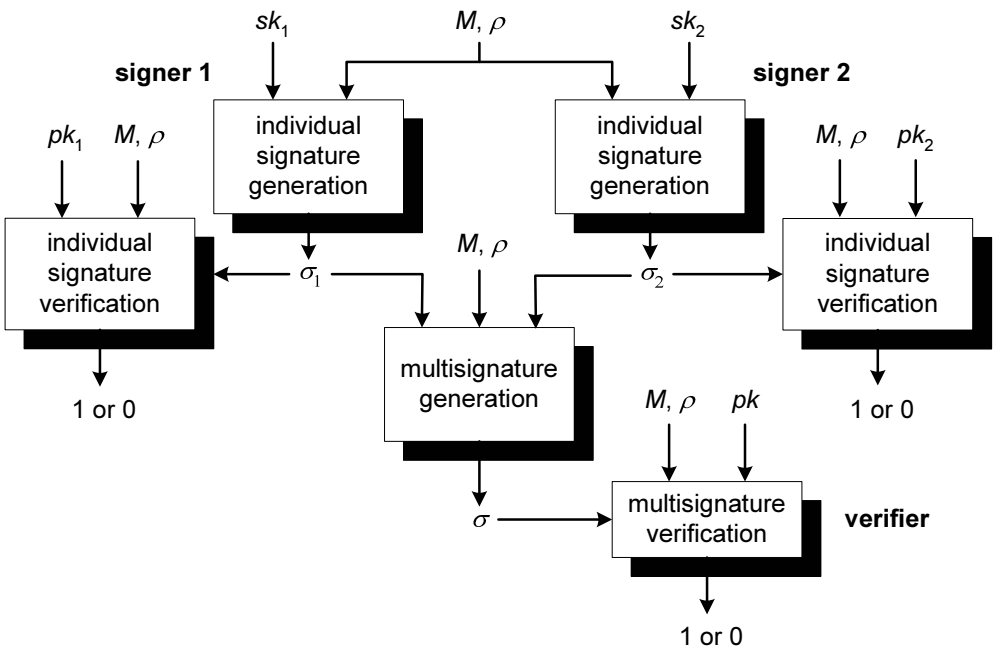

Fig. 3.2. The multisignature paradigm.

to conjointly compute the full signature. The keys are generated such that both partial private keys are required to compute the full signature, and neither partial private key reveals any useful information about the other partial private key or the full signature. These properties are essential for ensuring the fairness of the exchange when the gradational signature paradigm is applied to fair-exchange protocols.

- *Gradational structure*: As shown in Figure 3.1, a full signature is created in two phases: first, a partial signature is computed, and then the full signature is computed by combining the partial signature and the supplemental component. The keys sk_1 and sk_2 are required to compute the partial signature and the supplemental component, respectively. One of the most challenging aspects of designing optimistic fair-exchange protocols is the implementation of the fairness primitive (see Subsection 3.2.1). The gradational generation process of our signature paradigm is ideally suited for implementing such a fairness

primitive. We show in Section 3.5 how a partial signature and its associated partial public key are used to create a fairness primitive.

- *Compatibility*: The full signature is identical to a standard signature in form and functionality (cf. convertible undeniable signatures). That is, the verification algorithm of the full signature is the same as the algorithm used to verify the underlying standard signature scheme (of which the gradational signature scheme is based on). This means that it is possible to implement the fair-exchange feature on top of existing e-commerce exchange protocols without introducing unnecessary complexities.

- *Flexibility*: The gradational signature paradigm can be applied to a wide range of existing signature schemes. It can be applied to signatures based on the infeasibility of factoring large integers (e.g., RSA) as well as signatures based on the discrete logarithm problem (e.g., ElGamal). In Section 3.4, we describe four gradational signature schemes based on well-known signature algorithms.

- *Efficiency*: Unlike other special-purpose primitives used to create the fairness primitive (e.g., convertible undeniable signatures or verifiable escrow), gradational signatures do not require the signing or verifying entities to perform expensive operations. They are only required to perform operations that are typically required in generating/verifying signatures. Schemes such as verifiable escrow require the entities to engage in a "cut-and-choose" zero-knowledge protocol, which is quite costly in terms of computation and communication overhead.

3.3.3 Application of the Gradational Signature Paradigm to Optimistic Protocols

The gradational signature paradigm can be directly applied in the construction of an optimistic fair-exchange protocol. Specifically, elements of the paradigm are

used to construct the fairness primitive—the partial signature σ_1 is the *commitment*, and a signed certificate that certifies the partial public key pk_1 is the *voucher* (see Subsection 3.2.1). We would like to emphasize that this voucher is *not* issued by the CA. To clarify how optimistic protocols can be constructed using the gradational signature paradigm, we give an example, in the next paragraph, where our paradigm is applied to the optimistic-protocol framework given in Subsection 3.2.1.

Imagine a scenario where Alice (primary signer) and Bob (verifier) are trying to exchange signatures in a fair way, and Charlie (cosigner) is an entity trusted by both parties. We use Alice's full signature σ_A as the signature offered in the exchange, and use the corresponding partial signature σ_{A1} as her commitment. Before the exchange protocol, a *registration protocol* needs to be performed between Alice and Charlie so that Charlie can verify the relations of the keys, and issue a voucher if the keys are valid. This needs to be performed only once, after which it can support any number of exchanges.

REGISTRATION. In the registration protocol, Alice provides Charlie with the partial private key sk_{A2}, the public keys pk_A and pk_{A1}, and a certificate C_{CA} issued by the certification authority (CA) that certifies pk_A. The validity of pk_A is assured by C_{CA}, but sk_{A2} and pk_{A1} have no such certificates that prove their validity. The main theme of the registration protocol is Alice proving to Charlie that those keys are indeed valid (without revealing her partial private key sk_{A1}). The keys must satisfy a predefined set of relations to be valid. The validity of those keys must be checked to guarantee that Charlie can construct Alice's full signature σ_A from σ_{A1} and sk_{A2} if needed (i.e., in case a dispute needs to be resolved). The procedure for proving their validity depends on the specific gradational signature scheme used. After checking the keys' validity, Charlie creates a signed voucher V_C that assures the following: (i) pk_{A1} is Alice's valid partial public key, and (ii) Charlie can compute Alice's full signature using the partial signature if needed. We assume that pk_A and pk_{A1} can be recovered

from C_{CA} and V_C, respectively.[6] Charlie gives the voucher V_C to Alice, which is later used by Alice in the exchange protocol to convince Bob that her partial signature is indeed an honest commitment.

After the registration protocol has been successfully executed, Alice can initiate an exchange protocol with Bob. Recall that the registration phase is performed only once—after one successful execution, it can support any number of exchange protocols.

EXCHANGE.

1. Alice needs to provide a token that shows her commitment to the transaction without revealing the full signature itself. Alice's partial signature σ_{A1} is used for this purpose. The value σ_{A1} itself reveals no useful information about σ_A, but acts as a commitment because σ_A can be computed from σ_{A1} and sk_{A2}. Along with σ_{A1}, V_C and C_{CA} are sent to Bob.

2. Bob extracts the necessary keys and parameters from C_{CA} and V_C, and verifies σ_{A1}. If σ_{A1} is valid, he sends his signature σ_B to Alice. Otherwise, he stops the protocol.

3. If σ_B is valid, Alice computes her full signature σ_A, and sends it to Bob. Otherwise, she stops the protocol.

4. If σ_A is valid, Bob ends the exchange protocol. Otherwise, he initiates a dispute resolution protocol.

DISPUTE RESOLUTION.

1. Bob sends to Charlie σ_{A1}, C_{CA}, V_C, and σ_B.

2. Charlie extracts the keys and parameters from C_{CA} and V_C, and verifies σ_{A1} and σ_B. Here, we assume that Charlie knows Bob's public key needed to verify

[6]The plaintext of the certificate/voucher can be extracted from the certificate/voucher, either because the signature scheme (used for generating the certificate/voucher) is capable of message recovery, or because the plaintext is concatenated with the signature.

σ_B. If the items are valid, he computes the supplemental component $\hat{\sigma}_{A2}$ (using sk_{A2}), and computes σ_A by combining σ_{A1} and $\hat{\sigma}_{A2}$. The value σ_A is sent to Bob, and σ_B is forwarded to Alice.

3.4 Gradational Signature Schemes

In this section, we give details on how to construct gradational signature schemes based on four well-known conventional signature schemes. The security of the first two signature schemes are based on the intractability of factoring large integers, and the security of the other two is based on the intractability of the discrete logarithm problem.

3.4.1 Gradational Signatures Based on RSA Signatures

The RSA Signature Scheme

Before we discuss the RSA-based gradational signature scheme, we review the underlying RSA signature scheme [RiS78]. We consider the standard "hash-then-decrypt" variety. The signature space is the set of integers modulo N, denoted as $\mathbb{Z}_N$, where N is a product of two distinct primes p and q. The parameter N is chosen such that $2^{k-1} \leq N < 2^k$ holds for some security parameter k. The public key[7] is obtained by selecting a random integer e, $1 < e < \phi(N)$, such that $\gcd(e, \phi(N)) = 1$. Here, $\phi(\cdot)$ is the Euler's totient function. The private key is generated by finding the unique integer d, $1 < d < \phi(N)$, such that

$$ed \equiv 1 \,(\mathrm{mod}\, \phi(N)).$$

A signature σ on message M is created by computing

$$H(M)^d \bmod N,$$

[7]In some literature, (e, N) and d are called the public and private keys, respectively.

where $H\colon \{0,1\}^* \to \mathbb{Z}_N$ is a public hash function. (Here, $\{0,1\}^*$ denotes a binary string of arbitrary finite length.) To thwart attacks, the hash function should have the preimage, second-preimage, and collision-resistance properties (see p. 323, [MeV96]). The signing space[8] is $\mathbb{Z}_N$. The signature σ is considered to be a valid signature of M if

$$\sigma^e \bmod N = H(M).$$

RSA Gradational Signatures

Boyd [Boy89] proposed an RSA-based multisignature scheme that allows two signers to compute a multisignature efficiently. The core idea behind his scheme is to *multiplicatively* split the private key d into two keys d_1 and d_2, each associated with a different signer. That is,

$$d \equiv d_1 d_2 \pmod{\phi(N)}.$$

The multisignature computation is then based on the equation

$$H(M)^d \equiv H(M)^{d_1 d_2} \pmod{N}.$$

In [BeS01], the authors analyze a set of signature protocols based on Boyd's scheme. Like Boyd's multisignature scheme, our RSA-based gradational signature scheme also splits d into two partial private keys, but the splitting is done *additively* instead of multiplicatively.

In the gradational signature scheme, both partial private keys, d_1 and d_2, are generated by the primary signer. In fact, the primary signer generates all the keys. The primary signer reveals d_2 (but *not* d_1) to the cosigner after it has been generated. Because the primary signer is not trusted by the cosigner (although the reverse is true), the former must prove to the latter that d_2 is valid. The key d_2 is valid if it satisfies certain relations (relative to the other keys), which we describe in the descriptions of the key generation process. This requires that the generation process

[8]The set of elements to which the signature transformation is applied.

of the partial and full signatures be *different* from conventional RSA signatures. However, *compatibility* is preserved (see Subsection 3.3.2). We describe the RSA-based gradational signature scheme in the following paragraphs.

We restrict the modulus N to be a product of *safe primes* p and q, that is, p and q are primes such that $p = 2p' + 1$ and $q = 2q' + 1$ with p' and q' primes. Let $\mathbb{Z}_N^*$ denote the multiplicative group of integers modulo N. The signing space is the set of quadratic residues modulo N, denoted as Q_N. By definition, $Q_N \subset \mathbb{Z}_N^*$ is the set of elements $a \in \mathbb{Z}_N^*$ such that there exists an $x \in \mathbb{Z}_N^*$ with $x^2 \equiv a \,(\mathrm{mod} N)$. Note that Q_N is a cyclic subgroup of $\mathbb{Z}_N^*$, and that

$$|Q_N| = \frac{|\mathbb{Z}_N^*|}{4} = p'q', \tag{3.1}$$

where $|\cdot|$ denotes the order of a group. The private key d is split additively. That is,

$$d \equiv d_1 + d_2 \,(\mathrm{mod}\, \lambda),$$

where $\lambda = p'q'$. Observe that the modulus used above is λ instead of $\phi(N)$ because the signing space is Q_N. The full signature computation is then based on the congruence

$$H(M)^d \equiv H(M)^{d_1} \cdot H(M)^{d_2} \,(\mathrm{mod}\, N),$$

where $H\colon \{0,1\}^* \rightarrow Q_N$ is a public hash function. In addition to splitting d, we need to create a partial public key e_1 associated with the partial private key d_1 that satisfies

$$d_1\, e_1 \equiv 1 \,(\mathrm{mod}\, \lambda).$$

The key e_1 is used by the verifier to verify the primary signer's partial signature, and is made public. For the following discussions, we use the results of the following lemma proven in [GeK97].

Lemma 3.4.1 *Let $N = pq$, where $p = 2p' + 1$, $q = 2q' + 1$, and p, q, p', and q' are all prime numbers. We assume, without loss of generality, that $p < q$. Then,*

1. *The order of elements in $\mathbb{Z}_N^*$ is one of the integers in the set $\{1, 2, p', q', 2p', 2q', p'q', 2p'q'\}$.*

2. *Given an element* $a \in \mathbb{Z}_N^* \setminus \{-1, 1\}$ *such that* $\text{ord}(a) < p'q'$*, then either* $\gcd(a - 1, N)$ *or* $\gcd(a + 1, N)$ *is a prime factor of* N*.* ($\text{ord}(\cdot)$ *denotes the order of a group element.*)

As a consequence of the above lemma, any element $a \in \mathbb{Z}_N^* \setminus \{-1, 1\}$ selected by a party that does not know the factorization of N satisfies $\text{ord}(a) \geq p'q'$ with overwhelming probability. We can use the results of the lemma to show that the following corollary is true.

Corollary 3.4.1 *Let* $N = pq$*, where* $p = 2p' + 1$*,* $q = 2q' + 1$*, and* p*,* q*,* p'*, and* q' *are all primes. Any element* $a \in Q_N$*,* $a \neq \pm 1$*, such that* $\gcd(a + 1, N)$ *and* $\gcd(a - 1, N)$ *are not prime factors of* N*, satisfies* $\text{ord}(a) = p'q'$*.*

Proof By *Lagrange's Theorem*[9] and (3.1), the order of elements in Q_N is one of the integers in the set $\{1, p', q', p'q'\}$. Because Q_N is a subgroup of $\mathbb{Z}_N^*$, the results of Lemma 3.4.1 apply to any element $a \in Q_N$. That is, any element $a \in Q_N$, $a \neq \pm 1$, such that $\gcd(a + 1, N)$ and $\gcd(a - 1, N)$ are not prime factors of N, satisfies $\text{ord}(a) \geq p'q'$. Hence, it must be true that $\text{ord}(a) = p'q'$. ■

The above fact will play a vital role in the registration stage of our fair-exchange protocol using RSA-based gradational signatures. The details of the key and signature generation processes are given below.

