Design, analysis and applications of cryptographic techniques

Chan Yeob Yeun

Technical Report
RHUL–MA–2001–5
1 November 2001

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England
http://www.rhul.ac.uk/mathematics/techreports

Design, Analysis and Applications of Cryptographic Techniques

Chan Yeob Yeun

Thesis submitted to
The University of London
for the degree of
Doctor of Philosophy
2000.

Royal Holloway and Bedford New College,
University of London.

Abstract

Cryptographic techniques, such as encipherment, digital signatures, key management and secret sharing schemes, are important building blocks in the implementation of all security services.

In this thesis, we present a general model for online secret sharing schemes and investigate the design of online secret sharing schemes which are derived from this model such as Cachin and Pinch's schemes [13, 48]. We propose a modified version of the Pinch multiple secret sharing protocol, which identifies *all* cheaters, regardless of their number, improving on previous results by Ghodosi *et al.* [21]. A new scheme is then proposed for computationally secure online secret sharing, in which the shares of the participants can be reused. The security of the scheme is based on the intractability of factoring. This scheme has the advantage that it detects cheating and it enables the identification of *all* cheaters by an arbitrator, regardless of their number. The scheme does not rely on a "last participant" who reconstructs the secret on behalf of a minimal trusted set: the responsibility is diffused among all participants.

In addition, we cryptanalyse the recently proposed signature scheme by Shao, based on the discrete logarithm problem, and show it is subject to homomorphism attacks, despite a claim in [54] to the contrary. Moreover, we show that there are major differences between a digital signature with message recovery scheme and an authenticated encryption scheme and point out that the signature with message recovery scheme that was recently proposed by Chen [14] is actually not a signature scheme. It would more accurately be described as an authenticated encryption scheme.

Furthermore, we propose a modification to the Helsinki protocol [5] which prevents

attacks by an adversary.

Some of the material in Chapters 2, 3 and 4 of the thesis has appeared in published papers [40, 41, 59, 60, 61].

Acknowledgements

I would like to thank God for his divine guidance while I lost confidence in my abilities. I am also indebted to my supervisor and adviser, Professors Fred Piper and Peter Wild, for their reassurance and guidance whenever I encountered difficulties and hardships. Extra special thanks go to Professor Chris Mitchell for many hours of useful discussion and for tirelessly reading my work, and for suggesting many invaluable comments and ideas. I am also grateful to Dr. Mike Burmester for always being available, and for many hours of interesting and enlightening discussion.

I should like to thank the staff and postgraduates in the Mathematics Department for making my time at Royal Holloway an enjoyable one. In particular I would like to thank Dr. Karl Brincat for helpful comments.

My studies have been sponsored by a Research Studentship and a Maintenance award from the Mathematics Department.

Finally, and most importantly, I am most grateful to my family for their support, both financial and emotional, without which this course of study would not have been possible.

Contents

List of Figures

Chapter 1

Introduction

1.1 General introduction

As long as there are creatures endowed with language, there will be confidential messages intended for a limited audience. How can these messages be transmitted secretly, so that no unauthorised person gets knowledge of the content of message? And how can one guarantee that a message arrives in the right hands exactly as it was transmitted?

Traditionally, there are two ways to answer such questions. One can disguise the very existence of a message, perhaps by writing with invisible ink; or try to transmit the message via a trustworthy person. This is the method favoured throughout history by clandestine lovers and nearly all classical tragedies provide evidence of the method's shortcomings.

A totally different approach is to encipher (or encrypt) a message. In this case one does not disguise its existence. On the contrary, the message is transmitted over a public, insecure channel, but encrypted in such a way that no one except the

intended recipient may decipher it. This offers a rather tempting challenge to an enemy. Such challenges are usually accepted and not unusually overcome.

There is a satisfying appropriateness to cryptology's role in the birth of electronic computing. The arrival of the Information Age has revealed an urgent need for cryptography in the private sector. Today, vast amounts of sensitive information such as health and legal records, financial transactions, credit ratings and the like are routinely exchanged between computers via public communication facilities. Society turns to the cryptographer for help in ensuring the privacy and authenticity of such sensitive information.

Cryptographic techniques, such as encipherment, digital signatures, key agreement and secret sharing schemes, are important building blocks in the implementation of any security service. A cryptosystem defines encryption and decryption transformations, which depend on the value of keys. A symmetric cryptosystem uses one key for both transformations. A public key cryptosystem uses separate keys for each transformation.

The only publicly-standardized symmetric cryptosystem is the U.S. Data Encryption Standard (DES), which has been widely used since the 1970s. Due to technological advances, the useful life of single-encryption DES is running out. However, the use of multiple-encryption DES systems may provide adequate protection for many applications for some years to come.

The idea of the public-key technique was first introduced by Diffie and Hellman [18] in 1976, and began a revolution in cryptology. Public-key cryptosystems can be encryption or authentication schemes. The RSA algorithm can operate in both modes. The strength of RSA depends on the difficulty of factoring the product of two large numbers. The selection of an appropriate modulus size can make RSA

arbitrarily strong. The ElGamal algorithm is an alternative public-key algorithm, the strength of which depends on the difficulty of computing discrete logarithms.

A digital signature is an electronic equivalent of verifying the source of a written message on the basis of a written signature. A digital signature is stronger than a seal, in that the recipient must not be able to generate a digital signature which is indistinguishable from one generated by the originator. Digital signatures usually employ public-key cryptosystems, often in conjunction with a one-way hash-function.

'Key agreement' denotes a protocol whereby two (or more) parties jointly establish a secret key by communicating over a public channel. In a key agreement scheme, the value of the key is determined as a function of inputs provided by both parties.

1.2 Contents of this thesis

The remainder of Chapter 1 introduces the abstract algebra and number theory which is particularly useful for public-key cryptosystems as well as cryptographic techniques described in this thesis. It also presents a short survey of cryptographic evolution along with an overview of cryptographic techniques.

Chapter 2 gives a general model for online secret sharing scheme, and all the schemes considered in the remainder of this chapter such as Cachin's scheme and Pinch's scheme [13, 48] fit the basic model. A security analysis is given for the existing schemes and their shortcomings are considered, an enhanced version of Pinch's scheme is suggested.

Using the basic model we design a new online secret sharing scheme which detects cheating and enables the identification of all cheaters by an arbitrator, regardless of

their number. The scheme also does not rely on 'last participant' who reconstructs the secret on behalf of a minimal trusted set: the responsibility is diffused among all participants by contrast with another secret sharing scheme such as Shamir's original (t, n) threshold scheme [53].

Chapter 3 studies the cryptanalysis of digital signatures such as RSA [50] and ElGamal type [50, 19, 1]. We cryptanalyse the recently proposed signature by Shao [54] based on the discrete logarithm problem, which is claimed to prevent homomorphism attacks. However, we show that Shao's scheme is vulnerable to homomorphism attacks, and we also point out that substitution attacks can be avoided by the use of one-way hash-function. As a result there no longer appears to be any reason to use Shao's scheme.

We also show that there are major differences between a digital signature with message recovery and authentication encryption (signcryption) and we point out that the discrete logarithm based signature with message recovery scheme proposed by Chen [15] is actually not a signature scheme. It would more accurately be described as an authenticated encryption scheme. In the authenticated encryption schemes described in Horster-Michels-Petersen and Lee-Chang [28, 33], only the sender A and the receiver B can verify a protected message with the aid of his/her private decryption key. It is for this reason that the authors of both [28] and [33] have been careful to call their schemes authenticated encryption schemes rather the digital signature with message recovery.

Chapter 4 investigates a key transport public-key cryptographic technique called the Helsinki protocol and its shortcoming (first pointed out by Horng and Hsu). We describe a simple modification to the Helsinki protocol which prevents the Horng-Hsu attack, but yet which does not add significantly to the communications

overhead for the protocol. Note that the Horng-Hsu attack is closely related to the Lowe attack [34] on the Needham-Schroeder protocol. Moreover, the modification we propose in the revised Helsinki protocol corresponds directly to the modification Lowe proposes to the Needham-Schroeder protocol.

1.3 Abstract algebra

The purpose of this section is to introduce the algebraic definitions and results that are necessary for an understanding of the results in this thesis. Most of the theorems are quoted without a proof, but with references to where a proof can be found.

Definition 1.3.1 A binary operation '$\cdot$' on a set S is a mapping from $S \times S$ to S. That is, '$\cdot$' is a rule which assigns to each ordered pair of elements from S an element of S.

1.3.1 Groups

Definition 1.3.2 A group $G(\cdot)$ or simply G consists of a set G with a binary operation $\cdot$ on G satisfying the following properties.

(i) For every $a, b, c \in G$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$. (Associative)

(ii) There is an element $e \in G$ such that every a in G, $a \cdot e = e \cdot a = a$. (Identity)

(iii) For every $a \in G$, there is an element a^{-1} in G such that $a \cdot a^{-1} = a^{-1} \cdot a = e$. (Inverse)

A group G is abelian (or commutative) if, furthermore

(iv) $a \cdot b = b \cdot a$ for all $a, b \in G$.

Note that a group G is called an additive group when the operation is additive $(+)$, while a group G is called a multiplicative group under the operation of multiplicative $(\times)$. In a group with an additive operation, we have $a + (-a) = (-a) + a = 0$, where the inverse element of a is written as $-a$. In this case, the identity of element e is 0. Under the multiplicative operation, the identity element e is 1 and inverse element of a is written as a^{-1}, so that $a\ a^{-1} = a^{-1}\ a = 1$.

Definition 1.3.3 Let G be a group. If G has a finite number of elements, say n, then we say that the **order of** G is n. We write this symbolically by $o(G) = n$, and in this case we say G is a **finite group**.

Definition 1.3.4 A nonempty subset H of a group G is a **subgroup** if H is a group with the same binary operation as G.

Theorem 1.3.5 Let G be a group with binary operation $\cdot$, and let g be an element of G. Then

$$H = \{g^i |\ i \text{ is an integer}\}$$

is a subgroup of G.

A proof can be found in [32], page 40, Theorem 2.4.4.

Definition 1.3.6 A group H is said to be **cyclic** if there exists an element $g \in H$ such that every element of H can be written as g^n for some integer n. In this case, H is called the **cyclic group generated by** g and g is called a **generator** of H. If H is a subgroup of another group G, then H is called a **cyclic subgroup**.

Corollary 1.3.7 If G is a finite group of order n, then $g^n = e$ for all $g \in G$.

A proof can be found in [32], page 46, Corollary 2.5.7.

Corollary 1.3.8 The order of every element of a finite group is a divisor of the order of the group.

A proof can be found in [32], page 46, Corollary 2.5.8.

Corollary 1.3.9 If G is a finite group of order p where p is a prime integer, then G is cyclic and every element of G expect the identity is a generator of G.

A proof can be found in [32], page 46, Corollary 2.5.9.

Definition 1.3.10 A **homomorphism** from G to H is a mapping α from G to H such that

$$\alpha(g_1 \cdot g_2) = \alpha(g_1) \cdot \alpha(g_2)$$

for all $g_1, g_2 \in G$.

Definition 1.3.11 Let G and H be groups and let α be a homomorphism from G to H. If α is both onto and one-to-one, then α is called an **isomorphism**. If α is an isomorphism, then G and H are said to be **isomorphic**, and we write $G \cong H$.

1.3.2 Rings and Fields

We introduce another algebraic system called a *ring*. We will define *ring* and prove several elementary theorems about *rings*. Then we study subrings and their homomorphisms and isomorphisms. We also study two special types of rings namely,

integral domains and fields. These two algebraic systems are important because our usual 'arithmetic' is carried out in either an integral domain or a field.

Definition 1.3.12 A ring $R(+, \cdot)$ or simply R consists of a set R with two binary operations, denoted by $+$ and $\cdot$ and called addition and multiplication, which satisfy the following axioms.

(i) $(a + b) + c = a + (b + c)$ for all $a, b, c \in R$. (associativity of addition)

(ii) There exists an element $0 \in R$ such that $0 + a = a$ for all $a \in R$. (existence of additive identity)

(iii) For each $a \in R$, there exists $x \in R$ such that $a + x = 0$. (existence of additive inverse)

(iv) $a + b = b + a$ for all $a, b \in R$. (commutative of addition)

(v) $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for all $a, b, c \in R$. (associativity of multiplication)

(vi) $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ and $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$ for all $a, b, c \in R$. (distributive)

Definition 1.3.13 Let R be a ring. We say R is a **ring with unity** if there exists $e \in R$ such that $a \cdot e = e \cdot a = a$ for all $a \in R$. If such an element e exists, it is called a **unity element** of R.

Definition 1.3.14 Let R be a ring. Then R is said to be **commutative ring** if $a \cdot b = b \cdot a$ for all $a, b \in R$.

Definition 1.3.15 A nonempty set S is a **subring** of a ring R if S is a subset of R and if S itself is a ring with respect to the addition and multiplication of R.

We defined homomorphism between two groups. Since rings have two binary operations defined on them, rather than one, it is not unreasonable that a homomorphism between two rings must preserve both the addition and multiplication of the rings.

Definition 1.3.16 Let R and S be rings. A ring **homomorphism** is a mapping α from R to S such that

(i) $\alpha(a+b) = \alpha(a) + \alpha(b)$

(ii) $\alpha(a \cdot b) = \alpha(a) \cdot \alpha(b)$

for all $a, b \in R$.

Definition 1.3.17 A **field** F is a commutative ring in which all the nonzero elements form an abelian group under multiplication.

Definition 1.3.18 Let a and b be two elements of a commutative ring R, with $a \neq 0$. The element a **divides** b, denoted $a|b$, if there exists an element $c \in R$ such that $b = ac$.

Definition 1.3.19 Let $a_1, \ldots, a_n$ be elements of a commutative ring R. A nonzero element $c \in R$ is a **common divisor** of $a_1, \ldots, a_n$ if $c|a_i$ for $i = 1, \ldots, n$.

Definition 1.3.20 Let $a_1, \ldots, a_n$ be elements of commutative ring R. A nonzero element $d \in R$ is a **greatest common divisor** of $a_1, \ldots, a_n$, denoted $d = \gcd(a_1, \ldots, a_n)$, if

(i) d is a common divisor of $a_1, \ldots, a_n$, and

(ii) whenever $c|a_i$ for $i \in \{1, \ldots, n\}$, then $c|d$.

Definition 1.3.21 Let a and b be two elements in a commutative ring R with unity. Then a and b are **coprime** or **relatively prime** if $\gcd(a, b) = 1$.

Definition 1.3.22 Let R be a commutative ring. A **polynomial** in the indeterminate x over the ring R is an expression of the form

$$f(x) = a_n x^n + \cdots + a_2 x^2 + a_1 x + a_0$$

where each $a_i \in R$ and $n \geq 0$. The element a_i is called the **coefficient** of x^i in $f(x)$. The largest integer m for which $a_m \neq 0$ is called the degree of $f(x)$, denoted *deg* $f(x)$; a_m is called the **leading coefficient** of $f(x)$. If $f(x) = a_0$ (a **constant polynomial**) and $a_0 \neq 0$, then $f(x)$ has degree 0. If all the coefficients of $f(x)$ are 0, then $f(x)$ is called the **zero polynomial** and its degree, for mathematical convenience, is defined to be -1. The polynomial $f(x)$ is said to be **monic** if its leading coefficient is equal to 1.

Definition 1.3.23 If R is a commutative ring, the **polynomial ring** $R[x]$ is the ring formed by the set of all polynomials in the indeterminate x having coefficients from R. The two operations are the standard polynomial addition and multiplication, with coefficient arithmetic performed in the ring R.

Definition 1.3.24 Let $f(x) \in F[x]$ be a polynomial of degree at least 1. Then $f(x)$ is said to be **irreducible over** F if it cannot be written as the product of two polynomials in $F[x]$, each of positive degree.

Definition 1.3.25 (division algorithm for polynomial) If $g(x), h(x) \in F[x]$, with $h(x) \neq 0$, then ordinary polynomial long division of $g(x)$ by $h(x)$ yields polynomials $q(x)$ and $r(x) \in F[x]$ such that

$$g(x) = q(x)h(x) + r(x), \text{ where } \deg r(x) < \deg h(x).$$

Moreover, $q(x)$ and $r(x)$ are unique. The polynomial $q(x)$ is called the **quotient**, while $r(x)$ is called the **remainder**. The remainder of the division is sometimes denoted $g(x)$ mod $h(x)$, and the quotient is sometimes denoted $g(x)$ div $h(x)$.

Definition 1.3.26 An integral domain R is a **Euclidean ring** if for all nonzero $a \in R$, there is defined a non negative integer $d(a)$ such that

(i) for all nonzero $a, b \in R$, $d(a) \leq d(ab)$, and

(ii) for any $a, b \in R$ with $b \neq 0$, there exist $m, r \in R$ such that $a = mb + r$ with either $r = 0$ or $d(r) < d(b)$.

Lemma 1.3.27 Let R be a Euclidean ring. Any two elements a and b in R have a greatest common divisor d which can be expressed in the form $d = \lambda a + \mu b$ for some $\lambda, \mu \in R$.

A proof can be found in [26], page 145, Lemma 3.7.1.

The following theorem is often proved for integers or polynomial rings over fields. A proof is included here for the general case of Euclidean rings. The proof is adapted from [16], page 157, Theorem 4.

Theorem 1.3.28 Let a and b be two elements in a Euclidean ring R. The $\gcd(a, b)$ can be calculated in R as follows:

$$\begin{aligned}
a &= q_0 b + r_1, && \text{where } d(r_1) < d(b) \\
b &= q_1 r_1 + r_2, && \text{where } d(r_2) < d(r_1) \\
&\vdots && \vdots \\
r_{n-2} &= q_{n-1} r_{n-1} + r_n, && \text{where } d(r_n) < d(r_{n-1}) \\
r_{n-1} &= q_n r_n,
\end{aligned}$$

where $r_n = \gcd(a, b)$.

Proof

Note that $d(b) > d(r_1) > d(r_2) > \ldots$ is a strictly decreasing sequence of non negative integers, which must terminate when a remainder is zero. Now, $r_1 = a - q_0 b$. Therefore, $r_2 = b - q_1 r_1 = b - q_1(a - q_0 b) = -q_1 a + b(1 + q_0 q_1)$. Moreover, if $r_{i-1} = ax + by$ and $r_{i-2} = ax' + by'$, then $r_i = -r_{i-1}q_{i-1} + r_{i-2} = a(x' - xq_{i-1}) + b(y' - yq_{i-1})$, so that, by induction, $r_n = au + bv$, for some $u, v \in R$. Therefore any factor of a and b also divides r_n. Now, $r_n | r_n$ and $r_n | r_{n-1}$, so that $r_n | r_{n-2}$. Furthermore, if $r_n | r_{i+2}$ and $r_n | r_{i+1}$, then $r_n | r_i$, since $r_i = q_{i+1} r_{i+1} + r_{i+2}$. Therefore, by induction, $r_n | r_i$ for all i. In particular, $r_n | b$ and $r_n | a$, so that r_n is a common divisor of a and b. Therefore, $r_n = \gcd(a, b)$ by definition.

□

Definition 1.3.29 A **finite field** is a field F which contains a finite number of elements. The order of F is the number of elements in F.

Definition 1.3.30 A generator g of a **finite field** $\mathbb{F}_q$ is an element of order $q - 1$; equivalently, g is a generator if the powers of g run through all nonzero elements of $\mathbb{F}_q$.

Definition 1.3.31 $\mathbb{F}_q^*$ is a cyclic group of order $q - 1$. Hence $a^q = a$ for all $a \in \mathbb{F}_q$.

The next theorem gives a basic fact about finite fields. It says that the nonzero elements of any finite field form a cyclic group; in other words, they are all powers of a single element.

Theorem 1.3.32 Every **finite field** has a generator. If g is a generator of $\mathbb{F}_q^*$, then g^j is also a generator if and only if $\gcd(j, q - 1) = 1$. Thus, there are a total of $\phi(q - 1)$ different generators of $\mathbb{F}_q^*$, where ϕ denotes the Euler ϕ-function.

A proof can be found in [30], page 56–57, Theorem 2.1 and Lemma 2.1.

1.4 Number theory

The set of integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$ is denoted by the symbol $\mathbb{Z}$.

Definition 1.4.1 Let a, b be integers. Then a divides b (equivalently: a is a divisor of b, or a is a factor of b) if there exists an integer c such that $b = ac$. If a divides b, then we write $a|b$.

