Abstract

Digital commercial transactions are so important to us these days that authentication just cannot be overlooked or ignored in a secure cryptographic system. This thesis gives a survey of a few modern authentication schemes. It pays special attention to Undeniable Signature Schemes and their applications. It is believed that Undeniable Signatures have a great potential to be applied to the real world of practical electronic commercial transactions because they protect the interests of both signer and verifier. A form of Undeniable Signatures is implemented in the process of understanding the scheme and the mechanism of the technique.

Résumé

Les transactions commerciales électroniques nous sont si importantes aujourd'hui, que leur authenticité ne peut tout simplement pas être négligée ou ignorée dans un système cryptographique sure. Cette thèse donne un apperçu util de quelques techniques d'authentification courantes. Elle prête particulièrement attention aux signatures indéniables (Undeniable Signatures) et ses applications. Il se dit de la signature indéniable qu'elle a un très grand potentiel pour être appliquée dans le monde des transactions commerciales électroniques, parce qu'elle garde les intérêts des deux partis d'une transaction, soit la personne qui applique la signature, et celle qui la vérifie. Une forme de signatures indéniables est implantée.

Acknowledgments

It was quite a trip to have this thesis completed. With this opportunity, I would like to thank my thesis supervisors Prof. David Avis (at McGill University) and Prof. Claude Crépeau (at Université de Montréal at the time) for their patience, guidance and advice. Mr. Jacques Desmarais for translating the abstract. Ms. Hisako Mori for doing the final proof reading. My family for supporting me in every way that can be done. Ms. Elsa Ho and Ms. Mary Imamura for believing me and loving me always. Mr. Luc Boulianne for giving me an opportunity to work at SOCS. All system staff members at SOCS for their support and picking up the extra work and stress. All members of Arashi Daiko (Terry, May, Masa, Mikio, Miwako, Christian, Ignacio, Jean-Francios, Hisako, Satoko, Hibiki, Patrick, Phil and Zoe) in Montreal for their friendship and inspiration through the Japanese drum. Mr. Bob Turnbull and Mr. Anthony Teoli (Concordia University - Continuing Education) for changing my perspective of the world through camera lenses.

Now that this is done, I think it is time to turn a new page. So, here I go and I thank you all.

In the memorial of the tragic and sudden death of
Diana, Princess of Wales. 31 August, 1997. Paris, France.
and
Mother Teresa. 5 September, 1997. Calcutta, India.

Contents

Chapter 1

Introduction

Cryptography is the science of encoding messages in such a way that only the intended receiver can decode the message. Cryptography used to concern only the military and diplomats, but has recently become a public issue which has received a great deal of attention in the past 20 years. Cryptographic protocols are designed to discourage eavesdroppers from intervening during transmissions on public channels.

In recent years, the Internet has grown to be a major network for making business transactions and for exchanging information. It links tens of thousands of machines and millions of users around the world forming a "cyber community". Commercial uses of the Internet, namely web page advertising, on-line shopping and web page catalog sales activities, are likely to increase greatly. When the profit from commercial use of the Internet increases, the number of Internet Service Providers (ISP) will likely increase, which will lead to a substantial price drop for cyberspace access. This is likely to encourage more people to join this growing community. Regulations and guidelines must be introduced to ensure the safety and privacy of each Internet user, especially when commercial activities become a major activity on the net. Cryptographic protocols are used to maintain personal or company privacy, while authentication protocols are used to ensure the integrity of information and users. For example, it would be desirable for business partners to ensure that an offered digital contract came from a reliable source.

Digital signature schemes offer tools so that the sender and the receiver of the message can be convinced that the other person is who he claims to be. The first digital signature scheme was defined by Diffie-Hellman [DH76] back in 1976. Since then, it has evolved into many different forms, and has inspired several different ideas and perspectives. This thesis provides a survey of a few modern authentication schemes and their applications in the business world. It is the intention of this thesis to survey digital signature schemes and to focus on the undeniable signature scheme in particular.

The undeniable signature scheme was introduced by D. Chaum in 1989 [CvA89]. The "undeniable" notion here means that the signer of the signature cannot deny his own signature. In addition, he cannot falsely claim a signature that he did not issue. These are the properties of this signature technique that we believe will have good potential for benefiting electronic business transactions over the Internet.

Digital signatures are one of the many forms of cryptographic applications that involve number theory. Some important concepts from number theory, cryptography, discrete logarithms, zero-knowledge proofs and secret sharing are included in Chapter 2 to help the reader to understand the basis of digital signatures. All of the information listed in Chapter 2 may not be necessary for this thesis, but it is essential if the reader decides to consult the original articles listed in the bibliography. Chapter 3 gives an introduction to digital signatures and a description of different kinds of digital signature schemes. Chapter 4 gives an overview of undeniable signatures and their applications and possible attacks. For a better understanding of the undeniable signature scheme, we have implemented Pedersen's distributed undeniable signature scheme [Ped91]. A detailed description of the implementation is in Chapter 5. Chapter 6 concludes this thesis.

Chapter 2

Background Information

This chapter provides an overview of some important Number Theory concepts that are useful in order to understand this thesis. However, this chapter assumes that the reader has some basic concepts of abstract algebra. Information introduced here might not be referred to directly or entirely in this thesis. However, it is referred to directly by the original articles. Readers who wish to obtain additional background information should refer to the sources listed in the bibliography.

An **equivalence relation** on a set A is a binary relation $\sim$ on A such that for any $a, b, c \in A$, the following properties are true:

1. $a \sim a$. [**Reflexive**]
2. $a \sim b \rightarrow b \sim a$. [**Symmetric**]
3. $a \sim b$ and $b \sim c \rightarrow a \sim c$. [**Transitive**]

Let $\sim$ be an equivalence relation on a set A. The **equivalence class** of $a \in A$ is defined to be

$$\{x | x \in A \text{ and } x \sim a\}.$$

2.1 Integers

Let Z be the set of all integers $\{\cdots, -1, 0, 1, 2, \cdots\}$. We say the integer b is divisible by integer a where $a \neq 0$ if there exists $x \in Z$ such that $b = ax$ and we write $a \mid b$. We write $a \not| \, b$, otherwise.

We say a is a greatest common divisor (gcd) of (b, c) if $a \mid b$, $a \mid c$ and a is the greatest among all common divisors of b and c.

Theorem 2.1 (Division Algorithm) *For $a, b \in Z$, $a \neq 0$, there exist unique integers q and r such that*

$$b = aq + r, \qquad 0 \leq r < |a|.$$

Proof: The proof is found in [Gol73], page 24 or [PW66], page 39-40.

The gcd of two numbers can be calculated by Euclidean Algorithm ([Sti95], page 116). We say a and b are **relatively prime** to each other if $gcd(a, b) = 1$.

Theorem 2.2 *If g is the gcd of b and c then there exist integers x_0 and y_0 such that*

$$g = gcd(b, c) = bx_0 + cy_0.$$

Proof: See [NZM91], page 7.

Theorem 2.3 (Euclid) *Let p be a prime number and $a, b \in Z$. If $p|ab$ then either $p|a$ or $p|b$.*

Proof: Please refer to [PW66] page 45 for proof.

If an integer $m \mid (a - b)$ where $a, b \in Z$, then we say a is **congruent** to b modulo m and write $a \equiv b(mod\ m)$. The number of elements in Z that are prime to m is defined by **Euler's ϕ-function,**

$$\phi(m) = \#\{k \in Z | 1 \leq k \leq m, gcd(k, m) = 1\}$$

Theorem 2.4 *If $gcd(a, m) = 1$ then there is a solution x such that $ax \equiv 1(mod\ m)$.*

Proof: By Theorem 2.2 and the fact that the gcd of two prime numbers is 1. See [PW66], page 56 for the complete proof.

Corollary 2.1 *Let a, b, and p be integers. If p is prime and $p \not| a$, then $ax \equiv b(mod\ p)$ always has a unique solution modulo p.*

Proof: The proof can be derived from Theorem 2.4.

Since $gcd(a, m) = 1$ and $ax \equiv 1(mod\ m)$, x is called the inverse of a and is written as a^{-1}. Therefore, $aa^{-1} \equiv 1(mod\ m)$.

Theorem 2.5 (Chinese Remainder Theorem) *Let $m_1, \cdots, m_r$ denote r positive integers that are pairwise relatively prime, i.e. $gcd(m_i, m_j) = 1$ where $i \neq j$. Let $a_1, \cdots, a_r \in Z$. Then, the following r congruences*

$$x \equiv a_i\ (mod\ m_i) \qquad \text{where } (1 \leq i \leq r),$$

has a unique solution modulo $M = m_1 \times m_2 \times \cdots \times m_r$ given by

$$x = \sum_{i=1}^{r} a_i M_i n_i (mod\ M), \tag{2.1}$$

where $M_i = M/m_i$ and $M_i n_i \equiv 1(mod\ m_i)$.

Proof: Proof can be found in [PW66], page 57 or [NZM91], page 64.

2.2 Groups

A nonempty set G with a binary relation $\circ$ is called a group $\langle G, \circ \rangle$ if the following properties are satisfied:

1. If $a, b, c \in G$ then $(a \circ b) \circ c = a \circ (b \circ c)$. **[Associative]**
2. There exists an element $e \in G$ such that $e \circ a = a \circ e = a$ for every element a of G. The element e is called the identity of G and is unique if it exists. **[Existence of an Identity]**
3. For $a \in G$ there exists an element a^{-1} of G such that $a \circ a^{-1} = a^{-1} \circ a = e$. **[Existence of Inverse]**

If $a, b \in \langle G, \circ \rangle$ and $a \circ b = b \circ a$ then $\langle G, \circ \rangle$ is called an **abelian group**. $\langle G, + \rangle$ and $\langle G, \times \rangle$ are called an **additive group** and **multiplicative group** respectively. In $\langle G, + \rangle$ the identity element is 0, for $\langle G, \times \rangle$ the identity element is 1. A **finite group** is a group that contains a finite number of elements.

If G is a finite multiplicative group, the **order of the group** is the number of elements in G and it is denoted as $|G|$. The **order of an element** $x \in G$ is the smallest integer m such that $x^m = 1$ and it divides $|G|$. Subgroups of G are subsets of G which are themselves groups with respect to the operation defined in G.

A finite nonempty subset H is a **subgroup** of G if

1. H is a nonempty set.
2. H is closed under products and inverse. Meaning, if $x, y \in H$, then $x^{-1} \in H$ and $xy \in H$.

