2008

EFFICIENT AND SCALABLE NETWORK SECURITY PROTOCOLS BASED ON LFSR SEQUENCES

Saikat Chakrabarti
University of Kentucky, saikat.chakrabarti@siemens.com

ABSTRACT OF DISSERTATION

Saikat Chakrabarti

The Graduate School
University of Kentucky
2008

EFFICIENT AND SCALABLE NETWORK SECURITY PROTOCOLS BASED ON LFSR SEQUENCES

ABSTRACT OF DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Engineering
at the University of Kentucky

By

Saikat Chakrabarti

Lexington, Kentucky
Co-Directors: Dr. Mukesh Singhal
and Dr. Kenneth L. Calvert
Lexington, Kentucky

2008

ABSTRACT OF DISSERTATION

EFFICIENT AND SCALABLE NETWORK SECURITY PROTOCOLS BASED ON LFSR SEQUENCES

The gap between abstract, mathematics-oriented research in cryptography and the engineering approach of designing practical, network security protocols is widening. Network researchers experiment with well-known cryptographic protocols suitable for different network models. On the other hand, researchers inclined toward theory often design cryptographic schemes without considering the practical network constraints. The goal of this dissertation is to address problems in these two challenging areas: building bridges between practical network security protocols and theoretical cryptography. This dissertation presents techniques for building performance sensitive security protocols, using primitives from linear feedback register sequences (LFSR) sequences, for a variety of challenging networking applications. The significant contributions of this thesis are:

1. A common problem faced by large-scale multicast applications, like real-time news feeds, is collecting authenticated feedback from the intended recipients. We design an efficient, scalable, and fault-tolerant technique for combining multiple signed acknowledgments into a single compact one and observe that most signatures (based on the discrete logarithm problem) used in previous protocols do not result in a scalable solution to the problem.

2. We propose a technique to authenticate on-demand source routing protocols in resource-constrained wireless mobile ad-hoc networks. We develop a single-round multisignature that requires no prior cooperation among nodes to construct the multisignature and supports authentication of cached routes.

3. We propose an efficient and scalable aggregate signature, tailored for applications like building efficient certificate chains, authenticating distributed and adaptive content management systems and securing path-vector routing protocols.

4. We observe that blind signatures could form critical building blocks of privacy-preserving accountability systems, where an authority needs to vouch for the legitimacy of a message but the ownership of the message should be kept secret from the authority. We propose an efficient blind signature that can serve as a protocol building block for performance sensitive, accountability systems.

All special forms digital signatures—aggregate, multi-, and blind signatures—proposed in this dissertation are the first to be constructed using LFSR sequences. Our detailed cost analysis shows that for a desired level of security, the proposed signatures outperformed existing protocols in computation cost, number of communication rounds and storage overhead.

Keywords: Authentication, digital signature, signature aggregation, blind signature, LFSR sequences

Saikat Chakrabarti

July 11, 2008

EFFICIENT AND SCALABLE NETWORK SECURITY PROTOCOLS BASED ON LFSR SEQUENCES

By

Saikat Chakrabarti

Dr. Mukesh Singhal

Co-Director of Dissertation

Dr. Kenneth L. Calvert

Co-Director of Dissertation

Dr. Andrew M. Klapper

Director of Graduate Studies

July 11, 2008

RULES FOR THE USE OF DISSERTATIONS

Unpublished dissertations submitted for the Doctor's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgements.

Extensive copying or publication of the dissertation in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.

A library that borrows this dissertation for use by its patrons is expected to secure the signature of each user.

Name	Date

DISSERTATION

Saikat Chakrabarti

The Graduate School
University of Kentucky
2008

EFFICIENT AND SCALABLE NETWORK SECURITY PROTOCOLS BASED ON LFSR SEQUENCES

DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Engineering
at the University of Kentucky

By

Saikat Chakrabarti

Lexington, Kentucky
Co-Directors: Dr. Mukesh Singhal
and Dr. Kenneth L. Calvert
Lexington, Kentucky

2008

DEDICATION

Dedicated to my mother, the late Suravi Chakravarti.

ACKNOWLEDGMENTS

I would like to thank my advisors, Dr. Kenneth Calvert and Dr. Mukesh Singhal, for providing guidance, support, counseling, and technical knowledge throughout my graduate tenure. I would like to thank Dr. Raphael Finkel for serving on my dissertation committee and providing critical and constructive comments on my dissertation. Thanks to Dr. Andrew Klapper: His course in Cryptography immensely helped me build the foundation of the subject. I would also like to thank Dr. Uwe Nagel, Dr. William Dieter, and Dr. Manivannan for serving on my dissertation committee.

I would like to specially acknowledge Santosh Chandrasekhar's help and support throughout my graduate tenure. The brainstorming sessions with Santosh immensely helped me with complex protocol analysis. Santosh, thank you very much for your kindness and patience, and for lending unconditional help whenever I needed it. Thanks to Jody Larsen for providing guidance in the job-hunting phase during the last semester of my Ph.D. tenure. I enjoyed conversing with Venkata Giruka over lunch and coffee breaks. Venkata providing me with an initial understanding of what it takes to withstand the arduous journey of pursuing a Ph.D. Lei Zhu, your excellent culinary skills provided me with the much-needed invigoration during the last two semesters. Thanks to Jennifer Riggs for her constant help in handling official paper-work, and more importantly, for providing support and advice as a friend during the last two years of my Ph.D. tenure.

The Elliott family (Monica, Larry, and Donna) taught me to how achieve a well-rounded, balanced lifestyle during the—often delusional—final stages of writing my dissertation.

I would like to offer special thanks to my dad, Somnath Chakrabarti, for instilling the foundations of science during my early school years and teaching me strict discipline in life.

Mom, without your constant spiritual presence, my very existence loses its meaning and thus, I dedicate all my academic achievements so far, including my Ph.D., solely to you.

Table of Contents

List of Tables

List of Figures

List of Files

1. Dissertation_Chakrabarti_S_08.pdf 13,437 KB

Chapter 1

Introduction

This dissertation presents techniques to use primitives from a recently discovered cryptosystem [63, 44] based on LFSR sequences to build security protocols suitable for deployment in performance-sensitive networking contexts.

The gap between abstract, mathematics-oriented research in cryptography and the engineering approach of designing practical network security protocols is widening. Network researchers experiment with well-known, time-tested, cryptographic protocols suitable for different network models; researchers inclined toward theory often design new cryptographic schemes without considering the practical constraints of specific network scenarios and applications. For example, the cryptographic community has proposed special forms of digital signatures, like aggregate signatures, but these signatures are not aimed to solve a specific networking problem, like scalable collection of multicast feedback in a secure and reliable fashion or securing source routes in a mobile ad-hoc network. The goal of this dissertation is to address problems in the intersection of these two challenging areas; in other words, building bridges, or reaching agreements, between practical network security protocols and theoretical cryptography.

"Security is layered like an onion." [83] Cryptography is the core, the mathematical foundation, the very primitive building block of security. The second layer encapsulating this inner-most mathematical core consists of applied cryptographic protocols. The construction of these protocols is based on certain well accepted mathematical assumptions. The third layer, the first "tangible" one, consists of software implementations of the theoretical protocols residing in the second layer. The outer layers consist of computer systems, networks, human users, organizations, and relationships between networked computer systems and users.

Today's digital world is dependent on connectivity. People depend on computers and communication, ranging from networks for electronic mail to systems that monitor the nation's critical infrastructure; such remarkable connectivity has been made possible only through the Internet. However, the very egalitarian design of the Internet makes it vulnerable to a variety of attacks—the news is full of stories of vulnerabilities in network systems that were exploited by malicious software. Network security is crucial for proper

functioning of such a connected digital world.

Cryptography forms the core of most network security protocols. However, tangible security that can be implemented in the real world transcends the mathematical formulations of cryptography. The process of security involves human users and networked machines and the interactions between them. Moreover, security protocols need to achieve a balance between security and cost to be suitable for commercial use. Increasing the size of cryptographic keys (for encrypting or signing a message) enhances the security of the protocol, but large keys need large storage space and have high computation and communication cost.

Practical networking protocols often involve a large number of users and a security protocol designed to aid such a network service should scale well in that dimension. For example, Yahoo's servers, supporting 40,000 small Web businesses to manage e-commerce transactions, crashed due to an overload of online holiday season shoppers on "Cyber Monday", November 26, 2007. Though such overloading of systems is often induced by malicious users, as in this case, there were no reports of a security breach; the full-scale meltdown was simply due to poor scalability of the underlying protocols responsible for the customer checkout process. Thus, issues of efficiency and scalability are intrinsically related to security and are crucial to the seamless functioning of real-world network protocols.

1.1 Choosing an efficient cryptographic primitive

Many algorithms used in public-key cryptography are based on finite-field arithmetic; examples include the Diffie-Hellman key agreement algorithm [32] and the Digital Signature Algorithm (DSA) [34]. These cryptographic protocols, like the DSS, are based on the discrete logarithm (DL) problem, which can be informally stated as follows: Given a prime q, a generator α of the cyclic group $\mathbb{G}$ of order q and an element $\beta \in \mathbb{G}$, find an index k such that $\beta = \alpha^k$ or determine that there is no such index. It is believed that there is no solution that is substantially more efficient than searching through the space of all q possible values of k. (A formal definition of the discrete logarithm problem/assumption is given in Chapter 2.2.). However, the chosen field sizes q must be sufficiently large to withstand various attacks (algorithms to solve the discrete logarithm problem).

Traditionally, sequences generated by linear feedback shift registers (LFSR) have been used to generate key streams in stream ciphers for encrypting data. Around the year 2000, LFSR sequences made an appearance, in a rather small section of the cryptographic community, as a useful tool in building a new form of public-key cryptosystem (PKC). The LFSR-based public-key cryptosystems [44, 63] use reduced representations of finite

field elements, which enables us to represent finite field elements in the extension field $\mathbb{F}_{q^n}$ (containing q^n elements), by the corresponding minimal polynomials whose coefficients are chosen from the base field $\mathbb{F}_q$ (containing q elements). The security of LFSR-based public-key cryptosystems is based on the difficulty of solving the DL problem in the extension field $\mathbb{F}_{q^n}$. However, all computations, involving sequence elements, needed for the protocol are performed in the base field $\mathbb{F}_q$. This reduced representation of finite field elements leads to substantial savings, both in communication and computational overhead, for a desired security level. For example, 340-bit keys in the XTR (a phonetic acronym for Efficient and Compact Subgroup Trace Representation) PKC [63], based on cubic (third order) LFSR sequences, give security equivalent to 1024-bit keys in cryptosystems using a traditional representation of finite fields. The Digital Signature Algorithm (DSA), an NIST standard [34], is an example of a cryptosystem that uses conventional representation of finite fields. We regard the security obtained using 1024-bit keys in traditional finite-field cryptography as the current standard and thus investigate the feasibility and design of applied cryptographic protocols based on cubic LFSR sequences (in particular the XTR cryptosystem), to solve problems in network security. We provide a discussion on the reduced representation of finite-field elements in XTR in Chapter 2.

1.2 Thesis contributions and structure

An effective solution to a network security problem should achieve an adequate balance between comprehensive security and scalability. In this dissertation, we present techniques for building performance-sensitive authentication protocols using primitives from LFSR sequences, for networking applications involving a large set of users. Our security protocols are designed to work in a variety of challenging networking contexts, described in detail later in this section. In this dissertation, we make the following key contributions:

1. We examine techniques for using LFSR sequences to construct various forms of new digital signature schemes. Informally, digital signatures are electronic counterparts to hand-written signatures: An entity signs an electronic document, and the signature specifies the person responsible for the document. All our proposed forms of signatures—aggregate signatures, multisignatures, and blind signatures—are the first to be constructed using cryptographic primitives from LFSR sequences. The reader is referred to Chapter 2.2 for a description of these special forms of digital signatures.

2. We investigate the feasibility of transforming existing protocols into more efficient ones using LFSR sequences, suitable for use in resource-constrained environments such as mobile ad-hoc networks.

3. We evaluate the scalability and fault-tolerance of the existing protocols in emerging networking applications, such as collecting authenticated feedback in multicast applications, finding secure routes in resource constrained environments, and securing inter-domain routing protocols.

4. We conduct an extensive analysis of the security and performance of the proposed signature schemes. In the security analysis, we examine the most general mathematical assumptions that are conjectured to be hard, like the DL assumption, and aim to deduce relations between such assumptions and the proposed security protocols.

Next, we elaborate the networking contexts and outline the significant research contributions pertaining to these contexts.

1.2.1 Context 1: Authenticating feedback in multicast applications

A common problem faced by large-scale multicast applications, like software distribution, multimedia transmission, and real-time news feeds, is collecting authenticated feedback from the intended recipients. The feedback needs to be authenticated because the source wants to verify that the data was reliably delivered to the intended receivers, even in the presence of an adversary who might send bogus information to the source. The standard unicast-like solution would have receivers send signed acknowledgments (Acks) to the source, which unfortunately leads to what we call a *signed-Ack implosion problem*: The source needs to verify all individual signatures on the Acks that it receives. The challenge is to find an efficient, scalable and fault-tolerant technique for convincing the source that the intended multicast receivers have indeed received the multicast data. Informally, fault-tolerance in this case means that the protocol does not have to restart if some nodes fail to deliver data.

To solve this problem, we design an efficient, scalable and fault-tolerant technique for combining multiple signed Acks into a single compact one. We propose a third-order LFSR-based, single-signer signature scheme using a suitable variant of the ElGamal family of signatures [37, 51]. We then construct an efficient, single-round, tree-structured multisignature scheme, using the single-signer signature scheme. Multisignatures aim to prevent resources (signatures, storage elements, and computation) from growing linearly in the number of signers participating in a network protocol. Our performance analysis shows

that for a desired level of security, the proposed tree-structured multisignature outperforms existing protocols in computation cost, communication rounds and storage overhead [27].

1.2.2 Context 2: Securing routing in ad-hoc networks

Resource constraints of nodes, limited capacity of the wireless medium, node mobility and the cooperative, self-organized form of mobile ad-hoc networks make it difficult to transfer techniques for securing traditional wired networks to the ad-hoc networking environment. For example, delegating special functions to nodes or assuming the existence of a trust infrastructure to distribute certified public keys is not practical in mobile ad-hoc networks. The dynamic source routing protocol (DSR) [57] is perhaps the most popular on-demand source routing protocol designed for multi-hop wireless ad-hoc networks. However, the DSR protocol, in its original design, is vulnerable to several forms of attack by malicious nodes and thus, cannot guarantee authentic source routes.

We focus on the following problem: In DSR, how can a source node wanting to find a route to a destination be assured of the authenticity of the route advertised in a received routing packet? We would like to guarantee this authenticity without imposing substantial overhead on the nodes that help in discovering routes.

Recently, techniques of aggregating signatures (involving multiple communication rounds) have been applied to authenticate source routes in mobile ad-hoc networks. We use LFSR sequences to construct the single-round multisignature scheme [25]. The proposed multisignature is derived from a single-signer signature scheme, and is also a variant of the ElGamal signature family [38, 51]. The multisignature algorithm is engineered to produce an efficient technique for authenticated route discovery in DSR. Our scheme requires no prior cooperation to construct the multisignature and supports authentication of cached routes.

We introduce our idea behind authenticating routes in DSR assuming, for simplicity, that all nodes have access to certified public keys of other nodes in the route. We present solutions using a trusted third party (TTP) to help in distributing certified public keys. We acknowledge that assuming the existence of a TTP is paradigmatically unsuitable for ad-hoc networks. Using the concepts of PGP [101] and previous results of the small-world property [73] exhibited in trust graphs in self-organized systems [89, 90], we relax the assumption of the TTP and formulate policies for a fully distributed framework for individual and aggregate public-key management.

1.2.3 Context 3: Securing path-vector routing protocols

Recently, there is a noticeable movement in the network research community to rebuild networks following a clean-slate design approach—considering design parameters from scratch—that has the potential to enable substantial security improvements over existing network systems. Distributed applications involving a large number of participants, such as multi-player games and replicated databases, represent a large fraction of the traffic carried in the Internet today; all such modern applications are in dire need of well-defined and practical authentication mechanisms.

Aggregate signatures solve a small but crucial subset of the bigger problem of securing the whole Internet: They support scalable authentication of a large number of users. We focus on techniques for building efficient and scalable forms of aggregate signatures, suitable for applications like securing path-vector routing protocols (for example, the inter-domain routing protocol, BGP). The proposed aggregate signature scheme has other potential applications, such as building efficient certificate chains and authenticating distributed and adaptive content management systems.

We present a aggregate signature scheme constructed using LFSR sequences [24]. We believe that the proposed aggregate signature scheme can improve the processing latency as well as reduce space requirements in building secure, large-scale distributed network protocols. Our aggregate signature scheme offers constant-length signatures, fast signing, aggregation and verification operations at each node, and it requires fewer storage elements (public keys needed to verify the signature) than other traditional aggregate signature schemes.

1.2.4 Context 4: Providing accountability in privacy-preserving systems

Accountability, in the context of the modern Internet, refers to reliably identifying an entity that is responsible for sending a network packet. The modern Internet lacks a network-level mechanism offering accountability. Moreover, concerns about **privacy** complicate the design of any accountability service: The ability to trace communications at user level is undesirable and thus hinders (or even prevents) deployment of such accountability mechanisms.

The tension between accountability and privacy can be ameliorated to some extent through the use of **blind signatures** because they allow an authority to vouch for the legitimacy of a message while the ownership of the message can remain hidden from the authority.

Different forms of blind signature constructions exist in the literature and have found valuable use in areas such as E-Cash technology and E-voting schemes. However, conventional blind signatures require intensive computation; a direct application of these traditional signatures faces scalability and performance challenges. We present a cubic LFSR-based, single-signer signature scheme, following a variant of the ElGamal signature family. Using the single-signer scheme, and following fundamentals of a blind signature used in E-Cash systems [21], we present an efficient blind signature built with LFSR sequences [23]. We show that the proposed blind signature scheme can serve as a protocol building block for privacy-preserving accountability systems.

1.2.5 Thesis organization

In Chapter 2, we first discuss some standard cryptographic terminology to give the reader a high-level overview of some cryptographic concepts before going into formal discussions in the later chapters of this dissertation. Then, we discuss the basics of digital signatures and discuss related research on special signature schemes, such as aggregate signatures (generalized and sequential), multisignatures, and blind signatures. The chapter concludes with a discussion of the basic building blocks of cryptography based on LFSR sequences: The mathematics of LFSR sequences and cryptosystems based on LFSR sequences.

Chapter 3 presents our solution to the problem of authenticating feedback in multicast applications. Chapter 4 presents our research that addresses the problem of authenticating source routes in the DSR protocol, used in mobile ad-hoc routing environments. In Chapter 5, we present a novel technique of aggregating signatures, suitable for applications like securing path-vector routing protocols. In Chapter 6, we introduce a novel LFSR-based blind signature scheme. Chapter 7 concludes the dissertation and includes a summary of the significant research contributions and possible research directions in the area of efficient authentication protocols.

In Appendix A, we present a mathematical background on topics that researchers have used to solve network security problems similar to those we address in this dissertation. The appendix provides the reader with pointers to the terminology and underlying cryptographic definitions and mechanisms needed to rigorously compare our results with previous solutions. Topics discussed in the appendix include rudiments of bilinear pairings on elliptic curves and well-known variants of Diffie-Hellman problems and assumptions used in contemporary applied cryptographic protocols.

Chapter 2

Background and related work

This chapter is divided into three parts. The first part discusses some standard cryptographic terminology to give the reader a high-level overview of some cryptographic concepts before going into formal discussions in the later chapters of this dissertation. The second part contains a detailed description of digital signatures and also presents special forms of signatures relevant to the dissertation. The third part contains the mathematical background of LFSR sequences and construction of a cryptosystem using LFSR sequences. The second and third parts include corresponding related research. We assume that the reader has a basic knowledge of abstract algebra.

2.1 Some standard cryptographic terminology

Authentication Authentication (more precisely, entity authentication) is the process of identifying an entity in a reliable manner. It provides a means to verify that an entity is indeed who it claims to be. There are three primary bases for authenticating a human entity: (1) Something you are, as evidenced by biometric devices such as retinal scanners, fingerprint analyzers and voice recognition systems, (2) something you know, as evidenced by passwords, and (3) something you have: smartcards, physical keys.

There exists another concept of authentication, known as **message authentication**: It is defined as the process of reliably identifying the entity that originated a particular message. Message authentication provides implicit message integrity: If the message has been modified, message authentication should fail. In the case of entity authentication, both entities are active in communication, giving a timeliness guarantee [70]; message authentication provides no guarantees of timeliness.

Computationally infeasible In cryptography, we quantify an adversary by the amount of computing power it has. Cryptographic schemes that are breakable in principle are often not breakable in practice because the computing power of the adversary attempting to break the scheme is limited. In standard cryptographic terms, a problem is said to be computationally infeasible if an adversary, having access to a polynomial

time algorithm, cannot succeed in solving that problem in general, although the adversary may succeed in solving some instances.

Negligible function The notion of success with small probability is quantified in protocol analysis. A real-valued function $f(\lambda)$ is **negligible** if for every fixed $d \geq 0$, there exists a sufficiently large integer λ_d such that $f(\lambda) \leq \lambda^{-d}$ for every $\lambda \geq \lambda_d$.

Probabilistic polynomial-time algorithm Broadly defined, an algorithm is said to be probabilistic polynomial time (PPT), if it uses randomness, and its running time is bounded by some polynomial in the size of its input. The output of a PPT algorithm depends on the randomness—that is, the algorithm achieves completion (in polynomial time) with a certain probability derived from its input of randomness. In the analysis of modern cryptographic protocols, we allow the adversary to have access to a PPT algorithm. In this case, a problem is hard for an adversary to solve if the adversary, armed with a PPT algorithm, succeeds in solving the problem with small or negligible probability. Throughout this dissertation, we model our adversary as having access to a PPT algorithm. The reader is referred to lecture notes by Goldwasser et al. [42] for an elaborate and rigorous discussion on the topic.

Standard Model Since we still don't have proofs that any of the standard cryptographic building blocks have computational lower bounds, we make some complexity-theoretic hardness assumptions to achieve common cryptographic goals [11]. Examples of such assumptions are: (1) Factoring the product of large primes is hard, and (2) computing the discrete logarithm is hard in certain sufficiently large groups. A cryptographic protocol is said to be secure under the standard model the following holds true: If an adversary can break the cryptographic protocol, one can solve the underlying mathematical problem conjectured to be hard.

Random-oracle model Very few practical cryptographic protocols can be proved to be secure in the standard model. The random-oracle model is a model, alternative to the standard model, that researchers often refer to construct proofs, when the proofs in the standard model are "unappealing or provably impossible" [11]. The random-oracle model is constructed for cryptographic protocols (such as digital signatures) that use hash functions. The hash function is formalized by an oracle (denoted by H), which produces a random value for every new query. Formally, the oracle H is modeled as follows: (1) Entities including the adversary submit queries x to the oracle and receive $H(x)$. Queries are private: An adversary does not get to see the query x that

an honest party makes. (2) The oracle maintains a list $L = (x_i, y_i)$, where $x_i \in \{0,1\}^l$ is a query and $y_i \in \{0,1\}^n$ is the response. The list L is initially empty. When the oracle receives a query x_k, it searches the list L for the tuple (x_k, y_k). If the oracle finds the tuple, it returns y_k. Otherwise it chooses a random string $y_k \in \{0,1\}^n$, stores the tuple (x_k, y_k) and returns y_k. The random-oracle model is a useful tool for validating cryptographic constructions. However, the random oracle does not exist in the real world and a majority of the cryptographic community looks upon protocols using the random-oracle model for security proofs with a hint of doubt [99]. We have no guarantees of security once the random oracle model is instantiated with real world hash function such as SHA-1 or MD5.

Provably secure Researchers have proposed various models such as the standard model (containing the most general and well-accepted assumptions such as those mentioned above) and the random-oracle model [7] to construct proofs of cryptographic protocols. Provable security implies construction of proofs of security under these models. The reader should not misunderstand provable security to imply an absolute proof of security. The reader may refer to an exemplary discussion on provable security by Koblitz et al. [61].

Reduction A security algorithm A (a digital signature in our case) reduces to another security algorithm B (another digital signature) means an adversary who can forge B can also forge algorithm A. The power endowed to the adversary depends on the model of security, based on which the proof of reduction is constructed. In this dissertation, we give the adversary access to a PPT forger, and deduce a reduction from a well-known signature algorithm (conjectured to be computationally infeasible for an adversary to forge) to our proposed signature algorithm. None of our proposed protocols are, however, provably secure in any known security model such as the random oracle model.

2.2 Digital signatures: basics and special forms

Digital signatures have the same purpose as conventional handwritten signatures, namely message authentication. However, there are three fundamental differences in the construction and use of digital versus handwritten signatures:

- A person attaches a handwritten signature physically to the document being signed. A digital signature is not intrinsically bound to the electronic message; a signing algorithm must "somehow bind the signature to the message" [87].

- A handwritten signature is verified by comparing it with another authentic signature. The person verifying the signature should thus have access to another authentic signature. On the other hand, electronic signatures can be verified by anyone using a publicly known verification algorithm.

- A copy of a handwritten signature is easily distinguished from the original; copies of digital signatures are indistinguishable. To prevent re-use of signatures (often desired with signatures on one-time financial transactions), we usually mix a time-varying quantity with the message before we generate the digital signature on the message.

Formally, the digital signature scheme, `S`, can be structurally represented by the tuple $\langle$`Init`, `KeyGen`, `G`, `V`$\rangle$, whose components are described as follows:

Initialization (`Init`): A probabilistic polynomial-time (PPT) algorithm that outputs the public parameters, params. The nature of the public parameters depends on the underlying cryptosystem.

Key Generation (`KeyGen`): A PPT algorithm that takes public parameters params as input and outputs a private/public key-pair (SK, PK).

Signature Generation (`G`): A PPT signature generation algorithm that takes public parameters params, the private key SK, and a message m as inputs and outputs a signature σ. (The probabilistic nature of the signature generation is discussed in Chapter 3.)

Signature Verification (`V`): A deterministic algorithm that takes the public parameters params, public key PK, a signature σ, and the message m as inputs, and outputs a result *Valid* or *Invalid*.

Our analysis of all proposed signatures has two parts: Correctness and security. In the correctness analysis, we assume the following behavior from all entities participating in the signature scheme: (1) Both the signer and the verifier follows the initialization procedure and agrees in advance on the public parameters. scheme. (2) The signer honestly follows the key generation procedure to generates his private and public keys. (3) The signer honestly

generates his signature on the message. Under the above assumptions, the signature is correct if it passes the verification procedure.

What remains to be shown is: Given a message m, it should be computationally infeasible for any entity, other than the signer, to generate a signature which passes the public verification procedure. This is precisely the issue of forgery, which we address in our security analysis. In our security analysis, we reduce a variant of the ElGamal signature family (discussed next) to the proposed signature scheme.

The proposed protocols in the dissertation are all based on the ElGamal signature family [51]. The ElGamal signature [37] and its variants are based on a mathematical problem, called the discrete logarithm problem (DLP), which is conjectured to be infeasible over particular sets. An example of one such set is $\mathbb{Z}_p^*$, where p and Q are two large primes with $Q \mid (p-1)$. The NIST standard for digital signatures [34] is a variant of an ElGamal signature. In the following section, we describe the ElGamal family of signatures.

2.2.1 Signatures based on the discrete logarithm problem

Formally, the DL problem and assumption are defined as follows:

Definition 2.2.1 (DL problem/assumption)**.** *Let α be a generator of the multiplicative group $(\mathbb{F}_q)^*$, where q is a large prime and $\mathbb{F}_q$ a finite field. The **DL problem** in $\mathbb{F}_q$ is: Given $(q, \alpha \in (\mathbb{F}_q)^*, \beta \in \mathbb{F}_q)$, find an integer $k \geq 0$ such that $\beta = \alpha^k$ or determine that there is no such index.*

*Let $\mathcal{A}$ be a probabilistic polynomial time (PPT) algorithm that solves the **DL problem**. Define the advantage of the DL solver $\mathcal{A}$ as: $\mathsf{Adv}_{\mathcal{A}}^{DL} = Pr[\mathcal{A}(q, \alpha, \beta) = k \mid \alpha \in_R (\mathbb{F}_q)^*, 0 \leq k \leq ord(\alpha), \beta = \alpha^k]$, where the notation $a \in_R A$ means element a is randomly chosen from the set A, $ord(\alpha)$ denotes the order of α in the field $\mathbb{F}_q$. The probability is over the random choices of α, k, and the random bits of $\mathcal{A}$.*

DL Assumption*: The finite field $\mathbb{F}_q$ is said to satisfy the DL Assumption if $\mathsf{Adv}_{\mathcal{A}}^{DL}$ is negligible.*

The ElGamal signature scheme can be constructed as follows. Let entity A be the signer of message m, and entity B be the verifier.

1. Entities A and B agree (in advance) on the public parameters $\langle p, Q, \alpha \rangle$, where elements p and Q are two large primes with $Q \mid (p-1)$, and $\alpha \in \mathbb{Z}_p^*$ a generator of a cyclic subgroup of order Q.

2. Entity A randomly chooses $x \in_R \mathbb{Z}_Q^*$, the long-term private key, and computes $y = \alpha^x$, the corresponding long-term public key.

3. Entity A randomly chooses $k \in_R \mathbb{Z}_Q^*$, the ephemeral private key, and computes $r = \alpha^k$, the corresponding ephemeral public key.

4. Entity A generates the signature on hashed message $h = H(m)$ by solving for the variable t in the following signing equation:

$$h \equiv xr + kt \mod Q \tag{2.1}$$

5. Entity A sends the pair, (t, h), and the ephemeral key, r, to entity B.

6. Entity B verifies the signature by checking the following equivalence:

$$\alpha^h \equiv y^r r^t \mod Q \tag{2.2}$$

The ElGamal signature is correct since:

$$\begin{aligned} \alpha^h &\equiv \alpha^{(xr+kt)} \mod Q \\ &\equiv (\alpha^x)^r (\alpha^k)^t \mod Q \\ &\equiv y^r r^t \mod Q \end{aligned}$$

Horster et al. [51] propose several variations of the signing equation and the corresponding verification equation in the ElGamal signature scheme. The variants of the ElGamal signature are commonly known as the ElGamal family of signatures. The ElGamal signature family includes the popular DSA [34] and the Schnorr variant, commonly used by the cryptographic community to build provably secure protocols [84].

Table 2.1 shows the signing and verification equations of some well-known discrete log-based signature schemes, labeled by the corresponding ElGamal variant. Only EG II.3, the Schnorr variant has been proved secure in the random-oracle model. The signing equation of any variant in the ElGamal family of signatures can be generalized as:

$$\pm A \equiv \pm xB \pm kC \mod Q$$

where A, B and C are permutations of either m, r and t individually, or of functions $f(\cdot,\cdot)$, $g(\cdot,\cdot)$ of (m, r), (m, t) and (r, t). Table 2.2 lists the permutations of the parameters A, B and C, and the corresponding variant in the ElGamal family.

Table 2.1: Popular DL-based signature schemes

EG Variant	**Signing Eqn.**	**Verification Eqn.**	**Usage**
EG I.1	$m \equiv xr + kt$	$\alpha^m \equiv y^r r^t$	NIST Std. DSA [34]
EG II.3	$t \equiv xf(m,r) + k$	$\alpha^t \equiv y^{f(m,r)} r$	Schnorr type [84]
EG II.5	$f(m,r) \equiv xt + k$	$\alpha^{f(m,r)} \equiv y^t r$	Nyberg-Rueppel type 1
EG II.6	$f(m,r) \equiv xt + k$	$\alpha^{f(m,r)} \equiv yr^t$	Nyberg-Rueppel type 2 [76]
EG I.4	$t \equiv xm + kr$	$\alpha^t \equiv y^m r^r$	*Proposed multisignature*
EG I.3	$t \equiv km - xr$	$\alpha^t \equiv y^r r^m$	*Proposed blind signature*

Table 2.2: Permutations of (A, B, C) in the ElGamal signature family [51]

Permutation of $(\pm A, \pm B, \pm C)$	**Variant**
(m, r, t)	EG I.1 - EG I.6
$(f(m,r), t, 1)$	EG II.1 - EG II.6
$(f(m,r), g(m,t), 1)$	EG III.1 - EG III.6
$(f(m,t), g(r,t), 1)$	EG IV.1 - EG IV.6
$(f(m,r), g(r,t), 1)$	EG V.1 - EG V.6

The resulting signature can be verified under public key y by checking the equivalence $\alpha^A \equiv y^B r^C \mod Q$. The functions $f : \{0,1\}^* \times \mathbb{Z}_Q^* \mapsto \mathbb{Z}_Q^*$ and $g : \mathbb{Z}_q^* \times \mathbb{Z}_Q^* \mapsto \mathbb{Z}_Q^*$ are carefully chosen so the signing equation can be solved for the parameter, t, the essence of the signature. For example, functions f and g can be chosen (agreed in advance by the signer and verifier) as the modular multiplication operation (modulo q) or as a cryptographic hash function.

Next, we present brief overviews and related research in special-purpose digital signature schemes, namely, aggregate (generalized and sequential) signatures, multisignatures, and blind signatures.

2.2.2 Generalized aggregate signatures

An aggregate signature scheme allows us to combine n signatures from n different signers on n distinct messages into a single signature [17]. This compact signature, called the **generalized aggregate signature**, provides authentication simultaneously on all the n distinct messages for the n corresponding signers. Any entity (not necessarily one of the signers) can verify the aggregate signature, given the corresponding n public keys of the signers. Aggregate signatures come in different flavors, depending on the nature of the application. For example, multisignatures have all signers sign the same message. Sequential aggregate signatures have signers verify, sign and aggregate signatures sequentially.

Boneh et al. [17] first propose the idea of a generalized aggregate signature scheme. They develop their aggregate signature using groups with efficiently computable bilinear maps [18]. The aggregate signature scheme is provably secure in the random-oracle model [8] under the Bilinear Diffie-Hellman assumption (described in Appendix A.2). This algorithm requires the use of expensive pairing operations for signature verification, which limit its use in resource-constrained computing environments. Another limitation of their aggregation technique is scalability: the number of operations required to verify an aggregate signature increases linearly with the number of signers. Xu et al. [94] propose an identity-based generalized aggregate signature using bilinear pairings that require a constant number of pairing operations for verifying an aggregate signature (independent of the number of signers). However, their scheme suffers from a different problem: the size of the aggregate signature grows linearly with the number of signers.