KEY GENERATION. The primary signer runs the key generation algorithm. The key generation algorithm, on input of some security parameter k, first selects two safe primes p and q such that their product N satisfies $2^{k-1} \leq N < 2^k$. The full public key is obtained by selecting a random integer e, $1 < e < \lambda$, such that $\gcd(e, \lambda) = 1$. The full private key is generated by finding the unique integer d, $1 < d < \lambda$, such that

$$ed \equiv 1 \pmod{\lambda}.$$

[9]Lagrange's theorem states that if G is a finite group and H is a subgroup of G, then $|H|$ divides $|G|$. Thus, if $a \in G$, then $\text{ord}(a)$ divides $|G|$.

Now, the partial public key e_1, $1 < e_1 < \lambda$, is chosen randomly such that $\gcd(e_1, \lambda) = 1$. The corresponding (primary signer's) partial private key is generated by finding the unique integer d_1, $1 < d_1 < \lambda$, such that

$$e_1 d_1 \equiv 1 \,(\mathrm{mod}\, \lambda).$$

Next, the (cosigner's) partial private key d_2 is computed as

$$d_2 = d - d_1 \bmod \lambda.$$

The keys satisfy the following relations:

$$\begin{aligned} ed &\equiv 1 \,(\mathrm{mod}\, \lambda), \\ e_1 d_1 &\equiv 1 \,(\mathrm{mod}\, \lambda), \\ d_1 + d_2 &\equiv d \,(\mathrm{mod}\, \lambda). \end{aligned}$$

SIGNATURE GENERATION. The primary signer, using the partial private key d_1, generates the partial signature

$$\sigma_1 = H(M)^{d_1} \bmod N.$$

The cosigner, using the partial private key d_2, generates the supplemental component

$$\hat{\sigma}_2 = H(M)^{d_2} \bmod N.$$

These items are multiplied modulo N to form the full signature σ. That is,

$$\begin{aligned} \sigma &= \sigma_1 \cdot \hat{\sigma}_2 \bmod N, \\ &= H(M)^{d_1 + d_2} \bmod N. \end{aligned}$$

The partial signature σ_1 is considered valid if and only if

$$\sigma_1^{e_1} \bmod N = H(M).$$

The full signature σ is verified in the same way using the full public key e instead of the partial public key e_1.

Recall that in Boyd's scheme, d is split multiplicatively, whereas in our scheme, we split d additively. It is necessary to take the latter approach to avoid compromising the security of our protocol. Specifically, if d is split multiplicatively, the cosigner is able to create multisignatures on his own without the help of the primary signer. The cosigner can use the three keys available to him—partial private key d_2, partial public key e_1, and the joint public key e—to compute d_1. In Subsection 3.7.1, we give details on how this can occur. Note that, in the fair-exchange protocol of Boyd and Foo [BoF98], an RSA-based convertible undeniable signature scheme, which splits d multiplicatively, is used. In this signature scheme, however, the partial public key e_1 does not exist. Thus, although d is split multiplicatively, it does not cause any security problems there.

3.4.2 Gradational Signatures Based on Guillou-Quisquater Signatures

The Guillou-Quisquater Signature Scheme

Before we discuss the gradational signature scheme, we review the underlying Guillou-Quisquater (GQ) signature scheme [GuQ88a]. A trusted authority (perhaps the CA) chooses secret distinct primes p and q, and forms $N = pq$. Next, he selects an integer $v \in \{1, \ldots, N-1\}$ such that $\gcd(v, \phi(N)) = 1$. The integers v and N are the public system parameters shared by all signers. Note that sharing the modulus N among all signers renders the RSA scheme vulnerable to the "common modulus attack" (see p. 289, [MeV96]), but does not pose a security risk in the GQ cryptosystem. The authority also selects a public key J, $1 < J < N$, and a private key β for each signer. The integer J is a numeric representation of each signer's unique identity (e.g., name, bank account number, chip serial number, etc.), and β is determined by finding the solution to

$$J\beta^v \equiv 1 \pmod{N}.$$

The system parameters (v, N) and public key J are made available to all users with guaranteed authenticity (perhaps via certificates), and β is transmitted to the signer via a secure channel. The scheme can also be configured so that each signer chooses its own modulus N and the private key β. In this case, system parameters (v, N) are unique for each signer.

For every signing operation, the signer selects a different random integer r, and computes $T = r^v \mod N$. Using these components,

$$d = H(M \parallel T),$$

where M is the message being signed, and

$$D = r\beta^d \mod N$$

are computed. The pair (D, d) is the signature. To verify a signature, one needs to compute $T' = D^v J^d \mod N$ and $d' = H(M \parallel T')$, and compare d with d'. The signature is valid if and only if $d = d'$.

GQ Gradational Signatures

We construct the GQ gradational signature scheme by modifying the GQ multisignature scheme [GuQ88b].

KEY GENERATION. The values v, N, and J are selected in the same manner as in the GQ signature scheme. The primary signer randomly picks an integer J_1 that is relatively prime to N such that $1 < J_1 < N$, and computes an integer J_2 that satisfies

$$J = J_1 J_2 \mod N.$$

The value J_1 is the partial public key, and J is the full public key. The integer J_2 has no function as a verification key, and it is used only to create the partial private key β_2. The partial private keys β_1 and β_2 are determined by finding the solutions to

$$J_i \beta_i^v \equiv 1 \,(\mathrm{mod}\, N), \quad i = 1, 2,$$

respectively. If (v, N) are shared by all the primary signers, then the partial private keys are computed by a trusted authority, and securely transmitted to each primary signer. If the scheme is configured so that each primary signer chooses her own (v, N), then each individual primary signer generates the partial private keys (β_1, β_2). The full private key is given by

$$\beta = \beta_1 \beta_2 \bmod N.$$

SIGNATURE GENERATION. For every instance of a gradational signature operation, the primary signer selects a random integer r, and computes

$$T = r^v \bmod N.$$

Using these integers, the following values are computed:

$$\begin{aligned} d &= H(M \parallel T), \\ D_1 &= r\beta_1^d \bmod N. \end{aligned}$$

The pair (D_1, d) is the partial signature. To verify the partial signature, one needs to compute

$$d'' = H(M \parallel T''),$$

where $T'' = D_1^v J_1^d \bmod N$. The partial signature is valid if and only if $d = d''$. The supplemental component is generated as

$$D_2 = \beta_2^d \bmod N.$$

Note that D_2 is computed *differently* from D_1. To compute the full signature, one must first compute $D = D_1 \cdot D_2 \bmod N$. The pair (D, d) is the full signature. To verify the full signature, the verifier first computes the following:

$$\begin{aligned} T' &= D^v J^d \bmod N, \\ d' &= H(M \parallel T'). \end{aligned}$$

The full signature is valid if and only if $d = d'$.

3.4.3 Gradational Signatures Based on Meta-ElGamal Signatures

Harn's Meta-ElGamal Signature Scheme

MacKenzie and Reiter [MaR01b] showed how the Digital Signature Algorithm (DSA) [Kra93] or the ElGamal signature scheme [Elg85] can be converted into a two-party multisignature scheme. Although possible, converting an ElGamal signature into a multisignature incurs considerable computation overhead. This is because, in the ElGamal multisignature scheme, (i) a shared secret must be inverted, and (ii) two shared secrets must be multiplied. One can see why such computations are necessary by observing the *signing equation* of the ElGamal signature scheme, which is given by

$$s = \kappa^{-1}(H(M) - xr) \bmod (p-1),$$

where κ is the one-time secret integer, M is the message, x is the private key, r is the one-time public integer, and p is a prime number. To convert a standard (one signer) signature into a multisignature, κ and x must be shared among multiple signers.

Despite improvements over previous approaches, the scheme of MacKenzie and Reiter still requires relatively expensive computations. One can devise much more efficient ElGamal-based multisignatures by altering the signing equation, and such variants are studied in [HoP94]. Horster et al. investigated several ElGamal variants, and coined the term *Meta-ElGamal signatures* to denote them. These variants are more flexible, that is, efficient multisignatures can be readily constructed from them. In [Har94b], Harn proposed an ElGamal-variant signature scheme and its multisignature extension. Horster et al. [HoM95] showed that this multisignature scheme is one of the most efficient ElGamal-based multisignature schemes. In the next paragraph, we briefly review Harn's signature scheme.

A trusted authority chooses a large prime p (of least 768 bits) and a generator $g \in \mathbb{Z}_p^*$ as public system parameters, and publishes these items. The signer chooses a random number $x \in \mathbb{Z}_{p-1}^*$ as her private key, and computes $y = g^x \bmod p$ as her public key. The signer also chooses a random number $\kappa \in \mathbb{Z}_{p-1}^*$ as the one-time secret

integer, and computes $r = g^{\kappa} \bmod p$ as the one-time public integer. To sign message M, the signer computes s via the signing equation

$$s = x(H(M) + r) - \kappa \bmod (p-1),$$

where H is a hash function such that $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$. The pair (s, r) is the signature. Its correctness is verified by checking the congruence relation

$$rg^s \equiv y^{H(M)+r} \pmod p.$$

Meta-ElGamal Gradational Signatures

We directly use Harn's multisignature scheme to create Meta-ElGamal gradational signatures. However, unlike the multisignature scheme, all the partial private keys are generated by the primary signer.

KEY GENERATION. A trusted authority chooses a large prime p and a generator $g \in \mathbb{Z}_p^*$ as the public system parameters. The primary signer obtains the partial private keys, x_1 and x_2, by randomly selecting distinct integers from $\mathbb{Z}_{p-1}^*$. The full private key is

$$x \equiv x_1 + x_2 \pmod{(p-1)}.$$

The primary signer computes the partial public key as $y_1 = g^{x_1} \bmod p$. The full public key y is computed using the relation

$$y = y_1 \cdot g^{x_2} \bmod p.$$

SIGNATURE GENERATION. For every signing operation, the primary signer randomly selects a one-time secret integer $\kappa_1 \in \mathbb{Z}_{p-1}^*$, and computes the corresponding one-time public integer $r_1 = g^{\kappa_1} \bmod p$. In the same manner, the cosigner obtains κ_2 and $r_2 = g^{\kappa_2} \bmod p$. To obtain the partial signature, the primary signer needs to compute s_1 using the signing equation

$$s_1 = x_1(H(M) + r) - \kappa_1 \bmod (p-1),$$

where $r = r_1 r_2 \bmod p$. The partial signature is (s_1, r_1, r). The partial signature is verified by checking the relation

$$r_1 g^{s_1} \equiv y_1^{H(M)+r} \pmod{p}.$$

The cosigner's supplemental component is given by

$$s_2 = x_2(H(M) + r) - \kappa_2 \bmod (p-1).$$

To obtain the full signature, one needs to compute $s = s_1 + s_2 \bmod (p-1)$. The pair (s, r) is the full signature, which is verified by checking the congruence relation

$$r g^s \equiv y^{H(M)+r} \pmod{p}.$$

3.4.4 Gradational Signatures Based on Schnorr Signatures

The Schnorr Signature Scheme

Before we discuss Schnorr-based gradational signatures, we review the underlying Schnorr signature scheme [Sch91]. A trusted authority selects a large prime p such that $p-1$ is divisible by another prime q, and selects a generator g of the unique cyclic group of order q in $\mathbb{Z}_p^*$. The system parameters p, q, and g are published. All potential signers can use the same system parameters to create their respective signatures. Each signer randomly selects a private key $x \in \{1, \ldots, q-1\}$, and computes its corresponding public key $y = g^x \bmod p$. For every signing operation, the signer randomly selects a one-time secret integer κ, $1 \leq \kappa \leq q-1$, and computes $r = g^\kappa \bmod p$. Let H represent a hash function such that $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$. To generate a signature, one needs to compute $e = H(M \parallel r)$ and

$$s = xe + \kappa \bmod q,$$

where M is the message being signed. The signature is the pair (s, e). To verify a signature, one needs to compute the following values:

$$\begin{aligned} r' &= g^s y^{-e} \bmod p, \\ e' &= H(M \parallel r'). \end{aligned}$$

The signature is valid if and only if $e = e'$.

Schnorr Gradational Signatures

To generate Schnorr-based gradational signatures, we use the multisignature scheme of Ohta and Okamoto [OhO99] with some modifications. In the gradational signature scheme, all the keys (including the cosigner's partial private key) are generated by the primary signer.

KEY GENERATION. A trusted authority selects a large prime p such that $p-1$ is divisible by another prime q, and selects a generator g of the unique cyclic group of order q in $\mathbb{Z}_p^*$. The system parameters p, q, and g are published. All potential primary signers can use the same system parameters to create their respective signatures. The primary signer obtains the partial private keys, x_1 and x_2, by randomly selecting distinct integers from $\{1, \ldots, q-1\}$. The full private key is

$$x \equiv x_1 + x_2 \,(\mathrm{mod}\, q).$$

The primary signer computes the partial public key as $y_1 = g^{x_1} \bmod p$. The full public key y is computed using the relation

$$y = y_1 \cdot g^{x_2} \bmod p.$$

SIGNATURE GENERATION. For every signing operation, the primary signer randomly selects a one-time secret integer $\kappa_1 \in \{1, \ldots, q-1\}$, and computes the corresponding one-time public integer $r_1 = g^{\kappa_1} \bmod p$. In the same manner, the cosigner obtains κ_2 and $r_2 = g^{\kappa_2} \bmod p$. To obtain the partial signature, the primary signer needs to compute s_1 using the signing equation

$$s_1 = x_1 e + \kappa_1 \bmod q,$$

where $e = H(M \parallel r)$ and $r = r_1 \cdot r_2 \bmod p$. The partial signature is the three-tuple (s_1, r_1, r). To verify the partial signature, the verifier needs to compute $e = H(M \parallel r)$

and $r_1' = g^{s_1} y_1^{-e} \bmod p$. The partial signature is valid if and only if $r_1 \equiv r_1' \pmod p$. The cosigner's supplemental component is computed as

$$s_2 = x_2 e + \kappa_2 \bmod q.$$

To obtain the full signature, one needs to compute e and $s = s_1 + s_2 \bmod q$. The pair (s, e) is the full signature. To verify the full signature, the verifier needs to compute $r' = g^s y^{-e} \bmod p$ and $e' = H(M \parallel r')$. The full signature is valid if and only if $e = e'$.

3.5 The Fair-Exchange Protocols

3.5.1 Fair-Exchange Protocols with RSA Gradational Signatures

In the following protocol, Alice is the customer (or primary signer), Charlie is the TTP (or cosigner), and Bob is the merchant (or verifier). Using the following optimistic protocol, Alice purchases digital goods from Bob, and Bob receives her "digital check" in return. Alice's full signature acts as the digital check. Alice computes the digital check by signing the "purchase information" M. Information such as Alice's identity, Bob's account number, merchandise price, merchandise description, transaction date, and a serial number are contained in M. We assume that the public keys of the CA and Charlie are known to all parties involved in the transaction.