Proposition 1.4.2 Properties of divisibility For all $a, b, c \in \mathbb{Z}$, the following are true:

(i) $a|a$

(ii) If $a|b$ and $b|c$, then $a|c$.

(iii) If $a|b$ and $a|c$, then $a|(bx + cy)$ for all $x, y \in \mathbb{Z}$.

(iv) If $a|b$ and $b|a$, then $a = \pm b$.

Definition 1.4.3 Division algorithm for integers If a and b are integers with $b \geq 1$, then ordinary long division of a by b yields integer q (the quotient) and r (the remainder) such that

$$a = qb + r, \text{ where } 0 \leq r < b.$$

Moreover, q and r are unique. The remainder of the division is denoted $a \bmod b$, and the quotient is denoted $a \operatorname{div} b$.

Definition 1.4.4 A non-negative integer d is the **least common multiple** of integers a and b, denoted $d = \text{lcm}(a, b)$, if

(i) $a|d$ and $b|d$; and

(ii) where $a|c$ and $b|c$, then $d|c$.

Equivalently, $\text{lcm}(a, b)$ is the smallest non-negative integer that divides by both a and b. In fact, $\text{lcm}(a, b) = ab \ / \ \gcd(a, b)$.

Definition 1.4.5 If a and b are integers, then a is said to be **congruent** to b modulo n, write $a \equiv b \pmod{n}$, if n divides $(a - b)$. The integer n is called the *modulus* of the congruence.

Definition 1.4.6 The **equivalence class** modulo n of an integer b is the set of all integers congruent to b modulo n.

Definition 1.4.7 The ring of **integers modulo** n, denoted $\mathbb{Z}_n$, is the set of (*equivalence class of*) integers $\{0, 1, 2, \ldots, n-1\}$. Addition, subtraction, and multiplication in $\mathbb{Z}_n$are performed modulo n.

Definition 1.4.8 An integer $b \in \mathbb{Z}_n$ is said to be **invertible** or a **unit** of $\mathbb{Z}_n$, if there is an integer $x \in \mathbb{Z}_n$, such that $bx \equiv 1 \pmod{n}$. If such an x exists, then it is referred to as the **multiplicative inverse** of b in $\mathbb{Z}_n$, and denoted by b^{-1}.

Theorem 1.4.9 Let a, b and $n > 0$ be integers, and $g = \gcd(a, n)$. The congruence $ax \equiv b \pmod{n}$ has a solution if and only if $g|b$. If this condition is met, then the solutions form an arithmetic progression with common difference n/g, giving g solutions modulo n.

A proof can be found in [26], page 62, Theorem 2.17. Therefore, $b \in \mathbb{Z}_n$ has a multiplicative inverse if and only if $\gcd(b, n) = 1$. Therefore, if n is prime, every non-zero $b \in \mathbb{Z}_n$ has a multiplicative inverse.

Theorem 1.4.10 (Chinese Remainder Theorem) Suppose $n_1, n_2, \ldots, n_r$ are r positive integers that are pair-wise coprime, and let $a_1, a_2, \ldots, a_r$ denote any r integers. Then the congruences

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\vdots \\ x &\equiv a_r \pmod{n_r} \end{aligned}$$

have a common solution which is unique modulo $n = n_1 n_2 \cdots n_r$.

A proof can be found in [51], page 136, Theorem 3.12.

So given n_1 and n_2 are coprime any pair of simultaneous congruences $x \equiv a_1 \pmod{n_1}$ and $x \equiv a_2 \pmod{n_2}$ have the same solutions as the single congruence

$$x \equiv a_2 t_1 n_1 + a_1 t_2 n_2 \pmod{n_1 n_2},$$

where $t_1 n_1 + t_2 n_2 = 1$. In particular, if $a_1 = a_2$, then $x \equiv a_1 \pmod{n_1 n_2}$.

Definition 1.4.11 The multiplicative group of $\mathbb{Z}_n$ is $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n | \gcd(a, n) = 1\}$.

Definition 1.4.12 Let $a \in \mathbb{Z}_n^*$. The **order** of a, denoted o(a), is the least positive integer k such that $a^k \equiv 1 \pmod{n}$.

Theorem 1.4.13 (Fermat's Theorem) Let p be a prime. If $\gcd(p, a) = 1$, then $a^{p-1} \equiv 1 \pmod{p}$.

A proof can be found in [51], page 187, Theorem 5.3.

Definition 1.4.14 The **Euler Totient Function** $\phi(n)$ is the number of positive integers less than or equal to n that are coprime to n.

Note that $\phi(n) = |\mathbb{Z}_n^*|$.

Theorem 1.4.15 (Euler's Theorem) Let $n \geq 2$. If $\gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.

A proof can be found in [51], pages 203–204, Theorem 5.14.

Corollary 1.4.16 If $\gcd(a, n) = 1$ then $o(a)$ modulo n divides $\phi(n)$.

A proof can be found in [44], page 98, Corollary 2.32.

Theorem 1.4.17 Let $n = 1, 2, 4, p^\alpha$, or $2p^\alpha$, where p is an odd prime. If $\gcd(a, n) = 1$, then the congruence $x^b \equiv a \pmod{n}$ has $\gcd(b, \phi(n))$ solutions or no solutions, according as

$$a^{\phi(n)/\gcd(b,\phi(n))} \equiv 1 \pmod{n}$$

or not.

A proof can be found in [44], page 104, Corollary 2.42.

1.5 Integer factorisation

The problem of integer factorisation is one of the oldest in number theory and the advent of computers have stimulated considerable progress in recent years. However, the security of many cryptographic techniques depends upon the intractability of the integer factorisation problem. A partial list of such schemes includes

the RSA public-key encryption scheme (see Section 1.10.1) and the RSA signature scheme (see Section 1.11.2). This section focuses on the current knowledge on algorithms for the integer factorisation problem.

Definition 1.5.1 The **integer factorisation problem** is the following: given a positive integer n, find its prime factorisation; that is, write $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 1$.

This problem is believed to be hard for general n when n is large. Some ingenious methods have been devised in an attempt to factorise large composite numbers n. The three methods that are most effective on very large numbers are the quadratic sieve, the elliptic curve method and the number field sieve. Other well-known methods that were precursors include Pollard's rho-method and $p-1$ method, Williams's $p+1$ method, the continued fraction algorithm, and of course, trial division. A good overview of factoring methods can be found in [39, 57].

1.5.1 The RSA problem

The intractability of the RSA problem forms the basis for the security of the RSA public-key encryption scheme and the RSA signature scheme.

Definition 1.5.2 The RSA problem is the following: given a positive integer n that is a product of two distinct odd primes p and q, a positive integer e such that $\gcd(e, (p-1)(q-1)) = 1$, and an integer c, find an integer m such that $m^e \equiv c \pmod{n}$.

Clearly the RSA problem is no more difficult then factorisation, since if p and q can be found then it is simple to find m.

1.6 Discrete logarithms

The security of many cryptographic techniques depends on the intractability of discrete logarithm problem. A partial list of these includes the Diffie-Hellman key agreement [18] and its derivative, the ElGamal encryption (see Section 1.10.2), and the ElGamal signature scheme (see Section 1.11.3) and its variants. This section focuses on the current knowledge regarding methods for solving the discrete logarithm problem.

Let G be a finite cyclic group of order n with generator g. For a more concrete approach, one may find it convenient to think of G as the multiplicative group of integers modulo p (for p prime).

Definition 1.6.1 Let G be a finite cyclic group of order n. Let g be a generator of G, and let $y \in G$. The discrete logarithm of y to the base g, denoted $\log_g y$, is the unique integer x, $0 \leq x \leq n-1$, such that $y = g^x$.

Definition 1.6.2 The Discrete Logarithm Problem (DLP) is the following: given a prime p, a generator g of $\mathbb{Z}_p^*$, and an element $y \in \mathbb{Z}_p^*$, find the integer x, $0 \leq x \leq p-2$, such that $y \equiv g^x \bmod p$.

As in the case of factorisation, efficient techniques exist for solving the discrete logarithm problem when the group G has a particular structure, an example of such a technique being the Pohlig-Hellman algorithm [49], which efficiently computes discrete logarithms when the group G has order $n = p_1^{a_1} p_2^{a_2} \cdots p_r^{a_r}$, where $p_1, p_2, \ldots, p_r$ are primes less than or equal to a small bound B. A good overview of techniques for calculating discrete logarithms can be found in [39].

1.6.1 The Diffie-Hellman problem

The Diffie-Hellman problem is closely related to the well-studied discrete logarithm problem (DLP). It is of significance to public-key cryptography because its apparent intractability forms the basis for the security of many cryptographic techniques including the Diffie-Hellman key agreement and its derivatives, and the ElGamal public-key cryptosystem.

Definition 1.6.3 The Diffie-Hellman problem (DHP) is the following: given a prime p, a generator g of $\mathbb{Z}_p^*$, and elements $g^a \bmod p$ and $g^b \bmod p$, find $g^{ab} \bmod p$.

Definition 1.6.4 The generalized Diffie-Hellman problem (GDHP) is the following: given a finite cyclic group G, a generator g of G, and group elements g^a and g^b, find g^{ab}.

Suppose that the discrete logarithm problem in $\mathbb{Z}_p^*$ could be efficiently solved. Then given $g, p, g^a \bmod p$ and $g^b \bmod p$, one could first find a from g, p and $g^a \bmod p$ by solving a discrete logarithm problem, and then compute $(g^b)^a = g^{ab} \bmod p$. Thus, the DHP is no harder than DLP.

1.7 Complexity theory

Definition 1.7.1 Suppose that for all $n \geq n_0$ the two functions $f(n)$ and $g(n)$ are defined, take positive values, and for some constant C satisfy the inequality $f(n) \leq C \cdot g(n)$. Then we say that $f = O(g)$.

Definition 1.7.2 An algorithm to perform a computation is said to be a **polynomial time algorithm** if for an instance of length at most n there exists an

integer d such that the number of bit operations required to perform the algorithm is $O(n^d)$.

Thus, the usual arithmetic operations $+, -, \times, \div$ are examples of polynomial time algorithms; so is conversion from one base to another. On the other hand, computation of $n!$ is not. (However, if one is satisfied with knowing $n!$ to only a certain number of significant figures, e.g., its first 1000 binary digits, then one can obtain that by a polynomial time algorithm using Stirling's approximation for $n!$)

1.8 Cryptography

The word **cryptology** stems from Greek meaning 'hidden word', and is the umbrella term used to describe the entire field of secret communications. Cryptology splits into two subdivisions: cryptography and cryptanalysis.

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, data origin authentication and non-repudiation. The cryptanalyst seeks to undo the cryptographer's work by breaking a cipher or by forging coded signals that will be accepted as authentic.

General information on cryptography can be found in [39], [57] and [56]. There are two major types of cryptosystems. One is Symmetric-key cryptosystems and the other is Public-key cryptosystems. We will pay particular attention to Public-key cryptosystems. Thus, we first give a formal definition of a cryptosystem:

Definition 1.8.1 A cryptosystem is a five-tuple $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where the following condition are satisfied

1. $\mathcal{M}$ is a finite set of possible **plaintexts** or **messages**

2. $\mathcal{C}$ is a finite set of possible **ciphertexts** or **cryptograms**

3. $\mathcal{K}$ is a finite set of possible **keys**

4. For each $K \in \mathcal{K}$, there is an **encryption rule** $E_K \in \mathcal{E}$ and a corresponding **decryption rule** $D_K \in \mathcal{D}$. Each $E_K : \mathcal{M} \to \mathcal{C}$ and $D_K : \mathcal{C} \to \mathcal{M}$ are functions such that $D_K(E_K(x)) = x$ for every message $x \in \mathcal{M}$.

The main property is property 4. It is this property that enables a user to decrypt a received cryptogram, since $D_K(E_K(x)) = x$ for all messages $x \in \mathcal{M}$. For unambiguous decryption, it is obviously required that $E_K(x_1) \neq E_K(x_2)$ if $x_1 \neq x_2$. Otherwise, if $E_K(x_1) = E_K(x_2)$, $x_1 \neq x_2$, decryption is not unique, and therefore it is not possible for a user to decide whether the intended message was x_1 or x_2 upon receipt of $E_K(x_1) = E_K(x_2)$.

1.9 Symmetric-key cryptography

Consider an encryption scheme consisting of the sets of encryption and decryption transformations $E_K : K \in \mathcal{K}$ and $D_K : K \in \mathcal{K}$, respectively, where $\mathcal{K}$ is the key space. The encryption scheme is said to be *symmetric-key* if for each associated encryption/decryption transformation pair (E_K, D_K), it is computationally "easy" to determine D_K knowing only E_K, and to determine E_K from D_K.

Since $E_K = D_K$ in most practical symmetric-key encryption schemes, the term symmetric-key becomes appropriate. Other terms used in the literature are single-key, one-key and conventional encryption.

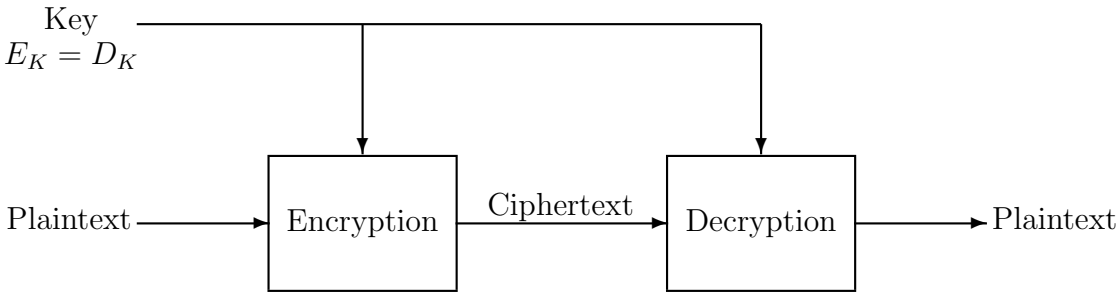

Figure 1.1: Symmetric cryptography

1.10 Public-key cryptography

In 1976, the idea of public-key cryptography was presented by Diffie and Hellman [18]. Although revolutionary, the idea is still very simple.

In public-key cryptosystems, one can safely publicise one's encryption method. This means that also the cryptanalyst will know it. However, he/she is still unable to decrypt your ciphertext. This is what public-key cryptography is all about: the encryption method can be made public.

What does safety in giving away the encryption method actually mean? Of course, the encryption method gives away the decryption method in a mathematical sense because the two are "inverses" of each other. However, suppose it will take hundreds of years for the cryptanalyst to compute the decryption method from the encryption method. Then in practice we do not compromise anything by publicizing the encryption method. This is how "safety" is to be understood.

With regard to Public-key cryptography, it is computationally infeasible to compute the decrypting transformation D_K from the encryption transformation E_K.

The encryption key E_K can therefore be made public without compromising the security of the decryption key D_K. Anyone can then encrypt a message using the public encryption key, but only the intended recipient can decrypt the message using the secret decryption key.

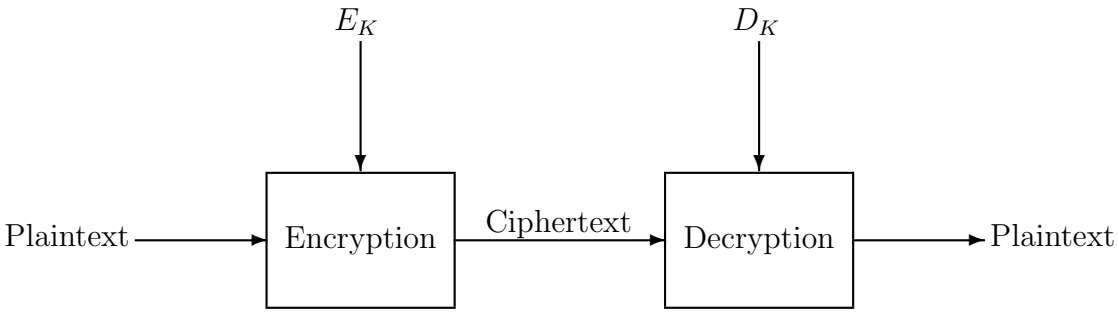

Figure 1.2: Asymmetric cryptography or public-key cryptography

Definition 1.10.1 A one-way function is a function f such that for each x in the domain of f, it is easy to compute $f(x)$; but for essentially all y in the range of f, it is computationally infeasible to find any x such that $y = f(x)$.

Definition 1.10.2 A trapdoor one-way function is a one-way function f with the additional property that given some extra information (the trapdoor information) it becomes computationally feasible to compute for any y in the range of f an x such that $y = f(x)$.

In public-key cryptography, the encryption rule E_K determined by the public encryption key e can be viewed as a trapdoor one-way function, with the secret decryption key d being viewed as the trapdoor.

It has yet to proved that there exist any true one-way functions. Consequently, it has yet to be proved that there exist any true trapdoor one-way functions. There

are, however, several good candidates for one-way functions. One candidate is based on the multiplication of two large primes p and q, the factorisation of the product $n = pq$ being an example of the integer factorisation problem. Another well known candidate one-way function is exponentiation in finite fields, which can be computed efficiently using the 'square and multiply' algorithm, but with the inverse being the discrete logarithm problem. An example of a candidate trapdoor one-way function is the RSA public-key cryptosystem, described in the following subsection.

1.10.1 RSA

The RSA public-key cryptosystem [50] was introduced in 1978, and may be used for both secrecy and digital signatures. The cryptosystem works in $\mathbb{Z}_n$, where n is the product of two large primes p and q, and its security is based on the difficulty of factoring n, that is, the integer factorisation problem as mentioned in section 1.5.

A general overview of the choice of primes p and q can be found in [39], page 290. An overview of the generation of primes can be found in [39], Chapter 4.

To use the RSA public-key cryptosystem, a user A first generates their public and secret keys by

(i) generating two large distinct primes p and q.

(ii) computing $n = pq$ and $\phi(n) = (p-1)(q-1)$, where $\phi(n)$ is as defined in Section 1.4 (Euler Totient Function).

(iii) choosing a random integer e such that $0 < e < \phi(n)$, and $\gcd(e, \phi(n)) = 1$,

(iv) using the Euclidean Algorithm to compute the unique integer d, where $0 < d < \phi(n)$, such that $ed \equiv 1 \pmod{\phi(n)}$, and

(v) publishing the pair (n, e) as the public key, and keeping d as the private key.

RSA is an example of block cipher, that is, a message is encrypted by being broken down into blocks (or strings) of a fixed length, and each block is encrypted individually. The plaintext and the ciphertext space are $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$. To encrypt a message block m for user A, a user B

(i) obtains A's authentic public key (n, e),

(ii) represents the message m as an integer in the range $0, \ldots, (n-1)$,

(iii) computes the ciphertext $E_K(m) = c = m^e \bmod n$, and

(iv) transmits the ciphertext c to user A.

To decrypt the ciphertext c, user A computes $D_K(c) = c^d = m \bmod n$. The fact that RSA works, that is, why $m^{ed} \equiv m \pmod{n}$ is an immediate corollary of Theorem 1.4.15.

RSA has the property that for any two distinct messages m_1 and m_2 with ciphertexts c_1 and c_2 respectively, the ciphertext of $m = m_1 \cdot m_2 \bmod n$ is

$$c \equiv m^e \equiv (m_1 \cdot m_2)^e \equiv m_1^e m_2^e \equiv c_1 \cdot c_2 \pmod{n}.$$

This is often referred to as the homomorphic property of RSA.

1.10.2 ElGamal encryption

The ElGamal public-key cryptosystem [19] is based on the discrete logarithm problem. We begin by describing this cryptosystem in the setting of finite field $\mathbb{Z}_p$.

Thus, the prime p must be chosen such that computing discrete logarithms in $\mathbb{Z}_p$ is hard. In particular, to guard against the Pohlig-Hellman algorithm [49], the prime p must be chosen such that $(p-1)$ contains at least one large prime factor q.

To use the ElGamal cryptosystem, a user A first generates their public and secret keys by the following procedures

(i) Let p be a prime such that the discrete logarithm problem in $\mathbb{Z}_p$ is intractable, and $p-1$ also contains at least one large prime factor q, and let $g \in \mathbb{Z}_p^*$ be a primitive element. Let $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$. The values p and g will often be shared by a group of users and are often referred to as domain parameters.

(ii) Choose a random integer x such that $0 < x < (p-2)$, and define

$$\mathcal{K} = \{(p, g, x, y) : y \equiv g^x \pmod{p})\}$$

(iii) Publish the triple (p, g, y) as the public key, and keep x as the secret key.

