Signatures for Network Coding

by

Ning Zhang

A research paper
presented to the University of Waterloo
for the degree of Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In communication networks, files commonly are separated into data packets and transmitted from the source node to a prescribed set of destination nodes by a method known as "store and forward", in which data packets received are stored and then forwarded to the next node. Network coding has been proposed to replace the traditional "store and forward" model, and to improve the throughput and robustness of networks. With linear network coding, instead of copying and forwarding, intermediate nodes compute linear combinations of previously received information, and create and transmit a new packet to the next nodes. However, some malicious intermediate nodes may modify the data of packets and the recipients not can distinguish the corrupted packets from uncorrupted ones. This is known as a "pollution attack".

Homomorphic hashings and homomorphic signatures were proposed to provide cryptographic protection against pollution attacks. First, we present Krohn et al.'s homomorphic hashing schemes for rateless erasure codes. This paper presented six homomorphic signature schemes for network coding including three pairing-based homomorphic signature schemes, three RSA-based homomorphic signature schemes, and a signature scheme using a vector orthogonal to the message linear subspace. We then discuss and compare the memory and communication overhead of five of these homomorphic signature schemes. Finally, we point out flaws with some of these schemes, even though some security analysis has been presented, and we present solutions to prevent such attacks.

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Edlyn Teske-Wilson, who gave me valuable help and supervision with her endless patience throughout the period of my MMath graduate degree program.

I would like to thank Professor Alfred Menezes at the University of Waterloo for answering my many questions on the security of cryptographic protocols for network coding. I am also grateful to Professor David Jao for all the discussions about the formal security proofs of cryptographic schemes.

I would like to give special thanks to all my (former) colleagues and friends for their valuable friendship and support.

Last but not least, I would like to thank my family and my friends for their support and for their endless love to me. Special thanks go out to my husband Xinxin Fan who gives me his unconditional love and valuable help.

To all of you thank you very much!

Contents

Chapter 1

Introduction

In communication networks, information is transmitted by a routing mechanism, in which data packets are sent from the source to the destination. *Network coding* has proposed to replace the traditional "store and forward" model, and to improve the capacity and robustness of networks. If we assume all nodes in the network system are honest, the data packets can be transmitted without being modified intentionally. However, some malicious intermediate nodes may modify the data of packets and the recipients cannot distinguish the corrupted packets from uncorrupted ones. This is known as a "pollution attack".

Network coding signatures have been proposed to protect against pollution attacks. This work focuses on six homomorphic signature schemes for network coding. We present their security and performance comparisons, and discuss and analyze some flaws of these signature schemes.

This chapter starts with a brief introduction to network coding theory and homomorphic signatures for network coding.

1.1 Network Coding

In various communication networks such as phone networks, Internet, Peer-to-Peer(P2P) networks, wireless ad hoc networks and sensor networks, files commonly are separated into data packets and transmitted from the source node to a prescribed set of destination nodes by a method known as "store and forward" (see Fig 1.1), in which data packets received are stored and then forwarded to the next node.

With (linear) network coding, instead of copying and forwarding, intermediate nodes compute (linear) combinations of previously received information, and create and transmit a new packet to the next nodes. Network coding has vast application potential and has been

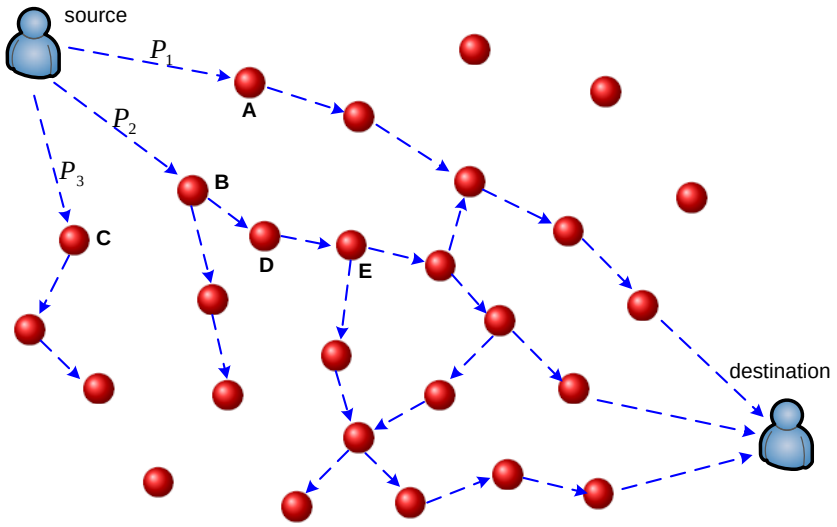

Figure 1.1: File Distribution in Communication Networks: Store-and-Forward

shown to offer a number of significant benefits: improved throughput, increased robustness of networks and data security. We will give more detailed explanation in Chapter 3.

1.2 Homomorphic Signatures for Network Coding

Network coding presents some challenges as well as advantages in networks. A major concern for any network coding system is the protection against pollution attacks. In such systems, malicious nodes can make modifications of packets intentionally. This problem can be particularly serious because errors introduced into even a single packet can propagate and pollute multiple packets. To solve this problem, a signature scheme can be used to provide cryptographic protection. By verifying the signature appended to each packet, intermediate nodes can detect errors and filter out any corrupted packets, and then make sure the destination node recovers the correct file (see Fig 1.2).

What we need is to guarantee the integrity of the message packets in the transmissions. However, since the content of packets in transmissions is modified by the intermediate nodes, the regular signature by the source node on the original message is not sufficient. Homomorphic signatures are appropriate for linear network coding. In linear network coding systems, the intermediate nodes need to verify the incoming homomorphic signatures and accept the packets passing the verification. Then the intermediate nodes can combine these correct signatures and construct a new valid signature without any access to the private key of the source node.

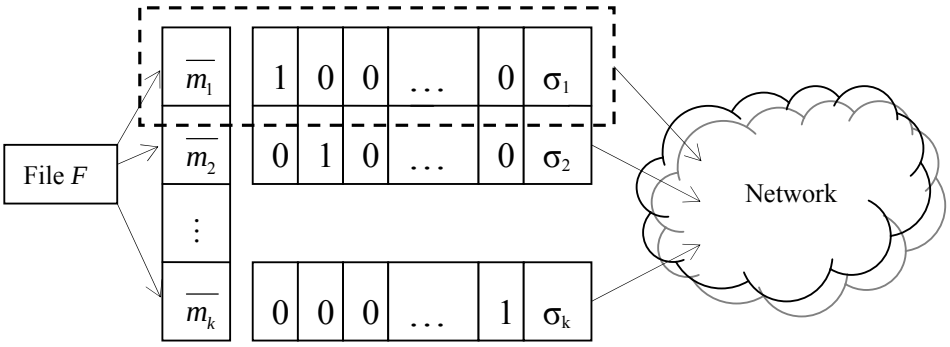

Figure 1.2: Network Coding Signatures

1.3 This Work

Over the past years, several homomorphic signature schemes for network coding have been proposed. In this paper, we introduce some related work in this research area. Homomorphic hashing was first proposed by Krohn et al. [33]; their scheme needs a reliable channel for pre-distributing the hash values. Zhao et al. [46] proposed a signature scheme using a vector orthogonal to the message subspace. Charles et al. [15] presented a homomorphic signature based on aggregate signatures. Boneh et al. presented two signature schemes $\mathcal{S}_1$ and $\mathcal{S}_2$ [8] that use bilinear pairings. The scheme $\mathcal{S}_2$ is a provably secure homomorphic signature with random oracles, and the scheme $\mathcal{S}_1$ is to authenticate all hash values using a standard signature scheme. Also, Two RSA-based homomorphic signature schemes were given for network coding [45, 23]. In particular, Gennaro et al. [23] proposed an RSA-based scheme in which small integers can be chosen as the coefficients in the linear combinations, and thus the total bandwidth overhead for the transmission from the source to the destination can be reduced significantly.

However, there are flaws in some of these schemes. In this paper, we present these flaws and some existential forgeries to the homomorphic signatures for network coding. Because of the homomorphic property, the adversary may forge a valid signature on a message packet chosen by herself, and so the next nodes will accept this corrupted packet as a correct one.

1.4 Outline

The outline of this paper is as follows:

- Chapter 1 introduces the background of network coding, the basic concepts of homo-

morphic signatures for network coding, typical attacks and security. The motivation and the context of the work in this paper are also presented.

- Chapter 2 gives a brief overview of the mathematical tools and security assumptions that will be extensively used throughout this work. We first introduce the properties of finite fields, followed by a description of elliptic curves and bilinear pairings. We also present some assumptions on the problems (Discrete-Logarithm Problem, Diffie-Hellman Problem, Elliptic Curve Discrete-Logarithm Problem and Elliptic Curve Diffie-Hellman Problem) that are widely used in the security proofs of many cryptographic primitives. In these assumptions, we say that the intractable problems cannot be solved by any probabilistic polynomial time (PPT) algorithm with non-negligible probability.

- Chapter 3 gives a brief introduction to network coding theory. We show some basic concepts of network coding, describe the applications of network coding in P2P content distribution networks and wireless ad hoc networks, and explain the benefits of network coding with respect to throughput, robustness and security. We also introduce linear network coding, which is used frequently in practice, and consider the problem of network security.

- Chapter 4 contains our main work. First, we introduce homomorphic hashings and the definitions of homomorphic signatures. Then six homomorphic signature schemes for network coding are presented:

  · Boneh-Freeman-Katz-Waters' two schemes $\mathcal{S}_1$ and $\mathcal{S}_2$ [8].

  · Charles-Jain-Lauter's scheme [15].

  · Yun-Cheon-Kim's RSA-based scheme [45].

  · Gennaro-Katz-Krawczyk-Rabin's RSA-based scheme [23].

  · Zhao-Kalker-Medard-Han's scheme [46].

  Next, we discuss the security and compare the memory and communication overhead of the six homomorphic signature schemes above.

- Chapter 5 discusses flaws of some of the signature schemes described in Chapter 4. We introduce a weakness of the homomorphic property in signature schemes based on the standard model, give a forgeability attack on Charles et al.'s scheme, a forgeability attack on Yun et al.'s RSA-based signature scheme, and describe two forgeries of the scheme by Zhao et al..

- Chapter 6 concludes this paper, summarizes our main work, and suggests some directions for future research.

Chapter 2

Preliminaries

In this chapter, we give a brief overview of the mathematic tools and security assumptions that will be extensively used throughout this paper. We first introduce the definition and some properties of finite fields, followed by the description of elliptic curves and bilinear pairings. We also present some intractability assumptions that are widely used in the security proofs of many cryptographic primitives, such as key exchange protocols, public key encryption algorithms, and digital signature schemes. In these assumptions, we say that the intractable problems cannot be solved by any probabilistic polynomial time (PPT) algorithm with non-negligible probability.

2.1 Finite Fields

The definition of finite fields is given and some basic properties are provided as well.

Definition 2.1.1 (Finite Fields [37]) *A finite field is a field $\mathbb{F}$ which contains a finite number of elements. The order of $\mathbb{F}$ is the number of elements in $\mathbb{F}$.*

We list some facts about finite fields which are important in many cryptographic schemes.

1. If $\mathbb{F}$ is a finite field, then $\mathbb{F}$ contains p^m elements for some prime p and integer $m \geq 1$.
2. For every prime power order p^m, there is a unique (up to isomorphism) finite field of order p^m. This field is denoted by $\mathbb{F}_{p^m}$.
3. If $\mathbb{F}_q$ is a finite field of order $q = p^m$, p is a prime, then the characteristic of $\mathbb{F}_q$ is p.

4. If $\mathbb{F}_q$ is a finite field of order $q = p^m$, then every subfield of $\mathbb{F}_q$ has order p^n, for some n that is a positive divisor of m. Conversely, if n is a positive divisor of m, then there is exactly one subfield of $\mathbb{F}_q$ of order p^n; an element $a \in \mathbb{F}_q$ is in the subfield $\mathbb{F}_{p^n}$, if and only if $a^{p^n} = a$. We have $a^q = a$ for all $a \in \mathbb{F}_q$.

Definition 2.1.2 (Cyclic Group [37]) *A group $\mathbb{G}$ is cyclic if there is an element $g \in \mathbb{G}$ such that for each $a \in \mathbb{G}$ there is an integer m with $a = g^m$. Such an element g is called a generator of $\mathbb{G}$.*

Definition 2.1.3 (Multiplicative Group [37]) *The non-zero elements of $\mathbb{F}_q$ form a group under multiplication called the multiplicative group of $\mathbb{F}_q$, denoted by $\mathbb{F}_q^*$.*

2.2 Elliptic Curves

In this section, we provide the necessary background for elliptic curves and the bilinear maps from elliptic curve pairings. Most results in this section come from [2, 3].

Let $\mathbb{F}_q$ be a finite field of the order q and characteristic p. An elliptic curve E over $\mathbb{F}_q$ is defined by the *affine Weierstrass* equation:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{2.1}$$

where $a_i \in \mathbb{F}_q$. The set $E(\mathbb{F}_q)$ of $\mathbb{F}_q$-rational points consists of the points $(x, y) \in \mathbb{F}_q \times \mathbb{F}_q$ that satisfy the affine Weierstrass equation, together with an point at infinity denoted by ∞.

Assuming that $p \neq 2, 3$, we consider the following admissible changes of variables given by

$$x = x' - \frac{b_2}{12},\ y = y' - \frac{a_1}{2}\left(x' - \frac{b_2}{12}\right) - \frac{a_3}{2}, \tag{2.2}$$

and substitute x, y in the affine Weierstrass equation; we get the equation of an isomorphic curve, called the *short Weierstrass* form,

$$y^2 = x^3 + ax + b \tag{2.3}$$

for some $a,\ b \in \mathbb{F}_q$.

The set of points $E(\mathbb{F}_q)$ forms an abelian group, where ∞ is the identity element. The group operation, denoted by $+$, is given by a chord-and-tangent rule, that can be best explained geometrically. Let P and Q be two distinct points on E. If the straight

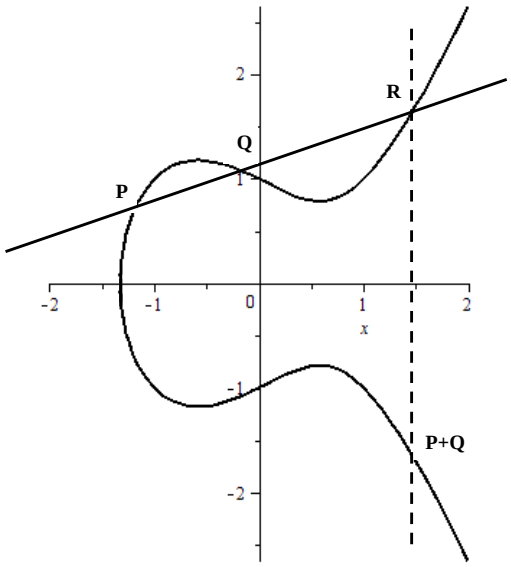

Figure 2.1: Addition of two points on an elliptic curve

line through P and Q intersects the curve at another point, say $R = (x, y)$, then we say $P + Q = -R = (x, -y)$, where $-R$ is a reflection of R in the x-axis (see Figure 2.1).

If $P = Q$, then $P + Q = 2P$ is called doubling operation of a point on an elliptic curve $E(\mathbb{F}_q)$. When the tangent line to $E(\mathbb{F}_q)$ at the point P intersects $E(\mathbb{F}_q)$ in exactly one other point, say $R = (x, y)$, then we say $2P = -R = (x, -y)$, where $-R$ is a reflection of R in the x-axis. (see Figure 2.2).

In a special case, if Q is the reflection of P in the x-axis, that is, $Q = -P$, the straight line through P and Q does not intersect the curve in any other points. In this case, we define $P+Q = P+(-P) = \infty$, so $-P$ is called the inverse of P. Also, we define $P+\infty = P$ and $\infty + P = P$ for any point $P \in E(\mathbb{F}_q)$. So ∞ is the identity element in $E(\mathbb{F}_q)$.