Before the exchange protocol takes place, Alice has to execute a series of message exchanges with the CA and Charlie—we call this the *registration protocol*. The central theme of the registration protocol is Alice proving to Charlie that the following congruences hold without revealing d_1 and λ (with rest of the keys known to Charlie):

$$e_1 d_1 \equiv 1 \pmod \lambda, \tag{3.2}$$

$$(d_1 + d_2) e \equiv 1 \pmod \lambda. \tag{3.3}$$

This is done by using a *reference message* ω and its corresponding *reference signature* Ω, and is described in the following steps. A similar approach is used to construct an undeniable signature scheme in [GeK97].

REGISTRATION. The registration protocol needs to be performed only *once*, after which it can support any number of exchanges. Note that most optimistic protocols require a registration stage (e.g., [Ate99, BoF98]). We assume that the registration protocol is performed via confidential and authenticated channels. In practice, such channels can be implemented by using message authentication codes (MAC) in conjunction with encryption schemes.

1. Alice first generates the parameters N, p, and q and the keys e, e_1, d, d_1, and d_2. Alice then contacts the CA to get the joint public key e certified. At this stage, Alice has to prove to the CA that N is a product of safe primes (without revealing p and q). This can be done using the zero-knowledge protocol of Camenisch and Michels [CaM99].

2. After verifying the construction of N, the CA issues a signed certificate C_{CA} to Alice. This certificate certifies the joint public key e and the modulus N. We assume that the values of e and N can be extracted from C_{CA}.[10]

3. Alice sends C_{CA}, e_1, and d_2 to Charlie. Note that d_2 is Charlie's partial private key, and e_1 is the partial public key.

4. Charlie checks the validity of C_{CA}. He then randomly chooses an integer $\bar{\omega} \in \mathbb{Z}_N^* \setminus \{-1, 1\}$, and checks that $\gcd(\bar{\omega} + 1, N)$ and $\gcd(\bar{\omega} - 1, N)$ are not prime factors of N. He then computes $\omega = \bar{\omega}^2 \bmod N$. Note that ω is a generator of Q_N (see Subsection 3.4.1). Charlie sends the reference message ω to Alice.

5. Alice computes the reference signature

$$\Omega = \omega^{d_1} \bmod N,$$

and sends this to Charlie.

[10]The plaintext of the certificate can be extracted from the certificate, either because the signature scheme (used for generating the certificate) is capable of message recovery, or because the plaintext is concatenated with the signature.

6. Now, Alice proves to Charlie that Ω is a power of ω without revealing d_1. This is done using the zero-knowledge protocol discussed in Subsection 3.2.3.

7. Charlie checks that Ω is constructed correctly by verifying the following congruence relations:

$$\begin{aligned} \Omega^{e_1} &\equiv \omega \,(\mathrm{mod}\, N), \\ \Omega^{e} \cdot \omega^{d_2 e} &\equiv \omega \,(\mathrm{mod}\, N). \end{aligned}$$

8. If the verifications (of Steps 4, 6, and 7) are passed, Charlie accepts Alice's claim that the congruence relations of (3.2) and (3.3) hold. He then creates a voucher V_C by signing on e_1. We assume that e_1 can be extracted from V_C. Charlie stores his partial private key d_2, and sends V_C to Alice.

Remark 3.5.1 The certificate C_{CA} certifies e and N. It might also include descriptions of the group Q_N and the hash function H: $\{0,1\}^* \rightarrow Q_N$.

Remark 3.5.2 After Step 6, Charlie can assume that $\Omega = \omega^{\delta} \bmod N$ for some integer δ (at this point, he cannot be sure that $\delta = d_1$). Because ω is a generator (i.e., $\mathrm{ord}(\omega) = p'q'$), the congruences in Step 7 imply that $\delta e_1 \equiv 1 \,(\mathrm{mod}\, \lambda)$ and $(\delta + d_2)e \equiv 1 \,(\mathrm{mod}\, \lambda)$, respectively. Hence, $\delta = d_1$.

Remark 3.5.3 The voucher V_C is a signed statement from Charlie that assures the following: (i) e_1 is Alice's valid partial public key, and (ii) the algebraic relations between the keys have been verified, and, as a result, Charlie can generate a full signature from the corresponding partial signature. The first statement is explicitly shown by the content of the voucher, and the second statement is implicitly assumed to be true—Charlie will not create the voucher without verifying the relations of the keys. Therefore, the voucher conveys the following important semantics: *Charlie can convert any signature on some arbitrary message M, which is verified using (e_1, N), to a signature on M that is verified with (e, N).*

Remark 3.5.4 If the number of users is large, it requires Charlie to securely store a correspondingly large number of partial private keys d_2 (one for each primary signer). This can be avoided by using the following technique: Charlie concatenates d_2 and Alice's unique identification, ID_A, to form $d_2||ID_A$, and then encrypts this value via some symmetric-key encryption algorithm $E_\psi(\cdot)$, where ψ denotes the secret key. Charlie then creates a signature of the concatenated value of e_1 and $E_\psi(d_2||ID_A)$. That is,

$$Sig_C(e_1||E_\psi(d_2||ID_A)),$$

where $Sig_C(\cdot)$ denotes Charlie's signature algorithm. This value is used as the voucher V_C. Now, Charlie can extract d_2 from V_C (using ψ), and only needs to securely store ψ.

EXCHANGE. Alice initiates the protocol with Bob. We assume that Alice and Bob have gone through a negotiation process to agree on the purchase information M prior to the start of the exchange protocol. This process may be as simple as Alice choosing fixed-priced goods from Bob's website. Note that Alice's full signature on M is her digital check. In addition, Alice and Bob agree on a session key using some key-agreement protocol (e.g., Diffie-Hellman key agreement). The session key is used to encrypt the digital merchandise to deter eavesdropping. We use the basic optimistic protocol of Subsection 3.2.1 as the protocol framework. The following steps describe the exchange protocol.

1. Alice computes her partial signature σ_1 (using d_1), and sends Bob C_{CA}, V_C, and σ_1.

2. Bob, using C_{CA}, verifies N and e. He then checks the validity of V_C, and verifies whether

   $$\sigma_1^{e_1} \bmod N = H(M).$$

   If everything is in order, Bob encrypts the digital merchandise μ with some symmetric encryption algorithm $E_\gamma(\cdot)$, where γ is the secret encryption key (i.e.,

the session key). The encrypted merchandise $E_\gamma(\mu)$ is sent to Alice. However, if any one of the items received from Alice is invalid, Bob does not send the merchandise, and stops the protocol.

3. Alice decrypts and verifies the merchandise. If Alice is satisfied with the merchandise, she computes her full signature σ, and sends it to Bob. Otherwise, Alice stops the protocol.

4. Bob verifies σ, and if it is valid, ends the protocol. Otherwise, Bob initiates the dispute resolution protocol.

Remark 3.5.5 Note that σ_1 and V_C constitute the fairness primitive.

Remark 3.5.6 The exchange protocol above requires the use of *timestamps* and *reliable channels*[11] (see [Aso98]) to ensure timely termination. Note that the protocol framework of Asokan et al. [AsS98, AsS00] does not require timestamps for timely termination, and only requires *resilient channels*[12]. We can also apply our approach to their framework. Doing so is straightforward—replace their multistep verifiable escrow procedure with the transmission of a partial signature and its associated voucher (i.e., the fairness primitive). Of course, the primary signer and the TTP would have to go through a one-time registration protocol a priori to the exchange protocol.

Remark 3.5.7 The above exchange protocol does not require confidential and authenticated channels. Observe that Alice's digital check (i.e., σ) is of no value to an eavesdropper because it specifies the intended recipient of the check. In addition, the digital merchandise is encrypted before transmission, and hence it is useless to an eavesdropper.

Figure 3.3 shows the messages exchanged between Alice and Bob in the exchange protocol when both parties act honestly.

[11] A channel that is always operational with a known upper bound of the time delay. An attacker cannot delay any messages beyond the known upper bound.

[12] A channel that is normally operational, but an attacker can succeed in delaying messages by an arbitrary but finite amount of time.

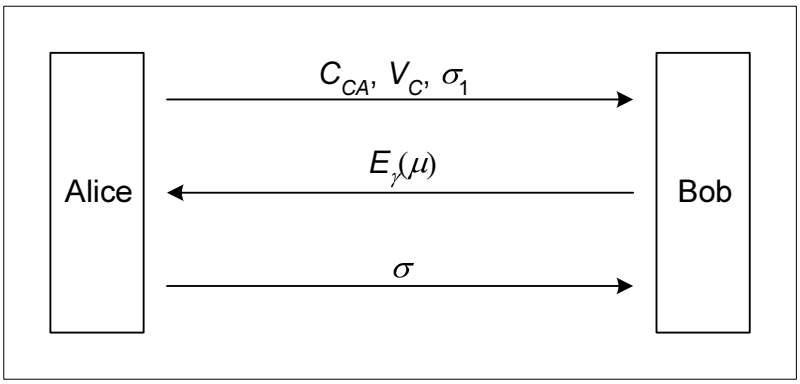

Fig. 3.3. The exchange protocol.

DISPUTE RESOLUTION. If Bob does not receive Alice's payment (i.e., σ) or if σ is invalid, he initiates a dispute resolution protocol by contacting Charlie. We assume that reliable channels exist between the parties.

1. Bob encrypts the session key γ as $AE_{pk_C}(\gamma)$, where pk_C is Charlie's public key, and $AE_{pk_C}(\cdot)$ is an asymmetric encryption algorithm. Bob then sends C_{CA}, V_C, σ_1, M, $E_\gamma(\mu)$, and $AE_{pk_C}(\gamma)$ to Charlie.

2. Charlie decrypts $AE_{pk_C}(\gamma)$, and uses γ to recover μ. He compares the merchandise with its description included in M. Next, he extracts all the system parameters and keys from C_{CA} and V_C, and then verifies σ_1 using those values. If everything is in order, Charlie generates the full signature σ, using σ_1 and his partial private key d_2, via the relation

$$\sigma = \sigma_1 \cdot H(M)^{d_2} \bmod N.$$

   The multisignature is sent to Bob, and the merchandise, after being encrypted with the session key, is forwarded to Alice. If any of the items received from Bob is invalid, Charlie halts the dispute resolution protocol without sending anything to either party.

3.5.2 Fair-Exchange Protocols with Meta-ElGamal Gradational Signatures

Again, Alice is the customer (or primary signer), Charlie is the TTP (or cosigner), and Bob is the merchant (or verifier). We assume that a trusted authority has chosen a large prime p and a generator $g \in \mathbb{Z}_p^*$ as the system parameters, and has published these parameters. Before initiating the registration protocol, Alice obtains the partial private keys, x_1 and x_2, by randomly selecting distinct integers from $\mathbb{Z}_{p-1}^*$. She also generates the partial public key $y_1 = g^{x_1} \bmod p$ and the full public key $y = y_1 \cdot g^{x_2} \bmod p$.

REGISTRATION. We assume that the registration protocol is performed via confidential and authenticated channels.

1. Alice contacts the CA to get the full public key certified. She sends to the CA the values y_1 and g^{x_2}, and shows that they are correctly generated by proving that they are both powers of g without revealing x_1 and x_2. For this purpose, Alice uses the zero-knowledge proof system of Subsection 3.2.3.

2. After verifying the proof, the CA issues a certificate C_{CA} that certifies

$$y = y_1 \cdot g^{x_2} \bmod p \tag{3.4}$$

   as Alice's full public key. The certificate is sent to Alice. We assume that y can be recovered from C_{CA}.

3. Alice sends to Charlie C_{CA}, x_2, and y_1.

4. Charlie extracts y from C_{CA}, and verifies that the relation of (3.4) holds. He then creates a voucher V_C by signing on y_1. We assume that y_1 can be extracted from V_C. Charlie stores his partial private key x_2, and sends V_C to Alice.

In the Meta-ElGamal gradational signature scheme, one needs to compute the one-time integer $r = r_1 r_2 \bmod p$ to generate a partial or full signature. Recall that κ_1 and $r_1 = g^{\kappa_1} \bmod p$ are generated by the primary signer, while κ_2 and $r_2 = g^{\kappa_2} \bmod p$ are

generated by the cosigner. To ensure fairness of the exchange protocol, the verifier (i.e., Bob) should be able to verify that a value of κ_2 chosen by the cosigner (i.e., Charlie) is used to compute $r = r_1 r_2 \bmod p$. If a value of κ_2 that is unknown to Charlie is used to generate r and the partial signature, then Charlie will be unable to generate Alice's full signature using the partial signature and his key x_2. Thus, Charlie would be unable to resolve any potential disputes between Alice and Bob. To show (Bob) that Alice is indeed using κ_2 chosen by Charlie, r_2 needs to be certified by Charlie. For this purpose, Charlie creates a *ticket* T_C, which is Charlie's signature on r_2. We assume that r_2 can be extracted from T_C. Note that because κ_2 cannot be revealed to Bob, r_2 is used instead to compute the ticket. This ticket acts as a guarantee from Charlie that ensures his knowledge of κ_2 associated with the value $r_2 = g^{\kappa_2} \bmod p$. Because of obvious security reasons, a different value of κ_2 should be used for every signing operation. To avoid frequent interactions with Charlie, Alice obtains κ_2's and the corresponding tickets in bulk beforehand. Of course, Alice would need to contact Charlie for new tickets once all the previously issued tickets are exhausted. In our fair-exchange protocols, the use of tickets is *only* needed for signature schemes based on the discrete logarithm problem, and is *not* needed for RSA or GQ signatures.

Note that this technique is similar in concept to the "off-line coupons" used in [AsS00]. However, their scheme has one important disadvantage. For DSS (Digital Signature Standard) signatures, the content of their coupon depends on the particular signature associated with the coupon. This means that the coupons cannot be created before knowing the message that will be signed. This drawback can be overcome, but requires the inclusion of a zero-knowledge proof in the coupon, which makes their scheme less efficient than our approach.

EXCHANGE. We assume that Alice has obtained κ_2 and the associated ticket in advance. Before the exchange protocol is initiated, Alice and Bob agree on the purchase information M and a session key γ. The following steps describe the exchange protocol.

1. Alice generates her partial signature $\sigma_1 = (s_1, r_1, r)$, and sends to Bob C_{CA}, V_C, σ_1, and T_C.

2. Bob extracts y, y_1, and r_2 from C_{CA}, V_C, and T_C, respectively. Next, he verifies whether

$$r = r_1 r_2 \bmod p.$$

   He also verifies the partial signature σ_1 using the partial public key y_1. If everything is in order, Bob encrypts the merchandise μ with a symmetric encryption algorithm $E_\gamma(\cdot)$, where γ is the session key. The encrypted merchandise is sent to Alice. However, if any one of the received items is invalid, Bob does not send the merchandise, and stops the protocol.