Like RSA, ElGamal is a block cipher, that is, a message is encrypted by being broken down into blocks of a fixed length, and each block encrypted individually. To encrypt a message block m for user A, a user B

(i) obtains A's authentic public key (p, g, y),

(ii) represents the message m as an integer in the range $0, \ldots, (p-1)$,

(iii) chooses a random integer k such that $1 \leq k \leq (p-2)$,

(iv) computes $E_K(m, k) = (c_1, c_2)$,

where

$$c_1 = g^k \bmod p,$$

and

$$c_2 = my^k \bmod p,$$

(v) transmits the ciphertexts (c_1, c_2) to A.

To decrypt the ciphertexts (c_1, c_2), user A computes

$$D_K(c_1, c_2) = c_2(c_1^x)^{-1} = m \bmod p.$$

It is essential to the security of ElGamal that the integer k is not used to encrypt more than one message, since if the same integer k is used to encrypt message m and m' with resulting ciphertext (c_1, c_2) and (c_1', c_2'), then

$$c_2(c_2')^{-1} \equiv m(m')^{-1} \pmod p,$$

so that if m is known, m' can easily be computed without knowing x.

1.11 Digital signature

Handwritten signatures have long been used as proof of authorship of, or at least agreement with, the contents of a document. A digital signature is the counterpart to a handwritten signature. A digital signature should comprise of some data which establishes that the signer was the originator. Moreover, it must be such that the receiver can save it as evidence which can be presented to an arbitrator who can then check the validity of the signature and settle any dispute. It should thus prevent fraud such as the forging of a signature by the receiver (or any third party) and the repudiation of the transmission of a message by the sender. A generic description follows.

- $\mathcal{M}$ is the set of messages which can be signed.

- $\mathcal{S}$ is a set of elements called signatures, possibly binary strings of fixed length.
- S_A is a transformation from the message set M to the signature set $\mathcal{S}$, and is called a *signing transformation* for entity A. The transformation S_A is kept secret by A, and will be used to create signatures for messages from $\mathcal{M}$.
- V_A is a transformation from the set $\mathcal{M} \times \mathcal{S}$ to the set {true, false}. V_A is called a *verification transformation* for A's signatures, is publicly known, and is used by other entities to verify signatures created by A. V_A has the property that $V_A(m, s) =$ true, if and only if $s = S_A(m)$

Signing procedure

Entity A (the signer) creates a signature for a message $m \in \mathcal{M}$ by doing the following:

(i) Compute $s = S_A(m)$.

(ii) Transmit the pair (m, s). s is called the signature for message m.

Verification procedure

To verify that a signature s on a message m was created by A, an entity B (the verifier) performs the following steps:

(i) Obtain the verification function V_A of A.

(ii) Compute $u = V_A(m, s)$.

(iii) Accept the signature as having been created by A if $u = true$, and reject the signature if $u = false$.

A digital signature scheme with message recovery can be constructed from a public key encryption scheme by defining the signature of a message m to be $s = D_K(M)$, where $M = R(m)$, and R is some function which adds redundancy to the message m, an example being $R(m) = m||h(m)$, where $||$ denotes concatenation and h is a hash function. Both R and R^{-1} must be made public. The signature s is verified by computing $E_K(s) = M'$, and checking that M' has the correct redundancy corresponding to m and R. Provided that $E_K(s)$ does have the required redundancy, the message m can be recovered by computing $R^{-1}(M') = m$.

1.11.1 One-way hash-functions

Cryptographic hash functions play a fundamental role in modern cryptography. A hash function is a function which maps bit-strings of arbitrary length to bit-strings of a fixed (short) length. For a hash function which outputs n-bit hash-values it is desirable that the probability that a randomly chosen string gets mapped to a particular n-bit hash-value (image) is 2^{-n}. The basic idea is that a hash-value serves as a compact representative of an input string. To be of cryptographic use, a hash function h is typically chosen such that it is computationally infeasible to find two distinct inputs which hash to a common value (i.e., two *colliding* inputs x and y such that $h(x) = h(y)$, and that given a specific hash-value y, it is computationally infeasible to find an input (pre-image) x such that $h(x) = y$.

The most common cryptographic uses of hash functions are with digital signatures and for data integrity. With digital signatures, a long message is usually hashed (using a publicly available hash function) and only the hash-value is signed. The party receiving the message then hashes the received message, and verifies that received signature is correct for this hash-value. This saves both time and space

compared to signing the message directly, which would typically involve splitting the message into appropriate-sized blocks and signing each block individually. Note that the inability to find two messages with the same hash-value is a security requirement, since if there are two messages with the same hash-value then the digital signature on one message's hash-value would be the same as that on another, allowing a signer to sign one message and at a later point in time claim to have signed another.

Hash functions may be used for data integrity as follows. The hash-value corresponding to a particular input is computed at some point in time. The integrity of this hash-value is protected in some manner. At a subsequent point in time, to verify that the input data has not been altered, the hash-value is recomputed using the input at hand, and compared for equality with the original hash-value. Specific applications include virus protection and software distribution.

Definition 1.11.1 A one-way hash-function is a one-way function h which has, as a minimum, the following two properties:

(i) compression –h maps an input x of arbitrary finite bitlength n.

(ii) ease of computation –given h and an input x, $h(x)$ is easy to compute.

In order to meet the requirements of a signature scheme the following three properties are required of a hash-function h with inputs $x,\ x'$ and outputs $y,\ y'$.

1. **preimage resistance**: for essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find any preimage x' such that $h(x') = y$ when given any y for which a corresponding input is not known.

2. **2nd-preimage resistance**: it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x, to find a 2nd-preimage $x' \neq x$ such that $h(x) = h(x')$.

3. **collision resistance**: it is computationally infeasible to find any two distinct inputs $x,\ x'$ which hash to the same output, i.e., such that $h(x) = h(x')$. (Note that there is free choice of both inputs.)

1.11.2 RSA signature

The RSA public-key cryptosystem [50] can be used to provide digital signatures by reversing the roles of encryption and decryption as follows:

The RSA signature with appendix

A user A also generates their public and private keys exactly as in the RSA public-key cryptosystem. The set of users of signatures also need to agree on a hash function h. Then to generate a signature of a message m, user A

(i) computes $M = h(m)$,

(i) computes $s = M^d \bmod n$, and

(ii) outputs s as the signature of m.

To verify the signature, a user B

(i) obtains A's authentic public key (n, e),

(iii) verifies that $s \leq n$; if not, then reject the signature

(ii) computes $M' = s^e \bmod n$,

(iii) computes $M = h(m)$

(iv) accepts the signature s if and only if $M' = M$.

The RSA signature with message recovery

A user A also generates their public and private keys exactly as in the RSA public-key cryptosystem. The set of users of signatures agree on a redundancy function R. Then to generate a signature of a message m, user A

(i) computes $M = R(m)$,

(i) computes $s = M^d \bmod n$, and

(ii) outputs s as the signature of m.

To verify the signature, a user B

(i) obtains A's authentic public key (n, e),

(ii) computes $M' = s^e \bmod n$,

(iii) verifies that M' has the required redundancy, and

(iv) recovers the message $m = R^{-1}(M')$.

Note that due to the homomorphic property of RSA, for any two distinct message m_1 and m_2 with corresponding signatures s_1 and s_2 respectively, the signature of $m = m_1 \cdot m_2 \bmod n$ is

$$s = (m_1 \cdot m_2)^d \equiv m_1^d \cdot m_2^d \equiv s_1 \cdot s_2 \pmod{n}.$$

In particular, for any message m_1 with signature s_1, the signature of $m = -m_1 \bmod n$ is $s = -s_1 \bmod n$. It is important, therefore, that the redundancy function R is not multiplicative, that is, $R(m_1 \cdot m_2) \neq R(m_1) \cdot R(m_2)$.

1.11.3 ElGamal signature

This section presents the ElGamal signature [19] and several related signature schemes such as the DSS [1], the Nyberg-Rueppel signature [46] i.e., the ElGamal signature scheme with message recovery.

The ElGamal signature scheme

A user A generates their public and private keys exactly as in the ElGamal public-key cryptosystem. Then to generate a signature of a message m, user A

(i) selects a random secret k, $1 \leq k \leq p-2$, with $\gcd(k, p-1) = 1$

(ii) computes $r = g^k \bmod p$, and

(iii) computes $k^{-1} \bmod p-1$,

(iv) computes $s = k^{-1}(m - xr) \bmod p-1$

(v) outputs the pair (r, s) as the signature.

To verify the signature, a user B

(i) obtains A's authentic public key (g, p, y),

(ii) verifies that $1 \leq r \leq p-1$; if not, then reject the signature

(iii) computes $v_1 = y^r \, r^s \bmod p$, and

(iv) computes $v_2 = g^m \bmod p$

(v) accept the signature if and only if $v_1 = v_2$.

Note that in step (iv) of signature generation it is standard practice to replace m with $h(m)$

(a) to allow signature generation on long messages.

(b) to prevent attacks.

where h is a hash function which has been agreed by the set of users.

The Digital Signature Standard (DSS)

To use the DSS, a user A first generates their public and private keys by

(i) letting p be a large prime such that the discrete logarithm problem in $\mathbb{Z}_p$ is intractable, letting q be a large prime that divides $p-1$, and letting $g \in \mathbb{Z}_p^*$ be a qth root of 1 modulo p,

(ii) choosing a random integer x such that $0 < x < q$, and calculating $y = g^x \bmod p$,

(iii) publishing the four tuples (p, q, g, y) as the public key, and keeping x as the secret key.

The set of users of signatures agree on a hash function h. Then to generate a signature of a message m, user A

(i) selects a random secret k, $1 \leq k \leq q-1$,

(ii) computes $r = (g^k \bmod p) \bmod q$,

(iii) computes $k^{-1} \bmod q$

(iv) computes $s = k^{-1}\{h(m) + xr\} \bmod q$

(v) outputs the pair (r, s) as the signature.

To verify the signature, a user B

(i) obtains A's authentic public key (p, q, g, y),

(ii) verifies that $1 \leq r \leq q-1$ and $1 \leq s \leq q-1$; if not, then reject the signature

(iii) computes $v_1 = s^{-1}h(m) \bmod q$ and $v_2 = rs^{-1} \bmod q$.

(iv) accept the signature if and only if $g^{v_1}y^{v_2} = r$.

The ElGamal signature scheme with message recovery

A user A also generates their public and private keys exactly as in the DSS. The set of users of signatures agree on a redundancy function R. Then to generate a signature of a message m, user A

(i) computes $M = R(m)$,

(ii) selects a random secret k, $1 \leq k \leq q-1$, and computes $r = g^{-k} \bmod p$, and

(iii) computes $e = Mr \bmod p$

(iv) computes $s = xe + k \bmod q$

(v) outputs the pair (e, s) as the signature.

To verify the signature, a user B

(i) obtains A's authentic public key (g, p, q, y),

(ii) verifies that $1 \leq r \leq p-1$; if not, then reject the signature

(iii) verifies that $1 \leq s \leq q-1$; if not, then reject the signature

(iv) computes $v = g^s y^{-e} \bmod p$ and $M' = ve \bmod p$.

(v) verifies that M' has the required redundancy, and

(vi) recovers the message $m = R^{-1}(M')$.

1.12 Properties of an authentication protocol

Authentication may be informally defined as the process of verifying that an identity is as claimed, including who, what, and when.

ISO 7498-2 [2] distinguishes between data origin authentication (i.e. verifying the origin of received data - a connectionless operation), and entity authentication (i.e. verifying the identity of one entity by another - a connection-oriented operation).

Definition 1.12.1 Entity authentication is the process whereby one party is assured (through acquisition of corroborative evidence) of the identity of a second party involved in a protocol, and that the second has actually participated (i.e., is active at, or immediately prior to, the time the evidence is acquired).

ISO 7498-2 [2] defines entity authentication as 'the corroboration that an entity is the one claimed'. We also need to distinguish between protocols providing unilateral authentication and those providing mutual authentication. Unilateral au-

thentication is defined as 'entity authentication which provides one entity with assurance of the other's identity but not vice versa' and mutual authentication is defined as 'entity authentication which provides both entities with assurance of each other's identity'.

Definition 1.12.2 Data origin authentication techniques provide to a party which receives a message assurance (through corroborative evidence) of the identity of the party which originated the message.

We now consider what properties we require an authentication protocol to satisfy. To do this we set up a model of an authentication protocol and consider what we might require of a protocol in general.

We suppose that there are a pair of communicating entities A and B. A and B wish to use an authentication protocol to provide either unilateral or mutual authentication. We suppose that the protocol consists of a finite sequence of messages as follows:

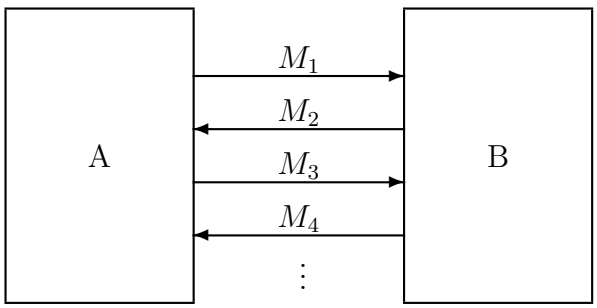

Figure 1.3: Authentication protocol model

More specifically, we suppose that A starts the protocol and sends B the message M_1. B then sends A the message M_2. A then sends B the message M_3, and so on

until the protocol is complete. Note that we assume that the messages sent by A and B are always alternating.

In addition we always assume that A or B will not send message M_i until message M_{i-1} has been correctly received (i.e. all cryptographic checks performed on the message give valid results).

1.12.1 Requirements

After completion of the protocol, both parties will have 'belief' requirements regarding the messages exchanged. We first consider the beliefs that A wishes to have (given that the protocol is designed to prove B's identity to A).

A will want to be sure that:

1. $M_2, M_4, \ldots$ were all sent by B (as received),
2. $M_2, M_4, \ldots$ are 'fresh', i.e. not replays of old messages,
3. $M_2, M_4, \ldots$ were intended for A, and not for any other entity, and
4. $M_2, M_4, \ldots$ were only generated by B after $M_1, M_3, \ldots$ (respectively) were received correctly by B.

In a similar way, B will want to be sure that:

1. $M_1, M_3, \ldots$ were all sent by A (as received),
2. $M_1, M_3, \ldots$ are 'fresh', i.e. not replays of old messages,
3. $M_1, M_3, \ldots$ were intended for B, and not for any other entity, and

4. $M_3, M_5, \ldots$ were only generated by B after $M_2, M_4, \ldots$ (respectively) were received correctly by A.

In some sense property (4) implies both of properties (1) and (2), except for the special case of B knowing message M_1 was sent by A and is fresh (since property (4) has nothing to say about message M_1).

It is important to observe that A and B may well not be sure of these 'beliefs' until after completion of the protocol.

The exact list of properties needed for a protocol in a particular application may be a subset of these lists; however, if we are defining 'generic' protocols for widespread use (as in ISO/IEC 9798 [6]) then our selected protocol [5] in Chapter 4 should have the specified properties which have been described above.

1.12.2 Authentication mechanisms

Authentication protocols require the use of a combination of either shared secrets (keys or passwords) or signature/verification key pairs and accompanying cryptographic mechanisms. These are used to ensure that the recipient of a protocol message knows:

- where it has come from (original checking),
- that it has not been interfered with (integrity checking).

Note that cryptographic mechanisms (by themselves) cannot provide **freshness checking**, i.e. the verification that a protocol message is not simply a replay of a previously transmitted (valid) protocol message, protected using a currently valid key. We consider the provision of freshness verification later.

A variety of different types of cryptographic mechanism can be used to provide integrity and origin checking for individual protocol messages. We mention three main possibilities:

- encipherment,
- integrity mechanism (MAC or Cryptographic Check Function),
- digital signature.

1.12.3 Freshness mechanisms

As we have already briefly noted, providing origin and integrity checking for protocol messages is not all that is required. We also need a means of checking the 'freshness' of protocol messages to protect against replays of messages from previous valid exchanges.

There are two main methods of providing freshness checking:

- the use of **time stamps** (either clock-based or 'logical' time-stamps),
- the use of **nonces** or challenges (as in the challenge-respond protocols).

We consider these two approaches in turn.

Clearly the inclusion of a **date/time stamp** in a message enables the recipient to check it for freshness, as long as the time-stamp is protected by cryptographic means.

However, in order for this to operate successfully all entities must be equipped with securely synchronised clocks.

Every entity receiving protocol messages will need to define a time acceptance 'window' either side of their current clock value. A received message will then be accepted as 'fresh' if and only if it falls within this window. This acceptance window is needed for two main reasons:

- clocks vary continuously, and hence no two clocks will be precisely synchronised, except perhaps at some instant in time, and
- messages take time to propagate from one machine to another, and this time will vary unpredictably.

The use of an acceptance window is itself a possible security weakness, since it allows for undetectable replay of messages for a period of time up to the length of the window. To avert this threat requires each entity to store a 'log' of all recently received messages.

One alternative to the use of clocks is for every pair of communicating entities to store a pair of **sequence numbers**, which are used only in communications between that pair. For example, for communications between A and B, A must maintain two counters: N_{AB} and N_{BA} (B will also need to maintain two counters for A).

Every time A sends B a message, the value of N_{AB} is included in the message, and at the same time N_{AB} is incremented by A.

Every time A receives a message from B, then the sequence number put into the message by B (N say) is compared with N_{BA} (as stored by A):

- if $N > N_{BA}$ then

  (i) the message is accepted as fresh, and

(ii) N_{BA} is reset to equal N,

- if $N \leq N_{BA}$ then

(i) the message is rejected as an 'old' message.

Note that all time stamp protocols (both clock-based and logical time-stamp based) have problems in providing property (4) from the list of desired properties for an authentication protocol.

Nonce-based (or challenge-response) protocols use a quite different mechanism to provide freshness checking. One party A say, sends the other party, B say, a nonce (Number used ONCE) as a challenge. B then includes this nonce in the response to A. Because the nonce has never been used before, at least within the lifetime of the current key, A can verify the 'freshness' of B's response (given that message integrity is provided by some cryptographic mechanism).

Note that it is always up to A, the nonce provider, to make sure it is 'new'. The main property is the 'one-time' property, and this, in theory, could be provided using a counter.

However, in order to prevent a special type of attack, many protocols also need nonces to be unpredictable to any third party. More specifically, there are problems with provision of properties (2) and (4) with some nonce-based protocols if nonces can be predicted by third parties. Hence nonces are typically chosen at random from a set sufficiently large to mean that the probability of the same nonce being used twice is effectively zero.

We will consider an example of an authentication protocol, its shortcoming and a fix in Chapter 4.

1.13 Key establishment, management, and certification

Cryptographic techniques depend upon cryptographic keys. Management of these keys is itself a complex subject and a crucial aspect of providing security.

Key management includes ensuring that key values generated have the necessary properties, making keys known in advance to the parties that will use them, and ensuring that keys are protected as necessary against disclosure and/or substitution. The methods of key management vary substantially depending on whether the keys being managed are those of symmetric cryptosystems or of public-key cryptosystems. This section gives a brief introduction to a methodology for ensuring the secure distribution of keys for cryptographic purposes.

Key establishment protocols and related cryptographic techniques provide shared secrets between two or more parties, typically for subsequent use as symmetric keys for a variety of cryptographic purposes including encryption, message authentication, and entity authentication. The main focus here is two party key establishment, with the aid of a trusted third party in some cases.

Definition 1.13.1 Key management is the set of processes and mechanisms which support key establishment and the maintenance of ongoing keying relationships between parties, including replacing older keys with new keys as necessary.

Definition 1.13.2 Key establishment is a process or protocol whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use.

Key establishment may be broadly subdivided into key transport and key agreement, as defined below.

Definition 1.13.3 A key transport protocol or mechanism is a key establishment technique where one party creates or otherwise obtains a secret value, and securely transfers it to the others.

Definition 1.13.4 A key agreement protocol or mechanism is a key establishment technique in which a shared secret is derived by two or more parties as a function of information contributed by, or associated with, each of these, such that no party can predetermine the resulting value.

1.13.1 Key management through public-key techniques

There are a number of ways to address the key management problem through public-key techniques. Chapter 4 describes some of these in detail. For the purpose of this section a very simple model is considered.

Each entity in the network has a public/private encryption key pair. The public-key along with the identity of the entity is stored in a central repository called a **public file**. If an entity A_i wish to send encrypted messages to entity A_j, A_i retrieves the public-key e_j of A_j from the public file, encrypts the message using this key, and sends the ciphertext to A_j, where $i = 1, \ldots, n,\ j = 1, \ldots, n,\ i \neq j$.