Both H and G share the same identity element and inverse ([DF91], page 45).

If $a \in G$ is of order m then

$$H = \{a^k | k \in Z\}$$

is a subgroup of G of order m. The multiplicative group G is said to be **cyclic** if it has an element $a \in G$ such that for any $b \in G$ there is some integer j with $b = a^j$. Such an element a is called **a generator** of the cyclic group and we write $G = \langle a \rangle$.

Example: The additive group of integers Z_p modulo p. This group is generated by element 1 of Z_p.

Theorem 2.6 .

1. *In a finite cyclic group $\langle a \rangle$ of order m, the element a^k is a generator of a subgroup of order $m/gcd(k, m)$ where k is an integer.*
2. *Let f be a positive divisor of the order of a finite cyclic group $\langle a \rangle$. Then $\langle a \rangle$ contains $\phi(f)$ elements of order f.*

3. *A finite cyclic group $\langle a\rangle$ of order m contains $\phi(m)$ generators - that is, elements $\langle a^r\rangle = \langle a\rangle$. Then generators are the powers $\langle a^r\rangle$ with gcd(r,m) = 1.*

Proof: For the proof of both theorems please see [LN83], page 7.

Let G and H be groups. A map $\psi : G \to H$ is called a group **homomorphism** if $\psi(\alpha\beta) = \psi(\alpha)\psi(\beta)$. A bijective homomorphism is an **isomorphism**. If $\psi : G \to H$ is an isomorphism, we say that G and H are **isomorphic** and write $G \simeq H$.

Corollary 2.2 (Euler's Theorem) *Let $a, p \in Z$ and $gcd(a,p) = 1$ then*

$$a^{\phi(p)} \equiv 1(mod\ p).$$

Proof: [Rom95], page 8.

Corollary 2.3 (Fermat's Theorem) *If a is any integer and p is a prime then $a^p \equiv a(mod\ p)$.*

Proof: If $a \equiv 0(mod\ p)$, then $a^p \equiv 0 \equiv a(mod\ p)$. If $a \not\equiv 0(mod\ p)$, then $gcd(a,p) = 1$. We know $\phi(p) = p-1$. Therefore by Euler's Theorem, $a^{p-1} \equiv 1(mod\ p)$ then $a^p \equiv a(mod\ p)$.

2.3 Rings

A nonempty set R with two binary operations $+$ and $\times$ is called a ring $\langle R, +, \times\rangle$ if

1. $\langle R, +\rangle$ is an abelian group.
2. $\langle R, \times\rangle$ is associative : $(a \times b) \times c \equiv a \times (b \times c)$ for all $a, b, c \in R$.
3. For all $a, b, c \in R$. $(a+b) \times c = (a \times c) + (b \times c)$ and $a \times (b + c) = (a \times b) + (a \times c)$.

Let $\langle R, +, \times\rangle$ be a ring, it is called a **commutative ring** if $a \times b = b \times a$ for all $a, b \in R$. $\langle R, +, \times\rangle$ is called a **ring with identity** if there exists an element $e \in R$ such that

$$ae = ea = a \text{ for all } a \in R$$

The element e is usually denoted by 1.

The set S of all real numbers of the form $x + y\sqrt{2}$ where $x, y \in Z$ with addition and multiplication defined in the usual way is an example of a ring. S is closed under these operations and S is a **commutative ring with identity**.

A **zero divisor** in a commutative ring R is a nonzero element $\alpha, \beta \in R$ such that $\alpha\beta = 0$ for some $\beta \neq 0$. A commutative ring with identity is called an **integral domain** if R does not contain a zero divisor.

Let R be a ring and let $r \in R$. For any positive integer n, we define

$$nr = \underbrace{r + r + \cdots + r}_{n\ terms}$$

The **characteristic** $char(R)$ of a ring R is the smallest positive integer n for which $n1 = 0$, where 1 is the identity element of R (or equivalently $nr = 0$), should such an integer exist. Otherwise, we say that R has the characteristic 0.

2.4 Fields

Let $\langle R, +, \times \rangle$ be a ring and let $R^* = R - \{0\}$. If R is a commutative ring with identity $e \neq 0$ and R^* is an abelian group then R is called a **field**. The set of all rational numbers, that is, all numbers of the form a/b, where $a, b \in Z$ with $b \neq 0$ is a field.

The **characteristic** of F denoted as $ch(F)$ has the same definition as the characteristic of a ring. However, if $ch(F)$ exists then it must either be a prime or 0 ([DF91], page 422 and [PW66], page 159).

If F is a field, then a **subfield** of F is a subset of F which is also a field with respect to the operations of F. The intersection of all subfields of F is the smallest subfield of F and is referred to as the **prime subfield** of F.

Theorem 2.7 *p is prime if and only if Z_p is a field.*

Proof: See [PW66], page 159.

Let E, F be fields where E is a subfield of F. Then we say F is an **extension field** of E. Let F be the extension field of E and $u \in F$. Let $E(u)$ denotes the intersection of all subfields of F that contains both E and u. By nature $E(u)$ is a field. By definition, $E(u)$ is contained in every subfield of F that contains E and u and hence $E(u)$ is the smallest subfield of F containing E and u. $E(u)$ is said to be a **simple extension** of E.

Let E be a subfield of F. If $E \neq F$ then we say E is a **proper subfield** of F. A field contains no proper subfield is called a **prime field**. Since F is an extension field of E, F can be viewed as a vector space over E. The dimension of F over E is denoted by $[F : E]$ and it is called the degree of F over E.

Lemma 2.1 *Let $E < K < F$*

$$[F : E] = [F : K][K : E]$$

Proof: See proof at [Rom95], page 40.

2.5 Polynomial Rings

Let R be an arbitrary ring. A polynomial of degree n over R is an expression of the form,

$$f(x) = \sum_{i=0}^{n} a_i x^i \tag{2.2}$$

where n is a non-negative integer and the coefficients $a_i \in F$ where $0 \leq i \leq n$ and $a_n \neq 0$, and x is a symbol not belonging to R and is called the indeterminate over R.

The element a_n is called the **leading coefficient** of $f(x)$, a_0 is the **constant term** and n is called the degree of $f(x)$. If not all coefficients a_i equal zero then $f(x)$ is called a **non-zero polynomial**. If all coefficients a_i equal zero then $f(x)$ is called a **zero polynomial**.

Given two polynomials

$$f(x) \;=\; \sum_{i=0}^{n} a_i x^i \quad \text{and} \quad g(x) \;=\; \sum_{i=0}^{n} b_i x^i, \tag{2.3}$$

we say they are equal if

$$a_i \;=\; b_i \qquad \text{for} \;\; i \;=\; 1 \cdots n.$$

Addition of two Polynomials:

$$f(x) + g(x) \;=\; \sum_{i=0}^{n} (a_i + b_i) x^i \tag{2.4}$$

Multiplication of two Polynomials :

$$f(x) \times g(x) \;=\; \sum_{k=0}^{m+n} c_k x^k \quad \text{where} \quad c_k \;=\; \sum_{\substack{i+j \;=\; k \\ 0 \le i \le n, 0 \le j \le m}} a_i b_j \tag{2.5}$$

A ring formed by the polynomial over R, with all of the above operations defined, is called the **polynomial ring over** R and denoted as $R[x]$. Let F be a field, then $F[x]$ is called the polynomial ring over F. That is, if $f(x) \in F[x]$ then $a_i \in F$, and $0 \le i \le n$.

Theorem 2.8 (Division Algorithm For Polynomials) *Let F be a field. Let $\alpha(x)$ and $\beta(x) \in F[x]$ and let $\alpha(x) = 0$. Then there exist unique polynomials $q(x)$ and $r(x)$ in $F[x]$ such that*

$$\beta(x) \;=\; \alpha(x)\; q(x) + r(x), \qquad degree\; r(x) < degree\; \alpha(x)$$

Proof: See [PW66], page 168.

Now let $f(x), g(x) \in F[x]$ over a field F. $f(x)$ divides $g(x)$ if there is a polynomial $h(x) \in F[x]$ such that $g(x) \;=\; f(x)h(x)$. We write $f(x)|g(x)$. We write $f(x) \nmid g(x)$ otherwise. If F is a field, an element $s \in F$ is a root of the polynomial $f(x) \in F[x]$ if $f(s) \;=\; 0$.

Theorem 2.9 *Let F be a field and $f(x) \in F[x]$, $deg f(x) = n \geq 0$. Then $f(x)$ has at most n distinct roots in F.*

Proof: For detailed proof, please refer to [PW66], page 171.

2.6 Finite Fields

The **finite field** has a special section of its own because it plays an important role in the applications of field theory in cryptography.

Let Z_p be a residue class ring of integers modulo a prime p. Let F_p be the set $\{0, 1, \cdots, p-1\}$ of integers and let $\varphi : Z_p \to F_p$ be the mapping defined by $\varphi([a]) = a$ for $\{a = 0, 1, \cdots, p-1\}$. Then F_p is a field structure induced by φ. It is called a **finite field** which contains only a finite number of elements. If F is a finite field, then F^* denotes the multiplicative group of all nonzero elements of F. The following are some facts about finite fields:

Theorem 2.10 *If F is a finite field, then*

1. *F has a prime characteristic,*
2. *F^* is cyclic,*
3. *Any finite extension of F is simple.*

Proof: Please refer to [Rom95], page 161 for proofs.

Remarks to part 2 of Theorem 2.10: A generator of the cyclic group F_q^* is called a primitive element of F_q. It follows from part 3 of Theorem 2.6 that F_q contains $\phi(q-1)$ primitive elements ([LN83], page 51).

Theorem 2.11 *The prime subfield of a field F is isomorphic to either F_p or Q if the characteristic of F is a prime or 0. Q is the field of rational numbers.*

Proof: See proof in [LN83], page 30.

Theorem 2.12 *Let F be a finite field with $char(F) = p$ then F has p^n elements where n is the degree of F over its prime subfield.*

Proof: Since F is finite its characteristic is a prime p as mentioned in the Field section. The prime subfield is isomorphic to F_p by Theorem 2.11 and thus contains p elements. The rest of the proof follows from Lemma 2.1.

2.6.1 About F_q

If F_q is a finite field of order $q = p^m$, where p is a prime then the characteristic of F_q is p. Moreover, F_q contains a copy of Z_p as a subfield. Hence, F_q can be viewed as an extension field of Z_p of degree m. The non-zero elements of F_q form a group under multiplication. It is called the multiplication group of F_q and it is denoted by F_q^*.