3. Alice decrypts and verifies the merchandise. If Alice is satisfied with the merchandise, she computes the full signature $\sigma = (s, r)$, and sends it to Bob. Otherwise, Alice stops the protocol.

4. Bob verifies σ, and if it is valid, ends the protocol. Otherwise, Bob initiates the dispute resolution protocol.

Remark 3.5.8 Charlie needs to securely store all values of κ_2 that are associated with the issued tickets. If the number of tickets is large, this can be a burden. Charlie can avoid such a burden by using the following technique: Charlie randomly chooses κ_2, and computes $r_2 = g^{\kappa_2} \bmod p$; he then encrypts κ_2 using a symmetric encryption algorithm $E_\psi(\cdot)$, where ψ is the secret key; Charlie creates a ticket by concatenating r_2 and $E_\psi(\kappa_2)$, and signing it with an arbitrary signing algorithm $Sig_C(\cdot)$. The value

$$Sig_C(r_2 \parallel E_\psi(\kappa_2))$$

is used as the ticket T_C. Now, Charlie only needs to securely store ψ.

DISPUTE RESOLUTION. If Bob does not receive Alice's check σ, or if σ is invalid, he initiates a dispute resolution protocol.

1. Bob encrypts the session key γ as $AE_{pk_C}(\gamma)$, where pk_C is Charlie's public key, and $AE_{pk_C}(\cdot)$ is an asymmetric encryption algorithm. Bob then sends to Charlie C_{CA}, V_C, T_C, σ_1, M, $E_\gamma(\mu)$, and $AE_{pk_C}(\gamma)$.

2. Charlie decrypts $AE_{pk_C}(\gamma)$, and uses γ to recover μ. Next, he extracts y, y_1, and (κ_2, r_2) from C_{CA}, V_C, and T_C, respectively. He then verifies the partial signature σ_1 by checking the following relations:

$$r = r_1 r_2 \bmod p,$$
$$r_1 g^{s_1} \equiv y_1^{H(M)+r} \pmod p.$$

   If σ_1 is valid, Charlie computes the supplemental component s_2 using the relation

$$s_2 = x_2(H(M) + r) - \kappa_2 \bmod (p-1).$$

   Charlie then computes $s = s_1 + s_2 \bmod (p-1)$. Alice's full signature (s, r) is sent to Bob, and the merchandise is forwarded to Alice. If any of the items received from Bob is invalid, Charlie halts the dispute resolution protocol without sending anything to either party.

3.6 Efficiency Evaluations

In general, the most computationally expensive part of an optimistic exchange protocol is creating and verifying the fairness primitive, that is, the "commitment" c_A and its associated "voucher" V_C (see Subsection 3.2.1). Moreover, modular exponentiation is the most costly operation required to create and verify those items. Recall that in the exchange protocol of Subsection 3.5.1, σ_1 corresponds to a commitment, and V_C corresponds to a voucher.

In this section, we compare the efficiency of our scheme with several previously proposed schemes. The schemes are compared in terms of four criteria: (i) number of modular exponentiations required for creating/verifying the fairness primitive in the exchange protocol, (ii) size of the fairness primitive (in bytes), (iii) whether a

registration protocol is needed, and (iv) whether intermittent contact with the TTP is required. The overall size of the fairness primitive is affected by the size of the voucher, which is determined by the *bandwidth efficiency* of the digital signature algorithm used to generate the voucher (see p. 437 of [MeV96]). Here, we assume that signature algorithms with the message recovery feature are used. Bandwidth efficiency for digital signatures with message recovery is determined by

$$\text{bandwidth efficiency} = \frac{\lg |\mathcal{M}_R|}{\lg |\mathcal{M}_S|},$$

where $\mathcal{M}_R$ is the image space of the redundancy function, $\mathcal{M}_S$ is the signing space, and $\lg(\cdot)$ denotes logarithm of base two.

To date, a diverse array of solutions for constructing the fairness primitive has been proposed. Some solutions are geared toward specific digital signature schemes, whereas others are more general, and can be applied to a wide range of signature algorithms. The majority of those solutions, recognizing the prevalence and importance of RSA signatures, are applicable to the RSA digital signature scheme. The previously proposed schemes discussed in Subsection 3.2.2 are all capable of constructing fairness primitives for RSA signatures, and therefore a direct comparison of those schemes is possible if we limit our comparison to just RSA-based solutions. In Table 3.1, we compare our RSA-based gradational signature approach to the verifiable escrow scheme [AsS00], the verifiable encryption scheme [Ate99], the convertible undeniable signature (CUS) scheme [BoF98], and the off-line coupons approach [AsS00].

In the comparison, we exclude the overhead related to the CA's certificate, and make the following assumptions: (i) a 1200-bit RSA modulus and a 128-bit hash function is used; (ii) for generating the ticket (of the CUS scheme) and the voucher V_C, a signature algorithm with the message recovery feature, whose bandwidth efficiency is 1/2, is employed; (iii) the public key included in an off-line coupon is no longer than 80 bytes; (iv) the partial public key included in the voucher V_C is no longer than 80 bytes; (v) simultaneous exponentiations of the form $b_1^{a_1} b_2^{a_2}$ are counted as

Table 3.1
Comparison of fairness primitive constructions for RSA signatures.

schemes	number of exponentiations	fairness primitive size (bytes)	registration protocol	intermittent contact
verifiable escrow	75	8000	no	no
verifiable encryption	7.5	400	yes	no
CUS	5.3	916	yes	no
off-line coupons	5	1010	yes	yes
gradational signatures	**3**	**310**	**yes**	**no**

1.17 exponentiations (see p.618, [MeV96]); (vi) simultaneous exponentiations of the form $b_1^{a_1} b_2^{a_2} b_3^{a_3}$ are counted as 1.25 exponentiations.

For fair-exchange protocols with gradational signatures, intermittent contact with the TTP is not needed for RSA and GQ-based gradational signatures, but is needed for Meta-ElGamal and Schnorr-based gradational signatures. Note that the numbers of Table 3.1 are approximate figures—the numbers can vary depending on the implementation. The numbers for the verifiable escrow and verifiable encryption schemes are taken from [Ate99].

3.7 Security Considerations

In this section, we discuss the security issues of the fair-exchange protocol when RSA-based gradational signatures are employed. In the first subsection, we investigate the security issues involved in splitting the RSA private key. In the proceeding subsections, we present a security analysis of our fair-exchange protocol.

3.7.1 Insecurity of Splitting the RSA Private Key Multiplicatively

By splitting the private key d, collaborative computation of RSA signatures can be done efficiently. For use in multisignatures (e.g., [Boy89]), one can split the private key additively or multiplicatively. However, in our RSA gradational signature scheme, one must split the full private key d *additively*. We show that splitting d multiplicatively renders our fair-exchange protocol insecure. Specifically, if d is split multiplicatively, then the cosigner can create full signatures on his own without the help of the primary signer. That is, the cosigner is able to use the three keys available to him—the partial private key d_2, the partial public key e_1, and the full public key e—to compute d_1. Although the cosigner is a trusted entity, it is important that he does not have the ability to forge full signatures so that the full signature's non-repudiation property is preserved. If we split d multiplicatively, then the keys satisfy the following relations:

$$d \equiv d_1 d_2 \pmod{\lambda}, \tag{3.5}$$

$$ed \equiv 1 \pmod{\lambda}, \tag{3.6}$$

$$e_1 d_1 \equiv 1 \pmod{\lambda}, \tag{3.7}$$

where $\lambda = p'q'$. Using (3.5), we can insert $d_1 \cdot d_2$ in place of d in (3.6) to obtain

$$ed_1 d_2 \equiv 1 \pmod{\lambda}.$$

Now, multiplying both sides of the above congruence relation with e_1 and using (3.7), we obtain

$$ed_2 \equiv e_1 \pmod{\lambda}. \tag{3.8}$$

The above congruence relation implies that $ed_2 - e_1$ is a multiple of λ.

Recall that the cosigner is given the values d_2, e_1, and e by the primary signer. This means that the cosigner can readily calculate the value of $ed_2 - e_1$, which is a multiple of λ. With this knowledge, the cosigner can factor N efficiently using Koblitz's probabilistic algorithm (see p. 91 of [Kob87]). For completeness, we describe how the cosigner can employ Koblitz's algorithm to factor N in the following paragraph.

We use the same notation that was used in Subsection 3.4.1. Suppose that the RSA modulus N is a product of two unknown safe primes p and q such that $p = 2p'+1$ and $q = 2q' + 1$ with p' and q' primes. Therefore,

$$\begin{aligned} \phi(N) &= (p-1)(q-1) \\ &= 4p'q' \\ &= 4\lambda, \end{aligned}$$

where $\lambda = p'q'$. Because of the congruence relation of Equation (3.8), the cosigner can readily generate an integer, say m', that is guaranteed to be a multiple of λ, that is, $m' = k\lambda$ for some integer k. Let $m = 4m'$. Then, m is guaranteed to be a multiple of $\phi(N)$. Euler's theorem states that for any arbitrary integer $n \geq 2$, the congruence relation $a^{\phi(n)} \equiv 1 \pmod{n}$ holds for any $a \in \mathbb{Z}_n^*$. Therefore, by Euler's theorem, the following congruence relation holds for all integers $a \in \mathbb{Z}_N^*$:

$$a^m \equiv 1 \pmod{N}.$$

Notice that m must be an even integer. Now, the cosigner runs the following `while` loop:

`while`$(a^m \equiv 1 \pmod{N}$ for all $a \in \mathbb{Z}_N^*)$
{
    $m \leftarrow m/2;$
}

One might think that determining whether the statement inside the parenthesis is true (or not) would be difficult or time consuming. It turns out to be quite simple to do. If $a^m \equiv 1 \pmod{N}$ is *not* true for all $a \in \mathbb{Z}_N^*$, then the congruence relation $a^m \equiv 1 \pmod{N}$ does *not* hold for at least half of the a's (i.e., elements) in $\mathbb{Z}_N^*$. Thus, if the cosigner randomly selects i number of a's, and finds that in all cases $a^m \equiv 1 \pmod{N}$, then the probability that the parenthetical statement of the `while` loop is true would be at least $1 - (1/2)^i$. This means that one would only need

to test several dozen randomly chosen a's to determine (with very high probability) that the parenthetical statement is true. After completion of the `while` loop, m is characterized by the following two possibilities:

- m is a multiple of one of the two integers $p-1$ or $q-1$ (say, $p-1$) but not both. In this case, $a^m \equiv 1 \pmod{p}$ is true for all $a \in \mathbb{Z}_N^*$, but for half of the a's in $\mathbb{Z}_N^*$, it is congruent to -1 modulo q.

- m is not a multiple of either $p-1$ or $q-1$. In this case, for a quarter of the a's in $\mathbb{Z}_N^*$, $a^m \equiv 1 \pmod{N}$, and for another quarter of the a's in $\mathbb{Z}_N^*$, $a^m \equiv -1 \pmod{N}$. For the remaining a's in $\mathbb{Z}_N^*$, a^m is congruent to $+1$ modulo one of the primes, and congruent to -1 modulo the other prime.

Therefore, by randomly selecting $a \in \mathbb{Z}_N^*$ and computing $a^m - 1$, the cosigner will soon find a value of a such that $a^m - 1$ is divisible by one of the two primes (say, p) but not the other. The expected number of trials before such a value of a is obtained is two. Once such an a is obtained, the cosigner can factor N because

$$\gcd(N, a^m - 1) = p.$$

Once N is factored, the cosigner can readily compute d_1 (i.e., the primary signer's partial private key) via the extended Euclidean algorithm, and thus is able to compute full signatures completely on his own.

3.7.2 Security Requirements and Adversarial Models

We present a security analysis of our fair-exchange protocol when RSA gradational signatures are employed. To provide a meaningful security analysis, we need to first define the security requirements and the adversarial models. We define security in terms of three specific requirements: *completeness*, *fairness for the verifier*, and *fairness for the primary signer* (or *unforgeability of full signatures*).

We adopt the definition of completeness that was used by Asokan et al. [AsS00] in their security model. As already discussed in Remark 3.5.6, the framework used in our

fair-exchange protocol does not guarantee *timely termination*, while the framework of [AsS00] does ensure timely termination. Therefore, we use a slightly modified version of their definition to account for this fact. We (informally) define completeness as follows (see Section III, [AsS00] for a formal definition):

- *Completeness*: We have two honest players A and B, a trusted third party T, and an adversary F. We assume that T's public key is known to both exchanging parties and the adversary F. Furthermore, we assume that F cannot remove messages in transit or delay any messages beyond a known upper bound (i.e., assume reliable channels). The adversary, however, can interact with the exchanging parties or T. We define completeness to mean that it is infeasible for F to prevent A and B from successfully exchanging their items.

Intuitively, fairness means that it is infeasible for a dishonest player A to obtain the honest player B's item without letting B obtain A's item too. In our fair-exchange protocol, ensuring fairness for each party—either the primary signer or the verifier—involves different security mechanisms within the protocol, and hence we treat them separately.

- *Fairness for the verifier*: Fairness for the verifier is guaranteed if the cosigner can compute the primary signer's full signature σ using a valid partial private key d_2 and a valid partial signature σ_1. This enables the cosigner to force a fair exchange (in the dispute resolution protocol) even if the primary signer is dishonest. For the cosigner to have such a capability, d_2 and σ_1 must be valid. We define fairness for the verifier to mean that it is infeasible for the primary signer to do either of the following without the other party (i.e., either the cosigner or the verifier) detecting her malicious intentions: (i) generate, and send invalid keys e, e_1, and d_2 to the cosigner; (ii) give an invalid partial signature to the verifier. The first criterion implies that the cosigner is guaranteed of being able to determine whether the keys e, e_1, and d_2 satisfy the congruence relations of Equations (3.2) and (3.3). If the three keys satisfy those relations, then the

cosigner has the ability to generate full signatures using the corresponding partial signatures and d_2, and thus has the capability to resolve potential disputes that might arise between the primary signer and the verifier.