Advantages of this approach include:

(i) No trusted third party is required,

(ii) The public file could reside with each entity,

(iii) Only n public keys need to be stored to allow secure communications between any pair of entities, assuming the only attack is that by a passive adversary.

The key management problem becomes more difficult when one must take into account an adversary who is active (i.e. an adversary who can alter the public file containing public keys). For example, the adversary alters the public file by replacing the public key e_j of entity A_j by the adversary's public key e_*. Any message encrypted for A_j using the public key from the public file can be decrypted by only the adversary. Having decrypted and read the message, the adversary can now encrypt it using the public key of A_j and forward the ciphertext to A_j. A_i however believes that only A_j can decrypt the ciphertext c.

To prevent this type of attack, the entities may use a Trusted Third Party (TTP) to certify the public key of each entity. The TTP has a private signing algorithm and a verification algorithm assumed to be known by all entities. The TTP carefully verifies the identity of each entity, and signs a message consisting of an identifier and the entity's authentic public key. This is a simple example of a **certificate**, binding the identity of an entity to its public key. A_i uses the public key of A_j only if the certificate signature verifies successfully.

Advantages of using a TTP to maintain the integrity of the public file include the following.

(i) It prevents an active adversary from impersonation on the network.

(ii) The TTP cannot monitor communications. Entities need only trust the TTP to bind identities to public keys properly.

(iii) Per-communication interaction with the public file can be eliminated if entities store certificates locally.

Even with the use of TTPs, some concerns still remain:

(i) If the signing key of the TTP is compromised, all communications become insecure.

(ii) All trust is placed with one entity.

1.13.2 Key authentication and key confirmation

It is generally desired that each party in a key establishment protocol be able to determine the true identity of all other entities who could possibly gain access to the resulting key, implying preclusion of any unauthorised additional parties from deducing the same key.

Definition 1.13.5 Key authentication is the property whereby one party is assured that no other party aside from a specifically identified second party (and possibly additional identified trusted parties) may gain access to a particular secret key.

Definition 1.13.6 Key confirmation is the property whereby one party is assured that a second (possibly unidentified) party actually has possession of a particular secret key.

Note that in key authentication it is the identity of the second party which is the significant feature, rather than the value of the key, whereas in key confirmation the opposite is true. Key confirmation typically involves one party receiving a message from the second containing evidence demonstrating the latter's possession of the key.

1.13.3 Adversaries in key establishment

When examining the security of protocols, it is assumed that the underlying cryptographic mechanisms used, such as encryption algorithms and digital signatures schemes, are secure. If not, then there is no hope of a secure protocol. An adversary is hypothesized to be not a cryptanalyst attacking the underlying mechanisms directly, but rather one attempting to subvert the protocol objectives by defeating the manner in which such mechanisms are combined, i.e., attacking the protocol itself. In addition, the presence of an unauthorised third party is given many names under various circumstances, including: adversary, intruder, opponent, enemy, attacker, eavesdropper and impersonator.

Definition 1.13.7 A **passive attack** involves an adversary who attempts to defeat a cryptographic technique by simply recording data and thereafter analysing it (e.g., in key establishment, to determine the session key). An **active attack** involves an adversary who modifies or injects messages.

An adversary in a key management protocol may pursue many strategies, including attempting to:

(i) deduce a session key using information gained by eavesdropping,

(ii) participate covertly in a protocol initiated by one party with another, and influence it, e.g., by altering messages so as to be able to deduce the key,

(iii) initiate one or more protocol executions, and combine (interleave) messages from one with another, so as to masquerade as some party or carry out one of the above attacks,

(iv) without being able to deduce the session key itself, deceive a legitimate party regarding the identity of the party with which it shares a key. A protocol susceptible to such an attack is not resilient.

In entity authentication, where there is no session key to attack, an adversary's objective is to arrange that one party receives messages which satisfy that party that the protocol has been run successfully with a party other than the adversary.

Distinction is sometimes made between adversaries based on the type of information available to them. An **outsider** is an adversary with no special knowledge beyond that generally available, e.g., by eavesdropping on protocol messages over open channels. An **insider** is an adversary with access to additional information (e.g., session keys or secret partial information), obtained by some privileged means (e.g. physical access to computer resources, conspiracy, etc). A one-time insider obtains such information at one point in time for use at a subsequent time, a permanent insider has continual access to privileged information.

1.14 Secret sharing

Secret sharing is an important and widely studied tool in cryptography and distributed computation. Informally, a secret sharing scheme is a protocol in which a dealer distributes a secret among a set of participants such that only specific subsets of them, defined by the access structure, can recover the secret at a later time. The surveys by Stinson [57] (see Chapter 11) and Simmons [56] (see Chapter 9) provide a general description of secret sharing schemes.

Much research in the area of secret sharing has concentrated on the size of the shares. Although the size of the shares is important because the shares have to

be transmitted and stored secretly, this is not the only information the participants must know to reconstruct the secret. Additional knowledge needed includes, for example, the identity of the participants and the description of the protocol, including the access structure. These parameters are publicly known, but at the same time it is vital that they are authentic, i.e. that no malicious participant has changed these descriptions. This is particularly important if the participants are computer systems that receive the descriptions over a potentially insecure communications link.

Suppose that you want to give enough information to a group of people so that a secret (e.g., a key or password), which we think of as an integer, say N, can be determined by any group of t of them, but if only $t-1$ collaborate they learn nothing about the secret. This problem can be solved by means of a secret sharing scheme which is a multi-party protocol related to key establishment. The original motivation for secret sharing is the following. To safeguard cryptographic keys from loss, it is desirable to create backup copies, although these copies are themselves a security risk. Secret sharing addresses this issue by allowing enhanced reliability without increased risk. They also facilitate distributed trust or shared control for critical activities (e.g., signing corporate cheques, opening bank vaults), by requiring cooperation by t out of n users for access to a critical action.

The idea of **secret sharing** is to start with a secret, and divide it into pieces called **shares** which are distributed amongst users such that the pooled shares of specific subsets of users allow reconstruction of the original secret. This may be viewed as a **key distribution** technique, facilitating one-time key establishment, wherein the recovered key is pre-determined (static), and in the basic case, the same for all groups. We provide below a **simple definition of a secret sharing scheme**.

Definition 1.14.1 A secret sharing scheme is a protocol involving a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of participants and a dealer D, where $D \notin \mathcal{P}$. Let $\Gamma \subset 2^{\mathcal{P}}$ be the set of subsets of participants permitted access to the secret; this is called the *access structure*. The dealer D chooses a secret K and distributes privately to each participant $P_i \in \mathcal{P}$ a share S_i of K such that:

(i) any authorized set $X \in \Gamma$ can reconstruct the secret K from its shares,

(ii) no unauthorized set $X \notin \Gamma$ can do so.

Definition 1.14.2 Let $\Gamma^* \subset \Gamma$ be the set of *minimal* authorised sets, that is, of sets $X \in \Gamma$ such that: $Y \subseteq X$ and $Y \in \Gamma$ implies that $Y = X$.

Definition 1.14.3 A secret sharing scheme is **perfect** if the shares corresponding to each unauthorized subset provide absolutely no information about the shared secret.

Definition 1.14.4 Let t, n be positive integers, $t \leq n$. A (t, n)**-threshold scheme** is a method of sharing a **key** K among a set of n participants (denoted by $\mathcal{P}$), in such a way that any t participants can compute the value of K, but no group of $t-1$ participants can do so. In other words it is a secret sharing scheme for which the access structure Γ consists of all t-subsets of $\mathcal{P}$.

1.14.1 The Shamir threshold scheme

The Shamir (t, n)-threshold scheme [53] is based on polynomial interpolation over a finite field $\mathbb{F}_p$, where p is a prime, and the fact that a polynomial $f(x)$ of degree $t-1$ is uniquely determined by a set of t pairs $(x_i, f(x_i))$, where all x_i are distinct.

Let the set of possible secrets be $\mathcal{K} = \mathbb{Z}_p$, where $p \geq t+1$ is prime, and let $S = \mathbb{Z}_p$. The secret key will be an element of $\mathbb{Z}_p$, as will be the shares given to the participants. The Shamir threshold scheme is described as follows:

(i) Initialization phase

D chooses t distinct, non zero elements of $\mathbb{Z}_p$, denoted x_i, $1 \leq i \leq t$ (this is where we require $p \geq t+1$). For $1 \leq i \leq t$, D gives the value x_i to P_i. The values x_i are public.

(ii) Distribution phase

1. Suppose D wants to share a key $K \in \mathbb{Z}_p$. D secretly chooses (independently at random) $t-1$ elements of $\mathbb{Z}_p$, $a_1, \ldots, a_{t-1}$.
2. For $1 \leq i \leq t$, D computes $y_i = f(x_i)$, where
$$f(x) = K + \sum_{j=1}^{t-1} a_j x^j \bmod p.$$
3. For $1 \leq i \leq t$, D gives the share y_i to P_i.

(iii) Reconstruction phase

Any set $X \in \Gamma$ of t where $X = \{x_{i_1}, \ldots, x_{i_t}\}$ or more participants can reconstruct the secret key $K = f(0)$ by substituting $x = 0$ into the Lagrange interpolation formula:
$$K = \sum_{j=1}^{t} y_{i_j} \prod_{1 \leq k \leq t, k \neq j} \frac{x_{i_k}}{x_{i_k} - x_{i_j}}.$$

In 1988, Tompa and Woll [58] demonstrated that Shamir's original (t, n) threshold scheme is vulnerable to cheating. That is, the last participant of an authorised set can always cheat the other participants during the reconstruction of the secret, without being detected. As a result, the dishonest participant obtains the true secret while the other participants obtain a false one.

1.14.2 Secret sharing schemes with extended capabilities

Secret sharing schemes with a variety of extended capabilities exist, including:

(i) pre-position secret sharing schemes. All necessary secret information is put in place excepting a single (constant) share which must later be communicated, e.g., by broadcast, to activate the scheme.

(ii) dynamic secret sharing schemes. There are pre-positioned schemes wherein the secrets reconstructed by various authorised subsets vary with the value of communicated activating shares.

(iii) multi-secret threshold schemes. In these secret sharing schemes different secrets are associated with different authorised subsets.

(iv) detection of cheaters and **verifiable secret sharing**. These schemes respectively address cheating by one or more group members, and the distributor of the shares.

(v) secret sharing with disenrollment. These schemes address the issue that when a secret share of a (t, n) threshold scheme is made public, it becomes a $(t-1, n)$ scheme.

Chapter 2

Online Secret Sharing Schemes

2.1 Introduction

In this chapter we will focus on novel computationally secure secret sharing schemes for general access structures, where all shares are as short as the secret.

Online secret sharing schemes provide the capability to dynamically change the secret and add participants, without having to redistribute new shares secretly to the current participants.

These capabilities are traded for the need to store **online** additional authentic (but not secret) information at a publicly accessible location, e.g. on a notice board. Alternatively, this information can be broadcast to the participants over a public channel. In particular, online secret sharing schemes have the following properties:

(i) All shares that must be transmitted and stored secretly once for every participant are as short as the secret.

(ii) Different secrets can be shared with different access structures without requiring the shares held by participants to change. This includes the ability for the dealer to change the secret after the shares have been distributed.

(iii) The dealer can update the scheme online: When a new participant is added and the access structure is changed, already distributed shares remain valid. Apart from the new participant's share that is secretly transmitted to him, only publicly readable information has to be changed.

Compared to traditional secret sharing schemes, online secret sharing schemes are very flexible and use only small shares. The difference lies in the additional use of publicly accessible information and in the security model being based on computational security. As for the use of authentic storage, we note that public information is needed in all traditional secret sharing schemes, and that authenticity usually costs much less than secrecy to implement.

Online secret sharing schemes have potential practical applications in situations where the participants and the access rules or the secret itself frequently change. No new shares have to be distributed secretly when new participants are included or participants leave. Such situations often arise in key management and escrowed encryption systems.

For example, consider a high security area within a bank where employees and managers are not permitted outside of normal working hours. At such times, only groups consisting of one manager and at least two employees may enter and a secret sharing scheme is used to share the access code. If, for instance, a manager is fired, he may disclose his share. With an online secret sharing scheme, only the access code and the notice board have to be updated, and the other managers and employees do not have to be given new shares.

Another example is a group of frequently changing participants of varying size, where at all times two thirds of the current group members are needed to invoke some action, for instance to reconstruct a master key used for escrowing keys of malicious users.

In this Chapter we will study and analyse online secret sharing schemes. In section 2.2, we give a model for online secret sharing schemes. We describe a method for detecting cheating and identifying *all* cheaters within the context of this general model. All the schemes considered in the remainder of this chapter are concrete examples derived from the basic model. Note that this general model, whilst implicit to the previously proposed schemes, has not previously been made explicit. In section 2.3, we study an online secret sharing scheme with general access structures due to Cachin [13]. This scheme has shares as short as the secret, and participants may be dynamically added or deleted without having to redistribute new shares secretly to the existing participants. However, this scheme does not allow the shares to be reused after the secret has been reconstructed without a further distributed computation subprotocol such as Goldreich *et al.* [22]. Alternatively, for access to a predetermined number of secrets in fixed order, an additional mechanism such as a variant of the one time user authentication protocol of Lamport [31] needs to be used.

Section 2.4 presents Pinch's scheme [48] which is a modified version of the Cachin protocol in which the participants' shares can be reused, thus overcoming these problems. It is based on the intractability of the Diffie-Hellman problem [18].

In section 2.5, we review the work of Ghodosi *et al.* [21], who pointed out that Pinch's scheme is vulnerable to cheating. They then modified the Pinch scheme in such a way that cheating is prevented.

However Ghodosi *et al.*'s modification to Pinch's scheme does not protect a minority of participants of the authorised set from a colluding majority, who falsely accuses the minority of cheating. Thus, we propose a further novel modification to Pinch's scheme to fix this problem in section 2.6.

In section 2.7, we propose a new scheme for computationally secure online secret sharing, in which the shares of the participants can be reused. The security of the scheme is based on the intractability of factoring. This scheme has the advantage that it detects cheating and enables the identification of *all* cheaters, regardless of their numbers, improving on the previous results by Pinch and Ghodoshi *et al.*

2.2 Model for online secret sharing

The main reasons for using online secret sharing in comparison to a traditional secret sharing are as follows:

In traditional secret sharing, the shares and key are fixed when the scheme is set up. Once the secret has been constructed, the participants' shares cannot be used again. Shares must be distributed to share new secrets or change the access structure. Whereas in online secret sharing, a participant can be added or removed **online** without having to distribute new secret shares to existing participants and the participants' shares can be used to construct **multiple secrets**. These capabilities are gained by using data stored in a publicly accessible location such as a notice board. Moreover, with this facility one can **detect cheating** by comparing the hashed value of the reconstructed secret with the hashed value of the secret which is stored in the publicly accessible location. Furthermore, one can **identify cheaters** by checking the signatures of the messages sent during the reconstruction

phase.

2.2.1 Requirements

There are four requirements for the general model for online secret sharing of multiple secrets.

1. Online: It must provide a method to update the scheme online.
2. Multiple secrets: It must provide a method to construct multiple secrets.
3. Detect cheating: It must provide a method to detect cheating when a participant or a group of collaborating participants deliberately contributes incorrect shares to the protocol and prevents incorrect reconstruction of the secret.
4. Identify cheaters: It must provide a method to identify cheaters once the cheating has occurred.

2.2.2 Properties of Model

We now describe the properties of the model which meet the above mentioned requirements.

1. The simplest example of a secret sharing scheme is where the key K is the exclusive-or of the shares $S_x, x \in X$, held by the participants of an authorised subset X.

$$K = \sum_{x:P_x \in X} S_x.$$

As Cachin [13] has shown this can be turned into an online scheme for a set Y of participants by choosing K and the shares S_y arbitrarily and publishing a function f and a value T_X for each authorised subset $X \subseteq Y$ where

$$T_X = K - f(\sum_{x:P_x \in X} S_x).$$

For new values of K and new authorised subset X it is sufficient to publish new value T_X. The function f must be one-way to ensure that information about the shares cannot be deduced by manipulating such equations. It is this method that we adopt to make our scheme online.

A participant can be added or removed online without having to distribute new secret shares to existing participants. These capabilities are gained by using data stored on the notice board. Also, the dealer can update the publicly readable information and the access structure from a publicly accessible location.

2. When multiple secrets are to be shared, it is sufficient to have a different one-way function f for each key K. However, this requires that the reconstruction protocol is such that the value $\sum_{x:P_x \in X} S_x$ does not become public in the reconstruction phase. This would be achieved, for example, by using a distributed computation subprotocol such as Goldreich *et al.* [22] in the reconstruction phase. Alternatively one can protect the shares from being revealed by encapsulating them. Thus instead of a simple sum, a one-way function of the shares is calculated in such a way that only an encapsulation of share from which no information about the share can be calculated, becomes public. In this case multiple secrets can be shared by using different encapsulating functions.

   A dealer and the participants use an encapsulation function to safeguard the

participants' shares to ensure that the reconstruction of one or more does not compromise the others of a collection of multiple secrets.

3. The public notice board provides an efficient means for detecting cheating. A dealer publishes a hash of the secret key on the notice board which is then compared with the hash of the reconstructed secret key by each participant to detect cheating.

4. The identification of cheaters depends on having a method of verifying what each participant has contributed to the reconstruction of the secret. This can be achieved using digital signatures.

   Each participant signs the encapsulated share. If cheating is detected, then the dealer verifies the signatures of the encapsulated share sent during the reconstruction phase to identify cheaters.

2.2.3 Preliminaries

We adopt the simple definition of secret sharing scheme (see Section 1.14). Let g and f be globally agreed functions, and let e be a encapsulation function which is used to conceal the secret shares. We shall require that finding the participant's shares is computationally infeasible given knowledge of g, f, K, e and the publicly accessible information. We also require that it is easy to compute K given knowledge of g, f, e, the participants' shares, and publicly accessible information.

We shall also make use of a one-way hash-function h which is collision-resistant (for further information see Definition 1.11.1 and Sections 9.2 and 9.7 of [39]). In order to identify all cheaters, every participant will use an agreed digital signature scheme, and must have selected a private/public key pair for this scheme. More-

over, every participant must have a means of obtaining a verified copy of the public signature verification key of every other participant. This could, for example, be provided by having a Trusted Third Party (e.g. the dealer, D) certify the public key of every participant, and having every participant distribute their certificate with every signed message they send.

2.2.4 The protocol

We present a model for online secret sharing scheme in which the participants of an authorised set compute the secret K by combining their encapsulated secret shares with a function f. In this way the participants will not reveal their secret shares during the process of recovering K. The protocol uses a publicly accessible location, e.g. a notice board, where the dealer can store non-forgeable information accessible to all participants. This location will, at least, indicate the number of participants n, the access structure Γ, and the function f for each authorised subset X of participants.

There are two versions of the protocol, according as the participants of an authorised set

(Version 1) perform calculation in sequence (each participant using the result of his/her predecessor in his/her own calculation) or

(Version 2) perform independent calculations which are then combined in a final single calculation.

Note that this protocol relies on globally agreed functions g and f.

Distribution phase

The basic protocol to share the secret K according to an access structure Γ is as follows.

First the dealer D randomly chooses secret shares $S_i,\ i = 1, 2, \ldots, n$. Then D transmits each share S_i to participant P_i over a secret channel in Figure 1.4, and securely stores S_i for subsequent use to identify cheaters, if cheating is detected. For each minimal authorised set $X \in \Gamma^*$ the dealer D selects an encapsulation function e and finds a value T_X for which

(Version 1) Calculation in sequence

$$K = g(T_X, f(e(S_1, \ldots, S_t)))$$

where $e(S_1, \ldots, S_t)$ denotes the outcome of a sequence of encapsulating function by participants $P_1, \ldots, P_t$ or

(Version 2) Independent calculation

$$K = g(T_X, f(e(S_1), \ldots, e(S_t)))$$

where $e(S_i)$ denotes encapsulation by participant P_i.

Thus the globally agreed function g should be invertible to allow the recovery of T_X from K and the output of f. Example of possible candidates for the function g include the invertible group operation or exclusive-or of bit-strings.

The function f must be one-way. Note that examples of one-way functions include RSA, ElGamal and one-way collision-resistant hash-functions such as MD4, MD5 and SHA-1.

The properties of the encapsulation function e are described below in the reconstruction method 1 and 2.