2.2.3 Sequential aggregate signatures

In a generalized aggregate signature scheme, the process of verifying a signature is independent of the order in which the messages are signed by users. Lysyanskaya et al. [68] introduce the concept of generating aggregate signatures sequentially, by an ordered set of signers. Lysyanskaya et al. instantiate their scheme using RSA [82]. However, their aggregate signature scheme has the following weaknesses: (1) The verifier of the aggregate signature must know the order in which the signatures were created. This knowledge adds significant overhead to the security protocol. (2) The verification cost of the aggregate signature grows linearly with the number of signers.

Lu et al. [67] propose a sequential aggregate signature scheme using bilinear pairings. They construct their scheme based on the signature scheme by Waters et al. [92] and derive a proof of security without using random oracles. However, their scheme has the following weaknesses: (1) The storage elements needed to verify the aggregate signature grows linearly with the number of signers. (2) The public and private keys, derived from the construction of Waters et al., inherit the drawback that the key sizes are extremely large and are not suitable for practical use.

Gentry et al. [40] propose the first id-based sequential aggregate signature scheme. However, their scheme suffers from the restriction that each time a new aggregate signature is generated, all participating signers must agree upon a common nonce not used by any signer before. More recently, Boldyreva et al. [15] propose an identity-based sequential aggregate signature scheme without that restriction.

2.2.4 Multisignatures

Multisignatures are specialized forms of aggregate signatures in which all signers sign the same message. In this dissertation, we build efficient multisignatures using primitives from LFSR sequences (described in the next section) to address the issues of efficiency and scalability in two contexts, namely, authenticating feedback in multicast applications, and authenticating source routes in mobile ad-hoc and sensor networks.

The concept of a multisignature was first proposed by Itakura et al. [55]. Itakura et al. build their multisignature scheme using RSA [82]; the signature shares the same drawback of Lysyanskaya et al. [68]: The verifier needs to know the signing order to verify the multisignature. Harn [48] propose a DL-based multisignature scheme in which the signers generate their individual signatures in parallel, and the signatures are combined by a designated node to form a multisignature. Harn's scheme requires the signers to

combine their individual ephemeral public keys into a common aggregate ephemeral public key with the help of the designated node; the scheme needs three communication rounds to achieve completion. Horster et al. [51] present a generalized ElGamal signature scheme [38], integrating several ElGamal variants, including Schnorr's signature [84] and the DSA, and present a generalized construction of a multisignature [50].

Micali et al. [72] formalize the concept of multisignatures and present the first formal model of security for multisignatures. Micali et al. also present a provably secure (in the random oracle model), three-round multisignature scheme based on the Schnorr variant [84]. The schemes of Horster et al. [50] and Micali et al. [72] used the idea of ephemeral public key aggregation, originally developed by Harn [48]. Boldyreva et al. [14] present a multisignature scheme based on short signatures by Boneh et al. [18].

2.2.5 Blind signatures

Blind signatures are a specialized form of digital signature in which signature generation involves an interactive protocol executed by an entity (the **owner**) possessing the message and another entity (the **signer**) possessing a long-term secret key, also known as the signing key. The owner transforms the message into a "blinded" message, and sends it to the signer. The signer uses its signing key to generate a signature on the blinded message and returns the signature to the owner. The owner transforms this signature such that (1) the transformed signature is a valid signature on the original message under the long-term public key of the signer and (2) the signer cannot associate the (message, transformed signature) pair with the owner. This transformed signature is known as a blind signature.

Blind signatures have potential as key building blocks of network security protocols, where an authority needs to vouch for the legitimacy of a message but the ownership of the message must be kept secret from the authority. Chaum [30] give the first proposal of blind signatures with the goal of developing untraceable payment systems that offer improved auditability but preserve personal privacy. Camenisch et al. [21] propose two blind signature schemes, one based on the variant of DSA and the other providing message recovery. Horster et al. [49] develop a blind multisignature scheme and use it as a building block to construct electronic voting applications. Petersen et al. [79] show how to use blind signatures to generate self-certified public keys. Pointcheval et al. [80] formalize the notion of security for blind signatures in the random-oracle model [8] and provide examples of provably secure blind signature schemes. Boldyreva [14] present a blind signature construction using bilinear pairings on elliptic curves. (We provide a short discussion on pairings on elliptic curves in

the Appendix.) Abe [3] develops a Schnorr-based blind signature scheme, provably secure in the random oracle model, and presents an application in E-Cash systems.

2.3 Use of LFSR sequences in cryptography

In this section, we describe the mathematics behind linear feedback shift register (LFSR) sequences and present various definitions, useful properties and representations of LFSR sequences. The section also contains the construction of two public-key cryptosystems based on LFSR sequences.

Chapters 3, 4, 5, and 6 are self-contained as regards construction of the security protocols. The reader interested in a particular problem in network security or construction of a security protocol proposed in this dissertation may safely skip this section, unless he or she wants to have a rigorous understanding of the underlying cryptographic primitives leading to the construction of the proposed protocols.

Table 2.3 lists the set of notations that we use throughout this dissertation.

2.3.1 LFSR sequences

Sequences in finite fields whose terms depend linearly on a fixed number of predecessors are called **linear recurring sequences**. Let $a, a_0, a_1, \ldots, a_{n-1}$ be elements of a finite field $\mathbb{F}_q$, where n is a positive integer. A sequence of elements $s_0, s_1, \ldots, s_{n-1}$ over $\mathbb{F}_q$ is called an nth order linear recurring sequence in $\mathbb{F}_q$ if the following property holds:

$$s_{n+k} = a + \sum_{i=0}^{n-1} a_i s_{k+i}, \quad k = 0, 1, \ldots \tag{2.3}$$

Given the elements $a, a_0, a_1, \ldots, a_{n-1}$, the terms $s_0, \ldots, s_{n-1}$ uniquely determine the rest of the sequence and are referred to as **initial values**. The linear recurring sequence is said to be homogeneous if $a = 0$. In this thesis, we focus on homogeneous linear recurring sequences. Such sequences can be implemented by an electronic switching circuit called an LFSR. An LFSR consists of the following three components.

- Shift register: An nth order LFSR consists of n storage units (implemented by flip-flops) regulated by a single clock. At each clock pulse, the content of each storage unit is shifted to the next unit in line. Each storage unit can have q (a prime number) states. For example, a binary shift register has 2-state storage units.

- Initial state: The contents of the n storage units at any particular time represent the state of the nth order LFSR. The row vector $(s_k, s_{k+1}, s_{k+2}, \ldots, s_{k+n-1})$ containing n

Table 2.3: Notations

Notation	**Meaning**
$\mathbb{F}_q$	Finite field where q is prime or a power of a prime.
$(\mathbb{F}_q)^*$	Corresponding multiplicative group.
$\mathbf{s}$	Sequence generated by an LFSR, represented by $s_0, s_1, \ldots$, where $s_i \in \mathbb{F}_q$.
$\mathbb{F}[x]$	Set of all polynomials $\{\sum_{i=0}^{n} a_i x^i, \quad a_i \in \mathbb{F}, n \geq 0\}$, where $\mathbb{F}$ is a field.
$\mathsf{Tr}(\alpha)$	Trace function with α in $\mathbb{F}_{q^n}$ as input, produces $\alpha + \alpha^q + \cdots + \alpha^{q^{n-1}}$ in $\mathbb{F}_q$.
s_k	The k^{th} sequence term.
$\bar{s_k}$	The k^{th} state of the sequence, represented by the row vector $(s_k, s_{k+1}, s_{k+2}, \ldots, s_{k+n-1})$ containing n consecutive sequence terms generated by an nth order LFSR.
$f(x)$	Characteristic polynomial of $\mathbf{s}$. For a cubic LFSR, $f(x) = x^3 - ax^2 + bx - 1$, where $a, b \in \mathbb{F}_q$. The polynomial $f(x)$ has roots of the form $\alpha, \alpha^q, \alpha^{q^2}$.
$f_k(x)$	Polynomial with roots $\alpha^k, \alpha^{kq}, \alpha^{kq^2}$. Represented by the term s_k in the XTR-PKC and by (s_k, s_{-k}) in GH-PKC. We drop the indeterminate x from the polynomials $f(x)$ and $f_k(x)$ for simplicity.
$s_r(f_e)$	The r^{th} term of the characteristic sequence generated by the polynomial f_e.
SK	Long-term private key. Node n_a randomly chooses x_a in $\mathbb{Z}_Q^*$ as its long-term private key.
PK	Long-term public key. Given $SK = x_a$, node n_a computes its long-term public key $PK = \bar{s_{x_a}}$.

consecutive sequence terms generated by an nth order LFSR is called the kth state of the sequence, and is represented by the notation $\bar{s_k}$. The initial state (or seed) of an nth order LFSR is represented by the row vector $\bar{s_0} = (s_0, s_1, \ldots s_{n-1})$. In a binary LFSR, each storage unit can have values 0 or 1, and $s_i \in \mathbb{F}_2$; more generally in a q-ary LFSR, $s_i \in \mathbb{F}_q$.

- Linear feedback function: The linear feedback function is of the form $f : \mathbb{F}_q^n \mapsto \mathbb{F}_q$, where $\mathbb{F}_q^n = \{(s_0, s_1, \ldots s_{n-1}) \mid s_i \in \mathbb{F}_q\}$ is a vector space over the finite field $\mathbb{F}_q$ of dimension n.

The linear feedback function of an LFSR can be expressed as follows:

$$f(x_0, x_1, \ldots, x_{n-1}) = a_0 x_0 + a_1 x_1 + \cdots + a_{n-1} x_{n-1}, \quad a_i \in \mathbb{F}_q \tag{2.4}$$

Given the above linear feedback function, the LFSR produces an nth order homogeneous linear recurring sequence represented as follows:

$$s_{n+k} = \sum_{i=0}^{n-1} a_i s_{k+i}, \quad k = 0, 1, \ldots \tag{2.5}$$

The polynomial

$$f(x) = x^n - a_{n-1}x^{n-1} - \cdots - a_0 \quad \in \mathbb{F}_q[x] \tag{2.6}$$

is called the **characteristic polynomial** of the above homogeneous linear recurring sequence. Later in this section, we provide the derivation of the characteristic polynomial.

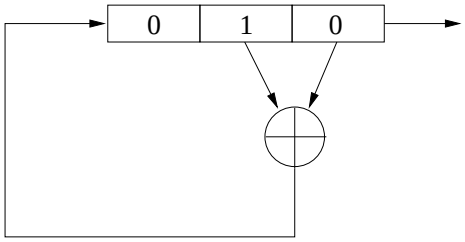
Sequences of large period are used in cryptography to thwart attempts of any brute-force attack that may be launched by an adversary. Periodicity of a sequence can be defined as follows.

Definition 2.3.1 (Periodicity of Sequences)**.** *Let* $\mathbf{s} = s_0, s_1, \cdots = \{s_k\}$ *be a sequence over* $\mathbb{F}_q$ *generated by a q-ary LFSR. If there exist integers* $Q > 0$ *and* $u \geq 0$ *such that* $s_{i+Q} = s_i$ *for all* $i \geq u$*, then the sequence is said to be* ***ultimately periodic*** *and the smallest integer* Q *is called the* ***period of the sequence****. The sequence is said to be* ***periodic*** *if* $u = 0$*.*

Let $\mathbf{s}$ be a sequence generated by an nth order LFSR over the finite field $\mathbb{F}_q$. The period of the sequence s, $per(\mathbf{s})$, is at most $q^n - 1$. Fig. 2.1 shows an example of a cubic LFSR, that is, $x = 3$, over the finite field $\mathbb{F}_2$, with a linear feedback function $f(x_0, x_1, x_2) = x_0 + x_1$. The linear recurring sequence generated by the third-order LFSR can be represented as:

$$s_{3+k} = s_{1+k} + s_k, \quad k = 0, 1, \ldots \tag{2.7}$$

Figure 2.1: A third order LFSR over $\mathbb{F}_2$, with feedback $f(x_0, x_1, x_2) = x_0 + x_1$



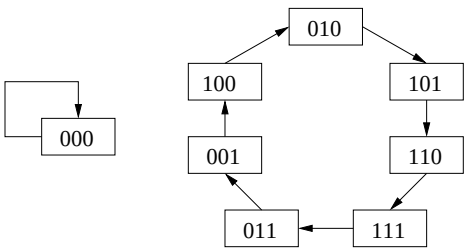
The corresponding truth table of the feedback function $f(x_0, x_1, x_2) = x_0 + x_1$ is shown in Table. 2.4.

Table 2.4: Truth table of $f(x_0, x_1, x_2) = x_0 + x_1$

$x_2x_1x_0$	$f(x_0, x_1, x_2) = x_0 + x_1$
000	0
001	1
010	1
011	0
100	0
101	1
110	1
111	0

The corresponding state diagram of the above LFSR is shown in Fig. 2.2.

Figure 2.2: State diagram of LFSR



Following the state diagram in the above figure, we observe that the output sequence with the initial state 010 is:

$$0101110\ldots$$

The above sequence has the period seven. Next, we describe the concept of characteristic and minimal polynomials of LFSRs.

2.3.2 Characteristic and minimal polynomials of LFSRs

Let $V(\mathbb{F}_q)$ be the set of all infinite sequences whose elements are taken from $\mathbb{F}_q$. The set $V(\mathbb{F}_q)$ can be expressed as:

$$V(\mathbb{F}_q) = \{(s_0, s_1, s_2, \ldots) \mid s_i \in \mathbb{F}_q\}$$

Consider the following two sequences in $V(\mathbb{F}_q)$.

$$\begin{aligned} \mathbf{a} &= (a_0, a_1, a_2, \ldots) \\ \mathbf{b} &= (b_0, b_1, b_2, \ldots), \quad c \in \mathbb{F}_q \end{aligned}$$

Addition and scalar multiplication of two sequences in $V(\mathbb{F}_q)$ can be defined as follows:

$$\begin{aligned} \mathbf{a} + \mathbf{b} &= (a_0 + b_0, a_1 + b_1, a_2 + b_2, \ldots) \\ c\mathbf{a} &= (ca_0, ca_1, ca_2, \ldots) \end{aligned}$$

The zero sequence $(0, 0, 0, \ldots)$ is denoted by $\mathbf{0}$.

Definition 2.3.2 (Left Shift Operator [43])**.** *The left shift operator L is defined on members of $V(\mathbb{F}_q)$ as follows:*

$$L^i(\mathbf{s}) = (s_i, s_{i+1}, s_{i+2}, \ldots)$$

By convention, we denote $L^0(\mathbf{s}) = I(\mathbf{s}) = \mathbf{s}$, where I is the identity operator. Using the above definition of the left-shift operator, Equation 2.5 can be written as:

$$\begin{aligned} L^n(\mathbf{s}) &= \sum_{i=0}^{n-1} a_i L^i(\mathbf{s}) \\ \Rightarrow (L^n &- \sum_{i=0}^{n-1} a_i L^i)(\mathbf{s}) = \mathbf{0} \end{aligned} \tag{2.8}$$

Now, consider any polynomial $f(x) \in \mathbb{F}[x]$ of degree n:

$$f(x) = x^n - \sum_{i=0}^{n-1} a_i x^i \tag{2.9}$$

Consider the function $F_f : \mathbf{s} \mapsto \mathbf{s}$. The range of this function not only depends on the sequence it takes as input, namely, $\mathbf{s}$, but also depends on the nature of the polynomial f. Now, consider the function F_f using the above polynomial given in Equation 2.9:

$$F_f(\mathbf{s}) = f(L)(\mathbf{s}) = (L^n - \sum_{i=0}^{n-1} a_i L^i)(\mathbf{s}) \tag{2.10}$$

From Equations 2.5 and 2.10, we observe that there is a set of infinite sequences in $V(\mathbb{F}_q)$, for which there exists a monic (leading coefficient equals unity), non-zero polynomial $f(x) \in \mathbb{F}[x]$ of degree n such that the function $F_f = f(L)(\mathbf{s})$ evaluates to the zero sequence $\mathbf{0}$. The polynomial $f(x)$ is called the **characteristic polynomial** of $\mathbf{s}$.

The following example [43] illustrates the idea of using the left-shift operator and the characteristic polynomial of an LFSR sequence. We leave it upto the reader to verify every step.

Example 1. *Consider an LFSR with the feedback function* $f(x_0, x_1, x_2, x_3) = x_0 + x_1$. *Also, let* $\mathbf{s} = (000100110101111)$ *be a sequence over* $\mathbb{F}_2$, *having period* $2^3 + 2^2 + 2 + 1 = 15$ *generated by the LFSR. (This sequence is obtained when the LFSR has initial state* 0001.*)*

The sequence $\mathbf{s}$ *satisfies the following linear recursive relation:*

$$s_{4+k} = s_k + s_{1+k}, \quad k = 0, 1, \ldots \tag{2.11}$$

The corresponding characteristic polynomial is given by:

$$f(x) = x^4 + x + 1 \tag{2.12}$$

The following equation can be easily verified (1. Apply the left shift operator four consecutive times on the sequence. 2. Add the resulting sequence to the sequence obtained by left shifting the original sequence once. 3. Finally add the resulting sequence to the original sequence):

$$f(L)(\mathbf{s}) = (L^4 + L + I)(\mathbf{s}) = (000000000000000) = \mathbf{0}$$

Let $G(f)$ denote the set of all sequences in $V(\mathbb{F}_q)$ with $f(L)(\mathbf{s}) = \mathbf{0}$ and $A(\mathbf{s})$ denote the set of all polynomials satisfying the condition $f(L)(\mathbf{s}) = \mathbf{0}$. Thus,

$$A(\mathbf{s}) = \{f(x) \in \mathbb{F}[x] \mid f(L)(\mathbf{s}) = 0\}$$

Definition 2.3.3 (Minimal polynomial of sequence)**.** *The minimal polynomial of* $\mathbf{s}$ *over* $\mathbb{F}_q$ *is the monic polynomial (leading coefficient equals unity) with the lowest degree in* $A(\mathbf{s})$*.*

We conclude this discussion with the following theorem, the proof of which is out of the scope of this document. The details of the proof can be found in [75].

Theorem 2.3.1. *Let* $\mathbf{s}$ *be an LFSR sequence with the minimal polynomial* $f(x)$ *of degree* n*. Assume that* $f(x)$ *is an irreducible polynomial over* $\mathbb{F}_q$ *of degree* n*. Let* α *be a root of* $f(x)$ *in* $\mathbb{F}_{q^n}$*. Then the period of the sequence generated by the LFSR, the period of the corresponding minimal polynomial,* $per(f(x))$*, and the order of the root of* $f(x)$ *are related as follows:*

$$per(\mathbf{s}) = per(f(x)) = ord(\alpha) \tag{2.13}$$

where the period of $f(x)$ *is defined as the smallest integer* Q *such that* $f(x)|(x^Q - 1)$ *and* $ord(\alpha)$ *denotes the order of the root of* $f(x)$*.*

The degree of the minimal polynomial of a sequence $\mathbf{s}$ is called the **linear complexity** of $\mathbf{s}$. The linear complexity of the sequence $\mathbf{s}$ represents the length of the shortest LFSR that can generate the sequence.

2.3.3 Trace representation of LFSRs

Every term of a sequence generated by an LFSR can be effectively represented using the trace function. In this section, we describe the trace function and its properties, and discuss the trace representation of LFSR sequences. We use the trace representation of sequence terms extensively in the security analysis of our proposed signature schemes.

Consider an irreducible polynomial over $\mathbb{F}_q$ of degree n.

$$f(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \tag{2.14}$$

Let α be a root of $f(x)$. Then we can construct $\mathbb{F}_{q^n}$ with $f(x)$ as a defining polynomial as follows:

$$\mathbb{F}_{q^n} = \{a_0 + a_1\alpha + \cdots + a_{n-1}\alpha^{n-1} \mid a_i \in \mathbb{F}_q\} \tag{2.15}$$

The trace function is defined as follows.

Definition 2.3.4 (Trace Function)**.** *Consider the base field $\mathbb{F}_q$ and the extension field $\mathbb{F}_{q^n}$ and let $\alpha \in \mathbb{F}_{q^n}$. The trace function $\mathsf{Tr} : \mathbb{F}_{q^n} \mapsto \mathbb{F}_q$ is defined by:*

$$\mathsf{Tr}(\alpha) = \alpha + \alpha^q + \cdots + \alpha^{q^{n-1}} \tag{2.16}$$

To understand that the trace function maps elements from the field $\mathbb{F}_{q^n}$ to the set $\mathbb{F}_q$, the reader should observe that $(\mathsf{Tr}(\alpha))^q = \mathsf{Tr}(\alpha)$. The trace function is a linear transformation from the field $\mathbb{F}_{q^n}$ to $\mathbb{F}_q$:

1. $\mathsf{Tr}(\alpha + \beta) = \mathsf{Tr}(\alpha) + \mathsf{Tr}(\beta)$, for all $\alpha, \beta \in \mathbb{F}_{q^n}$.
2. $\mathsf{Tr}(c\alpha) = c\mathsf{Tr}(\alpha)$, for all $\alpha \in \mathbb{F}_{q^n}$, $c \in \mathbb{F}_q$.

Proofs of the above properties are out of the scope of the dissertation and can be found in [65]. The following theorem states the relation between a sequence term and the root of the characteristic polynomial that generates the sequence.

Theorem 2.3.2. *Let $\mathbf{s} = \{s_k\}$ be a sequence generated by an nth order LFSR whose characteristic polynomial $f(x)$ is irreducible over $\mathbb{F}_q$. Let α be a root of $f(x)$. Then, there exists a uniquely determined $\beta \in \mathbb{F}_{q^n}$ such that the following holds for every integer $k \geq 0$:*

$$s_k = \mathsf{Tr}(\beta\alpha^k) \tag{2.17}$$

The reader is referred to the classical work by Niederreiter [75] for details the proof. LFSRs have been used extensively as key stream generators in stream ciphers and random number generators.

In the following section, we describe how LFSRs have been used to construct public-key cryptosystems.

2.3.4 Construction of two cryptosystems

Recently, Gong et al. [44], and Lenstra et al. [63] propose the Gong-Harn public-key cryptosystem (GH-PKC) and the Lenstra-Verheul public-key cryptosystem (XTR-PKC), respectively, based on cubic (third-order) LFSR sequences. The security of GH- and XTR-PKC is based on the difficulty of solving the discrete logarithm problem in the extension field $\mathbb{F}_{q^3}$. However, all computations involved with sequence terms are performed in the base field $\mathbb{F}_q$, which leads to substantial savings, both in communication and computational overhead, at a desired level of security.

Cubic LFSR-based PKCs are constructed as follows. Let $f(x)$ be a monic irreducible polynomial over the finite field $\mathbb{F}_q$.

$$f(x) = x^3 - ax^2 + bx - 1 \quad a, b \in \mathbb{F}_q \tag{2.18}$$

where q is prime ($q = p$) for the GH-PKC and the square root of q is a prime ($q = p^2$) for the XTR-PKC. Let us consider the following cubic-LFSR sequence $\{s_k\} = s_0, s_1, \ldots$ over $\mathbb{F}_q$ having the characteristic polynomial $f(x)$. The sequence $\{s_k\}$ is represented as:

$$s_{k+3} = as_{k+2} - bs_{k+1} + s_k; \quad k = 0, 1, \ldots \tag{2.19}$$

Following the trace representation of sequences (Theorem 2.3.2), each sequence term can be expressed as $s_k = \mathsf{Tr}(\alpha^k); k = 0, 1, \ldots$ where α is a root of $f(x)$ and the initial state of the cubic-LFSR is given by $s_0 = 3, s_1 = a$ and $s_2 = a^2 - 2b$. The sequence $\{s_k\}$ is called the third-order characteristic sequence of $f(x)$. The kth term of the characteristic sequence, s_k, generated by $f(x)$ is denoted $s_k(a, b)$ or $s_k(f)$. The period of $f(x)$ is defined as the smallest integer Q such that $f(x)|(x^Q - 1)$. Since the polynomial $f(x) \in \mathbb{F}_q[x]$ is irreducible over $\mathbb{F}_q$ and $\{s_k\}$ is generated by $f(x)$, $Q = per(\{s_k\}) = per(f) = ord(\alpha)$. In GH-PKC [44], $q = p$, and Q is a prime factor of $p^2 + p + 1$. In XTR-PKC [63], $q = p^2$, Q is a prime factor of $p^2 - p + 1$ and the characteristic polynomial $f(x)$ has the form:

$$f(x) = x^3 - ax^2 + a^p x - 1, \quad a \in \mathbb{F}_q \tag{2.20}$$

The monic irreducible polynomial $f(x)$ has roots of the form $\alpha, \alpha^q, \alpha^{q^2} \in \mathbb{F}_{q^3}$ ($f(x)$ is the minimal polynomial of α). Let $f_k(x)$ denote the minimal polynomial of α^k. Using the Newton Identity, $f_k(x)$ can be represented as [41, 44]:

$$f(x) = x^3 - s_k(a, b)x^2 + s_{-k}(a, b)x - 1 \tag{2.21}$$

where $s_{-k}(a, b) = s_k(b, a)$ is the kth term of the reciprocal sequence generated by $f^{-1}(x) = x^3 - bx^2 + ax - 1$. (In XTR, $s_{-k}(a, b) = s_k^p(a, b)$.) Hence, in GH (short for GH-PKC), f_k can be represented by (s_k, s_{-k}) and in XTR f_k by s_k.

Let the notation $s_r(f_e(x))$ denote the rth term of the characteristic sequence generated by the polynomial $f_e(x)$[1]. Algorithms for sequence term computations use the following **Commutative Law** [44] for characteristic sequences:

[1]We henceforth drop the indeterminate x from the notation $f_e(x)$ for simplicity.

Theorem 2.3.3. *Commutative Law [44]: For all integers r and e, the r^{th} term of the characteristic sequence generated by the polynomial f_e equals the the e^{th} term of the characteristic sequence generated by the polynomial f_r, which in turn equals the $(re)^{th}$ term of the characteristic sequence generated by the polynomial f.*

$$s_r(f_e) = s_r(s_e(a,b), s_{-e}(a,b)) = s_{re}(a,b) = s_e(f_r) \tag{2.22}$$

Proof: We know that $s_e = \mathsf{Tr}(\alpha^e) = \alpha^e + \alpha^{qe} + \alpha^{q^2 e}$. Also, the minimal polynomial of α^e can be expressed as $f_{\alpha^e}(x) = (x - \alpha^e)(x - \alpha^{qe})(x - \alpha^{q^2 e}) = x^3 - s_e x^2 + s_{-e} x - 1$. Now, the rth term of the characteristic sequence generated by the polynomial $f_e(x)$ can be written as:

$$\begin{aligned} s_r(s_e(a,b), s_{-e}(a,b)) &= (\alpha^e)^r + (\alpha^{qe})^r + (\alpha^{q^2 e})^r \\ &= s_{er}(a,b) \end{aligned}$$

This proves the theorem.

□

Sequence Operations: The sequence terms are computed by the following two sequence operations [45]:

- OP_1: Given an integer k and f_e, compute $\bar{s}_{ke}$.
- OP_2: Given $\bar{s}_k$ and $\bar{s}_e$ (both integers k and e need not be known), compute $\bar{s}_{k+e}$.

2.3.5 Cubic LFSR-based Diffie-Hellman

Gong et al. [44] and Lenstra et al. [63] independently propose the cubic-LFSR based Diffie-Hellman scheme. The Diffie Hellman scheme, `CLFSR-DH`, can be divided into three phases: initialization, key generation, and key agreement. Let entities A and B participate in the cubic-LFSR based DH scheme. The `CLFSR-DH` works as follows:

Initialization (`CLFSR-DH.Init`): Entities A and B agree upon the public parameters $\mathsf{params} = \langle p, Q, f(x) \rangle$, which can be described as follows:

- Find integer j such that $Q = j^2 - j + 1$ is a 160 bit prime.
- Find integer k such that $p = j + k * Q$ and p is of the order of 170 bits, and p is of the form 2 mod 3. Set $q = p^2$.

- Form characteristic polynomial $f(x) = x^3 - ax^2 + bx - 1$, an irreducible polynomial over $\mathbb{F}_p$, with period Q. (For XTR, $b = a^p$.)

The reader is referred to [63] for a thorough discussion on parameter selection for the XTR cryptosystem.

Key generation (`CLFSR-DH.KeyGen`):

Entity A randomly chooses integer $e \in \mathbb{Z}_Q^*$ as its private key and computes the public key $f_e = \mathsf{OP}_1(e, f)$. Similarly entity B randomly chooses private key $r \in \mathbb{Z}_Q^*$ and computes f_r. For GH, the public key f_e is represented by the dual terms (s_e, s_{-e}) and for XTR, f_e is represented by s_e.

Key agreement (`CLFSR-DH.KeyAgg`): Entities A and B engage in DH key agreement as follows:

1. $A \rightarrow B : f_e = s_e$ [$= (s_e, s_{-e})$ for GH]
2. $B \rightarrow A : f_r = s_r$ [$= (s_r, s_{-r})$ for GH]
3. A : $\mathsf{OP}_1(e, f_r) = s_{er}$ [And $\mathsf{OP}_1(-e, f_r) = s_{-er}$ for GH]
4. B : $\mathsf{OP}_1(r, f_e) = s_{er}$ [And $\mathsf{OP}_1(-r, f_e) = s_{-er}$ for GH]

In Steps 1 and 2, entities A and B send each other their respective public keys. In Steps 3 and 4, A and B compute the eth and rth sequence terms of characteristic sequences generated by f_r and f_e respectively. Following Theorem 2.3.3, entities A and B compute the common session key s_{er} in the XTR-mode (and (s_{er}, s_{-er}) in the GH-mode). In the XTR-DH protocol, entities A and B compute and share a single element in $\mathbb{F}_q$ as the session key; in the GH-DH protocol, A and B compute and share a pair of elements in $\mathbb{F}_q$ as the session key.

The following example illustrates `CLFSR-DH` in the GH cryptosystem.

Example 2. *Consider the finite field $\mathbb{F}_q$, where $q = 5$ and $f(x) = x^3 + x - 1$. The initial state of the cubic LFSR is given by $s_0 = 3, s_1 = a = 0, s_2 = a^2 - 2b = -2 \mod 5 = 3$.*

The characteristic sequence generated by $f(x)$ is:

$$3, 0, 3, 3, 2, 0, 1, 2, 4, 4, 3, 0, 1, 3, 4, 3, 4, 1, 4, 3, 2, 1, 1, 1, 0, 0, 1, 0, 4, 1, 1, 3, 0, 3, \ldots$$

The characteristic sequence generated by the reciprocal polynomial $f^{-1}(x) = x^3 - x^2 - 1$ *is the reverse of the characteristic sequence of* $f(x)$*:*

$$3, 1, 1, 4, 0, 1, 0, 0, 1, 1, 1, 2, 3, 4, 1, 4, 3, 4, 3, 1, 0, 3, 4, 4, 2, 1, 0, 2, 3, 3, 0, 3, 1, 1, \ldots$$

The period of the sequence (also the period of $f(x)$*) is* $5^2 + 5 + 1 = 31$*.*

Consider the polynomial $f_9(x) = x^3 - s_9x^2 + s_{-9}x - 1 = x^3 - 4x^2 + x - 1$*. The characteristic sequence generated by* $f_9(x)$ *is:*

$$3, 4, 4, 0, 0, 4, 1, 0, 3, 3, 4, 1, 3, 0, 3, 0, 2, 1, 2, 4, 0, 3, 1, 1, 1, 4, 1, 1, 2, 3, 1, 3, 4, 4, \ldots$$

Similarly, the characteristic sequence generated by $f_4(x) = x^3 - s_4x^2 + s_{-4}x - 1 = x^3 - 2x^2 - 1$ *is:*

$$3, 2, 4, 1, 4, 2, 0, 4, 0, 0, 4, 3, 1, 1, 0, 1, 3, 1, 3, 4, 4, 1, 1, 1, 3, 2, 0, 3, 3, 1, 0, 3, 2, 4, \ldots$$

Entity A sends $f_9(x) = (s_9, s_{-9}) = (4, 1)$ *to entity B. Entity B sends* $f_4(x) = (s_4, s_{-4}) = (2, 0)$ *to entity A.*

Entity A computes the ninth sequence term generated by the polynomial $f_4(x)$ *equals* $s_9(f_4) = 0$*. A also computes the term* $s_{-9}(f_4) = 1$*. Similarly, entity B computes the fourth sequence term generated by the polynomial* $f_9(x)$ *equals* $s_4(f_9) = 0$*. B also computes the term* $s_{-4}(f_9) = 1$*. The* ***DH key*** *is:*

$$(s_9(f_4), s_{-9}(f_4)) = (s_4(f_9), s_{-4}(f_9)) = (0, 1)$$

We construct all our protocols using the XTR cryptosystem [63], a cubic-LFSR based PKC. However, all protocols proposed in the dissertation can be seamlessly extended to PKCs based on higher-order LFSR sequences, with minor modifications, depending on the desired security level. A brief mathematical explanation behind reduced representations of finite field elements is as follows. Consider an element $\alpha \in (\mathbb{F}_{q^n})^*$. Using a polynomial basis over $\mathbb{F}_q$, given an integer i, α^i can be represented by $n \log q$ bits. The goal is to obtain a smaller representation of any such α^i. It can be shown that depending on the underlying finite field $\mathbb{F}_{q^n}$, α^i can be represented by [41]:

$n - 1 \log q$ bits	for general values of q and n;
$n/2 \log q$ bits	if $q = p^2$ and n is even;
$(n-1)/2 \log q$ bits	if $q = p^2$ and n is odd;

The XTR-PKC [63], is a special form of the third case and the GH-PKC [45] belongs to the first case.

We consider 1024-bit finite fields, that is $\mathbb{F}_{q^n}$ with $n \log q = 1024$ as the security benchmark. We choose the XTR-PKC, where $n = 3$ (cubic), p is a 170-bit prime and $q = p^2$ is 340 bits and α^i, and hence every sequence term, can be represented by $(n-1)/2 \log q = (3-1)/2 * 340$ bits $= 340$ bits.

2.4 Conclusion

In this chapter we have presented the basics and special forms of digital signatures. We have also provided the mathematical background of LFSR sequences and construction of a cryptosystem using LFSR sequences.