With the group law described above, $E(\mathbb{F}_q)$ is *commutative* and *associative* under addition, that is, $P + Q = Q + P$ and $(P + Q) + R = P + (Q + R)$ for all $P, Q, R \in E(\mathbb{E}_q)$.

The notation nP denotes the scalar multiplication of $P \in E(\mathbb{F}_q)$ by an integer n. The value of nP is the following:

- For $n = 0$, it is equal to ∞.
- For $n \geq 1$, it is equal to $\underbrace{P + \cdots + P}_{n \text{ points}}$.
- For $n < 0$, it is equal to $(-n)(-P)$.

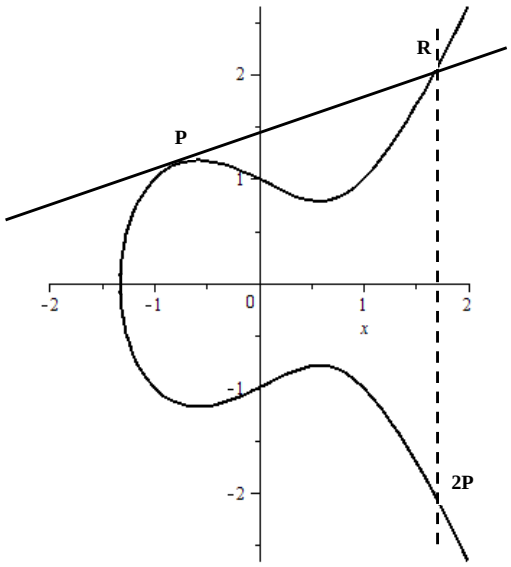

Figure 2.2: Doubling of a point on an elliptic curve

In general, scalar multiplication on elliptic curves is believed to be hard to invert. This problem is the so-called *Discrete Logarithm Problem* on an elliptic curve $E(\mathbb{F}_q)$, which is to be described in more detail in Section 2.3.1.

2.3 Bilinear Groups and Bilinear Maps

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be multiplicative groups of prime order q. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties [3]:

1. Bilinearity: For any $g_1, g_2 \in \mathbb{G}_1$, $h_1, h_2 \in \mathbb{G}_2$, we have $e(g_1 \cdot g_2, h_1) = e(g_1, h_1) \cdot e(g_2, h_1)$ and $e(g_1, h_1 \cdot h_2) = e(g_1, h_1) \cdot e(g_1, h_2)$.

2. Non-degeneracy: If $\hat{g}$ is a generator of $\mathbb{G}_1$, and $\hat{h}$ is a generator of $\mathbb{G}_2$, then $e(\hat{g}, \hat{h})$ is a generator of $\mathbb{G}_T$.

3. Computability: The map e can be efficiently computed.

The following properties of bilinear pairings can be easily verified. For any $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$:

1. $e(g, h^{-1}) = e(g^{-1}, h) = e(g, h)^{-1}$.

2. $e(g^a, h^b) = e(g, h)^{ab}$, for all $a, b \in \mathbb{Z}$.

3. $e(g, 1_{\mathbb{G}_2}) = 1_{\mathbb{G}_T}$, and $e(1_{\mathbb{G}_1}, h) = 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_1}, 1_{\mathbb{G}_2}$ and $1_{\mathbb{G}_T}$ are the identity elements of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, respectively.

4. If $e(g, h) = 1_{\mathbb{G}_T}$ for all $h \in \mathbb{G}_2$, then $g = 1_{\mathbb{G}_1}$; if $e(g, h) = 1_{\mathbb{G}_T}$ for all $g \in \mathbb{G}_1$, then $h = 1_{\mathbb{G}_2}$.

A bilinear group tuple is a tuple $\langle \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e \rangle$ with the properties above. For currently known bilinear group tuples, $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of groups of points on elliptic curves, $\mathbb{G}_T$ is a subgroup of a multiplicative group of a finite field, and the map e can be derived by modifying either the Weil pairing [3] or Tate pairing [3] on an elliptic curve E over $\mathbb{F}_q$. Generally the computational complexity of the Tate pairing is less than that of the Weil pairing.

2.4 Security Assumptions

For the security, all cryptographic protocols must rely on intractability assumptions, which state that problems cannot be solved in polynomial time. This section presents the standard Discrete-Logarithm (DL) Problem and Diffie-Hellman (DH) Problem, and also describes some variants and related security assumptions over elliptic curves.

2.4.1 Discrete-Logarithm (DL) and Diffie-Hellman (DH) Problems

Definition 2.4.1 (Discrete Logarithm Problem [2]) *Let $\mathbb{G}$ be a cyclic group of prime order q, and let g be a random generator of $\mathbb{G}$. Given a random element $a \in \mathbb{G}$, the Discrete Logarithm Problem in $\mathbb{G}$ is to find an $\omega \in \mathbb{Z}_q$ such that $a = g^\omega$.*

Such an ω is called *discrete logarithm* of a in $\mathbb{G}$, denoted $DL(a, g)$. We say that an algorithm $\mathcal{A}$ has advantage ϵ in solving the DL Problem in $\mathbb{G}$ if

$$Pr[\mathcal{A}(a, g) = DL(a, g)] \geq \epsilon,$$

where the probability is measured over the random choices of $a, g \in \mathbb{G}$ and the random inputs of $\mathcal{A}$, if any.

Diffie and Hellman [20] proposed the Diffie-Hellman Problem in the key exchange protocol, which is closely related to the DL Problem and is the basis for the security of many

cryptographic protocols. The DH Problem that we discuss comes in two forms: computational and decisional. Generally speaking, a decisional problem asks to distinguish a random element and a computational problem asks to output a valid element. We present the definitions of the computational DH Problem and decisional DH Problem as follows.

Definition 2.4.2 (Computational Diffie-Hellman Problem (CDH) [3]) *Let $\mathbb{G}$ be a cyclic group of prime order q, and let g be a random generator of $\mathbb{G}$. Given g and the random elements g^a and g^b in $\mathbb{G}$, where $a, b \in \mathbb{Z}_q$, the Computational Diffie-Hellman (CDH) Problem in $\mathbb{G}$ is to find $g^{ab} \in \mathbb{G}$.*

We say that an algorithm $\mathcal{A}$ has advantage ϵ in solving the CDH Problem in $\mathbb{G}$ if

$$Pr[\mathcal{A}(g^a, g^b) = g^{ab}] \geq \epsilon,$$

where the probability is measured over the random choices of $g, g^a, g^b \in \mathbb{G}$ and the random inputs of $\mathcal{A}$, if any.

If we suppose that the DL Problem can be efficiently solved, then one can solve the CDH Problem as follows. Given g and the random elements g^a and g^b in $\mathbb{G}$, find a from g^a by solving the DL Prolem, and then compute $(g^b)^a = g^{ab}$.

The decisional Diffie-Hellman (DDH) Problem is closely related to the CDH Problem.

Definition 2.4.3 (Decisional Diffie-Hellman (DDH) Problem [3]) *Let $\mathbb{G}$ be a cyclic group of prime order q, and let g be a random generator of $\mathbb{G}$. Given g and the random elements g^a, g^b and g^c in $\mathbb{G}$, where $a, b, c \in \mathbb{Z}_q$, the Decisional Diffie-Hellman (DDH) Problem in $\mathbb{G}$ is to determine if $g^{ab} = g^c$.*

If there exists a group $\mathbb{G}$ in which the CDH Problem is hard but the DDH Problem is easy, we call $\mathbb{G}$ a Gap Diffie-Hellman (GDH) group. The first example of a GDH group was given in [31].

Now we introduce two variants of the DH Problem: co-DH Problem and (q, k, N)-DH Problem. The co-Diffie-Hellman Problem was proposed by Boneh, Lynn and Shacham [10] to assist in the security proof of the BLS signature scheme. The (q, k, N)-Diffie-Hellman Problem was proposed by Zhao et al. [46] and is used in Chapter 4.

Definition 2.4.4 (Computational co-Diffie-Hellman (co-CDH) Problem) *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic groups of the same prime order q, and let g_1 be a random generator of $\mathbb{G}_1$ and g_2 be a random generator of $\mathbb{G}_2$. Given g_2 and the random elements $g_1^a \in \mathbb{G}_1$, $g_2^b \in \mathbb{G}_2$, where $a, b \in \mathbb{Z}_q$, the Computational co-Diffie-Hellman (co-CDH) Problem is to compute $g_1^{ab} \in \mathbb{G}_1$.*

Definition 2.4.5 (Decisional co-Diffie-Hellman (co-DDH) Problem) *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic groups of the same prime order q, g_1 be a random generator of $\mathbb{G}_1$ and g_2 be a random generator of $\mathbb{G}_2$. Given $g_1^a, h \in \mathbb{G}_1$ and $g_2, g_2^b \in \mathbb{G}_2$, where $a, b \in \mathbb{Z}_q$, the Decisional co-Diffie-Hellman (co-DDH) Problem is to determine if $h = g_1^{ab}$.*

We note that the co-CDH Problem and co-DDH Problem are the generalizations of the CDH Problem and DDH Problem. When $\mathbb{G}_1 = \mathbb{G}_2$ and $g_1 = g_2$, the co-CDH Problem and co-DDH Problem reduce to the standard CDH Problem and DDH Problem, respectively.

Definition 2.4.6 ((q, k, N)-Diffie-Hellman Problem [46]) *Let $\mathbb{G}$ be a multiplicative cyclic group of prime order q, k and N be two integers such that $k < N - 1$, and $\mathbf{\Gamma} = \{g_1, \ldots, g_N\}$ be a set of generators of $\mathbb{G}$. Given a linear subspace V of rank k in $\mathbb{F}_q^N$ such that for every $\mathbf{v} \in V$, the equality $\mathbf{\Gamma}^{\mathbf{v}} := \prod_{j=1}^{N} g_j^{v_j} = 1$ holds, the (q, k, N)-Diffie-Hellman Problem is defined as the problem of finding a vector $\mathbf{w} \in \mathbb{F}_q^N$ with $\mathbf{\Gamma}^{\mathbf{w}} = 1$ but $\mathbf{w} \notin V$.*

The (q, k, N)-Diffie-Hellman Problem is proposed to assist in the security proof of a signature for network coding [46], and is proven to be as hard as the Discrete Logarithm Problem. In this problem, we need to set $k < N - 1$, since if $k = N - 1$, then the (q, k, N)-Diffie-Hellman Problem has no solution. To prove this, we suppose that there exists a $\mathbf{w}'$ such that $\prod_{i=1}^{N} g_i^{w_i'} = 1$ and $\mathbf{w}' \notin V$. Then $\mathbf{w}' + V$ spans the whole space $\mathbb{F}_q^N$. Thus we have any vector $\mathbf{w} \in \mathbb{F}_q^N$ satisfying $\prod_{i=1}^{N} g_i^{w_i} = 1$. This is clearly impossible. Therefore, no such $\mathbf{w}'$ exists. When $k < N - 1$, there exist solutions to the (q, k, N)-Diffie-Hellman Problem, but finding a solution to the (q, k, N)-Diffie-Hellman Problem is as hard as solving the Discrete Logarithm Problem in the cyclic group $\mathbb{G}$ (see the proof in [46]).

2.4.2 Elliptic Curve DL Problem and Bilinear DH Problem

Some intractability assumptions provided in Section 2.3.1 can be extended to the bilinear group setting. The variant of the DL Problem on an elliptic curve E over some finite field $\mathbb{F}_q$ is given as follows.

Definition 2.4.7 (Elliptic Curve Discrete Logarithm (ECDL) Problem [2]) *Let $E(\mathbb{F}_q)$ be the group of points on an elliptic curve E over a finite field $\mathbb{F}_q$, and let P be an element of $E(\mathbb{F}_q)$. P generates a cyclic subgroup of $E(\mathbb{F}_q)$, denoted by $\langle P \rangle$. Given $Q \in \langle P \rangle$, the Elliptic Curve Discrete Logarithm (ECDL) Problem on E is to find an integer n such that $Q = nP$.*

The most natural generalization of the Diffie-Hellman Problem to a bilinear group setting is the series of bilinear Diffie-Hellman (BDH) problems. We present two variants of the BDH problems. The most basic BDH version appeared in Joux's construction of the three-party Diffie-Hellman key agreement protocol [30], and also in the security proof of the IBE scheme of Boneh and Franklin [7].

Definition 2.4.8 (Bilinear Diffie-Hellman (BDH) Problem) *Let $\mathbb{G}$ be a cyclic group of prime order q with an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}_T$ is a multiplicative group. Let g be a generator of $\mathbb{G}$. Given $g,\ g^a,\ g^b,\ g^c \in \mathbb{G}$, for any $a, b, c \in \mathbb{Z}_p$, the Bilinear Diffie-Hellman Problem is to compute $e(g,\ g)^{abc} \in \mathbb{G}_T$.*

Definition 2.4.9 (Decisional Bilinear Diffie-Hellman (DBDH) Problem) *Let $\mathbb{G}$ be a cyclic group of prime order q with an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}_T$ is a multiplicative group. Let g be a generator of $\mathbb{G}$. Given $g,\ g^a,\ g^b,\ g^c \in \mathbb{G}$ and $v \in \mathbb{G}_T$, for any $a, b, c \in \mathbb{Z}_p$, the Decisional Bilinear Diffie-Hellman Problem is to determine if $v = e(g,\ g)^{abc} \in \mathbb{G}_T$.*

The hardness of the BDH Problem forms the security foundation for many pairing-based cryptographic schemes. There also exist other important computational problems related to pairing-based schemes, most of which are based on the hardness of the standard DH Problem [4, 11].

2.5 Definitions of Digital Signatures

A digital signature scheme is one of the most important cryptographic primitives enabled by public key cryptography, and is fundamental in providing various cryptographic services such as data origin authentication, data integrity, and non-repudiation. In this section, we present the definition of digital signatures and security notions.

Definition 2.5.1 (Digital Signature Scheme [37]) *A digital signature scheme typically consists of four algorithms:* {**Setup**, **KeyGen**, **Sign**, **Verify**} *as follows:*

- **Setup**(l)*: On input security parameter l, outputs a set of parameters* params.
- **KeyGen**(params)*: On input the public parameters* params*, outputs a public key PK and a private key SK.*
- **Sign**($m,\ SK$)*: On input a message m and a private key SK, outputs a signature σ for the message m.*

- **Verify**$(m,\ PK,\ \sigma)$*: On input a message* m*, a public key* PK *and a signature* σ*, outputs either 0 (reject) or 1 (accept).*

Informally, signatures produced by the **Sign** algorithm should be valid, that is, accepted by the **Verify** algorithm. Creating a signature also should be computationally infeasible for any entity other than the signer. These two properties are the basis for provable security of signature schemes. We also present the definition of unforgeability for digital signatures.

Correctness

A digital signature scheme is *correct* if, for any σ produced by running **Sign** on message m and private key SK, then **Verify**$(m,\ \text{PK},\ \sigma)$ outputs *accept*.

Unforgeability

We define the existential unforgeability under a chosen message attack [26]. This is defined via the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

- **Initialization**: $\mathcal{C}$ runs **Setup** on security parameter l to generate the public parameters and runs **KeyGen** to obtain a public key PK and a private key SK. $\mathcal{C}$ sends the parameters and the public key PK to $\mathcal{A}$.
- **Sign**: $\mathcal{A}$ can query q times to $\mathcal{C}$, and $\mathcal{C}$ computes $\sigma_i = \textbf{Sign}(m_i, \text{SK})$, where $m_i,\ i = 1, \ldots, q$ can be any message chosen by $\mathcal{A}$. $\mathcal{C}$ gives $\sigma_i,\ i = 1, \ldots, q$ to $\mathcal{A}$.
- **Output**: $\mathcal{A}$ outputs a signature σ^* and a message m^*, where $m^* \notin \{m_1, \ldots, m_q\}$. $\mathcal{A}$ wins the game if **Verify**$(m^*, \text{PK},\ \sigma^*)$ outputs *accept*.

We say that a digital signature scheme is *existentially unforgeable under an adaptive chosen message attack* if the probability of success of any polynomially bounded adversary in the above game is negligible.