In order to detect cheating the dealer D posts the following items on the notice board: (X, T_X) for every $X \in \Gamma^*$, and the value $h(K)$. Also note that we assume that the items on the notice board can be altered only by the dealer D. In other words, only the dealer can write data items on the notice board in Figure 1.4. We assume that each participant can read data items from the notice board.

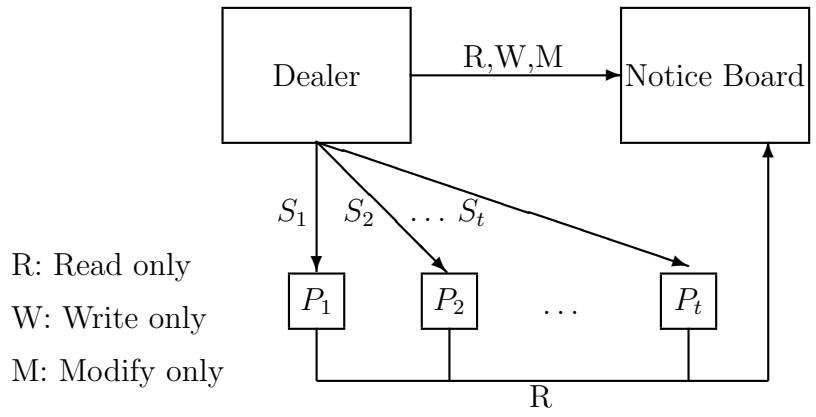

Figure 1.4: Online secret sharing

We now describe two different ways to perform secret reconstruction. In version 1 reconstruction method 1 relies on a 'last participant' who reconstructs the secret on behalf of a minimal trusted set. In version 2 reconstruction method 2 diffuses the responsibility among all participants who reconstruct the secret.

Reconstruction method 1

In this sequential method participant P_i uses the output of participant P_{i-1}, denoted $e(S_1, \ldots, S_{i-1})$, to encapsulate his/her share S_i and we write $e(S_1, \ldots, S_i)$ for his/her output too.

Note that function e is used to conceal the values of secrets $S_1, S_2, \ldots, S_t$. Thus, function e must have the following properties for use in this reconstruction method.

(i) Given S_i and $e(S_1, S_2, \ldots, S_{i-1})$ it must be feasible to compute $e(S_1, S_2, \ldots, S_i)$.

(ii) Knowledge of $e(S_1, S_2, \ldots, S_i)$ yields no useful information about the values of $S_1, S_2, \ldots, S_i$

To recover the secret K, a minimal trusted set $X = \{P_1, P_2, \ldots, P_t\}$ of participants comes together and performs the following steps:

Step 1 Participant P_1 reads $h(K)$ and computes and sends $e(S_1)$ and a signature on a data string consisting of $e(S_1)$ and X (using his/her private signature key) i.e. $s_{P_1} = sign_{P_1}(e(S_1)||X)$ to P_2, where $||$ denotes concatenation of data items.

Step 2 Each subsequent participant P_i, for $1 < i < t$, reads $h(K)$ and receives $e(S_1, S_2, \ldots, S_{i-1})$. P_i verifies the signature, $s_{P_{i-1}} = sign_{P_{i-1}}(e(S_1, S_2, \ldots, S_{i-1})||X)$, it received from the previous participant P_{i-1} by using his/her public key. P_i then computes $e(S_1, S_2, \ldots, S_i)$ from S_i and $e(S_1, S_2, \ldots, S_{i-1})$, signs it, and sends the result and its signature s_{P_i} to P_{i+1}.

Step 3 The final participant P_t also reads $h(K)$ and receives $e(S_1, S_2, \ldots, S_{t-1})$, verifies the signature $s_{P_{t-1}}$, and uses this value with S_t to form

$$V_X = e(S_1, S_2, \ldots, S_t).$$

The last participant sends V_X to the other participants using a secret channel.

Step 4 Each participant reads T_X from the notice board and can now reconstruct K as $K = g(T_X, f(V_X))$.

One can easily verify the completeness of the protocol: every authorised subset $X \in \Gamma^*$ can recover K.

Reconstruction method 2

In this method the participants individually encapsulate their shares using a function e and the results are combined.

Note that the function e is used to conceal the values of secrets $S_1, S_2, \ldots, S_t$. Function e must have the following properties for use in this reconstruction method.

(i) Given S_i it must be feasible to compute $e(S_i)$.

(ii) Knowledge of $e(S_i)$ yields no useful information about the value of S_i.

A minimal authorised set $X \in \Gamma^*$ of participants can compute K by performing the following steps:

Step 1 Each participant $P_i \in X$ reads $h(K)$ and the value T_X corresponding to the appropriate set X from the notice board. Then P_i signs the data string consists of $e(S_i)$ and X (using his/her private signature key) i.e. $s_{P_i} = sign_{P_i}(e(S_i)||X)$. $e(S_i)$ and s_{P_i} are sent by each participant P_i to all the other participants in X on a secret channel.

Step 2 Each participant $P_i \in X$ verifies all the signatures it has received, by using the public keys of the senders, and then combines encapsulated t secret shares to form

$$V_X = e(S_1), \ldots, e(S_t).$$

Step 3 Each participant $P_i \in X$ reads T_X from the notice board and reconstructs K as follows:

$$K = g(T_X, f(V_X)).$$

One can easily verify the completeness of the protocol: every authorised subset $X \in \Gamma^*$ can recover K.

2.2.5 Online multiple secret sharing

The protocol satisfies requirement 1 of section 2.2.1 i.e. it is online. Indeed if the key is changed or new participants join the collection of authorised subsets then only the possible values T_X need to be updated (after shares are chosen for the new participants).

In order to share multiple secrets K_h, one can replace f by a family $F = \{f_h : h = 1, 2, \ldots, w\}$ of functions, so that different functions are employed for different secrets. However, as discussed above for the case of the Cachin scheme, this would require V_X to be calculated using a distributed computation subprotocol. An alternative method that enables reuse of the same shares S_i and the same function f, is to ensure that each entry on the notice board has a fresh encapsulation function e_h attached, e.g. $K = g(T_X, f(e_h(S_1, \ldots, S_t)))$ for reconstruction method 1 and $K = g(T_X, f(e_h(S_1), \ldots, e_h(S_t)))$ for reconstruction method 2. Whenever e_h is changed, new values of T_X are placed on the notice board. By the properties of the encapsulating function no information about the shares is revealed in the reconstruction of a secret. Hence the scheme may be used for multiple secret sharing and so the protocol satisfies requirement 2.

2.2.6 How cheating may occur

Note that almost all secret sharing schemes are vulnerable to cheating. For instance, an untrustworthy participant P_i may cheat by submitting a share S_i' different than its own S_i, but carefully computed such that pooling of shares provides other participants with no information about the secret K while allowing P_i to recover K. Whereas the reconstruction calculates a fake secret K' (from the false

share S_i'), participant P_i can use K' and S_i to calculate the correct secret K.

2.2.7 How to detect cheating

We now describe a simple and general method for detecting cheating which applies to any online secret sharing scheme.

In the initialisation phase of the scheme, the dealer D publishes $h(K)$ on the notice board for every secret K that is being shared. Every participant, having reconstructed the secret, say K', can verify its validity by hashing it and comparing the resulting hashed value $h(K')$ with the value on the notice board. If the verification fails, then most probably cheating has occurred in the protocol and thus the computed secret is not correct. Thus the protocol satisfies requirement 3.

This technique detects cheating but does not identify the cheater(s). We now show how to identify *all* the cheaters.

2.2.8 How to identify all cheaters

In the event of cheating having been detected by the method just described, the participants in the authorised set X can appeal to the dealer D to help to discover the identity of the cheaters. Notice that the dealer will only be involved in arbitration *after cheating* has been detected, and will not need to be actively involved in the normal operation of the reconstruction phase of the scheme.

If cheating is detected by the method described above, then every participant sends to the dealer the signed data strings they received during execution of the protocol and the hash of the key.

There are two ways of identifying cheaters, the choice of which depends on the

method used in the reconstruction phase.

Theorem 2.2.1 In an online secret sharing scheme using reconstruction method 1 (described in section 2.2.4), the dealer D can identify all cheaters if a secure digital signature scheme is used.

Proof

Every participant sends to the dealer the signed data strings they received during execution of the protocol. The dealer D calculates $e(S_1), e(S_1, S_2), \ldots, e(S_1, \ldots, S_{t-1})$ in sequence, checking that these are what was submitted by $P_1, P_2, \ldots, P_{t-1}$. As soon as a calculated value $e(S_1, S_2, \ldots, S_i)$ does not equal the submitted value, D knows that P_i cheated. P_i cannot claim to have been framed, since D has P_i's signature on s_{P_i}.

Then D uses the cheater's submission to check P_{i+1}'s submission and so on (i.e. for every i, D verifies that the value signed by P_i used to encapsulate S_{i+1} produces an encapsulation equal to the value signed by P_{i+1}). Thus, D will then be able to identify *all* the parties who sent incorrect values during the protocol. If all submission are correct but the hash of the key is not correct then participant P_t has cheated.

□

Theorem 2.2.2 In an online secret sharing scheme using reconstruction method 2 (described in section 2.2.4), the dealer D can identify all cheaters if a secure digital signature scheme is used.

Proof

Every participant $P_i \in X$ sends to the dealer the data received during execution

of the protocol, signed with their private key. The dealer verifies the signed data received from each P_i, and compares the submitted value of $e(S_i)$ with that computed by using the stored value of the share S_i. If a submitted value is different from the calculated value, then P_i cheated. P_i cannot claim to have been framed, since D has P_i's signature s_{P_i} on the data. Therefore, the dealer will be able to identify *all* the parties who sent incorrect values during the protocol.

□

By theorem 2.2.1 and 2.2.2 the protocol satisfies requirement 4. In the remainder of this chapter we study and analyse several online secret sharing schemes which are examples of the basic general model.

2.3 The Cachin scheme

In 1995, Cachin [13] proposed the first computationally secure secret sharing scheme for general access structures providing the capability to share multiple secrets and to dynamically add or remove participants *online*, without the need to redistribute new shares secretly to the current participants.

The scheme is secure given a secure one-way function in the sense that a non-qualified set of participants running a polynomial time algorithm cannot determine the secret with non-negligible probability. To prevent an attack by exhaustive search, however, the set of possible secrets must not be too small.

2.3.1 Preliminaries

We assume the simple definition of secret sharing scheme (see Section 1.14). Cachin assumed that the secret K is an element of a finite Abelian group G, e.g. the set of ℓ-bit strings under bitwise addition modulo 2.

We will make use of a one-way function $f : G \to G$ such that $f(x)$ is easy to compute for all $x \in G$ (i.e. can be computed in time polynomial in ℓ) and that it is computationally infeasible, for a given $y \in G$, to find an $x \in G$ such that $f(x) = y$.

To achieve reasonable security, the security parameter ℓ and thus the set of possible secrets have to be chosen to be sufficiently large. Today, many apparently secure one-way functions exist with typical l ranging from 64 to 160.

2.3.2 The basic scheme

We first describe Cachin's online scheme for sharing one secret.

Distribution phase

The basic protocol to share a secret $K \in G$ amongst n participants $P_1, P_2, \ldots, P_n$ works as follows:

Step 1 The dealer randomly chooses n elements $S_1, S_2 \ldots, S_n$ from G according to the uniform distribution.

Step 2 For all $i = 1, 2, \ldots, n$ the dealer transmits S_i over a secret channel to P_i.

Step 3 For each minimal qualified subset $X \in \Gamma^*$, the dealer computes

$$T_X = K - f(\textstyle\sum_{x:P_x \in X} S_x)$$

and publishes T_X and $X \in \Gamma^*$ on the bulletin board.

Reconstruction phase

To recover the secret K, a qualified set of participants Y proceeds as follows:

Step 1 The members of Y agree on a minimal qualified subset $X \subseteq Y$.

Step 2 The members of X add their shares together to get $V_X = \sum_{x:P_x \in X} S_x$ and apply the one-way function f to the result V_X.

Step 3 They fetch T_X from the bulletin board and compute $K = T_X + f(V_X)$.

One can easily verify the completeness of the protocol, i.e. every qualified subset $X \in \Gamma^*$ can recover K. This scheme only meets requirement 1 of section 2.2.1.

2.3.3 Sharing multiple secrets

In order to share multiple secrets, Cachin replaced f by a family $F = \{f_m\}$ of one-way functions so that different one-way functions are employed for different secrets.

Distribution phase & Reconstruction phase

The following protocol is used to share w secrets K_m amongst n participants $P_1, P_2, \ldots, P_n$ with different access structures Γ_m for $m = 1, 2, \ldots, w$:

Step 1 The dealer randomly chooses n elements $S_1, S_2, \ldots, S_n$ from G according to the uniform distribution.

Step 2 For all $i = 1, 2, \ldots, n$, the dealer transmits S_i over a secret channel to P_i.

Step 3 For each secret K_m to share, $m = 1, 2, \ldots, w$, and for each minimal qualified subset $X \in \Gamma_m^*$, the dealer computes

$$T_{X,m} = K_m - f_m(\textstyle\sum_{x:P_x \in X} S_x)$$

and publishes $T_{X,m}$ and f_m.

It is important to note that the shares have to be protected from other participants during the reconstruction phase. Otherwise, these participants could subsequently recover other secret shares S_i which they are not allowed to know (without agreement of the other memebers of the authorised subset). Therefore, in the reconstruction phase that the computation of $f_m(V_X)$ must be performed without revealing the set of inputs $\{S_i | P_i \in X\}$. Cachin suggested possible ways of achieving this including the presence of a trusted device to perform the computation or the use of a distributed evaluation protocol such as Goldreich *et al.* [22]. Thus this scheme satisfies requirements 1 and 2 of section 2.2.1.

2.3.4 Comments on Cachin's scheme

Cachin never considers two requirements, namely the detection of cheating and the identification of cheaters. Thus, there are some drawbacks to the Cachin scheme, notably that if a participant or a group of participants decides to cheat then there is no way to identify the cheat (observing that the scheme omits the hashed key $h(K)$ included in the general model of section 2.2).

2.4 Pinch's scheme

First note that Cachin's protocol, as discussed in Section 2.3, does not allow the shares to be reused after the secret has been reconstructed without a further distributed computation subprotocol such as Goldreich *et al.* [22] as part of the reconstruction. Alternatively for access to a predetermined number of secrets in fixed order, a variant of the one time user authentication protocol of Lamport [31] or some other analogous technique can be used.

Pinch's scheme [48] is a modified protocol which overcomes these problems. It is based on the intractability of the Diffie-Hellman problem [18].

2.4.1 Preliminaries

We assume the simple definition of secret sharing scheme (see Section 1.14) as in the scheme of Cachin. In the remainder of this section we work within the ring of integers modulo p, for some prime p. We suppose $p-1$ has a large prime factor q, and we choose an element $g \in \mathbb{Z}_p$ of order q. The primes p and q must be chosen so that determining discrete logarithms to the base g modulo p is computationally infeasible. Most of our calculations involve working within the multiplicative cyclic group of order q generated by g. It is possible to describe the schemes in a more general group-theoretic framework, although we do not consider this here. We also use a one-way function $f : G \to G$.

2.4.2 The basic scheme

Distribution phase

The basic protocol to share $K \in \mathbb{Z}_p$ amongst n participants $P_1, P_2, \ldots, P_n$ works as follows:

Step 1 The dealer D, who knows the secret K, randomly chooses secret shares $S_i < q$ for each participant P_i and transmits S_i over a secure channel to P_i.

Step 2 For each minimal trusted set $X \in \Gamma^*$ the dealer D randomly chooses g_X to be an element of multiplicative order $q \pmod p$, and computes

$$T_X = K - f(g_X^{\prod_{P_x \in X} S_x}) \bmod p$$

and posts the pair (g_X, T_X) on the notice board.

Reconstruction phase

To recover the secret K, a minimal trusted set $X = \{P_1, P_2, \ldots, P_t\}$ of participants comes together and performs the following steps:

Step 1 Participant P_1 reads g_X from the notice board and sends ${g_X}^{S_1} \bmod p$ to P_2.

Step 2 Each subsequent participant P_i, for $1 < i < t$, receives ${g_X}^{S_1 S_2 \ldots S_{i-1}} \bmod p$, raises it to the power S_i and sends the result, which equals ${g_X}^{S_1 S_2 \ldots S_i} \bmod p$, to P_{i+1}.

Step 3 The final participant P_t receives ${g_X}^{S_1 S_2 \cdots S_{t-1}} \bmod p$ and raises this value to the power S_t to form

$$V_X = {g_X}^{S_1 S_2 \cdots S_t} \bmod p = g_X^{\prod_{P_x \in X} S_x} \bmod p.$$

Step 4 On behalf of group X, member P_t reads T_X from the notice board and can now reconstruct K as $K = T_X + f(V_X) \bmod p$.

This scheme meets requirements 1 and 2 of section 2.2.1. If there are multiple secrets K_i to share, then it is possible to use the same shares S_i and one way function f, provided that each entry on the notice board has a fresh value g_X attached.

Pinch also has a variant proposal which avoids the necessity for the first participant P_1 to reveal ${g_X}^{S_1} \bmod p$ at step 1. P_1 takes r modulo q at random and forms ${g_X}^{rS_1} \bmod p$ and passes the result to P_2, who continues as before. At the end of the protocol, P_t returns the computed value ${g_X}^{rS_1S_2\cdots S_t} \bmod p$ to P_1 who computes

$$V_X = ({g_X}^{rS_1S_2\cdots S_t})^{r^{-1}} \bmod p$$

where r^{-1} is the inverse of $r \bmod q$ (the other parts of the protocol are the same as the original protocol).

2.4.3 Security remarks

Pinch claims in his scheme [48], the Diffie-Hellman Problem (DHP) is used for safeguarding the secret shares S_i. In fact if the DHP were a polynomial-time computable function, anyone could obtain the secret shares S_i. Therefore computing S_i from

$$g_X^{S_1S_2\cdots S_t} \bmod p$$

reduces to the DHP. That is, given elements g, g^x and g^y in the ring of integer modulo p it is computationally infeasible to obtain g^{xy}. This implies in particular the intractability of the corresponding Discrete Logarithm Problem (DLP): i.e. given g and g^x in the ring of integer modulo p it is computationally infeasible to recover the exponent x. The DHP can certainly be solved using the discrete logarithm, and Maurer [38] presents some evidence for the equivalence of the two

problems.

2.4.4 Comments on Pinch's scheme

Note that, in Pinch's scheme, if a participant or a group of participants decides to cheat then, just as in Cachin's scheme, there is no way to identify the cheat.

Cheats can be detected and identified by adopting the approach described in Section 2.2.7 and 2.2.8 (as described in detail in Section 2.6).

2.5 A modified version of Pinch's scheme

An important issue in a secret sharing scheme is that the reconstruction must provide the valid secret to all participants from an authorised set. That is, a dishonest participant must not be able to fool the others so that they obtain an invalid secret while the deceiver is able to get the valid secret. This problem has been discussed by several authors [11, 58].

We review the work of Ghodosi *et al.* [21], who pointed out that Pinch's scheme is vulnerable to cheating. They then modified Pinch's scheme in such a way that cheating is prevented. In this section we describe the cheating problem and also the solution given in [21].

2.5.1 A vulnerability in Pinch's scheme

As we have already observed, Pinch's scheme has a major disadvantage in that it is vulnerable to cheating. In this scheme, a dishonest participant $P_i \in X$ may contribute a fake share $S' = \alpha S_i$, where α is a random integer modulo q. Since

every participant of an authorised set $X(|X| = t)$ has access to the final results $g_X^{S_1 \cdots S_i' \cdots S_t}$, the participant P_i can calculate the value

$$(g_X^{S_1 \cdots S_i' \cdots S_t})^{\alpha^{-1}} = g_X^{S_1 \cdots S_i \cdots S_t} = g_X^{\prod_{P_x \in X} S_x} \bmod p = V_X$$

and hence obtain the correct secret, while the other participants calculate an invalid secret.

2.5.2 Ghodosi *et al's* method for detection of cheating

Suppose in the initialisation phase of the Pinch scheme, the dealer publishes $g_X^{V_X}$ corresponding to every authorised set X. Let the reconstruction protocol be the same as in the original Pinch scheme and let V_X' be the final result. Every participant $x \in X$, can verify whether

$$g_X^{V_X} \stackrel{?}{=} g_X^{V_X'}$$

If the verification fails, then cheating has occurred in the protocol and thus the computed secret is not valid. This protocol detects cheating but does not detect the cheat(s) and also cannot prevent cheating. That is, the cheat(s) obtain the secret while the others gain nothing.