- *Fairness for the primary signer* (*unforgeability of full signatures*): To ensure fairness for the primary signer, the following must be true: (i) it is infeasible for the verifier to generate a full signature σ on some message M using the associated partial signature σ_1 and the (full or partial) public keys, and (ii) it is infeasible for the cosigner to generate σ on some message M using d_2, the public keys, and σ_1 on some message M', where $M \neq M'$. The above criteria imply that the full signature should be unforgeable to ensure fairness for the primary signer. Before discussing the unforgeability of full signatures in greater detail, we need to define our adversarial models. Two adversaries (whose goal is to forge full signatures) with different abilities should be considered: adversary F_1 and adversary F_2. Adversary F_1 has all the abilities of the verifier, that is, he has knowledge of σ_1 on some message M', e, and e_1. Adversary F_2 has all the abilities of the cosigner, that is, he has knowledge of σ_1 on some message M', e, e_1, and d_2. Although, in our paradigm, the cosigner is a trusted entity, he should not have the ability to create full signatures on his own. If the cosigner has such an ability, then the full signatures can no longer guarantee non-repudiation associated with the primary signer. Security against F_1 implies that he cannot generate σ on some message M using the associated partial signature σ_1 and the public keys, while security against F_2 implies that he cannot compute σ on some message M using d_2, e, e_1, and σ_1 on some message M', where $M \neq M'$. Because F_2 is more powerful than F_1 (i.e., F_2 has all the abilities of F_1 and more), security against F_2 implies security against F_1. Therefore, we only consider security against F_2's full signature forgery attacks. Throughout the rest of this chapter, we will denote such an adversary as F. We define fairness for the primary signer as security against forgery under a *chosen-message attack* [GoM88] performed by F in the *random oracle model* [BeR93].

In a random oracle model, an appropriately chosen hash function is treated as a random function[13], and all entities, including the adversary, has access to this random function.

3.7.3 Security Analysis

We show that our fair-exchange protocol (using RSA gradational signatures) meets all of the security requirements discussed in Subsection 3.7.2. The *completeness* of our fair-exchange protocol is immediate from an inspection of the protocol. For the rest of this subsection, we focus on the two *fairness* requirements.

Fairness for the Verifier

Fairness for the verifier is ensured as long as the cosigner has the ability to generate the full signature σ using the associated partial signature σ_1 and the partial private key d_2. This means that it should be infeasible for the primary signer to do either of the following without the other party (i.e., either the cosigner or the verifier) detecting her malicious intentions: (i) generate and send invalid keys e, e_1, and d_2 to the cosigner; or (ii) give an invalid partial signature to the verifier.

First, we show that the first criterion is guaranteed by our registration protocol. Recall that in the fifth and sixth steps of the registration protocol, the primary signer sends to the cosigner the reference signature Ω along with a zero-knowledge proof claiming that Ω is a power of ω. If the zero-knowledge proof system is sound, then we can assume that

$$\Omega = \omega^{\delta} \bmod N,$$

where δ is some arbitrary exponent, and $\omega \in Q_N$. If Ω does not satisfy the above relation, then this is readily detected by the cosigner (because the zero-knowledge

[13]See p. 190 of [MeV96] for the definition of a random function.

proof system is assumed to be sound). In the seventh step, the cosigner checks the construction of Ω by verifying the following congruence relations:

$$\begin{aligned} (\omega^{\delta})^{e_1} &\equiv \omega \,(\mathrm{mod}\, N), \\ (\omega^{\delta})^{e} \cdot \omega^{d_2 e} &\equiv \omega \,(\mathrm{mod}\, N). \end{aligned}$$

Note that we have replaced Ω with $\omega^{\delta} \bmod N$. Because ω is a generator of Q_N (see Corollary 3.4.1), the above congruence relations indicate that the following congruence relations must hold:

$$\begin{aligned} \delta e_1 &\equiv 1 \,(\mathrm{mod}\, \lambda), \\ (\delta + d_2) e &\equiv 1 \,(\mathrm{mod}\, \lambda). \end{aligned}$$

Therefore, the keys e, e_1, and d_2 satisfy the congruence relations of Equations (3.2) and (3.3), and are considered to be valid.

Now, we show that the second criterion is guaranteed by our registration and exchange protocols. In the registration protocol, after the cosigner has checked the validity of the keys, he issues a voucher by signing on the partial public key e_1. This key e_1 is used by the verifier to verify the partial signature via the conventional RSA signature verification algorithm. If we assume that the RSA verification algorithm and the signature algorithm used to compute the voucher are secure, then it is infeasible for the primary signer to give the verifier an invalid partial signature without the verifier detecting this fact.

We have shown that our fair-exchange protocol guarantees the two criteria discussed previously. Fairness for the verifier is ensured as long as the cosigner has the ability to generate the full signature σ using the associated partial signature σ_1 and the partial private key d_2. It is trivial to show that the cosigner has this capability if the two criteria are satisfied.

Fairness for the Primary Signer

Fairness for the primary signer is ensured if our RSA gradational signature paradigm is secure against forgery under a chosen-message attack performed by F in the random oracle model. First, we discuss the attack scenario in the conventional RSA single-signer signature model, and then discuss the corresponding attack scenario in our RSA gradational signature paradigm. In the conventional RSA signature model, a chosen-message attack is performed by an adversary F_s who is given the modulus N, public key e, and has *oracle access* to the signature oracle (i.e., the RSA signature generation algorithm) Sig_s. By oracle access to Sig_s, we mean that the adversary is allowed to obtain valid signatures (from Sig_s) for a list of chosen messages before attempting to forge a signature. The public and private keys are assumed to be generated using a standard RSA key generator with a security parameter input k. In addition, if security is considered within the random oracle model, F_s also has oracle access to a random oracle H, which is a random function. In practice, a hash function (which is assumed to be a random function) is used as the random oracle H. The adversary is considered to have carried out a *successful attack* if he outputs a message M and a signature σ such that $\sigma^e \bmod N = H(M)$, but M was not one of the messages given to Sig_s as a signature-oracle query. The adversary's probability of success is denoted as $Success_{F_s}(k)$. The conventional signature scheme is considered to be secure against forgery under a chosen-message attack if the value of $Success_{F_s}(k)$ is negligible[14] for every polynomial-time adversary F_s.

In the forgery attack scenario of our gradational signature paradigm, the adversary F, who has all the abilities of the cosigner, carries out a chosen-message attack in the random oracle model. The adversary F is initialized with the modulus N, joint public key e, partial public key e_1, and the partial private key d_2. In addition, F has oracle access to the partial signature generator Sig_p and oracle access to a random oracle H. Here, Sig_p is the signature oracle. Note that oracle access to Sig (i.e., the full

[14]The probability of an event E is considered to be negligible if for any $c > 0$, there exists a (sufficiently large) security parameter k such that $\Pr(E) < k^{-c}$.

signature generation algorithm) is not needed because F can obtain full signatures for a list of chosen messages by using d_2 and Sig_p. All the keys are generated by the gradational signature key generator KG with a security parameter input k. The adversary F carries out a *successful attack* if he outputs a message M and a full signature σ such that $\sigma^e \bmod N = H(M)$, but M was not one of the messages queried to Sig_p. The adversary's probability of success is denoted as $Success_F(k)$. Our RSA gradational signature paradigm is secure against forgery under a chosen-message attack if the value of $Success_F(k)$ is negligible for every polynomial-time adversary F.

Before we present a formal statement of the full signature's unforgeability, we need to introduce an RSA-related computational problem that we call the Additive Split Key RSA Single Target Inversion (ASK-RSA-STI) problem. This problem is an extension of the RSA Single Target Inversion (RSA-STI) problem defined in [BeS01], and can be defined in a similar manner.

Definition 3.7.1 *ASK-RSA-STI*. Let KG_0 denote the key generation algorithm of a conventional RSA signature scheme with a slight variation—the modulus N is a product of safe primes p and q. The adversarial algorithm is represented by F_0. The ASK-RSA-STI problem is solved if the adversary can produce a value of ζ such that the following experiment returns a value of one. Here, the notation $\pi \leftarrow_R \Pi$ represents the process of randomly selecting the element π from the set Π according to the uniform distribution.

$\mathbf{Exp}^{\mathrm{ASK-RSA-STI}}_{KG_0,F_0}(k)$

$(N, e, d, p, q) \leftarrow_R \{KG_0(k)\}$;
$e_1 \leftarrow_R \mathbb{Z}^*_\lambda$; $d_1 \leftarrow e_1^{-1} \bmod \lambda$; $d_2 \leftarrow d - d_1 \bmod \lambda$;
$z \leftarrow_R Q_N$; $\zeta \leftarrow F_0(k, N, e, e_1, d_2, z)$;
`if` $\zeta^e \equiv z \pmod N$ return 1;
`else` return 0;

Informally, the ASK-RSA-STI problem is to find $z^d \bmod N$, given the values k, N, e, e_1, d_2, and z. A formal statement of the unforgeability of full signatures (i.e., fairness for the primary signer) is given by the following proposition. In the proof, we utilize the proof techniques used in [BeS01, MaR01a].

Proposition 3.7.1 *Let KG_0 denote the key generation algorithm of the ASK-RSA-STI problem. If solving the ASK-RSA-STI problem with respect to KG_0 is hard*[15] *in the random oracle model, then the RSA gradational signature scheme is secure against forgery under chosen-message attacks in the random oracle model.*

Proof We prove the contraposition, that is, given F who can successfully carry out a chosen-message attack against the RSA gradational signature scheme with non-negligible probability, we show that there exists an adversary F_0 who can solve the ASK-RSA-STI problem with nonnegligible probability. The adversary F_0's strategy is to run F as a subroutine to produce the solution to the ASK-RSA-STI problem (i.e., $z^d \bmod N$). Recall that F has the abilities of the cosigner, which means that it is initialized with k, N, e, e_1, and d_2. In addition, F is given access to the signature oracle Sig_p (i.e., the partial signature generator) and the random oracle H. Because F is a subroutine of F_0, adversary F_0 must provide all of F's required inputs (i.e., the initialization values and the answers to signature-oracle and random-oracle queries).

The values k, N, e, e_1, and d_2 are the same values that F_0 is initialized with in the ASK-RSA-STI problem, and hence these values can be directly forwarded to F by F_0. The main obstacle is to "simulate" Sig_p and H so that F_0 can answer F's oracle queries. Adversary F_0 must consider the following criteria when simulating the random oracle:

- Answers to the random oracle queries must be given in such a way that enables F_0 to answer the signature-oracle queries made to Sig_p.

[15] A problem is considered hard if any polynomial-time algorithm's probability of solving it is negligible, in the sense defined earlier.

- Adversary F_0 must simulate the random oracle's randomness, that is, F_0 must answer F's random-oracle queries with random values.

- Adversary F_0 must be able to produce $z^d \bmod N$ using F's forgery output.

Adversary F_0 can satisfy the first criterion by answering the random-oracle query on M (i.e., $H(M)$) with the value $r^{e_1} \bmod N$, where $r = (r')^2 \bmod N$ and r' is a randomly selected integer from $\mathbb{Z}_N^*$. This means that F_0 can answer F's query to Sig_p with the value r because $H(M)^{d_1} \equiv (r^{e_1})^{d_1} \equiv r \pmod N$. Note that this also satisfies the second criterion. Recall that F_0's goal is to produce a solution to the ASK-RSA-STI problem, and this cannot be achieved with the method that was just described. A naive method of satisfying the third criterion is to always reply with a value of z when F makes a random-oracle query on message M (i.e., $H(M)$). Now, if F successfully produces a full signature forgery σ, then F_0 can use σ as his solution to the ASK-RSA-STI problem because $\sigma \equiv z^d \pmod N$. This method, however, does not satisfy the first and second criteria. If F_0 answers the random-oracle query with z, then he is unable to answer the signature-oracle query because he does not have knowledge of d_1, and furthermore this does not simulate the random oracle's randomness. To satisfy all the criteria, a combination of the two techniques is applied: adversary F_0 "guesses" which random-oracle query made by F will be used eventually to produce the full signature forgery. For the guessed random-oracle query, z is returned, otherwise $r^{e_1} \bmod N$ is returned. Using the strategy outlined above, F_0 runs the following algorithm using F as a subroutine. Adversary F_0 is initialized with the inputs k, N, e, e_1, d_2, and z. Let $\rho(\cdot)$ denote a polynomially-bounded function such that the number of signature-oracle plus random-oracle queries made by F is strictly upper bounded by $\rho(k)$. Adversary F_0 starts the algorithm with the following initializations:

set counter $i \leftarrow 0$
initialize F with inputs k, N, e, e_1, and d_2
select l at random from $\{1, \ldots, \rho(k)\}$

Adversary F_0 is now ready to run F. During execution, F makes multiple queries to the random oracle H. Adversary F_0 can answer a random-oracle query on M by running the following algorithm, and returning the output to F.

HSimulator(M)
`if` (there exists a $j \leq i$ such that $M_j = M$)
    return (H_j, j);
`else`
    $i \leftarrow i + 1$;
    $M_i \leftarrow M$;
    `if` $(i = l)$
        $H_i \leftarrow z$;
    `else`
        $r' \leftarrow_R \mathbb{Z}_N^*$;
        $r_i \leftarrow (r')^2 \bmod N$;
        $H_i \leftarrow r_i^{e_1} \bmod N$;
    return (H_i, i);

Adversary F also makes queries to Sig_p. A signature-oracle query (for M) can be answered by F_0 using the following algorithm's output.

SigSimulator(M)
$(H, j) \leftarrow$ HSimulator(M);
`if` $(j = l)$
    STOP;
`else`
    return r_j;

In SigSimulator(M), the STOP instruction means that F_0 stops running the algorithm with no output. If the STOP instruction is executed, F_0 fails to produce a solution for the ASK-RSA-STI problem. If STOP is not executed, then eventually F outputs

a message M and its forged full signature σ. In this case, adversary F_0 outputs σ as his solution to the ASK-RSA-STI problem, and halts.

Now, we derive a relation between $Success_{F_0}(k)$ and $Success_F(k)$ considering the following events:

- $\mathbf{E}_1$: $\sigma^e \bmod N = H(M)$, and M was not one of the messages queried to the signature oracle.
- $\mathbf{E}_2$: adversary F_0 answers to all signature-oracle queries.
- $\mathbf{E}_3$: $M_l = M$.

Adversary F_0 has to answer all of F's signature-oracle queries (i.e., the STOP instruction is never executed) to successfully simulate Sig_p. If Sig_p is successfully simulated, then F_0's answers to the signature-oracle queries are distributed identically to the actual responses that F would obtain from oracle access to Sig_p in the experiment measuring $Success_F(k)$. Therefore,

$$Success_F(k) = \Pr\{\mathbf{E}_1|\mathbf{E}_2\}. \tag{3.9}$$

Using (3.9) and the events defined previously, we derive a relation between $Success_{F_0}(k)$ and $Success_F(k)$.

$$\begin{aligned} Success_{F_0}(k) &\geq \Pr\{\mathbf{E}_1 \wedge \mathbf{E}_3\} \\ &> \Pr\{\mathbf{E}_1\}\frac{1}{\rho(k)} \\ &> \Pr\{\mathbf{E}_1 \wedge \mathbf{E}_2\}\frac{1}{\rho(k)} \\ &= \Pr\{\mathbf{E}_1|\mathbf{E}_2\}\Pr\{\mathbf{E}_2\}\frac{1}{\rho(k)} \\ &> Success_F(k)\frac{1}{\rho(k)^2}. \end{aligned}$$

The first inequality is true because the intersection of events $\mathbf{E}_1$ and $\mathbf{E}_3$ describe the event in which F outputs a forgery that can be used by F_0 as his solution to the ASK-RSA-STI problem. The last inequality indicates that if F succeeds with non-negligible probability, then F_0 succeeds with non-negligible probability. ■

3.8 Conclusions

We presented an efficient method of constructing optimistic fair-exchange protocols. In general, the most costly part of an optimistic protocol is the generation of the *fairness primitive*. To simplify the exchange protocol and increase its efficiency, we used a novel signature paradigm—the *gradational signature* scheme—to construct the fairness primitive. Unlike the vast majority of previously proposed protocols, our approach does not use any zero-knowledge proof systems in the exchange protocol, and thus avoids most of the costly computations. Use of zero-knowledge proofs is needed only in the protocol setup phase (i.e., the registration protocol). The registration protocol needs to be executed only once, after which it can support any number of exchanges. The resulting protocol is extremely efficient. In fact, it requires even less computation and space overhead than the most efficient optimistic protocols known to date. Our approach has other advantageous features: (i) unlike convertible undeniable signatures, gradational signatures are completely compatible with the underlying conventional signature scheme; (ii) the gradational signature paradigm can be applied to most conventional signature algorithms; (iii) our approach is flexible enough to be used with the protocol framework of Bao et al. [BaD98] as well as the framework of Asokan et al. [AsS98, AsS00]. The first feature would facilitate the integration of the fair-exchange feature with existing e-commerce systems without introducing unnecessary complexities.