Classification of Signature Schemes

Signature schemes are usually based on computational intractability assumptions. A signature scheme is said to be secure in the *standard model* [4, 6], if it can be proven secure using only computational complexity assumptions.

Security proofs of signature schemes are not easy to achieve in the standard model, so signature schemes often employ random oracles to provide proofs of security. A random

oracle is an oracle (or a black box) that responds to every query by selecting a random element from its range. Random oracles can be used in cryptographic proofs of security. A signature scheme that is proven secure using such a proof is said to be secure in the random oracle model, as opposed to secure in the standard model.

In practice, random oracles are typically used to model cryptographic hash functions in schemes where strong randomness requirements are needed of the hash function's output [7, 9, 8]. No practical function can implement a true random oracle. In fact, certain signature schemes are known which are proven secure in the random oracle model, but which are trivially insecure when any real function is substituted for the random oracle [14]. However, not all uses of cryptographic hash functions require random oracles: schemes which require only the property of collision resistance can be proven secure in the standard model.

Chapter 3

Introduction to Network Coding Theory

The fundamental concept of network coding was first proposed for satellite communication networks [44], and then fully developed as a new theory by Ahlswede et al. [1]. Due to its generality and its vast application potential, network coding has generated much interest in information theory, wireless communications, complexity theory, cryptography and graph theory.

The aim of this chapter is to give a brief introduction to network coding theory. We show some basic concepts of network coding and explain where and why network coding needs to be used. We also introduce linear network coding, which is used frequently in practice, and consider the problem of network security.

3.1 Basic Concepts of Network Coding

In this section, we begin with the definition of network coding, and give several simple but convincing examples of network coding. We also describe applications of network coding in a variety of networks.

3.1.1 What is Network Coding?

There are several definitions about network coding that have been used for different network environments. Based on the descriptions of networking coding in [28, 1], we give the following definition to capture the main characteristics of network coding.

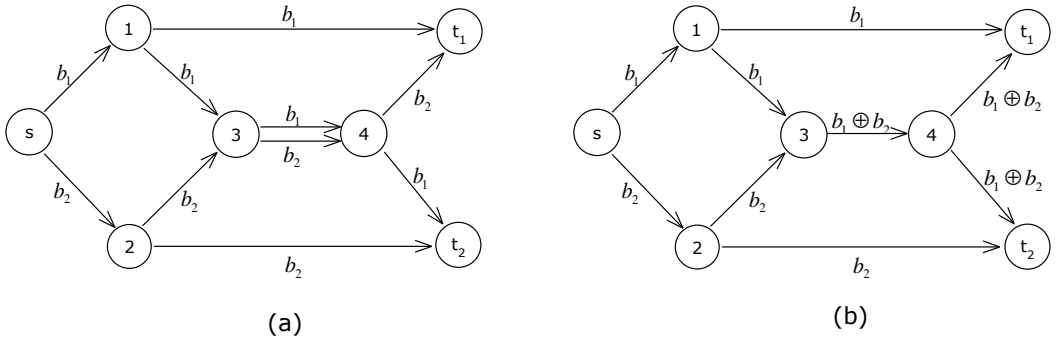

Figure 3.1: Multicasting two packets, b_1 and b_2, from the source node S to the destination nodes t_1 and t_2 in the butterfly network

Definition 3.1.1 (Network Coding) *Network coding is a particular data processing technique in networks that is applied for the purpose of increasing the capacity or the throughput of networks in the broadcast communication channel.*

Unfortunately, the above definition does not distinguish the study of network coding from communication network or information theory. Throughout this paper the concept of network coding we use means coding at a node in a packet network, in which data is separated into packets and the coding is applied to the content of each packet.

Communication networks today share the same fundamental principle of operations. In the traditional *store-and-forward* routing mechanisms, data packets are transmitted from the source node to each destination node through a chain of intermediate nodes. Each intermediate node, after receiving a data packet from an input link, replicates and stores the data, and then forwards it to the next node via an output link. In contrast to the store-and-forward method, network coding refers to a new class of routing mechanisms, in which intermediate nodes modify the received data packets in transit. For instance, instead of simply forwarding data, intermediate nodes can recombine several input packets into one or several output packets. A simple example is network coding on the butterfly network ([28], see Figure 3.1), which features a multicast from a single source to two sinks, or destinations.

Assume that source node S multicasts two packets, b_1 and b_2, to both destination nodes t_1 and t_2. In Figure 3.1 (a), every channel carries one packet, b_1 or b_2, and every intermediate node needs to replicate the received packets and then send them out. Figure 3.1 (b) describes a different way to multicast two packets in the same network. Instead of copying and forwarding each single packet, the intermediate node 3 computes and outputs the bitwise xor of b_1 and b_2, $b_1 \oplus b_2$. The channel from node 3 to node 4 is used to transmit $b_1 \oplus b_2$, which is then replicated at node 4 for passing on to nodes t_1 and t_2. The destination

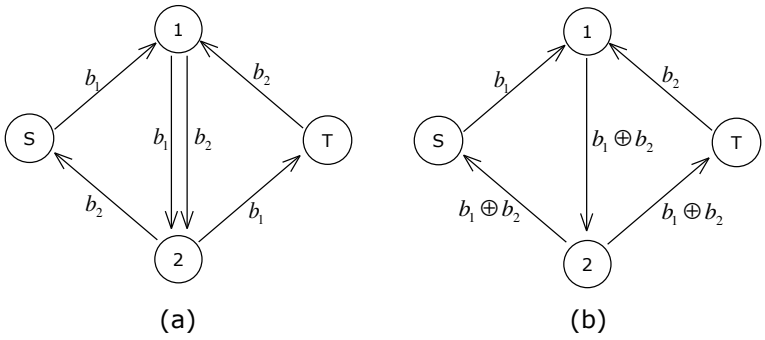

Figure 3.2: Conversation between two parties

nodes decode by performing further decoding operations on the received packets, that is, the node t_1 can recover b_2 by computing the xor of the received packets b_1 and $b_1 \oplus b_2$, and similarly, the node t_2 can recover b_1 by computing the xor of b_2 and $b_1 \oplus b_2$.

Another example is the conversation between two parties S and T (see Figure 3.2). In the network as shown in Figure 3.2 (a), the two parties S and T send one data packet to each other through the network in the straightforward manner, say, S sends a packet to T through the intermediate nodes 1 and 2, and vice versa. Figure 3.2 (b) shows the same network as in Figure 3.2 (a) but with one less channel. Upon receiving b_1 form S and b_2 from T, the intermediate node 1 derives a new packet $b_1 \oplus b_2$, which is sent to the intermediate node 2, and then to S and T. As a result, S and T can get the data packet from each other by performing the bitwise *xor*.

Linear network coding is similar to this example; the difference is that the *xor* operation is replaced by a linear combination of the data, interpreted as numbers over some finite field. In practice, linear network coding can achieve the best possible benefits of network coding. This allows for a much larger degree of flexibility in the way packets can be combined. Linear network coding is discussed further in Section 3.3.

3.1.2 Where is Network Coding Used?

In the following, we introduce applications of networks coding in wired network and wireless networks, and discuss how network coding improves the performance in concrete settings.

Peer-to-Peer (P2P) Content Distribution Network

In practical P2P content distribution networks, the file is split into many packets by the server before distributing. In a set of selected peers, peer nodes maintain connections to a

limited number of neighboring peers that can exchange packets. Peer nodes first download packets of the original file and then distribute them to their neighbors.

When using network coding in a content distribution network, instead of helping the server to distribute the packets intact, the peers compute randomly linear combinations of the packets before they forward them to neighbor peers; this system is called Avalance [25]. The coding coefficients are also transmitted together with packets. According to the matrix formed by coding coefficients of different packets, a peer node can determine how many new packets are needed to be transmitted to the neighbor, and these new packets can be generated by linear combinations of the original packets.

Network coding can improve the performance of P2P networks in several aspects [16, 21, 28]. Firstly, in a large scale content distribution system, optimal packet scheduling is very complicated since hosts only have very limited information about the underlying network topology. Employing network coding, the performance of the system depends much less on the specific topology structure and the scheduling mechanism. Secondly, in some special cases, for example, the server leaves before all peers have finished their download, or peer nodes only join for a short period of time or leave immediately after finishing their download. Network-coding based solutions are much more robust, because they can generate different coded packets with different linear combinations of original packets.

Wireless Network

The wireless network medium is different from the wired one. The wireless network supports mobility and portability, however, current wireless networks suffer from unreliability, unpredictability, low throughput and inadequate mobility support. However the characteristics of wireless networks provide opportunities for the application of network coding.

In a wireless environment, network coding can be used to offer benefits in terms of throughput, wireless bandwidth, and delay. We use a wireless ad hoc network as an example (see Figure 3.3). The nodes S and T exchange the packets b_1 and b_2 via the relay M. We assume that the time is slotted, that is, a device either transmits or receives a packet during a time slot.

Figure 3.3 (a) depicts a standard approach [22]: nodes S and T send their packets to the relay M, then M forwards each packet to the corresponding destination. With network coding in Figure 3.3 (b), the relay M first creates a new packet $b_1 \oplus b_2$, then sends it to both S and T. As a result S and T can decode the packet from each other by using a bitwise xor operation.

In the above example, the node M that uses network coding transmits once instead of twice, and the transmission needs two instead of three time slots. Moreover, the wireless

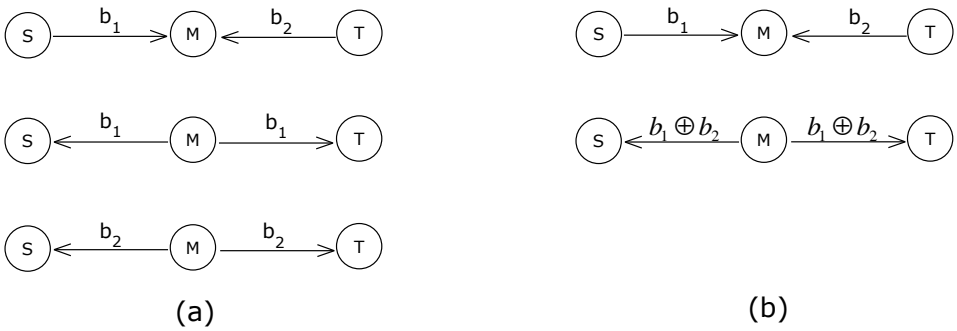

Figure 3.3: Information exchange in wireless ad hoc network

bandwidth is only occupied for a shorter time. Therefore, network coding in this example offers benefits in terms of resource utilization, throughput, delay and bandwidth.

3.2 Benefits of Network Coding

In this section, we discuss how network coding can improve the throughput and the robustness of networks.

3.2.1 Throughput

Network coding has been shown to offer a number of advantages, the most well-known of which is the possibility of increasing throughput in certain network topologies [28, 21]. The throughput benefit is achieved by using fewer packet transmissions to communicate more information in networks.

To demonstrate this benefit of network coding, we review the first example in Section 3.1.1 (see Figure 3.1) in the butterfly network. The source node S multicasts two packets, b_1 and b_2, to both the destination nodes t_1 and t_2. With the traditional approach ten transmissions are needed for communicating two packets of data, whereas there is one less transmission by using network coding in the same network. The butterfly network clearly illustrates that network coding can improve throughput for multicast in a wired network.

In wireless networks, it becomes much easier to find examples in which network coding yields a throughput advantage over routing. The example in Figure 3.3 illustrates that, using network coding, the relay M transmits only once instead of twice, therefore the downlink bandwidth can be reduced by 50%.

3.2.2 Robustness

Network coding has also been suggested as a means of improving robustness against random network failures since the destination can recover the original data (with high probability) once it has received sufficiently many correct packets, even if a large fraction of packets are lost [28, 21].

The reasons for packet loss are various in wireless networks, such as link failure, buffer overflow and collision. Network coding can be used to protect against packet loss, because the linear combination of packets is performed all over the network, not only at the source node. Thus the destination nodes can recover the original file with incomplete information they receive. Another method used in practical networks is called *erasure coding*, which introduces some redundancy information to each packet so that the original file can be recovered if the destination nodes receive a sufficient number of encoded packets. This is discussed further in Section 3.3.

Besides robustness against packet loss, network coding can provide protection against link failure. A natural solution is to transmit both a primary and a backup packet for each connection, and it is helpful for a fast recovery from link failure. However, double network bandwidth is occupied by using this method. Koetter and Médard [32] investigated the problem of network recovery from link failure, and proved that there exist coding strategies that provide maximally robust networks for the multicast setup, and that there exists a static network coding solution for network recovery under any failure pattern without rerouting.

3.3 Linear Network Coding

In this section, we focus on the introduction to *linear network coding*, which can offer the best possible benefits [35]. Linear network coding regards each packet of data as a vector over a certain base field and allows a node to apply a linear transformation to a vector before passing it on.

3.3.1 Encoding

Assume that k original packets are generated by a file. Each packet can be viewed as an n-dimensional vector $\bar{m}_i \in \mathbb{F}_q^n$, $i = 1, \ldots, k$, where q is prime. Before transmission, the source node creates k *augmented vectors* $m_1, \ldots, m_k$ as follows:

$$m_i = (\bar{m}_i, \overbrace{\underbrace{0, \ldots, 0, 1}_{i}, 0, \ldots, 0}^{k}) \in \mathbb{F}_q^{n+k},$$

that is, each original vector $\bar{m}_i$ is appended with a vector of length k containing a single '1' in the ith position. Typically, one chooses $k \ll n$. These augmented vectors are then sent by the source as innovative packets in the network.

Let $N := n + k$. Upon receiving k packets $m_1, \ldots, m_k \in \mathbb{F}_q^N$, a node forms a new packet (vector) $m = \sum_{j=1}^{k} \alpha_j m_j$, where $\alpha_j \in \mathbb{F}_q$ for $j = 1, \ldots, k$. The coefficient vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$ is called an *encoding vector*, and the encoded data $m = \sum_{j=1}^{k} \alpha_j m_j$ is called an *information vector* [17].

Encoding can be performed recursively for packets that have been encoded. Considering that an intermediate node receives k encoded packets $\mathbf{w}_1, \ldots, \mathbf{w}_k$ which are linear combinations of packets $m_1, \ldots, m_k$, the node can generate a new packet $\mathbf{w} = \sum_{j=1}^{k} \beta_j \mathbf{w}_j$ by picking a new coefficient vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_k)$. In fact, the new packet $\mathbf{w}$ is still a linear combination of packets $m_1, \ldots, m_k$.

3.3.2 Decoding

When any such node receives k linearly independent encoded packets $\mathbf{w}_1, \ldots, \mathbf{w}_k$, it can recover the original file as follows [35, 8]: for a received vector $\mathbf{w}_i = (\bar{\mathbf{w}}_i, \tilde{\mathbf{w}}_i) \in \mathbb{F}_q^N$, where $\bar{\mathbf{w}}_i$ is the left-most n positions of the vector, and $\tilde{\mathbf{w}}_i$ the right-most k positions. The receiver first computes a $k \times k$ matrix G such that

$$\mathbf{G} = \begin{pmatrix} \tilde{\mathbf{w}}_1 \\ \vdots \\ \tilde{\mathbf{w}}_k \end{pmatrix}.$$

The matrix G is invertible as long as all the received vectors are correct, and the original file $\bar{m}_1, \ldots, \bar{m}_k$ can be obtained by

$$\begin{pmatrix} \bar{m}_1 \\ \vdots \\ \bar{m}_k \end{pmatrix} = \mathbf{G}^{-1} \cdot \begin{pmatrix} \bar{\mathbf{w}}_1 \\ \vdots \\ \bar{\mathbf{w}}_k \end{pmatrix}.$$

Note that the recipient does not need to know the coefficient vectors $\boldsymbol{\alpha}$ used by any intermediate node in the network in order to recover the file. On the other hand, if all the coefficient vectors $\boldsymbol{\alpha}$ are known to the recipient, then the matrix G can be computed in advance and the scheme can be built on the original file vectors $\bar{m}_1, \ldots, \bar{m}_k$ rather than on the augmented vectors $m_1, \ldots, m_k$. For security purposes, these augmented vectors are necessary.