2.5.3 Ghodosi *et al's* method for prevention of cheating

Let $C = \sum_{P_x \in X} g_X^{S_x} \bmod p$ correspond to an authorised set X. We assume that in the initialisation phase of the Pinch scheme the dealer also publishes

$$C_X = g_X^C.$$

Ghodosi *et al.* argue that this extra public information gives no useful information about the secret or about participants' shares. They argue that one would have to

solve the discrete logarithm problem and thus easily solve the DHP.

Let X be an authorised set of participants. At the reconstruction phase, every participant $P_i \in X$ computes $g_X^{S_i}$ and broadcasts it to all participants in the set X. Thus, every participant $P_i \in X$ receives $t-1$ values $g_X^{S_j}$ corresponding to all $P_j \in X$, $P_j \neq P_i$. Each participant computes C and verifies

$$C_X \stackrel{?}{=} g_X^C.$$

If the verification fails, then the protocol stops. Suppose the participants agree to perform computation in the cyclic order $P_1, P_2, \ldots, P_t$. If the check is successful, then each participant P_i $(i = 1, 2, \ldots, t)$ knows the true value $g_X^{S_{i-1}}$ of its predecessor (P_t is the predecessor of P_1). So participant P_i $(i = 1, \ldots, t)$ initiates the protocol by computing $(g_X^{S_{i-1}})^{S_i}$ and passing it to P_{i+1}. The protocol proceeds as in the original Pinch scheme and ends at P_{i-2}. In this way, the participant P_{i-1} cannot directly contribute to the computation which started by P_i.

Suppose there exists only one cheat, P_i $(1 \leq i \leq t)$ in the system. If P_i cheats, the computation initiated by P_{i+1} must be correct (the correctness can be verified as $g_X^{V_X} \stackrel{?}{=} g_X^{V'_X}$, where V'_X is the result obtained by P_{i-1}). That is, although cheating has occurred, the honest set of participants can recover the secret.

2.5.4 Comments on Ghodosi *et al's* version

As we have already observed, Pinch never considered two of the requirements listed in section 2.2.1. One is to detect cheating and the other is to identify cheaters. However, Ghodosi *et al's* version of Pinch's scheme detects cheating and recovers the secret if a majority of participants of the authorised set are honest.

However, if there exists a group of two or more collaborating cheats, then this

protocol (see Section 2.5.3) is not able to identify the cheaters or enable the honest participants to recover the secret. We will discuss how to fix this problem in the next section.

2.6 How to identify all cheaters in Pinch's scheme

We propose an enhanced modified version of Pinch's secret sharing protocol which has advantages over the original scheme, and its modification by Ghodosi *et al.*, in that it detects cheating and enables the identification of *all* cheaters by an arbitrator, regardless of their number. This modified scheme is based on the general model specified in Section 2.2.

2.6.1 How to detect cheating

As already detailed, Ghodosi *et al.* [21] describe a method for detecting cheating in the above protocols. Suppose in the initialisation phase of the scheme, the dealer D sends ${g_X}^{V_X} \bmod p$ to every authorised set X. Let the reconstruction protocol be the same as in the above scheme and let V'_X be the computed result. Every participant $x \in X$ can verify that

$$ {g_X}^{V'_X} \equiv {g_X}^{V_X} \pmod{p}. $$

If the verification fails, then cheating has occurred in the protocol and thus the computed secret is not correct.

However, this method should be carefully implemented to prevent attacks which exploit the arithmetic of exponents. Since we choose our generator g_X to have

order q, we know that ${g_X}^{V'_X} \equiv {g_X}^{V_X} \pmod{p}$ if and only if $V'_X \equiv V_X \pmod{q}$. Hence, if a malicious participant could arrange for everyone to accept $V'_X = V_X + rq$ for some non-zero integer r, then cheating will not be detected.

For this reason, we propose an alternative way of detecting cheating. Suppose in the initialisation phase of the scheme, the dealer D publishes $h(K_i)$ on the notice board for every secret K_i that is being shared (where h is a one-way collision-resistant hash-function). Every participant, having reconstructed the secret (K'_i, say), can verify its validity by hashing it and comparing the resulting hash-code $h(K'_i)$ with the value on the notice board. If the verification fails, then cheating has occurred in the protocol and thus the computed secret is not correct.

Note that the second method requires less storage space on the notice board than the first method. In the first method D stores ${g_X}^{V_X} \bmod p$ on the notice board, and hence needs to store $|\Gamma|$ values for every secret. In the second method, D stores $h(K_i)$ on the notice board, and hence D only needs to store one hash-code for every secret. Thus, the second method is a more efficient way of detecting cheating.

2.6.2 An enhanced protocol which identifies all cheaters

We now describe an enhanced version of the protocol, which will enable the identification (by the dealer) of all cheaters. As a pre-requisite to using the scheme, every participant must have an implementation of an agreed digital signature scheme, and must have selected a key pair for this signature scheme which is described in the basic model in section 2.2.8.

In order to identify the cheaters in Pinch's scheme, one must adopt the approach given in section 2.2.8. In particular the modified protocol will operate exactly

as described in section 2.4.2, with the exception of the following modifications. In distribution phase Steps 1 and 2 of the protocol, participant P_i, as well as forwarding ${g_X}^{S_1 S_2 \cdots S_i} \bmod p$, also forwards a signature on a data string, signed using his or her private signature key. More specifically, if $s_{P_i}(Y)$ denotes the digital signature on data Y computed using the private signature key of P_i, then P_i computes and forwards the signature

$$s_{P_i}({g_X}^{S_1 S_2 \cdots S_i} \bmod p || X || g_X)$$

to the next participant P_{i+1} (where $||$ denotes concatenation of data items). Also, when participant P_i receives ${g_X}^{S_1 S_2 \ldots S_{i-1}} \bmod p$ and the signed string containing ${g_X}^{S_1 S_2 \ldots S_{i-1}} \bmod p$, P_i checks the signature before proceeding with the protocol.

If cheating is detected by the method described in the second scheme in section 2.6.1, then every participant sends to the dealer the signed data strings they received during execution of the protocol. The dealer D calculates $g_X^{S_1}, g_X^{S_1 S_2}, \ldots, g_X^{S_1 \cdots S_t}$ in sequence, checking that what D gets is what was submitted by $P_1, P_2, \ldots, P_t$. As soon as a calculated value $g_X^{S_1 S_2 \cdots S_i}$ does not equal the submitted value, D knows that P_i cheated. P_i cannot claim to have been framed, since D has P_i's signature on $s_{P_i}({g_X}^{S_1 S_2 \cdots S_i} \bmod p || X || g_X)$.

Then D uses the cheater's submission to check P_{i+1}'s submission and so on (i.e. for every i, D verifies that the value signed by P_i raised to the power $S_{i+1} \bmod p$ is equal to the value signed by P_{i+1}). Thus, D will then be able to identify *all* the parties who sent incorrect values during the protocol. If all signatures verify correctly but the key is not verified by its hash then P_t has cheated.

This use of signatures will also protect a minority of the members of an authorised set against a majority colluding to falsely accuse the minority of cheating.

2.7 An online secret sharing scheme which identifies all cheaters

The main motivation of our proposed scheme is as follows:

Ghodosi *et al.* [21] pointed out that Pinch's scheme is also vulnerable to cheating. They presented a modified version of Pinch's protocol which detects and prevents cheating, under the assumption that a majority of the participants of the authorised reconstruction set are honest. However, this scheme does not protect a minority of participants of the authorised set from a colluding majority, who falsely accuse the minority of cheating. Thus, we proposed the modified Pinch's scheme in the previous section to fix this problem.

Moreover, all previously described schemes rely on a "last participant" to reconstruct the secret which could be a disadvantage. That is, the last participant for secret reconstruction always has the opportunity to cheat and obtain the true secret, while the others obtain a false one.

We design a computationally secure online secret sharing scheme which is based on the intractability of the RSA problem by using our basic model. Compared to Pinch's scheme, and its modification by Ghodosi *et al.*, our scheme has the following advantages: it detects cheating and it enables the identification of *all* cheaters by an arbitrator, regardless of their number. The scheme does not rely on a "last participant" who reconstructs the secret on behalf of a minimal trusted set: the responsibility is diffused among all participants by adopting reconstruction method 2 in section 2.2.4. Thus the proposed scheme is a good example of an online secret sharing scheme which is based on the model of Section 2.2 which enables the detection of cheating and the identification of all cheaters.

2.7.1 Preliminaries

We assume the simple definition of secret sharing scheme (see Section 1.14). Let $N = pq$ be the product of two large primes p and q, and let e $(1 < e < \phi(N))$ be chosen so that $(e, \phi(N)) = 1$, where $\phi(N) = (p-1)(q-1)$. The values N and e are public, and the values p, q and $\phi(N)$ are secret. Throughout this scheme we work within the multiplicative group of integers modulo N, and we shall assume that factoring N is infeasible [50]. In other words, we assume that the RSA problem is difficult to solve in polynomial time.

In the secret sharing schemes we will describe below we shall make use of a one-way hash-function h which is collision-resistant, as described in the basic model in section 2.2.7. As a pre-requisite to using the scheme, every participant must have an implementation of an agreed digital signature scheme, and must have selected a key pair for this signature scheme, as described in the basic model in section 2.2.8.

2.7.2 A secret sharing protocol

We now present a new secret sharing protocol in which the participants of an authorised set compute the secret K by combining their secret shares in encrypted form. In this way the participants will not reveal their secret shares during the process of recovering K. The protocol uses a publicly accessible location, e.g. a notice board, where the dealer can store non-forgeable information accessible to all participants. This location will, at least, indicate the number of participants n and the access structure Γ.

Distribution phase

The basic protocol to share the secret K is as follows:

First the dealer D selects N and e, and randomly chooses secret shares $S_i < N$, $i = 1, 2, \ldots, n$. Then D transmits to each P_i over a secure channel the share S_i, and securely stores S_i for subsequent use to identify cheaters, if cheating is detected. For each minimal authorised set $X \in \Gamma^*$ the dealer D uses e and N to compute

$$T_X = K \oplus h(\prod_{x:P_x \in X} S_x^e \bmod N),$$

where $\oplus$ denotes exclusive-or of bit-strings. The dealer D posts the following items on the notice board: the four-tuple (X, e, N, T_X) for every $X \in \Gamma^*$, and the value $h(K)$.

Reconstruction phase

A minimal authorised set $X \in \Gamma^*$ of participants can compute K by performing the following steps:

Step 1 Each participant $P_i \in X$ reads $h(K)$ and the values e, N, T_X from the four-tuple corresponding to the appropriate set X on the notice board. Then P_i computes $S_i^e \bmod N$ and signs the data $(S_i^e \bmod N, X, e, N)$ using his/her private signature key to form $s_{P_i} = sign_{P_i}(S_i^e \bmod N||X||e||N)$, where $||$ denotes concatenation of data items. Finally, $S_i^e \bmod N$ and s_{P_i} are sent by each participant P_i to all the other participants in X on a secret channel.

Step 2 Each participant $P_i \in X$ verifies all the signatures it has received, by using the public keys of the senders, and then computes

$$V_X = \prod_{x:P_x \in X} S_x^e \bmod N.$$

Step 3 Each participant $P_i \in X$ reads T_X from the notice board and reconstructs K as follows:

$$K = T_X \oplus h(V_X).$$

One can easily verify the completeness of the protocol: every authorised subset $X \in \Gamma^*$ will recover K.

New participants can be added online by updating new values of T_X on the notice board. Similarly the key K can be changed by updating new values of T_X for all authorised subset X. So the scheme satisfies requirement 1 of section 2.2.1

2.7.3 A multiple secret sharing protocol

If there are multiple secrets K_m (with $m = 1, 2, \ldots, w$) to share, it is now possible to use the same one-way hash-function h and the set of secrets $S_1, S_2, \ldots, S_n$, provided that each entry on the notice board has a fresh value of e_m, N_m and $T_{X,m}$ attached.

In the following protocol, the dealer D initially chooses secret shares S_i for each participant P_i. The dealer D distributes S_i over a secure channel to P_i. Whenever a new secret is to be shared, the protocol below is repeated, without any need to distribute new shares to the participants. Note also that, whenever a new secret K_m is to be shared, the access structure may be different to that used for previous secrets, and hence we denote the access structure for secret K_m by Γ_m.

Distribution phase

The multiple protocol to share w secrets K_m amongst n participants $P_1, P_2, \ldots, P_n$ work as follows: The dealer generates a new pair (N_m, e_m), to be used in sharing the secret K_m. For each minimal trusted set $X \in \Gamma_m^*$ the dealer D uses his encryption key e_m and computes

$$T_{X,m} = K_m \oplus h(\textstyle\prod_{x:P_x \in X} S_x^{e_m} \bmod N_m)$$

and posts the four-tuples $(X, e_m, N_m, T_{X,m})$ on the notice board.

Reconstruction phase

To recover the secret K_m, a minimal authorised set $X = \{P_1, P_2 \ldots, P_t\}$ of participants comes together and performs the following steps:

Step 1 Each participant $P_i \in X$ reads $h(K)$ and the values $e_m, N_m, T_{X,m}$ from the four-tuple corresponding to the appropriate set X on the notice board. Then P_i computes $S_i^{e_m} \bmod N_m$ and signs the data $(S_i^{e_m} \bmod N_m, X, e_m, N_m)$ using his/her private signature key to form $s_{P_i} = sign_{P_i}(S_i^{e_m} \bmod N_m||X||e_m||N_m)$, where $||$ denotes concatenation of data items. Finally, $S_i^{e_m} \bmod N_m$ and s_{P_i} are sent by each participant P_i to all the other participants in X on a secret channel.

Step 2 Each participant $P_i \in X$ verifies all the signatures it has received, by using the public keys of the senders, and then computes

$$V_X = \prod_{x:P_x \in X} S_x^{e_m} \bmod N_m.$$

Step 3 Each participant $P_i \in X$ reads $T_{X,m}$ from the notice board and reconstructs K_m as follows:

$$K_m = T_{X,m} \oplus h(V_X).$$

Thus one can easily verify the completeness of the protocol : every qualified subset $X \in \Gamma_m^*$ can recover K_m.

The shares S_x are encapsulated as $S_x^{e_m} \bmod N_m$. Since the RSA problem is difficult no users obtain useful information about S_x from the values $S_x^{e_m} \bmod N_m$. Hence the shares can be reused for multiple secrets. Thus the scheme satisfies requirement 2 of section 2.2.1.

2.7.4 How cheating may occur

In both the proposed protocol and its generalisation to multiple secrets it is possible for one of the participants to cheat the others in such a way that the cheater will get the correct secret but the other participants do not.

Suppose that participant P_j contributes a fake encrypted share S' instead of $S_j^e \bmod N$. Then every participant of the authorised set X will compute V_X incorrectly as $V_X' = S' \cdot \prod_{x \neq j : P_x \in X} S_x^e \bmod N$ instead of $V_X = \prod_{x : P_x \in X} S_x^e \bmod N$. However P_j, who knows $S_j^e \bmod N$, can calculate the correct secret V_X.

2.7.5 How to detect cheating

In the initialisation phase of the scheme, the dealer D publishes $h(K_m)$ on the notice board for every secret K_m that is being shared. Every participant, having reconstructed the secret, say K_m', can verify its validity by hashing it and comparing the resulting hashed value $h(K_m')$ with the value on the notice board. If the verification fails, then most probably cheating has occurred in the protocol and thus the computed secret is not correct. Thus the scheme satisfies requirement 3 of section 2.2.1. This test detects cheating but does not identify the cheater(s). We now show how to identify *all* the cheaters.

2.7.6 How to identify all cheaters

In the event of cheating having been detected by the method just described, the participants in the authorised set X can appeal to the dealer D to help to discover the identity of the cheaters. Notice that the dealer will only be involved in arbitration *after cheating* has been detected, and will not need to be actively involved

in the normal operation of the reconstruction phase of the scheme.

In order to identify *all* cheaters, every participant $P_i \in X$ sends to the dealer the data received during execution of the protocol, together with the signatures on it. The dealer verifies the signed data received from each P_i, and compares the submitted value of $S_i^e \bmod N$, with that computed by using the stored value of the share S_i. If a submitted value is different from the calculated value, then P_i cheated. P_i cannot claim to have been framed, since D has P_i's signature s_{P_i} on $(S_i^e \bmod N||X||e||N)$. Therefore, the dealer will be able to identify *all* the parties who sent incorrect values during the protocol. Thus the scheme satisfies requirement 4 of section 2.2.1.

This use of signatures will also protect a minority of participants of an authorised set from a colluding majority who falsely accuses the minority of cheating.

2.7.7 Security remarks

In our proposed scheme, RSA encryption is used for safeguarding the secret shares S_i. In fact if RSA were a polynomial-time computable function, anyone could obtain the secret shares S_i. Therefore computing S_i from $S_i^e \bmod p$ reduces to computing the RSA problem (see Section 1.5.1). In other words, the security against safeguarding the secret shares S_i depends on the difficulty of breaking RSA.

Attacks resulting from repeated use of RSA key values

In the description of the protocol we stated that new values N and e must be used for every new secret to be shared. If a value is used more than once then problems

can arise, as we now describe.

Suppose the same value of N is used twice. Then an interceptor might obtain

$$S_i^{e_1} \bmod N \text{ and } S_i^{e_2} \bmod N$$

for $e_1 \neq e_2$ and some i. Then, if $(e_1, e_2) = 1$, the interceptor could find values u and v such that

$$ue_1 - ve_2 = 1,$$

and hence the interceptor could compute

$$S_i = (S_i^{e_1})^u \cdot (S_i^{e_2})^{-v} \bmod N.$$

If the same (small) value of e was used more than e times, then an interceptor might obtain

$$S_i^e \bmod N_j \;\; (j = 1, 2, \ldots, t;\; t \geq e)$$

Using the Chinese Remainder Theorem (see Theorem 1.4.10) these values can be combined to obtain

$$S_i^e \bmod N_1 N_2 \cdots N_t.$$

If $t \geq e$ (as we assumed) then this will simply equal S_i^e, and it will then be simple to deduce S_i.

Thus, the properties of well-chosen pairs (e_m, N_m) and the function h, ensure that the reuse of the set of secret shares $S_1, S_2, \ldots, S_n$ does not leak any information which may be useful to cheaters and/or other malicious users.

2.8 Conclusion

We have given a model for online secret sharing. We have studied online secret sharing schemes which are examples of the basic model, such as those of Cachin

[13] and Pinch [48], and have outlined their shortcomings. We have suggested an enhanced version of Pinch's scheme [59] which can be used in such a way that cheating by participants can be detected, in which case the participants in an authorised set X can request help from the dealer D, who can always uniquely identify the cheaters.

In addition, we have presented a scheme (also described in [60]) which allows the reconstruction of an arbitrary number of secrets and provides the capability to dynamically add or remove participants online, without having to redistribute new shares secretly, by storing additional authentic (but not secret) information on a notice board.

Moreover, our scheme can be used in such a way that cheating by participants will be detected, in which case the honest participants can also request help from the dealer D, who will always be able to uniquely identify the cheaters by adopting the model of section 2.2.

Furthermore, compared to previous schemes, our scheme does not rely on a *last participant* who reconstructs the secret on behalf of a minimal trusted set of participants: we have diffused responsibility among all participants.

Chapter 3

Cryptanalysis of Digital Signatures

3.1 Introduction

A cryptographic technique which is fundamental to authentication, authorisation, and non-repudiation is the *digital signature*. The purpose of a digital signature is to provide a means for an entity to bind its identity to a piece of information. The process of *signing* entails transforming the message and some secret information held by the entity into a tag called a *signature*.

Section 1.11 provides terminology used throughout the chapter, and describes digital signatures that permits a useful classification of the various schemes. Section 1.11.2 provides a description of the RSA signature which is based on the intractability of the factorisation problem. Section 1.11.3 describes cryptographic techniques based on the intractability of the discrete logarithm problem, such as ElGamal signature, DSS, and the ElGamal signature with message recovery.

The goal of an adversary is to forge signatures; that is, produce signatures which will be accepted as those of some other entity. The following provides a set of criteria for what it means to break a signature scheme (see Section 11.2.4 of [39]).

(i) Total break. An adversary is either able to compute the private key information of the signer, or find an efficient signing algorithm functionally equivalent to the valid signing algorithm.

(ii) Selective forgery. An adversary is able to create a valid signature for a particular message or class of messages chosen a priori. Creating the signature does not directly involve the legitimate signer.