2.7 Cryptography

Cryptography involves applying mathematics to ensure the integrity and confidentiality of messages that are sent from sender to receiver. The original message is called **plaintext**. The sender uses a cryptographic **encrypting function** to transform the plaintext into some unreadable **ciphertext**. The receiver uses the corresponding **decrypting function** which is the inverse of the encoding function to transform the ciphertext back to plaintext.

Encryption and decryption require the use of some secret information which we refer to as the **key**. Depending on the cryptographic mechanism used, the same key might be used for both encryption and decryption, this is called **symmetric cryptography**. It is called **asymmetric cryptography** if different keys are used. In an asymmetric cryptographic environment, if the encryption key is publicly known and the decryption key remains private, then we refer to such a system as a **Public-Key Cryptosystem**.

The field of cryptography comprises more than just encoding and decoding messages. With a few basic cryptographic tools it is possible, for example, to build elaborate schemes and protocols which allow us to pay bills with electronic money, to prove we know certain information without revealing the actual information itself and to share a secret in such a way that no fewer than three out of five people can reconstruct the secret.

As the Internet and other forms of electronic communication become more prevalent, electronic security becomes increasingly important. Protecting our email, credit card information and corporate data are essential. However, authentication is as fundamental as privacy. We sign our name on documentation to show our ownership or knowledge of the document. We now need to replicate such authentication procedures to meet our new lifestyle which has become heavily dependent on electronic communications and transactions.

The following sections not only explain some fundamental cryptographic terms, but also introduce some special techniques and concepts that will be needed in order to understand the rest of this thesis.

2.7.1 Public-Key Cryptosystems

Traditional symmetric cryptography is problematic in that both the sender and receiver of a message must agree on one secret key before any kind of secret transmission can occur. If they are physically separated, they must trust a courier or a secure phone line to send the secret key across. In a symmetric cryptographic system with n users, each pair of users may potentially need to communicate securely, which implies that each user pair must share a distinct encryption key. For an n users' environment, each user may have to keep $n-1$ keys. Therefore, the total number of keys in the symmetric cryptographic system that are needed to be generated is $n(n-1)/2$ (or approximately n^2). With the asymmetric cryptographic system (i.e. the Public-Key Cryptosystem), each user needs to generate one encryption key and one decryption key, therefore n encryption keys and n decryption keys are needed for the system. The total number of keys needed in this asymmetric cryptographic system is $2n$. If the number of users is large then the symmetric cryptographic system could have a serious key management problem.

Diffie and Hellman [DH76] introduced the concept of public-key cryptography in order to solve the key management problem. Each person creates a pair of keys, the **public key** and the **private key**. The public key is published on some trusted directory and the private key is kept secret by the owner. Anyone can send an encrypted message by using the public key of the receiver, and only the receiver can decrypt the message by using his private key.

An example of the public key system is the RSA system introduced by Rivest, Shamir and Adleman[RSA78]. The scheme works as follows. Let p, q be two large prime numbers and $n = pq$ as modulus. Choose a number e where $e < n$ and $gcd(e, \phi(n)) = 1$. Find d such that $ed \equiv 1 \ mod \ \phi(n)$. Let (n, e) be the public key and (n, d) be the private key. The numbers p and q can be kept with the private key or destroyed.

Suppose Alice wishes to send message m to Bob using RSA encryption and decryption. Then the system can be defined as follows:

RSA encryption on Alice's side,

$$c = m^e \ mod \ n$$

where e and n is Bob's public key and c is: the ciphertext. Alice sends c to Bob through a regular, insecure channel.

When Bob receives the ciphertext, the RSA decryption on Bob's side is,

$$m = c^d \ mod \ n$$

Since only Bob knows d, normally speaking only Bob can decrypt the message correctly.

There exist efficient algorithms which can perform multiple-precision arithmetics on large integers, like p, q, d. Examples include the Radix Representation for multiple-precision integer arithmetics using the classic methods, the Extended Euclidean algorithm for finding gcd of two integers, and the Montgomery algorithm for calculating modulo exponentiation. These algorithms and many others can be found in [MvOV97], Chapter 14.

If an RSA system is to be secure, it is necessary to have $n = pq$ be as large as possible so that factoring n is computationally infeasible. In 1994, Atkin was able to factor numbers having up to 129 decimal digits using the Quadratic Sieve (QS) algorithm [AGLL94]. The 129-digit number was factored in eight months with the help of 1600 computers all over the world through the Internet. But, 1600 computers are only a very small fraction of the potential computing power that is available on the entire Internet. It is estimated by Atkin[AGLL94] that if the whole Internet resource were fully used, factoring a number of 130 digits would take only a few hours. However, the resource management of such a factoring algorithm would be a very complex problem.

Year	Number of digits	mips years
1984	71	0.1
1988	106	140
1993	120	825
1994	129	5000

Table 2.1: Estimated running time for numbers factored with QS.

A progress report for factoring integers using the QS algorithm is given in Table 2.1[MvOV97]. The estimation time is measured in mips years[1]. Factoring a 129 digit number using the QS algorithm will take one computer 5000 years at the rate of one mips or equivalently 1600 computers for eight months only. Therefore, to ensure the security of the system, p and q should be prime numbers with about 100 digits and n be about 200 digits ([Sti95], page 126). Interested readers can find the QS algorithm on page 95 of [MvOV97].

2.7.2 Discrete Log Problem

The security of many cryptographic techniques depends on the intractability of the **Discrete Logarithm Problem(DLP)**. The Diffie-Hellman key agreement, El-Gamal signature scheme and encryption scheme are good examples of cryptographic systems which are based on the DLP.

Let G be a finite cyclic group of order n. Let α be a generator of G and let $\beta \in G$. The **discrete logarithm** of β to the base α denoted as $\log_\alpha \beta$ is the unique integer x, $0 \leq x \leq n-1$ such that $\beta \;=\; \alpha^x \; (mod \; n)$.

The group of most interest in cryptography is the multiplicative group F_q^* of the finite field F_q, particularly of the multiplicative group Z_p^* of the integers modulo a prime p. The order of this is $p-1$. For a review, please refer to the Subsection 2.6.1.

There are algorithms available to compute the discrete logarithm. For example: i) The baby-step-giant-step algorithm, with a running time of $\mathcal{O}(\sqrt{n})$ group multiplications where n is the input; ii) The Pohlig-Hellman algorithm with a running time of $\mathcal{O}(\sum_{i=i}^{r} e_i(\log\; n + \sqrt{p_i})$ groups multiplications where p_i and e_i is a prime factorization of n of the form $n = p_1^{e_1} p_2^{e_2} p_3^{e_3} \cdots p_r^{e_r}$; iii) The index-calculus algorithm with a running time of $L_q[\frac{1}{2}, c]^2$.

[1] A mips year is equivalent to the computational power of a computer that is rated at 1 mips (one million instructions per second) and utilized for one year.

[2] This is called Subexponential Running Time. Let A be an algorithm whose inputs are either elements of

The **Diffie-Hellman Problem(DHP)** is the following: Given a prime p, a generator α of Z_p^* and the elements $\alpha^a \ (mod\ p)$ and $\alpha^b \ (mod\ p)$, finding $\alpha^{ab} \ (mod\ p)$ is difficult. This is based on the assumption that computing $\alpha^{ab} \ (mod\ p)$ from α^a and α^b is **as hard as** obtaining x from $\alpha^x = \beta$ (i.e. the DLP). If an eavesdropper attempts to derive a from $\alpha^a(mod\ p)$ or b from $\alpha^b(mod\ p)$, just as the owner of a and b would, such computation would be an instances of the DLP. Therefore, the DHP is no harder than the DLP. The DLP is used to implement the Diffie-Hellman key agreement protocol. The details of the protocol can be found in [MvOV97] page 515.

There are several algorithms for solving DLP, but none of them perform in polynomial time. Shanks' algorithm and the Pohlig-Hellman algorithm are among the strongest attacks and they can be found in [Sti95], page 165-170.

2.7.3 Zero-Knowledge Proof

A **Zero-Knowledge Proof** is a mathematic protocol that was first introduced by Goldwasser, Micali and Rackoff [GMR89]. This technique promises to defend the validity of the information but hides the proof, therefore it is used by many highly secure cryptographic schemes.

Zero-knowledge proofs are instances of an **interactive proof system** wherein a prover and a verifier exchange multiple messages (challenges and responses), typically dependent on random numbers which they may keep secret. The prover's objective is to convince the verifier about the truth of an assertion. The verifier either "accepts" or "rejects" the proof. The traditional mathematical notion of a proof, is altered to an interactive proof which is probabilistic rather than absolute. An interactive proof in this context needs to be corrected only with bounded probability.

An interactive protocol possesses the following properties which are necessary for cryptographic applications:

1. **Completeness:** If both prover and verifier follow the protocol and the prover really knows the fact, then the verifier always accepts the proof with overwhelming probability.

a finite field F_q or an integer q. If the expected running time of A is of the form $L_q[\alpha, c] = \mathcal{O}(exp((c + \mathcal{O}(1))(\ln q)^{\alpha}(\ln\ \ln\ q)^{1-\alpha}))$, where c is a positive constant, and α is a constant satisfying $0 < \alpha < 1$, then A is a subexponential-time algorithm. Observe that for $\alpha = 0$, $L_q[0, c]$ is a polynomial in $\ln\ q$, while for $\alpha = 1$, $L_q[1, c]$ is a polynomial in q and thus fully exponential in $\ln q$.

2. **Soundness:** If the verifier follows the protocol but the prover does not know the fact, then the verifier always rejects the proof with overwhelming probability.

The definition of overwhelming depends on the application but it generally implies that the probability of failure is not of practical significance.

An interactive proof which has the completeness and soundness properties is said to be a **proof of knowledge**. An interactive proof must have a soundness property to be of cryptographic use, and the main property of zero-knowledge protocol is the zero-knowledge aspect of it as described as follows:

3. **Zero-Knowledge:** The verifier learns nothing about the fact that is being proven, except that it is correct. The verifier cannot later prove the fact to anyone else.

The zero-knowledge property does not guarantee that a protocol is secure (i.e. the probability of it being easily defeated is negligible). The term "defeated" here means that soundness or completeness or both properties no longer hold. Neither the completeness nor the soundness property has much value unless the underlying problem faced by an adversary is computationally hard.

We shall use the graph 3-colouring problem to illustrate conceptually how a typical round of zero-knowledge protocol works. This is followed by a more concrete example of a simplified protocol.