4. AN APPLICATION OF FAIR-EXCHANGE PROTOCOLS: CERTIFIED E-MAIL

One of the practical applications of fair-exchange protocols is certified e-mail. In a certified e-mail protocol, a receiver is given access to the e-mail's contents if and only if the sender receives an irrefutable proof-of-receipt. Recall that in Chapter 3, we proposed a novel signature paradigm—the gradational signature scheme—that can be used to construct very efficient optimistic fair-exchange protocols. In this chapter, we apply our gradational signature paradigm to construct a "lightweight" certified e-mail protocol. This protocol employs RSA gradational signatures to minimize the computation and communication burden of the sender and receiver. Because of its efficiency, the proposed certified e-mail protocol is ideal for the wireless mobile setting, that is, exchanging e-mails using cellular phones or handheld personal digital assistants (PDAs).

4.1 Introduction

E-mail is fast replacing postal mail as the preferred method of correspondence. However, specialized services similar to those offered by postal mail need to be supported by e-mail before the latter can gain wider acceptance. One such service is certified e-mail. Here, a receiver gets to access the e-mail's contents if and only if the sender receives an irrefutable proof-of-receipt. We propose a novel certified e-mail protocol that uses an off-line trusted third party (TTP). Low computation overhead and minimal participation of the TTP make our scheme particularly suitable for mobile environments, where the communication devices have limited computation and storage abilities.

The problem of certified e-mail belongs to the more general class of problems known as the fair-exchange problem [Aso98, RaR02]; the goal is to ensure the fair exchange of an e-mail message and its corresponding signed receipt. Additionally, the mobile environment imposes considerable restrictions in terms of computation, storage, and communication overhead. As a result, existing fair-exchange protocols cannot be adapted easily for certified e-mail. Recall that fair-exchange protocols can be classified into three categories: (i) gradual-exchange protocols, (ii) protocols using an on-line TTP, and (iii) protocols requiring an off-line TTP. Gradual exchange protocols are impractical for certified e-mail delivery in mobile environments because they require considerable communication overhead. Protocols using an on-line TTP require the TTP to be available for the entire lifetime of the exchange. Thus, although such protocols are efficient in terms of computation, they are not exactly suitable for the wireless mobile setting. The third class of protocols, also called *optimistic* protocols, seems more suitable. Recall that in optimistic protocols, the TTP is involved only if one of the parties behaves unfairly or aborts prematurely. Such situations are more the exception than the rule, and thus the TTP's involvement is minimal.

To the best of our knowledge, the scheme of Ateniese and Nita-Rotaru [AtN02] is at the vanguard of optimistic certified e-mail protocols—it offers several advantages over previous optimistic protocols. However, the protocol uses *verifiable encryption* to encrypt the receipt, which makes it more computationally intensive than on-line TTP protocols, and hence it might be impractical for the mobile setting. We present an optimistic certified e-mail protocol that is more computationally efficient. We employ RSA-based gradational signatures (see Subsection 3.4.1) to distribute the computation of the receipt between the receiver and the TTP. This makes our scheme comparable, in terms of computation and communication overhead, to on-line TTP protocols.

In the next section, we discuss related work. We discuss properties of certified e-mail protocols that are desirable for the mobile environment in Section 4.3. We present our certified e-mail protocol in Section 4.4, and compare its features with

other protocols in Section 4.5. The concluding remarks for this chapter are given in Section 4.6. Portions of material from this chapter will appear in [PaR03].

4.2 Related Work

Because of efficiency, many certified e-mail protocols employ an on-line TTP. Bahreman and Tygar [BaT94] propose a protocol that employs the TTP as a courier of the e-mail message and receipt. In [ZhG96], Zhou and Gollman describe a scheme that delivers the e-mail via a sequence of TTPs. The receipt signed by the recipient is routed back to the originator by the TTPs. In [DeG96], Deng et al. propose a more efficient on-line TTP protocol that requires only four messages. In [AbG02], Abadi et al. discuss a protocol that uses an on-line TTP as a decryption server, that is, the e-mail message is encrypted under a key that is encrypted under the TTP's public key.

In protocols with an on-line TTP, no e-mail transmissions can occur without the TTP, and hence the TTP needs to be highly fault tolerant. Obviously, maintaining such a TTP can be expensive. Moreover, the TTP can become a bottleneck, and pose scalability problems. Some schemes avoid this problem by using optimistic protocols. Certified e-mail protocols such as [AtN02, ScR98] fall into this class. Detailed discussions on optimistic protocols are given in Subsection 3.2.1. In [AtD01], Ateniese et al. discuss a hybrid model, TRICERT, that combines the features of on-line and off-line TTP protocols. Their scheme employs semi-trusted servers, called "postal agents" (PA), to carry out the exchanges, and invokes the fully-trusted TTP only when disputes need to be settled. This approach alleviates the cost of maintaining a fully-trusted on-line TTP. The PA, however, can become a bottleneck, and pose scalability problems.

4.3 Certified E-Mail in Mobile Environments

A certified e-mail protocol should have the following properties (some of which are taken from [AtN02]):

- *Fairness*: No party should be able to corrupt the protocol or abort prematurely to gain an advantage. At the termination of the protocol, either each party gets the other party's item, or neither party does.

- *Timeliness*: Both parties should be able to terminate the exchange within a given finite time. Observe that if the items being exchanged are time-sensitive, then fairness cannot be ensured without the timeliness property.

- *Confidentiality*: Only the intended receiver (and not even the TTP) should have access to the contents of the e-mail.

- *Minimal participation of the TTP*: The TTP's computation and storage requirements per exchange should be kept to a minimum.

The fairness and timeliness properties are basic requirements, while the remaining properties are optional requirements that might be desired in certain cases.

In mobile environments, additional properties are desirable because of the computation and storage limitations of the mobile devices. This is especially true if tamper-proof hardware such as smart cards is used to handle cryptographic computations. A typical smart card has a storage capacity limited to a few kilobytes, and employs a 32-bit RISC microprocessor operating at 15-25 MHz. We concentrate on the case where the receiver is a mobile device—such devices are far more likely to be used for receiving certified e-mail rather than sending them. The following additional properties are desirable for mobile environments:

- *Stateless receiver*: The receiver does not need to store any state information to execute the exchange protocol.

- *Minimal involvement in dispute resolutions*: The receiver's participation in dispute resolutions is minimal.

- *Computational efficiency*: The protocol's computation overhead should not be excessive for the receiver.

- *Minimum number of message exchanges*: The number of required message exchanges should be small.

The first property is desirable because of the mobile device's storage limitation. The rest of the properties, directly or indirectly, determines the receiver's computation load, which affects the device's power consumption and battery life.

4.4 The Certified E-Mail Protocol

In our certified e-mail protocol, Alice is the sender, Bob is the receiver, and Charlie is the TTP. In terms of the gradational signature paradigm, Alice is the verifier, Bob is the primary signer, and Charlie is the cosigner. We assume that the public keys of the certification authority (CA) and the three parties are known to everyone. In the protocol, M denotes an e-mail message, σ denotes a receipt, and σ_1 denotes a partial receipt. The receiver's full signature on the hash of the e-mail message (or an encrypted version of it) is the receipt. The certified e-mail protocol ensures that Bob gets to access the e-mail's contents, M, if and only if Alice receives the corresponding receipt, σ, computed by Bob.

REGISTRATION. The registration protocol needs to be performed only once, after which it can support any number of exchanges. We assume that the registration protocol is performed via private and authenticated channels. In practice, this can be accomplished by encrypting the sensitive information with a session key, and employing Message Authentication Codes (MAC).

1. Bob, using the key generation algorithm, generates the parameters N, p, and q and the keys e, e_1, d, d_1, and d_2. We restrict the modulus N to be a product

of safe primes p and q, that is, p and q are primes such that $p = 2p' + 1$ and $q = 2q' + 1$ with p' and q' primes. Bob then contacts the CA to get e and N certified. At this stage, Bob has to prove to the CA that N is a product of safe primes (without revealing p and q). This can be done by using the zero-knowledge protocol of Camenisch and Michels [CaM99].

2. After verifying the construction of N, the CA issues a certificate C_{CA}, which certifies e and N. We assume that e and N can be extracted from the certificate. The certificate is sent to Bob.

3. Bob sends C_{CA}, e_1, and d_2 to Charlie. Note that d_2 is Charlie's partial private key, and e_1 is the partial public key.

4. Charlie checks the validity of C_{CA}, and extracts e and N from it. He then randomly chooses an integer $\bar{\omega} \in \mathbb{Z}_N^* \setminus \{-1, 1\}$, and checks that $\gcd(\bar{\omega} + 1, N)$ and $\gcd(\bar{\omega} - 1, N)$ are not prime factors of N. He then computes $\omega = \bar{\omega}^2 \bmod N$. Note that ω is a generator of Q_N. Charlie sends the reference message ω to Bob.

5. Bob computes the reference signature

$$\Omega = \omega^{d_1} \bmod N,$$

and sends this to Charlie.

6. Now, Bob proves to Charlie that Ω is a power of ω without revealing d_1. This is done by using the zero-knowledge proof system discussed in Subsection 3.2.3.

7. Charlie checks that Ω is constructed correctly by verifying the following congruence relations:

$$\begin{aligned} \Omega^{e_1} &\equiv \omega \pmod N, \\ \Omega^{e} \cdot \omega^{d_2 e} &\equiv \omega \pmod N. \end{aligned}$$

8. If the verifications (of Steps 4, 6, and 7) are passed, then Charlie accepts Bob's claim that the keys e, e_1, and d_2 are valid. That is, the keys satisfy the following congruence relations:

$$\begin{aligned} e_1 d_1 &\equiv 1 \,(\mathrm{mod}\, \lambda), \\ (d_1 + d_2)e &\equiv 1 \,(\mathrm{mod}\, \lambda). \end{aligned}$$

   If the keys are valid, then Charlie creates a voucher V_C by signing on e_1. We assume that e_1 can be extracted from V_C. Charlie stores his partial private key d_2, and sends V_C to Bob.

EXCHANGE. After Bob has gone through the registration protocol with the CA and Charlie, he can start receiving certified e-mails from Alice.

1. Alice encrypts M with Bob's public key via some asymmetric encryption scheme $AE_B(\cdot)$, and computes its hash value. Note that this public key is different from e. Alice concatenates $H(AE_B(M))$ and a header HD_1, which contains a short description of the e-mail message, and signs this value via some signature algorithm $Sig_A(\cdot)$. It is assumed that the information being signed can be extracted from the signature, either because the signature scheme is capable of message recovery, or because the information is concatenated with the signature. The following value is sent to Bob:

$$s_A = Sig_A(HD_1 \parallel H(AE_B(M)))$$

2. Based on the information in HD_1, Bob decides on whether to accept or reject the e-mail. If he decides to accept it, then he computes the partial receipt

$$\sigma_1 = H(AE_B(M))^{d_1} \bmod N.$$

   He then concatenates HD_2, T, and s_A, where HD_2 is header information and T is a timestamp. This value is signed via some signature scheme $Sig_B(\cdot)$ to produce

$$s_B = Sig_B(HD_2 \parallel T \parallel s_A).$$

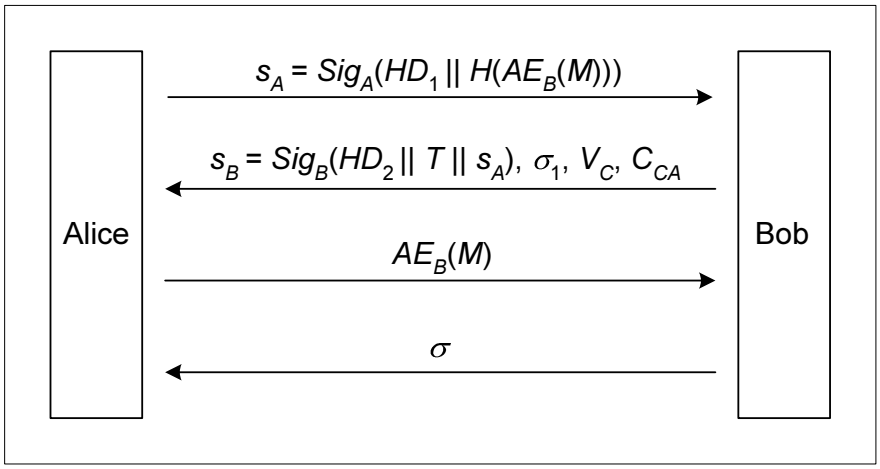

Fig. 4.1. The exchange protocol for certified e-mail.

Bob sends s_B, σ_1, V_C, and C_{CA} to Alice.

3. Using e_1 and N (extracted from V_C and C_{CA}, respectively), Alice verifies the partial receipt σ_1 by checking that

$$\sigma_1^{e_1} \bmod N = H(AE_B(M)).$$

If it is valid, Alice sends $AE_B(M)$ to Bob.

4. Bob decrypts $AE_B(M)$, and reads the e-mail message M. He then computes the receipt

$$\sigma = \sigma_1 \cdot H(AE_B(M))^{d_2} \bmod N,$$

and sends σ to Alice.

The messages exchanged in the exchange protocol are illustrated in Figure 4.1.

Remark 4.4.1 The headers (i.e., HD_1 and HD_2) contain the necessary protocol information such as the exchanging parties' identities and the cryptographic algorithms being used. In HD_1, a short description of the e-mail content should also be included. Using this information, Bob can decide whether to accept or reject the e-mail before committing to it.

Remark 4.4.2 The e-mail message (i.e., M) is encrypted under Bob's public key to ensure confidentiality.