(iii) Existential forgery. An adversary is able to forge a signature for at least one message. The adversary has little or no control over the message whose signature is obtained, and the legitimate signer may be involved in the deception.

In section 3.2, we study possible attacks on RSA signatures such as integer factorisation, existential forgery and using the multiplicative property of RSA. Similarly, in section 3.3 we study possible attacks on ElGamal type signatures, including selective forgery, existential forgery and making use of the homomorphic property of ElGamal.

In section 3.4, we describe Shao's digital signature scheme [54], which is based on the difficulty of computing discrete logarithms. Shao claims that his scheme can resist both homomorphism and substitution attacks. Contrary to Shao's claim, we show in section 3.4.2 that Shao's scheme is vulnerable to a homomorphism attack just like all ElGamal type signature schemes.

In section 3.5 we show that there are major differences between a digital signature with message recovery, and an authenticated encryption scheme. We point out that the discrete logarithm based signature with message recovery scheme proposed by Chen in [14] is actually not a signature scheme. It would more accurately be described as an authenticated encryption scheme.

3.2 Possible attacks on RSA signatures

The security of RSA signatures is based on the intractability of the integer factorisation problem (see Section 1.5). RSA can be used as the basis of digital signatures with and without message recovery (see Section 1.11.2).

Three possible attacks on the RSA signature scheme are as follows (for further details see Section 11.3.2 of [39]):

(i) Attack 1 (Integer factorisation)

If an adversary is able to factor the public modulus n of some entity A, then the adversary can compute $\phi(n)$ and then, using the extended Euclidean algorithm (see Theorem 1.3.27), deduce the private key d from $\phi(n)$ and public exponent e by solving $ed \equiv 1 \pmod{\phi(n)}$. This constitutes a total break of the system. To guard against this, one must select p and q so that factoring n is a computationally infeasible task. For further information, see section 1.5.

(ii) Attack 2 (Existential forgery)

The basic idea behind RSA signatures is to compute $s = M^d \pmod{n}$ where M is (some function of) the message. This means that an adversary can choose an arbitrary s^* and compute $m^* = (s^*)^e \pmod{n}$ and claim s^*

is a valid signature on m^*.

This is one reason why RSA signatures are always either of the form

(a) $s = (h(m))^d \pmod{n}$, where h is a one-way collision resistance hash-function, giving a signature with appendix, or

(b) $s = (R(m))^d \pmod{n}$, where R is a redundancy-adding function, giving a signature with message recovery for a message m of limited length.

(iii) Attack 3 (Multiplicative property of RSA)

The RSA signature scheme (as well as the encryption scheme) has the following multiplicative property (see Section 1.11.2), sometimes referred to as the **homomorphic property**. If $s_1 = m_1^d \bmod n$ and $s_2 = m_2^d \bmod n$ are signatures on messages m_1 and m_2, respectively (or, more properly, on messages with redundancy added), then $s = s_1 s_2 \bmod n$ has the property that $s = (m_1 m_2)^d \bmod n$. If $m = m_1 m_2$ has the proper redundancy, then s will be valid signature for it. Hence, it is important that the redundancy function R is not multiplicative, i.e., $R(m_1\ m_2) \neq R(m_1)\ R(m_2)$. Alternatively this homomorphism weakness of RSA can be eliminated by applying some one-way hash-function h to m before signing m, as long as h is not multiplicative.

3.3 Possible attacks on ElGamal type schemes

The digital signature scheme proposed by ElGamal in 1984 [19] (see Section 1.11.3) is of considerable practical importance. The security of the ElGamal scheme is based on the difficulty of computing discrete logarithms. There are, however, some technical problems concerning its security; for example, each time a message is to be signed, a new session key k is needed and $r = g^k \bmod p$ is publicised as one

component of the signature. If k is ever divulged (e.g. because it was not chosen at random) then the secret key is compromised.

Briefly, the ElGamal signature scheme works as follows. Let p be a large prime number such that $p-1$ has a large prime factor, and let g be a primitive element modulo p. Any user A has a private key x $(1 < x < p-1)$ and a public key $y = g^x \bmod p$. To sign a message m, user A does the following:

(i) Randomly chooses an integer k, $1 < k < p-1$.

(ii) Computes

$$r = g^k \bmod p$$

(iii) Computes

$$s = (m - xr)k^{-1} \bmod (p-1)$$

(iv) Sends $\mathrm{Sig}(m) = (r, s)$ as the signature.

Possible attacks on the ElGamal signature scheme are as follows.

(i) Attack 1 (Total break)

An adversary might attempt to forge A's signature on m by selecting a random integer k and computing $r = g^k \bmod p$. The adversary must then determine $s = (m - xr)k^{-1} \bmod p-1$. If the discrete logarithm problem is computationally infeasible, the adversary cannot do better than to choose an s at random; the success probability is only p^{-1}, which is negligible for large p.

(ii) Attack 2

A different k must be selected for each message signed; otherwise, the private

key can be determined with high probability as follows. Suppose $s_1 = (m_1 - xr)k^{-1} \bmod p-1$ and $s_2 = (m_2 - xr)k^{-1} \bmod p-1$. Then $(s_1 - s_2)k \equiv (m_1 - m_2) \pmod{p-1}$. If $s_1 - s_2 \not\equiv 0 \pmod{p-1}$, then $k = (s_1 - s_2)^{-1}(m_1 - m_2) \bmod p-1$. Once k is known, x is easily found.

(iii) Attack 3 (Existential forgery)

The signing equation is $s = (m - xr)k^{-1} \bmod (p-1)$. It is then easy for an adversary to mount an existential forgery attack as follows. Select any pair of integers (u, v) with $\gcd(v, p-1) = 1$. Compute $r = g^u y^v \bmod p = g^{u+xv} \bmod p$ and $s = -rv^{-1} \bmod p-1$. The pair (r, s) is a valid signature for the message $m = su \bmod p-1$, since $(g^m g^{-xr})^{s^{-1}} = g^u y^v = r$. This attack is one reason that, in practice, the signing equation is normally specified as $s = (h(m) - xr)k^{-1} \pmod{p-1}$ where h is a one-way, collision resistant, hash-function.

(iv) Attack 4 (Homomorphism attack)

He and Kiesler [25] describe the following 'homomorphism attack'. Suppose that, in computing three distinct signatures, the random values k used satisfy $k_3 = k_1 + k_2$. An observer of these signatures will be able to deduce this by noting that $r_3 = r_1 r_2$. This will then immediately yield the private key from

$$x = (m_1 s_2 s_3 + m_2 s_1 s_3 - m_3 s_1 s_2) \times (r_1 s_2 s_3 + r_2 s_1 s_3 - r_1 r_2 s_1 s_2)^{-1}.$$

Note that the biggest threat to ElGamal type schemes comes from the progress being made in the computation of the discrete logarithm. Steady improvements have been occurring in the computation of discrete logarithms. We can cope with this threat by using larger keys.

3.4 Shao's modification to ElGamal signatures

3.4.1 Scheme description

Shao [54] proposed a new digital signature scheme which does not use a one-way hash-function, the security of which is based on the difficulties of computing discrete logarithms, and performance of which is similar to those of DSS and RSA.

The purpose of Shao's scheme is to prevent ElGamal attack 3 without requiring use of a hash-function, and to prevent ElGamal attack 4 by removing the 'homomorphic' property.

We now briefly describe the scheme of Shao. The system parameters known to all users are a large prime modulus p, a prime divisor q of $p-1$, and an integer g with order q, that is, $g^q \equiv 1 \pmod{p}$, and $1 < g < p$. Any user A has two secret keys x_1, x_2 $(1 < x_1, x_2 < q)$, and two public keys:

$$y_1 = g^{x_1} \bmod p, \quad y_2 = g^{x_2} \bmod p.$$

The digital signature of a message m $(1 < m < p)$ is the triple (r, s_1, s_2) such that

$$r = ((y_1^{s_1} + m y_2^{s_2}) g^{r+m} \bmod p) \bmod q$$

To sign a message m, user A does the following:

(i) Randomly chooses two integers k_1 and k_2, $1 < k_1, k_2 < q$.

(ii) Computes

$$r^* = g^{k_1} + m g^{k_2} \bmod p$$

$$r = r^* \bmod q$$

(iii) Computes

$$s_1 = (k_1 - r - m)x_1^{-1} \bmod q \tag{3.1}$$

$$s_2 = (k_2 - r - m)x_2^{-1} \bmod q \tag{3.2}$$

(iv) Sends $\mathrm{Sig}(m) = (r, s_1, s_2)$ as the signature.

From equations 3.1 and 3.2

$$x_1 s_1 + r + m \equiv k_1 \pmod q$$

$$x_2 s_2 + r + m \equiv k_2 \pmod q$$

so

$$y_1^{s_1} g^{r+m} \equiv g^{k_1} \pmod p$$

$$m y_2^{s_2} g^{r+m} \equiv m g^{k_2} \pmod p$$

Taking sums

$$(y_1^{s_1} + m y_2^{s_2}) g^{r+m} \equiv r^* \pmod p$$

so

$$((y_1^{s_1} + m y_2^{s_2}) g^{r+m} \bmod p) \equiv r \pmod q$$

Therefore, if the signer follows the above protocol, the recipient always accepts the signature.

Shao claims that the partial signature $r = g^k \bmod p$ of the ElGamal type signature scheme is replaced by $r^* = g^{k_1} + m g^{k_2} \bmod p$ in his scheme, and hence the isomorphism relation between the multiplicative cyclic group g and the additive cyclic group $\mathbb{Z}_p$ no longer exists. Hence the scheme is not subject to the homomorphism attacks. However, in the following section we show that this is not the case.

3.4.2 The attack

The signer cannot prevent the attackers from computing $g^{k_1}, g^{k_2} \pmod{p}$ from the corresponding published message m and its signature (r, s_1, s_2), since

$$g^{k_i} \equiv g^{x_i s_i + r + m} \equiv y_i^{s_i} g^{r+m} \pmod{p},\ i = 1, 2.$$

Suppose three pairs of session keys (k_1, k_2), (k_1', k_2'), (k_1'', k_2'') were used to generate the signatures (r, s_1, s_2), (r', s_1', s_2'), (r'', s_1'', s_2'') of the messages m, m', m'' respectively as in equations (1.1) and (1.2). If $k_1 = k_1' + k_1''$, then this relation can be recognised by the attackers, as

$$g^{k_1} \equiv g^{k_1'} g^{k_1''} \pmod{p}.$$

Now, we have three linear equations in x_1, k_1, k_1', k_1'' from equation (3.1):

$$x_1 s_1' \equiv k_1' - r' - m' \pmod{q},$$

$$x_1 s_1'' \equiv k_1'' - r'' - m'' \pmod{q},$$

and

$$x_1 s_1 \equiv k_1 - r - m \pmod{q}.$$

From these equations, and knowing that $k_1 = k_1' + k_1''$, one can easily obtain the first part of private key

$$x_1 = \{(r' + r'' - r) + (m' + m'' - m)\} \times (s_1 - s_1' - s_1'')^{-1} \bmod q.$$

Similarly, if $k_2 = k_2' + k_2''$, then this relation can also be recognised by the attackers, as

$$g^{k_2} \equiv g^{k_2'} g^{k_2''} \pmod{p}.$$

Now, we have three linear equations in x_2, k_2, k_2', k_2'' from equation (3.2):

$$x_2 s_2' \equiv k_2' - r' - m' \pmod{q},$$

$$x_2 s_2'' \equiv k_2'' - r'' - m'' \pmod{q},$$

and

$$x_2 s_2 \equiv k_2 - r - m \pmod{q}.$$

From these equations, and knowing that $k_2 = k_2' + k_2''$, one can also easily obtain the second part of private key

$$x_2 = \{(r' + r'' - r) + (m' + m'' - m)\} \times (s_2 - s_2' - s_2'')^{-1} \bmod q.$$

Contrary to Shao's claim, we have shown that Shao's scheme is vulnerable to homomorphism attacks just like all ElGamal type signature schemes.

The main justification given in [54] for the use of Shao's scheme is its resistance to homomorphism and substitution attacks. Substitution attacks can be avoided by the use of a one-way hash-function, and thus there no longer appears to be any reason to use Shao's scheme.

Although the ElGamal scheme and its variants (e.g. DSS) are subject to homomorphism attacks, such an attack being successful appears to be no more likely than finding a discrete logarithm, as long as the random integer used to construct the signature is chosen at random.

3.5 Authenticated encryption schemes

In many applications it is necessary to provide both confidentiality and integrity/origin protection for a transmitted message. This can be achieved using a

combination of encryption and a digital signature. However, this doubles the cost of protection, and motivates the work of Horster, Michels and Petersen [28] who introduced an authenticated encryption scheme, designed to provide a combination of services at reduced cost.

Subsequently Lee and Chang [33] modified the HMP scheme to remove the need for a one-way function, whilst keeping communication costs the same. This scheme may be advantageous in environments where implementing a one-way function is difficult, e.g. in a smart card with limited memory and/or computational capability.

More recently, Chen [14] introduced a variant of the Lee and Chang scheme, which is claimed to provide the same security level with a simpler specification. However, some of the claims made by Chen are incorrect as we point out in section 3.5.2.

3.5.1 Chen's scheme

Chen's scheme [14] is based on the discrete logarithm problem and is claimed to combine the same efficiency as Horster-Michels-Petersen and Lee-Chang [28, 33], with a simpler specification.

We now briefly describe the scheme of Chen. Let p and q be large primes with $q|(p-1)$ and g an element in $\mathbb{Z}_p^*$ of order q. The signer has a secret key $x_A \in \mathbb{Z}_p^*$ and public key $y_A \equiv g^{x_A} \pmod{p}$. To generate a signature for message m, the signer A does the following:

(i) picks a random number $k \in \mathbb{Z}_q^*$,

(ii) computes $e \equiv y_B^k \pmod{p}$ and $e' = \frac{e}{\gcd(e,p-1)}$,

(iii) computes $r \equiv m^{e'} \pmod{p}$,

(iv) solves s from $k \equiv (r+s)x_A \pmod{q}$.

Signer A then sends (r, s) as the signature to receiver B. After receiving (r, s), B does the following:

(i) computes $e \equiv y_B^k \equiv y_B^{(r+s)x_A} \equiv y_A^{(r+s)x_B} \pmod{p}$,

(ii) solves d from $e'd \equiv 1 \pmod{q}$, where $e' = \frac{e}{\gcd(e,p-1)}$,

(iii) recovers the message $m \equiv r^d \pmod{p}$.

Note that the public key y_B is also an element of order q in $\mathbb{Z}_p^*$, so all values e' are different. If p and q are cryptographic safe primes and are chosen carefully, then the probability that an e is computed from two different e' is negligible, especially for $p = 2q + 1$.

To avoid a forgery attack on the new signature scheme, the message m is required to contain redundancy (see Section 1.11 page 39 or Section 3.2, Attack 2).

3.5.2 Comments on Chen's scheme

In a signature with message recovery scheme, see for example (See Section 1.11.3 or [46]), the Trusted Third Party (TTP) can always verify the signatures which are sent by the receiver B without B having to divulge any long term secret information to the TTP. However, in the authenticated encryption schemes described in [28, 33], only the sender A and the receiver B can verify a protected message sent from A to B. This is because B can only verify such a message with the aid of his private decryption key. It is for this reason that the authors of both [28] and [33]

have been careful to call their schemes authenticated encryption schemes rather than combined signature/encryption schemes, although nowhere is this point made explicit in [28] or [33].

In order to verify a signature generated using the scheme proposed by Chen, the receiver B needs to use his private key. It seems that the only feasible way a third party can verify the 'signature' (r, s) is that the shared secret $y_A^{x_B} = y_B^{x_A}$ is divulged. It is not possible however to tell whether A or B produced the signature, but with this information the third party can read all the messages sent between A and B. This holds even if B is prepared to supply the recovered plaintext message m and both A's and B's public keys, in addition to the received signature.

This is an unacceptable property for a true signature scheme, where one would normally expect signature verification to be possible without compromise of any private keys. Thus it would be more appropriate to refer to the scheme as an authenticated encryption scheme, analogously to the terminology used in [28, 33]. Finally note that similar remarks have been made in [47] regarding schemes recently proposed by Zheng.

3.6 Conclusion

We have described several general types of attack against RSA and ElGamal signatures. For RSA signatures the homomorphism property could only be used by a forger to forge a signature, but in the ElGamal scheme that property could be used to recover the signer's secret.

We then considered the Shao signature scheme and showed that it is subject to homomorphism attacks, contrary to the claim of Shao. This essentially means that

Shao's scheme has no practical value.

We pointed out that the discrete logarithm based signature with message recovery scheme proposed by Chen in [14] is actually not a signature scheme. It would more accurately be described as an authenticated encryption scheme.

Chapter 4

Key Transport

4.1 Introduction

Key transport techniques can be divided into two classes: those using symmetric encryption, and those using asymmetric encryption.

This chapter considers key transport techniques based on asymmetric, or public-key, encryption. Such methods involve one party choosing a symmetric key and transferring it to a second, using that party's encryption public key. This provides key authentication to the originator (only the intended recipient has the private key allowing decryption), but the originator itself obtains neither entity authentication nor key confirmation. The second party receives no source authentication. Authentication assurances can be provided with digital signatures or by other methods such as entity authentication via public-key decryption (see Section 4.2). In the latter case, the intended recipient may authenticate itself by returning some time-variant value which it alone may produce or recover. This allows authentication of both the entity and a transferred key.

By using public-key encryption alone, assurances of mutual entity authentication and mutual key authentication may be obtained through additional messages, as illustrated in the Helsinki protocol presented in section 4.2.

Previous work, as described in Section 4.3, has shown that Helsinki protocol is vulnerable to attack.

We provide a simple modification to the protocol which prevents all possible attacks in section 4.4.

4.2 Helsinki protocol

4.2.1 Introduction

The Helsinki protocol is actually a derivation of a protocol originally described by Needham and Schroeder in 1978 [43]. It also embodies features from the COMSET protocol, which was devised as part of the RIPE project, [9]. For further information see Section 12.5.1 and 12.10 of [39].

This protocol is designed to establish a shared secret key between two entities A and B, and is specified as Key Transport Mechanism 6 in Clause 7.6 of ISO/IEC 11770-3, [5]. This mechanism securely transfers in three passes two secret keys, one from A to B and one from B to A. In addition, both entities are authenticated and obtain key confirmation about their respective keys. This mechanism has the following two requirements:

(i) Each entity X has an asymmetric encipherment system (E_X, D_X).

(ii) Each entity has access to an authenticated copy of the public encipherment

transformation of the other entity.

4.2.2 Specification

The mechanism operates in a series of stages labelled $A1, B1, A2.1, A2.2$ and $B2$, as described below.

Message contraction $(A1)$

A has obtained a key K_A and wants to transfer it securely to B. A selects a random number r_A and constructs a key data block consisting of its distinguishing name A, the key K_A, the number r_A and an optional data field $Text1$. Then A enciphers the key block using B's public encipherment transformation E_B, thereby producing the enciphered data block

$$BE_1 = E_B(A||K_A||r_A||Text1)$$

A constructs the message M_1, consisting of enciphered data block and some optional data field $Text2$. A sends M_1 to B:

$$M_1 : \ A \rightarrow B : \ BE_1||Text2$$

Message contraction $(B1)$

B extracts the enciphered key block BE_1 from the received message M_1 and deciphers it using its private decipherment transformation D_B. Then B verifies the sender identity A included in the enciphered data block.

B has obtained a key K_B and wants to transfer it securely to A. B selects a random number r_B and constructs a key data block consisting of the key K_B, the random number r_B, the random number r_A (as extracted from the deciphered block) and an optional data field $Text3$. Then B enciphers the key block using A's public

encipherment transformation E_A, thereby producing the enciphered data block

$$BE_2 = E_A(K_B||r_A||r_B||Text3)$$

Then B constructs the message M_2, consisting of the enciphered data block BE_2 and an optional data field $Text4$. B sends M_2 to A.

$$M_2 : \ B \rightarrow A : \ BE_2||Text4$$

Key and Entity confirmation ($A2.1$)

A extracts the enciphered key block BE_2 from the received message M_2 and deciphers it using its private decipherment transformation D_A. Then A checks the validity of the message through comparison of the random number r_A with the random number r_A contained in the enciphered block BE_2. If the verification is successful, A has authenticated B and at the same time obtained confirmation that K_A has safely reached entity B.