Graph 3-colouring

Suppose that the prover Picard is given a graph. We shall visualize the vertices as small balls containing little coloured lights and joined by bars wherever there is an edge. The light in each ball can either be red, green or blue. All the vertices are ordered. Picard has a device A which assigns each ball the colour red, green or blue, and a device B which chooses a random colour permutation of three colours and assigns each vertex according to the colour permutation.

Assumptions: Suppose the lights inside the vertex ball are hidden from view. The lights are visible only when the bar connecting two balls (vertices) is being grabbed.

Suppose Picard has figured out a 3-colouring of the graph and uses the device A to set the vertices with the corresponding colours. Let $n = |V|$ (the number of vertices) and $m = |E|$ (the number of edges). The following is the protocol between Picard and the verifier Vivian:

1. Vivian is allowed to grab any one of the edge-bars, Picard reveals the vertices' colour at each end. If Vivian sees two vertices with different colours, then Picard has a valid colouring.

2. Picard uses device B to permute the colours.

3. Picard and Vivian repeat steps 1 and 2, until Vivian has tested as many bars as she wishes.

Suppose the above steps are executed m^2 times. If Picard has 3-coloured the graph, the verifier accepts the proof with probability 1. If Picard cannot 3-colour the graph, eventually in step 1, the two vertices will have the same colour; and no matter how Picard plays at each round, the verifier will reject them with the probability of at least $1/m$. This implies that the verifier will accept the proof (without detecting that "something is wrong") with probability at most $(1 - m^{-1})^{m^2}$. At the end of the protocol, Vivian will learn nothing about the colouring because of the random permutation of the colours. However, she will be convinced about Picard's ability to 3-colour the graph.

Fiat-Shamir Identification Protocol

The graph 3-colouring example above serves as a good conceptual example. Here we show a more concrete example which will better suit the signature scheme which will be introduced in the next two chapters. This example is cited from [MvOV97], page 408. It is a rather simplified version, but it serves the purpose of illustrating the idea of this section of the thesis.

Suppose A wishes to prove his knowledge of s to B in t executions of a 3-pass protocol. Before the actual protocol, both sides agree on the following conditions:

1. A trusted center T selects and publishes an RSA-like modulus $n = pq$ but keeps primes p and q secret.

2. A selects a secret s which is coprime to n, $1 \leq s \leq n-1$, computes $v = s^2 \ mod \ n$, and registers v with T as his public key.

The Actual Protocol: The following steps are iterated t times (sequentially and independently). B accepts the proof if all t rounds succeed.

1. A chooses a random r, $1 \leq r \leq n-1$, and sends $x = r^2 \ (mod \ n)$ to B.

2. B randomly selects a challenge bit $e = 0$ or $e = 1$, and sends e to A.

3. A computes and sends B the response y, either $y = r$ (if $e = 0$) or $y = rs \ (mod\ n)$ (if $e = 1$).

4. B rejects the proof if $y = 0$, otherwise accepts upon verifying $y^2 \equiv x \cdot v^e \ (mod\ n)$.

The challenge e requires that A be capable of answering two questions, one of which demonstrates his knowledge of the secret s. The other question prevents cheating. An adversary impersonating A might try to cheat by selecting any r and setting $x = r^2/v$, then answering the challenge $e = 1$ with a correct answer $y = r$; but he would be unable to answer the same $e = 0$ challenge which requires knowing a square root of $x \ (mod\ n)$. A knowing s can answer both questions, but otherwise can at best answer one of the two questions, and so has a probability of only $1/2$ of escaping detection. To decrease the probability of cheating to an acceptably small value 2^{-t}, the protocol is iterated t times, with B accepting A's identity only if all t questions are successfully answered.

2.7.4 Shamir's Secret Sharing

Secret sharing is a multi-party protocol which is related to **Key Establishment**. Key establishment refers to a protocol where a shared secret becomes available to two or more parties for subsequent cryptographic use.

It is desirable to create backup copies of a cryptographic key in order to safeguard the key from loss. However, the greater the number of copies made, the greater the risk of security exposure; the fewer the number, the greater the risk that all are lost. Secret sharing schemes address this issue by allowing enhanced reliability without increasing risk. They also facilitate distributed trust or shared control for critical activities by granting critical actions to k of n users. The idea of **secret sharing** is to start with a secret and divide it into **shares** which are then distributed amongst users such that the pooled shares of specific subsets of users allow reconstruction of the original secret.

A (k, n) threshold scheme $(k \leq n)$ is a method by which a trusted party computes secret shares s_i, $1 \leq i \leq n$ from an initial secret S and securely distributes s_i to user p_i such that any k or more users who pool their shares may easily recover S, but any group knowing only $k - 1$ or fewer may not. A perfect thresholding scheme is one in which knowing only $k - 1$ or fewer shares provides no advantage to an opponent, over knowing no pieces.

Shamir's (k, n) thresholding scheme[Sha79] is based on polynomial interpolation over a finite field F_q. Suppose the polynomial

$$f(x) \;=\; a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

is constructed such that the coefficient a_0 is the original secret and all other coefficients are random elements in F_q. For each user i, the partial secret share s_i can be calculated with $s_i = f(x_i)$ where x_i is the user's private information ranging over F_q. Each (x_i, s_i) is a point on the curve defined by the polynomial $f(x)$. Given any k shares, the polynomial is uniquely determined and the secret a_0 can be computed. However, given $k-1$ or fewer shares, the secret can be any element in the field. The scheme is rather efficient since the polynomial evaluation and interpolation can be performed in $\mathcal{O}(n \ \log \ n)$ time.

Shamir's scheme can be applied to a practical situation such as the following. Suppose a company uses digital signatures of company executives to sign all of its cheques. It is unwise to give the company's signature to each and every company executive, and it is very inconvenient to ask all of them to cooperate whenever a cheque is needed to be signed. Therefore, the most logical way to solve such a problem is to use the thresholding scheme, where only a subset of the executives is needed when a cheque needs to be signed.

Shamir's scheme has the following properties which are useful to cryptographic applications:

1. The size of each s_i does not exceed the size of the original data.
2. When k is fixed each s_i can be added or deleted dynamically (i.e. when an executive joins or leaves the company) without affecting the other s_i pieces.
3. It is easy to change s_i without changing the original data - in such a case the only thing that is needed is a new polynomial function which has the same number of free terms.

In general, Shamir's scheme is ideal for applications in which a group of mutually suspicious individuals with conflicting interests must cooperate. If one chooses parameters k and n appropriately, the scheme gives any sufficiently large majority the power to take some action, while giving any sufficiently large minority the power to block it.

Chapter 3

Digital Signature Schemes

In the traditional business world, a signed contract serves as a legal document of an agreement between two parties. With the advance of modern technology, we can expect a greater universal adaptation of telecommunications in business transactions. Therefore, electronic authentication of any business system that involves contracts and billing becomes a rather important issue. Without it, business would not function. Digital signatures are methods of signing a message and storing it in an electronic form. They are the digital counterpart of a handwritten signature that can be transmitted over a computer network. For a purely digital replacement of handwritten signatures, the signer must be able to produce an authentication that can be checked by every receiver, but cannot be reproduced by anyone including the receiver. To use the signature, the receiver will need to show the verification transaction script to a third party, just like a handwritten signature on a document. Take for example, the record registration clerk who issues student cards. In order for the student to obtain the student card, the student must show the departmental signed registration form to the clerk. By looking at the registration form the clerk can then decide whether to grant the privilege to the receiver who has presented this signed document.

Digital signatures have many applications in information security, authentication and data integrity. The concept of digital signatures was recognized some years before any practical realization was available. Diffie and Hellman introduced the concept in 1976[DH76]; Rivest, Shamir and Adleman (RSA) put the concept into practical implementation in 1978[RSA78]. Since then, many researches have been conducted regarding the development of alternative digital signature techniques which are used in different applications. Some of these provide significant advantages in terms of functionality, characteristics and implementation. However, the RSA signature scheme remains the most versatile and popular cryptographic technique today.

This chapter presents the difference between digital signatures and handwritten signatures, the framework of digital signatures, and gives a brief survey of some practical techniques, developments and attacks. Detailed descriptions of the El-Gamal and RSA schemes and a brief description of interesting implementations such as Group Signature Schemes and Blind Signature Schemes will be given. Special attention is paid to undeniable signatures.

3.1 Digital Signatures vs Handwritten Signatures

Digital signatures and handwritten signatures are meant to solve the same problems. However, they do have some fundamental differences.

1. **The signing process**. In the handwritten method, the signer physically signs the document and the signature becomes part of the document that is being signed. A digital signature is not physically attached to the message; it is produced separately and is attached to the message afterwards.

2. **The verifying process**. A handwritten signature is verified by comparing it to other authentic signature(s), usually by human sight and/or memory. Digital signatures can be verified using a publicly known verification process. This confirms the ownership of the signature. Anyone can perform a verification of a digital signature.

3. The 'copies' of a signed digital message are identical to the original. However, a handwritten copy of a signed paper document can usually be distinguished from the original.

Digital signatures are believed to be superior to handwritten signatures in that it attests to the content of the message, as well as to the identity of the signer. As long as a secure data formatting function is used (i.e. hash function, redundancy function which will be introduced later), it is very difficult to obtain someone's signature from one document and attach it to another. Even a subtle alteration of the signed message will cause the verification process to fail.

3.2 Basic Definitions

This section introduces the concept and functionality of functions and algorithms that could be involved in different digital signature schemes. Throughout this section we will use $\mathcal{M}$ to represent the message space, which is a finite set of messages(elements) to which a signer can affix digital signatures; $\mathcal{M}_S$ will represent the finite signing space where the signature transformation takes place. Note that $\mathcal{M} \neq \mathcal{M}_S$ and the set $\mathcal{M}_S$ may have more elements than $\mathcal{M}$. The finite set $\mathcal{S}$ is the signature space. Elements of $\mathcal{S}$ are associated with elements of $\mathcal{M}$.

The redundancy function R is a 1-to-1 mapping from $\mathcal{M} \rightarrow \mathcal{M}_S$. Its inverse R^{-1} maps $\mathcal{M}_R \rightarrow \mathcal{M}$ where $\mathcal{M}_R$ is the image set of R and $\mathcal{M}_R \subset \mathcal{M}_S$. The redundancy function transforms the original message $m \in \mathcal{M}$ to a larger space, namely $\mathcal{M}_S$. Having a redundancy function enhances the security of the system because, with the function, the signing space $\mathcal{M}_S$ is larger than the message space M. This implies that for an attacker who manages to obtain the signature, searching for the corresponding message will not be a trivial task. The redundancy function is often used in digital signature schemes with recovery, such as the RSA Signature Scheme which will be explained in Section 3.5.2. A more graphical perspective of the redundancy function can be found in Fig.3.3.