In the following chapters, we present our research in building efficient and scalable authentication techniques in four networking contexts: (1) authenticating feedback in multicast applications [27], (2) authenticating source routes in mobile ad hoc networks [25], (3) authenticating routes advertisements in inter-domain routing protocols [24], and (4) providing accountability in privacy-preserving systems [23].

Chapter 3

Authenticating feedback in multicast applications

Distributed applications involving large numbers of participants proliferate in the modern Internet. Examples include file sharing, software updates, news feeds, video conferencing, multi-player games and replicated databases. The Internet Protocol (IP) multicast service [31] was developed to support such applications. It offers an **abstraction** mechanism: The network hides the number and location of recipients from the sender, enabling it to treat an arbitrary number of participants as a single group. For a variety of reasons, however, network-level IP multicast service has never achieved widespread deployment. More recently, researchers have proposed **overlay** multicast schemes, implemented at the application layer, as a substitute to traditional multicast schemes. Significant research has been done in designing deployable, scalable overlay multicast schemes [2, 5]. Overlays offer several advantages as a vehicle for multicast implementation: They are incrementally deployable, adaptable and customizable [56]. Many such schemes offer the same abstraction mechanism as network-level IP multicast: Participants need not be aware of the identities of other participants but can treat them as a group. Multicast applications are sometimes classified based on the number of data sources: One-to-many multicast schemes contain a single source, which transmits data to many receivers; many-to-many schemes allow every participant to be a potential source. Only the former type is generally considered to scale to a large number (thousands to millions) of receivers; examples of applications include software distribution, multimedia transmission, and real-time news feeds.

3.1 Problem statement

A common problem faced by large-scale one-to-many multicast applications is **implosion**. If the source needs to obtain feedback from the receivers—for example, information about loss rates, which might be used for adaptive congestion control, or acknowledgment information—it is not feasible for receivers to simply transmit their information to the source. First, the **funneling** effect of the multicast topology causes congestion as feedback traffic nears the multicast source (for both native and overlay networks). Second, the receipt of individual unicast messages from receivers breaks the multicast abstraction, forcing the

source to deal with receivers as individuals. Researchers have developed a number of approaches to address these issues, including sampling [16] and **concast** [20], an "inverse" abstraction service analogous to multicast.

The design of efficient and scalable security mechanisms for multicast applications, whether IP- or overlay-based, also poses significant challenges. The problem is inherently more complex than point-to-point two-party security, and the mechanisms used for unicast communication cannot be directly applied. A significant amount of research has, therefore, focused on sender and data authentication, anonymity, non-repudiability, and key management for multicast-based applications.

In this chapter, we focus on the problem of collecting authenticated feedback in multicast applications. Consider a source multicasting information to a large number of nodes, where the transmitted information is subject to loss. The source would, therefore, like to verify—securely—that the data has been reliably delivered to the intended receivers, even in the presence of an adversary who might send bogus information to the source.

The "standard" unicast-like solution would have receivers send signed acknowledgments (Acks) to the source. The use of digital signatures on the Acks prevents the source from being fooled by fake Acks that the adversary might inject. Unfortunately, this approach leads to implosion at the source, and also forces the source to deal with receivers individually. We call this the **signed-Ack implosion problem** in which the source needs to verify all individual signatures on the Acks that it receives. The challenge is to find an efficient, scalable and fault-tolerant technique of convincing the source that the intended multicast receivers have indeed received the multicast data.

3.2 Protocol overview

We propose a cryptographic technique for combining multiple signatures on a message into a single compact signature as a solution to the signed-Ack implosion problem. As a building block of our solution, we present a novel cubic (third-order) linear feedback shift register (LFSR)-based single-signer signature scheme, `CLFSR-S`, following the EG I.4 [51] variant (shown in Table 2.1) of the ElGamal signature family. (A single-signer signature scheme means there is a single entity signing the message, as opposed to multiple signers in the case of a multisignature. The verifier is also a single entity, though the verification algorithm is public and thus any number of entities can verify the signature provided they have the public key of the signer.) Although `CLFSR-S` is not provably secure, it can serve as the basis of a highly efficient, single round, tree-based multisignature scheme `CLFSR-MS` owing to its

unique construction/signature format. Before we discuss the cryptographic details of the signature schemes, we present the system model, the basic idea of the protocol, and discuss the concept of public key aggregation using a trusted third party.

3.2.1 System model

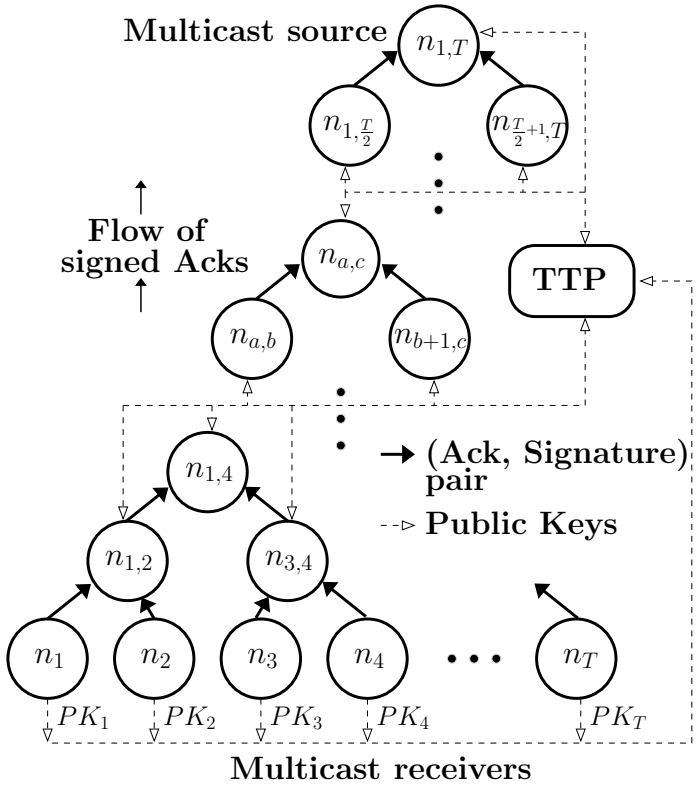
Multicasting via an overlay network offers several advantages compared to using traditional IP multicast service [56]. Overlay multicast networks are built on top of a substrate network. The substrate network consists of nodes (routers) that are connected by links (IP paths). A set of specialized nodes, called Multicast Service Nodes (MSNs), are distributed in the substrate network following MSN placement strategies [5].

Each MSN acts as a replication engine able to create multiple copies of incoming data and forward each copy to other MSNs or deliver it to an end host. A multicast receiver registers with a single MSN to receive multicast data. Prior to multicast data distribution, the MSNs organize themselves into a multicast delivery tree using a suitable routing algorithm. The root of the multicast delivery tree is the MSN to which the multicast source subscribes. Multicast feedback originates at the receivers of the multicast data and rises through the multicast delivery tree to reach the root MSN. An arbitrary internal node of the tree only needs to know its parent, and not the entire network topology, to deliver the multicast feedback. In addition to the primary functions, such as data replication, tree formation, group management, and feedback processing, the MSNs can serve as key-management and key-distribution centers.

Fig. 3.1 depicts a simple model of the multicast/feedback delivery network as a tree rooted at the source. We model our solution in the form of a regular binary tree for simplicity, although the proposed solution works for any rooted tree. The T leaf nodes $n_1, n_2, \ldots, n_T$ are the multicast receivers that send signed Acks toward the source $n_{(1,T)}$. An arbitrary intermediate node $n_{(a,b)}$ is the root of the subtree with leaves $n_a, n_{a+1}, \ldots, n_b$ (n_i is a simplified notation for $n_{(i,i)}$). The set of all nodes in the multicast delivery tree is denoted as $\mathbb{N}$. Any node $n_{(a,b)} \in \mathbb{N}$ maintains local information about the network topology (children and parent) for receiving and forwarding signed Acks.

Each leaf node n_a generates its own private and public keys (SK_a, PK_a) following the key-generation technique of the underlying cubic LFSR-based cryptosystem and registers its public key PK_a with the TTP. The TTP checks the validity of the registered public keys and provides each node trusted copies of public keys of its children. Each leaf node n_a signs each Ack with its public key PK_a, following the LFSR-based signature scheme presented in

Figure 3.1: Multicast feedback delivery tree



Section 3.4.1 and sends the (Ack, signature) pair to its parent. Each parent, $n_{(a,b)}$, of leaf nodes n_a and n_b does the following:

1. Requests and receives trusted copies of public keys PK_a and PK_b of its children, n_a and n_b, from a trusted third-party (TTP). (This step is only needed for the first Ack; the TTP can go offline for subsequent Acks.)

2. Verifies the received signatures.

3. Combines PK_a and PK_b to form the aggregate public key $PK_{(a,b)}$ by the procedure described in Section 3.2.2.

4. Combines the signatures to form a multisignature.

5. Sends the (Ack, multisignature) pair to its parent node.

6. Registers the aggregate public key $PK_{(a,b)}$ with the TTP.

Each intermediate node $n_{(a,c)}$ does the same with the multisignatures that it receives from its children $n_{(a,b)}$ and $n_{(b+1,c)}$ ($b = \frac{a+c-1}{2}$), using the respective aggregate public keys

$PK_{(a,b)}$ and $PK_{(b+1,c)}$ received from the TTP, sends the (Ack, multisignature) pair to its parent and registers the aggregate public key $PK_{(a,c)}$ with the TTP.

This wave of signed-Ack verification, multisignature generation and public key aggregation propagates up the tree to the source. The source needs to verify only two multisignatures (those of its children $n_{(1,\frac{T}{2})}$ and $n_{(\frac{T}{2}+1,T)}$) to authenticate whether the multicast message was reliably delivered to the intended multicast receivers.

In this chapter, we focus on the construction of a highly efficient multisignature suitable for authenticating feedback in multicast applications. We omit a complete discussion on the details of distributed public key management and distribution via multiple special function MSNs. In this thesis, a centralized trusted third-party (TTP) is modeled as a form of abstraction, which serves to validate and distribute public keys to nodes in the multicast delivery tree.

Next, we describe the cryptographic details of aggregating public keys of the multicast receivers.

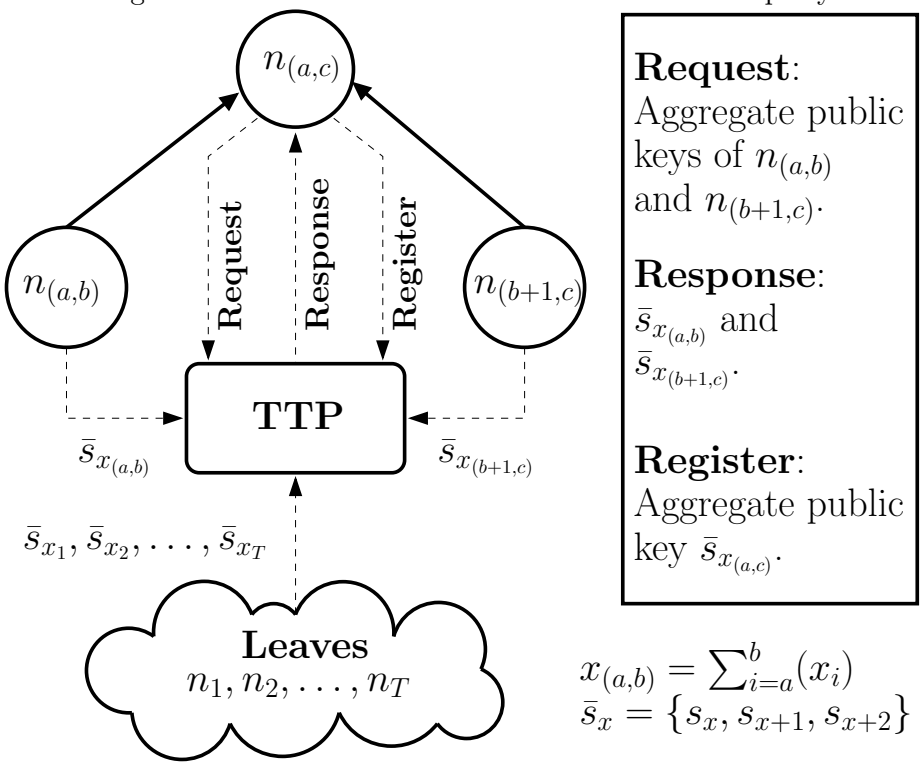
3.2.2 Aggregating public keys

An authentication protocol is composed of two distinct phases: bootstrapping phase and authentication phase. In the bootstrapping phase, the verifier securely provides the claimant with something the claimant should know, namely, the **bootstrapping material**. The verifier later requires the claimant to demonstrate knowledge of the bootstrapping material to prove its identity. If the bootstrapping material consists of symmetric key data, the channel over which the data is sent needs to be both authenticated and confidential. In the realm of public-key cryptography, the concept of the bootstrapping is an exchange of public keys; the channel need not be confidential (passive eavesdropping is allowed), but it must an authentic channel.

In our protocol, we use a TTP to validate, generate (optionally), and distribute the bootstrapping material constituting of individual and aggregate public keys. Fig. 3.2 depicts a functional model of the TTP. Each leaf node, n_a, randomly chooses a long-term private key $SK = x \in_R \mathbb{Z}_Q^*$, computes a public key $PK_a = \bar{s}_{x_a} = \{s_x, s_{x+1}, s_{x+2}\}$, and registers the public key PK_a the TTP.

Each node, $n_{(a,b)}$, whose children are leaf nodes, requests and receives the individual public keys, $\bar{s}_{x_a}$ and $\bar{s}_{x_b}$ of its children n_a and n_b, from the TTP and uses them to verify the signatures that node $n_{(a,b)}$ receives. Then the node computes the aggregate public key $PK_{(a,b)} = \bar{s}_{x_{(a,b)}} = \bar{s}_{(x_a+x_b)} \leftarrow \mathsf{OP}_2(\bar{s}_{x_a}, \bar{s}_{x_b})$ (the terms x_a, x_b, and $x_{(a,b)}$ all belong to $\mathbb{Z}_Q^*$)

Figure 3.2: A functional model of the trusted third party



and registers it with the TTP. (The sequence operation OP_2 is defined in Chapter 2.) The TTP validates the aggregate public keys and stores them. Each intermediate node, $n_{(a,c)}$ ($n_{(a,c)} \neq n_{(1,T)}$) requests and receives the aggregate public keys, $\bar{s}_{x_{(a,b)}}$ and $\bar{s}_{x_{(b+1,c)}}$, of its respective (non-leaf) children $n_{(a,b)}$ and $n_{(b+1,c)}$, from the TTP and uses them to verify the multisignatures that the intermediate node receives. Node $n_{(a,c)}$ computes the aggregate public key $PK_{(a,c)} = \bar{s}_{x_{(a,c)}} = \bar{s}_{(x_{(a,b)}+x_{(b+1,c)})} \leftarrow \mathsf{OP}_2(\bar{s}_{x_{(a,b)}}, \bar{s}_{x_{(b+1,c)}})$ and registers it with the TTP. Thus, an aggregate public key $PK_{(a,b)}$ has the form $PK_{(a,b)} = \bar{s}_{x_{(a,b)}}$, where $x_{(a,b)} = \sum_{i=a}^{b}(x_i)$ is in $\mathbb{Z}_Q^*$.

There is no existence of an aggregate private key corresponding to a particular aggregate public key. An intermediate node $n_{(a,c)}$ computes the aggregate public key $PK_{(a,c)} = \bar{s}_{x_{(a,c)}}$ given public keys $PK_{(a,b)}$ and $PK_{(b+1,c)}$ and does not know the corresponding aggregate private key $x_{(a,c)}$. Only leaves have individual private keys.

The protocols proposed in this chapter do not require the existence of additional special nodes serving as the TTP. Existing MSNs can have added functionality, constituting a distributed TTP service. Once the MSNs have organized themselves into the multicast delivery tree, the TTP service can distribute keys and go offline. The TTP service needs to be available only in the following cases: (1) A single or group of nodes join the multicast

tree. (2) Node(s) leave the multicast tree. (3) Nodes are unavailable temporarily (the length of time is a network design parameter) due to link failure(s). In these cases, nodes need to generate new aggregate public keys and also register them with the TTP so that other nodes can receive certified copies of new aggregate We omit a detailed description of public-key management services by a third party service in this thesis. We refer the reader to mechanisms for authentic public-key distribution via zero-knowledge proof of knowledge techniques by Micali et al. [72] for a thorough discussion on the subject.

In following sections, we present the construction of the LFSR-based signature schemes (single-signer and multisignature) used to authenticate the feedback in multicast applications. However, before we do so, we need to address the problem of choosing the underlying signature variant to construct a scalable and fault-tolerant multisignature. Scalability in a tree-based multisignature requires that each Ack round involve a single round of communication between the leaf nodes (multicast receivers) and the root node (the source). Fault-tolerance requires that leaf nodes need not re-send their signed Acks when intermediate nodes receive invalid signatures or do not receive a signature from one of their children. We now describe our rationale for choosing a suitable variant within the ElGamal family of digital signatures.

3.3 Choosing a suitable ElGamal variant for scalability

The challenge is to choose a signature variant within the ElGamal family that can result in a single-round, efficient, and fault-tolerant tree-based multisignature.

3.3.1 Eliminating variants of the ElGamal family

Horster et al. [51] propose 5088 signature variants of the ElGamal family. They analyze the variants and observe that some variants involve faster signature generation and verification than others. Four variants (EG I.3, I.4, II.3 and II.4) do not require computation of an inverse (modulo Q) during signature generation. If we exclude the multiplication operation from the possibilities of the function g (described in Chapter 2), we are left with the EG III.3, III.4, V.3 and V.4 variants. Horster et al. also observed that variants in which one of the parameters, A, B or C equals unity, need only two modular exponentiations, instead of three, during signature verification. So, the EG II/III/IV/V variants involve fast signature verification. Horster et al. proposed the EG I/II/III/V.3 and I/II/III/V.4 variants of the ElGamal family for fast signature generation and the EG II/III/IV/V variants for fast signature verification.

However, we observe that most efficient signature variants proposed by Horster et al. cannot be used to construct a scalable (single round) tree-based multisignature scheme. More specifically, we claim that any signature variant involving a permutation of $(\pm A, \pm B, \pm C)$ of the form $(f(m, *), \cdot, \cdot)$ as described in Table 2.2, where the second argument $* = r$ or $* = t$, will inherently result in a multisignature requiring at least three communication rounds to achieve protocol completion. One communication round consists of elements (like ephemeral public keys) traversing the entire tree from the leaf nodes to the source or vice versa.

We pick the Schnorr variant, an EG II.3 variant, and demonstrate the problem with using signature variants, involving a permutation of $(\pm A, \pm B, \pm C)$ of the form $(f(m, *), \cdot, \cdot)$, to construct a tree-based multisignature.

3.3.2 Problem with using the Schnorr variant

The first intuition to construct the tree-based multisignature is to start with a popular, provably secure signature variant, namely, the Schnorr variant. Both Micali et al. [72] and Castelluccia et al. [22] build their multisignatures with the Schnorr-type signing equation.

In the signing equation of the EG II.3 variant, the function $f(\cdot, \cdot)$ is modeled as a cryptographic hash function; f takes message m, along with the ephemeral public key r (unique to a signer) as inputs. In a tree-based multisignature, all signers n_i $(a \leq i \leq b)$ sign the same message and thus, also need to agree on a common aggregate ephemeral public key if the individual ephemeral public keys are combined with the messages. Specifically, in a tree-based multisignature, the least number of rounds can be described as follows:

Round 1: All signers n_i $(a \leq i \leq b)$ generate their own ephemeral public key r_i and send it to the multicast source.

Round 2: The source combines the individual r_i's to form the common aggregate ephemeral public key, $R = \prod_{i=a}^{b} r_i$, and sends R down the multicast tree to all nodes n_i.

Round 3: All signers generate their individual signatures on the hash $H(m, R)$. The hash value $H(m, R)$ is common to all signers and thus, the internal nodes in the tree can aggregate the individual signatures to form multisignatures, which can be propagated up the tree to the source.

In a multi-round tree-based multisignature, the signers (leaf nodes) need to cooperate in the process of signature aggregation. A single-round multisignature only requires the

leaf nodes to sign their respective Acks and send them toward the source; the leaves do not need to participate further in constructing the aggregate signature.

Moreover, a direct application of the Schnorr variant in building such multisignatures (as proposed by Micali et al. [72]) does not result in a fault-tolerant construction; whenever there is an invalid signature or a missing signature, the entire protocol has to be started from scratch. This property is undesired in distributed applications involving multicast feedback, since missing signatures can be common due to faults in the network and invalid signatures can result when an adversarial node suppresses a multicast receiver and injects bogus signed Acks in the delivery tree. An internal node that detects a missing or invalid signature should be still able to aggregate the remaining valid signatures and forward the multisignature toward the source. Castelluccia et al. [22] use the Schnorr variant and design a three-round, fault-tolerant version of a tree-based multisignature scheme using a Merkle-tree [71] aggregation technique to combine individual ephemeral public keys into a common aggregate ephemeral public key. However, the Merkle-tree aggregation technique results in degradation of efficiency and scalability of the protocol: Both communication and computation costs increase linearly with the depth of the tree (edges from aggregating node to source). We present a performance comparison of the schemes in Section 3.5.3.

We choose the EG I variant to construct our tree-based multisignature scheme and use primitives from LFSR sequences to enhance the efficiency of the proposed signature schemes.

3.4 Construction of the LFSR-based signature schemes

In this section, we present the cryptographic constructions of a novel single-signer digital signature scheme, `CLFSR-S`, using cubic LFSR sequences, and an efficient, single round multisignature scheme, `CLFSR-MS`, based on the `CLFSR-S`.

3.4.1 The single-signer signature scheme

`CLFSR-S` follows the EG I.4 variant of the ElGamal signature scheme family. `CLFSR-S` is a building block for the multisignature scheme, `CLFSR-MS`.

`CLFSR-S` consists of four phases: Initialization, key generation, signature generation and signature verification. During the initialization phase, the signer and the verifier choose and agree on the system public parameters: $\mathsf{params} = \langle p, Q, f(x), H \rangle$, where p, Q and $f(x)$ are chosen following the procedure described in Chapter 2, and $H : \{0,1\}^* \mapsto \mathbb{Z}_Q$ is a

cryptographic hash function. The signer generates its long-term private and public keys, $(SK, PK) = (x, \bar{s}_x)$ following the procedure described in Chapter 2.

Figure 3.3: Authenticating feedback in multicast applications: The CLFSR-S signature scheme

Signature Generation	**Signature Verification**
1. Randomly choose ephemeral private key $k \in_R \mathbb{Z}_Q^*$ and compute ephemeral public key $\bar{s}_k \leftarrow \mathsf{OP}_1(k, f)$. Let r denote the integer $s_k \mod Q$.	1. Compute $h = H(m)$.
2. Compute hash of message $h = H(m)$; Solve for t in the following equation: $t \equiv kr - xh \mod Q$.	2. Compute $A = f_{(th^{-1}+x)} \leftarrow \mathsf{OP}_2(th^{-1}, \bar{s}_x)$.
3. Compute $\bar{s}_{kr} \leftarrow \mathsf{OP}_1(\bar{s}_k, r)$.	3. Compute $B = f_{(rh^{-1}k)} \leftarrow \mathsf{OP}_1(h^{-1}, f_{kr})$. f_{kr} can be directly derived from $\bar{s}_{kr}$.
4. Send the signature $\sigma = \langle \bar{s}_{kr}, t \rangle$ and the message m to verifier.	4. Accept signature if $A = B$, else reject signature.

Fig. 3.3 describes the signature generation and signature verification phases of CLFSR-S. A naive cubic LFSR variant of EG I.4 generates a signature of the form $\sigma = \langle f_k, t \rangle$. We perform an additional computation in Step 3 of the signature generation process to compute the term $\bar{s}_{kr}$. The specific format of the individual signature that CLFSR-S generates enables us to efficiently construct the multisignature in a single round.

3.4.2 The proposed multisignature

All internal nodes of the multicast feedback delivery tree, shown in Fig. 3.1, follow the multisignature scheme, `CLFSR-MS`, to verify, aggregate, and create multisignatures on the signed Acks. We provide a definition of a multisignature, and present constructions of the cubic LFSR-based `CLFSR-MS` scheme following the definition.

Definition 3.4.1. *A multisignature scheme* `MS` *is the tuple* ⟨`Init`, `KeyGen`, `G`, `V`, `A`⟩ *whose components are defined as follows:*

Initialization (`Init`)*: A probabilistic polynomial-time (PPT) algorithm that takes a security parameter* λ *as input and outputs system public parameters* params*.*

Key Generation (`KeyGen`)*: A PPT algorithm that takes system parameters* params *as input and outputs a private and public key pair* (SK, PK)*.*

Signature Generation (`G`)*: A PPT signature generation algorithm that takes the system public parameters* params*, the ephemeral private key* k*, the long-term private key* SK*, and a message* m *as inputs, and outputs a signature* σ*. Not all signature generation algorithms are probabilistic. RSA signature generation is deterministic, which means there is no concept of ephemeral private/public key and repeated signing of a message with the long-term private key produces the same signature. However, in ElGamal signatures repeated signing of the same message with different randomly generated ephemeral private keys will result in different signatures. Also, the same ephemeral private key should not be used for signing more than once; there exists a type of forgery, called* ***Selective forgery*** *[87] if the ephemeral private key is re-used for signing messages.*

Signature Verification (`V`)*: A deterministic algorithm that takes the system public parameters* params*, public-key* PK*, a signature* σ *and the message* m *as inputs, and outputs a result,* ***Valid*** *or* ***Invalid****.*

Signature Aggregation (`A`)*: A deterministic algorithm that takes the system public parameters* params*, an array of multisignatures (or individual signatures in case of leaf nodes)* $\sigma_1, \sigma_2, \ldots, \sigma_n$ *as input, and outputs multisignature* $\sigma_{(1,n)}$*.*

Multisignature Verification (V)*: A deterministic algorithm that takes the system public parameters* params*, aggregate public-key* $PK_{(1,n)}$[1]*, the multisignature* $\sigma_{(1,n)}$ *and the messages* m *as inputs, and outputs a result,* ***Valid*** *or* ***Invalid****.*

The form of aggregation depends on the underlying network topology and the underlying signing equation. In the proposed tree-based multisignature, the internal nodes verify and aggregate the signatures and do not contribute their own signatures in the resulting multisignature. However, in the case of authenticating source routes in the DSR protocol, described in Chapter 4, the aggregating node includes its own signature on the source route in the multisignature. Following the above definition, we construct our proposed multisignature scheme CLFSR-MS as follows:

Initialization (CLFSR-MS.Init): All nodes choose and agree upon the system public parameters $\mathsf{params} = \langle p, Q, f(x), H \rangle$, where p, Q and $f(x)$ are as described in Chapter 2 and $H : \{0,1\}^* \mapsto \mathbb{Z}_Q$ is a cryptographic hash function.

Key Generation (CLFSR-MS.KeyGen): Each multicast receiver (leaf node), n_a, generates its long term private, public-key pair $(SK_a, PK_a) = (x_a, \bar{s}_{x_a})$. Each internal node generates aggregate public keys as described in Section 3.2.2.

Multisignature Generation (CLFSR-MS.G): Each multicast receiver, n_a, generates a signature $\sigma_a = \langle \bar{s}_{k_a r_a}, t_a \rangle$ on the hashed multicast acknowledgment, $h = H(m)$, following the CLFSR-S signature generation. Node n_a sends (h, σ_a) to its parent.

Multisignature Verification (CLFSR-MS.V): Each intermediate node $n_{(a,c)}$, receives signatures $\sigma_{(a,b)} = \langle t_{(a,b)}, \bar{s}_{K_{(a,b)}} \rangle$ and $\sigma_{(b+1,c)} = \langle t_{(b+1,c)}, \bar{s}_{K_{(b+1,c)}} \rangle$ from its children $n_{(a,b)}$ and $n_{(b+1,c)}$, where $b = \frac{a+c-1}{2}$, $K_{(a,b)} = \sum_{i=a}^{b}(k_i r_i)$ and $t_{(a,b)} = \sum_{i=a}^{b}(t_i)$. Node $n_{(a,c)}$ verifies the signatures $\sigma_{(a,b)}$ and $\sigma_{(b+1,c)}$ using the aggregate keys $PK_{(a,b)}$ and $PK_{(b+1,c)}$ and following the CLFSR-S signature verification process. For nodes that are parents of leaf nodes, the signature $\sigma_{(a,a)}$ denotes σ_a.

Multisignature Aggregation (CLFSR-MS.A): If the signatures $\sigma_{(a,b)}$ and $\sigma_{(b+1,c)}$ pass CLFSR-MS.V, the parent node $n_{(a,c)}$, if it is not the source node, generates the multisignature $\sigma_{(a,c)}$ by computing:

- $t_{(a,c)} \equiv t_{(a,b)} + t_{(b+1,c)} \mod Q$ and

[1]The aggregation of public keys is part of the initial set-up procedure and has been discussed in Section 3.2.2.

- $\bar{s}_{K_{(a,c)}} = \bar{s}_{K_{(a,b)}+K_{(b+1,c)}} \leftarrow \mathsf{OP}_2(\bar{s}_{K_{(a,b)}}, \bar{s}_{K_{(b+1,c)}})$.

A multisignature $\sigma_{(a,c)} = \langle t_{(a,c)}, \bar{s}_{K_{(a,c)}} \rangle$, under the aggregate public-key $PK_{(a,c)}$ propagates up the tree to the source. If the multisignatures $\sigma_{(1,\frac{T}{2})}$ and $\sigma_{(\frac{T}{2}+1,T)}$ pass the verification procedure `MS.V` at the source, the individual signatures $\sigma_1, \ldots, \sigma_T$ of all leaves $n_1, \ldots, n_T$ are verified collectively, and the source is convinced that the multicast data has been reliably delivered to the intended recipients.

In the following section, we perform an extensive analysis (correctness, security, and cost) of the multisignature scheme.

3.5 Analysis

3.5.1 Correctness

Correctness of a security protocol assumes honest behavior from the entities participating in the protocol. Dishonest behavior, for example forging of signatures, and dishonestly verifying a signature, is captured through through the security analysis of the protocol.

A tree-based multisignature scheme constructed in the above fashion is correct if the multisignatures, $\sigma_{(1,\frac{T}{2})}$ and $\sigma_{(\frac{T}{2}+1,T)}$, of the respective left and right children of the source, $n_{(1,T)}$, pass the verification procedure `CLFSR-MS.V` under the respective aggregate public keys $PK_{(1,\frac{T}{2})}$ and $PK_{(\frac{T}{2}+1,T)}$, provided:

1. Each node $n_{(a,b)} \in \mathbb{N}$ chooses and agrees upon the system public parameters $\mathsf{params} = \langle p, Q, f(x), H \rangle$.

2. Every leaf node n_a honestly executes `CLFSR-MS.K` to generate its public and private key pair (PK_a, SK_a) and executes `CLFSR-MS.G` to generate its signature σ_a on hashed Ack.

3. Every intermediate node $n_{(a,b)}$, honestly executes `CLFSR-MS.A` to generate multisignature $\sigma_{(a,b)}$.

Theorem 3.5.1. *The multisignature scheme* `CLFSR-MS` *is correct.*

Proof: Consider any arbitrary node $n_{(a,c)} \in \mathbb{N}$ whose left and right children are $n_{(a,b)}$ and $n_{(b+1,c)}$ ($b = \frac{a+c-1}{2}$). In this proof we show that the multisignatures, $\sigma_{(a,b)}$ and $\sigma_{(b+1,c)}$, of the respective left and right children of $n_{(a,c)}$ pass the corresponding verification procedure `CLFSR-MS.V` executed at $n_{(a,c)}$ under the aggregate public keys $PK_{(a,b)} = \bar{s}_{x_{(a,b)}}$ and $PK_{(b+1,c)} = \bar{s}_{x_{(b+1,c)}}$ provided the conditions mentioned above hold.

Observing the procedure CLFSR-MS.V for verification of the multisignature $\sigma_{(a,b)}$ using the aggregate public key $PK_{(a,b)}$, we know:

$$A = \mathsf{OP}_2(v, \bar{s}_{x_{(a,b)}}) = f_{v+x_{(a,b)}}$$

where $v = h^{-1} \sum_{i=a}^{b}(t_i)$ and $x_{(a,b)} = \sum_{i=a}^{b}(x_i)$. Since, $t_i \equiv k_i r_i - x_i h \mod Q$ for $a \leq i \leq b$ and $K_{(a,b)} = \sum_{i=a}^{b}(k_i r_i)$, we observe:

$$\begin{aligned} A &= f_{\sum_{i=a}^{b}(h^{-1}t_i + x_i)} = f_{\sum_{i=a}^{b}(h^{-1}k_i r_i)} \\ &= f_{h^{-1}K_{(a,b)}} = \mathsf{OP}_1(h^{-1}, f_{K_{(a,b)}}) \\ &= B \end{aligned}$$

Thus, $\sigma_{(a,b)}$ is valid under $PK_{(a,b)}$. Similarly, by replacing the terms $PK_{(a,b)}$, $t_{(a,b)}$ and $K_{(a,b)}$ with $PK_{(b+1,c)}$, $t_{(b+1,c)}$ and $K_{(b+1,c)}$, respectively, we can show that $\sigma_{(b+1,c)}$ passes the verification procedure CLFSR-MS.V under the public key $PK_{(b+1,c)}$. □

What remains to be shown is that it is hard for an adversary to deviate from the key pair and signature generation protocols and still generate a correct signature. However, this is precisely the issue of forgery that we discuss next.

3.5.2 Security

The security of CLFSR-MS is based on the difficulty of forging any individual signature or multisignature generated by any node in the multicast/feedback delivery tree. An adversary can eavesdrop on a channel, inject false messages, and run a PPT algorithm to forge a (multi)signature constructed by an arbitrary node. A successfully forged (multi)signature passes the CLFSR-MS.V verification procedure. We show that the EG I.4 variant reduces (in the security sense) to the multisignature CLFSR-MS. In other words, an adversary who successfully forges a CLFSR-MS multisignature, can forge an EG I.4 signature.