3.4 Network Security

As we have seen, network coding can provide significant benefits in practical networks, such as increasing throughout, improved robustness and higher reliability. However, from a security point of view, network coding also raises serious concerns [21].

Firstly, network coding offers a natural way to take advantage of multipath diversity for protecting data packets from an *eavesdropping attack*, since nodes in the network send linear combinations of packets instead of uncoded data. Information spread makes it more difficult for the attackers to eavesdrop. Cai and Yueng [13] discussed how to design secure network coding in wiretap networks, where some links are tapped by attackers. More specifically, the source combines the original data with random information and designs a network code in such a way that only the receivers are able to decode the original packets.

Without additional protection in networks, an attacker can arbitrarily modify data packets without being detected by the recipient. This is called the *malicious node attack* for networks. Fortunately, due to the randomly linear combinations made by the intermediate nodes, the attacker cannot control the outcome of the decoding process at the destination, without knowing all other coded packets the destination will receive. Therefore, network coding can provide protection against the malicious node attack and make the *man-in-the-middle attack* almost impossible.

However, the malicious modification to packets can cause a more serious attack, named *pollution attack*, in which attackers inject modified packets involving some errors (called corrupted packets) into the network. Since intermediate nodes forward packets coded from their received packets, as long as at least one of the input packets is corrupted, the errors will spread and all output packets forwarded by the node will be corrupted. Moreover, traditional error correction codes that deal with a limited proportion of corrupted packets are less effective.

More recently, a homomorphic signature scheme has been proposed, which is based on elliptic curves and allows nodes to sign linear combinations of packets. Under the assumption of the hardness of the computational co-Diffie-Hellman problem on elliptic curves, the proposed signature scheme prevents the forging of signatures and detects corruption of packets. We will introduce it and related signature schemes in the following chapters.

Chapter 4

Related Signature Schemes for Network Coding

A cryptographic function $f : D \to R$ is defined to be *homomorphic* if, given $f(x)$ and $f(y)$ for any $x, y \in D$, any one can compute $f(xy)$ without access to the private key. The basic RSA signature scheme is homomorphic: the message m is signed as $\sigma_d(m) = m^d \pmod{n}$, where d is the private key, and for two different messages m_1 and m_2, we have $\sigma_d(m_1 \cdot m_2) \equiv (m_1 \cdot m_2)^d \equiv m_1^d \cdot m_2^d = \sigma_d(m_1) \cdot \sigma_d(m_2) \pmod{n}$. That means anyone can get the valid signature of the message $m_1 \cdot m_2$ without any need of the private key, only if he can capture the signatures $\sigma_d(m_1)$ and $\sigma_d(m_2)$. The homomorphic property was previously considered to be a drawback and should be avoided in signature schemes. However, in recent years a wide range of positive applications of this property were found [39, 40, 18, 19]. An important application is homomorphic signatures for network coding, which can be used to protect against malicious code(s) and pollution attacks.

In this chapter, we firstly introduce homomorphic hashing and the definitions of homomorphic signatures. Then six homomorphic signature schemes for network coding are presented:

- Boneh-Freeman-Katz-Waters' two schemes $\mathcal{S}_1$ and $\mathcal{S}_2$ [8].
- Charles-Jain-Lauter's scheme [15].
- Yun-Cheon-Kim's RSA-based scheme [45].
- Gennaro-Katz-Krawczyk-Rabin's RSA-based scheme [23].
- Zhao-Kalker-Medard-Han's scheme [46].

In the last section, we discuss the security and compare the memory and communication overhead of five of these homomorphic signature schemes.

4.1 Krohn et al.'s Homomorphic Hashing Schemes for Rateless Erasure Codes

Krohn et al. [33] suggested homomorphic hashing for preventing pollution attacks in peer-to-peer content distribution networks (P2P-CDNs). In these hashing schemes, the sender computes a hash $h_i = H(\bar{m}_i)$ of each packet of a file so that the verifier can check the integrity of any received packet. Krohn et al. [33] proposed two authentication protocols based on a homomorphic collision-resistant hash function (CRHF): global homomorphic hashing and per-publisher homomorphic hashing.

In global homomorphic hashing [33], an original file F can be separated into k packets $(\bar{m}_1, \ldots, \bar{m}_k)$, and each packet $\bar{m}_i = (m_{i1}, \ldots, m_{in})^T \in \mathbb{F}_q^n$ is mapped to $H(\bar{m}_i)$ by a one-way hash function with the global parameters $G = (p, q, \mathbf{g})$, where p and q are two large primes with $q|(p-1)$, and $\mathbf{g} = (g_1, g_2, \ldots, g_n)$ is an n-dimensional vector composed of elements of order q in $\mathbb{F}_p^*$. Typically, a file can be represented as an $n \times k$ matrix as follows:

$$F = (\bar{m}_1, \ldots, \bar{m}_k) = \begin{pmatrix} m_{1,1} & \ldots & m_{k,1} \\ \vdots & \ddots & \vdots \\ m_{1,n} & \ldots & m_{k,n} \end{pmatrix} \in \mathbb{F}_q^{n \times k}.$$

We can compute:

$$\bar{m}_i + \bar{m}_j = (m_{i1} + m_{j1}, \ldots, m_{in} + m_{jn}) \in \mathbb{F}_q^n. \tag{4.1}$$

This scheme consists of five algorithms: precoding, encoding, hashing generation, hash verification and decoding.

Precoding

The precoding stage is to construct a binary $k \times (k + k\delta t)$ matrix $Y = (I|P)$, where I is the $k \times k$ identity matrix, and each column of P is the sum (via xor) of some number of the columns of the identity matrix I. Then $F' = FY$ is the precoded file. The first k columns of $F' = (\bar{m}_1, \ldots, \bar{m}_k, \ldots, \bar{m}_{k+k\delta t})$ are the message packets and the last $k\delta t$ columns are auxiliary packets, each of which is the sum (via xor) of some number of the message packets. Auxiliary packets can be generated by a public deterministic algorithm of the input packet number k, and the parameters t and δ ($\delta < 1$), which are fixed beforehand and guarantee that the original message fails to be recovered completely with probability less than δ^t [36, 33].

Encoding

Let $n' = k + k\delta t$. The encoder randomly chooses an n'-dimensional bit vector $X = (x_1, \ldots, x_{n'})^T \in \{0,1\}^{n'}$, and computes $\mathbf{c} = F'X = \sum_{j=1}^{n'} x_j \bar{m}_j \in \mathbb{F}_q^n$. $\mathbf{c} = (c_1, \ldots, c_n)$ is called a check block, where $c_J = \sum_{j=1}^{n'} x_j m_{jJ}, J = 1, \ldots, n$. Then $\langle X, \mathbf{c} \rangle$ is sent to the receiver.

Hashing Generation

For any message packet $\bar{m}_i \in \mathbb{F}_q^n$, $i = 1, \ldots, k$, its hash is defined as follows:

$$h(\bar{m}_i) = \prod_{J=1}^{n} g_J^{m_{iJ}} \in \mathbb{F}_p^*, \tag{4.2}$$

where $h(\cdot)$ is a homomorphic hash function. The sender computes hash values of all message packets, obtains

$$H(F) = (h(\bar{m}_1), h(\bar{m}_2), \ldots, h(\bar{m}_k)), \tag{4.3}$$

and sends $\langle H(F), G \rangle$ to the receiver. Typically, $n \gg \log p / \log q$ so that the length of each hash value is much shorter than that of each message packet.

However, we assume a reliable channel, in which data can be transmitted without any error, for pre-distributing the hashes before the receiver can verify the hashing and decode for the original file. We can use a standard signature scheme also to achieve a secure solution without a reliable channel (see Section 4.2.2).

Hashing Verification

Since auxiliary packets are generated by a public deterministic algorithm of the input packet number k and the pre-fixed encoding parameters t and δ, the receiver can compute Y and then obtain the hash of F' after receiving $\langle X, \mathbf{c} \rangle$ and $\langle H(F), G \rangle$:

$$H(F') = H(FY) = H(F) \cdot Y = (h(\bar{m}_1), \ldots, h(\bar{m}_k), \ldots, h(\bar{m}_{n'})). \tag{4.4}$$

Then the receiver computes $h(\mathbf{c}) = \prod_{J=1}^{n} g_J^{c_J}$ and verifies if

$$h(\mathbf{c}) = \prod_{j=1}^{n'} h(\bar{m}_j)^{x_j}. \tag{4.5}$$

Correctness

$$\begin{aligned}
\prod_{j=1}^{n'} h(\bar{m}_j)^{x_j} &= \prod_{j=1}^{n'} \left(\prod_{J=1}^{n} g_J^{m_{jJ}} \right)^{x_j} \\
&= \prod_{J=1}^{n} \left(\prod_{j=1}^{n'} g_J^{m_{jJ} x_j} \right) \\
&= \prod_{J=1}^{n} g_J^{\sum_{j=1}^{n'} m_{jJ} x_j} \\
&= \prod_{J=1}^{n} g_J^{c_J} \\
&= h(\mathbf{c}). \qquad (4.6)
\end{aligned}$$

Decoding

After receiving n' check blocks $(\mathbf{c}_1, \ldots, \mathbf{c}_{n'})$ and corresponding $X_1, \ldots, X_{n'}$, where $X_j = (x_{j1}, \ldots, x_{jn'})$, the receiver can decode by solving the following equations:

$$\begin{pmatrix} x_{1,1} & \cdots & x_{1,n'} \\ \vdots & \ddots & \vdots \\ x_{n',1} & \cdots & x_{n',n'} \end{pmatrix} \cdot \begin{pmatrix} \bar{m}_1 \\ \vdots \\ \bar{m}_{n'} \end{pmatrix} = \begin{pmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_{n'} \end{pmatrix}.$$

Then, the original file can be recovered from the solutions $(\bar{m}_1, \ldots, \bar{m}_{n'})$.

In the pre-publisher hashing scheme [33], each publisher, who wishes to distribute a file, can pick individual hash parameters $G = (p, q, \mathbf{g})$, thus different publishers can generate different hashes of the same file F. To generate $\mathbf{g}$, a publisher picks a random $g \in_R \mathbb{F}_p^*$ of order q, and an n-dimensional vector $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{F}_q^n$, and then computes $\mathbf{g} = g^{\mathbf{v}} = (g^{v_1}, \ldots, g^{v_n})$. The hashes of the file F can be written as

$$H(F) = g^{\mathbf{v}F} = (g^{\mathbf{v}\bar{m}_1}, \ldots, g^{\mathbf{v}\bar{m}_k}), \qquad (4.7)$$

where $g^{\mathbf{v}\bar{m}_i} = \sum_{l=1}^{n} v_l m_{il}, i = 1, \ldots, k$. Thus, to obtain the full file hash $H(F)$, the publisher needs to perform only one modular exponentiation for each message packet. So the pre-publisher hashing scheme enables publishers to generate hashes more efficiently with the same verification overhead.

4.2 Definitions of Homomorphic Signatures

The homomorphic property in signature schemes was first presented in a positive way by Rivest in 2000 [39]. Rivest presented two new signature schemes: a prefix aggregation signature scheme and a transitive signature scheme, both of which take advantage of the homomorphic property. In the first scheme, anyone can compute the signature $\sigma_{SK}(m)$ from $\sigma_{SK}(m \parallel 0)$ and $\sigma_{SK}(m \parallel 1)$ without any need for the private key SK. For instance, in a tree-based routing network, in which each parent route node has two child route nodes, if the child node x_0 can route to IP addresses of the form $(100 * **)$, and another child node x_1 can route to IP addresses of the form $(101 * **)$, then the parent node x can route to IP addresses of the form $(10 * * * *)$. The scheme also has another property: the child signature $\sigma_{SK}(m \parallel 1)$ can be computed from signatures $\sigma_{SK}(m)$ and $\sigma_{SK}(m \parallel 0)$. An open problem was left to find a scheme satisfying that the signature $\sigma_{SK}(m)$ can be computed from $\sigma_{SK}(m \parallel 0)$ and $\sigma_{SK}(m \parallel 1)$, but the signature $\sigma_{SK}(m \parallel 1)$ is not computable from $\sigma_{SK}(m)$ and $\sigma_{SK}(m \parallel 0)$.

Rivest's second signature scheme [39] is a so-called transitive signature scheme on an undirected graph. A graph $G(E, V)$ is *transitively closed* if for any edges $(u, v) \in E$ and $(v, w) \in E$, there exists the edge $(u, w) \in E$. Then the *transitive closure* of $G(E, V)$ is a graph such that there is an edge for any two vertices in V. In the transitive signature scheme on an undirected graph, from two signatures $\sigma_{SK}((u, v))$ and $\sigma_{SK}((v, w))$ on two edges (u, v) and (v, w), a valid signature $\sigma_{SK}((u, w))$ on the edge (u, w) in the transitive closure can be computed without any need for the private key SK. An open problem is to find a similar signature scheme for directed graphs.

Rivest also proposed an open problem of finding a concatenation signature scheme, in which a valid signature $\sigma_{SK}(m_1 \parallel m_2)$ can be computed from two signatures $\sigma_{SK}(m_1)$ and $\sigma_{SK}(m_2)$ without any need for the private key SK.

For a message space $\mathcal{M}$, a signature space $\mathcal{S}$, a set of public keys $\mathcal{PK}$ and a set of private keys $\mathcal{SK}$, a signature scheme includes a signing algorithm $\mathbf{Sign} : \mathcal{SK} \times \mathcal{M} \to \mathcal{S}$ and a verifying algorithm $\mathbf{Verify} : \mathcal{PK} \times \mathcal{M} \times \mathcal{S} \to \{0, 1\}$ so that if the verifying algorithm outputs 1, the signature is accepted; otherwise, it is rejected. For a binary operation $\odot : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$, and a set $S \subseteq \mathcal{M}$, $\mathrm{span}_{\odot}(S)$ denotes the smallest set T with $S \subseteq T$ and $m_1 \odot m_2 \in T$ for all $m_1, m_2 \in T$. Johnson et al. [29] defined the notion of a homomorphic signature scheme as follows.

Definition 4.2.1 (Homomorphic Signature Scheme [29]) *A signature scheme is homomorphic with respect to $\odot$ if it comes with an efficient family of binary operations $\otimes : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ so that $\sigma_1 \otimes \sigma_2 = \mathbf{Sign}(SK, m_1 \odot m_2)$ for all $\sigma_1, \sigma_2 \in \mathcal{S}, m_1, m_2 \in \mathcal{M}$, and $SK \in \mathcal{SK}$ satisfying $\mathbf{Verify}(PK, m_1, \sigma_1) = \mathbf{Verify}(PK, m_2, \sigma_2) = 1$ with $PK \in \mathcal{PK}$.*

For the security of homomorphic signature schemes, the standard existential unforgeability does not hold, since in a homomorphic signature scheme, given two signatures $\mathbf{Sign}(SK, m_1)$ and $\mathbf{Sign}(SK, m_2)$ on the messages m_1 and m_2, one can generate a valid signature on the message $m = m_1 \odot m_2$ without any need for the private key SK. So Johnson et al. [29] gave a new definition of security for homomorphic signature schemes as follows.