The hash function h is a mapping $\mathcal{M} \rightarrow \mathcal{M}_h$ where $\mathcal{M}_h$ is the hash value space and $\mathcal{M}_h \subset \mathcal{M}_S$. It is a transformation that takes a variable size input string $m \in \mathcal{M}$ and returns a smaller fixed-size string called the hash value $h(m) = y$ where $y \in \mathcal{M}_h$. Given m and function h, $h(m)$ is easy to compute. The hash value will be signed instead of the potentially large message because it is easier to manage by most algorithms. There are two types of hash functions — strong and weak. Suppose we have inputs m and m' and outputs y and y'. For a weak hash function, given y, it is computationally infeasible to find m' such that $h(m') = y$. An adversary may easily precompute outputs for any small set of inputs and thereby invert the hash function trivially for such output with tabular look up of all the pre-computed pairs. For a strong hash function, it is computationally infeasible to find any two distinct inputs m and m' such that $h(m) = h(m')$. We say $h(m)$ is a strong function. The hash function is widely used by the signature scheme with an appendix such as the El-Gamal Scheme which will be introduced in Section 3.5.1.

A **digital signature** is a data-string which associates a message with its originator. A **digital signature signing process** consists of a digital signature generating algorithm along with, or without, a method of formatting data into a form which can be signed. An example of such a message formatting function would be the hash function or the redundancy function which were introduced earlier. The following is the general signing model. $\mathcal{S}_x$ is the signing function of signer x and $\mathcal{S}_x : \mathcal{M}_h \rightarrow \mathcal{S}$ or $\mathcal{S}_x : \mathcal{M}_s \rightarrow \mathcal{S}$ depending on the signing function used.

For signer x to sign a message $m \in \mathcal{M}$, he/she must:

1. Format the message m into $\tilde{m}$ using either $h(m) = \tilde{m}$ or $R(m) = \tilde{m}$ depending on the signature scheme used.
2. Compute $s = S_x(\tilde{m})$.
3. Send the pair (s, m) to the receiver (s is called the signature for message m by signer x).

A **digital signature verifying process** consists of a verification algorithm, along with or without a method of recovering data from a formatted messages and is publicly available. An example would be the inverse of a redundancy function R^{-1}. The following is a general verification model with the verification function of x denoted as $\mathcal{V}_x()$ which maps $\{\mathcal{M} \times \mathcal{S}\} \rightarrow \{True, False\}$:

For the receiver to verify signature s on a message m that was created by x, the verifier must:

1. Obtain the publicly available verification function.
2. Unformat the message $\tilde{m}$ if necessary.
3. Compute $u = \mathcal{V}_x(m, s)$.
4. Accept the signature if $u = True$, or reject it otherwise.

A **digital signature scheme** consists of a signature process and an associated verification process. If Alice wishes to send Bob a message, Alice computes her signature using her signature signing process and sends the signed document to Bob. Bob uses Alice's publicly known verification process to verify Alice's signature. The verification process must satisfy the conditions: $\mathcal{V}_x(m, s) = True$ and $\mathcal{V}_x(m, s^*) = False$ where $s^* \neq s$.

3.3 Attacks to Digital Signature Schemes

The goal of the adversary is to produce a false signature on a previously unsigned document that will be accepted by a verifier. The following gives a set of criteria for what it means to break a signature scheme.

1. **Total break.** The adversary is either able to compute the signer's private information or to find an efficient algorithm that produces the same result as the signing algorithm.

2. **Selective forgery.** The adversary is able to create a valid signature for some messages. The forged signature generating algorithm does not involve the legitimate signer.

3. **Existential forgery.** The adversary is able to forge a signature for at least one message. The adversary has little or no control over the message bearing the compromised signature, and the legitimate signer may be involved in the deception.

3.4 Classification of Digital Signature Schemes

There are usually two general classes of digital signature schemes.

1. Digital signature schemes with appendix.
    - This kind of signature scheme requires the original message to be part of the verification algorithm's input. See Fig.3.1 and Fig.3.2.
    - Example: El-Gamal Signature Scheme.

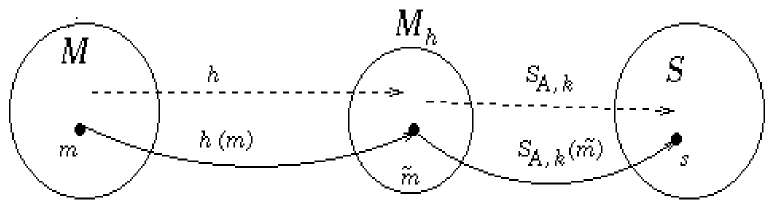

Figure 3.1: Digital signatures with appendix - the signing process.

The dotted line in Fig.3.1 indicates the direction of the transformation. The hash function h is a mapping from $\mathcal{M}$ to $\mathcal{M}_h$. The signature function $\mathcal{S}_{A,k}$ of signer A with parameter k is a mapping from $\mathcal{M}_h$ to the signature space $\mathcal{S}$. The solid line indicates the input of the functions. The message m is transformed to $\tilde{m}$ by the hash function h, and $\mathcal{S}_{A,k}$ takes $\tilde{m}$ as input and produce signature $s \in \mathcal{S}$.

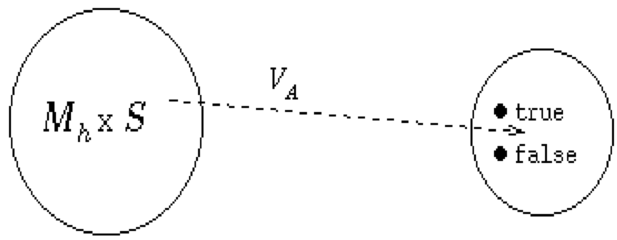

Figure 3.2: Digital signatures with appendix - the verification process.

The dotted lines indicate the direction of the verification function in Fig.3.2.

2. Digital signature schemes with message recovery

    - This kind of signature scheme does not require the original message to be part of the verification algorithm's input. The original message can be retrieved from the signature itself. See Fig.3.3
    - Example : RSA Signature Scheme.

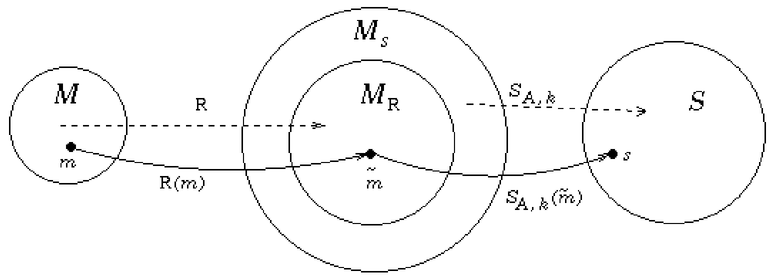

Figure 3.3: Digital signatures with message recovery.

The redundancy function R is a mapping from $\mathcal{M}$ to $\mathcal{M}_R$ as indicated by the dotted line in Fig.3.3 and the signature function $\mathcal{S}_{A,k}$ is a mapping from $\mathcal{M}_S$ to $\mathcal{S}$. Notice that the set $\mathcal{M}_R$ is a subset of set $\mathcal{M}_S$. The message m is the input of R that produces $\tilde{m}$. The formatted message $\tilde{m}$ is passed to the signing function $\mathcal{S}_{A,k}$ and produces s as output, just as the solid line indicates.

3.5 Different Digital Signature Schemes

3.5.1 El-Gamal Signature Schemes

The El-Gamal scheme was introduced by El-Gamal in 1984 [EG85]. It is a public-key algorithm which bases its security on the difficulty of solving the Discrete Log Problem (DLP). It generates digital signatures with an appendix on a binary message of arbitrary length, and it requires a hash function $h : \{0,1\}^* \rightarrow Z_p$ where p is a large prime number. The public key is used for both enciphering and verifying. However, these two transformations are distinct.

Key Generation for the El-Gamal Scheme

The signer should do as follows:

1. Generate a large random prime p and a generator α of the multiplicative group Z_p^*.[1]
2. Select a random integer a, $1 \leq a \leq p-2$.
3. Compute $y = \alpha^a \ (mod\ p)$.
4. Make the public key (p, α, y) and the private key a.

El-Gamal Signature Generation and Verification

1. Signature generation

   The signer signs a message m of arbitrary length. The signature can be verified by using the signer's public key. To generate the signature the signer should:

   (a) Select a random secret integer k, $1 \leq k \leq p-2$, with $gcd(k, p-1) = 1$.[2]
   (b) Compute $r = \alpha^k \ (mod\ p)$.
   (c) Compute $k^{-1}\ mod\ (p-1)$.
   (d) Compute $s = k^{-1}\{h(m) - ar\}\ mod\ (p-1)$.
   (e) Signer's signature for m is the pair (r, s).

[1] Efficient algorithm for generating a random prime and a generator of Z_p^* is listed in [MvOV97], page 164.

[2] The Extended gcd Algorithm is an efficient algorithm for calculating gcd of two numbers [MvOV97], page 608.

2. Verification

   To verify a signer's signature (r, s) on m , the verifier should do the following:

   (a) Obtain the signer's authentic public key (p, α, y).
   (b) Verify that $1 \leq r \leq p-1$; if not, then reject the signature.
   (c) Compute $v_1 = y^r r^s \ (mod\ p)$.
   (d) Compute $h(m)$ and $v_2 = \alpha^{h(m)} \ (mod\ p)$.
   (e) Accept the signature if and only if $v_1 = v_2$.

Proof that the signature verification works: If the signature was generated by A, then $s \equiv k^{-1}\{h(m) - ar\} \ (mod\ p-1)$. Multiplying both sides by k gives $ks \equiv h(m) - ar(mod\ p-1)$, and rearranging yields $h(m) \equiv ar + ks \ (mod\ p-1)$. This implies that $\alpha^{h(m)} \equiv \alpha^{ar+ks} \equiv (\alpha^a)^r r^s (mod\ p)$. Thus, $v_1 = v_2$, as required.

3.5.2 RSA Signature Schemes

RSA was the first practical signature scheme [RSA78]. It is a deterministic digital signature scheme which provides message recovery. The signing space $\mathcal{M}_\mathcal{S}$ and signature space $\mathcal{S}$ are both in Z_n^*. A **redundancy function** $R : \mathcal{M} \rightarrow Z_n$ is used and is publicly known.