To the best of our knowledge, all existing multisignatures based on the DL problem have been constructed using variants of the ElGamal family of signatures. The EG I.4 variant uses the signing equation: $t \equiv xh + kr \mod Q$, where p and Q are large primes of the order of 1024 bits and 160 bits, respectively, with the condition that $Q|(p-1)$; $\alpha \in \mathbb{Z}_p^*$ is a generator of the cyclic subgroup of order Q, h is the hashed message, t is the signature on h, $x \in_R \mathbb{Z}_Q^*$ is the private key, α^x is the public key, $k \in_R \mathbb{Z}_Q^*$ is the ephemeral private key and $r = \alpha^k$ is the ephemeral public key.

The security of the proposed single-signer scheme, CLFSR-S, is based on the difficulty of solving the trace discrete logarithm (Tr-DL) problem [41, 44, 45, 63] in $\mathbb{F}_q$ [2]. The trace discrete logarithm (Tr-DL) problem and assumption (with $\mathbb{F}_{q^n} = \mathbb{F}_{q^3}$) can be defined as follows. We henceforth drop the suffix $\mathbb{F}_{q^n}/\mathbb{F}_q$ from the trace function, for simplicity of notation.

Definition 3.5.1 (Trace (Tr)-DL Problem/Assumption)**.** *Let α be a generator of the multiplicative group $(\mathbb{F}_{q^3})^*$, where q is a large prime or a power of a large prime. The Tr-DL Problem in $\mathbb{F}_q$ is: Given $(q, \alpha \in (\mathbb{F}_{q^3})^*, \beta \in \mathbb{F}_q)$, find an index k such that $\beta = \mathsf{Tr}(\alpha^k)$ or determine that there is no such index.*

Let $\mathcal{A}$ be a probabilistic polynomial time (PPT) algorithm that solves the Tr-DL problem. Define the advantage of the (t, ϵ) Tr-DL solver $\mathcal{A}$ as: $\mathsf{Adv}_{\mathcal{A}}^{TrDL} = Pr[\mathcal{A}(q, \alpha, \beta) = k \mid \alpha \in_R (\mathbb{F}_{q^3})^*, k \in_R \mathbb{Z}_Q^*, \beta = \mathsf{Tr}(\alpha^k)]$. The probability is over the random choices of α in $(\mathbb{F}_{q^3})^*$, k in $\mathbb{Z}_Q^*$ and the random bits of $\mathcal{A}$.

Tr-DL Assumption*: The finite field $\mathbb{F}_q$ satisfies the Tr-DL Assumption if $\mathsf{Adv}_{\mathcal{A}}^{TrDL}$ is negligible.*

Lemma 3.5.2 (Giuliani et al. [41])**.** *The Tr-DL Problem is equivalent to the DL problem.*

A **total break** of CLFSR-MS occurs if, given the public key $PK_{(a,b)} = \bar{s}_{x_{(a,b)}}$ of any node $n_{(a,b)}$, the adversary is able to compute the corresponding private key $SK_{(a,b)} = x_{(a,b)}$. In such a case, $n_{(a,b)}$'s signature can be forged. However, given $\bar{s}_x$, finding x is equivalent to solving the DL problem in the extension field $\mathbb{F}_{q^3}$ [45]. Using the following lemmas we show that, assuming a **total break** has not occurred, an adversary who can successfully forge a CLFSR-MS multisignature can successfully forge a signature in EG I.4.

Lemma 3.5.3. *The single-signer signature scheme CLFSR-S is equivalent to the EG I.4 variant.*

Proof: [$\Longrightarrow$] Given a valid CLFSR-S signature $\sigma = \langle \bar{s}_{kr}, t \rangle$ on hashed message h under the public key $\bar{s}_x$, we know that $f_{(h^{-1}t+x)} = f_{(h^{-1}kr)}$. Let $\alpha \in \mathbb{F}_{q^3}$ be a root of the characteristic polynomial $f(x)$. By the definition of $f_k(x)$ (given in Chapter 2), the roots of $f_{(h^{-1}t+x)}$ are:

$$\alpha^{(h^{-1}t+x)}, \alpha^{(h^{-1}t+x)q}, \alpha^{(h^{-1}t+x)q^2} \in \mathbb{F}_{q^3}$$

Also, the roots of $f_{(h^{-1}kr)}$ are:

[2] Refer to Chapter 2 for definition and properties of a trace function.

$$\alpha^{(h^{-1}kr)}, \alpha^{(h^{-1}kr)q}, \alpha^{(h^{-1}kr)q^2} \in \mathbb{F}_{q^3}$$

Furthermore, we know from the signing equation of CLFSR-S that $h^{-1}t+x \equiv h^{-1}kr \mod Q$. Thus, the root $\alpha^{(h^{-1}t+x)}$ of $f_{(h^{-1}t+x)}$ is equal to the root $\alpha^{h^{-1}kr}$ of $f_{(h^{-1}kr)}$. We now have $\alpha^{h^{-1}t+x} = \alpha^{h^{-1}kr}$ with $t \equiv kr - xh \mod Q$, which is the EG I.4 scheme. Thus, the CLFSR-S reduces to EG I.4.

[$\Longleftarrow$] Given a valid EG I.4 signature $t = \langle r, t \rangle$ on hashed message h under the public key α^x, we know that $\alpha^{(h^{-1}t+x)} = \alpha^{(h^{-1}kr)}$. Also, $\alpha^{(h^{-1}t+x)q} = \alpha^{(h^{-1}kr)q}$ and $\alpha^{(h^{-1}t+x)q^2} = \alpha^{(h^{-1}kr)q^2}$ where, $q = p^2$ and p is a large prime. We know:

$$\begin{aligned} s_{(h^{-1}t+x)} &= \mathsf{Tr}(\alpha^{h^{-1}t+x}) \\ &= \alpha^{(h^{-1}t+x)} + \alpha^{(h^{-1}t+x)q} + \alpha^{(h^{-1}t+x)q^2} \\ s_{(h^{-1}kr)} &= \mathsf{Tr}(\alpha^{h^{-1}kr}) \\ &= \alpha^{(h^{-1}kr)} + \alpha^{(h^{-1}kr)q} + \alpha^{(h^{-1}kr)q^2} \end{aligned}$$

Thus, $s_{(h^{-1}kr)} = s_{(h^{-1}kr)}$ with $t \equiv kr - xh \mod Q$, which is the CLFSR-S scheme. Thus, EG I.4 variant reduces to CLFSR-S. □

Lemma 3.5.4. *The single-signer signature scheme* CLFSR-S *reduces to the proposed multisignature scheme* CLFSR-MS.

Proof: Let there be $n = (b - a + 1) > 1$ signers. Suppose there exists a PPT forger $\mathcal{F}$, which given system parameters $\mathsf{params} = \langle p, Q, f(x), H \rangle$, public keys $\bar{s}_{x_a}, \bar{s}_{x_{a+1}}, \ldots, \bar{s}_{x_b}$ $(a < b)$ and message m as inputs, outputs a forged multisignature $\sigma^F_{(a,b)} = \langle t^F_{(a,b)}, \bar{s}^F_{K_{(a,b)}} \rangle$ on $h = H(m)$ with non-negligible probability. This means the forged signature $\sigma^F_{(a,b)}$ passes the verification procedure, CLFSR-MS.V, under the aggregate public key $\bar{s}_{x_{(a,b)}}$.

We show that given access to the PPT forger $\mathcal{F}$, system parameters params, an arbitrary public key $PK = \bar{s}_x$ (corresponding private key, $x \in \mathbb{Z}^*_Q$, unknown to the adversary), and message m^F, an adversary can output a forged signature $\sigma^F = \langle t^F, \bar{s}^F_{kr} \rangle$ on $h = H(m)$ that passes the verification procedure of the single-signer signature scheme, CLFSR-S, under public key PK. Algorithm 1 shows a polynomial-time reduction from the single-signer signature scheme CLFSR-S to the multisignature scheme CLFSR-MS.

Since the forged multisignature $\langle t^F_{(a,b)}, \bar{s}^F_{K_{(a,b)}} \rangle$ on $h^F = H(m^F)$ passes the verification procedure CLFSR-MS.V under the aggregate public key $\bar{s}_{x_{(a,b)}}$, we have:

input : $(\bar{s}_x, m^F)$
output: σ^F

1 Pick $(b-a)$ arbitrary long-term private keys $x_a, x_{a+1}, \ldots, x_{b-1} \in_R \mathbb{Z}_Q^*$, where, $b > (a+1)$, and compute the corresponding public keys $\bar{s}_{x_a}, \bar{s}_{x_{a+1}}, \ldots, \bar{s}_{x_{b-1}}$.
2 Compute $\bar{s}_{x_b} \leftarrow \mathsf{OP}_2(-\sum_{i=a}^{b-1} x_i, \bar{s}_x)$. $x_b = x - \sum_{i=a}^{b-1} x_i$.
3 $\langle t^F_{(a,b)}, \bar{s}^F_{K_{(a,b)}} \rangle \leftarrow \mathcal{F}(\mathsf{params}, \bar{s}_{x_a}, \bar{s}_{x_{a+1}}, \ldots, \bar{s}_{x_b}, m^F)$
4 Set $\bar{s}^F_{kr} = \bar{s}^F_{K_{(a,b)}}$ and $t^F = t^F_{(a,b)}$.
5 Return $\sigma^F = \langle t^F, \bar{s}^F_{kr} \rangle$ as the forged CLFSR-S signature on message m^F under public key $\bar{s}_x$.

Algorithm 1: Reduction of CLFSR-S to CLFSR-MS

$$A = \mathsf{OP}_2(t^F_{(a,b)}(h^F)^{-1}, \bar{s}_{x_{(a,b)}}) = \mathsf{OP}_1((h^F)^{-1}, f^F_{K_{(a,b)}}) = B$$

Given that $\bar{s}^F_{kr} = \bar{s}^F_{K_{(a,b)}}$, $t^F = t^F_{(a,b)}$ and $x_b = x - \sum_{i=a}^{b-1} x_i$, we have:

$$A = \mathsf{OP}_2(t^F(h^F)^{-1}, \bar{s}_x) = \mathsf{OP}_1((h^F)^{-1}, f^F_{kr}) = B$$

Thus, $\sigma^F = \langle t^F, \bar{s}^F_{kr} \rangle$ is a valid signature on m^F under public key $\bar{s}_x$ following the verification procedure of CLFSR-S. □

Theorem 3.5.5. *The EG I.4 variant of the Generalized ElGamal signature scheme reduces to the proposed multisignature scheme* CLFSR-MS.

Proof: The proof of the theorem is immediate from Lemmas 3.5.3 and 3.5.4. □

There can be an alternative solution to avoid the signed Ack implosion problem that works as follows. All nodes (leaf and intermediate) in the multicast feedback delivery tree are assumed to possess (private, public) key pairs. To deliver feedback to the source, each leaf node generates a signature on the Ack and sends the signed Ack to its parent. After receiving the signed Ack from its children, each intermediate node verifies the signatures. If signature verifications succeed, the intermediate node generates its own signature (using its private key) on the Ack and sends the signed Ack to its parent. This process continues until the source verifies the signed Acks received from its children to complete the feedback process. At each stage, a node receives (and needs to verify) the signatures from its children.

This solution has a serious security drawback. Any intermediate node can generate a signed Ack (dishonestly) even if the node did not receive signatures from its children. Each node must trust its children to honestly generate a signed Ack only after successful

verification of the signatures from its children. This trust relationship must be transitive throughout the multicast feedback delivery tree.

On the other hand, in our solution nodes need not trust their children to honestly report the receipt of signatures and generate signed Acks. The intermediate do not generate their own signatures on Acks and thus would need to forge signatures of its children to create falsely signed Acks. Replay attacks can be avoided by including a sequence number in each Ack.

The multisignature scheme, `CLFSR-MS`, though not provably secure, is engineered to be an efficient and scalable building block to solve the signed-Ack implosion problem in performance-sensitive multicast applications. In contrast, the multisignature scheme in Castelluccia et al.'s proposal [22], though provably secure (in the random oracle model), takes three communication rounds to achieve completion. We omit a thorough discussion on provable security; the reader may refer to an exemplary discussion on the subject by Koblitz et al. [61]. We present a performance comparison of the multisignature scheme `CLFSR-MS` with those of Castelluccia et al. [22] and Nicolosi et al. [74] in the following section.

3.5.3 Cost

Cubic LFSR-based PKCs [44, 63, 75] use reduced representations of finite field elements. Elements in an extension field $\mathbb{F}_{q^n}$ are represented by their corresponding minimal polynomials with coefficients in the base field $\mathbb{F}_q$. The security of LFSR-based PKCs is based on the difficulty of solving the DL problem in the extension field $\mathbb{F}_{q^n}$. However, all computations are performed in the base field $\mathbb{F}_q$.

Table 3.1 shows direct comparisons of `CLFSR-MS` with other schemes, namely, `ASM` by Micali et al. [72], `CASM` by Castelluccia et al. [22], and `NBLS` by Nicolosi et al. [74]. In Table 3.1, the term e is the cost of modular exponentiation, the term m is the cost of modular multiplication, h is the cost of hash operation, f is the cost of hashing onto a GDH group, p is the cost of a pairing computation, L represents the number of tree-edges along the path from node till source. `ASM` demonstrates a construction of a multisignature and is not intended to address the signed-Ack implosion problem; the data presented in Table 3.1 represents a direct application of `ASM` to a tree-based multisignature scheme. The computation/communication cost is based on a security benchmark of 1024 bits: The public parameters of `ASM` and `CASM` are given by the tuple $\texttt{params} = \langle p, Q, \alpha \rangle$, where p and Q are 1024 and 160 bit primes, respectively, and α is an element of order Q in $\mathbb{Z}_p^*$.

Table 3.1: Authenticating feedback in multicast applications: Cost comparison of CLFSR-MS with existing schemes.

	ASM [72]	CASM [22]	NBLS [74]	CLFSR-MS
Rounds	3	3	1	1
Computation cost per node	$4e + 3m + h$	$4e + (L+2)m$ $+(L+1)h$	$4p + f$ $+1m$	$2\mathsf{OP}_1 + h +$ $3\mathsf{OP}_2 + 2m$
Communication cost per link (bits)	480	$640 + 320L$	160	500
Underlying Problem	DLP	DLP	GDH	DLP/Tr-DLP
Fault Tolerance	NONE	Limited	FULL	FULL
Public-key size (bits)	2048	2048	766	680

CASM and ASM complete in three rounds due to the specific cryptographic construction of the Schnorr signature [84]. Loosely speaking, commitments from the signers, needed to construct the multisignature, need to traverse up the multicast tree and the common challenge for all signers needs to propagate down the tree, resulting in two additional rounds.

CLFSR-MS uses extremely fast LFSR sequence operations [63, 78] and can achieve the highest computational efficiency. Given $\alpha \in \mathbb{F}_{p^6}$, where p is a 170-bit prime, computing α^k for any integer k requires approximately $23.4 \log_2 Q$ multiplications in $\mathbb{F}_p$, where Q, the order of α, is a 160-bit prime [63]. However, computing the kth sequence term $s_k = \mathsf{Tr}(\alpha^k)$ given f (represented by $\mathsf{Tr}(\alpha)$) using sequence operation OP_1 takes only $8 \log_2(k \mod Q)$ multiplications in $\mathbb{F}_p$ which is approximately three times faster than computing α^k given α [63]. Thus, the computational cost for one $\mathsf{OP}_1 \approx 0.33e$.

NBLS uses considerably more expensive bilinear pairing operations in its signature/multisignature verification phase. The lowest known cost for computing a single Tate pairing equals approximately 11110 multiplications in $\mathbb{F}_q$, where q is a 171-bit prime (for a security benchmark of 1024 bits) [6].

In CASM, both communication and computation costs increase with the depth of the tree and CASM incurs the highest communication overhead (excluding overhead due to faults) per link of all the protocols we compare. CASM offers limited fault-tolerance: the protocol

can only tolerate a threshold number of communication faults (missing signatures) along the multicast feedback tree. The threshold depends on the modulus of the underlying DL problem and the total number of leaves in the tree. A direct application of `ASM` to the problem of authenticating multicast feedback does not result in a fault-tolerance protocol; the protocol needs to restart from scratch if any (multi)signature is missing or fails to verify.

The public key size (equivalent to 1024-bit RSA, excluding shared components of the public key) for `CLFSR-MS` is the least per node, followed by `NBLS` and, `CASM` or `ASM`. Moreover,

3.6 Related research

In this section, we provide a brief discussion on prior research related to the problem of authenticating multicast feedback. The large body of literature on reliable multicast and implosion avoidance is out of the scope of this dissertation. Readers can find a taxonomy of reliable multicast protocols in a survey by Levine et al. [64], and protocols on implosion avoidance can be found in works by Bolot et al. [16] and Calvert et al. [20]. We provide a background and related research on various forms of aggregate signatures in Chapter 2.

Any solution to the signed Ack implosion problem in large scale, performance-sensitive multicast applications must address the issue of efficiency and scalability. Scalability is achieved by minimizing or eliminating increase in communication and computational overhead with the number of multicast receivers, or, more generally, the size of the multicast group which includes the intermediate nodes in the multicast/feedback delivery network. Nicolosi et al. [74] present an Ack aggregation technique (`NBLS`) for peer-to-peer multicast using the multisignature scheme of Boldyreva et al. [14]. The `NBLS` scheme is scalable and requires a single communication round between the source and the receivers for multisignature construction. But, due to the expensive pairing operations performed at each node in the multicast/feedback delivery network, the `NBLS` scheme is inefficient. Castelluccia et al. [22], design an Ack compression technique (`CASM`) using the DL-based multisignature scheme of Micali et al. [72]. Though the operations in the `CASM` scheme are less expensive than pairing operations, the scheme requires three rounds of communication between the source and the receivers. Moreover, scalability of the `CASM` scheme is impaired by the linear increase in communication and computational overhead with the size of the multicast group.

3.7 Summary

The problem of collecting authenticated feedback in distributed applications is inherently more complex than point-to-point two-party security, and the mechanisms used for unicast communication typically cannot be directly applied. Although a large number (and varieties) of multisignatures have been proposed, only two protocols, namely `CASM` by Castelluccia et al. [22] and `NBLS` by Nicolosi et al. [74], are specifically tailored to the problem of collecting authenticated feedback in multicast applications.

The choice of the underlying signing equation and the network topology dictate the construction, and thus the efficiency and scalability, of the resulting multisignature scheme. We have considered the use of different variants of the ElGamal signature scheme and argued that choosing any of the EG II/III/IV/V variants (including the popular Schnorr variant [84]) cannot result in a single round, scalable and fault-tolerant construction of tree-based multisignatures.

Our multisignature scheme uses comparatively fast LFSR operations to achieve better computational efficiency and lower storage overhead than its competitors: `CASM`, `NBLS`, and a direct application of `ASM`. The security of the tree-based multisignature scheme `CLFSR-MS` is based on the Tr-DL Problem in $\mathbb{F}_q$. Our multisignature scheme was constructed using the XTR public-key cryptosystem for simplicity, although it can be seamlessly constructed using the GH public-key cryptosystem and can also be extended to PKCs based on higher order LFSR sequences, with minor modifications, depending on the desired security level.

Chapter 4

Authenticating source routing protocols in ad-hoc networks

An ad-hoc network can be best described as a network that can be quickly deployed in unfavorable conditions, where no wired infrastructure exists, and in situations where it is impractical to build and expect support from a fixed infrastructure. The idea of generating these "on-the-fly" networks was first proposed by the US Department of Defense in the 1970's as packet radio networks [58], which evolved into the Survivable Adaptive Radio Network program in the 1980's [35]; the goal was to rapidly deploy packet-switched networks and provide timely and assured communication for soldiers working in hostile environments. Since then, ad-hoc networks have gained immense popularity for commercial use, and have evolved—depending on application—to static ad-hoc networks (or sensor networks, for example to collect and monitor environmental data), mobile ad-hoc networks (or MANETs, used primarily for personal communication), and hybrid networks (fixed sensor nodes connected and managed by a higher tier of mobile nodes). Regardless of the use, ad-hoc networks are characterized by resource-constrained nodes that need to work in a cooperative and self-organized manner to perform network functions such as routing, and mobility management.

Designing secure routing protocols for mobile ad-hoc networks is a challenging task. Resource constraints of nodes, limited capacity of the wireless medium, node mobility, and the self-organized form of the network make it difficult to transfer techniques for securing traditional wired networks to the ad-hoc networking environment. The dynamic source routing protocol (DSR) is the most popular on-demand source routing protocol designed for multi-hop wireless ad-hoc networks [57]. DSR is simple and efficient in construction, offers loop-free routing guarantees and load balancing, uses only soft state, and is robust [57]. However, DSR does not consider an adversarial model of the underlying network. Thus, it is vulnerable to several forms of attack by malicious nodes, such as injection of bogus routing information and formation of feedback loops by colluding adversarial nodes [53, 59].

4.1 Problem statement

In the DSR protocol, the source initiates route discovery by generating an RREQ (route request) packet and broadcasting it to all its neighbors. The RREQ packet contains the

identity of the destination and the accumulated route from the source. Each node that is not the destination and has not previously encountered the RREQ packet appends its address to the source route and re-broadcasts the packet to its own neighbors. RREQ propagation continues until the destination is encountered. When the destination receives the RREQ packet, it generates a route reply (RREP) packet containing the accumulated source route and unicasts the RREP to the source along the reverse path of the source route.

We focus on the following problem: How can a source wanting to find a route to a destination be assured of the authenticity of the source route advertised in a received RREP packet? We would like to guarantee authenticity without imposing substantial overhead on the nodes that help in discovering routes.

4.2 Protocol overview

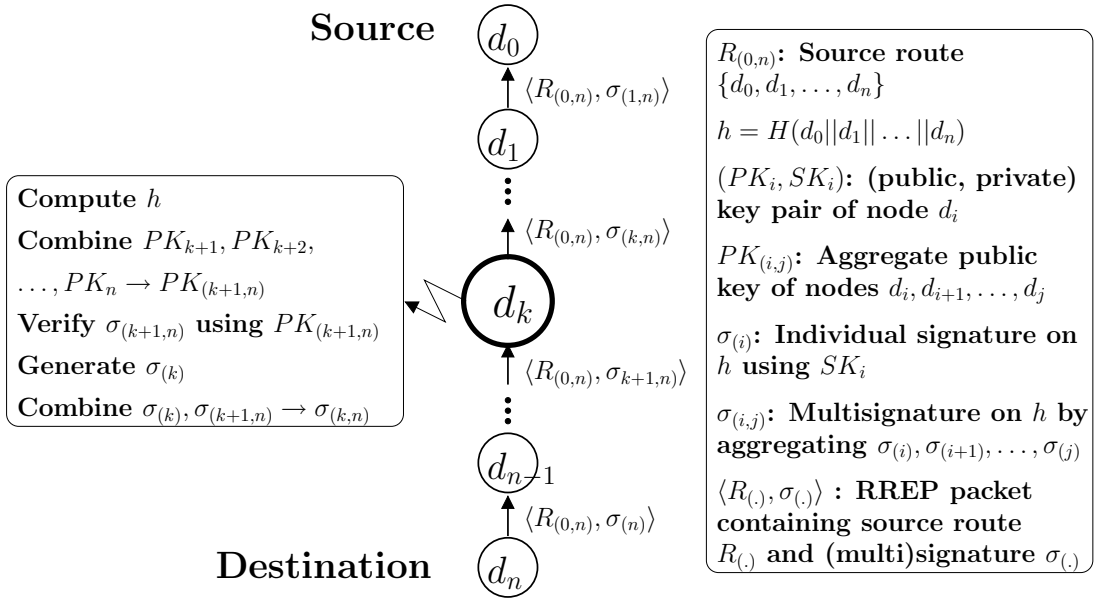
DSR is vulnerable to several forms of attacks by malicious nodes, including injection of bogus routing information and formation of feedback loops by colluding adversarial nodes [59]. The classical approach to mitigating attacks on DSR is to use cryptographic tools to authenticate information exchanged during the route discovery process. In this section, we present existing techniques for authenticating route discovery in DSR based on multisignatures.

4.2.1 Basic idea

A first technique for authenticating route discovery in DSR is to have each node sign RREQ packets they forwarded toward the destination, so that the destination can authenticate the accumulated source route before generating an RREP packet. However, DSR floods the network with RREQ packets during route discovery, so most nodes waste computation and communication resources by signing, verifying and forwarding RREQ packets that are not included in the eventual route. Also, combining signed RREQ packets implies combining signatures on different messages (source route accumulates more nodes as the RREQ propagates toward the destination) and a sequential aggregate signature must be used; such signatures are usually computationally more expensive than multisignatures. We therefore suggest an alternative technique to authenticate the source route contained in the RREP packet using an efficient, single round multisignature scheme, requiring no prior cooperation among nodes to construct the signature.

We first present our authentication method without considering caching of routes. Let nodes $\{d_0, \ldots, d_k, \ldots, d_n\}$ constitute a source route, where d_0 is the source and the d_n is the destination. We identify a node and its address by the same notation, d_k, for

Figure 4.1: Propagation and authentication of route replies



simplicity. First, let us assume that an arbitrary node d_k has authentic copies of public keys $PK_{k+1}, \ldots, PK_n$ of all nodes leading from it to the destination. Fig. 4.1 depicts the propagation of authenticated RREP packets from the destination d_n to the source d_0. Node d_k does the following:

1. Combines the public keys $PK_{k+1}, ..., PK_n$ to form aggregate public key $PK_{(k+1,n)}$.
2. Verifies multisignature $\sigma_{(k+1,n)}$ that it receives from node d_{k+1}. Aborts if verification fails.
3. Signs the hashed concatenation of the addresses $d_0, ..., d_n$ contained in the source route to create $\sigma_{(k)}$.
4. Combines $\sigma_{(k)}$ and $\sigma_{(k+1,n)}$ to form multisignature $\sigma_{(k,n)}$.
5. Replaces $\sigma_{(k+1,n)}$ with $\sigma_{(k,n)}$ in the RREP packet.
6. Sends the RREP packet to node d_{k-1}.

At the source d_0, successful verification of multisignature $\sigma_{(1,n)}$ under the aggregate public key $PK_{(1,n)}$ establishes the authenticity of all signatures on the source route. Signature verification by intermediate nodes facilitates early detection of bogus routes injected by an adversary. The procedures for combining public keys, generation, verification and aggregation of signatures are presented in further detail in Section 4.3.

The basic scheme described above can be exploited by an adversarial node $d_{n'}$ in the following way: Node $d_{n'}$ can store signed RREP packets and send an old signed RREP packet at a later time toward the source (an instance of a replay attack). This attack can be prevented by inserting a sequence number in the RREQ packet. The corresponding RREP packet also contains the same sequence number, which is signed along with the message.

Acs et al. [4] present an extensive discussion on various kinds on attacks on route finding in mobile ad-hoc networks. They propose a protocol `endairA` containing a set of mechanisms, including one similar to the above, for preventing such attacks.

We are primarily concerned with building efficient techniques of aggregating signatures on messages; our proposed multisignature using LFSR sequences can be applied to other protocols like `endairA`, which contains only single-signer signatures for authentication, for efficiently authenticating source routes.

4.2.2 Incorporating path caching

DSR is an on-demand routing protocol; it attempts to discover a route to a destination only when a source needs to send a data packet to that node. To avoid initiating route discovery before each data packet is sent, the source may decide to cache routes [52]. The RREP packet contains a complete sequence of links leading to the destination at all times. Similarly, intermediate nodes, forwarding the RREP packets, can (optionally) store complete paths in **path caches** so that they can efficiently reply to route requests at a later time. Path caches are simple to implement and also guarantee that all routes are loop-free, since all source routes contained in the RREP are loop-free themselves. The mechanism of caching is one of the most important enhancements made to DSR.

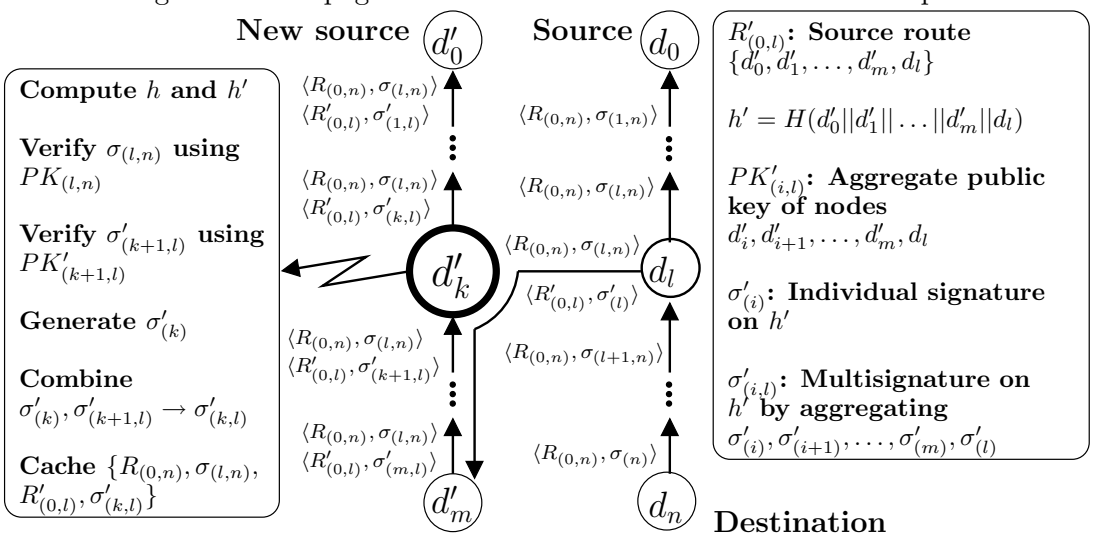
We extend the above technique to incorporate path caching[1]. Consider the case where source d_0 has already established a route to destination d_n as shown in Fig. 4.2. All nodes $\{d_0, d_1, \ldots, d_{n-1}, d_n\}$ cache the route, $R_{(0,n)}$, along with their respective multisignatures $\{\sigma_{(0,n)}, \sigma_{(1,n)}, \ldots, \sigma_{(n-1,n)}, \sigma_{(n)}\}$ (the destination caches its own signature $\sigma_{(n)}$). Suppose node d'_0 (a new source) now attempts to discover a route to the same destination d_n and its RREQ packet containing the accumulated route $\{d'_0, d'_1, \ldots, d'_m\}$ reaches node d_l as shown in Fig. 4.2.

Node d_l prepares the RREP packet containing the following:

1. Cached information $\langle R_{(0,n)}, \sigma_{(l,n)} \rangle$

[1]We use multisignatures in authenticating cached routes and do not consider using link caches [52].

Figure 4.2: Propagation and authentication of cached route replies



2. Accumulated route, signature pair $\langle R'_{(0,l)} = \{d'_0, \ldots, d'_k, \ldots d'_m, d_l\}, \sigma'_{(l)}\rangle$, where $\sigma'_{(l)}$ is node d_l's own signature on the hashed concatenation of the address in the accumulated route $h' = H(d'_0||\ldots||d'_m||d_l)$.

Node d_l sends the RREP packet to node d'_m. Now, consider an arbitrary node d'_k on the route back to d'_0. Node d'_k does the following:

1. Verifies multisignatures $\sigma_{(l,n)}$ on $R_{(0,n)}$ and $\sigma'_{(k+1,l)}$ on h' that it receives from node d'_{k+1} using aggregate public keys $PK_{(l,n)}$ and $PK'_{(k+1,l)}$ respectively. Abort if verification fails.

2. Generates its own signature $\sigma'_{(k)}$ on h'.

3. Combines signatures $\sigma'_{(k)}$ and $\sigma'_{(k+1,l)}$ to form multisignature $\sigma'_{(k,l)}$ on h'.

4. Caches $\{R_{(0,n)}, \sigma_{(l,n)}, R'_{(0,l)}, \sigma'_{(k,l)}\}$ as a route to d_n.

5. Sends an RREP packet containing $\langle R_{(0,n)}, \sigma_{(l,n)}\rangle$ and $\langle R'_{(0,l)}, \sigma'_{(k,l)}\rangle$ to node d'_{k-1}.

The RREP packet propagates to the source d'_0 in this fashion, and the source performs the same operations as node d'_k. Successful verification of multisignatures $\sigma_{(l,n)}$, $\sigma'_{(1,l)}$ under the aggregate public keys $PK_{(l,n)}$, $PK'_{(1,l)}$ establishes the authenticity of the route $R'_{(0,l)}$ and the partial route $\{d_{l+1}, \ldots, d_n\}$ contained in $R_{(0,n)}$. The irrelevant addresses $\{d_0, \ldots, d_{l-1}\}$ are not authenticated. Finally, source d'_0 extracts $\{d_{l+1}, \ldots, d_n\}$ from $R_{(0,n)}$ and appends the extracted route to $R'_{(0,l)}$ to obtain the desired route $\{d'_0, \ldots, d'_m, d_l, \ldots, d_n\}$. Similarly,

nodes in the route $R'_{(0,l)}$ may use cached information to reply to future RREQs encountered for destination d_n.

In the following section, we present an efficient, single round, multisignature scheme based on cubic LFSR sequences, suitable for authenticating route discovery in DSR.

4.3 The proposed multisignature scheme

First, we construct a single-signer signature scheme based on the EG I.4 variant of the ElGamal signature family (discussed in Chapter 2). The rationale behind choosing the particular variant from the entire ElGamal family is exactly the same as we discussed in Chapter 3. Choosing the Schnorr variant would require prior cooperation among the nodes participating in the RREQ phase of DSR and result in multiple communication rounds for a single instantiation of the multisignature. The signature scheme `CLFSR-S` consists of four phases: initialization, key generation, signature generation and signature verification; all four phases follow the exact procedures described in Chapter 3. The multisignature scheme, `CLFSR-M`, is built using `CLFSR-S` as a building block.

The multisignature scheme, `CLFSR-M`, consists of five phases: initialization, key generation (`CLFSR-M.K`), signature generation (`CLFSR-M.G`), multisignature verification (`CLFSR-M.V`) and multisignature generation (`CLFSR-M.A`). During the initialization phase, all nodes choose and agree upon the system public parameters $\mathsf{params} = \langle p, Q, f(x), H \rangle$. The process of key generation consists of the following steps:

1. Generation of individual long-term private/public key pair $(SK_l, PK_l) = (x_l, \bar{s}_{x_l})$ of node d_l.

2. Generation of aggregate public key $PK_{(l,n)} = \bar{s}_{x_{(l,n)}} \leftarrow \mathsf{OP}_2(\bar{s}_{x_l}, \bar{s}_{x_{(l+1,n)}})$ of nodes $d_l, d_{l+1}, \ldots, d_n$, where $x_{(l,n)} = \sum_{i=l}^{n}(x_i)$.