Remark 4.4.3 Note that Bob does not need to store any state information (regarding the transaction) to execute the exchange protocol.

DISPUTE RESOLUTION. If Alice does not receive the receipt, or if the receipt is invalid, she initiates a dispute resolution protocol by contacting Charlie.

1. Alice sends to Charlie s_B, σ_1, V_C, C_{CA}, and $AE_B(M)$.

2. Charlie verifies that

$$\sigma_1^{e_1} \bmod N = H(AE_B(M)).$$

   He also checks to see that the value of $H(AE_B(M))$ matches the corresponding information in s_B (i.e., the value $H(AE_B(M))$ included in s_A, which can be extracted from s_B). He then checks to see whether the time on T has expired—if the time has expired, Charlie stops the protocol, and refuses Alice's request. If everything is in order, Charlie computes the receipt σ using the relation

$$\sigma = \sigma_1 \cdot H(AE_B(M))^{d_2} \bmod N.$$

   Charlie sends σ to Alice, and forwards $AE_B(M)$ to Bob.

Remark 4.4.4 Bob needs to include a timestamp in s_B (see Step 2 of the exchange protocol) to ensure the timeliness property. Consider the following scenario: after Step 2 of the exchange protocol, Alice aborts the protocol. After a long delay, she initiates a dispute resolution protocol with Charlie. Unaware of Alice's malicious intentions, Charlie sends σ to Alice, and forwards $AE_B(M)$ to Bob. If the contents of the e-mail message are time-sensitive, then the forwarded information to Bob is useless. The inclusion of T in s_B prevents such a scenario.

Remark 4.4.5 In Step 2 of the dispute resolution protocol, Charlie needs to forward $AE_B(M)$ to Bob to ensure fairness. This is because Alice might attempt to obtain the

receipt via the dispute resolution protocol after intentionally aborting the exchange protocol at the end of Step 2.

Remark 4.4.6 Note that Bob does not participate in the dispute resolution protocol. He only needs to receive $AE_B(M)$ if it is forwarded to him.

4.5 Comparison with Other Certified E-Mail Protocols

In Table 4.1, we compare our certified e-mail protocol with some of the other protocols discussed in this chapter. The alphanumeric characters inside the square brackets indicate the reference literature that proposed the particular protocol, and "GS" (gradational signatures) denotes our protocol. Protocols with an on-line TTP do not have dispute resolution protocols, and hence the fourth criterion is not applicable to them. Note that for optimistic protocols, the last criterion refers to the number of message exchanges in the exchange protocol (and does not include the messages of the dispute resolution protocol). In on-line TTP protocols, the TTP can become a bottleneck, and pose difficult scalability problems. Moreover, reliable channels between the two parties and the TTP need to be maintained at all times, which might be difficult for wireless mobile environments. Although optimistic protocols avoid these problems, many schemes adopt an on-line TTP because of the considerable computation and communication overhead incurred by optimistic protocols. The optimistic protocol of Asokan et al. [AsS00] requires the parties to perform a cut and choose zero-knowledge protocol, which is quite expensive. In [ScR98], Schneier and Riordan propose an optimistic protocol that requires much less overhead. However, their protocol does not ensure fair exchange, but provide a weaker guarantee: the protocol gathers evidence during execution so that if one party obtains the other's item without sending his, the dishonest party can be prosecuted using the evidence. Ateniese and Nita-Rotaru [AtN02] propose an optimistic protocol that ensures fair exchange while incurring considerably less overhead compared to the scheme of Asokan et al. However, their scheme still incurs noticeably more computation overhead com-

Table 4.1
Comparison of certified e-mail protocols.

	[ZhG96]	[DeG96]	[BaT94]	[ScR98]	[AsS00]	[AtN02]	GS
confidentiality	no	no	no	yes	yes	yes	**yes**
TTP participation	on-line	on-line	on-line	off-line	off-line	off-line	**off-line**
stateless receiver	yes	yes	no	no	no	yes	**yes**
receiver's participation in dispute resolutions	n/a	n/a	n/a	yes	yes	no	**no**
receiver's computation load	light	light	light	light	heavy	medium	**light**
number of message exchanges	6	4	6	4	8	4	**4**

pared to on-line TTP protocols. We avoid such costs by performing most of the costly computations in the one-time registration phase. As a result, our exchange protocol is relatively simple, and requires noticeably less overhead.

In the mobile setting, where the receiver is a mobile device (e.g., a cellular phone), the last four criteria in Table 4.1 are especially important. With the advancement of microprocessor/memory technology, today's mobile devices are not as constrained for processing power or storage capacity as before. However, they will still benefit from an efficient "lightweight" protocol because an efficient protocol directly translates to less power consumption and prolonged battery life.

4.6 Conclusions

In this chapter, we presented a novel approach for constructing certified e-mail protocols. Despite the advantages of optimistic protocols, many protocols adopt an

on-line TTP because of the considerable computation and communication overhead of optimistic protocols. We solved this problem by performing most of the costly computations in the one-time registration phase instead of in the actual exchange phase. This is possible because of the intrinsic fairness properties of the gradational signature paradigm. In fact, our certified e-mail exchange protocol requires very little computation and communication overhead beyond what is typically required in on-line TTP protocols. Such features are particularly suitable for mobile devices with limited computation and storage capabilities.

5. CONCLUSIONS

5.1 Thesis Summary

E-commerce provides businesses with many competitive advantages, and provides customers with the convenience and flexibility of being able to purchase just about any merchandise without leaving the confines of their homes. These factors have made the e-commerce arena the ideal place to sell and buy products and services. In the near future, a wider breadth of items and services will be provided via e-commerce applications. The success and profitability of these applications will be largely determined by their "customer-friendly" qualities. Most certainly, such qualities would include convenience and trustworthiness. To gain the confidence and trust of the customers, providing an adequate level of *security* for the e-commerce transactions is of the utmost importance. In this thesis, we investigated two challenging security problems that are crucial to certain types of e-commerce applications. Our focus was on finding secure and efficient solutions—solutions that are appropriate for practical applications. The first problem deals with authenticating packet streams in multicast or broadcast networks, and the second problem deals with the issue of fairness in exchange protocols involving items of intrinsic monetary value (e.g., digital money). The former security problem is crucial for applications such as information broadcasts (e.g., news feeds and weather updates) and multiparty videoconferencing. The latter security problem is important in the successful deployment of applications such as (digital money) payment protocols, electronic contract signing, and certified e-mail delivery.

In Chapter 2, we studied the security problems related to multicast or broadcast streaming applications. For this type of applications, confidentiality and authentication are the security requirements that are needed. Providing confidentiality for

multicast/broadcast streaming applications can be readily handled by employing off-the-shelf symmetric-key cryptosystems such as the Advanced Encryption Standard (AES). For confidentiality, the main concern is the complexity involved in key management (e.g., key distribution, revocation, and group updates), which we do not investigate in this thesis.

Unfortunately, the problem of providing authentication cannot be handled by employing off-the-shelf solutions or techniques. Thus, special-purpose modifications of existing techniques or totally novel approaches are needed. Our approach, called Signature Amortization using IDA (SAIDA), is based on the technique of *signature amortization*, that is, amortizing a single signing operation over multiple packets. Obviously, this decreases the computational burden of the sender and the receiver when generating and verifying digital signatures, respectively. By using a combination of hash values (computed from collision-resistant hash functions) and digital signatures, we were able to authenticate a block (or group) of packets, consisting of several packets, with only one digital signature. Because we also needed to consider the high-loss characteristics of the multicast/broadcast environment, the Information Dispersal Algorithm (IDA) was used to encode the hash and signature values. Our approach is especially efficient in terms of space (or communication) overhead because just the essential elements needed for authentication (i.e., one hash per packet and one signature per block of packets) are used in conjunction with an erasure code that is space optimal. Note that this use of erasure codes is similar in concept to one of the reliable multicast protocol instantiations that the Reliable Multicast Transport (RMT) working group of the Internet Engineering Task Force (IETF) is standardizing.

According to our simulation results, SAIDA out performs all of the previous (probabilistic) stream authentication schemes. Specifically, it achieves the highest verification rates with the same amount of authentication space overhead, where verification rate is defined as the number of verifiable packets divided by the number of received packets of a stream. In addition to performance evaluations through simulation experiments, we were able to derive meaningful analytical results of SAIDA's

performance in two different bursty loss models. The analyses revealed the following important characteristic of SAIDA: the authentication probability, defined as $\Pr\{P_i \text{ is verifiable} | P_i \text{ is received}\}$ (here, P_i denotes the i-th packet of a block), can be improved, while a constant authentication space overhead is maintained, by increasing the number of packets per block.

SAIDA can be vulnerable to denial-of-service (DoS) attacks, where the attacker inserts bogus packets into the message stream to interfere with the verification process at the receiving end. One way of combating such an attack is to implement additional functionality into the multicast router or the receivers' computers so that the source Internet Protocol (IP) address can be screened against a list of authentic transmitter addresses. This solution can be easily defeated, however, if the attacker is capable of spoofing source IP addresses. In Section 2.6, we provided an extension to our basic scheme that is resistant to such DoS attacks. In the extended scheme, we employed a cryptographic tool called *distributed fingerprints* to make the authentication information (i.e., hash segments and signature segments) resistant to loss as well as modification. This is achieved by using IDA in combination with an error-correcting code (e.g., Reed-Solomon codes). The extended authentication scheme provides protection against the type of DoS attacks discussed above, and is asymptotically space optimal (in the size of the original message).

In Chapter 3, we investigated the problem of ensuring fairness in e-commerce exchange protocols. Very efficient fair-exchange protocols (for e-commerce) can be designed by employing a trusted third party (TTP) to act as an intermediary in carrying out the exchange. These are called fair-exchange protocols with an on-line TTP. These schemes, however, have several drawbacks. Maintaining a highly fault-tolerant TTP that needs to be involved in every exchange can be expensive. Moreover, the TTP can become a bottleneck, and pose scalability problems. One can avoid such shortcomings by employing *optimistic* fair-exchange protocols. In this type of protocols, the TTP is involved in the exchange only if one of the parties behaves maliciously, or if aborts the protocol prematurely. Despite the advantageous

features of optimistic protocols, many e-commerce applications use exchange protocols with an on-line TTP. This is primarily because optimistic protocols entail intricate cryptographic primitives that incur considerable computation and communication overhead.

Although our protocol is an optimistic protocol, it is very efficient in terms of computation and communication overhead. In fact, the exchange procedure requires very little overhead beyond what is typically required for on-line TTP protocols, and to the best of our knowledge, it is one of the most computationally efficient optimistic protocols known to date. The significant improvement in efficiency (over previous protocols) was possible because we employed a novel signature paradigm that we call *gradational signatures*. Note that in exchange protocols for e-commerce, the typical item offered by one of the players (i.e., customer or merchant) is essentially a digital signature or an extension of it (e.g., digital checks). The intrinsic features of the gradational signature paradigm make it possible to devise simple and efficient optimistic fair-exchange protocols.

In Section 3.4, we proposed four gradational signature instantiations based on four well-known conventional signature schemes—RSA, Guillou-Quisquater (GQ), Meta-ElGamal, and Schnorr signature schemes. The security of the first two signature schemes are based on the intractability of factoring large integers, and the security of the other two is based on the intractability of the discrete logarithm problem. We showed that these gradational signature schemes can be employed to construct very efficient and simple optimistic protocols in Section 3.5.

For obvious reasons, security must be one of the first criteria that should be considered when designing fair-exchange protocols. In Section 3.7, we discussed the security issues related to our fair-exchange protocol when RSA-based gradational signatures are employed. We showed that splitting the RSA private key multiplicatively (instead of additively) renders our protocol insecure. In addition, we described the specific security requirements of our protocol, and defined the relevant adversarial models. We also provided proofs of security for our fair-exchange protocol.

As a practical application of optimistic fair-exchange protocols, we presented a "lightweight" certified e-mail protocol in Chapter 4. We discussed the concept of certified e-mail, and examined some of the challenges in designing efficient certified e-mail protocols. Because of the computation and storage limitations of mobile devices (e.g., cellular phones and PDAs), certified e-mail protocols for the wireless mobile environment require special properties. Our certified e-mail protocol satisfies those properties, and hence is well suited for applications in the mobile environment. The efficiency of our protocol was evident when we compared our protocol with six other protocols in Section 4.5.

5.2 Directions for Future Research

In Section 3.7, we only discussed the security issues related to RSA-based gradational signatures. We plan to extend the work of Section 3.7 to include the security analyses of the other gradational signature schemes discussed in the section. The security of RSA and GQ signature schemes are based on the intractability of factoring large integers, while the security of Meta-ElGamal and Schnorr signature schemes are based on the intractability of the discrete logarithm problem. Therefore, the security analyses of the gradational signatures based on the latter two signature schemes will require techniques different from the ones used in Section 3.7.

With growing occurrences of cyber attacks such as denial-of-service (DoS) attacks, increasing the robustness of a system or network against failure has become more crucial than ever. This is also an important issue within the context of e-commerce, especially for service providers or merchants. Even a temporary lapse in service can cost the merchant millions of dollars in terms of lost business. As one of the extensions to our work, we plan to investigate the construction of fault-tolerant archival systems for data depositories. For obvious reasons, merchants handling valuable data such as the customers' transaction information must take extreme care to store the data in a safe manner. Because of the sheer amount of data involved, it might be impractical

simply to keep the same copies of the data in multiple depositories. This can be done more efficiently by encoding the data in a space efficient manner, and dispersing the encoded pieces into multiple depositories to provide fault tolerance.

In this thesis, we only considered e-commerce protocols with a single trusted third party (TTP). This severely limits the fault tolerance of the protocol because it provides a potential attacker with a single point of attack. Because of security and scalability issues, the concept of *distributed* trusted computing entities and TTPs are gaining interest in the research community. Crucial components of the e-commerce infrastructure such as certification authorities (e.g., VeriSign Inc.) and TTPs (of exchange protocols) can be prime targets for malicious attacks. The certification authorities are needed to issue certificates and repudiation lists (for public keys), and TTPs are needed to handle e-commerce transactions. If any of these crucial services are interrupted for a prolonged period, then the entire e-commerce infrastructure fails to function properly. Distributing such entities will greatly increase the fault tolerance of the entire e-commerce infrastructure. However, distributing these entities will almost certainly involve the distributed computation of cryptographic functions. Protocols for distributed cryptographic functions are often very complex and expensive. The investigation of efficient solutions will be an interesting and challenging area for future research.

LIST OF REFERENCES

LIST OF REFERENCES

[AbG02] M. Abadi, N. Glew, B. Horne, and B. Pinkas, "Certified email with a light on-line trusted third party: design and implementation," in *International World Wide Web Conference*, May 2002, pp. 387–395.

[AsS00] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," *IEEE Journal on Selected Areas in Communications*, Vol. 18, No. 4, 2000, pp. 593–610.

[AsS98] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," in *Conference on Advances in Cryptology—EUROCRYPT '98*, 1998, pp. 591–606.