Definition 4.2.2 (Existential Unforgeability [29]) *A homomorphic signature scheme is $(t, q, \epsilon)-$secure against existential forgeries with respect to $\odot$ if every adversary $\mathcal{A}$ making at most q chosen-message queries and running in time at most t has advantage* $\mathrm{Adv}(\mathcal{A}) \leq \epsilon$. *The advantage of an adversary $\mathcal{A}$ is defined as the probability that, after queries on the message $m_1, \ldots, m_q$, $\mathcal{A}$ outputs a valid signature $\langle m, \sigma \rangle$ on some message $m \notin \mathrm{span}_{\odot}(m_1, \ldots, m_q)$. In other words,*

$$\mathrm{Adv}(\mathcal{A}) = \Pr[(\mathcal{A}(m_1, \ldots, m_q) = \langle m, \sigma \rangle) \wedge (\mathbf{Verify}(PK, m, \sigma) = 1) \wedge (m \notin \mathrm{span}_{\odot}(m_1, \ldots, m_q))].$$

4.2.1 Definitions of Homomorphic Signatures for Network Coding

In linear network coding system, to prevent the malicious modifications and pollution attacks, what is needed is a way for intermediate nodes to be able to verify the validity of incoming message packets. Since we need to guarantee the integrity of the message packets in the transmissions, the regular signature by the source node on the original message is not sufficient, so we introduce a new concept, the network coding signature, to solve the problem in the network coding system. Here is the formal definition of network coding signature given by Boneh et al. [8]:

Definition 4.2.3 (Network Coding Signature [8]) *A network coding signature scheme is composed of four probabilistic, polynomial-time algorithms:* $\mathbf{Setup}, \mathbf{KeyGen}, \mathbf{Sign}, \mathbf{Verify}$ *as follows:*

- $\mathbf{Setup}(l, N)$. *Given a security parameter l and an integer N, outputs a prime q.*
- $\mathbf{Keygen}(q)$. *Given the prime q, let $\mathbb{F}_q$ be a finite field, and output the public-private key pair (PK, SK).*
- $\mathbf{Sign}(SK, id, V)$. *For a private key SK, a file identifier $id \in \{0,1\}^*$, and a k-dimensional subspace $V \subset \mathbb{F}_q^N$ (with $0 < k < N$) described as a set of basis vectors $\{m_1, \ldots, m_k\}$, outputs a signature σ.*

– **Verify**$(PK, id, \boldsymbol{w}, \sigma)$. *On input a public key* PK*, an identifier* $id \in \{0,1\}^*$*, a vector* $\boldsymbol{w} = \sum_{i=1}^{k} \alpha_i m_i \in \mathbb{F}_q^N$*, and a signature* σ*, outputs either 0 (reject) or 1 (accept).*

First, we give a weak definition of security for a network coding signature. To prove the security of the network coding signature against existential forgery under the chosen plaintext attack, the following game between a challenger and an adversary $\mathcal{A}$ can be used:

– **Setup**. The challenger runs algorithm **KeyGen** to obtain the public key PK and the private key SK, and the adversary $\mathcal{A}$ is given PK.

– **Queries**. $\mathcal{A}$ requests signatures with PK on at most q_s vectors $m_1, m_2, \ldots, m_{q_s} \in \mathbb{F}_q^N$ of her choice, and the challenger responds to each query with a signature σ_i.

– **Output**. $\mathcal{A}$ outputs a signature σ and a vector $m \in \mathbb{F}_q^N$ $(m \notin \text{span}(m_1, m_2, \ldots, m_{q_s}))$. The adversary $\mathcal{A}$ succeeds if the verification goes through with her outputs.

We define $\text{Adv}(\mathcal{A})$ to be the probability that $\mathcal{A}$ succeeds in the above game.

Definition 4.2.4 (Weak Secure Network Coding Signature) *A network coding signature scheme is* weakly secure *against existential forgeries under chosen plaintext attack, if the probability that any probabilistic, polynomial-time adversary* $\mathcal{A}$ *can succeed in the above game is negligible; that is,* $\text{Adv}(\mathcal{A}) \leq \epsilon$.

Now we introduce a definition of secure network coding signature given by Boneh et al. [8], which is stronger than the security definition above, since a file identifier $id \in \{0,1\}^*$ is applied for each file in the following game between a challenger and an adversary $\mathcal{A}$:

– **Setup**. The challenger runs algorithm **KeyGen** to obtain the public key PK and the private key SK, and the adversary $\mathcal{A}$ is given PK.

– **Queries**. Proceeding adaptively, $\mathcal{A}$ requests signatures with PK on at most q_s vector subspaces $V_1, V_2, \ldots, V_{q_s} \subset \mathbb{F}_q^N$ by her choice, and the challenger randomly chooses $id_i \in \{0,1\}^*$ and responds to each query with a signature $\sigma_i = \text{Sign}(SK, id_i, V_i)$.

– **Output**. $\mathcal{A}$ outputs an identifier $id \in \{0,1\}^*$, a signature σ and a vector $m \in \mathbb{F}_q^N$.

The adversary wins this game above in the following two cases:

(1) **Verify**$(PK, id, m, \sigma) = 1$ and $id \neq id_i$ for all i;

(2) **Verify**$(PK, id, m, \sigma) = 1$ and $id = id_i$ for some i, but $m \notin V_i$.

We define $\mathrm{Adv}(\mathcal{A})$ to be the probability that $\mathcal{A}$ succeeds in the above game.

Definition 4.2.5 (Secure Network Coding Signature [8]) *A network coding signature scheme is* secure *against existential forgeries under chosen plaintext attack, if the probability that any probabilistic, polynomial-time adversary $\mathcal{A}$ wins in the above game is negligible; that is,* $\mathrm{Adv}(\mathcal{A}) \leq \epsilon$.

In the most of current work on network coding signatures, the homomorphic property is used in the signature schemes for network coding such that the signature on a linear combination of message packets results in a corresponding homomorphic combination of signatures on each message packet. The following definitions of the homomorphic signatures for network coding can completely describe the property and the security.

Definition 4.2.6 (Homomorphic Signature for Network Coding [8]) *A homomorphic signature scheme for network coding is defined by five probabilistic, polynomial-time algorithms:* **Setup**, **KeyGen**, **Sign**, **Combine**, **Verify***:*

- **Setup**(l, N). *Given a security parameter l and an integer N, this algorithm outputs a prime q.*
- **Keygen**(q). *Given the prime q, let $\mathbb{F}_q$ be a finite field, and output the public-private key pair (PK, SK).*
- **Sign**(SK, id, m_i). *For a secret key SK, a file identifier $id \in \{0,1\}^*$, and a packet $m_i \in \mathbb{F}_q^N$, this algorithm outputs a signature σ_i on m_i.*
- **Combine**$(PK, id, \{(\alpha_i, \sigma_i)\}_{i=1,\ldots,k})$. *For a public key PK, a file identifier id, and a set of tuples $\{(\alpha_i, \sigma_i)\}_{i=1,\ldots,k}$, where σ_i is a signature on m_i, with $\alpha_i \in \mathbb{F}_q$, this algorithm outputs a signature σ on $\mathbf{w} = \sum_{i=1}^{k} \alpha_i m_i$.*
- **Verify**$(PK, id, \boldsymbol{w}, \sigma)$. *Upon obtaining a public key PK, an identifier $id \in \{0,1\}^*$, a packet $\boldsymbol{w} = \sum_{i=1}^{k} \alpha_i m_i \in \mathbb{F}_q^N$, and a signature σ, this algorithm outputs either 0 (reject) or 1 (accept).*

A homomorphic signature for network coding is a special case of network coding signature ([8], Lemma 3), since if each σ_i is a valid homomorphic network coding signature on the vector m_i, then σ is a valid network coding signature on any linear combination $\sum_{i=1}^{k} \alpha_i m_i$.

4.3 Boneh et al.'s Signatures on a Linear Subspace for Network Coding

Boneh et al. [8] proposed two signature schemes for network coding to prevent malicious modification of data. The first scheme is a homomorphic signature scheme on linearly independent packets, and the security proof is given in the random oracle model. The second signature scheme can be viewed as signing the linear subspace V spanned by the augmented basis $m_1, \ldots, m_k$ which are independent packets received. The second signature scheme does not rely on random oracles but uses a standard signature scheme based on the assumption of the hardness of the Discrete Logarithm (DL) Problem. Here, a standard signature scheme is a signature scheme which is proven secure in the standard model [5, 6].

Boneh et al. ([8], Lemma 3) showed that a homomorphic network coding signature scheme $\mathcal{S}_2$ is secure against forgeability under chosen-plaintext attack if the network coding signature scheme $\mathcal{S}_1$ constructed from $\mathcal{S}_2$ is secure against forgeability under chosen-plaintext attack. But the converse does not hold.

4.3.1 A Homomorphic Network Coding Signature Scheme $\mathcal{S}_2$ with Random Oracles

The homomorphic network coding signature scheme $\mathcal{S}_2 = \{\textbf{Setup}, \textbf{KeyGen}, \textbf{Sign}, \textbf{Combine}, \textbf{Verify}\}$ with random oracles is described as follows:

- **Setup**(l, N). Given a security parameter l and a positive integer N, choose cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of some large prime order q of l-bit length, and let $\mathbb{G}_T$ be a multiplicative group of the same size as $\mathbb{G}_1$ and $\mathbb{G}_2$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing (see Section 2.2.2), and $H : \{0,1\}^l \times \{0,1\}^l \to \mathbb{G}_1$ be a hash function, viewed as a random oracle. g is a generator in $\mathbb{G}_2$. params $= \langle \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, H, g \rangle$ is the public system parameter.
- **KeyGen**(params). Choose $s \in \mathbb{F}_q^*$ as the private key SK, and set $u = g^s$ as the public key.
- **Sign**(SK, id, m_i). For the packets $m_i = (m_{i1}, \ldots, m_{iN}) \in \mathbb{F}_q^N$, for $i = 1, 2, \ldots, k$, a secret key $SK = s$, and an identifier $id \in \{0,1\}^l$, generate the signature

$$\sigma_i = (\prod_{j=1}^{N} H(id, j)^{m_{ij}})^s. \tag{4.8}$$

– **Combine**$(PK, id, \{(\alpha_i, \sigma_i)\}_{i=1,\ldots,k})$. Given a public key PK, an identifier id, and $\{(\alpha_i, \sigma_i)\}_{i=1,\ldots,k}$ with $\alpha_i \in \mathbb{F}_q$, $\sigma = \prod_{i=1}^{k} \sigma_i^{\alpha_i}$ is the signature on the vector $\mathbf{w} = \sum_{i=1}^{k} \alpha_i m_i$, which is a linear combination of the packets m_i.

– **Verify**$(PK, id, \mathbf{w}, \sigma)$. Given a public key $PK = (H, g, u)$, an identifier id, after receiving a signature σ, and a vector $\mathbf{w} \in \mathbb{F}_q^N$, check whether

$$e(\sigma, g) = e(\prod_{j=1}^{N} H(id, j)^{w_j}, u). \tag{4.9}$$

If yes, this algorithm outputs 1; otherwise it outputs 0.

Correctness

If the vector $\mathbf{w}$ and the signature σ received are valid, then the equation (4.9) holds, since

$$\begin{aligned}
e(\sigma, g) &= e\left(\prod_{i=1}^{k} \sigma_i^{\alpha_i}, g\right) \\
&= e\left(\prod_{i=1}^{k} \left(\prod_{j=1}^{N} H(id, j)^{m_{ij}}\right)^{s\alpha_i}, g\right) \\
&= e\left(\prod_{i=1}^{k}\prod_{j=1}^{N} H(id, j)^{m_{ij} s\alpha_i}, g\right) \\
&= e\left(\prod_{i=1}^{k}\prod_{j=1}^{N} H(id, j)^{m_{ij}\alpha_i}, g^s\right) \\
&= e\left(\prod_{j=1}^{N} H(id, j)^{\sum_{i=1}^{k} m_{ij}\alpha_i}, u\right) \\
&= e\left(\prod_{j=1}^{N} H(id, j)^{w_j}, u\right).
\end{aligned} \tag{4.10}$$

In fact, this signature scheme is based on the aggregate signature schemes of Boneh et al. [9]. The signer in this scheme uses the hash value H of the identifier id to sign the packets m_i. In this scheme both the signature σ and the public key have size $O(l)$, where l is a cryptographic security parameter. The security of this signature scheme is based on the assumption of the hardness of the co-computational Diffie-Hellman (co-CDH) Problem.

4.3.2 A Network Coding Signature Scheme $\mathcal{S}_1$ without Random Oracles

Firstly, define a standard signature scheme by $\mathcal{S}_0 = \{\mathbf{Gen}_0, \mathbf{Sign}_0, \mathbf{Verify}_0\}$ with public-private key pair (PK_0, SK_0). The network coding signature scheme $\mathcal{S}_1 = \{\mathbf{Setup}, \mathbf{Sign}, \mathbf{Verify}\}$ is described as follows:

- **Setup**(l, N). Given a security parameter l and a positive integer $N > k$, choose a cyclic group $\mathbb{G}_1$ of prime order q of l-bit length, and params $= \langle N, k, q, \mathbb{G}_1 \rangle$ is the public system parameter.
- **KeyGen**(params). Choose generators $g_1, \ldots, g_N \in \mathbb{G}_1$. The public key is $PK = (g_1, \ldots, g_N, PK_0)$, and the private key is $SK = SK_0$.
- **Sign**(SK, id, V). Given a secret key SK, a file identifier id, and a k-dimensional subspace V spanned by an augmented basis $\{m_1, \ldots, m_k\}$, where $m_i = (m_{i1}, m_{i2}, \ldots, m_{iN}) \in \mathbb{F}_q^N$ for $i = 1, 2, \ldots, k$, compute $\sigma_i = \prod_{j=1}^{N} g_j^{-m_{ij}} \in \mathbb{G}_1$ for $i = 1, \ldots, k$. Sign $id, \sigma_1, \ldots, \sigma_k$ using the signing algorithm in the standard signature scheme, that is, $\tau \leftarrow \mathbf{Sign}_0(SK, (id, \sigma_1, \ldots, \sigma_k))$. Output $\sigma = (\sigma_1, \ldots, \sigma_k, \tau)$.
- **Verify**$(PK, id, \mathbf{w}, \sigma)$. Upon obtaining a vector $\mathbf{w} = \sum_{i=1}^{k} \alpha_i m_i = (w_1, \ldots, w_N) \in \mathbb{F}_q^N$, a signature $\sigma = (\sigma_1, \ldots, \sigma_k, \tau)$, the public key PK and the identifier id, first check whether $\mathbf{Verify}_0(PK_0, (id, \sigma_1, \ldots, \sigma_k), \tau) = 1$ using the verification algorithm of the standard signature scheme $\mathcal{S}_0$. If no, output 0 (reject) and abort; if yes, then check whether $\left(\prod_{j=1}^{N} g_j^{w_j}\right)\left(\prod_{i=1}^{k} \sigma_i^{\alpha_i}\right) = 1$, where $w_j = \sum_{i=1}^{k} \alpha_i m_{ij}$, $j = 1, 2, \ldots, N$. If yes, output 1 (accept); otherwise, output 0 (reject).

Correctness

If the vector $\mathbf{w}$ and the signature σ received are valid, then we have the equation

$$\left(\prod_{j=1}^{N} g_j^{w_j}\right)\left(\prod_{i=1}^{k} \sigma_i^{\alpha_i}\right) = 1,$$

since

$$\begin{aligned}
\left(\prod_{j=1}^{N} g_j^{w_j}\right)\left(\prod_{i=1}^{k} \sigma_i^{\alpha_i}\right) &= \left(\prod_{j=1}^{N} g_j^{w_j}\right)\left(\prod_{i=1}^{k}(\prod_{j=1}^{N} g_j^{-m_{ij}})^{\alpha_i}\right) \\
&= \left(\prod_{j=1}^{N} g_j^{w_j}\right)\left(\prod_{j=1}^{N}\prod_{i=1}^{k} g_j^{-m_{ij}\alpha_i}\right) \\
&= \left(\prod_{j=1}^{N} g_j^{w_j}\right)\left(\prod_{j=1}^{N} g_j^{-\sum_{i=1}^{k}\alpha_i m_{ij}}\right) \\
&= \prod_{j=1}^{N} g_j^{w_j-\sum_{i=1}^{k}\alpha_i m_{ij}} \\
&= \prod_{j=1}^{N} g_j^{w_j-w_j} \\
&= 1. \qquad (4.11)
\end{aligned}$$

The network coding signature scheme $\mathcal{S}_1$ described above can be viewed as an instantiation of the scheme of Krohn et al. [33], who proposed authenticating network coding data using a homomorphic hash function (see Section 4.1). In fact, the network coding signature scheme $\mathcal{S}_2$ by Boneh et al. takes advantage of the standard signature scheme to authenticate all the hash values along with the file identifier, so that basis vectors from different files can also be combined. In this scheme, the signature σ has size $O(l(k+1))$, and the public key has size $O(l(n+1))$. This signature scheme is proven secure based on the assumption of the hardness of the DL Problem.