The signature generation, multisignature verification and multisignature generation phases of `CLFSR-M` work as follows:

Signature generation ($\texttt{CLFSR-M.G}(\mathsf{params}, SK_l, m = d_0||\ldots||d_n) \rightarrow \sigma_{(l)}$): Each node, d_l, participating in the RREP propagation generates an individual signature $\sigma_{(l)} = (\bar{s}_{k_l r_l}, t_l)$ on the hashed concatenation of the address in the source route $h = H(m)$ following the `CLFSR-S` signature generation (described in Chapter 3).

Multisignature Verification ($\texttt{CLFSR-M.V}(\mathsf{params}, PK_{(l+1,n)}, \sigma_{(l+1,n)}, m) \rightarrow (Valid, Invalid)$): Each intermediate node (other than the destination), d_l, receives a signed RREP

packet containing the multisignature $\sigma_{(l+1,n)} = (t_{(l+1,n)}, \bar{s}_{k_{(l+1,n)}})$, where $t_{(l+1,n)} = \sum_{i=l+1}^{n}(t_i)$ and $k_{(l+1,n)} = \sum_{i=l+1}^{n}(k_i r_i)$. Node d_l verifies $\sigma_{(l+1,n)}$ following the CLFSR-S signature verification procedure, using the aggregate public key $PK_{(l+1,n)} = \bar{s}_{x_{(l+1,n)}}$, where $x_{(l+1,n)} = \sum_{i=l+1}^{n}(x_i)$. For the node d_{n-1} (the last hop before the destination d_n) the signature $\sigma_{(l+1,n)}$ denotes σ_n.

Multisignature Generation ($\texttt{CLFSR-M.A}(\mathsf{params}, \sigma_{(l+1,n)}, \sigma_{(l)}) \rightarrow \sigma_{(l,n)}$): If the signature $\sigma_{(l+1,n)}$ passes the verification procedure, CLFSR-M.V, node d_l, generates the multisignature $\sigma_{(l,n)}$ by computing $t_{(l,n)} = t_{(l+1,n)} + t_l$ and $\bar{s}_{k_{(l,n)}} = \bar{s}_{k_{(l+1,n)}+k_l r_l} \leftarrow \mathsf{OP}_2(\bar{s}_{k_l r_l}, \bar{s}_{k_{(l+1,n)}})$. Node d_l finally replaces the multisignature $\sigma_{(l+1,n)}$ with the multisignature $\sigma_{(l,n)} = (t_{(l,n)}, \bar{s}_{k_{(l,n)}})$ to the RREP packet before forwarding the RREP to the next hop node d_{l-1}.

The wave of signature generation, multisignature verification and multisignature aggregation continues until the RREP packet containing the multisignature, $\sigma_{(1,n)} = (t_{(1,n)}, \bar{s}_{k_{(1,n)}})$, is delivered to the source. If the multisignature $\sigma_{(1,n)}$ passes the verification procedure, CLFSR-M.V, under the aggregate public key $PK_{(1,n)}$, the individual signatures $\sigma_{(1)}, \ldots, \sigma_{(n)}$ of corresponding nodes $d_1, \ldots, d_n$ in the discovered source route to the destination d_n are verified collectively.

In the following section, we present a discussion on the policy aspects of bootstrapping authentication protocols in ad-hoc networks.

4.4 A discussion on distributing public keys

An authentication protocol is typically composed of two distinct phases, namely, the bootstrapping phase and the authentication phase. In the realm of public key cryptography, entities need to use authentic channels (need not be confidential) to exchange public keys constituting the "bootstrapping material" [28]. Once this exchange has taken place, entities can authenticate each other by proving the possession of their corresponding private keys.

4.4.1 Using a trusted third party

A trusted third party (TTP) can distribute certified public keys (the bootstrapping material) and also provide a way to check the validity of certificates via publishing certificate revocation lists. An online TTP works as follows: An arbitrary node d_k wanting to authenticate the source route can request and receive certified copies of public keys $PK_{k+1}, \ldots, PK_n$ of nodes leading to the destination from the TTP. However, an online

TTP in an ad-hoc network introduces circular dependency between the need for a TTP to perform secure routing and the need to find a secure route to the TTP. Moreover, public keys have to be redistributed when network membership changes, that is when nodes join or leave the network. To avoid the necessity of an online TTP, an offline TTP can distribute all certified public keys to all nodes when the network is set up. Such an offline TTP may not be viable, since nodes would need to store all certified public keys. Various solutions of bootstrapping authentication have been built for securing ad hoc networks, each having its own disadvantages [91]. In essence, the assumption of a TTP-based public key management policy in an ad-hoc networking paradigm is not practical. Delegating specialized functions to a single node or a small subset of nodes [100, 62] does not suit the ad-hoc networking paradigm. These restrictions motivate us towards a fully distributed public key management policy.

4.4.2 Toward a fully distributed, self-organized bootstrapping

Pretty Good Privacy (PGP) [101] is a policy-based mechanism for public key management that can distribute certified copies of public keys in the absence of a centralized TTP. In PGP, each node generates its own (public, private) key pair and certifies its own public key as well as public keys of other nodes based on trust policies. Similarly, in an ad-hoc network, when two nodes come within radio range of each other, they can certify each other's public keys, based on policies. This process of certification creates a **certificate graph** $G = (V, E)$, where the set of vertices is represented as:

$$V = \{d_0, d_1, \ldots, d_N\} \tag{4.1}$$

and the set of edges in the graph is represented as:

$$E = \{(d_i, d_j) : \forall i, j : 0 \leq i, j \leq N, \exists\, \sigma_{SK_i}(d_j, PK_j)\} \tag{4.2}$$

where N is the total number of nodes in the network and $\sigma_{SK_i}(d_j, PK_j)$ denotes node d_i's signature on node d_j's public key. When a node d_i wants to verify the authenticity of public key PK_j of node d_j, node d_i tries to find a simple path in the certificate graph that can be expressed as:

$$d_i \rightsquigarrow d_j = d_i \rightarrow d_{i_0} \rightarrow \ldots d_{i_n} \rightarrow d_j$$

where $d_{i_k} \rightarrow d_{i_l} \implies (d_{i_k}, d_{i_l}) \in E$. Capkun et al. [89, 90] study PGP certificate graphs and observe that trust graphs in self-organized systems, for example mobile ad-hoc

networks, naturally exhibit the **small-world** phenomenon. Informally, a graph is said to exhibit the small-world property if any two nodes in the network are likely to be connected through a short sequence of intermediate acquaintances. Since the first experimental study by Milgram [73], several network models [60, 93] have been built to study the problem analytically.

In our public key management model, individual nodes store, manage and distribute certificates themselves in a such a way that the size of the certificate repository at each node is small compared to the total number of certificates in the network, while still maintaining a high probability of finding a trust path from one node to another. We assume the process of routing initiates after the certificate graph converges to a steady state, meaning nodes discover paths enough for the entire certificate graph to attain the properties of a **small-world** graph.

4.4.3 Policy variants

We describe two trust policies, based on which the nodes can sign, manage, and distribute PGP certificates. The two policies can be described as follows:

Policy 1: Node d_i completely[2] trusts node d_j means:

1. Node d_i believes that node d_j's public key PK_j is valid and authentic.
2. Node d_i trusts node d_j's decision to sign the public key PK_k of any node d_k. Node d_j is careful not to sign any bogus public key.

Thus, the following condition should hold for authenticating the route discovery process:

$$\forall i, \quad \exists\, (d_i \rightsquigarrow d_j), i < j \leq n$$

Informally, this condition means that any node d_i wanting to authenticate the route from itself to the destination $\{d_i, d_{i+1}, \ldots, d_n\}$ needs to find a way to verify the authenticity of all corresponding public keys $\{PK_i, PK_{i+1}, \ldots, PK_n\}$.

Policy II: This trust policy has an added condition. Node d_i completely trusts node d_j implies the same conditions as listed above, along with the following: node d_i trusts node d_j to honestly aggregate and sign other public keys $PK_{j+1}, \ldots, PK_n$ of nodes

[2]For simplicity, we assign trust either a true or false value. We do not model marginal or partial trust.

$d_{j+1}, \ldots, d_n$. In Policy II, the following condition should hold for authenticating the route discovery process:

$$\forall i, \quad \exists\, (d_i \rightsquigarrow d_{i+1}), 0 \leq i \leq (n-1)$$

Any node d_i wanting to authenticate the route $\{d_i, d_{i+1}, \ldots, d_n\}$ from itself to the destination needs to look up a single node in the certificate graph, that is to verify the authenticity of one public key PK_{i+1}. Node d_{i+1} can sign the aggregate public public key $PK_{(i+1,n)}$ and deliver $(PK_{(i+1,n)}, \texttt{Cert}_{(i+1,n)} = \sigma_{SK_{i+1}}(d_{i+1}, \ldots, d_n, PK_{(i+1,n)}))$ to node d_i, where $\texttt{Cert}_{(i+1,n)}$ denotes the certificate on the aggregate public key $PK_{(i+1,n)}$.

4.5 Analysis

We present a concise theoretical analysis of correctness, security and performance of the multisignature algorithm CLFSR-M.

4.5.1 Correctness

A multisignature scheme constructed following the procedures described in Section 4.3 is correct if an arbitrary multisignature, $\sigma_{(l+1,n)}$, received by node $d_l \in \{d_0, \ldots, d_{n-1}\}$ from node, d_{l+1}, passes the verification procedure MS.V at node d_l under the aggregate public key $PK_{(l+1,n)}$ provided the following conditions hold:

1. Each node $d_i \in d_{l+1}, \ldots, d_n$ chooses and agrees upon the system public parameters $\mathsf{params} = \langle p, Q, f(x), H \rangle$ and honestly executes the key generation algorithm, $\texttt{MS.K}(\mathsf{params}) \rightarrow (PK_i, SK_i)$ and the signature generation algorithm, $\texttt{MS.G}(\mathsf{params}, SK_i, m) \rightarrow \sigma_{(i)}$, where $m = d_0 || \ldots || d_n$.

2. Each node $d_i \in \{d_{l+1}, \ldots, d_{n-1}\}$, honestly executes the multisignature generation algorithm, $\texttt{MS.A}(\mathsf{params}, \sigma_{(i+1,n)}, \sigma_{(i)}) \rightarrow \sigma_{(i,n)}$.

Proposition 4.5.1. *The multisignature scheme* CLFSR-M *has the correctness property.*

Proof: Consider any arbitrary node in the source route: $d_l \in \{d_0, \ldots, d_{n-1}\}$. We show that the multisignature $\sigma_{(l+1,n)}$ generated by node d_{l+1} passes the verification procedure $\texttt{MS.V}(\mathsf{params}, PK_{(l+1,n)}, \sigma_{(l+1,n)}, m) \rightarrow (Valid, Invalid)$ executed at node d_l under the aggregate public key $PK_{(l+1,n)} = \bar{s}_{x_{(l+1,n)}}$ provided the above mentioned conditions hold. In the verification of the multisignature $\sigma_{(l+1,n)}$ using the algorithm MS.V, we observe:

$$A_{(l+1,n)} = f_{v+x_{(l+1,n)}} \leftarrow \mathsf{OP}_2(v, \bar{s}_{x_{(l+1,n)}})$$

where $v = h^{-1} \sum_{i=l+1}^{n}(t_i)$ and $x_{(l+1,n)} = \sum_{i=l+1}^{n}(x_i)$. All nodes use the signing equation: $t_i \equiv k_i r_i - x_i h \mod Q$, where $(l+1) \leq i \leq n$ and $k_{(l+1,n)} = \sum_{i=l+1}^{n}(k_i r_i)$. Thus,

$$\begin{aligned} A_{(l+1,n)} &= f_{\sum_{i=l+1}^{n}(h^{-1}t_i + x_i)} = f_{\sum_{i=l+1}^{n}(h^{-1}k_i r_i)} \\ &= f_{h^{-1}k_{(l+1,n)}} = \mathsf{OP}_1(h^{-1}, f_{k_{(l+1,n)}}) = B_{(l+1,n)} \end{aligned}$$

Thus, the multisignature, $\sigma_{(l+1,n)}$, is valid under $PK_{(l+1,n)}$.

□

Now, we need to show that it is hard for an adversary to deviate from the key pair and signature generation algorithms and still generate a correct signature. However, this is precisely the issue of forgery which we discuss in the following section.

4.5.2 Security

The security of the proposed multisignature protocol `CLFSR-M` is based on the difficulty of solving the trace discrete logarithm (Tr-DL) problem in $\mathbb{F}_q$ [41, 44, 45, 63]. Informally, the trace function $\mathsf{Tr} : \mathbb{F}_{q^3} \mapsto \mathbb{F}_q$ is given as $\mathsf{Tr}(\alpha) = \alpha + \alpha^q + \alpha^{q^2}$. The Tr-DL problem and assumption has been formally defined in Chapter 3.

A **total break** of `CLFSR-M` occurs if, given a public key $\mathsf{PK}_i = \bar{s}_{x_i}$ of an arbitrary node d_i, the adversary is able to compute the corresponding private key $\mathsf{SK}_i = x_i$. In such a case, any node's signature can be forged. However, given $\bar{s}_x$, finding x is equivalent to solving the DL problem in the extension field $\mathbb{F}_{q^3}$ [45]. Using the following lemmas we show that, assuming a **total break** has not occurred, an adversary that can successfully forge a `CLFSR-M` multisignature can also successfully forge a signature in the EG I.4 variant of the ElGamal signature family.

Lemma 4.5.2 (Chakrabarti et al. [26])**.** *The 2-party signature scheme* `CLFSR-S` *is equivalent to EG I.4.*

Proof: The proof of the above lemma is presented in Chapter 3. □

Lemma 4.5.3. *The single-singer signature scheme* `CLFSR-S` *reduces to the proposed multisignature scheme* `CLFSR-M`*.*

Proof: Suppose there exists a PPT forger $\mathcal{F}$, which given public keys $\bar{s}_{x_0}, \bar{s}_{x_1}, \ldots, \bar{s}_{x_n}$, system parameters $\mathsf{params} = \langle p, Q, f(x), H \rangle$ and a message m, successfully forges a

multisignature $\sigma^F_{(0,n)} = (t^F_{(0,n)}, \bar{s}^F_{k_{(0,n)}})$ on $h = H(m)$ with non-negligible probability, i.e., $\sigma^F_{(0,n)}$ passes the verification procedure, MS.V, under the aggregate public key $\bar{s}_{x_{(0,n)}}$.

We show that if an adversary has access to PPT forger $\mathcal{F}$, then given a public key $PK = \bar{s}_x$, params and a message m^F, the adversary can successfully forge a signature $\sigma^F = (\bar{s}^F_{kr}, t^F)$ on $h^F = H(m^F)$ that passes the verification procedure of CLFSR-S under public key PK. Algorithm 2 shows a polynomial-time reduction from the single-signer signature scheme CLFSR-S to the multisignature scheme CLFSR-M.

input : $(\bar{s}_x, m^F)$
output: σ^F

1 Pick n arbitrary long-term private keys $x_0, x_1, \ldots, x_{n-1} \in_R \mathbb{Z}^*_Q$, and compute the corresponding public keys $\bar{s}_{x_0}, \bar{s}_{x_1}, \ldots, \bar{s}_{x_{n-1}}$.
2 Compute $\bar{s}_{x_n} \leftarrow \mathsf{OP}_2(-\sum_{i=0}^{n-1} x_i, \bar{s}_x)$. Thus, $x_n = x - \sum_{i=0}^{n-1} x_i$.
3 $\langle t^F_{(0,n)}, \bar{s}^F_{K_{(0,n)}} \rangle \leftarrow \mathcal{F}(\mathsf{params}, \bar{s}_{x_0}, \bar{s}_{x_1}, \ldots, \bar{s}_{x_n}, m^F)$.
4 Set $\bar{s}^F_{kr} = \bar{s}^F_{k_{(0,n)}}$ and $t^F = t^F_{(0,n)}$.
5 Return $\sigma^F = \langle t^F, \bar{s}^F_{kr} \rangle$ as forged CLFSR-S signature on message m^F under public key $\bar{s}_x$.

Algorithm 2: Reduction of the single-signer CLFSR-S scheme to the proposed multisignature scheme CLFSR-M.

Since the forged multisignature $\langle t^F_{(0,n)}, \bar{s}^F_{K_{(0,n)}} \rangle$ on $h^F = H(m^F)$ passes the verification procedure, MS.V, under the aggregate public key $\bar{s}_{x_{(0,n)}}$, we have:

$$A = \mathsf{OP}_2(t^F_{(0,n)}(h^F)^{-1}, \bar{s}_{x_{(0,n)}}) = \mathsf{OP}_1((h^F)^{-1}, f^F_{K_{(0,n)}}) = B$$

Given that $\bar{s}^F_{kr} = \bar{s}^F_{K_{(0,n)}}$, $t^F = t^F_{(0,n)}$ and $x_n = x - \sum_{i=0}^{n-1} x_i$, we have:

$$A = \mathsf{OP}_2(t^F(h^F)^{-1}, \bar{s}_x) = \mathsf{OP}_1((h^F)^{-1}, f^F_{kr}) = B$$

Thus, $\sigma^F = \langle t^F, \bar{s}^F_{kr} \rangle$ is a valid signature on m^F under public key $\bar{s}_x$ following the verification procedure of CLFSR-S single-signer signature scheme.

□

Using a similar argument presented in Chapter 3, it is immediate that the EG I.4 variant of the ElGamal signature family reduces to the multisignature scheme CLFSR-M.

Next, we present a performance comparison of the multisignature scheme CLFSR-M with existing schemes in the literature.

4.5.3 Cost

In Table 4.1, we show a direct performance comparison of CLFSR-M with three signature aggregation techniques used to instantiate SRDP [59], namely the multisignature by Micali et al. (ASM) [72], the generalized aggregate signature by Boneh et al. (MBLS) [17] and the sequential aggregate signature by Lysyanskaya et al. (SAS) [68]. The term e is the cost of modular exponentiation, the term m is the cost of modular multiplication, h represents the cost of a hash operation, p is the cost of a pairing computation, s is the cost of one scalar multiplication. The term n represents the number of signers, $*$: represents the ephemeral public key propagated during RREQ phase.

Table 4.1: Authenticating sources routes in mobile ad-hoc networks: Cost comparison of CLFSR-M with existing schemes

	SAS	ASM	MBLS	CLFSR-M
Rounds	2	1	1	1
Generation cost	$e+h$	$e+2m+h$	$s+h$	$2\mathsf{OP}_1+h+2m$
Verification cost	$n(h+e)$	$2e+m+h$	$2p+h$	$\mathsf{OP}_1+\mathsf{OP}_2+h+m$
Aggregation cost	–	–	m	OP_2
Signature size (bits)	1024	$320+(160^*)$	160	500
PK size (bits)	2048	2048	766	680

The original construction of Micali et al's multisignature scheme [72] takes three communication rounds; ASM in SRDP requires two rounds for completion, with prior cooperation (though small: one exponentiation and one modular multiplication) among nodes during the RREQ phase, which might be wasteful if the node is not included in the final route. CLFSR-M, uses extremely fast LFSR sequence operations [63, 78] and achieves the best computational efficiency. The public key sizes equivalent to 1024-bit RSA (excluding shared components of the public key) are highest in SAS and ASM, followed by MBLS. CLFSR-M offers the smallest public key size. In ASM, nodes need to additionally propagate the accumulated ephemeral public keys (160 bits) during the RREQ phase, wasting bandwidth. Signature sizes are lowest for MBLS, followed by ASM and CLFSR-M, while SAS incurs the highest sizes.

4.6 Related research

The original design of DSR [57] does not incorporate any security mechanism, making it vulnerable to several attacks [53]. Papadimitratos et al. [77] and Hu et al. [54] independently propose secure on-demand routing protocols, SRP and Ariadne, respectively, to authenticate routes using message authentication codes (MACs). In SRP, intermediate nodes in the route are not authenticated, thus exposing SRP to attacks, including addition and deletion of honest nodes from the route. In Ariadne, route request packets grow in size due to accumulation of MACs. Ariadne also requires loose time synchronization. Kim et al. [59] present a generic DSR authentication protocol, `SRDP`, using MACs and aggregate signature schemes of Micali et al. [72], Boneh et al. [17], and Lysyanskaya et al. [68]. However, Kim et al. do not consider authentication of routes using cached information in the proposed authentication protocol, `SRDP`. Moreover, the signature-based variants of `SRDP` have performance drawbacks that we discuss in Section 4.5.3.

Acs et al. [4] develop a mathematical framework to facilitate the analysis of secure on-demand source routing protocols in mobile ad-hoc networks. Acs et al. present attacks on the `Ariadne` protocol and also describe the construction of a protocol, `endairA`, which is provably secure in the proposed model of security. Bhaskar et al. [10] develop a MAC-based aggregate signature scheme for authenticating DSR. The aggregate signature developed by Bhaskar et al. has the following weaknesses: (1) Only a single designated entity can verify the aggregate signatures; all intermediate nodes from the source node to the destination are incapable of signature verification. (2) The protocol does not support authentication of cached routes. (3) The protocol is MAC-based, and thus does not offer non-repudiation. (4) Also as in any other MAC-based schemes, early detection of invalid MACs by intermediate nodes requires additional key setup overhead.

Capkun et al. [89] analyze PGP trust graphs and show that such graphs exhibit the small-world phenomenon [73, 60]. Informally, a graph is said to exhibit the small-world property if any two nodes in the graph are likely to be connected through a short sequence of intermediate acquaintances. The reader is referred to the noteworthy paper by Kleinberg [60] for an algorithmic perspective to the small-world phenomenon. Capkun et al. [90] also present a PGP-like, self-organized public key management system for ad-hoc networks.

4.7 Summary

Ad hoc networks are characterized by resource-constrained nodes that need to work in a cooperative and self-organized manner to perform all network functions. The problem of finding secure routes in mobile ad-hoc networks is long-standing and has been extensively studied by researchers. Recently, techniques of aggregating signatures have been applied to efficiently authenticate on-demand routing protocols in ad-hoc networks.

In this chapter, we present the first LFSR sequence based multisignature scheme `CLFSR-M` using a EG I.4, variant of the ElGamal signature family. We engineer the multisignature to produce an efficient technique to authenticate route discovery in the dynamic source routing (DSR) protocol. The proposed protocol also works with cached routing information. The multisignature scheme, `CLFSR-M`, is derived from a cubic LFSR sequence-based, single-signer signature scheme, `CLFSR-S` [26], and uses extremely fast LFSR operations, small public keys (smallest among schemes proposed by Kim et al. [59]) and generates a reasonably small multisignature (500 bits). The security of the scheme, `CLFSR-M`, is based on the Tr-DL(DL) Problem in $\mathbb{F}_q(\mathbb{F}_{q^3})$.

Distributing authentic public keys among nodes in a mobile ad-hoc network to bootstrap authentication protocols is a challenging task. Delegating special functions to nodes or assuming the existence of a TTP to distribute certified public keys is paradigmatically unsuitable for ad-hoc networks. We consider a fully distributed mechanism of public key distribution and presented two trust policies, based on PGP, for effective management of individual and aggregate public keys.

Chapter 5

Authenticating path-vector routing protocols

Recently, there has been a significant movement in the networking research community to make the modern Internet a safe and secure medium of communication. Distributed applications such as distributed content management, multi-player games and replicated databases involving a large number of participants represent a large fraction of today's Internet; all such applications are in serious need of authentication mechanisms.

In this chapter, we focus on building an efficient and scalable technique of authentication, tailored for securing path-vector routing protocols used in the Internet. The proposed technique can also be used for other applications like building efficient certificate chains and authenticating distributed and adaptive content-management systems.

5.1 Problem statement

Consider the problem of securing path-vector routing protocols like the Border Gateway Protocol (BGP) [81]. Each node sends each of its neighbors a message containing the node's own identity and information about the destinations reachable through it (in the Internet, destinations correspond to sets of IP addresses, represented as prefixes). When a node receives such a message from a neighbor, it appends its own identity to the path and relays the message to its other neighbors. Thus, at any time the message indicates the sequence of nodes through which it has passed; the protocol is designed so that this sequence corresponds to a path to the indicated destination(s). Because the sequence of node identities in the message defines the path, recipients need to verify that the message indeed traversed that path, in order to detect injection of bogus path information by an adversary. Using traditional (individual) digital signature schemes, each node would sign the message before forwarding it to its neighbors. Upon receiving a message, the recipient first verifies all the signatures, using the individual public keys of the nodes named in the sequence. However, this approach requires resources that grow linearly in the number of nodes in the path: The amount of space used for storing signatures, the storage elements (public keys) needed to verify the signatures, and the computation used for signature verification.

Aggregate signatures have the potential to address a small, but crucial, subset of the bigger problem of securing the whole Internet: Building scalable authentication protocols

in networked systems involving a large number of users. An aggregate signature scheme allows us to combine n signatures from n different signers on n distinct messages into a single signature. This compact signature provides authentication simultaneously on all the n distinct messages for the n corresponding signers. Any entity (not necessarily one of the signers) can verify the aggregate signature, given the corresponding n public keys of the signers. Aggregate signatures come in different flavors, depending on the nature of the application. For example, multisignatures have all signers sign the same message. Sequential aggregate signatures have signers verify, sign and aggregate signatures sequentially. Aggregate signatures have been recently suggested by Zhao et al. [98] as an effective mechanism to authenticate path-vector routing protocols.

Next, we give a broad overview of two potential applications of aggregate signatures: construction of efficient certificate chains, and authentication in distributed content management systems

5.2 Potential applications

5.2.1 Building efficient certificate chains

Informally, a certificate is a signed statement issued by a certification authority (CA) binding a public key to an entity. The certificate contains an issuer (identity and public key), a subject (identity and public key), a validity period (often referred to as the certificate window) and a digital signature by the issuer on the certificate. A root certification authority, $\mathsf{CA_0}$, generates a certificate, $\mathsf{Cert_1}$, for its subordinate, $\mathsf{CA_1}$, which in turn generates certificate, $\mathsf{Cert_2}$, for its own subordinate, $\mathsf{CA_2}$, and this process continues to form a chain of certificates. The subject $\mathsf{CA_i}$ of an arbitrary certificate $\mathsf{Cert_i}$ in the chain is the issuer of the next certificate in the chain $\mathsf{Cert_{i+1}}$.

A certificate authority, say $\mathsf{CA_k}$, at an arbitrary depth k in the certificate chain needs to store all k certificates—including its own—back to $\mathsf{CA_0}$. To check the validity of $\mathsf{CA_k}$'s certificate, all individual signatures $\sigma_0, \ldots, \sigma_\mathsf{k}$ contained in the certificates $\mathsf{Cert_0}, \ldots, \mathsf{Cert_k}$ of the chain need to be verified. More specifically, $\mathsf{CA_k}$ uses the public key of $\mathsf{CA_0}$ to verify the signature σ_0 on certificate $\mathsf{Cert_0}$, the public key of $\mathsf{CA_{i-1}}$ to verify the signature σ_i on certificate $\mathsf{Cert_i}$, and so on.

Computation and communication costs increase linearly with the length of these chains. It would be better if the k individual signatures $\sigma_0, \ldots, \sigma_\mathsf{k}$ in the chain could be compressed into one compact signature $\sigma_{(0,\mathsf{k})}$ that validates all the individual signatures simultaneously. This is precisely the purpose of an aggregate signature. Moreover, this aggregation can

be done sequentially: any intermediate authority $\texttt{CA}_\texttt{i}$ can generate a signature $\sigma_{\texttt{i+1}}$ on a certificate $\texttt{Cert}_{\texttt{i+1}}$, aggregate the signature $\sigma_{\texttt{i+1}}$ with $\sigma_{(0,\texttt{i})}$ contained in its own certificate, and include the new aggregate signature $\sigma_{(0,\texttt{i+1})}$ in the certificate $\texttt{Cert}_{\texttt{i+1}}$. However, an existing attempt by Lysyanskaya et al. [68] to create such a sequential aggregate signature has a drawback: Verification overhead increases linearly with the number of signers.

5.2.2 Authenticating distributed content management systems

Distributed and adaptive multimedia content distribution in the Internet has accelerated the demand for rich multimedia data allowing dynamic content adaptation for a highly personalized user experience. In a multimedia delivery system, the content is composed of meta-data, which specifies the media data and how the content is handled by end-user media players. Secondary content providers (tier-2) can insert or delete a media file from a composition created by the primary provider (tier-1) by manipulating the meta-data without touching the internal content of the files. This type of content adaptation is useful in dynamic insertion of targeted advertisements. However, existing systems cannot handle content adaptation while preserving end-to-end security [88].

Binding usage rules to the meta-data can be somewhat useful but does not prevent malicious corruption (insertion and deletion) of the content by untrusted tier-2 providers. Recently, Suzuki et al. [88] presented a technique that preserves end-to-end authenticity of the original content while allowing adaptation of the content by intermediate providers. Suzuki et al. introduced a multi-hop signature scheme that provides authenticity of usage rules and meta data. A Merkle tree-based signature scheme is used to ensure secure adaptation of content while allowing detection of any rule violation. Placeholders—to specify the positions in the meta-data where a designated entity can insert an additional element—can control insertions, but they do not prevent malicious deletion of tier-2's inserted content itself. One way to prevent this is by having each tier-2 provider sign each of its placeholders. Any placeholder with a missing signature denotes an illegal deletion.

This technique however, leads to a linear increase in the computational and bandwidth requirement as the number of placeholders (or tier-2 providers) increases. To avoid this scenario, aggregate signatures can be used to compress the individual signatures of all tier-2 providers on placeholders and the primary provider's signature on the Merkle tree into a single aggregate signature. Suzuki et al. suggest the use of the generalized aggregate signature by Boneh et al. [17], which requires expensive pairing operations during verification.

5.3 Protocol overview

We propose an enhanced (efficient and scalable) form of aggregate signature, `CLFSR-A`, using cubic LFSR-based public key cryptosystems. The aggregate signature scheme has the following desirable properties:

1. An arbitrary node $E_l \in \mathbb{E} = \{E_1, E_2, \ldots, E_n\}$ does not need to know the order in which the aggregate signature σ_{l-1} was created in order to verify σ_{l-1} that E_l receives from E_{l-1}.

2. The aggregate signature is of constant length (number of elements/bits), independent of the number of nodes in the path.

3. The aggregate signature scheme uses a constant number of sequence operations for signing, aggregation and verification of the signatures at each node.

4. The aggregate signature is scalable with respect to a particular sequence of nodes. Each node in the path needs to store a constant number of storage elements (public keys needed to verify the incoming signature).

With these features, we believe that the proposed aggregate signature scheme has potential to form a crucial building block for applications such as authenticating path-vector routing protocols, building efficient certificate chains, and authenticating distributed and adaptive multimedia content management.

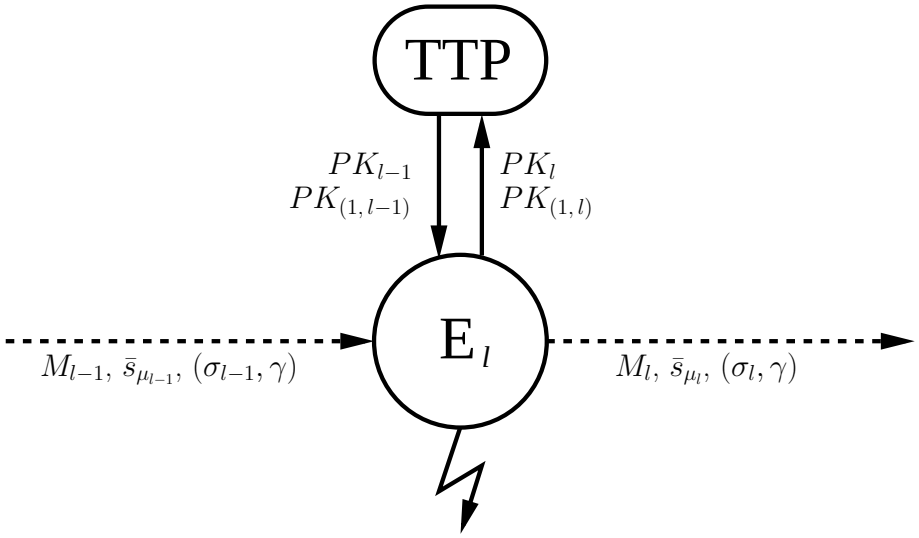
Let the sequence of nodes $\mathbb{E} = \{E_1, \ldots, E_l, \ldots, E_n\}$ represent the participants of `CLFSR-A`. The elements of the signature scheme described here—for example, a node E_l and message M_l, etc.—portray an abstraction of the underlying security protocol. For example, in a secure path-vector routing protocol, the message represents a sequence of nodes—prefixes if applied to BGP—whereas in a certificate chain, the message represents a sequence of public keys and other user data to be certified.

All nodes generate their individual signatures according to the ElGamal-like signature scheme `CLFSR-S′`. The individual signature generated by an arbitrary node $E_l \in \mathbb{E} - \{E_n\}$ on the hashed message $h_l = H(m_l)$ is of the form (f_{k_l}, t_l). Fig. 5.1 shows the basic idea of the proposed signature scheme.

Node E_l receives the following from node E_{l-1}:

1. Message M_{l-1} parsed as $\langle m_1, m_2, \ldots, m_{l-1} \rangle$.

Figure 5.1: An abstraction of the node functionality



2. An aggregate signature $\sigma_{l-1} = \left(f_{(\prod_{i=1}^{l-1} k_i)}, \prod_{i=1}^{l-1} t_i, \prod_{i=1}^{l-1} r_i\right)$, sequentially constructed on hashed messages $h_1 = H(m_1), \ldots, h_{l-1} = H(m_{l-1})$.

3. A quantity $\bar{s}_{\mu_{l-1}}$, where $\mu_{l-1} = \left(\prod_{i=1}^{l-1}(v_i + x_i) - \prod_{i=1}^{l-1} v_i - \prod_{i=1}^{l-1} x_i\right)$. The term $\bar{s}_{\mu_{l-1}}$ helps node E_l in signature verification. The necessity of the helper term $\bar{s}_{\mu_{l-1}}$ in the verification procedure is presented in detail in Algorithm 4.

4. Individual signature γ on $s_{\mu_{l-1}}$.

The term $\prod_{i=1}^{l-1} r_i$ is included in the aggregate signature, σ_{l-1}, since the term $\prod_{i=1}^{l-1} r_i = \prod_{i=1}^{l-1} s_{k_i}$ cannot be obtained from $f_{(\prod_{i=1}^{l-1} k_i)}$, whereas in the `CLFSR-S'` scheme, the term $r = s_k$ can be obtained from f_k.