[Aso98] N. Asokan, *Fairness in Electronic Commerce*. PhD thesis, Dept. of Computer Science, University of Waterloo, Ontario, Canada, 1998.

[AtD01] G. Ateniese, B. de Medeiros, and M. Goodrich, "TRICERT: Distributed certified e-mail schemes," in *Network and Distributed System Security Symposium (NDSS)*, Feb 2001.

[AtN02] G. Ateniese and C. Nita-Rotaru, "Stateless-recipient certified e-mail system based on verifiable encryption," in *RSA 2002*, February 2002.

[Ate99] G. Ateniese, "Efficient verifiable encryption (and fair exchange) of digital signatures," in *ACM Conference on Computer and Communications Security*, November 1999, pp. 138–146.

[BaD98] F. Bao, R. Deng, and W. Mao, "Efficient and practical fair exchange protocols with off-line TTP," in *IEEE Symposium on Security and Privacy*, May 1998, pp. 77–85.

[BaT94] A. Bahreman and J. D. Tygar, "Certified electronic mail," in *Symposium on Network and Distributed Systems Security*, February 1994, pp. 3–19.

[Bao98] F. Bao, "An efficient verifiable encryption scheme for encryption of discrete logarithms," in *International Conference on Smart Card Research and Applications (CARDIS '98)*, Sep 1998, pp. 213–220.

[BeR93] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *ACM Conference on Computer and Communications Security*, Nov 1993, pp. 62–73.

[BeS01] M. Bellare and R. Sandhu, *The security of practical two-party RSA signature schemes*, unpublished manuscript, available at http://www.cs.ucsd.edu/users/mihir/papers/splitkey.html, 2001.

[Bla84] R. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley, Reading, MA, 1984.

[Blu83] M. Blum, "How to exchange (secret) keys," *ACM Transactions on Computer Systems*, Vol. 1, No. 2, 1983, pp. 175–193.

[BoC90] J. Boyar, D. Chaum, and I. Damgard, "Convertible undeniable signatures," in *Conference on Advances in Cryptology—CRYPTO '90*, 1990, pp. 189–205.

[BoD01] D. Boneh, G. Durfee, and M. Franklin, "Lower bounds for multicast message authentication," in *Conference on Advances in Cryptology—EUROCRYPT '01*, 2001, pp. 437–452.

[BoF98] C. Boyd and E. Foo, "Off-line fair payment protocols using convertible signatures," in *Conference on Advances in Cryptology—ASIACRYPT '98*, 1998.

[Boy89] C. Boyd, "Digital multisignatures," *Cryptography and Coding*, 1989, pp. 241–246.

[ByL98] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," in *Conference of the ACM's Special Interest Group on Data Communication (SIGCOMM '98)*, 1998, pp. 56–67.

[CaD00] J. Camenisch and I. Damgard, "Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes," in *Conference on Advances in Cryptology—ASIACRYPT '00*, 2000, pp. 331–345.

[CaG99] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," in *IEEE Conference on Computer Communications (INFOCOM '99)*, 1999, pp. 708–716.

[CaM00] J. Camenisch and M. Michels, "Confirmer signature schemes secure against adaptive adversaries," in *Conference on Advances in Cryptology—EUROCRYPT '00*, 2000, pp. 243–258.

[CaM99] J. Camenisch and M. Michels, "Proving in zero-knowledge that a number is the product of two safe primes," in *Conference on Advances in Cryptology—EUROCRYPT '99*, 1999, pp. 106–121.

[ChE87] D. Chaum, J. H. Evertse, and J. van der Graaf, "An improved protocol for demonstrating possession of a discrete logarithm and some generalizations," in *Conference on Advances in Cryptology—EUROCRYPT '87*, 1987, pp. 127–141.

[ChV90] D. Chaum and H. van Antwerpen, "Undeniable signatures," in *Conference on Advances in Cryptology—CRYPTO '90*, 1990, pp. 212–216.

[Che98] L. Chen, "Efficient fair exchange with verifiable confirmation of signatures," in *Conference on Advances in Cryptology—ASIACRYPT '98*, 1998, pp. 286–299.

[CoT95] B. Cox, J. D. Tygar, and M. Sirbu, "Netbill security and transaction protocol," in *1st USENIX Workshop on Electronic Commerce*, July 1995, pp. 77–88.

[Com95] D. E. Comer, *Internetworking with TCP/IP Vol. I: Principles, Protocols, and Architecture*, 3 Edition, Prentice Hall, Englewood Cliffs, NJ, 1995.

[DaP96] I. Damgard and T. Pedersen, "New convertible undeniable signature schemes," in *Conference on Advances in Cryptology—EUROCRYPT '96*, 1996, pp. 372–386.

[DeF89] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *Conference on Advances in Cryptology—CRYPTO '89*, 1989, pp. 307–315.

[DeF92] Y. Desmedt, Y. Frankel, and M. Yung, "Multi-receiver/multi-sender network security: efficient authenticated multicast/feedback," in *IEEE Conference on Computer Communications (INFOCOM '92)*, 1992, pp. 2045–2054.

[DeG96] R. H. Deng, L. Gong, A. A. Lazar, and W. Wang, "Practical protocols for certified e-mail," *Journal of Network and Systems Management*, Vol. 4, No. 3, 1996, pp. 279–297.

[Elg85] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, Vol. 31, No. 4, 1985, pp. 469–472.

[EvG85] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Communications of the ACM*, Vol. 28, No. 6, 1985, pp. 637–647.

[FiS86] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Conference on Advances in Cryptology—CRYPTO '86*, 1986, pp. 186–194.

[FlJ97] S. Floyd, V. Jacobson, C. Liu, S. Mccanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6, 1997, pp. 784–803.

[GeK97] R. Gennaro, H. Krawczyk, and T. Rabin, "RSA-based undeniable signatures," in *Conference on Advances in Cryptology—CRYPTO '97*, 1997, pp. 132–149.

[GeR97] R. Gennaro and P. Rohatgi, "How to sign digital streams," in *Conference on Advances in Cryptology—CRYPTO '97*, 1997, pp. 180–197.

[GoM01] P. Golle and N. Modadugu, "Authenticating streamed data in the presence of random packet loss," in *Network and Distributed System Security Symposium (NDSS '01)*, 2001, pp. 13–22.

[GoM88] S. Goldwasser, S. Micali, and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal of Computing*, Vol. 17, No. 2, 1988, pp. 281–308.

[GuQ88a] L. C. Guillou and J. J. Quisquater, "A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory," in *Conference on Advances in Cryptology—EUROCRYPT '88*, 1988, pp. 123–128.

[GuQ88b] L. C. Guillou and J. J. Quisquater, "A paradoxical identity-based signature scheme resulting from zero-knowledge," in *Conference on Advances in Cryptology—CRYPTO '88*, 1988, pp. 216–231.

[Har94a] L. Harn, "Group-oriented (t, n) threshold digital signature scheme and digital multisignature," *IEE Proceedings—Computers and Digital Techniques*, Vol. 141, No. 5, 1994, pp. 307–313.

[Har94b] L. Harn, "New digital signature scheme based on discrete logarithm," *Electronics Letters*, Vol. 30, No. 5, 1994, pp. 396–398.

[HoM95] P. Horster, M. Michels, and H. Petersen, "Meta-multisignature schemes based on the discrete logarithm problem," in *International Conference on Information Security (IFIP/SEC '95)*, May 1995, pp. 128–142.

[HoP94] P. Horster, H. Petersen, and M. Michels, "Meta-ElGamal signature schemes," in *ACM Conference on Computer and Communications Security*, nov 1994, pp. 96–107.

[ItN83] K. Itakura and K. Nakamura, "A public-key cryptosystem suitable for digital multisignatures," *NEC Research and Development*, Vol. 71, oct 1983, pp. 1–8.

[KoZ96] A. Koifman and S. Zabele, "RAMP: A reliable adaptive multicast protocol," in *IEEE Conference on Computer Communications (INFOCOM '96)*, 1996, pp. 1442–1451.

[Kob87] N. Koblitz, *A Course in Number Theory and Cryptography*, Springer-Verlag, New York, New York, 1987.

[Kra93] H. Krawczyk, "Distributed fingerprints and secure information dispersal," in *ACM Symposium on Principles of Distributed Computing (PODC '93)*, 1993, pp. 207–218.

[LuM97] M. Luby, M. Mitzenmacher, M. Shokrollahi, D. Spielman, and V. Stemann, "Practical loss-resilient codes," in *ACM Symposium on Theory of Computing (STOC '97)*, 1997, pp. 150–159.

[LuV02] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "The use of forward error correction in reliable multicast." IETF Internet Draft draft-ietf-rmt-info-fec-03.txt, 2002. Available at http://www.ietf.org/internet-drafts/draft-ietf-rmt-info-fec-03.txt.

[MaR01a] P. MacKenzie and M. K. Reiter, "Networked cryptographic devices resilient to capture," in *IEEE Symposium on Security and Privacy*, May 2001, pp. 12–25.

[MaR01b] P. MacKenzie and M. K. Reiter, "Two-party generation of DSA signatures," in *Conference on Advances in Cryptology—CRYPTO '01*, 2001, pp. 137–154.

[MeV96] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, Florida, 1996.

[Mer89] R. Merkle, "A certified digital signature," in *Conference on Advances in Cryptology—CRYPTO '89*, 1989, pp. 218–238.

[MiO01] S. Micali, K. Ohta, and L. Reyzin, "Accountable-subgroup multisignatures," in *ACM Conference on Computer and Communications Security*, November 2001, pp. 245–254.

[MiS01] S. Miner and J. Staddon, "Graph-based authentication of digital streams," in *IEEE Symposium on Security and Privacy*, 2001, pp. 232–246.

[MiS97] M. Michels and M. Stadler, "Efficient convertible undeniable signature schemes," in *Annual Workshop on Selected Areas in Cryptography (SAC '97)*, August 1997, pp. 231–243.

[NaS98] D. Naccache and J. Stern, "A new public key cryptosystem based on higher residues," in *ACM Conference on Computer and Communications Security*, 1998, pp. 59–66.

[OhO99] K. Ohta and T. Okamoto, "Multi-signature schemes secure against active insider attacks," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, Vol. E82-A, No. 1, jan 1999, pp. 21–31.

[OkU98] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring," in *Conference on Advances in Cryptology—EUROCRYPT '98*, 1998, pp. 308–318.

[Oka88] T. Okamoto, "A digital multisignature scheme using bijective public-key cryptosystems," *ACM Transactions on Computer Systems*, Vol. 6, No. 8, 1988, pp. 432–441.

[PaC02] J. Park, E. K. P. Chong, and H. J. Siegel, "Efficient multicast packet authentication using signature amortization," in *IEEE Symposium on Security and Privacy*, 2002, pp. 227–240.

[PaC03a] J. Park, E. K. P. Chong, and H. J. Siegel, "Efficient multicast stream authentication using erasure codes," *ACM Transactions on Information and System Security*, Vol. 6, No. 2, 2003, pp. 258–285.

[PaC03b] J. Park, E. K. P. Chong, H. J. Siegel, and I. Ray, "Constructing fair-exchange protocols for e-commerce via distributed computation of RSA signatures," in *22nd Annual ACM Symposium on Principles of Distributed Computing (PODC 2003)*, 2003, pp. 172–181.

[PaM02] A. Pannetrat and R. Molva, "Authenticating real time packet streams and multicasts," in *7th International Symposium on Computers and Communications (ISCC '02)*, 2002, pp. 490–495.

[PaR03] J. Park, I. Ray, E. K. P. Chong, and H. J. Siegel, "A certified e-mail protocol suitable for mobile environments," to appear in *IEEE Global Communications Conference (GLOBECOM)*, 2003.

[PeC00] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," in *IEEE Symposium on Security and Privacy*, 2000, pp. 56–73.

[Per01] A. Perrig, "The BiBa one-time signature and broadcast authentication protocol," in *8th ACM Conference on Computer and Communications Security (CCS '01)*, 2001, pp. 28–37.

[RaR02] I. Ray and I. Ray, "Fair exchange in e-commerce," *ACM SIGecom Exchange*, Vol. 3, No. 2, May 2002, pp. 9–17.

[Rab89] M. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *Journal of the ACM*, Vol. 36, No. 2, 1989, pp. 335–348.

[RiS78] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of the ACM*, Vol. 21, No. 2, 1978, pp. 120–126.

[Roh99] P. Rohatgi, "A compact and fast hybrid signature scheme for multicast packet authentication," in *6th ACM Conference on Computer and Communications Security (CCS '99)*, 1999, pp. 93–100.

[Ros96] S. Ross, *Stochastic Processes*, 2 Edition, John Wiley and Sons Inc., New York, NY, 1996.

[ScR98] B. Schneier and J. Riordan, "A certified e-mail protocol," in *Annual Computer Security Applications Conference*, Dec 1998, pp. 100–106.

[Sch91] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, Vol. 4, 1991, pp. 161–174.

[Sim84] G. J. Simmons, "Authentication theory/coding theory," in *Conference on Advances in Cryptology—CRYPTO '84*, 1984, pp. 411–431.

[WeW01] H. Weatherspoon, C. Wells, P. Eaton, B. Zaho, and J. Kubiatowicz, "Silverback: A global-scale archival system," Computer Science Division, University of California, Berkeley, 2001.

[WoL98] C. Wong and S. Lam, "Digital signatures for flows and multicasts," Department of Computer Sciences, University of Texas at Austin, 1998.

[YaK96] M. Yajnik, J. Kurose, and D. Towsley, "Packet loss correlation in the mbone multicast network," in *IEEE Global Internet Conference*, 1996.

[YaM99] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modeling of the temporal dependence in packet loss," in *IEEE Conference on Computer Communications (INFOCOM '99)*, 1999, pp. 345–352.

[ZhG96] J. Zhou and D. Gollmann, "A fair non-repudiation protocol," in *IEEE Symposium on Security and Privacy*, May 1996, pp. 55–61.

VITA

VITA

Jung Min Park was born in Seoul, Republic of Korea on March 6, 1972. He received his Bachelor's degree and Master's degree both in Electronic Engineering from Yonsei University, Seoul, Republic of Korea, in 1995 and 1997, respectively. From 1997 to 1998, he was a cellular systems engineer at Motorola Korea, Inc. Since then he has been working towards his Ph.D. degree in the School of Electrical and Computer Engineering at Purdue University. His primary area of research is network/communications security, which includes secure multicast, applied cryptography, and electronic-commerce protocols. He has also conducted research in networks and wireless communications.

He is a recipient of the 1998 AT&T Leadership Award. This award is supported, in part, through a grant from the AT&T Foundation, and is given annually to 30 selected students in the Asia/Pacific region.

Starting in the fall of 2003, he will join the faculty of the Department of Electrical and Computer Engineering at Virginia Polytechnic Institute and State University (Virginia Tech) as an assistant professor.