Key Generation for the RSA Signature Scheme

To generate the public key and the private key for the RSA signature, the signer should:

1. Generate two large distinct random primes p and q.
2. Compute $n = pq$ and $\phi = (p-1)(q-1)$.
3. Select a random integer e, $1 < e < \phi$ such that $gcd(e, \phi) = 1$.
4. Compute integer d, $1 < d < \phi$ such that $ed = 1(\ mod\ \phi)$.[3]
5. The public key is (n, e) and the private key is d.

[3] The Extended Euclidean Algorithm can be used. For the complete algorithm, please refer to [MvOV97], page 67 and page 71.

RSA Signature Generation and Verification

The signer signs a message $m \in \mathcal{M}$. Anyone can verify the signer's signature and recover the original message m from the signature.

1. Signature Generation. The signer should do the following:
   (a) Compute $\tilde{m} = R(m)$, an integer in the range $[0, n-1]$.
   (b) Compute $s = \tilde{m}^d \ (mod\ n)$.
   (c) Then let s be the signature for the signer.
2. Verification. The verifier should:
   (a) Obtain the signer's authentic public key (n, e).
   (b) Compute $\tilde{m} = s^e \ (mod\ n)$.
   (c) Verify that $\tilde{m} \in \mathcal{M}_R$; if not , reject the signature.
   (d) Recover $m = R^{-1}(\tilde{m})$.

Proof that the signature verification works: If s is the signature for message m, then $s \equiv \tilde{m}^d \ (mod\ n)$ where $\tilde{m} = R(m)$. Since $ed \equiv 1 (mod\ \phi)$, $s^e \equiv \tilde{m}^{ed} \equiv \tilde{m}(mod\ n)$. Finally, $R^{-1}(\tilde{m}) = R^{-1}(R(m)) = m$.

3.5.3 Blind Signature Schemes

The blind signature scheme was first introduced by Chaum [Cha83]. It is a protocol between two parties: the sender A and the signer B. The idea is simple: for A to send a piece of information m^* (this is not the original message m) to B, which B signs and returns to A. From this signature, A can compute B's signature on a prior message m of A's choice. After the completion of the protocol, B knows neither the message m nor the signature associated with it. The purpose of a blind signature is to prevent signer B from observing the message he has signed, and the signature associated with it. The blind signature scheme is useful in digital cash (or electronic money) applications.

3.5.4 Group Signature Schemes

Chaum and Van Heijst [Cv91] introduced the notion of the group signature. A member of a group can digitally sign a document. The verifier can confirm that the document comes from the group, but not which individual in the group has signed the document. Unfortunately, each time a group member signs a document, a new set of keys (both public and private) must be generated for the signer. This disadvantage can become a big problem when a large number of messages needs to be signed. In case of dispute, the identity of the group member can be revealed by a designated group authority.

3.5.5 Undeniable Signature Schemes

The undeniable signature scheme was first formally defined by Chaum and Van Antwerpen [CvA89]. It is a non-self-authenticating scheme where the signature must be verified with the signer's cooperation. However, a dishonest signer may refuse to authenticate a genuine document. The notion of undeniability is that the signer of the message cannot prove his own signature is a forgery, and he cannot prove that a false signature is genuine. These properties are achieved by the verification protocol and by the disavowal protocol of the scheme. Each protocol is carried out in a challenge-response exchange model where the verifier, Alice, sends a challenge to the signer, Bob, and views the response from Bob to verify the signature. The scheme is implemented using a public-key algorithm based on the DLP. The signing part of the scheme is very much like any other DLP signature scheme. In the disavowal process, Alice sends a challenge and Bob's response shows that the signature is not his. The probability that a dishonest signer is able to successfully mislead the verifier in either verification or disavowal is $1/p$ where p is the prime number in the signer's private key.

Key Generation for Chaum-Van Antwerpen Undeniable Signatures

To generate an undeniable signature the signer must follow these steps:

1. Select a random prime $p = 2q + 1$ where q is also a prime.
2. Select a generator α for the subgroup of order q in Z_p^*.
   (a) Select a random element $\beta \in Z_p^*$ and compute $\alpha = \beta^{(p-1)/q} \ (mod\ p)^{\dagger}$.
   (b) If $\alpha = 1$ then go to step 2.a.
3. Select a random integer $a \in \{1, 2, \cdots, q-1\}$ and compute $y = \alpha^a \ (mod\ p)$.
4. The signer's public key is (p, α, y) ; the private key is a.

† Efficient algorithm for finding exponential - Repeated square-and-multiply algorithm can be found in [MvOV97], page 71.

Verifying Protocol for Chaum-Van Antwerpen Undeniable Signatures

The signer A signs a message m belonging to the subgroup of order q in Z_P^*. Any verifier B can verify this signature with the cooperation of A. The signature generation and verification protocols proceed as follows:

1. Signature Generation.
   (a) Compute $s = m^a \ (mod\ p)$.
   (b) A's signature on message m is s.
2. Verification. The protocol for B to verify A's signature s on m is as follows:
   (a) B obtains A's authentic public key (p, α, y).
   (b) B selects random secret integers $x_1, x_2 \in \{1, 2, \cdots, q-1\}$.
   (c) B computes $z = s^{x_1} y^{x_2} \ (mod\ p)$ and sends z to A.
   (d) A computes $w = (z)^{a^{-1}} \ (mod\ p)$ and sends w to B. Where $aa^{-1} \equiv 1 \ (mod\ q)$.
   (e) B computes $w' = m^{x_1} \alpha^{x_2} \ (mod\ p)$ and accepts the signature if and only if $w = w'$.

Disavowal Protocol for Chaum-Van Antwerpen for Undeniable Signatures

The procedure for disavowal of a signature s, is as follows:

1. B obtains A's authentic public key (p, α, y).
2. B selects random secret integers $x_1, x_2 \in \{1, 2, \cdots, q-1\}$, and computes $z = s^{x_1} y^{x_2} \ (mod\ p)$ and sends z to A.
3. A computes $w = (z)^{a^{-1}} \ (mod\ p)$ and sends w to B. Where $aa^{-1} \equiv 1 \ (mod\ q)$.
4. B selects random secret integers $x_1', x_2' \in \{1, 2, \cdots, q-1\}$, and computes $z' = s^{x_1'} y^{x_2'} \ (mod\ p)$, and sends z' to A.
5. A computes $w' = (z')^{a^{-1}} \ (mod\ p)$ and sends w' to B.
6. B computes $c = (w\alpha^{-x_2})^{x_1'} \ (mod\ p)$ and $c' = (w'\alpha^{-x_2'})^{x_1} \ (mod\ p)$. If $c = c'$, then B concludes that s is a forgery. A did not issue the signature s.

The purpose of the disavowal protocol is to convince the receiver that an invalid signature is a forgery. In addition, the signer cannot make the receiver believe that a valid signature is a forgery except with a very small probability. For a detailed and complete proof of the protocol, please refer to [Sti95], pages 223-224.

Chapter 4

Case study of Undeniable Signatures

4.1 Introduction

This chapter gives a brief overview of undeniable signatures, its properties and disadvantages. Also, two techniques will be described — Convertible Undeniable Signatures and Distributed Prover Undeniable Signatures, which can overcome some of the disadvantages of the original undeniable signature schemes. However, these two techniques do have some weaknesses of their own which will be addressed in Section 4.5. The main idea of the two techniques will be given in Sections 4.3 and 4.4 along with algorithms. Readers who are not interested in the implementation details can gain a good conceptual understanding of these two schemes without going through the algorithms. Finally, some suggestions for future work along these lines are provided at the end of the chapter.

4.2 Properties and Disadvantages of Undeniable Signatures

Undeniable signatures were first introduced by Chaum and Antwerphen in 1989[CvA89] and Chaum refined the protocol by adding the zero-knowledge interactive proof model to the signature confirmation and disavowal protocols [Cha90] in 1990. We have described the algorithm for undeniable signature schemes and zero-knowledge interactive protocols previously (Section 3.5.5 and Section 2.7.3 respectively), therefore, only a brief overview is presented in this chapter. In the undeniable signature schemes, the prover and the verifier exchange challenge and response messages. At the end of the exchange, the verifier will be convinced about the prover's knowledge with significantly high probability that the signature belongs to the prover in the confirmation process. In the disavowal process, the verifier is convinced with a significantly high probability that the signature does not belong to the prover.

Undeniable signatures protect the interests of both signer and recipient. They ensure the signatures will not be subsequently misused by the recipient. Any recipient who holds the signature can challenge its signer and the signer will not be able to give false responses. The signer is always able to convince the recipient that a valid, signature is valid and that an invalid signature is invalid. These properties are very desirable for commercial applications, especially those involving contracts, invoices and licenses. A software company can protect its reputation, and the customers will be assured of having quality software.

A disadvantage of undeniable signatures is that the signer has the ability to decide freely at any time to whom and when he wants to prove the validity of his signature on a document. Desmedt and Yung[DY91] proposed that the signer can be cheated by several verifiers not trusting each other. A group of verifiers can verify a signature simultaneously, without the prover of the signature being aware of proving the signature to more than one person. Attack of this kind can be done by setting the challenge collectively so that no subset of the verifiers could set the challenge on their own. This attack was criticized by Chaum in [Cha91] but it was later strengthen by Jakobsson [Jak94].

As we mentioned earlier, the cooperation of the signer is required during verification of undeniable signatures, but it is up to the signer to decide when or whether to participate in the verification. If the signer chooses not to cooperate, then the recipient is not able to use the signature he obtained. Chaum introduced the Designated Confirmer Scheme[Cha94] which overcomes this disadvantage. The Convertible Undeniable Signatures [BCD90] which will be introduced in the next section are another twist of the idea for dealing with the absence of the signer in the verification process.

4.3 Convertible Undeniable Signatures

The power of verifying undeniable signatures can be given to a designated agent. The agent has the power to verify a signature during a signer's absence but cannot produce a valid signature on behalf of the signer at any time. This idea is proposed by Boyar, Chaum and Damgård in [BCD90] [DP96], and is called a convertible undeniable signature.

The rest of this section gives a conceptual description of the scheme. A more technical description is covered in Sections 4.3.1 to 4.3.7.

The signing process requires two private keys (K_{s_1} and K_{s_2}) and one public key K_p. The primary private key K_{s_1} will be kept secret at all times and the secondary private key K_{s_2} may be released when desired. After K_{s_2} is released, the undeniable signatures on the document will be converted to an ordinary self-authenticating signature. The signer can have different keys for signing different messages.