On receiving the above quantities from node E_{l-1}, node E_l performs the following tasks:

1. Obtain the public key $PK_{l-1} = \bar{s}_{x_{l-1}}$ of the previous node and the aggregate public key, $PK_{1,\ldots,l-1} = \bar{s}_{(\prod_{i=1}^{l-1} x_i)}$, corresponding to the nodes $E_1, \ldots, E_{l-1}$ from the trusted third-party (TTP). The aggregate public key facilitates efficient verification of the aggregate signature pair (σ_{l-1}, γ). We present the details of the construction of the aggregate public key in Section 5.4.

2. Verify the individual signature γ on $s_{\mu_{l-1}}$ using public key PK_{l-1} (of node E_{l-1}) following CLFSR-S′.

3. If signature γ is a valid signature, verify the signature σ_{l-1} using aggregate public key $PK_{1,l-1}$. The signature aggregation algorithm (described in Algorithm 3) is designed so that verification of the aggregate signature, (σ_{l-1}, γ), provides simultaneous verification of individual signatures of the form $(f_{k_1}, t_1), \ldots, (f_{k_{l-1}}, t_{l-1})$ on corresponding hashed messages $h_1, \ldots, h_{l-1}$. Both signatures, σ_{l-1} and γ, are verified using the verification algorithm, described in Algorithm 4.

4. Generate signature $\sigma'_l = (f_{k_l}, t_l)$ on hashed message $h_l = H(m_l)$.

5. Use $\bar{s}_{\mu_{l-1}}$ to construct the helper term $\bar{s}_{\mu_l}$, which node E_{l+1} will use to verify the signature that node E_l sends. Sign s_{μ_l} to create γ.

6. Use σ_{l-1} and σ'_l to construct signature $\sigma_l = \left(f_{(\prod_{i=1}^{l} k_i)}, \prod_{i=1}^{l} t_i, \prod_{i=1}^{l} r_i\right)$.

7. Construct the aggregate public key $PK_{1,l} = \bar{s}_{(\prod_{i=1}^{l} x_i)}$ using the trusted copy of the aggregate public key $PK_{1,l-1} = \bar{s}_{(\prod_{i=1}^{l-1} x_i)}\mathsf{OP}_1(SK_l, PK_{1,l-1})$ and the private key $SK_l = x_l$. Upload $PK_{1,\ldots,l}$ to the TTP.

Node E_l sends messages $m_1, \ldots, m_l, \bar{s}_{\mu_l}$ and the aggregate signature (σ_l, γ) to node E_{l+1}. Nodes following E_l perform the tasks mentioned above to receive and verify the signatures, sign their own messages and construct aggregate signatures. In the following section, we describe the construction of the proposed LFSR-based aggregate signature scheme.

5.4 The proposed LFSR-based aggregate signature scheme

In this section, we first describe a construction of a single signer signature scheme using LFSR sequences. Then, we present the proposed aggregate signature scheme built using the single signer construction.

5.4.1 The single-signer signature scheme

The single-signer signature scheme, CLFSR-S′, is built following the EG I.1 variant of the ElGamal signature family, using primitives from cubic LFSR sequences.

The first two phases of CLFSR-S′, namely, initialization and key generation follow the exact procedures described earlier in Chapter 3. The signing and verification procedures of CLFSR-S′ are based on EG I.1 as opposed to those of CLFSR-S, which follow EG I.4. Fig. 5.2 describes the signing and verification procedures of CLFSR-S′.

Figure 5.2: Authenticating path-vector routing protocols: The CLFSR-S′ single-signer signature scheme

Signature generation	**Signature verification**
1. Randomly choose ephemeral private key $k \in_R \mathbb{Z}_Q^*$ and compute ephemeral public key $f_k \leftarrow \mathsf{OP}_1(k, f(x))$. Denote $r = s_k \mod Q$ as an integer.	1. Compute $v = hr^{-1} \mod Q$ and $u = tr^{-1} \mod Q$. The quantity r can be directly derived from f_k.
2. Compute hash of message $h = H(m)$; Solve for t in the following equation: $h \equiv kt - xr \mod Q$.	2. Compute $A = f_{(v+x)} \leftarrow \mathsf{OP}_2(v, \bar{s}_x)$.
3. Send the signature $\sigma = (f_k, t)$ and the message m to entity B.	3. Compute $B = f_{(uk)} \leftarrow \mathsf{OP}_1(u, f_k)$.
	4. Accept the signature if $A = B$, else reject the signature.

Next, we present the proposed aggregate signature scheme that uses the above algorithm as a building block.

5.4.2 Construction of the aggregation signature

The reader who wants to get a high-level understanding of the aggregate signature scheme may skip this section (containing details of key generation, signature generation, and signature verification) and directly proceed to Section 5.4.5 containing a sample instantiation of the protocol with four nodes.

The process of key generation can be divided into two phases: generation of individual private and public keys and generation of aggregate public keys. Each node $E_l \in \mathbb{E}$ follows the procedure described in the single-signer scheme to generate its individual private, public key pair: long-term private key for node E_l is $\mathsf{SK}_l = x_l$, and the long-term public key of node E_l is $\mathsf{PK}_l = \bar{s}_{x_l} = \{s_{x_l}, s_{x_{l+1}}, s_{x_{l+2}}\}$.

An arbitrary node E_l in the route receives a trusted copy of the aggregate public key $PK_{(1,l-1)} = \bar{s}_{(\prod_{i=1}^{l-1} x_i)}$ (corresponding to users $E_1, \ldots, E_{l-1}$) from the TTP, uses its own private key $\mathsf{SK}_l = x_l$ to create the aggregate public key $PK_{(1,l)} = \bar{s}_{(\prod_{i=1}^{l} x_i)} \leftarrow \mathsf{OP}_1(x_l, f_{(\prod_{i=1}^{l-1} x_i)})$ and uploads it to the TTP. Node E_l only needs the aggregate public key $PK_{(1,l-1)}$ and the public key of the immediate upstream neighbor E_{l-1} and does not need to store the individual public keys of all nodes $E_1, E_2, \ldots, E_{l-1}$.

5.4.3 Aggregate signature generation

Algorithm 3 describes the process of generating aggregate signatures. All nodes in $\mathbb{E}' = \mathbb{E} - \{E_1, E_n\}$ execute the aggregate signature generation algorithm `CLFSR-A.SG` to create an aggregate signature on hashed messages $h_1, \ldots, h_l$ and s_{μ_l}, given the aggregate signature (σ_{l-1}, γ) on $h_1, \ldots, h_{l-1}, s_{\mu_{l-1}}$, the public key $\mathsf{PK}_{l-1} = \bar{s}_{x_{l-1}}$ of node E_{l-1} and the aggregate public key $PK_{(1,l-1)} = \bar{s}_X$, where $X = \prod_{i=1}^{l-1} x_i$). The aggregate signature generation algorithm `CLFSR-A.SG` works as follows:

Step 1: Node E_l generates its ephemeral private, public key pair.

Steps 2 and 3: Node E_l calls the signature verification algorithm `CLFSR-A.SV` twice to verify the aggregate signature (σ_{l-1}, γ) (described in Algorithm 4).

Step 4: Node E_l uses its secret key $\mathsf{SK}_l = x_l$ to generate a signature of the form (f_{k_l}, t_l) on the hashed message $h_l = H(m_l)$.

input : $\sigma_{l-1}, \bar{s}_X(= \bar{s}_{(\prod_{i=1}^{l-1} x_i)}), x_l, \gamma$
output: σ_l.

1 Randomly choose ephemeral private keys $k_l, k'_l \in_R \mathbb{Z}_Q^*$ and compute ephemeral public keys $f_{k_l}, f_{k'_l}$.
2 Compute $v = s_{\mu_{l-1}}(r'_{l-1})^{-1}$ and $u = t'_{l-1}(r'_{l-1})^{-1}$, where $r'_{l-1} = s_{k'_{l-1}} \mod Q$. If CLFSR-A.SV$(\gamma, v, u, \bar{s}_{x_{l-1}}) =$ Invalid then Abort.
3 Compute $V \leftarrow \prod_{i=1}^{l-1} h_i(r_i)^{-1}$, where $h_i = H(m_i)$; $U \leftarrow \prod_{i=1}^{l-1} t_i(r_i)^{-1}$; $\bar{s}_{(X+\mu_{l-1})} \leftarrow \mathsf{OP}_2(\bar{s}_X, \bar{s}_{\mu_{l-1}})$. If CLFSR-A.SV$(\sigma_{l-1}, V, U, \bar{s}_{(X+\mu_{l-1})}) =$ Invalid, then Abort.
4 Let integer $r_l = s_{k_l} \mod Q$; $h_l = H(m_l)$. Solve for t_l in the equation: $h_l \equiv k_l t_l - x_l r_l \mod Q$.
5 Compute $f_{(V+\mu_{l-1})} \leftarrow \mathsf{OP}_2(V, \bar{s}_{\mu_{l-1}})$. Compute $\bar{s}_{\mu_l} \leftarrow \mathsf{OP}_2\big((\mathsf{OP}_1(v_l, f_{(X+\mu_{l-1})}), \mathsf{OP}_1(x_l, f_{(V+\mu_{l-1})})\big)$, where $v_l = h_l r_l^{-1}$.
6 Let integer $r'_l = s_{k'_l}$. Solve for t'_l in the equation: $s_{\mu_l} \equiv k'_l t'_l - x_l r'_l \mod Q$. $\gamma = \big(f_{k'_l}, t'_l\big)$.
7 Compute $\sigma_l = \big(f_{(\prod_{i=1}^{l} k_i)}, \prod_{i=1}^{l} t_i, \prod_{i=1}^{l} r_i\big)$ and send (σ_l, γ) to **node** E_{l+1}.

Algorithm 3: Aggregate Signature Generation CLFSR-A.SG for node $E_l \in \mathbb{E}'$.

Step 5,6, and 7: The aggregate signature on messages $h_1, \ldots, h_l$ and s_{μ_l} is given by the pair (σ_l, γ). The process of signature aggregation includes generation of $\bar{s}_{\mu_l}$, and (σ_l, γ):

Step 5 : Node E_2 generates the first helper term $\bar{s}_{\mu_2}$ from the public key of node E_1, its own secret key, and the terms v_1 and v_2 using the sequence operations OP_1 and OP_2. Each intermediate node $E_l \in \mathbb{E} - \{E_1, E_2, E_n\}$ similarly computes the terms $\bar{s}_{\mu_l}$ using $\bar{s}_{\mu_{l-1}}$, the aggregate public key $PK_{(1,l-1)}$, and the terms v_l, and $\prod_{i=1}^{l-1} v_i$.

Step 6: Node E_l uses its secret key to generate a signature γ, which authenticates s_{μ_l} to the next hop node E_{l+1}.

Step 7: The node E_l generates the aggregate signature σ_l.

Node E_1 signs hashed message $h_1 = H(m_1)$ using its secret key x_1 to create signature σ_1 and sends it to node E_2. The last node, E_n, verifies the aggregate signature (σ_{n-1}, γ) sent by the node E_{n-1}.

5.4.4 Aggregate signature verification

Nodes $\mathbb{E}' = \mathbb{E} - \{E_1\}$ execute the aggregate signature verification algorithm described in Algorithm 4. Node $E_l \in \mathbb{E} - \{E_1, E_2\}$ invokes `CLFSR-A.SV` twice within the `CLFSR-A.SG` algorithm to verify the signature $\gamma = (f_{k'_{l-1}}, t'_{l-1})$ on $s_{\mu_{l-1}}$ and the signature $\sigma_{l-1} = \left(f_{(\prod_{i=1}^{l-1} k_i)}, \prod_{i=1}^{l-1} t_i, \prod_{i=1}^{l-1} r_i\right)$ on $h_1, \ldots, h_{l-1}$, where, $r_i = s_{k_i} \mod Q$. Node E_2 invokes `CLFSR-A.SV` once to verify σ_1. The verification of the aggregate signature (σ_{l-1}, γ) proceeds as follows:

Verification of γ: To verify the signature γ on $s_{\mu_{l-1}}$, node E_l computes (in Step 2 of Algorithm 3) the terms $V = (s_{\mu_{l-1}})(r'_{l-1})^{-1}$, where $r'_{l-1} = s_{k'_{l-1}}$ and $U = (t'_{l-1})(r'_{l-1})^{-1}$. Node E_l calls the algorithm `CLFSR-A.SV` with parameters V, U, γ and E_{l-1}'s public key $PK_{l-1} = \bar{s}_{x_{l-1}}$ that E_l obtains from the TTP. The signature γ under the public key PK_{l-1} is verified by computing the terms A and B as follows:

1. Compute $A = f_{(s_{\mu_{l-1}} {r'_{l-1}}^{-1} + x_{l-1})}$ from V and the public key $\bar{s}_{x_{l-1}}$ using sequence operation OP_2.
2. Compute $B = f_{(t'_{l-1} {r'_{l-1}}^{-1} k'_{l-1})}$ from U and the ephemeral public key $f_{k'_{l-1}}$ (obtained from the signature γ) using sequence operation OP_1.

The signature γ on $s_{\mu_{l-1}}$ is a valid signature under the public key $\mathsf{PK}_{l-1} = \bar{s}_{x_{l-1}}$ if $A = B$.

Verification of σ_{l-1}: The algorithm `CLFSR-A.SV` is used to simultaneously verify the individual signatures $(f_{k_1}, t_1), \ldots, (f_{k_{l-1}}, t_{l-1})$ by checking the equality:

$$f_{\prod_{i=1}^{l-1}(v_i + x_i)} = f_{\prod_{i=1}^{l-1}(u_i k_i)}. \tag{5.1}$$

where, $v_i = h_i r_i^{-1} \mod Q$, $u_i = t_i r_i^{-1} \mod Q$ and $r_i = s_{k_i} \mod Q$. In addition to the aggregate public key $PK_{(1,l-1)}$ and the aggregate signature σ_{l-1}, node E_l needs to know the helper term $\bar{s}_{\mu_{l-1}}$ to compute the terms in Equation 5.1.

To verify signature σ_{l-1}, node E_l computes (in Step 3 of Algorithm 3) the terms $V = \prod_{i=1}^{l-1}(h_i r_i^{-1})$, where $r_i = s_{k_i}$; $U = \prod_{i=1}^{l-1}(t_i r_i^{-1})$; and the term $\bar{s}_{(X+\mu_{l-1})}$ computed from $\bar{s}_{\mu_{l-1}}$ and the aggregate public key $PK_{(1,l-1)}$, using sequence operation OP_2. Node E_l calls the algorithm `CLFSR-A.SV` with parameters V, U, σ_{l-1} and $\bar{s}_{(X+\mu_{l-1})}$ and checks the equality in (5.1) by computing the quantities A and B as follows:

1. Compute $A = f_{\prod_{i=1}^{l-1}(h_i r_i^{-1} + x_i)}$ from V, $\bar{s}_{(X+\mu_{l-1})}$ using sequence operation OP_2.
2. Compute $B = f_{\prod_{i=1}^{l-1}(t_i r_i^{-1} k_i)}$ from U and $f_{(\prod_{i=1}^{l-1} k_i)}$ (given by the signature σ_{l-1}) using sequence operation OP_1.

The aggregate signature σ_{l-1} on hashed messages $h_1, \ldots, h_{l-1}$ is a valid signature under the aggregate public key $PK_{(1,l-1)}$ if $A = B$.

Note that node E_1 does not perform any verification. Node E_2 receives a signature σ_1 of the form (f_{k_1}, t_1) on hashed message $h_1 = H(m_1)$ from node E_1 node; thus, E_2 only runs `CLFSR-A.SV` once to verify σ_1 using $\mathsf{PK}_1 = \bar{s}_{x_1}$.

input : $(\sigma, V, U, \bar{s}_Y)$
output: Valid, Invalid.

1 Compute $A \leftarrow \mathsf{OP}_2(V, \bar{s}_Y)$.
2 Compute $B \leftarrow \mathsf{OP}_1(U, F)$.
/* F denotes the ephemeral public key given by σ. */
3 **if** $(A = B)$ **then**
4 Return Valid.
5 **else**
6 Return Invalid.
7 **end**

Algorithm 4: Aggregate Signature Verification `CLFSR-A.SV` for node $E_l \in \mathbb{E}'$.

5.4.5 A sample instantiation of the protocol

Here, we present a sample run of the aggregate signature scheme with four nodes, E_1 (the source), E_2, E_3 (intermediate nodes), and E_4 (the destination).

The source node E_1 does the following:

1. Choose ephemeral private key $k_1 \in_R \mathbb{Z}_Q^*$; compute ephemeral public key f_{k_1}.
2. Set $r_1 = s_{k_1}$; $h_1 = H(m_1)$. Solve for t_1 in the equation: $h_1 \equiv k_1 t_1 - x_1 r_1 \mod Q$.
3. Send $\sigma_1 = \left(f_{k_1}, t_1\right)$ to node E_2.

Intermediate node E_2 does the following:

1. Choose ephemeral private keys $k_2, k_2' \in_R \mathbb{Z}_Q^*$; compute ephemeral public keys $f_{k_2}, f_{k_2'}$.
2. Compute $V = h_1 r_1^{-1}$ and $U = t_1 r_1^{-1}$.

3. Compute $A = f_{(v_1+x_1)} \leftarrow \mathsf{OP}_2(V, \bar{s}_{x_1})$, where $v_1 = V$. Compute $B = f_{(u_1 k_1)} \leftarrow \mathsf{OP}_1(U, f_{k_1})$, where $u_1 = U$.

4. If $A \neq B$, **Abort**. Compute integer $r_2 = s_{k_2}$; $h_2 = H(m_2)$. Solve for t_2: $h_2 \equiv k_2 t_2 - x_2 r_2 \mod Q$.

5. Compute the helper term $\bar{s}_{\mu_2} \leftarrow \mathsf{OP}_2\big((\mathsf{OP}_1(v_2, f_{x_1}), \mathsf{OP}_1(x_2, f_{v_1})\big)$, where $v_2 = h_2 r_2^{-1}$; $\mu_2 = (v_2 x_1 + x_2 v_1)$.

6. Compute $r_2' = s_{k_2'}$; Solve for t_2': $s_{\mu_2} \equiv k_2' t_2' - x_2 r_2' \mod Q$. $\gamma = (f_{k_2'}, t_2')$.

7. Send $\sigma_2 = \big(f_{k_1 k_2}, t_1 t_2, r_1 r_2\big)$, γ to node E_3.

Intermediate node E_3 does the following:

1. Choose ephemeral private key $k_3, k_3' \in_R \mathbb{Z}_Q^*$; compute ephemeral public key $f_{k_3}, f_{k_3'}$.

2. Compute $v = s_{\mu_2}(r_2')^{-1}$ and $u = t_2'(r_2')^{-1}$. Compute $A \leftarrow \mathsf{OP}_2(v, \bar{s}_{x_2})$. Compute $B \leftarrow \mathsf{OP}_1(u, f_{k_2'})$. If $A \neq B$, **Abort**.

3. Compute $V = h_1 h_2 (r_1 r_2)^{-1}$; $U = t_1 t_2 (r_1 r_2)^{-1}$; $\bar{s}_{(x_1 x_2 + \mu_2)} \leftarrow \mathsf{OP}_2(\bar{s}_{(x_1 x_2)}, \bar{s}_{\mu_2})$.

4. Compute $A = f_{(v_1+x_1)(v_2+x_2)} \leftarrow \mathsf{OP}_2(V, \bar{s}_{(x_1 x_2+\mu_2)})$, where $v_2 = h_2 r_2^{-1}$. Compute $B = f_{(u_1 k_1)(u_2 k_2)} \leftarrow \mathsf{OP}_1(U, f_{k_1 k_2})$, where $u_2 = t_2 r_2^{-1}$.

5. If $A \neq B$, **Abort**. Compute $r_3 = s_{k_3}$; $h_3 = H(m_3)$. Solve for t_3: $h_3 \equiv k_3 t_3 - x_3 r_3 \mod Q$.

6. Compute $f_{(V+\mu_2)} \leftarrow \mathsf{OP}_2(V, \bar{s}_{\mu_2})$ and $\bar{s}_{\mu_3} \leftarrow \mathsf{OP}_2\big((\mathsf{OP}_1(v_3, f_{(x_1 x_2+\mu_2)}), \mathsf{OP}_1(x_3, f_{(V+\mu_2)})\big)$, where $v_3 = h_3 r_3^{-1}$; $\mu_3 = v_3(x_1 x_2 + v_2 x_1 + x_2 v_1) + x_3(v_1 v_2 + v_2 x_1 + x_2 v_1)$. The quantity $f_{(x_1 x_2+\mu_2)}$ can be obtained from step 3.

7. Compute $r_3' = s_{k_3'}$; Solve for t_3': $s_{\mu_3} \equiv k_3' t_3' - x_3 r_3' \mod Q$. $\gamma = (f_{k_3'}, t_3')$.

8. Send $\sigma_3 = \big(f_{k_1 k_2 k_3}, t_1 t_2 t_3, r_1 r_2 r_3\big)$, γ to node E_4.

Destination node E_4 does the following:

1. Compute $v = s_{\mu_3}(r_3')^{-1}$ and $u = t_3'(r_3')^{-1}$. Compute $A \leftarrow \mathsf{OP}_2(v, \bar{s}_{x_3})$. Compute $B \leftarrow \mathsf{OP}_1(u, f_{k_3'})$. If $A \neq B$, **Abort**.

2. Compute $V = h_1 h_2 h_3 (r_1 r_2 r_3)^{-1}$; $U = t_1 t_2 t_3 (r_1 r_2 r_3)^{-1}$; $\bar{s}_{(x_1 x_2 x_3 + \mu_3)} \leftarrow \mathsf{OP}_2(\bar{s}_{(x_1 x_2 x_3)}, \bar{s}_{\mu_3})$.

3. Compute $A = f_{(v_1+x_1)(v_2+x_2)(v_3+x_3)} \leftarrow \mathsf{OP_2}(V, \bar{s}_{(x_1x_2x_3+\mu_3)})$, where $v_3 = h_3 r_3^{-1}$. Compute $B = f_{(u_1k_1)(u_2k_2)(u_3k_3)} \leftarrow \mathsf{OP_1}(U, f_{k_1k_2k_3})$, where $u_3 = t_3 r_3^{-1}$.

4. If $A = B$, the aggregate signature is verified for nodes E_1, E_2 and E_3.

In the following section, we present a thorough analysis (including correctness, security, efficiency, and scalability) of the aggregate signature scheme.

5.5 Analysis

5.5.1 Correctness

The aggregate signature scheme, `CLFSR-A`, satisfies the following correctness property: if every node $E_l \in \mathbb{E}$ uses the system parameters $\mathsf{params} = \langle p, Q, f(x), H \rangle$ and honestly executes the algorithms $(\mathsf{SK}_l, \mathsf{PK}_l) \leftarrow \mathtt{A.KG}(\mathsf{params})$ and $(\sigma_l, \gamma) \leftarrow \mathtt{A.SG}(\sigma_{l-1}, \bar{s}_{\prod_{i=1}^{l-1} x_i}, \mathsf{SK}_l, \gamma)$, then `A.SV` always outputs Valid.

During verification of the aggregate signature σ_{l-1}, node E_l computes the following: $V = \prod_{i=1}^{l-1} h_i(r_i)^{-1} = \prod_{i=1}^{l-1} v_i$, where $v_i = h_i(r_i)^{-1}$ and $U = \prod_{i=1}^{l-1} t_i(r_i)^{-1} = \prod_{i=1}^{l-1} u_i$, where $u_i = t_i(r_i)^{-1}$. Node E_l computes the terms A and B in the following way:

$$A = \mathsf{OP_2}\big(V, \bar{s}_{(\prod_{i=1}^{l-1} x_i)+\mu_{l-1}}\big) = s_{(V+\prod_{i=1}^{l-1} x_i+\mu_{l-1})}$$

Since $\mu_{l-1} = \big(\prod_{i=1}^{l-1}(v_i + x_i) - \prod_{i=1}^{l-1} v_i - \prod_{i=1}^{l-1} x_i\big)$,

$$A = s_{\prod_{i=1}^{l-1}(v_i+x_i)} = s_{\prod_{i=1}^{l-1}(u_ik_i)} = s_{(U\prod_{i=1}^{l-1} k_i)} = \mathsf{OP_1}(U, f_{\prod_{i=1}^{l-1} k_i}) = B$$

Thus, the verification algorithm `CLFSR-A.SV` returns Valid and the `CLFSR-A` is correct.

5.5.2 Security

The security of the scheme, `CLFSR-A`, is based on the difficulty of forging a single-signer signature in the well-known EG I.1 scheme. In this section, we first prove the equivalence of the single-signer scheme `CLFSR-S′` to EG I.1. Next, we present a reduction from the single-signer signature scheme, `CLFSR-S′` to the multi-party aggregate signature scheme, `CLFSR-A`. Informally, we show that if an adversary can successfully forge the aggregate signature by running a probabilistic polynomial time (PPT) forger, then the adversary can forge a signature in the EG I.1 scheme.

Before we present the reductions, we make a few observations about the adversary. An adversary in our scheme `CLFSR-A` is allowed to do the following:

1. Eavesdrop on the channels and record all communication between any two nodes.

2. Run a PPT forger $\mathcal{F}$ and try to forge any signature created by any node E_l.

3. Inject false messages and forged signatures in the channels.

We say that the adversary has successfully forged an aggregate signature σ_l, generated by an arbitrary intermediate node $E_l \in \mathbb{E}' = \mathbb{E} - \{E_1, E_n\}$ on $h_1, \ldots, h_l$ if the forged signature σ_l^F on $h_1, \ldots, h_{l-1}, h^F$ passes the verification of node E_{l+1}, i.e., `A.SV` outputs Valid.

The security of `CLFSR-S'` is based on the difficulty of solving the discrete logarithm (DL) problem in $\mathbb{F}_{q^3}$ or equivalently, the difficulty of solving the trace discrete logarithm (Tr-DL) problem in $\mathbb{F}_q$ [41, 44, 45, 63].

We show that—assuming a **total break** has not occurred—if an adversary can successfully forge an aggregate signature in the `CLFSR-A` scheme, he can successfully forge a signature in EG I.1 variant (reduction in the security sense).

Lemma 5.5.1. *The single-signer signature scheme* `CLFSR-S'` *is equivalent to EG I.1.*

Proof: [$\Longrightarrow$] Given a valid `CLFSR-S'` signature $\sigma = \langle f_k, t\rangle$ on hashed message h under the public key $\bar{s}_x$, we know that $f_{(hr^{-1}+x)} = f_{(tr^{-1}k)}$. Let $\alpha \in \mathbb{F}_{q^3}$ be a root of the characteristic polynomial $f(x)$. By the definition of $f_k(x)$ (given in Chapter 2), the roots of $f_{(hr^{-1}+x)}$ are $\alpha^{(hr^{-1}+x)}, \alpha^{(hr^{-1}+x)q}, \alpha^{(hr^{-1}+x)q^2} \in \mathbb{F}_{q^3}$, the roots of $f_{(tr^{-1}k)}$ are $\alpha^{(tr^{-1}k)}, \alpha^{(tr^{-1}k)q}$, $\alpha^{(tr^{-1}k)q^2} \in \mathbb{F}_{q^3}$. Also, we know from the signing equation of `CLFSR-S'` that $hr^{-1} + x \equiv tr^{-1}k \mod Q$. Thus, the root $\alpha^{(hr^{-1}+x)}$ of $f_{(hr^{-1}+x)}$ is congruent to the root $\alpha^{tr^{-1}k}$ of $f_{(tr^{-1}k)}$ in $\mathbb{F}_q$. We now have $\alpha^{hr^{-1}+x} = \alpha^{tr^{-1}k}$ with $h \equiv kt - xr \mod Q$, which is precisely the signing equation of the EG I.1 scheme. Thus, the `CLFSR-S'` reduces to EG I.1.

[$\Longleftarrow$] Given a valid EG I.1 signature $t = \langle r, t\rangle$ on hashed message h under the public key α^x, we know that $\alpha^{(h^r-1+x)} = \alpha^{(tr^{-1}k)}$. Also, $\alpha^{(h^r-1+x)q} = \alpha^{(tr^{-1}k)q}$ and $\alpha^{(hr^{-1}+x)q^2} = \alpha^{(tr^{-1}k)q^2}$ where, $q = p^2$. We know:

$$\begin{aligned} f_{(hr^{-1}+x)} &= \mathsf{Tr}(\alpha^{hr^{-1}+x}) \\ &= \alpha^{(hr^{-1}+x)} + \alpha^{(hr^{-1}+x)q} + \alpha^{(hr^{-1}+x)q^2} \\ f_{(tr^{-1}k)} &= \mathsf{Tr}(\alpha^{tr^{-1}k}) \\ &= \alpha^{(tr^{-1}k)} + \alpha^{(tr^{-1}k)q} + \alpha^{(tr^{-1}k)q^2} \end{aligned}$$

Thus, $f_{(hr^{-1}+x)} = f_{(tr^{-1}k)}$ with $h \equiv kt - xr \mod Q$, which is the CLFSR-S′ scheme. Thus, the EG I.1 scheme reduces to CLFSR-S′ scheme.

Hence, the equivalence. □

Lemma 5.5.2. *The single-signer signature scheme* CLFSR-S′ *reduces to the proposed aggregate signature scheme* CLFSR-A.

Proof: We say that an aggregate signature σ_l generated by an arbitrary intermediate node $E_l \in \mathbb{E}' = \mathbb{E} - \{E_1, E_n\}$ on messages $m_1, \ldots, m_l$ using the aggregate public key $\bar{s}_X$ is successfully forged if the forged signature σ_l^F on messages $m_1, \ldots, m_{l-1}, m^F$, with $m^F \neq m_l$, passes the verification of the node E_{l+1} under the same aggregate public key $\bar{s}_X$, i.e., CLFSR-A.SV returns Valid.

We show that if an adversary has access to a PPT forger $\mathcal{F}$, with which he forges an aggregate signature in CLFSR-A, then the adversary can successfully forge the signature in CLFSR-S′. Algorithm 5 shows a reduction from the single-signer signature scheme CLFSR-S′ to the multi-party aggregate signature scheme CLFSR-A.

input : $(\bar{s}_x, m^F)$
output: σ^F

1 Choose long-term private key x_1 from $\mathbb{Z}_Q^*$ and ephemeral private key $k_1, \in_R \mathbb{Z}_Q^*$, where $\bar{s}_{x_1} \neq \bar{s}_x$. Choose message m_1; sign hashed message $h_1 = H(m_1)$ following CLFSR-S′ to create signature $\sigma_1 = (f_{k_1}, t_1)$.
2 Choose arbitrary integer $2 < l$
3 **for** $i = 2$ *to* $l - 1$ **do**
4 Choose long-term private key x_i from $\mathbb{Z}_Q^*$ and ephemeral private key $k_i, \in_R \mathbb{Z}_Q^*$, where $\bar{s}_{x_i} \neq \bar{s}_x$.
5 Compute aggregate public key $\bar{s}_X = \bar{s}_{\prod_{j=1}^{i} x_j}$.
6 Choose message m_i and sign hashed message $h_i = H(m_i)$ following CLFSR-S′ and create aggregate signature $\sigma_i = (f_{\prod_{j=1}^{i} k_j}, \prod_{j=1}^{i} t_j, \prod_{j=1}^{i} r_j), \gamma$.
7 **end**
8 $(\sigma^F, \gamma^F) \leftarrow \mathcal{F}(\sigma_{l-1}, \gamma, m_1, \ldots, m_{l-1}, m^F, \bar{s}_x, \bar{s}_X)$.
9 $\sigma^F = (f^F_{(\prod_{i=1}^{l} k_i)}, \prod_{i=1}^{l} t_i^F, \prod_{i=1}^{l} r_i^F)$.
10 Compute $t^F = (\prod_{i=1}^{l} t_i^F)(\prod_{i=1}^{l-1} t_i)^{-1}$; $f_k^F = \mathsf{OP}_1((\prod_{i=1}^{l-1} k_i)^{-1}, f^F_{\prod_{i=1}^{l} k_i})$.
11 Return (f_k^F, t^F) as the forged CLFSR-S′ signature on message m^F under public key $\bar{s}_x$.

Algorithm 5: Reduction of the single-signer CLFSR-S′ scheme to the aggregate signature scheme CLFSR-A.

The adversary chooses a bogus message m^F and has access to a public key $\bar{s}_x$ of an arbitrary intermediate node. To forge a CLFSR- S′ signature on m, the adversary simulates the generation of an aggregate signature $\sigma_{l-1} = \left(f_{(\prod_{i=1}^{l-1} k_i)}, \prod_{i=1}^{l-1} t_i, \prod_{i=1}^{l-1} r_i\right)$ on random messages $m_1, \ldots, m_{l-1}$, where $m_i \neq m$ for $1 \leq i \leq l-1$, using randomly chosen ephemeral private keys $k_1, \ldots, k_{l-1}$ and long-term private keys $x_1, \ldots, x_{l-1}$, such that $\bar{s}_{x_i} \neq \bar{s}_x$ for $1 \leq i \leq l-1$. The aggregate signature σ_{l-1} is valid under the aggregate public key $\bar{s}_X = \bar{s}_{(\prod_{i=1}^{l-1} x_i)}$. The adversary then calls the PPT forger $\mathcal{F}$ with the following inputs:

- Messages $m_1, \ldots, m_{l-1}$
- Aggregate public key $\bar{s}_X$
- Aggregate signature (σ_{l-1}, γ)
- Public key $\bar{s}_x$
- Bogus message m^F.

The PPT forger $\mathcal{F}$ returns a forged aggregate signature (σ_l^F, γ^F) on messages $m_1, \ldots, m_{l-1}$ and m^F under the aggregate public key $\bar{s}_X = \bar{s}_{(\prod_{i=1}^{l} x_i)}$, where $\bar{s}_{x_l} = \bar{s}_x$. The adversary extracts the forged CLFSR-S′ signature (f_k^F, t^F) on m^F as shown in Step 10 of Algorithm 5.