4.4 Charles et al.'s Signatures for Network Coding

Charles et al. [15] proposed a homomorphic signature scheme for network coding to detect pollution attacks (see Section 3.4). The homomorphic property is that, given valid signatures $\sigma_1, \ldots, \sigma_k$ on packets $m_1, \ldots, m_k$, the scheme can produce a valid signature σ on any linear combination of $m_1, \ldots, m_k$. That is, $\sigma(\sum_{i=1}^{k} \alpha_i m_i) = \prod_{i=1}^{k} \sigma_i(m_i)^{\alpha_i}$. The homomorphic signature scheme is described as follows:

– **Setup**(l, N). Given the security parameter l and an integer N, let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be multiplicative groups of some large prime order q of l-bit length. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing. Choose distinct $g_1, g_2, \ldots, g_N \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$, and define a

hash function $H_{g_1,g_2,\ldots,g_N} : \mathbb{F}_q^N \to \mathbb{G}_1$ as follows: for a vector $\mathbf{v} = (v_1, v_2, \ldots, v_N) \in \mathbb{G}_1$,

$$H_{g_1,g_2,\ldots,g_N}(\mathbf{v}) = \prod_{j=1}^{N} g_j^{v_j}.$$

Then the system parameters are params $= \langle e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, N, q, g_1, g_2, \ldots, g_N, h, H_{g_1,g_2,\ldots,g_N} \rangle$.

– **KeyGen**(params). For the packets $m_i = (m_{i1}, \ldots, m_{iN}) \in \mathbb{G}_1$, for $i = 1, 2, \ldots, k$, the signer chooses $(s_1, \ldots, s_N) \in \mathbb{F}_q^N$ as the private key SK, and the public key PK is $(g_1, g_2, \ldots, g_N, h, h_1, h_2 \ldots, h_N)$, where $h_j = h^{s_j}, j = 1, 2, \ldots, N$.

– **Sign**$(SK, g_1, g_2, \ldots, g_N, m_i)$. Any packet $m_i = (m_{i1}, \ldots, m_{iN}), i = 1, \ldots, k$ can be signed as follows:

$$\sigma_i = H_{s_1 g_1, s_2 g_2, \ldots, s_N g_N}(m_i) = \prod_{j=1}^{N} g_j^{m_{ij} s_j}, \ i = 1, \ldots, k. \tag{4.12}$$

– **Combine**$(PK, \{(\alpha_i, \sigma_i)\}_{i=1,\ldots,k})$. Suppose a vector $\mathbf{w}$ received is a linear combination $\mathbf{w} = \sum_{i=1}^{k} \alpha_i m_i$. Since the signing function is a homomorphism, the signature on $\mathbf{w}$ is

$$\sigma = \prod_{i=1}^{k} \sigma_i^{\alpha_i}. \tag{4.13}$$

– **Verify**$(PK, \mathbf{w}, \sigma)$. After receiving the packet $\mathbf{w} = (w_1, \ldots, w_N)$ appended to the signature σ, the verifier checks whether

$$\prod_{j=1}^{N} e(g_j^{w_j}, h_j) = e(\sigma, h). \tag{4.14}$$

Correctness

The correctness of the verification process is shown below:

$$\begin{aligned} e(\sigma, h) &= e\left(\prod_{i=1}^{k} \sigma_i^{\alpha_i}, h\right) \\ &= \prod_{i=1}^{k} e\left(\sigma_i^{\alpha_i}, h\right) \\ &= \prod_{i=1}^{k} e\left(g_j^{\alpha_i \sum_{j=1}^{N} m_{ij} s_j}, h\right) \\ &= \prod_{i=1}^{k} \prod_{j=1}^{N} e\left(g_j^{\alpha_i m_{ij} s_j}, h\right) \\ &= \prod_{j=1}^{N} e\left(g_j^{\sum_{i=1}^{k} \alpha_i m_{ij}}, h^{s_j}\right) \\ &= \prod_{j=1}^{N} e\left(g_j^{w_j}, h_j\right). \end{aligned} \tag{4.15}$$

The hash function H in this scheme is proven to be collision resistant, and the security of Charles et al.'s signature scheme is based on the assumption of the hardness of the co-CDH Problem. In this scheme, the signature σ has size $O(l)$, and the public key has size $O(l(n+k))$. Charles et al.'s signature scheme can only sign a single file, and after one file is distributed, the public key must be refreshed. If not, a malicious node can forge a valid signature on a message chosen by this node. Moreover, the computation of the verification is large, because the verifier needs to compute $(N+1)$ pairings. This restriction limits the scheme's applicability.

4.5 RSA-Based Homomorphic Signature Schemes for Network Coding

In this section, we discuss two RSA-based homomorphic signature schemes for network coding. As we discussed in the beginning of this chapter, the basic RSA scheme is homomorphic, however, there exist some pollution attacks on the RSA-based signature schemes for network coding. We will start with the definition of the Standard RSA Problem.

Definition 4.5.1 (Standard RSA Problem) *Given an RSA public key (N, e), and a ciphertext $C \equiv P^e \pmod N$, the RSA Problem is to compute P, where N is a product of two large primes, $2 < e < N$ and e is coprime to $\phi(N)$. C is chosen randomly in $[0, N)$.*

In the *Standard RSA Assumption*, we say that the RSA problem is intractable. That is, given the RSA public key (N, e) and a ciphertext C, it is infeasible to find the plaintext P such that $C \equiv P^e \pmod N$. The RSA problem reduces to factoring the integer N [37], since if one can factor N, he can compute the private key d, such that $ed = 1 \pmod{\phi(N)}$, from the public key (N, e), and then solve the RSA problem. However, we do not know whether the converse is true, that is, whether an algorithm for integer factoring can be efficiently constructed from an algorithm for solving the RSA Problem.

4.5.1 Yun et al.'s signature scheme

The security of the signature schemes in this section are all based on the Standard RSA Assumption. Firstly we introduce an RSA-based homomorphic signature scheme for network coding by Yun et al. [45] in the standard model, which is a revision of Yu et al.'s signature scheme [43].

- **Setup**(l, k, n). Given the security parameter l, choose two primes p and q of $l/2$-bit length, $q|(p-1)$, and pick $g_1, g_2, \ldots, g_{k+n}$ of order q in $\mathbb{Z}_p$.
- **KeyGen**$(g_1, g_2, \ldots, g_{k+n}, p, q)$. For the RSA public key $PK = (N, e)$ and private key $SK = (\phi(N), d)$, where $ed \equiv 1 \pmod{\phi(N)}$ and $N = pq$. Suppose that all nodes know $g_1, g_2, \ldots, g_{k+n}$ as well as the RSA public key PK.
- **Sign**$(SK, g_1, g_2, \ldots, g_{k+n}, m_i)$. For each augmented message packet m_i, $i = 1, 2, \ldots, k$, which can be viewed as a $(k+n)$-dimensional vector $m_i = (m_{i1}, m_{i2}, \ldots, m_{i(k+n)}) \in \mathbb{Z}_q^{k+n}$, the source node calculates the signature

$$\sigma_i = (\prod_{j=1}^{k+n} g_j^{m_{ij}})^d \pmod N,$$

  and appends σ_i to m_i.
- **Combine**$(PK, \{(\alpha_i, \sigma_i)\}_{i=1,\ldots,k})$. The intermediate nodes can calculate the signature on the linear combination of original message packets, $\mathbf{w} = \sum_{i=1}^{k} \alpha_i m_i \in \mathbb{Z}_q^{k+n}$ as

$$\sigma = \prod_{i=1}^{k} \sigma_i^{\alpha_i} \pmod N.$$

– **Verify**$(PK, \mathbf{w}, \sigma, g_1, g_2, \ldots, g_{k+n})$. To verify the signature σ on $\mathbf{w} = (w_1, w_2, \ldots, w_{k+n})$, the destination node needs to check whether

$$\sigma^e \pmod N = \prod_{j=1}^{k+n} g_j^{w_j} \pmod N.$$

Correctness

If the vector $\mathbf{w}$ and the signature σ received are valid, then the verification process outputs 1, since:

$$\begin{aligned}
\sigma^e \pmod N &= \left(\prod_{i=1}^{k} \sigma_i^{\alpha_i}\right)^e \pmod N \\
&= \left(\prod_{i=1}^{k} \left(\prod_{j=1}^{k+n} g_j^{m_{ij}}\right)^{d\alpha_i}\right)^e \pmod N \\
&= \prod_{i=1}^{k} \left(\prod_{j=1}^{k+n} g_j^{m_{ij}}\right)^{\alpha_i} \pmod N \\
&= \prod_{i=1}^{k} \left(\prod_{j=1}^{k+n} g_j^{\alpha_i m_{ij}}\right) \pmod N \\
&= \prod_{j=1}^{k+n} g_j^{\sum_{i=1}^{k} \alpha_i m_{ij}} \pmod N \\
&= \prod_{j=1}^{k+n} g_j^{w_j} \pmod N.
\end{aligned}$$

In this homomorphic signature scheme, a pollution attack can be derived because of the weakness of the homomorphic property, which we will discuss in detail in Chapter 5.

4.5.2 Gennaro et al.'s signature scheme

Gennaro et al. [23] also proposed an RSA-based homomorphic signature scheme for network coding in the random oracle model. Before signing each packet, the source needs to bound the coordinates of any vectors $m_i = \bar{m}_i \parallel u_i = (m_{i1}, \ldots, m_{in}, u_{i1}, \ldots, u_{ik})$, $i = 1, 2, \ldots, k$, where u_i is the i-th unit vector (see Section 3.3.1). To encode the data using the linear

combination, we suppose that the coefficients are chosen uniformly from $\mathbb{Z}_p = \{0, \ldots, p-1\}$, where p is a prime of 8-bit length.

Let L be an upper bound on the path length from the source to the destination, and a bound $B = (kp)^L$ represents the largest possible value of a u-coordinate in any vector transmitted in the network. If M denotes an upper bound on the magnitude of the coordinates of the initial vectors $\bar{m}_i$, and then $B^* = BM$ is the upper bound on the magnitude of any coordinates in any vectors transmitted in the network.

- **Setup**(l). Given the security parameter l, let N of l-bit length be the product of two secure primes of the form $2n + 1$, where n is also a prime, and QR_N be the quadratic residues in $\mathbb{Z}_N^*$. The output parameters are params $= (N, M, B, B^*)$ as defined above.

- **KeyGen**(params). Choose the RSA public key (N, e) and the private key $SK = (\phi(N), d)$, where e is a prime larger than kB^*, N is a product of two primes a and b of half the size of N, and $ed \equiv 1 \pmod{\phi(N)}$. Pick n generators $g_1, g_2, \ldots, g_n$ in QR_N, and the public key is $PK = (N, e, g_1, g_2, \ldots, g_n)$. For a file identifier id, use a hash function $H : \{0,1\}^* \to QR_N$, and compute $h_i = H(i, id) \in QR_N$, $i = 1, 2, \ldots, k$.

- **Sign**$(SK, F, g_1, g_2, \ldots, g_n)$. If a file F is split into k packets $\bar{m}_1, \bar{m}_2, \ldots, \bar{m}_k$, the source node can generate the augmented vectors $m_i = \bar{m}_i \parallel u_i = (m_{i1}, \ldots, m_{in}, u_{i1}, \ldots, u_{ik})$, $i = 1, 2, \ldots, k$, where u_i is the i-th unit vector (see Section 3.3.1). Then the signature on m_i is defined by

$$\sigma_i = \left(\prod_{J=1}^{n} g_J^{m_{iJ}} \prod_{j=1}^{k} h_j^{u_{ij}} \right)^d \pmod{N}.$$

- **Combine**$(PK, \{(\alpha_i, \sigma_i)\}_{i=1,\ldots,k})$. Upon receiving packets $\mathbf{w}_i$ with the same file identifier id and valid signatures σ_i, the intermediate node first discards any $\mathbf{w}_i$ having a u-coordinate larger than B/kp or a $\bar{m}_i$-coordinates larger than B^*/kp. After that, the signature on the linear combination of non-discarded vectors, $\mathbf{w} = \sum_{i=1}^{k} \alpha_i m_i$, can be computed as

$$\sigma = \prod_{i=1}^{k} \sigma_i^{\alpha_i} \pmod{N}.$$

- **Verify**$(PK, \mathbf{w}, \sigma)$. To verify the signature σ on the vector $\mathbf{w} = \sum_{i=1}^{k} \alpha_i m_i = (w_1, w_2, \ldots, w_{n+k})$, we need to check whether

$$\sigma^e \pmod{N} = \prod_{J=1}^{n} g_J^{w_J} \prod_{j=1}^{k} h_j^{w_{n+j}} \pmod{N}.$$

Correctness

For any valid vector $\mathbf{w}$ and signature σ, the verification equation holds, since

$$\begin{aligned}
\sigma^e \pmod N &= \left(\prod_{i=1}^{k} \sigma_i^{\alpha_i}\right)^e \pmod N \\
&= \prod_{i=1}^{k}\left(\prod_{J=1}^{n} g_J^{m_{iJ}} \prod_{j=1}^{k} h_j^{u_{ij}}\right)^{ed} \pmod N \\
&= \prod_{i=1}^{k}\prod_{J=1}^{n} g_J^{m_{iJ}} \prod_{j=1}^{k} h_j^{u_{ij}} \pmod N \\
&= \prod_{J=1}^{n} g_J^{\sum_{i=1}^{k} m_{iJ}} \prod_{j=1}^{k} h_j^{\sum_{i=1}^{k} u_{ij}} \pmod N \\
&= \prod_{J=1}^{n} g_J^{w_J} \prod_{j=1}^{k} h_j^{w_{n+j}} \pmod N. \qquad (4.16)
\end{aligned}$$

There are some improvements of this scheme compared with Yun et al.'s scheme [45]:

1. For the RSA public key (N, e) and the private key $(\phi(N), d)$, where N is the product of two primes, choose a cyclic subgroup QR_N of quadratic residues in $\mathbb{Z}_N^*$, so we can choose n generators $g_1, g_2, \ldots, g_n$ in QR_N.

2. Using a file identifier id for each file, the immediate notes can identify and distinguish multiple files which are distributed simultaneously.

3. This signature scheme for network coding is performed over the integers rather than over a field as is traditionally done, so we can choose small integer coefficients for the linear combinations and reduce the bandwidth overhead for the transmission.

Under the RSA assumption, the scheme is proven to be a secure homomorphic signature for network coding with random oracles.

4.6 Zhao et al.'s Signatures for Network Coding

Zhao et al. [46] proposed a signature scheme for network coding to detect malicious codes in content distribution systems. This scheme takes advantage of the linearly independent

packets, and allows nodes to check validity and integrity the packets received without decoding. And this scheme supplies the security in the areas of detection and correction of Byzantine attacks [27]. Zhao et al. proved the security based on the assumption of the hardness of the (q, k, N)-Diffie-Hellman Problem (see Definition 2.4.6).