Message response ($A2.2$)

A extracts the random number r_B from the deciphered key block and constructs the message M_3, consisting of the random number r_B and an optional data field $Text5$. A sends M_3 to B.

$$M_3 : \ A \rightarrow B : \ r_B||Text5$$

Key and Entity confirmation ($B2$)

B verifies that the response r_B extracted from M_3 is consistent with the random number r_B sent in enciphered form in M_2. If the verification is successful, B has authenticated A and at the same time has obtained confirmation that K_B has safely reached entity B.

Note that this key transport mechanism has following properties:

1. Number of passes: 3

2. Entity authentication: This mechanism provides mutual entity authentication.

3. Key confirmation: This mechanism provides mutual key confirmation.

4. Key control: A can choose the key K_A, since it is the originating entity. Similarly, B can choose the key K_B. Mutual key control can be achieved by each entity by combining the two keys K_A and K_B on both sides to form a shared secret key K_{AB}. However, the combination function must be one-way, otherwise B can choose the shared secret key. This mechanism could then be classified as a key agreement mechanism.

5. Key usage: This mechanism uses asymmetric techniques to mutually transfer two secret keys: K_A from to A to B and K_B from B to A. The following cryptographic technique separation may be derived from the mechanism: A uses its key K_A to encipher messages for B and to verify authentication codes from B. B in turn uses the received key K_A to decipher message from A and generate authentication codes for A. The cryptographic functions of K_B may be separated in an analogous manner. In such a way, the asymmetric basic of the key transport mechanism may be extended to the usage of the secret keys.

6. Origin: This mechanism is derived from the three pass protocol known as COMSET.

7. Background: This mechanism is based on zero-knowledge techniques. From the execution of the mechanism neither of the entities learns anything that it could not have computed itself.

4.2.3 Discussion

We consider the four requirements we listed for an authentication protocol (see Section 1.12):

First of all, we consider the four requirements for B:

1. Can B be sure that M_1 and M_3 were generated by A? B can check that M_3 was generated by A because it contains r_B which only A has access to from M_2. B can also check that M_1 was generated by A because B included r_A in M_2 that A replied to, and A must recognise, so 'Yes'.

2. Can B be sure that M_1 and M_3 are fresh? Given that r_B is unpredictable then B can check the freshness of M_3, and hence is convinced of the freshness of M_1. This is because M_3 was in response to a message containing both r_A from M_1 and r_B, and these could only be read by A, so 'Yes'.

3. Can B be sure that M_1 and M_3 were intended for it? M_1 is intended for B because it is encrypted with B's public key. Also B can reason that A would not have sent M_3 unless it was A who sent M_1, and hence we have 'Yes'.

4. Can B be sure that M_3 was reply to M_2? 'Yes' because of the inclusion of r_B.

We next consider A's requirements:

1. Can A be sure that M_2 was generated by B? Because M_2 contains r_A which only B can get from M_1. But B could give it to anyone so 'No'.

2. Can A be sure that M_2 is fresh? 'Yes' because of inclusion of r_A (r_A must be unpredictable).

3. Can A be sure that M_2 is intended for it? M_2 is intended for A because it is encrypted with A's public key, so 'Yes'.

4. Can A be sure that M_2 is reply to M_1? 'Yes' because of the inclusion of r_A.

Thus, we would suggest that the Helsinki protocol is not secure against an insider attack. In the following section we describe an attack proposed by Horng and Hsu.

4.3 The Horng-Hsu attack and an observation

Horng and Hsu [27] observed that the Helsinki protocol is subject to attack. The attack operates as follows. C commences the attack by causing A to inaugurate a run of the protocol with C. A then sends the following message:

$$M_1 : A \rightarrow C : E_C(A||K_A||r_A)$$

C decrypts the message to obtain r_A, and uses it to create a forged message M_1' containing a new key component K_A', which C sends to B. When sending this message, C pretends that it is from A.

$$M_1' : C \rightarrow B : E_B(A||K_A'||r_A)$$

B responds to C (thinking it is responding to A) with the following message:

$$M_2' : B \rightarrow C : E_A(K_B||r_A||r_B)$$

C intercepts this message and forwards it (unchanged) to A. A responds to C with following message:

$$M_3 : A \rightarrow C : r_B$$

C then forwards this message to B.

After these exchanges:

(i) A believes it has established a shared secret key with C, based on the key components K_A and K_B (although C does not know K_B), and

(ii) B believes it has established a shared secret key with A, based on the key components K'_A and K_B (although A does not know K'_A).

Note that this is an example of an 'Insider attack'. This holds since, in order to launch the attack, C must persuade A to inaugurate a run of the protocol, and hence C must be an entity with whom A is prepared to establish a shared secret key.

Note also that this attack is possible since, whereas B actually generates M_2, A will believe it comes from C. This is possible because message M_2 contains no indication of its source (unlike message M_1). Hence although C cannot discover the precise contents of message M_2, C can forward it to A and have it accepted as originating from C, although it was actually generated by B.

4.4 A revised version of the protocol

4.4.1 Description of modification

Based on the observation we have just made about why the attack is possible, we propose that the protocol should be modified in the following minimal way. The

second message M_2 should be replaced by a modified message, which we call N_2:

$$N_2 : B \rightarrow A : E_A(B||K_B||r_A||r_B)$$

That is, the only change is to insert an identifier for B in the second protocol message. The other two protocol messages remain unchanged. We now discuss this modification to the Helsinki protocol.

4.4.2 Discussion

We have discussed the basic protocol in section 4.2.1, where we come to the conclusion that the Helsinki protocol is vulnerable for an insider attack. However, we show that a revised version of the protocol overcomes this attack. Let us consider the four requirements we listed for an authentication protocol (see Section 1.12).

First of all, we consider the four requirements for B:

1. Can B be sure that M_1 and M_3 were generated by A? B can check that M_3 was generated by A because it contains r_B which only A has access to from N_2. A has no reason to respond M_3 unless A sent M_1 and wishes to acknowledge receipt of N_2. B can also check that M_1 was generated by A because B included r_A in N_2 that A replied to, and A must recognise r_A, so 'Yes'.

2. Can B be sure that M_1 and M_3 are fresh? Given that r_B is unpredictable then B can check the freshness of M_3, and hence is convinced of the freshness of M_1. This is because M_3 was in response to a message containing both r_A from M_1 and r_B, and these could only be read by A, so 'Yes'.

3. Can B be sure that M_1 and M_3 were intended for it? M_1 is intended for B because it is encrypted with B's public key. Also B can reason that A would not have sent M_3 unless it was A who sent M_1, and hence we have 'Yes'.

4. Can B be sure that M_3 was reply to N_2? 'Yes' because of the inclusion of r_B.

We next consider A's requirements:

1. Can A be sure that N_2 was generated by B? 'Yes' because it contains B's name and r_A which only B can get from M_1. If N_2 was generated by a third party then B has revealed r_A to that third party and that third party has not followed the protocol (by including B's name in a message it has generated). Thus B and the third party are colluding participants.

2. Can A be sure that N_2 is fresh? 'Yes' because of inclusion of r_A (r_A must be unpredictable).

3. Can A be sure that N_2 is intended for it? N_2 is intended for A because it is encrypted with A's public key, so 'Yes'.

4. Can A be sure that N_2 is reply to M_1? 'Yes' because of the inclusion of r_A.

Therefore, the revised version of the protocol is capable of providing all of the specified properties which are described in section 1.12.

In short, B accepts A as valid and sends message N_2. When A receives N_2, it checks on the data string for the freshness and the inclusion of name B to prevent the insider attack which is described in section 4.3.

4.4.3 Related Work

Lowe [34] pointed out an attack upon the Needham-Schroeder authentication protocol [43]. The attack allows an intruder to impersonate another agent. Also the revised protocol proposed by Lowe prevents all other attacks.

The Horng-Hsu attack is closely related to the Lowe attack on the Needham-Schroeder protocol. Moreover, the modification we propose in the revised Helsinki protocol corresponds directly to the modifications Lowe proposes to the Needham-Schroeder protocol.

4.5 Conclusion

We have described a key transport technique called the Helsinki protocol. We have also described an attack on this protocol due to Horng and Hsu.

We have described a simple modification to the Helsinki protocol which prevents the Horng-Hsu attack [27], but yet which does not add significantly to the communications or computational overhead for the protocol. Note that both the original and amended protocols depend very much on an implicit property of the public key encryption scheme. Specifically, the protocols require the encryption scheme to provide a measure of integrity protection for encrypted strings.

Chapter 5

Conclusion

In this chapter we provide a summary of the work presented in this thesis and present some conclusions and further areas for work.

In Chapter 1 we gave an introduction to the abstract algebra, number theory and cryptography necessary for the remainder of the thesis. We followed the history behind some of the issues leading to the development of the design, analysis and application of cryptographic techniques.

Chapter 2 presented the first part of the novel contribution of this thesis. We proposed a general model for online secret sharing which makes it possible to detect cheating and identify all cheaters. All the schemes considered in the remainder of this chapter such as those of Cachin [13] and Pinch [48] fit the general model. We identified shortcomings of existing schemes, and suggested an enhanced version of Pinch's scheme [59] which solved cheating problems by requesting help from the dealer, who can always uniquely identify the cheaters. We then introduced a new online secret sharing scheme [60] which can be used in such a way that cheating by participants will be detected, in which case the honest participants can also

request help from the dealer, who will always be able to uniquely identify the cheaters. Moreover, compared to previous schemes the new protocol does not rely on a 'last participant' who reconstructs the secret on behalf of a minimal trusted set of participants: we have diffused responsibility among all participants.

Chapter 3 introduced possible attacks on ElGamal type signature schemes. In section 3.4, we described Shao's digital signature scheme [54] which is based on the difficulties of computing discrete logarithms. Contrary to Shao's claim, we have shown that Shao's scheme is vulnerable to homomorphism attacks just like all ElGamal type signature schemes. Moreover, substitution attacks can be avoided by the use of a one-way hash-function, and therefore, there no longer appears to be any reason to use Shao's scheme. In section 3.5 we studied authenticated encryption schemes and identified the differences between an authenticated encryption scheme and a digital signature with message recovery. We observed that the discrete logarithm based signature with message recovery scheme proposed by Chen [15] is actually not a signature scheme. It would more accurately be described as an authenticated encryption scheme.

In Chapter 4 we examined the Helsinki key transport protocol based on public-key encryption. We identified its shortcoming by considering four requirements (see Section 1.12) for an authentication protocol. In section 4.4 we described a simple modification to the Helsinki protocol which prevents the Horng-Hsu attack, but yet which does not add significantly to the communications or computational overhead for the protocol.

In general, one of the better ways to design a cryptographic technique is to embed several computationally hard problems within the cryptographic technique itself. However care must be taken to ensure that the strength of the scheme does not

rest solely on the hard problems involved, but also on their interaction, otherwise there may still be the possibility of attacks, as has been demonstrated in several cryptographic techniques (see Section 3.4 and 4.2).

Bibliography

[1] "The digital signature standard proposed by NIST". *Communications of the ACM*, 35(7):36–40, 1992.

[2] ISO 7498-2: 1989. "Open systems Interconnection - Basic reference Model - Part 2: Security Architecture". *International Organization for Standardization*, 1989.

[3] ISO/IEC 9796: 1991. "Digital signature scheme giving message recovery". *International Organization for Standardization*, 1991.

[4] ISO/IEC 11770-2: 1996. "Key management - Part 2: Mechanisms using symmetric techniques". *International Organization for Standardization*, 1996.

[5] ISO/IEC 11770-3: 1999. "Key management - Part 3: Mechanisms using asymmetric techniques". *International Organization for Standardization*, 1999.

[6] ISO/IEC 9798. "Entity authentication - Parts 1 to 5". *International Organization for Standardization*.

[7] G.R. Blakley. "Safeguarding cryptographic keys". In *Proceedings of AFIPS National Computer Conference*, pages 313–317, 1979.

[8] R. Blom. "An optimal class of symmetric key generation schemes". In *Advances in Cryptology – Proceedings of EUROCRYPT '84*, pages 335–338. Springer-Verlag, 1985.

[9] A. Bosselaers and B. Preneel. "Integrity Primitives for Secure Information Systems: Final Report of RACE Integrity Primitives Evaluation RIPE-RACE 1042". In *Number 1007 in Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[10] E. Brickell, D. Stinson, and S. Goldwasser. "The detection of cheaters in threshold schemes". In S. Goldwasser, editor, *Advances in Cryptology – Proceedings of CRYPTO '88*, pages 564–577. Springer-Verlag, 1988.

[11] E. Brickell, D. Stinson, and S. Goldwasser. "The detection of cheaters in threshold schemes". In *Advances in Cryptology – Proceedings of CRYTO '88*, pages 564–577. Springer-Verlag, 1990.

[12] David M. Burton. *"Elementary number theory"*. Wm. C. Brown Publishers, 1994.

[13] C. Cachin. "On-line secret sharing". In C. Boyd, editor, *Proceedings of the 5th IMA Conference on Cryptography and Coding*, pages 190–198. Springer-Verlag, 1995.

[14] K. Chen. "Signature with message recovery". *Electronics Letters*, 34(20):1934, 1998.

[15] L. Chen, D. Gollmann, C.J. Mitchell, and P. Wild. "Secret sharing with reusable polynomials". In Vijay Varadharajan, Josef Pieprzyk, and Yi Mu, editors, *Proceedings of ACISP '97*, pages 183–193. Springer-Verlag, 1997.

[16] P. Cohn. *"Algebra, volume 1"*. Wiley, 1982.

[17] D. Denning. "Digital signatures with RSA and other public-key cryptosystems". *Communications of the ACM*, 27(4):388–392, 1984.

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

[19] T. ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[20] A. Fiat and A. Shamir. "How to prove yourself: practical solutions to identification and signature problems". In *Advances in Cryptology – Proceedings of CRYPTO '86*, pages 186–194. Springer-Verlag, 1987.

[21] H. Ghodosi, J. Pieprzyk, G.R. Chaudhry, and J. Seberry. "How to prevent cheating in Pinch's scheme". *Electronics Letters*, 33(17):1453–1454, 1997.

[22] O. Goldreich, S. Micali, and A. Widgerson. "How to play any mental game or a completeness theorem for protocols with honest majority". In *Proceedings of 19th ACM Symposium on the Theory of Computing*, pages 218–229, 1987.

[23] L.C. Guillou and J.J. Quisquater. "A practical zero knowledge protocol fitted to security micoprocessor minimising both transmission and memory". In *Advances in Cryptology – Proceedings of EUROCRYPT '88*, pages 123–128. Springer-Verlag, 1988.

[24] L. Harn. "Digital signature with (t, n) shared verification based on discrete logarithms". *Electronics Letters*, 29(24):2094–2095, 1993.

[25] J. He and T. Kiesler. "Enhancing the security of ElGamal's signature scheme". *IEE Proc. Digit. Tech.*, 141(4):249–252, 1994.

[26] I. Herstein. *"Topics in Algebra"*. Wiley, 1987.

[27] G. Horng and C.K. Hsu. "Weakness in the Helsinki protocol". *Electronics Letters*, 34(4):354–355, 1998.

[28] P. Horster, M. Michels, and H. Petersen. "Authenticated encryption schemes with low communication costs". *Electronics Letters*, 30(15):1212–1213, 1994.

[29] C.L. Hsu and T.C. Wu. "Authenticated encryption scheme with (t, n) shared verification". *IEE Proc. Digit. Tech.*, 145(2):117–120, 1998.

[30] N. Koblitz. *"Algebraic aspects of cryptography"*. Springer-Verlag, 1998.

[31] L. Lamport. "Password authentication with insecure communication". *Communication in ACM*, 34:770–772, 1981.

[32] M.D. Larsen. *"Introduction to modern algebraic concepts"*. Addison-Wesley Publishing Company, 1969.

[33] W. Lee and C. Chang. "Authenticated encryption scheme without using a one-way function". *Electronics Letters*, 31(19):1656–1657, 1995.

[34] G. Lowe. "An attack on the Needham-Schroeder public-key authentication protocol". *Information Processing Letters*, 56:131–133, 1995.

[35] K. Manders and L. Adleman. "NP-complete decision problems for binary quadratics". *Journal of Computer and System Sciences*, 16:168–184, 1978.

[36] J.L. Massey and J.K. Omura. "A new multiplicative algorithm over finite fields and its applicability in public key cryptography". *Presented at EUROCRYPT '83 Udine, Italy*, 1983.

[37] J.L. Massey and J.K. Omura. "Method and apparatus for maintaining the privacy of digital messages conveyed by public transmission". *U.S Patent No: 4,567,600 28 Jan*, 1986.

[38] U.M. Maurer. "Towards the equivalence of breaking the Diffie Hellman protocol and discrete logarithms". In Y.G. Desmedt, editor, *Advances in Cryptology – Proceedings of CRYPTO '94*, pages 271–281. Springer-Verlag, 1994.

[39] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *"Handbook of Applied Cryptography"*. CRC Press, 1996.

[40] C.J. Mitchell and C.Y. Yeun. "Fixing a problem in the Helsinki protocol". *ACM Operating Systems Review*, 32(4):21–24, 1998.

[41] C.J. Mitchell and C.Y. Yeun. "Comment–Signature scheme with message recovery". *Electronics Letters*, 35(3):217, 1999.

[42] T. Nagell. *"Introduction to number theory"*. Chelsea Publishing Company, 1981.

[43] R.M. Needham and M.D. Schroeder. "Using encryption for authentication in large networks of computers". *Communications of the ACM*, 21:993–999, 1978.

[44] I. Niven, H. Zuckerman, and H. Montgomery. *"An Introduction to the theory of numbers"*. Wiley, 1991.

[45] K. Nyberg. "New digital signature scheme based on discrete logarithm". *Electronics Letters*, 30(6):481, 1994.

[46] K. Nyberg and R.A. Rueppel. "Message recovery for signature schemes based on the discrete logarithm problem". In *Advances in Cryptology – Proceedings of EUROCRYPT '94*, pages 175–190. Springer-Verlag, 1995.

[47] H. Petersen and M. Michels. "Crytanalysis and improvement of signcryption schemes". *IEE Proceedings on Computers and Digital Techniques*, 145:149–151, 1998.

[48] R.G.E. Pinch. "Online multiple secret sharing". *Electronics Letters*, 32(12):1087–1088, 1996.

[49] S. Pohlig and M.E. Hellman. "An improved alogrithm for computing logarithms over $GF(P)$ and its cryptographic significance". *IEEE Transactions on Information Theory*, 24:106–110, 1978.

[50] R.L. Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signatures and public key cryptosystems". *Communications of the ACM*, 21:120–126, 1978.

[51] K. H. Rosen. *"Elementary number theory and its applications"*. Wiley, 1993.

[52] C.P. Schnorr. "Efficient signature generation by smart cards". *Journal of Cryptology*, 4(3):161–174, 1991.

[53] A. Shamir. "How to share a secret". *Communications of the ACM*, 22:612–613, 1979.

[54] Z. Shao. "Signature scheme based on discrete logarithm without using one-way hash-function". *Electronics Letters*, 34(11):1079–1080, 1998.

[55] Z. Shao. "Signature schemes based on factoring and discrete logarithms". *IEE Proc. Digit. Tech.*, 145(1):33–36, 1998.

[56] G.J. Simmons. *"Contemporary Cryptography: The Science of Information Integrity"*. IEEE Press, 1992.

[57] D.R. Stinson. *"Cryptography theory and practice"*. CRC Press, 1995.

[58] M. Tompa and H. Woll. "How to share a secret with cheaters". *Journal of Cryptology*, 1(2):133–138, 1988.

[59] C.Y. Yeun and C.J. Mitchell. "How to identify all cheaters in Pinch's scheme". In *Proceedings of JWIS'98, Singapore*, pages 129–133, 1998.

[60] C.Y. Yeun, C.J. Mitchell, and M.Burmester. "An online scret sharing scheme which identifies all cheaters". In *Proceedings of NORDSEC'98, Trondheim, Norway, November 1998, and presented at the 1st IMA Conference on Mathematics in Communication, Loughborough, UK, December 1998.*

[61] C.Y. Yeun, C.J. Mitchell, and S.L. Ng. "Comment–Signature scheme based on discrete logarithm without using one-way hash-function". *Electronics Letters*, 34(24):2329–2330, 1998.

[62] Y. Zheng. "Digital signcryption or how to achieve cost(signature+encryption) << cost(signature)+cost(encryption)". In *Advances in Cryptology – Proceedings of Crypto '97*, pages 165–179. Springer-Verlag, 1997.