Claim 5.5.1. *The forged signature (f_k^F, t^F) generated by the Algorithm 5 with inputs $(\bar{s}_x, m)$, passes the verification of* CLFSR-S′*, on message m under the public key $\bar{s}_x$.*

Proof: The signature $\sigma_l^F = \left(f^F_{(\prod_{i=1}^{l} k_i)}, \prod_{i=1}^{l} t_i^F, \prod_{i=1}^{l} r_i^F\right), \gamma^F$ on messages $m_1, \ldots, m_{l-1}$ and m^F is a forged aggregate signature under the aggregate public key $\bar{s}_X = \bar{s}_{(\prod_{i=1}^{l} x_i)}$, where $\bar{s}_{x_l} = \bar{s}_x$. Since CLFSR-A.SV returns Valid for the forged signature σ^F, we have:

$$f^F_{\prod_{i=1}^{l}(v_i + x_i)} = f^F_{\prod_{i=1}^{l}(u_i k_i)}$$

With $i = l$ we have,

$$f^F_{(v_l + x_l)} = f^F_{(u_l k_l)}$$

where, $v_l = (H(m^F))(r_l^F)^{-1}$ and $u_l = (t_l^F)(r_l^F)^{-1}$, where $r_l^F = s^F_{k_l} \mod Q$.

From Step 10 in Algorithm 5, we get $t_l^F = t^F$ and $f^F_{k_l} = f_k^F$. □

Thus, if an adversary can forge the aggregate signature, CLFSR-A, then he can forge the single-signer signature CLFSR-S′.

□

Theorem 5.5.3. *EG I.1 reduces to the proposed aggregate signature scheme* `CLFSR-A`.

Proof: The proof of the theorem is immediate from Lemmas 5.5.1, and 5.5.2. □

An effective solution to a network security problem should achieve an adequate balance between comprehensive security and efficiency/scalability. Next, we present a thorough performance analysis (including efficiency and scalability) of the signature scheme `CLFSR-A`.

5.5.3 Efficiency

Table 5.1 shows a performance comparison of `CLFSR-A` with those of Boneh et al. [17] (`BLS`) and Xu et al. [94] (`X`), and the sequential aggregate signature schemes of Lu et al. [67] (`L`), Lysyanskaya et al. [68] (`LMRS`) and Boldyreva et al. [15] (`IBAS`). The term `e` is the cost of one RSA encryption, the term `d` is the cost of one RSA decryption, `s` is the cost of one scalar multiplication, `f` is the cost of hashing onto a GDH group, `p` is the cost of a pairing computation. The term n represents the number of signers who contributed to the aggregate signature, and the table heading Seq. denotes whether the aggregate signature is sequentially generated or not.

In `CLFSR-A`, each node $E_l \in \{E_3, \ldots, E_{n-1}\}$ needs to perform five OP_1 operations and three OP_2 operations in Algorithm 3, and two OP_1 operations and two OP_2 operations in Algorithm 4. Node E_2 needs to perform five OP_1 operations and one OP_2 operation in Algorithm 3 and one OP_1 operation and one OP_2 operation in Algorithm 4. The last node in the sequence, E_n, performs two OP_1 operations and three OP_2 operations in Algorithm 4. The GH variant takes twice the number of operations as XTR.

`CLFSR-A` uses extremely fast LFSR sequence operations [63, 78] and can achieve high computational efficiency. A detailed comparison of modular exponentiations, sequence operations, and pairing computations has been presented in Chapter 3.

5.5.4 Scalability

An arbitrary node generates an aggregate signature σ_l, γ, where the signature $\sigma_l = (f_{\prod_{i=1}^{l} k_i} \in \mathbb{F}_q, \prod_{i=1}^{l} t_i \in \mathbb{Z}_Q^*, \prod_{i=1}^{l} r_i \in \mathbb{F}_q)$ contains 3 elements and the signature $\gamma = (f_{k_l'} \in \mathbb{F}_q, t_l' \in \mathbb{Z}_Q^*)$ contains 2 elements. Thus, the length of the aggregate signature is constant and is independent of the number of signers. TO verify of σ_l, node E_{l+1} needs the aggregate public key $\bar{s}_{\prod_{i=1}^{l} x_i}$. To verify γ, node E_{l+1} needs $\bar{s}_{x_l}$, the public key of E_l. Thus, a total of four elements in the base field $\mathbb{F}_q$ (not six elements due to redundancy of sequence terms [45]) are needed to verify the aggregate signature in `CLFSR-A`.

Table 5.1: Authenticating path-vector routing protocols: Cost comparison of CLFSR-A with existing schemes

	Seq.	**Underlying Problem**	**Signature (Bits)**	**Storage (Bits)**	**Computation**
CLFSR-A	Yes	Tr-DLP	1160	1020	$7\mathsf{OP}_1 + 5\mathsf{OP}_2$
X	No	GDH	$237(n+1)$	$237(n+1)$	$2\mathtt{s}+(2n+1)(\mathtt{p}+\mathtt{f})$
L	Yes	GDH	474	$237*162n$	$(n+4)\mathtt{s}+2\mathtt{p}$
BLS	No	GDH	160	$1024n$	$1\mathtt{s}+(n+1)(\mathtt{p}+\mathtt{f})$
LMRS	Yes	IFP	1024	$1536n$	$1\mathtt{d}+n\mathtt{e}$
IBAS	Yes	GDH	711	$160n$	$\frac{4n+6+n(n-1)}{2}\mathtt{f}$ + $(n+7)\mathtt{s}+6\mathtt{p}$

The sequential, RSA-based, LMRS scheme requires that the verifier know the order in which the aggregate signature was created. The third column in Table 5.1 gives the signature lengths in terms of the number of bits. In the id-based (using bilinear pairings) aggregate signature scheme X, the size of the aggregate signature (on n messages) grows linearly with the number of signers: The aggregate signature contains all n ephemeral public keys required for verification. The fourth column shows the number of storage elements (public keys) needed to verify an aggregate signature on n messages. CLFSR-A is the only protocol that achieves constant (independent of the number of signers) storage for verifying the aggregate signature. The bilinear pairing-based sequential aggregate signature scheme L requires maximum storage for verification due to the large public key size (requiring approximately 5KB each). Each node's private key consists of $k+2$ elements in $\mathbb{Z}_p$, and the public key consists of one element in $\mathbb{G}$ and $k+1$ elements in $\mathbb{G}'$ (both $\mathbb{G}$ and $\mathbb{G}'$ are multiplicative cyclic groups of prime order p), where k is the length of the output of a collision-resistant hash function $H_k : \{0,1\}^* \mapsto \{0,1\}^k$. In order to verify an aggregate signature on n messages, the node E_{l+1} needs to know the public keys of all n signers preceding it. Thus, E_{l+1} needs to store $n(160+2)$ elements or $162n*237$-bits for verification—elements of $\mathbb{G}$ (and $\mathbb{G}'$) require

237-bits for representation [18]. The public key size is variable in the LMRS scheme (as the RSA scheme) depending upon the choice of private-key size. On an average, each public key in the LMRS scheme requires 1536 bits. The last column shows the total computational cost at each node which includes the number of operations required for aggregate signature verification, individual signature generation and signature aggregation.

CLFSR-A is the only scheme that requires a constant number of operations at each entity participating in the aggregate signature scheme. Thus, CLFSR-A is more efficient and scalable than any of the other schemes presented in Table 5.1. Related research on generalized aggregate signatures, sequential aggregate signatures, and multisignatures can found in Chapter 2.

5.6 Summary

Aggregate signatures aim to provide scalable authentication of a large number of users in distributed applications and ideally bound the growth of resources to a constant. We have observed that all previous aggregate signature schemes fall short of fulfilling the goal of limiting the growth of resources to a constant. We have presented an efficient and scalable aggregate signature scheme, CLFSR-A, based on cubic LFSRs tailored for applications like building efficient certificate chains, authenticating distributed and adaptive content management systems and securing path-vector routing protocols. CLFSR-A requires less storage (public keys needed to verify the signature) than previous aggregate signature schemes [15, 17, 67, 68, 94] and offers constant-length signatures and efficient signing, aggregation and verification operations at each node. Each node needs to store a total of 1020 bits (public keys) for verification purposes. Also, CLFSR-A is the only scheme requiring a constant number of operations at each node participating in the aggregate signature scheme.

We believe that our results solve a small but crucial subset of the bigger problem of securing the whole Internet by forming building blocks for authenticating large-scale distributed applications. In the following chapter, we present our research in building an efficient technique using blind signatures to provide accountability in privacy-preserving systems.

Chapter 6

Providing accountability in privacy-preserving systems

6.1 Introduction

The current Internet architecture lacks a network-level **accountability** mechanism—a means to reliably identify an entity that can be held accountable for sending a packet. Undesirable consequences of this omission include inability to attribute attacks of various kinds to higher-level users. The ability to attribute packets to a particular source is clearly desirable from the standpoint of punishing those responsible for launching attacks like the denial-of-service attack. This problem of thwarting denial-of-service attacks has led to recent interest in adding some form of accountability to the network service [9, 97]. The idea of one proposal [9] is that an "accountability provider" certifies each packet by signing it. The signature can be verified by ISPs along the path from sender to receiver, and ultimately by the receiver.

However, concerns about **privacy** complicate the design of any accountability service: The ability to trace communications at user level could hinder or even prevent deployment of such mechanisms. Users may not be comfortable with the notion that a record of their communications exists, even at the level of IP addresses, and even if it will not be revealed except in the case of wrongdoing. (Certainly this information is regarded as highly sensitive by Internet Service Providers [95].)

The tension between accountability and privacy can be ameliorated to some extent through the use of **blind signatures**. Blind signatures are a specialized form of digital signatures where the signature generation involves an interactive protocol executed by an entity (the owner) possessing the message m and another entity (the signer) possessing a long-term private key, also known as the signing key. The owner transforms the message into a "blinded" message m' and sends it to the signer. The signer uses its private signing key to generate a signature on the blinded message $\sigma(m')$ and returns the signature to the owner. The owner makes a transformation on this signature to create $\sigma'(m)$ such that the following two conditions are satisfied:

1. The transformed signature is a valid signature on the original message under the long-term public key of the signer, that is $\sigma(m') = \sigma'(m)$.

2. The signer cannot associate the message, transformed signature pair $(m, \sigma'(m))$ with the owner.

The transformed signature is known as a blind signature. Different constructions of blind signatures have found valuable use in various areas such as E-Cash technology [30], self-certified public keys [79] and E-voting systems [49]. Blind signatures have potential as key building blocks of network security protocols, where an authority needs to vouch for the legitimacy of a message but there is also a need to keep the ownership of the message secret from the authority.

Although blind signature techniques have been known for some time, they are traditionally built with heavyweight cryptographic techniques. The heavy construction of blind signatures limits their utility for the design of protocols where performance is important—for example, where signatures have to be created on the fly, or verified per-packet. We introduce a new blind signature scheme built with primitives from LFSR-based cryptosystems. The performance properties of the scheme make it more suitable than traditional blind signature schemes for use in performance-sensitive network protocols. We do not claim to have solved the difficult problem of accountability completely; however, we believe our new primitive can be a useful building block for such solutions.

6.2 Protocol overview

We first develop a cubic LFSR-based single-signer signature scheme `EGCLFSR` using the EG I.3 variant of the ElGamal family of signatures [51]. We construct our blind signature scheme, `BCLFSR`, based on `EGCLFSR` and using fundamentals of the blind signature originally used in E-Cash systems [21, 30]. The use of LFSR sequences makes the blind signature scheme a good candidate for performance-sensitive network protocols. `BCLFSR` can be useful in providing ISP-level accountability while preserving customer anonymity.

In our scheme, a customer within the domain of an ISP blinds a message $m \rightarrow m'$, and presents m', along with the customer's **personal** credentials, to the accountability server of the ISP. The server uses its long-term secret key to generate a signature on the blinded message, $\sigma(m')$, and returns it to the customer. The customer transforms the signature into a blind signature on the original message $\sigma'(m)$. Given a valid signature on the original message, $\sigma'(m)$ under the ISP's public key, the ISP cannot repudiate signing the message m. This signature provides ISP-level accountability. The use of blind signatures preserves message anonymity, since the ISP cannot associate the (message, blind signature) pair $(m, \sigma'(m))$ to a particular customer. The price paid for this feature is that the ISP

(or third-party accountability service) must have some other means of determining the originating user when presented with a malicious packet. (One possibility is for the ISP to track which of its IP addresses used by which customers.)

The core of our blind signature scheme, BCLFSR, consists of an ElGamal-like signature. An ElGamal-like signature σ on a message m is a tuple consisting of an ephemeral public key, and a parameter t that is the result of solving an ElGamal-like signing equation [51]. Loosely speaking, the ephemeral public key provides some randomness for the signature, and the parameter t serves as the signature relative to that randomness. Informally, in an ElGamal signature-based blind signature scheme, the customer needs to transform both the ephemeral public key and the parameter t to ensure that the (blinded message, signature) pair, (m', σ) and the (original message, blind signature) pair (m, σ') are statistically independent[1]. This feature of unlinkability in blind signatures ensures privacy to the customer.

6.3 Constructing the LFSR-based Blind Signature Scheme

To construct our blind signature scheme, we proceed as in previous chapters: First we develop a single-signer scheme, and then we extend it to a special form of signature, in this case, a blind signature scheme.

6.3.1 The LFSR-based single-signer signature scheme

We present our cubic LFSR-based individual signature scheme, EGCLFSR [26] constructed using the EG I.3 variant of the ElGamal family.

EGCLFSR consists of four phases: initialization, key generation, signature generation and signature verification. During the initialization phase, both entities, i.e., the signer and the verifier, choose and agree on the system public parameters: $\mathsf{params} = \langle p, Q, f(x), H \rangle$, where p, Q and $f(x)$ are as described in Chapter 2 and $H : \{0,1\}^* \mapsto \mathbb{Z}_Q$ is a cryptographic hash function. The signer generates its long-term private and public key pair, $(SK, PK) = (x, \bar{s}_x)$. Fig. 6.1 describes the signature generation and signature verification phases of EGCLFSR.

Next, we present an efficient blind signature scheme using EGCLFSR.

[1]For a formal definition and analysis of unlinkability of blind signatures, the reader is referred to [21, 30, 80]

Figure 6.1: Providing accountability in privacy preserving systems: The EGCLFSR single-signer signature scheme

Signature Generation	**Signature Verification**
1. Randomly choose ephemeral private key $k \in_R \mathbb{Z}_Q^*$ and compute ephemeral public key $f_k \leftarrow \mathsf{OP}_1(k, f)$. Denote $r = s_k \mod Q$ as an integer. 2. Compute hash of message $h = H(m)$; Solve for t in the following equation: $t \equiv kh - xr \mod Q$. 3. Send the signature $\sigma = \langle f_k, t \rangle$ and the message m to verifier.	1. Compute $h = H(m)$, $v = tr^{-1}$ and $u = hr^{-1}$. 2. Compute $A = f_{(v+x)} \leftarrow \mathsf{OP}_2(v, \bar{s}_x)$. 3. Compute $B = f_{(uk)} \leftarrow \mathsf{OP}_1(u, f_k)$. 4. Accept signature if $A = B$, else reject signature.

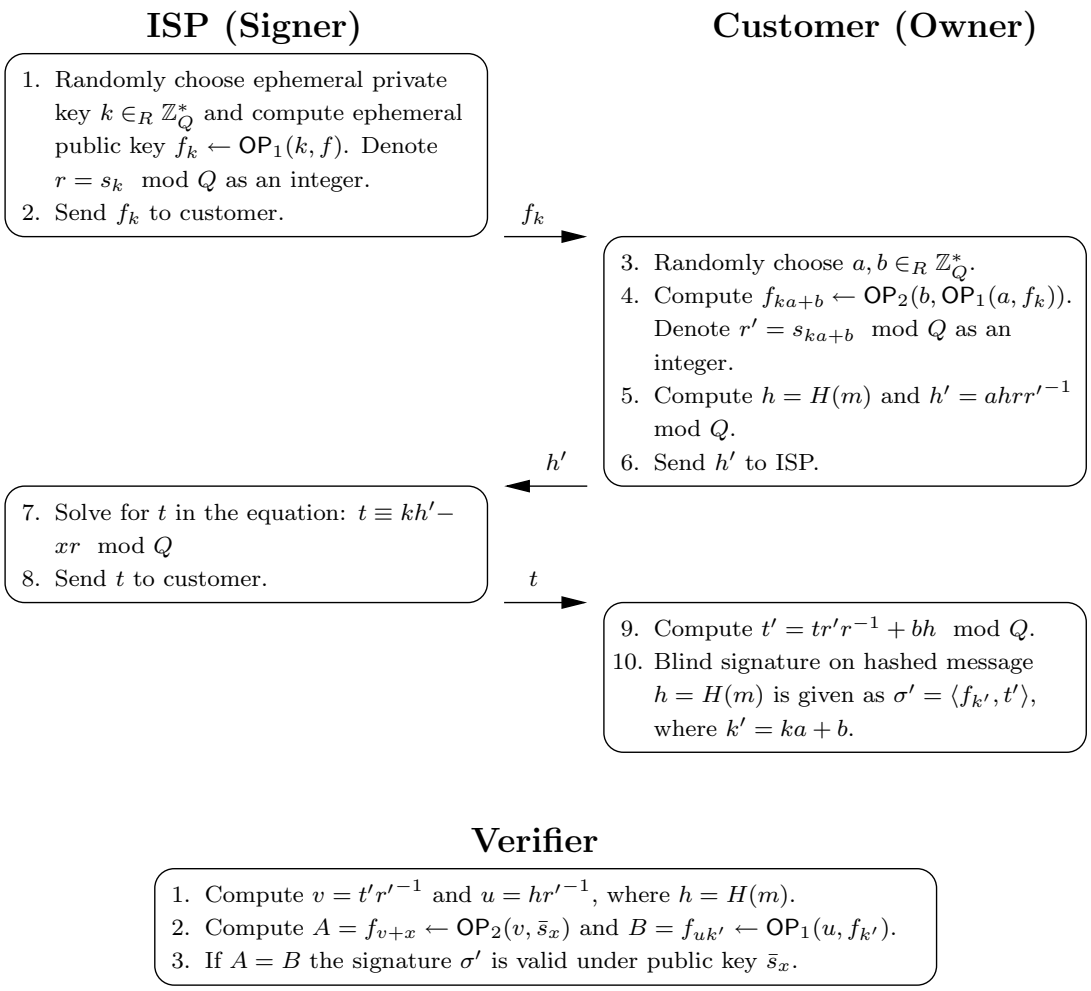
6.3.2 The proposed blind signature

Our new blind signature scheme, BCLFSR, is a cubic LFSR-based instantiation of the EG I.3 variant-based blind signature scheme proposed by Camenisch et al. [21]. Similar to EGCLFSR, during the initialization phase, all entities choose and agree on the system public parameters, $\texttt{params} = \langle p, Q, f(x), H \rangle$. During the key generation phase, the ISP generates its long-term (private, public) key pair, $(SK, PK) = (x, \bar{s}_x)$.

Fig. 6.2 depicts the signature generation and verification phases of the proposed cubic LFSR-based blind signature scheme BCLS. The customer and the ISP interactively generate a blind signature σ' on a hashed message $h = H(m)$ as follows.

1. The ISP generates an ephemeral key pair (k, f_k) and sends the public part, f_k, to the customer.

Figure 6.2: Providing accountability in privacy preserving systems: The blind signature scheme BCLFSR blind signature scheme



2. The customer transforms f_k into the blind signature parameter $f_{k'}$, uses $f_{k'}$ to transform h into the blinded message h' and sends h' to the ISP.

3. The ISP generates a signature $\sigma = \langle f_k, t \rangle$ on the blind hashed message h' using its private key, x, and sends the parameter t to the customer.

4. The customer transforms the parameter t into t' such that the tuple $\sigma' = \langle f_{k'}, t' \rangle$ is a valid signature on hashed message $h = H(m)$ under the ISP's public key, $\bar{s}_x$.

The validity of a blind signature under the ISP's public key implies that the ISP cannot repudiate signing the message, thus, providing ISP-level accountability. Blinding of the message and signature transformations guarantees that when the customer reveals the

(message, blind signature) pair (m, σ') to the verifier, the ISP cannot associate the pair to a particular customer.

6.4 Analysis

In this section, we present a theoretical analysis of correctness, security and performance of the proposed cubic LFSR-based blind signature scheme.

6.4.1 Correctness

The cubic LFSR-based blind signature scheme, `BCLFSR`, is correct if the signature $\sigma' = \langle f_{k'}, t' \rangle$ generated by the customer with the cooperation of the ISP on hashed message h passes the verification under the public key, $\bar{s}_x$, of the ISP, provided:

1. All entities choose and agree upon the system public parameters $\mathsf{params} = \langle p, Q, f(x), H \rangle$.
2. The ISP honestly executes the key generation algorithm of the underlying PKC.
3. The ISP and the customer honestly execute signature generation algorithm of the `BCLFSR` scheme.

Proposition 6.4.1. `BCLFSR` *is correct.*

Proof: In the verification phase of the blind signature scheme, `BCLFSR`, the equality of the terms A and B can be shown as follows:

$$\begin{aligned} A &= f_{v+x} = f_{t'r'^{-1}+x} = f_{(tr'r^{-1}+bh)r'^{-1}+x} \\ &= f_{(kh'-xr)r^{-1}+bhr'^{-1}+x} = f_{(kahrr'^{-1})r^{-1}-x+bhr'^{-1}+x} \\ &= f_{kahr'^{-1}+bhr'^{-1}} = f_{(ka+b)hr'^{-1}} = f_{uk'} = B \end{aligned}$$

Thus, the signature σ' on $h = H(m)$ is valid under public key, $\bar{s}_x$ of the ISP.

□

6.4.2 Security

The security of `BCLFSR` is based on the difficulty of solving the trace discrete logarithm (Tr-DL) problem in $\mathbb{F}_q$ [41, 44, 45, 63]. Informally, the trace function $\mathsf{Tr} : \mathbb{F}_{q^3} \mapsto \mathbb{F}_q$ is given as $\mathsf{Tr}(\alpha) = \alpha + \alpha^q + \alpha^{q^2}$. The Tr-DL problem and assumption has been formally defined in Chapter 3.

Lemma 6.4.2. `EGCLFSR` *is equivalent to EG I.3.*

Proof: [Proof Sketch] $\Rightarrow$: Given a `EGCLFSR` signature $\sigma = \langle f_k, t\rangle$, we know $f_{(tr^{-1}+x)} = f_{(hr^{-1}k)}$. Let α be a root of the irreducible polynomial $f(x)$. Then, we have $\alpha^{tr^{-1}+x} = \alpha^{hr^{-1}k}$ with $t \equiv kh - xr$ which is EG I.3. Thus, the `EGCLFSR` reduces to (in the security sense) EG I.3.

$\Leftarrow$: In EG I.3, $\alpha^{tr^{-1}+x} = \alpha^{hr^{-1}k}$ with $t \equiv kh - xr$. This equality implies $\mathsf{Tr}(\alpha^{tr^{-1}+x}) = \mathsf{Tr}(\alpha^{hr^{-1}k})$. Thus, EG I.3 reduces to (in the security sense) `EGCLFSR`. Hence, the equivalence. □

Theorem 6.4.3. `BCLFSR` *is a blind signature scheme.*

Proof: [Proof Sketch] In the `BCLFSR` scheme, during generation of a blind signature $\sigma' = \langle f_{k'}, t'\rangle$ on a hashed message $h = H(m)$, the parameters known to the ISP (also referred to as the **view** of the ISP) are $\langle h', k, f_k, t\rangle$. `BCLFSR` is blind if the (message, blind signature) pair (m, σ') is statistically independent from the view of the ISP and σ' is a valid signature on $h = H(m)$ under the ISP's public key $\bar{s}_x$ [21].

By Proposition 6.4.1, the signature σ' on the hashed message $h = H(m)$, generated following `BCLFSR`'s signature generation algorithm, is valid under the public key, $\bar{s}_x$. What remains to be shown is that given an arbitrary view of the ISP, $\langle h', k, f_k, t\rangle$, and a valid (message, blind signature) pair (m, σ'), the elements a and b can be uniquely determined. Following the proof of Theorem 2 in [21] we can show that elements a and b can be uniquely determined as: $a = h'h^{-1}r'r^{-1} \mod Q$ and $b = (t' - tr'r^{-1})h^{-1} \mod Q$. Also, since the customer randomly chooses the elements $a, b \in \mathbb{Z}_Q^*$, the statistical independence of the (message, blind signature) pair and ISP's view is immediate [21]. □

6.4.3 Cost

Cubic LFSR-based PKCs [44, 63, 75] use reduced representations of finite field elements. Elements in an extension field $\mathbb{F}_{q^n}$ are represented by their corresponding minimal polynomials with coefficients in the base field $\mathbb{F}_q$. The security of LFSR-based PKCs is based on the difficulty of solving the DL problem in the extension field $\mathbb{F}_{q^n}$. However, all computations are performed in the base field $\mathbb{F}_q$.

Table 6.1 shows direct comparisons of `BCLFSR` with the scheme by Camenisch et al. [21] (`C`), Boldyreva [14] (`B`), Abe [3] (`A`) and Pointcheval et al. [80] (`P`). The term e is the cost of modular exponentiation, s is the cost of one scalar multiplication, p is the cost of one

pairing computation. The computational cost for one $\mathsf{OP}_1 \approx 0.33e$. The lowest known cost for computing a single Tate pairing equals approximately 11110 multiplications in $\mathbb{F}_q$, where q is a 171-bit prime (for a security benchmark of 1024 bits) [6].

The system public parameters of C, P and A are given by the tuple $\mathsf{params} = \langle p, q, \alpha \rangle$, where p and q are 1024 and 160-bit primes, respectively, and α is an element of order q in $\mathbb{Z}_p^*$. Public key sizes include the generators of the underlying groups.

Table 6.1: Providing accountability in privacy preserving systems: Cost comparison of BCLFSR with existing schemes

	C [21]	P [80]	B [14]	A [3]	BCLFSR
Generation cost	$3e$	$7e$	$3s$	$16e$	$3\mathsf{OP}_1$
Verification cost	$2e$	$3e$	$2p$	$8e$	$2\mathsf{OP}_1$
PK size (bits)	2048	2048	766	4096	680
Signature size (bits)	320	1344	160	3008	500
Underlying problem	DLP	DLP	GDH	DLP	Tr-DLP

Given $\alpha \in \mathbb{F}_{p^6}$, where p is a 170-bit prime, computing α^k for any integer k requires approximately $23.4 \log_2 Q$ multiplications, where Q, the order of α, is a 160-bit prime [63]. However, computing the kth sequence term $s_k = \mathsf{Tr}(\alpha^k)$ given f (represented by $\mathsf{Tr}(\alpha)$) using sequence operation OP_1 takes only $8 \log(k \mod Q)$ multiplications which is approximately three times faster than computing α^k, given α [63]. The cost of computing one sequence operation OP_1 (without special hardware) is approximately equal to computing 0.33 exponentiations.

Blind signature generation and verification in BCLFSR are more efficient than in C, P, B and A. BCLFSR also has the smallest public key size. The pairing-based scheme B has the smallest signature, followed by C, BCLFSR, and P; A has the largest signature.

6.5 Related research

Researchers have proposed a number of network-layer mechanisms for identifying sources of malicious packets [66, 86, 96, 97]. Virtually all involve additional operations as part of

forwarding. In some schemes, these operations are performed on every packet, while in others they are performed probabilistically on a small fraction of packets. In some, routers record information about packets forwarded, but in most, routers add to or modify packets in some way to "mark" them. Many of these mechanisms have the property that ability to determine the path followed by a packet is probabilistic; many packets must follow the same path in order for them to be effective. Most also focus on determining the (network-level) source after the fact, and at best offer deterrence rather than prevention.

The recent proposal by Bender et al. [9] takes a different approach. They posit a third-party accountability service that vouches for the legitimacy of its customers. Transit providers require packets to be signed (using public-key cryptography) by such a service in order to forward them. Packets that are not vouched for may not be forwarded. In case of mischief, the victim can take an offending packet to the accountability server, which can identify the perpetrator.

We discuss the background and related research in the area of blind signatures, and cryptosystems based on LFSR sequences in Chapter 2.

6.6 Summary

Blind signatures are useful in protocols that require the origin of a message to be certified in some way, but the actual originator of the message must remain anonymous. Although blind signature techniques have been known for some time, traditionally they are built from heavyweight cryptographic techniques. This limits their utility for the design of protocols where performance is important.

We have presented a new third-order linear feedback shift register (LFSR) sequence-based, single-signer signature scheme, `EGCLFSR`, based on EG I.3. Using `EGCLFSR`, and following fundamentals of a well known blind signature originally used for E-Cash systems, we have presented an efficient blind signature scheme, `BCLFSR`, the first one based on LFSR sequences, which can serve as a protocol building block for privacy-preserving accountability systems. We have analyzed the correctness and security of `BCLFSR` and have also presented a performance (computation and communication costs, storage overhead) comparison of the proposed scheme with previous schemes. The proposed blind signature scheme achieves superior performance and lower storage overhead compared to previous constructions in the literature [3, 14, 21, 80].

Chapter 7

Conclusion and future work

7.1 Significant contributions

This thesis demonstrates how to construct various special forms of digital signatures, such as aggregate signatures, multisignatures, and blind signatures, using primitives from LFSR sequences. We use third-order (cubic) LFSR sequences to construct all security protocols. Our performance analysis shows that for a desired level of security, the proposed signature schemes outperform previous protocols in computation cost, number of communication rounds and storage overhead. To summarize, we have explored the following areas in the domain of network security.

- A common problem faced by large-scale multicast applications, like software distribution, multimedia transmission, and real-time news feeds, is collecting authenticated feedback from the intended recipients. We have designed a technique for combining multiple signed-Acks into a single compact one and have observed that most DL-based signature variants, including the Schnorr variant, cannot be used to produce a single round, fault-tolerant solution to the problem. We proposed using the ElGamal variant, EG I.4, on which we built `CLFSR-MS`, an LFSR-based, efficient and scalable multisignature to solve the signed Ack implosion problem. Its public keys (of size 680 bits) are considerably shorter than keys used in previous protocols addressing the same problem with a security benchmark of 1024 bits. The signature size is around 500 bits, which is the second shortest multisignature among all such protocols.

- The Dynamic Source Routing (DSR) protocol is perhaps the most popular routing protocol used in mobile ad hoc networking environments. DSR, in its original design, is vulnerable to several forms of attack by malicious nodes and thus cannot guarantee authentic source routes. We have focussed on the following problem: In DSR, how can a source node wanting to find a route to a destination be assured of the authenticity of the source route advertised in a received routing packet? We have proposed `CLFSR-M`, a single-round, LFSR-based multisignature scheme to authenticate route discovery information in DSR. The proposed scheme is efficient (500-bit multisignatures and

680-bit public keys), requires no prior cooperation to construct the multisignature, and supports authentication of cached routes. We considered a fully distributed mechanism of public key distribution and presented two PGP-based trust policies for effective management of aggregate public keys in the ad-hoc networking environment.

- We have proposed an efficient and scalable aggregate signature scheme, `CLFSR-A`, tailored for applications like building efficient certificate chains, authenticating distributed and adaptive content-management systems and securing path-vector routing protocols. `CLFSR-A` requires all nodes (that need to verify signatures) to store public keys of size 1020 bits (least among all previous schemes), and generates an aggregate signature of size 1160 bits.

- We have observed that blind signatures can form critical building blocks of privacy-preserving accountability systems, where an authority needs to vouch for the legitimacy of a message but the ownership of the message must be kept secret from the authority. Conventional blind signatures, typically used in E-Cash and E-voting schemes, are built using heavyweight cryptographic primitives. The use of expensive cryptographic operations limits their utility for the design of protocols where performance is important—for example, where signatures have to be created on the fly, or verified per-packet. We have proposed `BCLFSR`, an efficient LFSR-based blind signature that can serve as a protocol building block for performance sensitive, accountability systems.

We construct all our protocols using the XTR-PKC, a cubic-LFSR based PKC. However, all schemes proposed in the dissertation can be seamlessly extended to PKCs based on higher-order LFSR sequences, with minor modifications, depending on the desired security level.

7.2 Continuing and future research

Our continuing and future research can be divided into three parts: (1) Investigating the feasibility of transforming existing DL-based signatures to LFSR-based ones. (2) Conducting experiments to study the rate of discovery of new paths in BGP. (3) Investigating formal mechanisms that produce guarantees of scalability and fault-tolerance in aggregate signature schemes.

7.2.1 Transformation of existing DL-based signatures

We are investigating the feasibility of transforming existing DL-based proxy signatures into efficient ones, by using our techniques based on LFSR sequences. Specifically, we are looking into the following problem: How can we secure systems where nomadic clients need to search for special services or products, negotiate with potential business entities and perform remote operations on behalf of some other client? Proxy signatures have found extensive use in such scenarios of mobile agent-based network systems.

Proxy delegation is a process by which an entity, the delegator, transfers its signing rights and capabilities to another entity, the proxy. Following delegation, the proxy can generate signatures on behalf of the delegator. The messages signed by the proxy conform to a set of business policies, typically embodied in a **warrant**, agreed in advance by the delegator and the proxy. Any entity wanting to verify a proxy signature must check the validity of the proxy signature as well as delegator's agreement on the signed message.

Conventional proxy signatures use traditional public key cryptosystems, and a direct application of such mechanisms would invariably face performance challenges in resource-constrained environments involving mobile agents. To address this problem, we have recently proposed the use of LFSR sequences to construct proxy signatures [29].

We understand that our results have not completely solved the difficult problem of authentication in mobile agent systems, but we strongly believe that our on-going work involving LFSR-based proxy signatures has significant potential to form crucial building blocks for securing performance-sensitive ubiquitous systems. It would be interesting to investigate techniques of transforming conventional DL-based digital signatures into efficient LFSR-based signature schemes that can be used for a variety of performance-sensitive networking contexts.

7.2.2 Path stability of inter domain routing protocols

The Border Gateway Protocol (BGP) remains the de facto interdomain routing protocol on the Internet. The network security and applied cryptographic community—that includes us as well—are trying to develop efficient mechanisms of authenticating BGP path advertisements. However, no cryptographic protocol has yet been deployed in the Internet to guarantee an end-to-end secure solution to inter-domain routing. The search for an efficient cryptographic primitive mitigates the problem of building a practical solution to securing BGP but does not completely solve the the problem of wide-scale deployability across the Internet.

Regardless of the underlying cryptographic primitive being used, all security protocols aiming to optimize cost through signature aggregation are based on two core assumptions: (1) The total number of distinct paths for a prefix advertised by an administrative domain is small, and (2) the advertised paths are stable over time.