4.6.1 The Signature Scheme

The signature scheme for a subspace is defined by the following algorithms:

- **Setup**(l, k, n). p and q are two big primes, and $q|(p-1)$. Given the security parameter l, let $\mathbb{G} = \langle g \rangle$ be a cyclic subgroup of order q of l-bit length in $\mathbb{F}_p$, where g is a generator. A file is separated into k original packets $\bar{m}_i \in \mathbb{F}_q^n$, $i = 1, \ldots, k$, where q is prime, and the augmented vectors $m_1, \ldots, m_k$ span a subspace V of $\mathbb{F}_q^N$, where $N = n + k$ (see Section 3.3.1). The system parameters are params $= (p, q, \mathbb{G}, g, N)$
- **KeyGen**(params). Given the system parameters, the signer chooses $SK = \{s_j\}_{j=1,\ldots,N}$ as the private key, where the s_j are random elements in $\mathbb{F}_q^*$, and the public key is $PK = \{g^{s_j}\}_{j=1,\ldots,N}$ in $\mathbb{F}_p^N$.
- **Sign**$(SK, \{(\alpha_i, \sigma_i)\}_{i=1,\ldots,k})$. Using the vectors $m_1, \ldots, m_k \in \mathbb{F}_q^N$ and a linear combination of these vectors, $\mathbf{w} = \sum_{i=1}^k \alpha_i m_i$, the signer finds a vector $\mathbf{u} = (u_1, \ldots, u_N) \in \mathbb{F}_q^N$ orthogonal to the subspace V (using the Gram-Schmidt method, for example), that is,

$$m_i \cdot \mathbf{u} = \sum_{j=1}^{N} m_{ij} u_j = 0, \quad i = 1, \ldots, k. \tag{4.17}$$

  Then the signature is $\sigma = \{\sigma_j = u_j / s_j\}_{j=1,\ldots,N} \in \mathbb{F}_q^N$.
- **Verify**$(PK, \mathbf{w}, \sigma)$. After obtaining the vector $\mathbf{w} = \sum_{i=1}^k \alpha_i m_i = \{w_1, \ldots, w_N\}$, and the signature $\sigma = (\sigma_1, \ldots, \sigma_N)$, the receiver computes

$$d = \prod_{j=1}^{N} (g^{s_j})^{\sigma_j w_j} \pmod{p}, \tag{4.18}$$

  and verifies whether $d = 1$. If yes, this algorithm outputs 1; otherwise it outputs 0.

Correctness

For any valid vector $\mathbf{w}$ and signature σ, d is equal to 1 in the verification process, since

$$\begin{aligned} d &= \prod_{j=1}^{N} (g^{s_j})^{\sigma_j w_j} \\ &= \prod_{j=1}^{N} (g^{s_j})^{u_j w_j / s_j} \\ &= \prod_{j=1}^{N} g^{u_j w_j} \\ &= g^{\sum_{j=1}^{N} u_j w_j} \\ &= g^{\sum_{j=1}^{N} u_j \sum_{i=1}^{k} \alpha_i m_{ij}} \\ &= g^{\sum_{i=1}^{k} \alpha_i \sum_{j=1}^{N} u_j m_{ij}} \\ &= g^0 \\ &= 1. \end{aligned} \tag{4.19}$$

4.6.2 The Security Analysis of the Signature Scheme

For the security of this signature scheme, Zhao et al. [46] gave the definition of the (q, k, N)-Diffie-Hellman Problem (see Definition 2.4.6), and proved the problem is as hard as the Discrete Logarithm Problem in $\mathbb{F}_p$ for any $k < N - 1$. In the (q, k, N)-Diffie-Hellman Problem, $\{h_1, \ldots, h_N\}$ is a set of generators of G, and V is a linear subspace of rank k in $\mathbb{F}_q^N$ such that for every $\mathbf{v} \in V$, the equality $\prod_{i=1}^{N} h_i^{v_i} = 1$ holds.

The signature scheme requires the sender to know the entire file before the signature can be computed. In this scheme, the public key must be refreshed after the system distributes a single file. This is because using information from a previous download of files, a malicious node can generate a random vector and a corresponding signature which can also pass the verification function.

4.7 Discussion and Comparison

In this section, we discuss the pros and cons of some of these signature schemes described in this chapter, and then give a comparison of the computational complexities of these signature schemes.

Boneh et al.'s two signature schemes (see Section 4.3) can be used to provide cryptographic protection against pollution attacks, e.g., corrupted or modified intermediate nodes, eavesdroppers on all network traffic (see Section 3.4). The homomorphic signature scheme $\mathcal{S}_2$ (see Section 4.2.1) has low communication overhead since both the signature and the public key have sizes independent of the file size. The destination needs to receive a minimum number of packets to recover the original file, however the intermediate nodes need not to be aware of the entire file before computing the signature. This scheme is proven secure in the random oracle model under the security definition given in this paper (see Definition 4.2.5), assuming that the co-CDH Problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is infeasible. The network coding signature scheme $\mathcal{S}_1$ (see Section 4.2.2) is less efficient than $\mathcal{S}_2$ since both the signature and the public key are comparatively large. This scheme is based on a standard signature scheme, and if the standard signature scheme is existentially unforgeable under the chosen plaintext attack [4], the network coding signature is secure in the standard model under Definition 4.2.5 assuming hardness of the DL Problem in $\mathbb{G}_1$.

Charles et al.'s homomorphic signature scheme (see Section 4.5) provides a solution to the problem of detecting pollution attacks. In this scheme the signature has constant size, but the public key is large. This scheme is a variant of the aggregate signature scheme of Boneh et al. [9]. Under the assumption of the co-CDH Problem, this scheme is proven secure in the standard model according to the Definition 4.2.4. However, the scheme can only sign a single file, after which the public key must to be refreshed, so it will be impractical for key distribution. The verification is expensive since $(n + k + 1)$ pairings need to be computed.

Gennaro et al. presented an RSA-based homomorphic signature scheme for network coding (see Section 4.6) that works modulo a composite number N. A core technique in this scheme is to apply network coding over the integers rather than over a field as is traditionally done. By working over the integers we can choose small (e.g., 8-bit) integer coefficients for the linear combinations. Therefore, the computational efficiency at intermediate nodes can be improved and the total bandwidth overhead for the transmission from the source to destination is reduced. In this scheme, they used a hash function of the file identifier as a random oracle, and under the Standard RSA assumption, the signature scheme is secure in the random oracle model under Definition 4.2.5.

Zhao et al. (see Section 4.4) proposed a signature scheme for network coding that allows nodes to check the validity and integrity of the packets received without decoding. In this scheme a node computes the signature derived from a vector orthogonal to the space $\text{span}(m_1, \ldots, m_k)$. Zhao et al. defined a (q, k, N)-Diffie-Hellman Problem (see Definition 2.4.6), in which given a set S of vectors satisfying the verification criterion it is proven to be as hard as solving the DL Problem to find a new set of vectors which satisfy the verification criterion but are not any linear combinations of S. However, attacks described later show that Zhao et al.'s scheme is insecure under Definition 4.2.4 against the pollution attacks.

The communication overhead of this scheme is high: both the signature and the public key are comparatively large; the public key must be refreshed after distributing each single file. The sender is required to know the entire file before the signature can be computed.

Table 4.1 compares the underlying problem, the computational model, the reusable public key and the security proof of each scheme above.

Table 4.1: Performance analysis

	underlying problem	computational model	reusable public key	security proof
Boneh et al. [8] (scheme $\mathcal{S}_2$)	co-CDH Problem on Elliptic curve	random oracle model	yes	yes
Boneh et al. [8] (scheme $\mathcal{S}_1$)	DL Problem	standard model	yes	yes
Charles et al. [15]	co-CDH Problem on Elliptic curve	standard model	no	yes
Gennaro et al. [23]	RSA Problem	random oracle model	no	yes
Zhao et al. [46]	(q, k, N)-DH Problem	standard model	no	no

Now we discuss the complexities of each scheme: the signature size, the public key size and the verification computation (see Table 4.2). Assuming that a 1MB file will be distributed and the maximum packet size is up to 65535 Bytes, the file can be separated into 16 packets, that is, $k = 16$. For this example, Table 4.3 gives the comparison of the signature size and public key size in each scheme at the security level of 1024-bit RSA.

Table 4.2: Complexity analysis

	public key size	signature size	verification computation		
			pairings	exponentiations	multiplications
Boneh et al. [8] (scheme $\mathcal{S}_2$)	$O(l)$[1]	$O(l)$	2	$O(n+k)$ in $\mathbb{G}_1$	$O(n+k)$ in $\mathbb{G}_1$
Boneh et al. [8] (scheme $\mathcal{S}_1$)	$O(l(n+k+1))$	$O(l(k+1))$	1	$O(n+k)$ in $\mathbb{G}_1$	$O(n+k)$ in $\mathbb{G}_1$
Charles et al. [15]	$O(l(n+k+1))$	$O(l)$	$n+1$	$O(n+k)$ in $\mathbb{G}_1$	$O(n+k)$ in $\mathbb{G}_T$
Gennaro et al. [23]	$O(l(n+2))$	$O(l)$	–	$O(n+k)$ in $\mathbb{Z}_N$	$O(n+k)$ in $\mathbb{Z}_N$
Zhao et al. [46]	$O(l(n+k))$	$O(l(n+k))$	–	$O(n+k)$ in $\mathbb{F}_q$	$O(n+k)$ in $\mathbb{F}_q$

[1] l is the security parameter, k is the number of packets of a file and n is the dimension of a packet.

Boneh et al.'s scheme $\mathcal{S}_2$ has constant public key and constant signature (a point on an elliptic curve), and $\mathcal{S}_1$ requires longer public key and signatures to be delivered for each session, which could reduce the advantage of the network coding. The public key is

$(k+n+1)$ points and the signature is $(k+1)$ points on the same elliptic curve. Suppose that the signature scheme is to achieve a security level that is equivalent to the security provided by the RSA scheme with 1024-bit modulus. Boneh et al.'s schemes require the 160-bit security parameter l since they are based on bilinear maps. So the length of q is approximately 160 bits and the size of a point on the elliptic curve is 161 bits. We assume that a file can be separated into 16 packets, that is, $k = 16$. Each packet is viewed as an n-dimensional vector in $\mathbb{F}_q^n$, where

$$n = \frac{\text{packet size(bits)}}{|q|} = \frac{65535 \times 8}{160} = 3277,$$

where $|q|$ is the binary length of q. If we assume that a short signature scheme [5] is used in $\mathcal{S}_1$ as a standard signature scheme, then only one bilinear pairing is needed in the verification phase.

The security of Zhao et al.'s signature scheme is based on the hardness of (q, k, N)-Diffie-Hellman Problem, which is as hard as the Discrete Logarithm Problem in $\mathbb{F}_p$. If the signature scheme is to achieve a security level that is equivalent to the security provided by the RSA scheme with 1024-bit modulus, then the security parameter l of this scheme should be 1024 bits, that is, the length of p is 1024 bits, and we need a 160-bit q. In Zhao et al.'s signature scheme, both the public key and the signature are $(k+n)$ elements in $\mathbb{F}_q$, where $q|(p-1)$. And in the verification process, we need to compute $(k+n)$ exponentiations in G, which is a cyclic subgroup of order q in $\mathbb{F}_p$. We assume that a 1MB file can be separated into 16 packets, that is, $k = 16$. Each 65535-byte packet is viewed as an n-dimensional vector in $\mathbb{F}_q^n$, where

$$n = \frac{\text{packet size(bits)}}{|q|} = \frac{65535 \times 8}{160} = 3277,$$

where $|q|$ is the binary length of q.

Charles et al.'s signature scheme is based on the bilinear map on elliptic curves. If this scheme is to achieve a security level of 1024-bit RSA, 160-bit security parameter l is enough and the length of q is also approximately 160 bits. In Charles et al.'s scheme, the signature is a point on an elliptic curve (161 bits), and the public key includes $2(k+n)+1$ points on the same elliptic curve. In the verification process, we need to compute $(k+n+1)$ bilinear pairings, which make this scheme less efficient. We assume that a 1MB file can be separated into 16 packets ($k = 16$), and each 65535-byte packet can be split into n elements in $\mathbb{F}_q$, where $n = 3277$.

The security of Gennaro et al.'s RSA-based signature scheme is based on the hardness of RSA problem. If this scheme is to achieve a security level of 1024-bit RSA, then we choose the integer N of 1024-bit length. In Gennaro et al.'s scheme, each message packet

Table 4.3: Memory and communication overhead on the security level of 1024-bit RSA

	security parameter (bits)	public key size (bits)	signature size (bits)
Boneh et al. [8] (scheme $\mathcal{S}_2$)	160	161	161
Boneh et al. [8] (scheme $\mathcal{S}_1$)	160	530334	2737
Charles et al. [15]	160	1060507	161
Gennaro et al. [23]	1024	526336	1024
Zhao et al. [46]	1024	526880	526880

belongs to $\mathbb{Z}_N$, and all operations (e.g., exponentiation and multiplication) are done over integers in $\mathbb{Z}_N$. The public key includes the integer N and $n+1$ elements in $\mathbb{Z}_N$, and in the verification process, we need to compute $(k+n+1)$ exponentiations and $(k+n-1)$ multiplications in $\mathbb{Z}_N$. We assume that a 1MB file can be separated into 16 packets, that is, $k = 16$. Each 65535-byte packet is viewed as an n-dimensional vector in $\mathbb{Z}_N$, where

$$n = \frac{\text{packet size(bits)}}{|N|} = \frac{65535 \times 8}{1024} = 512.$$

For a larger file, k becomes increasingly large. Therefore, the memory and communication overhead per file is very high, if the signature size and public key size depend on the file size. So one of the design principles for network coding signatures is that signature and public key have sizes independent of the file size.

Chapter 5

Attacks on Signature Schemes for Network Coding

As we discussed in the previous chapter, the homomorphic property can be viewed to be positive in signature schemes for linear network coding, since by this property the intermediate nodes can construct a valid signature without any access to the private key of the source node. The intermediate nodes can combine the signatures appended to the received message packets and forward a new valid signature with the combined message packet.

On the other hand, however, the homomorphic property can be considered to be negative, and there exist some flaws to the current homomorphic signature schemes for network coding. In this chapter, we discuss attacks on some of the signature schemes described in the previous chapter.

5.1 A Weakness of the Homomorphic Property

In this section, we discuss a weakness of the homomorphic property used in network coding signatures. Let $m_1, m_2, \ldots, m_N$ be augmented vectors from one file, and $V = \text{span}(m_1, m_2, \ldots, m_N)$. Any intermediate node can collect the packet-signature pairs (w_i, σ_i), where $w_i \in V$, and generate a new signature $\sigma = \prod_{i=1}^{k} \sigma_i^{\alpha_i}$ on the combined packet $w = \sum_{i=1}^{k} \alpha_i w_i \in V$. Meanwhile, there are malicious nodes who can construct a linear combination of incoming packets in different subspaces and can forge a signature satisfying the verification criterion. Here we claim that it is not a new "attack", since we can solve this problem by generating a new public key for each file. We take Charles et al.'s scheme [15] to illustrate this process as follows.

Suppose that the source node will share the same public key $(g_1, g_2, \ldots, g_N, h, h_1, h_2 \ldots, h_N)$ for simultaneously distributing two files F_1, F_2, where $N = n + k$. $V_1, V_2 \subset \mathbb{F}_q^N$ are two subspaces spanned by two sets of basis vectors from F_1 and F_2, respectively. Let σ_1 be the signature on $\mathbf{w}_1 \in V_1$, then we have $\sigma_1 = \prod_{j=1}^{N} g_j^{w_{1j}s_j}$ and

$$\prod_{j=1}^{N} e(g_j^{w_{1j}}, h_j) = e(\sigma_1, h).$$

Similarly, let σ_2 be the signature on $\mathbf{w}_2 \in V_2$, we have $\sigma_2 = \prod_{j=1}^{N} g_j^{w_{2j}s_j}$ and

$$\prod_{j=1}^{N} e(g_j^{w_{2j}}, h_j) = e(\sigma_2, h).$$

After receiving $(\mathbf{w}_1, \sigma_1)$ and $(\mathbf{w}_2, \sigma_2)$, the malicious node can construct a new valid signature $\sigma = \sigma_1^{c_1}\sigma_2^{c_2}$ for a new message $\mathbf{w} = c_1\mathbf{w}_1 + c_2\mathbf{w}_2 \in span(\mathbf{w}_1, \mathbf{w}_2)$, since

$$\begin{aligned}
\prod_{j=1}^{N} e(g_j^{w_j}, h_j) &= \prod_{j=1}^{N} e\left(g_j^{c_1 w_{1j} + c_2 w_{2j}}, h_j\right) \\
&= \left(\prod_{j=1}^{N} e\left(g_j^{c_1 w_{1j} s_j + c_2 w_{2j} s_j}, h\right)\right) \\
&= e\left(\prod_{j=1}^{N} g_j^{c_1 w_{1j} s_j} \prod_{j=1}^{N} g_j^{c_2 w_{2j} s_j}, h\right) \\
&= e\left(\sigma_1^{c_1}\sigma_2^{c_2}, h\right) \\
&= e\left(\sigma, h\right). \qquad (5.1)
\end{aligned}$$

But $\mathbf{w} \notin V_1, V_2$. This process will generate a pollution in the further transmission, since the following nodes cannot decide which file the received packets comes from. So it is infeasible for the source to distribute multiple files at the same time.