There are two types of convertible undeniable signatures; the signature can be **totally converted** or **selectively converted**. Supposing the signer uses the same key for signing all of his message, for total conversion, after K_{s_2} is released, all undeniable signatures will be converted into ordinary self-authenticating signatures. Supposing the signer uses different keys for signing different messages, for selective conversion, the signer is required to remember the secondary private key K_{s_2} used to construct the signature on message m. Later on, when K_{s_2} is released to the public, only the undeniable signature on message m will be converted to an ordinary signature and other undeniable signatures will not be affected.

4.3.1 The Setup for Convertible Undeniable Signature Schemes

One chooses two large primes p, q with $q|(p-1)$ and a group generator $\alpha \ (mod\ p)$ of integers with order q.

4.3.2 Key Generation for Convertible Undeniable Signature Schemes

The signer performs the following calculations:

1. Generates secret parameters $x, z \in Z_q^*$.
2. Computes $y = \alpha^x \ \ (mod\ p)$.
3. Computes $u = \alpha^z \ \ (mod\ p)$.
4. The public key is (y, u) and the secret keys are (x, z).

Note : Let $x = K_{s_1}$ and $z = K_{s_2}$.

4.3.3 Signature Generation for a Convertible Undeniable Signature Schemes

To sign a message m, the signer performs the following:

1. Picks two secret random numbers $t, k \in Z_q^*$.
2. Computes $T = \alpha^t \ (mod\ p)$.
3. Computes $r = \alpha^k \ (mod\ p)$.
4. Computes $s = k^{-1}(m - xr) \ (mod\ q)$.
5. The signature for message m is the triple (T, r, s).

4.3.4 Confirmation Protocol for Undeniable Signatures

Let $w = T^{Tm}$ and $v = y^r r^s$ which can be computed from public information. The signature is valid if the equality $w^z = v$ holds. The signer proves the equality by proving $\log_w v = \log_\alpha u$ using the zero-knowledge protocol.

4.3.5 Disavowal Protocol for Undeniable Signatures

The signature is invalid if the equality $w^z = v$ does not hold where w and v are defined as in Section 4.3.4. The signer needs to prove the inequality that the discrete logarithm $\log_w v \neq \log_\alpha u$ using the zero-knowledge protocol.

4.3.6 Selective Conversion

By releasing the secret value t, only the signature that was signed by t will be converted into an ordinary signature. The signer can check the signature by checking $u^{tTm} \equiv y^r r^s \ (mod\ p)$ and $T \equiv \alpha^t \ (mod\ p)$.

4.3.7 Total Conversion

By releasing the secret parameter z, every signature that was signed by the signer with z can be verified by the verifier by checking $w^z = v$.

4.3.8 Example

Suppose Alice signs all documents during her lifetime with the convertible undeniable signature schemes. The secret information which is needed to convert all her signatures to ordinary signatures is placed with her lawyer Bob. After Alice dies, Bob can make all the secret information publicly known and all of Alice's signatures will be converted to ordinary signatures. If Alice uses different sets of keys for each message that she signed, after she passes away, Bob the lawyer may decide to release only a subset of Alice's keys. In this case, only messages which are signed by those subsets of the keys will be converted to ordinary signatures.

4.4 Distributed Prover with Undeniable Signatures

In regular convertible undeniable signature schemes, the signer gives an agent authority to verify signatures during his absence. However, this requires the signer to trust the agent completely. If the signer does not want to (or does not) trust a single person, he may want to authorize n agents such that verification requires at least k of these n persons to cooperate. The signing process is the same as for regular undeniable signature signing. The signer distributes K_{s_2} to n agents which are carefully chosen by the signer. During the verification process the verifier must interact with k of the n agents.

The idea of distributing the power of verifying was first experimented with by Ben-Or, Goldwasser and Wigderson in [BOGW88] as well as Chaum, Crépeau and Damgård in [CCD88]. Pedersen in [Ped91] made the practical implementation of the method more efficient. He used Shamir's secret sharing scheme to distribute the secondary private key to n agents (recall Section 2.7.4 for Secret Sharing Scheme). This setup is called a distributed prover protocol because at least k of n agents will be needed during the verification phase. Each agent is located in a different geographical location. The verification procedure allows k persons to verify signatures without finding the secret. Interactions between k persons are needed only in the setup phase when the secret key is being distributed.

4.4.1 Key Generation

The signer:

1. Selects two large integers prime p, q where q divides $(p-1)$.

2. Selects a generator α for the subgroup G_q.
3. Selects secret elements $x, z \in Z_q^*$.
4. Computes $y = \alpha^x$ and $u = \alpha^z$.
5. Public keys are (p, q, α, y, u), the first private key (K_{s_1}) is x and the secondary private key (K_{s_2}) is z .

4.4.2 Signature Generation

This will be the same as the convertible signature schemes using El-Gamal, as mentioned in Section 4.3.3. The signature for message m will be the triple (T, r, s).

4.4.3 Secret Distribution from the Signer

The signer distributes his secret with each agent i where $i = 1 \cdots n$, as follows:

1. He computes shares $z_i = f(x_i)$ for each agent using $f(x) = f_0 + f_1 x + \cdots + f_{k-1} x^{k-1}$, where $f_0 = z$, f_i where $i = 1 \cdots k-1$ are randomly chosen and x_i is the agent's identification code.
2. He computes $h_i = \alpha^{z_i}$ and publishes h_i. This is the corresponding public information of the secret share z_i.
3. He sends the share z_i to agent i and broadcasts $(\alpha^{f_i})_{i=0,\cdots,k-1}$ to all n agents.

4.4.4 Verifying Shares on the Agent's Side

Each agent i , upon the receiving of shares, does the following:

1. Computes $h_l = \prod_{j=0}^{k-1} (\alpha^{f_j})^{x_l^j}$ for all $l = 1, \cdots, n$.
2. Verifies that $h_i = \alpha^{z_i}$.
3. If this is false, then broadcasts z_i and stops; otherwise, accepts the share.

Although all h_i are public information, if agent i is not able to obtain the public information, h_i can be calculated as in step 1.

4.4.5 Distributed Verifying

Given a message m and its signature (T, r, s), both the signer and the verifier can compute $w = T^{Tm}$ and $v = y^r r^s$. The verifier is required to interact with each and every k agent through the following interactive protocol. Notice that these k agents can be any subset in n :

1. The verifier chooses $a, b \in Z$ computes the challenge $ch = w^a \alpha^b$ and sends ch to prover i.
2. The prover i selects a random number $r_i \in Z$ and computes $h_{i1} = ch^{r_i}$ and $h_{i2} = h_{i1}^{z_i}$. The prover sends h_{i1} and h_{i2} to the verifier.
3. The verifier sends a, b to the prover.
4. The prover i sends r_i to the verifier.
5. The verifier collects all k numbers of r_i and computes $h_{i1} = (w^a \alpha^b)^{r_i}$ and $h_{i2}^{a_i r_i^{-1}}$.

If $\prod_{i=1}^{k} h_{i2}^{a_i r_i^{-1}} = v^a T^b$ and $h_{i1} = (w^a \alpha^b)^{r_i}$ for all $i = 1, \cdots, n$, then the verifier accepts the signature.

Note : a_i is the agent's special identity number which is used by Pedersen [Ped91] in his paper. It is defined as $a_i = \prod_{h \neq j} \frac{x_{i_h}}{x_{i_h} - x_{i_j}}$.

4.4.6 Distributed Disavowal

The disavowal protocol is the same as the verification process, except that the prover will need to prove the inequality $\prod_{i=1}^{k} h_{i2}^{a_i r_i^{-1}} \neq v^a T^b$ to the verifier. If the inequality holds then the verifier rejects the signature.

4.4.7 Selective Conversion

This will be the same as in the regular convertible undeniable signatures. When t is released, that single undeniable signature is converted to an ordinary signature, leaving the others unaffected.

4.4.8 Total Conversion

This will be the same as in the regular convertible undeniable signatures. When z is released, all undeniable signatures are converted into ordinary signatures.

4.4.9 Example

Suppose a software company gives a unique and distinct signature to each of its software products. The company divides each of the keys into n shares and distributes them to n assigned dealers. Customers who need to verify the software's quality during holidays, weekends or after hours can call up any k of n dealers to carry out the verification process. If one day the company decides to go bankrupt, it can selectively convert for the software that is still available in the market during the bankruptcy process, and then have a total conversion when the bankruptcy is finalized.

4.5 Attacks

After discussing different kinds of convertible undeniable signature applications, this section describes attacks that can be applied to convertible undeniable signatures. Michel, Petersen and Horster discovered that the totally convertible undeniable signature does not convert all undeniable signatures into conventional signatures after the secret piece of information is released [MPH96]. Also, the underlying conventional signature is insecure; forgery can be made easily. The same attack can be applied easily to Pedersen's [Ped91] distributed undeniable signature schemes if the signer wishes to have a total conversion after the secret information is released. The attacks do not work if the scheme is selectively converted and done by distributing pieces of the parameter to several agents. Michel, Petersen and Horster have also suggested a heuristic method that will repair the scheme but unfortunately, the security of the modified scheme cannot be proven.

The attack is as follows: For a total conversion, if z is released and y is publicly known, the task of an attacker is to find or produce the forgery (T, r, s) for a given message m, such that:

$$(T^{Tm})^z \equiv y^r r^s (mod\ p)$$

To find the forged signature, the attacker will do the following. He picks a random number $a \in Z_q^*$ and computes $r = y^a\ (mod\ p)$ and $T = r^d\ (mod\ p)$ using the arbitrary integer $d \in Z_q^*$. Then the confirmation protocol will be transfered to:

$$y^a = r = y^{r(Tdmz-s)^{-1}}\ (mod\ p).$$

Therefore $a = r(Tdmz - s)^{-1}\ (mod\ q)$. To solve for s, the attacker will need to calculate the following

$$s = Tdmz - a^{-1}r\ (mod\ q).$$

As a result (T, r, s) is the signature for message m because

$$(T^{Tm})^z \equiv r^{dTmz} \equiv y^{adTmz} \equiv y^{a(s+a^{-1}r)} \equiv r^s y^r\ (mod\ p).$$

Therefore, the total conversion scheme is insecure and forgery can be made easily after conversion takes place.