The network research community has extensively studied the Internet topology and BGP path stability [19, 33, 39, 46]. However, none of these studies were focussed on the problem of using aggregate signatures to secure routes in BGP. We are conducting extensive experiments with the Internet topology, obtained from various data sources such the Oregon Route Views Project [1], and study the rate of discovery of new paths in BGP.

7.2.3 Techniques for aggregating signatures

Aggregate signatures are crucial in building authentication mechanisms involving a large number of users, such as distributed content-management systems, multi-player games, and secure path-vector routing protocols like BGP. An efficient cryptographic primitive does not automatically result in scalable and fault tolerant network security protocols. Scalability of practical aggregate signature protocols depends on a variety of factors, including the underlying signing equations, the nature of network topology, and the nature of aggregation.

The area of aggregate signatures and applications is relatively new, and the techniques used in aggregation is still quite unexplored. Researchers have resorted to alternative security models, such as the random oracle model [8], to construct proofs of security. We believe that aggregation techniques need more than heuristic arguments to provide guarantees of scalability. It would be interesting to investigate formal mechanisms that can produce guarantees of scalability and fault-tolerance. Such guarantees will include metrics of scalability, such as the number of communication rounds needed for protocol completion, and measures of fault tolerance—whether the protocol needs to be restarted if some (or all) nodes fail to deliver data.

Appendix A

Rudiments of bilinear pairings and Diffie-Hellman problems

The appendix is divided into two distinct parts. The first part discusses the preliminaries of bilinear pairings on certain families of elliptic curves. The second part lists commonly used Diffie-Hellman problems and assumptions that are commonly used in research related to building authentication mechanisms in various networking contexts. The appendix provides the reader with pointers to some terminology and underlying cryptographic mechanisms needed to rigorously compare our results with previous solutions.

A.1 Rudiments of bilinear pairings on elliptic curves

A.1.1 A short note on elliptic curves

Let $E/\mathbb{F}_p$ be an elliptic curve over the finite field $\mathbb{F}_p$ satisfying an equation of the following form:

$$y^2 = x^3 + ax + b \mod p \quad a, b \in \mathbb{F}_p \tag{A.1}$$

where, characteristic $p \geq 3$ and the discriminant of the curve $\triangle = 4a^3 + 27b^2 \neq 0 \mod p$ [12, 47]. The curve $E/\mathbb{F}_p$ is represented by the set of all points $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$ satisfying Equation A.1 together with an extra point $\mathcal{O}$, called the point at infinity. There are explicit formulas for adding two points on the curve [47]. Under the addition rule, the set of points on $E/\mathbb{F}_p$ form an Abelian group, i.e., $Q + R = R + Q$ for all points $Q, R \in E/\mathbb{F}_p$. Scalar multiplication over points on the curve forms the basis of elliptic curve based cryptographic schemes. Given $Q \in E/\mathbb{F}_p$ and integer d, the result of scalar multiplication is obtained by adding Q to itself d times, denoted as $[d]Q$. The number of points on the curve over a finite field, denoted as $\#E/\mathbb{F}_p$, is called the **order of the curve**; it can be calculated by Schoof's algorithm [85] in polynomial time. The order of the curve is given by $\#E/\mathbb{F}_p = p + 1 - t$, where t is the trace of Frobenius at p [12, 47]. The curve $E/\mathbb{F}_p$ is called **supersingular** if the characteristic divides the trace of Frobenius, i.e., $p \mid t$.

A.1.2 Bilinear pairings

Recently, bilinear pairings have emerged as a popular tool for construction of secure and efficient identity-based schemes dealing with encryption, key agreements and digital

signatures [13]. We treat bilinear pairings largely as abstract mappings and do not digress into the mathematics needed to construct such pairings. In this section, we enumerate the attributes of such mappings, focussing on the usage in our key agreement protocol, although the practicality of any real-life security mechanism demands a closer look at the details of the selected pairing mechanism.

Let $\mathbb{G}_0 = (\langle P_0 \rangle, r, +)$ denote an additive group of large prime order r. Let $\mathbb{G}_1 = (\langle P_1 \rangle, r, *)$ denote a multiplicative group of the same large prime order r. Define a modified pairing as: $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$. The pairing is considered admissible [17] if it has the following attributes:

1. Bilinearity: For all $P, Q \in \mathbb{G}_0$ and for all $a, b \in \mathbb{Z}$, $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
2. Non-degeneracy: The pairing does not map all pairs in $\mathbb{G}_0$ to the identity element of $\mathbb{G}_1$, i.e., $\exists P \in \mathbb{G}_0 : \hat{e}(P, P) \neq 1$.
3. Computability: For all $P, Q \in \mathbb{G}_0$, there exists an algorithm that computes $\hat{e}(P, Q)$ in polynomial time.

Admissible bilinear pairings used in modern cryptographic mechanisms are implemented by the modified Weil pairing or the modified Tate pairing [12].

A.1.3 The MOV/Frey-Rück attack

The discussion of bilinear maps and its application would be incomplete without a word on the MOV/Frey-Rück Attack. Even though there exist algorithms, like the Index Calculus Algorithm, to solve the discrete logarithm problem (DLP) over finite fields in probabilistic sub-exponential time under certain conditions, the elliptic curve discrete logarithm problem (ECDLP) over certain elliptic curves cannot be solved in sub-exponential time; the Index Calculus Algorithm does not work with specific elliptic curves [12, 47]. Menezes et al. [69] demonstrated a reduction of an instance of the discrete logarithm problem in an elliptic curve to the logarithm problem in a finite field, thus, providing a probabilistic sub-exponential time algorithm for the ECDLP. Menezes et al. [69] achieve this reduction using Weil pairings, while Frey-Rück [36] used the Tate pairing. Algorithms that solve the discrete logarithm problem over finite fields $\mathbb{F}_{p^k}$ have running time sub-exponential in the order of the field p^k. Menezes et al. [69] observe that supersingular elliptic curves have small values of k and hence are vulnerable to this attack. Hence, pairing-based cryptosystems constructed using supersingular elliptic curves must choose the security parameters in such a way that the discrete logarithms on both $\mathbb{G}_0$ and $\mathbb{G}_1$ are hard.

A.2 The Diffie-Hellman family of problems and assumptions

Applied cryptographic protocols are based on several well-known mathematical assumptions, such as the Computational Diffie-Hellman assumption. We describe some of these common assumptions that we use (and research we refer to) throughout this dissertation.

Computational Diffie-Hellman (CDH) parameter generator A CDH parameter generator, $\mathsf{Gen}_{\mathsf{CDH}}$, is a probabilistic polynomial time (PPT) algorithm with the following description: (1) $\mathsf{Gen}_{\mathsf{CDH}}$ takes a security parameter λ as input, (2) runs in polynomial time in λ, and (3) outputs the description of $\mathbb{G}_0 = (\langle P_0 \rangle, r, +)$.

CDH problem in $\mathbb{G}_0$ Given $(P_0, [a]P_0, [b]P_0) \in \mathbb{G}_0^3$ and $a, b \xleftarrow{R} \mathbb{Z}_r^*$, compute $[ab]P_0$. Let $\mathcal{A}$ be a PPT algorithm that solves the CDH problem. Define the advantage of the CDH-solver $\mathcal{A}$ as:

$$\mathsf{Adv}_{\mathcal{A},\mathbb{G}_0}^{CDH}(t) = Pr\left[\mathcal{A}(\mathbb{G}_0, P_0, [a]P_0, [b]P_0, [c]P_0) = [ab]P_0 \mid \mathbb{G}_0 \leftarrow \mathsf{Gen}_{\mathsf{CDH}}, a, b \xleftarrow{R} \mathbb{Z}_r^*\right]$$

where the probability is over the random choice of a, b in $\mathbb{Z}_r^*$, the random choice of the generator P_0 of $\mathbb{G}_0$ and the random bits of $\mathcal{A}$.

CDH assumption The CDH parameter generator, $\mathsf{Gen}_{\mathsf{CDH}}$, satisfies the CDH assumption, if for every $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A},\mathbb{G}_0}^{CDH}$ is negligible.

Bilinear Diffie-Hellman (BDH) parameter generator A BDH parameter generator $\mathsf{Gen}_{\mathsf{BDH}}$ is a PPT algorithm with the following description: (1) $\mathsf{Gen}_{\mathsf{BDH}}$ takes a security parameter λ as input, (2) runs in polynomial time in λ, and (3) outputs the description of two groups, $\mathbb{G}_0 = (\langle P_0 \rangle, r, +)$ and $\mathbb{G}_1 = (\langle P_1 \rangle, r, *)$, and an admissible bilinear pairing $\hat{e} : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$.

BDH Problem in $(\mathbb{G}_0, \mathbb{G}_1, \hat{e})$ Given $(P_0, [a]P_0, [b]P_0, [c]P_0) \in \mathbb{G}_0^4$ and $a, b, c \xleftarrow{R} \mathbb{Z}_r^*$, compute $\hat{e}(P_0, P_0)^{abc} \in \mathbb{G}_1$. Let $\mathcal{A}$ be a PPT algorithm that solves the BDH Problem. The advantage of the BDH-solver $\mathcal{A}$ can be defined as:

$$\mathsf{Adv}_{\mathcal{A},\mathbb{G}_0,\mathbb{G}_1,\hat{e}}^{BDH}(t) = Pr\big[\quad \mathcal{A}(\mathbb{G}_0, \mathbb{G}_1, P_0, [a]P_0, [b]P_0, [c]P_0) = \hat{e}(P_0, P_0)^{abc} \mid (\mathbb{G}_0, \mathbb{G}_1, \hat{e}) \leftarrow \mathsf{Gen}_{\mathsf{BDH}}, a, b, c \xleftarrow{R} \mathbb{Z}_r^*\big]$$

BDH assumption The BDH parameter generator, $\mathsf{Gen}_{\mathsf{BDH}}$, satisfies the BDH assumption, if for every $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A},\mathbb{G}_0}^{BDH}$ is negligible.

Decisional Bilinear Diffie-Hellman (DBDH) Problem in $(\mathbb{G}_0, \mathbb{G}_1, \hat{e})$ Given $a, b, c \xleftarrow{R} \mathbb{Z}_r^*$ and $x \xleftarrow{R} \mathbb{G}_1$, the problem for a PPT distinguisher D running in time t, is to distinguish between the distributions: $(P_0, aP_0, bP_0, cP_0, \hat{e}(P_0, P_0)^{abc})$ and $(P_0, aP_0, bP_0, cP_0, x)$. Let $\mathcal{A}$ be a PPT distinguisher that runs solves the DBDH Problem. Define the advantage of the DBDH-solver $\mathcal{A}$ as:

$$\begin{aligned}\mathsf{Adv}^{DBDH}_{\mathcal{D},\mathbb{G}_0,\mathbb{G}_1,\hat{e}}(t) = \big| Pr \quad \big[\quad & \mathcal{A}(\mathbb{G}_0, \mathbb{G}_1, P_0, [a]P_0, [b]P_0, [c]P_0, \hat{e}(P_0, P_0)^{abc}) = 1 \mid \\ & (\mathbb{G}_0, \mathbb{G}_1, \hat{e}) \leftarrow \mathsf{Gen}_{\mathsf{BDH}}, a, b, c \xleftarrow{R} \mathbb{Z}_r^* \big] \\ & - Pr\big[\mathcal{A}(\mathbb{G}_0, \mathbb{G}_1, P_0, [a]P_0, [b]P_0, [c]P_0, x) = 1 \mid \\ & (\mathbb{G}_0, \mathbb{G}_1, \hat{e}) \leftarrow \mathsf{Gen}_{\mathsf{BDH}}, a, b, c \xleftarrow{R} \mathbb{Z}_r^*, x \xleftarrow{R} \mathbb{G}_1 \big] \big|\end{aligned}$$

DBDH assumption For every $\mathcal{D}$, $\mathsf{Adv}^{DBDH}_{\mathcal{D},\mathbb{G}_0,\mathbb{G}_1,\hat{e}}$ is negligible.

Bibliography

[1] University of Oregon Route Views Project. Available at http://www.routeviews.org.

[2] Cristina Abad, William Yurcik, and Roy H. Campbell. A survey and comparison of end-system overlay multicast solutions suitable for network centric warfare. In Raja Suresh, editor, *Proceedings of SPIE: Battlespace Digitization and Network-Centric Systems IV*, volume 5441, pages 215–226. SPIE, 2004.

[3] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 136–151, London, UK, 2001. Springer-Verlag.

[4] Gergely Acs, Levente Buttyan, and Istvan Vajda. Provably Secure On-Demand Source Routing in Mobile Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 5(11):1533–1546, 2006.

[5] Suman Banerjee, Christopher Kommareddy, Koushik Kar, Bobby Bhattacharjee, and Samir Khuller. OMNI: an efficient overlay multicast infrastructure for real-time applications. *Computer Networks*, 50(6):826–841, 2006.

[6] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. On the selection of pairing-friendly groups. In Mitsuru Matsui and Robert J. Zuccherato, editors, *Proceedings of SAC: Tenth Annual International Workshop on Selected Areas in Cryptography, Revised Papers*, volume 3006 of *LNCS*, pages 17–25. Springer, 2003.

[7] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT*, pages 139–155, 2000.

[8] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of CCS: First ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[9] Adam Bender, Neil Spring, Dave Levin, and Bobby Bhattacharjee. Accountability as a service. In *Proceedings of SRUTI: Third conference on Steps to Reducing Unwanted Traffic on the Internet*, Berkeley, CA, USA, June 2007. USENIX Association.

[10] Raghav Bhaskar, Javier Herranz, and Fabien Laguillaumie. Efficient authentication for reactive routing protocols. In *AINA '06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 2 (AINA'06)*, pages 57–61, Washington, DC, USA, 2006. IEEE Computer Society.

[11] John Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. Cryptology ePrint Archive, Report 2005/210, 2005.

[12] Ian F. Blake, G. Seroussi, and N. P. Smart. *Elliptic curves in cryptography*. Cambridge University Press, New York, NY, USA, July 1999.

[13] Ian F. Blake, G. Seroussi, and N. P. Smart. *Advances in Elliptic Curve Cryptography*. Cambridge University Press, New York, NY, USA, May 2005.

[14] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In Yvo Desmedt, editor, *Proceedings of PKC: Sixth International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *LNCS*, pages 31–46. Springer, 2003.

[15] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In Sabrina De Capitani di Vimercati and Paul Syverson, editors, *Proceedings of CCS '07: the 14th ACM conference on Computer and communications security*, pages 276–285. ACM Press, 2007.

[16] Jean-Chrysostome Bolot, Thierry Turletti, and Ian Wakeman. Scalable feedback control for multicast video distribution in the internet. In *Proceedings of SIGCOMM: Conference on Communications Architectures, Protocols and Applications*, pages 58–67. ACM Press, 1994.

[17] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Proceedings of EUROCRYPT: International Conference on the Theory and Applications of Cryptographic Techniques*, volume 2656 of *LNCS*, pages 416–432. Springer, 2003.

[18] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[19] Kevin Butler, Patrick McDaniel, and William Aiello. Optimizing bgp security by exploiting path stability. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 298–310, New York, NY, USA, 2006. ACM.

[20] Kenneth L. Calvert, James Griffioen, Billy C. Mullins, Amit Sehgal, and Su Wen. Concast: Design and implementation of an active network service. *IEEE Journal on Selected Areas in Communications (JSAC)*, 19(3):426–437, March 2001.

[21] Jan Camenisch, Jean-Marc Piveteau, and Markus Stadler. Blind signatures based on the discrete logarithm problem. In Alfredo De Santis, editor, *Proceedings of EUROCRYPT: Workshop on the Theory and Application of Cryptographic Techniques, May 9-12, Perugia, Italy*, volume 950 of *LNCS*, pages 428–432. Springer, 1994.

[22] Claude Castelluccia, Stanislaw Jarecki, Jihye Kim, and Gene Tsudik. Secure acknowledgment aggregation and multisignatures with limited robustness. *Computer Networks*, 50(10):1639–1652, 2006.

[23] Saikat Chakrabarti, Santosh Chandrasekhar, Kenneth L. Calvert, and Mukesh Singhal. Efficient blind signatures for accountability. In *Proceedings of NPSec: The Third Workshop on Secure Network Protocols, Beijing, China*, October 2007.

[24] Saikat Chakrabarti, Santosh Chandrasekhar, Mukesh Singhal, and Kenneth L. Calvert. An Efficient and Scalable Quasi-Aggregate Signature Scheme Based on LFSR Sequences. Technical report TR467-06, University of Kentucky, July 2006.

[25] Saikat Chakrabarti, Santosh Chandrasekhar, Mukesh Singhal, and Kenneth L. Calvert. Authenticating DSR using a novel multisignature scheme based on cubic LFSR sequences. In Frank Stajano, Catherine Meadows, and Srdjan Capkun, editors, *Proceedings of ESAS: The Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks*, volume 4572 of *LNCS*, pages 156–171. Springer, 2007.

[26] Saikat Chakrabarti, Santosh Chandrasekhar, Mukesh Singhal, and Kenneth L. Calvert. Authenticating DSR using a novel multisignature scheme based on cubic LFSR sequences. In *Proceedings of The Fourth European Workshop on Security and*

Privacy in Ad hoc and Sensor Networks (ESAS), volume 4572 of *LNCS*, pages 156–171. Springer, 2007.

[27] Saikat Chakrabarti, Santosh Chandrasekhar, Mukesh Singhal, and Kenneth L. Calvert. Authenticating feedback in multicast applications using a novel multisignature scheme based on cubic LFSR sequences. *AINAW: 21st International Conference on Advanced Information Networking and Applications Workshops*, 1:607–613, 2007.

[28] Saikat Chakrabarti, Venkata C. Giruka, and Mukesh Singhal. *Security in Distributed, Grid, and Pervasive Computing, Edited by Prof. Yang Xiao.* Auerbach Publications, CRC Press, 2006.

[29] Santosh Chandrasekhar, Saikat Chakrabarti, and Mukesh Singhal. Efficient proxy signatures for ubiquitous computing. *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, 0:106–113, 2008.

[30] David Chaum. Blind signatures for untraceable payments. In *Proceedings of CRYPTO: Second Annual International Cryptology Conference*, pages 199–203, 1982.

[31] Stephen E. Deering. Multicast routing in internetworks and extended LANs. In *Proceedings of SIGCOMM: ACM Symposium on Communications Architectures and Protocols*, pages 55–64. ACM Press, 1988.

[32] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[33] Anja Feldmann, Olaf Maennel, Z. Morley Mao, Arthur Berger, and Bruce Maggs. Locating internet routing instabilities. *SIGCOMM Comput. Commun. Rev.*, 34(4):205–218, 2004.

[34] FIPS. Digital Signature Standard (DSS), January 2000.

[35] James A. Freebersyser and Barry Leiner. A dod perspective on mobile ad hoc networks. pages 29–51, 2001.

[36] Gerhard Frey and Hans-Georg Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62(206):865–874, April 1994.

[37] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO: Fourth Annual International Cryptology Conference*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[38] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[39] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 307–317, New York, NY, USA, 2000. ACM.

[40] Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Proceedings of PKC'06: 9th International Conference on Theory and Practice of Public-Key Cryptography, April 24-26, New York, NY, USA*, volume 3958 of *LNCS*, pages 257–273. Springer, 2006.

[41] Kenneth J. Giuliani and Guang Gong. New LFSR-based cryptosystems and the trace discrete log problem (trace-DLP). In Tor Helleseth, Dilip V. Sarwate, Hong-Yeop Song, and Kyeongcheol Yang, editors, *Proceedings of SETA: Third International Conference on Sequences and Their Applications*, volume 3486 of *LNCS*, pages 298–312. Springer, 2004.

[42] Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography. Summer Course "Cryptography and Computer Security" at MIT, 1996–1999, 1999.

[43] Solomon W. Golomb and Guang Gong. *Signal Design for Good Correlation: For Wireless Communication, Cryptography, and Radar*. Cambridge University Press, New York, NY, USA, 2004.

[44] Guang Gong and Lein Harn. Public-key cryptosystems based on cubic finite field extensions. *IEEE Transactions on Information Theory*, 45(7):2601–2605, 1999.

[45] Guang Gong, Lein Harn, and Huapeng Wu. The GH public-key cryptosystem. In Serge Vaudenay and Amr M. Youssef, editors, *Proceedings of SAC: Eighth Annual International Workshop on Selected Areas in Cryptography, Revised Papers*, volume 2259 of *LNCS*, pages 284–300. Springer, 2001.

[46] Timothy G. Griffin, F. Bruce Shepherd, and Gordon Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions On Networking*, 10(2):232–243, 2002.

[47] Darrel Hankerson, Alfred Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer, New York, 2004.

[48] Lein Harn. New digital signature scheme based on discrete logarithm. *Electronics Letters*, 30(5):396–398, Mar 1994.

[49] Patrick Horster, Markus Michels, and Holger Petersen. Blind multisignature schemes and their relevance to electronic voting. In *Proceedings of ACSAC: 11th Annual Computer Security Applications Conference*, pages 149–155. IEEE Press, 1995.

[50] Patrick Horster, Markus Michels, and Holger Petersen. Meta-Multisignature schemes based on the discrete logarithm problem. In *Proceedings of IFIP/SEC: 11th. International Conference on Information Security*, pages 128–141. Chapman and Hall, 1995.

[51] Patrick Horster, Holger Petersen, and Markus Michels. Meta-elgamal signature schemes. In *Proceedings of CCS: Second ACM Conference on Computer and Communications Security*, pages 96–107. ACM Press, 1994.

[52] Yih-Chun Hu and David B. Johnson. Caching strategies in on-demand routing protocols for wireless ad hoc networks. In *Proceedings of MobiCom: 6th Annual international conference on Mobile computing and networking*, pages 231–242, New York, NY, USA, 2000. ACM.

[53] Yih-Chun Hu and Adrian Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy*, 2(3):28–39, 2004.

[54] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. *Wireless Networks*, 11(1-2):21–38, 2005.

[55] Ken'ichi Itakura, Hiroshi Nakamura, and Kisaburo Nakazawa. A public-key cryptosystem suitable for digital multisignatures. NEC Research and Development, October 1983.

[56] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and James O'Toole Jr. Overcast: Reliable multicasting with an overlay network. In

Proceedings of OSDI: Fourth Symposium on Operating System Design and Implementation, pages 197–212. USENIX Association, 2000.

[57] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The Dynamic Souce Routing Protocol for Mobile Ad Hoc Networks (DSR). Internet draft draft-ietf-manet-dsr-10, IETF MANET Working Group, July 2004.

[58] J. Jubin and J. D. Tornow. The DARPA packet radio network protocols. In *IEEE Proceedings*. IEEE, 1987.

[59] Jihye Kim and Gene Tsudik. SRDP: Securing route discovery in DSR. In *Proceedings of MobiQuitous: Second Annual International Conference on Mobile and Ubiquitous Systems, 17-21 July, San Diego, CA, USA*, pages 247–260. IEEE Computer Society, 2005.

[60] Jon Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of STOC: 32nd. Annual ACM symposium on Theory of computing*, pages 163–170, New York, NY, USA, 2000. ACM.

[61] Neal Koblitz and Alfred Menezes. Another Look at "Provable Security". II. In Rana Barua and Tanja Lange, editors, *Proceedings of INDOCRYPT: Seventh International Conference on Cryptology*, volume 4329 of *LNCS*, pages 148–175. Springer, 2006.

[62] J. Kong, P. Zerfos, H. Luo, and S. Lu. Providing robust and ubiquitous security support for mobile ad hoc networks. In *Proceedings of ICNP: Ninth International Conference on Network Protocols*, page 251, Washington, DC, USA, 2001. IEEE Computer Society.

[63] Arjen K. Lenstra and Eric R. Verheul. The XTR Public Key System. In Mihir Bellare, editor, *Proceedings of CRYPTO: 12th Annual International Cryptology Conference on Advances in Cryptology*, volume 1880 of *LNCS*, pages 1–19. Springer, 2000.

[64] Brian Neil Levine and Jose Joaquin Garcia-Luna-Aceves. A comparison of reliable multicast protocols. *Multimedia Systems*, 6(5):334–348, 1998.

[65] Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications*. Cambridge University Press, New York, NY, USA, 1986.

[66] Xin Liu, Xiaowei Yang, David Wetherall, and Thomas Anderson. Efficient and secure source authentication with packet passports. In *SRUTI'06: Proceedings of*

the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet, pages 2–2, Berkeley, CA, USA, 2006. USENIX Association.

[67] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *Proceedings of EUROCRYPT: 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. May 28 - June 1, Petersburg, Russia*, volume 4004 of *LNCS*, pages 465–485. Springer, 2006.

[68] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan Camenisch, editors, *Proceedings of EUROCRYPT: International Conference on the Theory and Applications of Cryptographic Techniques*, volume 3027 of *LNCS*, pages 74–90. Springer, 2004.

[69] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[70] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[71] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Proceedings of CRYPTO: Ninth Annual International Cryptology Conference*, volume 435 of *LNCS*, pages 218–238. Springer, 1989.

[72] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: extended abstract. In *Proceedings of CCS: Eighth ACM conference on Computer and Communications Security*, pages 245–254. ACM Press, 2001.

[73] Stanley Milgram. The small world problem. *Psychology Today*, 61(2):60–67, 1967.

[74] Antonio Nicolosi and David Mazières. Secure acknowledgment of multicast messages in open peer-to-peer networks. In Geoffrey M. Voelker and Scott Shenker, editors, *Proceedings of IPTPS: Third International Workshop on Peer-to-Peer Systems, Revised Selected Papers*, volume 3279 of *LNCS*, pages 259–268. Springer, 2004.

[75] Harald Niederreiter. A public-key cryptosystem based on shift register sequences. In Franz Pichler, editor, *Proceedings of EUROCRYPT: Workshop on the Theory and*

Application of Cryptographic Techniques, volume 219 of *LNCS*, pages 35–39. Springer, 1986.

[76] Kaisa Nyberg and Rainer A. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. In Alfredo De Santis, editor, *Proceedings of EUROCRYPT: Workshop on the Theory and Application of Cryptographic Techniques*, volume 950 of *LNCS*, pages 182–193. Springer, 1994.

[77] Panagiotis Papadimitratos and Zygmunt J. Haas. Securing mobile ad hoc networks, the handbook of ad hoc wireless networks. pages 551–567, 2003.

[78] Eric Peeters, Michael Neve, and Mathieu Ciet. XTR implementation on reconfigurable hardware. In Marc Joye and Jean-Jacques Quisquater, editors, *Proceedings of CHES: Sixth International Workshop on Cryptographic Hardware and Embedded Systems*, volume 3156 of *LNCS*, pages 386–399. Springer, 2004.

[79] Holger Petersen and Patrick Horster. Self-certified keys – concepts and applications. In *Proceedings of CMS: Third Communications and Multimedia Security*, pages 102–116. Chapman & Hall, 1997.

[80] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[81] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006.

[82] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[83] Bruce Schneier. *Secrets and lies: digital security in a networked world.* Wiley, New York, 2000.

[84] Claus Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[85] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod *p*. *Mathematics of Computation*, 44:483–494, 1985.

[86] Alex C. Snoeren. Hash-based ip traceback. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–14, New York, NY, USA, 2001. ACM.

[87] Douglas Stinson. *Cryptography: Theory and Practice,Second Edition.* CRC/C&H, 2002.

[88] Takashi Suzuki, Zulfikar Ramzan, Hiroshi Fujimoto, Craig Gentry, Takehiro Nakayama, and Ravi Jain. A system for end-to-end authentication of adaptive multimedia content. In *Proceedings of CMS '04: Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*, volume 175 of *LNCS*, pages 237–249. Springer, 2005.

[89] Srdjan Čapkun, Levente Buttyán, and Jean-Pierre Hubaux. Small worlds in security systems: an analysis of the pgp certificate graph. In *Proceedings of NSPW: Workshop on New security paradigms*, pages 28–35, New York, NY, USA, 2002. ACM.

[90] Srdjan Čapkun, Levente Buttyán, and Jean-Pierre Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52–64, 2003.

[91] Srdjan Čapkun and Jean-Pierre Hubaux. Biss: building secure routing out of an incomplete set of security associations. In *WiSe '03: Proceedings of the 2nd ACM workshop on Wireless security*, pages 21–29, New York, NY, USA, 2003. ACM.

[92] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Proceedings of EUROCRYPT: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, May 22-26, Aarhus, Denmark*, volume 3494 of *LNCS*, pages 114–127. Springer, 2005.

[93] Duncan J. Watts. *Small Worlds: The Dynamics of Networks Between Order and Randomness.* Princeton University Press, 1999.

[94] Jing Xu, Zhenfeng Zhang, and Dengguo Feng. Id-based aggregate signatures from bilinear pairings. In Yvo Desmedt, Huaxiong Wang, Yi Mu, and Yongqing Li, editors, *Proceedings of CANS '05: Fourth International Conference on Cryptology and Network Security, December 14-16, Xiamen, China*, volume 3810 of *LNCS*, pages 110–119. Springer, 2005.

[95] Jun Xu, Jinliang Fan, Mostafa H. Ammar, and Sue B. Moon. Prefix-preserving ip address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *ICNP '02: Proceedings of the 10th IEEE International Conference on Network Protocols*, pages 280–289, Washington, DC, USA, 2002. IEEE Computer Society.

[96] A. Yaar, A. Perrig, and D. Song. Pi: A path identification mechanism to defend against DDoS attacks. In *IEEE Symposium on Security and Privacy*, 2003.

[97] Xiaowei Yang, David Wetherall, and Thomas Anderson. A dos-limiting network architecture. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 241–252, New York, NY, USA, 2005. ACM.

[98] Meiyuan Zhao, Sean W. Smith, and David M. Nicol. Aggregated path authentication for efficient BGP security. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *Proceedings of CCS '05: 12th ACM Conference on Computer and Communications Security, November 7-11, Alexandria, VA, USA*, pages 128–138. ACM Press, 2005.

[99] Zhu Zhao, Zhongqi Dong, and Yongge Wang. Security analysis of a password-based authentication protocol proposed to IEEE 1363. *Theoretical Computer Science*, 352(1):280–287, 2006.

[100] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.

[101] Philip R. Zimmermann. *The official PGP user's guide*. MIT Press, Cambridge, MA, USA, 1995.

VITA

Name: Saikat Chakrabarti

Date and Place of Birth: 5^{th} January 1977, Calcutta, India.

Education: B.E., Electrical Engineering
Bengal Engineering and Science University, India, 1999

Employment History:

1. June 2008—Present: Sr. Security Architect, Siemens IT Solutions and Services, Inc.
2. August 2001—July 2002: Technical Architect, General Electric, via Tata Consultancy Services (TCS)
3. March—September 2001: Technical Lead, GE-Corporate, via TCS
4. October 2000—February 2001: Senior Software Engineer, Cunningham Lindsey, via TCS.
5. August 1999—September 2000: Software Engineer, Hammond Suddards Edge/Praxis Partners, via TCS.
6. May 1998—July 1998: Summer Intern, Siemens, Inc.

Honors: Kentucky Graduate Scholarship
University of Kentucky, 2002—2008

NSF student travel grant
ESAS 2007, Cambridge, UK 2007

Kentucky SBIR Phase 0 Grant
Department of Homeland Security, 2007

Second place in Annual Entrepreneurship Competition
Lexington Venture Club and Kentucky Science and Technology Center, 2007

Placed among top 1% of 58,000 candidates
Joint Entrance Examination (National Level, India), 1995

Selected Publications:

1. Saikat Chakrabarti and Mukesh Singhal, "Password-based Authentication: Dictionary Attacks and Prevention", IEEE Computer, vol. 40, no. 6, June, 2007, pp. 68-74.

2. Venkata C. Giruka, Saikat Chakrabarti and Mukesh Singhal, "A Distributed Multi-Party Key Agreement Protocol for Dynamic Collaborative Groups using ECC", Journal of Parallel and Distributed Computing (JPDC) , vol. 66, no. 7, July 2006, pp 959-970

3. Saikat Chakrabarti, Venkata C. Giruka and Mukesh Singhal, "Authentication in Wireless Networks", Book Chapter 5 in "Security in Distributed, Grid, and Pervasive Computing", Edited by Prof. Yang Xiao, Auerbach Publications, CRC Press 2006

4. Venkata C. Giruka, Saikat Chakrabarti, and Mukesh Singhal, "Key Management and Agreement in Distributed Systems", Book Chapter 2 in "Security in Distributed, Grid, and Pervasive Computing", Edited by Prof. Yang Xiao, Auerbach Publications, CRC Press 2006

5. Saikat Chakrabarti, Santosh Chandrasekhar, Mukesh Singhal and Kenneth L. Calvert, "Authenticating DSR using a Novel Multisignature Scheme based on Cubic LFSR Sequences", in Proc. of The Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS), Cambridge, United Kingdom, July 2-3, 2007, LNCS, vol. 4572, Springer 2007, pp. 156-171

6. Saikat Chakrabarti, Santosh Chandrasekhar, Mukesh Singhal and Kenneth L. Calvert, "Authenticating Feedback in Multicast Applications Using a Novel Multisignature Scheme Based on Cubic LFSR Sequences", in Proc. of The Third IEEE International Symposium on Security in Networks and Distributed Systems (SSNDS) , in conjunction with AINA, Niagara Falls, Canada, May 21-23, 2007, IEEE Computer Society, vol. 1, pp. 607-613

7. Saikat Chakrabarti, Santosh Chandrasekhar, Kenneth L. Calvert and Mukesh Singhal,"Efficient Blind Signatures for Accountability", in Proc. of The Third Workshop on Secure Network Protocols (NPSec 2007), in conjunction with the 15th IEEE International Conference on Network Protocols (ICNP), Beijing, China, October 16-19, 2007, IEEE Xplore 2.0, pp 01-06

8. Santosh Chandrasekhar, Saikat Chakrabarti, and Mukesh Singhal, "Efficient Proxy Signatures for Ubiquitous Computing", in Proc. of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2008), Taichung, Taiwan, June 11-13, 2008

9. Saikat Chakrabarti, George V. Landon, and Mukesh Singhal, "Graphical Passwords: Drawing a Secret with Rotation as a New Degree of Freedom", in Proc. of the Fourth IASTED Asian Conference on Communication Systems and Networks (AsiaCSN 2007), Phuket, Thailand, April 2-4 2007, ACTA Press, pp. 561-173