The weakness of homomorphic property exists in most network coding signature schemes in the standard model [45, 15], though these schemes are proven secure according to the security definition (see Definition 4.2.4). To solve this problem, the source node is required to refresh the public key for distributing each file; this will introduce extra overhead in transmissions and significantly reduce the performance of network coding. Another approach is in these signature schemes to introduce a hash function of the file identity to label different files, and the hash function can be viewed a random oracle in the security proofs [8, 23].

5.2 An Attack on Yun et al.'s RSA-based Signature Scheme

Yun et al. [45] provided an RSA-based signature scheme for network coding, which is a homomorphic scheme in the standard model. In this section, we introduce a pollution attack on Yun et al.'s scheme.

For two primes p and q with $q|(p-1)$, the RSA public key is $PK = (N, e)$, where $N = pq$, $1 < e < \phi(N)$ and $gcd(e, \phi(N)) = 1$. Each augmented message packet m_i, $i = 1, 2, \ldots, k$, can be viewed as a $(k+n)$-dimensional vector $m_i = (m_{i1}, m_{i2}, \ldots, m_{i(k+n)}) \in \mathbb{Z}_q^{k+n}$. Given a valid signature-message pair (σ, m_i), an adversary can construct a new message m_i^* and forge a valid signature σ^* on m_i^* as follows. The adversary can select β and compute a new message $m_i^* = (m'_{i1}, m_{i2}, \ldots, m_{i(k+n)})$, where $m'_{i1} = m_{i1} + e \cdot \beta \in \mathbb{Z}_q$. Then the signature on the message $\mathbf{w}^*$ is $\sigma^* = \sigma \cdot g_1^{\beta}$, since

$$\begin{aligned}
\left(\sigma \cdot g_1^{\beta}\right)^e &\equiv \sigma^e \cdot g_1^{\beta \cdot e} \\
&\equiv \left(\prod_{j=1}^{k+n} g_j^{m_{ij}}\right)^{de} \cdot g_1^{e \cdot \beta} \\
&\equiv \prod_{j=1}^{k+n} g_j^{m_{ij}} \cdot g_1^{e \cdot \beta} \\
&\equiv g_1^{m_{i1} + e \cdot \beta} \prod_{j=2}^{k+n} g_j^{m_{ij}} \\
&\equiv \left(g_1^{m'_{i1}} \prod_{j=2}^{k+n} g_j^{m_{ij}}\right)^{de} \\
&\equiv (\sigma^*)^e \pmod{N}. \qquad (5.2)
\end{aligned}$$

Hence, Yun et al.'s signature scheme is existentially forgeable under the known-message attacks. Similarly, the adversary can generate a valid signature σ^{**} on any message $m_i^{**} = (m'_{i1}, m'_{i2}, \ldots, m'_{i(k+n)})$ by her choice.

5.3 Attacks on Zhao et al.'s Signature Scheme

In Zhao et al.'s scheme [46], we need to calculate a vector u orthogonal to all the message packets m_i, that is, $m_i \cdot \mathbf{u} = 1$, $i = 1, 2, \ldots, k$, and the signature is $\sigma = (u_1/s_1, u_2/s_2, \ldots, u_n/s_n)$,

where $(s_1, s_2, \ldots, s_n)$ is the private key of the source node. If an adversary can attain k combined message packets with linearly independent coefficient vectors, then she can recover the original messages packets $m_1, m_2, \ldots, m_k$.

If the algorithm of computing the orthogonal vector $\mathbf{u}$ is public and deterministic in this scheme, the orthogonal vector $\mathbf{u}$ can be uniquely recovered by the adversary, who will then be able to recover the private key of the source node using a valid signature. Keeping the algorithm of computing $\mathbf{u}$ secret or using a probabilistic algorithm to compute $\mathbf{u}$ can protect the private key of the source node. In this section we discuss two forgeability attacks to Zhao et al.'s signature scheme under the security definition 4.2.4,

5.3.1 Attack 1

There exists a forgeability attack mentioned in the original paper [46]. If σ_1 is the signature on File 1, and $\mathbf{w}$ a valid received vector which is a combination of original packets of File 1, then we have

$$d = \prod_{j=1}^{N} (g^{s_j})^{\sigma_{1j} w_j} = 1$$

.

If the source node then distributes File 2 using the same public key, a malicious node, instead of forwarding the signed File 2, can construct a signature σ_2 on a vector $\mathbf{v}$ chosen by the malicious node, as $\sigma_{2j} = (\sigma_{1j} w_j)/v_j$. The signature σ_2 also satisfies

$$d = \prod_{j=1}^{N} (g^{s_j})^{\sigma_{2j} v_j} = \prod_{j=1}^{N} (g^{s_j})^{\sigma_{1j} w_j} = 1.$$

But $\mathbf{v}$ may not be any valid combination of the packets of File 2.

To prevent this attack, the source node needs to refresh the public key of the source key or to recalculate the orthogonal vector u for distributing each file. Alternatively, Zhao et al. introduced another method, which will cost less overhead in the network coding system. For each new file, the source node can replace a random element s_i in the private key, and publish the new g_{s_i} in the public key. To compute the signature on the new file, we need to compute a new vector $\mathbf{u}$ which is orthogonal to the subspace V spanned by the new file. Since V has dimension k, we just modify the entries $u_{n+1}, \ldots, u_{n+k}$ in the vector $\mathbf{u}$, and then republish $\sigma_{n+1}, \ldots, \sigma_{n+k}$ for the new signature. The overhead of this method is less than the previous one of refreshing whole public key or the orthogonal vector $\mathbf{u}$. And security can be increased by changing more elements in the private key for each new file.

5.3.2 Attack 2

Wang [41] presented a pollution attack on Zhao et al.'s signature scheme. For the augmented message packets $m_i = (m_{i1}, m_{i2}, \ldots, m_{in}, \underbrace{0, \ldots, 0}_{i-1}, 1, 0, \ldots, 0)$, $i = 1, 2, \ldots, k$, we can construct a vector

$$u = (\underbrace{1, 1, \ldots, 1}_{n}, -\sum_{j=1}^{n} m_{1j}, -\sum_{j=1}^{n} m_{2j}, \ldots, -\sum_{j=1}^{n} m_{kj}),$$

which is orthogonal to all augmented message packets m_i, $i = 1, 2, \ldots, k$.

A malicious node can construct messages $m_i^* = (m_{i1}^*, m_{i2}^*, \ldots, m_{in}^*, 0, \ldots, 0, 1, 0, \ldots, 0))$, where $\sum_{j=1}^{n} m_{ij}^* = \sum_{j=1}^{n} m_{ij}$. It is clear that m_i^* is orthogonal to the vector u, and the signature on m_i^* will be accepted as a valid signature. The malicious node inserts the corrupted messages m_i^*, and the pollution will be spread in any further transmission.

Chapter 6

Conclusion and Future Work

This chapter summarizes the work in this paper. A summary of the main results as well as recommendations for future research is provided.

6.1 Conclusion

In this paper, we gave a brief introduction of network coding theory, and then described six homomorphic signature schemes for network coding that are targeted to protect against the pollution attacks.

Network coding has vast application potential and significant benefits: improved throughput, increased robustness of networks and data security. In this paper, we presented linear network coding which is used frequently in practice, and discussed some security problems of network coding such as eavesdropping attacks, pollution attacks.

Considering the security of network coding, we introduced homomorphic hashing and homomorphic signatures. We first gave the definitions of homomorphic signature for network coding and two security definitions. Then we described three pairing-based homomorphic signature schemes [8, 15], three RSA-based homomorphic signature schemes [45, 23], and a signature scheme using a vector orthogonal to the message linear subspace [46].

Next, we presented a detailed analysis of the security and the complexity. A preliminary security analysis showed that, in the different security models and under various intractability assumptions, these signature schemes can provide cryptographic protection for network coding against pollution attacks. A comparison of the computational complexities of these signature schemes showed that Boneh et al.'s scheme $\mathcal{S}_2$ [8] has constant public-key size and per-packet overhead; Gennaro et al.'s RSA-based signature scheme over

the integers [23] can reduce the bandwidth overhead for transmission since small integer coefficients can be chosen for the linear combinations.

Finally, we pointed out flaws of some of these schemes, even though some security analysis had been presented. We gave existential forgeries for some of the schemes presented in this paper. Further, we gave solutions (e.g., refreshing the public key) to prevent such attacks.

6.2 Future Work

Signatures for network coding is an interesting and challenging research area. While many solutions have been proposed, there are many problems that still need further study. Even the problems that have been addressed might have to be revisited in terms of the various applications in different networks and improvements of the performance and the efficiency. Further work could be pursued in the following research areas.

- Designing a secure RSA-based homomorphic signature for network coding in wireless ad hoc network still needs to be investigated.
- Finding more applicable homomorphic functions for network coding is another interesting topic to increase the protocol efficiency or to reduce the system overhead.
- The homomorphic property is the only currently available and effective tool for signatures for network coding. Other new solutions without the homomorphic property might bring more novel ideas or innovations in this research area.

Bibliography

[1] R. Ahlswede, N. Cai, S. Li, and R. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46:1204–1216, 2000. 15

[2] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, 1999. 6, 9, 11

[3] I. Blake, G. Seroussi, and N. Smart. *Advances in Elliptic Curves Cryptography*. Cambridge University Press, 2005. 6, 8, 9, 10

[4] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer Berlin/Heidelberg, 2004. 12, 13, 43

[5] D. Boneh and X. Boyen. Short signatures without random oracles. In *Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer Berlin / Heidelberg, 2004. 31, 45

[6] D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. pages 149–177, Secaucus, NJ, USA, 2008. Springer-Verlag New York, Inc. 13, 31

[7] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM, Journal on Computing*, 32(3):586–615, 2003. 12, 14

[8] D. Boneh, D. Freeman, J. Katz, and B. Waters. Signing a linear subspace: Signature schemes for network coding. In *Public Key Cryptography (PKC) 2009*, volume 5443 of *Lecture Notes in Computer Science*, pages 68–87. Springer Berlin/Heidelberg, 2009. 3, 4, 14, 21, 23, 28, 29, 30, 31, 44, 46, 48, 52

[9] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, 2003. 14, 32, 43

[10] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *ASIACRYPT '01*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, 2001. 10

[11] X. Boyen. The Uber-assumption family – a unified complexity framework for bilinear groups. In *2nd International Conference on Pairing-based Cryptography 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 39–56. Springer-Verlag/Berlin, 2008. 12

[12] E. Bresson, D. Catalano, and D. Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In *ASIACRYPT'03*, volume 2894 of *Lecture Notes in Computer Science*, pages 37–54. Springer-Verlag/Berlin, 2003.

[13] N. Cai and R. Yeung. Secure network coding. In *IEEE International Symposium on Information Theory*, pages 323–346, 2002. 22

[14] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *Journal of ACM*, 51(4):557–594, 2004. 14

[15] D. Charles, K. Jain, and K. Lauter. Signatures for network coding. *International Journal of Information and Coding Theory (IJICOT)*, 1(1):3–14, 2006. 3, 4, 23, 34, 44, 46, 47, 48, 52

[16] D. Chiu, R. Yeung, J Huang, and B. Fan. Can network coding help in p2p networks? In *4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2006. 18

[17] Y. Chou, P.and Wu and K. Jain. Practical network coding. *Allerton Conference on Communication, Control, and Computing, Monticello*, IL, October, 2003. 21

[18] J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *26th Annual Symposium on Foundations of Computer Science, 1984*, pages 372–382, 21–23 Oct. 1985. 23

[19] R. Cramer and I. Damgård. Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? In *CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 424–441. Springer-Verlag, 1998. 23

[20] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976. 9

[21] C. Fragouli, J. Le Boudec, and J. Widmer. Network coding: an instant primer. *SIGCOMM Computer Communication Review*, 36(1):63–68, January 2006. 18, 19, 20, 22

[22] C. Fragouli and E. Soljanin. *Network coding fundamentals*, volume 2. Now Publishers Inc., Hanover, MA, USA, 2007. 18

[23] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin. Secure network coding over the integers. In *PKC 2010*, volume 6056 of *Lecture Notes in Computer Science*, pages 142–160. Springer-Verlag, 2010. 3, 4, 23, 38, 44, 46, 48, 52, 53

[24] C. Gkantsidis and P. Rodriguez. Cooperative security for network coding file distribution. In *IEEE INFOCOM 2006. 25th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 1–13, 2006.

[25] C. Gkantsidis and R. Rodriguez. Network coding for large scale content distribution. In *IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 2235–2245, 2005. see `http://dx.doi.org/10.1109/INFCOM.2005.1498511`. 18

[26] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17:281–308, 1988. 13

[27] K. Han, T. Ho, R. Koetter, M. Médard, and F. Zhao. On network coding for security. In *IEEE Military Communications Conference MILCOM 2007*, pages 1–6, Oct. 2007. 41

[28] T. Ho and D. Lun. *Network Coding: An Introduction*. Cambridge University Press, 2008. 15, 16, 18, 19, 20

[29] R. Johnson, D. Molnar, D Song, and D Wagner. Homomorphic signature schemes. In *Topics in Cryptology: CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 204–245. Springer Berlin/Heidelberg, 2002. 27, 28

[30] A. Joux. A one round protocol for tripartite Diffie-Hellman. In *ANTS-IV*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–393, London, UK, 2000. Springer-Verlag. 12

[31] A. Joux and K. Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. In *Journal of Cryptology 16 (2003)*, pages 239–247, 2003. 10

[32] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11:782–795, 2001. 20

[33] M. Krohn, M. Freedman, and D. Mazieres. On-the-fly verification of rateless erasure codes for efficient content distribution. In *IEEE Symposium on Security and Privacy*, pages 226–240, 9–12 May 2004. 3, 24, 26, 34

[34] Q. Li, D. Chiu, and J. Lui. On the practical and security issues of batch content distribution via network coding. In *ICNP*, pages 158–167, 2006.

[35] S. Li, R. Yeung, and N. Cai. Linear network coding. In *IEEE Transactions on Information Theory*, volume 49, pages 371–381, 2003. 20, 21

[36] P. Maymounkov. Online codes. Technical Report 2002-833, New York University, see `http://pdos.csail.mit.edu/~petar/papers/maymounkov-online.pdf`. 24

[37] A. Menezes, P. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Inc., 1996. 5, 6, 12, 37

[38] C. Ran, G. Oded, and H. Shai. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[39] R. Rivest. Two new signature schemes. 2000. Presented at Cambridge seminar, see `http://www.cl.cam.ac.uk/Research/Security/seminars/2000/rivest-tss.pdf`. 23, 27

[40] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–177. Academic Press, 1978. 23

[41] Y. Wang. Insecure "provable secure network coding". Cryptology ePrint Archive, Report 2010/060, 2010. see `http://eprint.iacr.org/`. 51

[42] M. Yang and W. Yan. Fast signature scheme for network coding. In *Distributed Computing and Algorithms for Bussiness Engineering and Sciences (DCABES)*, 2009. see `http://dcabes.meeting.whut.edu.cn/DCABES2009/Files/presentation/Mingxi$%$20Yang.ppt`.

[43] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan. An efficient signature-based scheme for securing network coding against pollution attacks. In *INFOCOM 2008. 27th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, pages 1409–1417, 2008. 37

[44] R. Yueng and Z. Zhang. Distributed source coding for satellite communications. *IEEE Transactions on Information Theory*, 45:1111 – 1120, 1999. 15

[45] A. Yun, J. Cheon, and Y. Kim. On homomorphic signatures for network coding. *IEEE Transactions on Computers*, 2009. 3, 4, 23, 37, 40, 48, 49, 52

[46] F. Zhao, T. Kalker, M. Medard, and K. Han. Signatures for content distribution with network coding. In *IEEE International Symposium on Information Theory ISIT 2007*, pages 556–560, 24–29 June 2007. 3, 4, 10, 11, 23, 40, 42, 44, 46, 49, 50, 52