4.6 Future Work

There are a few things that one can do to improve the security of the convertible undeniable signature schemes or distributed undeniable signature schemes. Carpentieri, De Santis and Vaccaro [CSV93] suggested some relations between the size of the shares and the security of the secret sharing scheme. Furthermore, He and Kiesier [HK94] suggested that the El-Gamal signature schemes can be made more secure by using factoring as well as discrete logarithms in the scheme. To sum up, more work is needed to develop genuinely secure, efficient, convertible undeniable signature schemes.

Chapter 5

Description of the Implementation

As we mentioned in the beginning of this thesis, an implementation of Pedersen's protocol has been done in order to better study the topic. This chapter describes the environment for implementation and the tools used at different stages (for data format, calculations and communication). The details of the scheme will not be repeated here since we have discussed these in Chapter 4. The original code is available from the author of this thesis.

Briefly, recall the setup of the scheme. The recipient must ask for the signer's cooperation in order to verify the received signature. The verification and disavowal protocol proceed in the form of a challenge and response exchange between the signer and the verifier.

5.1 System Environment

The system is implemented on a Solaris 2.4 machine with Java-JDK 1.1, Perl 5 and ANCI C installed. We assume the reader is an average Unix system user and is familiar with its directory and file system structure.

The actual system environment is based on an interactive model which involves a signer, a prover and a verifier. Each sends information to the other or receives information from the other. Notice that a signer is not necessarily the same person as the prover. All parties are in different geographic locations and all need to perform modulo arithmetic and exponentiation over large integers.

We have used one machine to simulate the environment and the simulated environment is as follows. All parties are physically connected to one machine, so to achieve the distributed effect, the signer, the prover, and the verifier are placed in different directories. For interaction between two parties, two xterm windows are needed, one for each directory. Each xterm acts as a separate machine that is located in a different geographical location. Each party has a copy of all the necessary data files, hidden files and functions, and each has the responsibility of maintaining and protecting these files.

Data Format

Data files are stored in hexadecimal format because large integers are involved. Each file is distinguished by its file name. Since the implementation environment is Unix based, the data files must be world readable if they are public information, and readable by owner only if they are private information. All data files are ASCII text files and therefore can be read by Java, Perl and C programs.

Calculations Procedures

All calculations are done in the C language and with the C library multi-precision calculation utility $\langle mp.h \rangle$. The system checks the validity of a signature by exchanging challenges and responses between the prover and the verifier. All calculations deal mainly with large integers, therefore multiple precision calculation is essential.

Communications Procedures

Communications between two parties are established through Java (the Socket class) because of its ease of use and flexibility over many different platforms.

Languages Integration

The system works as follows. The Java code establishes the communication channel and acts as the core of the program. During program execution, the Java code calls Perl scripts which call the appropriate C calculation functions. Notice that Java provides a feature called **native method** which is able to integrate the C functions in the Java script. However, during this system implementation unresolvable technical difficulties were encountered as a system user, so Perl was used as an alternative to link the core program in Java and the C calculation functions instead of Java's native method. The C functions are called from the Perl scripts. Java calls the shell scripts and starts up a child process in the main program (the parent process). The result of the C function calculation is written in hexadecimal format to an ASCII text file which can be easily read by Perl, Java and C later if necessary.

Handling Multi-Callers

The verifier is required to interact with multiple provers. To speed up the procedures, the verifier will send out challenge sets to all potential provers. The verifier will take the first k provers who respond and carry out the subsequent protocol procedures with them. However, the verifier does not have control over who will respond and when they are going to respond. Collision of callers may occur. We used Multi-Thread in Java to handle this problem. A thread is like a sequential program; it has a beginning series of control statements and an end. It is not a program however, but rather it is a single sequential flow of control within a program. A thread must carve out its own resources (ie: execution stack and program counter in the program). It cannot run on its own and must be called by a master program. Multiple threads are made up of many single threads and each thread behaves in its own way and manages its resources without interfering with other threads. With the use of Multi-Thread in the implementation, any additional prover or caller that wishes to interact with the verifier is appended to a first-in-first-out queue. Without multi-thread, a late caller would receive a busy signal and be disconnected.

5.2 Limitations

1. The commitment protocol in the paper was not implemented. The reason is that it does not seem to be important for the simulation. However, it would play a significant role in a real life system.

2. Each variable is stored in a separate file. This results in many file opening and closing operations. One might wish to put all data in a configuration file such that file open and close operations are at a minimum.

3. The channel established by Java in this protocol is a public channel. One might consider securing the communication channel itself by using an encryption scheme.

Chapter 6

Conclusion

A complete and secure cryptographic system, requires a secure cryptographic function and a suitable key, a good key-management method, and a good authentication scheme. It does not matter how secure your encrypting function is, if the receiver cannot be sure of the genuineness of the message, the system will not be useful and practical. We are marching toward an electronic communication era, being able to perform basic debit and credit business transactions or exchange information of any kind reliably and securely is absolutely essential. We need to be able to trust the person at the other end of the cable, whom we have not even met.

This thesis gives an overview of a few modern authenticating schemes and their applications. In order to help the readers to better understand this thesis, we have introduced a few concepts from number theory. We have also included an extended overview of digital signatures, with special attention given to the undeniable signature schemes, its applications and attacks. It is believed that undeniable signatures have great potential to be merged into the electronic communication world. The scheme not only protect both sides, it secures the signer's ownership of the signature even further by requiring the signer's involvement in the signature verification process. In addition, the scheme provides the ability for the signer to deny a forgery.

For all the schemes discussed, to make the scheme more secure, it is necessary to increase the parameter size. By doing so, the actual processing/calculation time does not increase significantly but it increases the consumption of resources and time required tremendously for the attacker when he wishes to break the scheme or produce a forgery, see Table 2.1 in Chapter 2. This will buy the owner of the signature more time to find an appropriate countermeasure. On the other hand, since we are handling a large number of parameters, better algorithm is needed to bring the cost (or time) of producing and verifying the signature down. One must also have an efficient resource management technique to manage the CPU resources, memory and data. Different platform compatibility and software integration are very important as well.

Bibliography

[AGLL94] Derek Atkins, Micheal Graff, Arjen K Lenstra, and Paul C Leyland. The Magic Words are Squeamish Ossifrage. *Advances in Cryptology - ASIACRPT'94*, 917:263–277, 1994.

[BCD90] Joan Boyar, David Chaum, and Ivan Damgtård. Convertible Undeniable Signatures. *Advances in Cryptology, CRYPTO' 90*, 537:189–205, 1990.

[BOGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (extended abstract). *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 1–10, May 1988.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty Unconditionally Secure Protocols (extened abstract). *In Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 11–19, May 1988.

[Cha83] D. Chaum. Blind Signatures for Untraceable Payments. *Advances in Cryptology - CRYPTO'82*, pages 199–203, 1983.

[Cha90] David Chaum. Zero-Knowledge Undeniable Signatures. *Advances in Cryptology - EUROCRYPT' 90. Springer-Verlag*, 473:458–464, May 1990.

[Cha91] David Chaum. Some Weaknesses of Weaknesses of Undeniable Signatures. available from author. pages 1 – 3, 1991.

[Cha94] David Chaum. Designated Confirmer Signatures. *Advances in Cryptology - EUROCRYPT'94*, pages 86–91, 1994.

[CSV93] Marco Carpentieri, Alfredo De Santis, and Ugo Vaccaro. Size of Shares and Probability of Cheating in Threshold Schemes. *Advances in Cryptology - EUROCRYPT'93*, 765:118–125, 1993.

[Cv91] D. Chaum and E. vanHeijst. Group Signatures. *Advances in Cryptology - EUROCRPT'91*, pages 257–265, 1991.

[CvA89] David Chaum and Hans van Antwerpen. Undeniable Signatures. *Advances in Cryptology - CRYPTO' 89. Springer-Verlag*, 435:212–216, 1989.

[DF91] David S Dummit and Richard M Foote, editors. *Abstract Algebra*. Prentice Hall, Englewood Cliffs, N.J., 1991.

[DH76] Whitfield Diffie and Martin Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT.22(6):644–654, November 1976.

[DP96] Ivan Damgtård and Torben. Pedersen. New convertible Undeniable Signatures. *Advances in Cryptology, EUROCRYPT' 96*, 1070:372–386, 1996.

[DY91] Yvo Desmedt and Moti Yung. Weaknesses of Undeniable Signature Schemes. *Advances in Cryptology - EUROCRYPT' 91. Springer-Verlag*, 547:205–220, April 1991.

[EG85] T. El-Gamal. A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The Complexity of Interactive Proof-Systems. *SIAM Journal of Compututing*, 18(1):186–208, Feb 1989.

[Gol73] Larry Joel Goldstein, editor. *Abstract Algebra: A First Course*. Prentice Hall, Inc, Englewood Cliffs, N.J., 1973.

[HK94] J. He and T. Kiesier. Enhancing the Security of El-Gamal's Signatue Scheme. *IEE Proceedings of Computing and Digital Technique*, 141(4):249–252, July 1994.

[Jak94] Markus Jakobsson. Blackmailing using Undeniable Signature. *Advances in Cryptology - EUROCRYPT'94*, pages 425–427, 1994.

[LN83] Rudolf Lidl and Harald Niederreiter, editors. *Encyclopedia of Mathematics and its Applications vol.20 - Finite Field*. Addison-Wesley Publishing Company, USA, 1983.

[MPH96] Markus Michels, Holger Petersen, and Patrick Horster. Breaking and Repairing a Convertible Undeniable Signature Scheme. *The 3rd ACM Conference on Computer and Communications Security*, pages 148–152, March 1996.

[MvOV97] Alfred J Menezes, Paul C van Oorschot, and Scott A Vanstone, editors. *Handbook of Applied Cryptography*. CRC Press, Inc, New York, 1997.

[NZM91] Ivan Niven, Herbert S. Zuckerman, and Hugh L. Montgomery, editors. *An Introduction to The Theory of Numbers. 5th Ed.* John Wiley & Sons, Inc., New York, 1991.

[Ped91] Torben Pryds Pedersen. Distributed Provers with Applications to Undeniable Signatures. *Advances in Cryptology - EUROCRYPT' 91. Springer-Verlag*, 547:221–242, April 1991.

[PW66] Hiram Paley and Paul M Weichsel, editors. *A First Course In Abstract Algebra*. Holt, Rinehart and Winston. Inc., USA, 1966.

[Rom95] Steven Roman, editor. *Field Theory*. Springer-Verlag, New York, New York, 1995.

[RSA78] R.L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Sha79] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[Sti95] D. Stinson. *Cryptography: Theory and Practice*. CRC Press Inc., Boca Raton, 1995.