Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption) *

Yuliang Zheng

The Peninsula School of Computing and Information Technology
Monash University, McMahons Road, Frankston
Melbourne, VIC 3199, Australia
Email: yuliang.zheng@monash.edu.au
Phone: +61 3 9904 4196, Fax: +61 3 9904 4124

29 March 1999

*An extended abstract was presented at CRYPTO'97.

Contents

List of Figures

List of Tables

Abstract

Secure and authenticated message delivery/storage is one of the major aims of computer and communication security research. The current standard method to achieve this aim is "(digital) signature followed by encryption". In this paper, we address a question on the cost of secure and authenticated message delivery/storage, namely, *whether it is possible to transport/store messages of varying length in a secure and authenticated way with an expense less than that required by "signature followed by encryption".* This question has apparently never been addressed in the literature since the invention of public key cryptography. We then present a positive answer to the question. In particular, we discover a new cryptographic primitive termed as "signcryption" which *simultaneously* fulfills both the functions of digital signature and public key encryption in a logically single step, and with a cost *significantly* smaller than that required by "signature followed by encryption". For typical security parameters for high level security applications (size of public moduli = 1536 bits), signcryption costs 58% (50%, respectively) less in computation time and 85% (91%, respectively) less in message expansion than does "signature followed by encryption" based on the discrete logarithm problem (factorization problem, respectively). The saving in cost brought to society by this new technology is potentially huge, especially in the emerging era of electronic commerce in which the assurance of secure and authenticated transactions and communications is a key to its success as well as public acceptance.

Keywords

Authentication, Digital Signature, Encryption, Key Distribution, Secure Message Delivery/Storage, Public Key Cryptography, Security, Signcryption.

1 Introduction

To avoid forgery and ensure confidentiality of the contents of a letter, for centuries it has been a common practice for the originator of the letter to sign his/her name on it and then seal it in an envelope, before handing it over to a deliverer. This two-step process is graphically depicted in Figure 1.

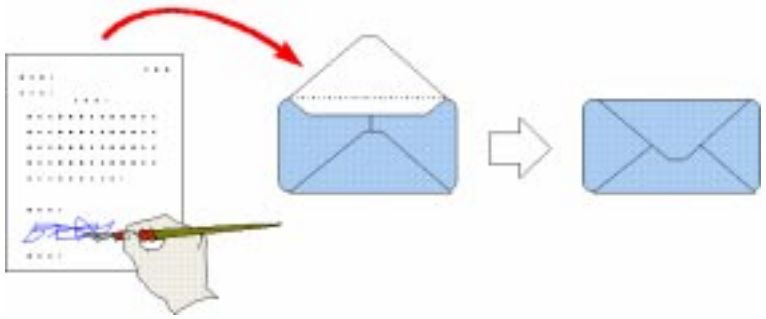

Figure 1: Centuries-Old "Signature-Then-Seal"

Public key cryptography discovered nearly two decades ago [12] has revolutionized the way for people to conduct secure and authenticated communications. It is now possible

for people who have never met before to communicate with one another in a secure and authenticated way over an open and insecure network such as Internet. In doing so the same two-step approach has been followed. Namely before a message is sent out, the sender of the message would sign it using a digital signature scheme, and then encrypt the message (and the signature) using a private key encryption algorithm under a randomly chosen message encryption key. The random message encryption key would then be encrypted using the recipient's public key. We call this two-step approach signature-then-encryption.

Signature generation and encryption consume machine cycles, and also introduce "expanded" bits to an original message. Symmetrically, a comparable amount of computation time is generally required for signature verification and decryption. Hence the cost of a cryptographic operation on a message is typically measured in the message expansion rate and the computational time invested by both the sender and the recipient. With the current standard signature-then-encryption approach, the cost for delivering a message in a secure and authenticated way is essentially the sum of the cost for digital signature and that for encryption.

In this paper, we address a question on the cost of secure and authenticated message delivery [1], namely, *whether it is possible to transfer a message of arbitrary length in a secure and authenticated way with an expense less than that required by signature-then-encryption.* This question seems to have never been addressed in the literature since the invention of public key cryptography. We then present a positive answer to the question. In particular, we discover a new cryptographic primitive termed as "signcryption" which *simultaneously* fulfills both the functions of digital signature and public key encryption in a logically single step, and with a cost *significantly* smaller than that required by signature-then-encryption. As shown in Tables 5 and 6, for the minimum security parameters recommended for the current practice (size of public moduli = 512 bits), signcryption costs 58% less in average computation time and 70% less in message expansion than does signature-then-encryption based on the discrete logarithm problem, while for security parameters recommended for long term security (size of public moduli = 1536 bits), it costs on average 50% less in computation time and 91% less in message expansion than does signature-then-encryption using the RSA cryptosystem. The saving in cost grows proportionally to the size of security parameters. Hence it will be more significant in the future when larger parameters are required to compensate theoretical and technological advances in cryptanalysis.

The organization of the rest of this paper is as follows: Section 2 surveys current standard signature-then-encryption methods for secure and authenticated message delivery. Our new approach "signcryption" is introduced in Section 3, where two concrete signcryption schemes are also demonstrated. A detailed comparison of the computational and communication cost of signcryption with that of signature-then-encryption is conducted in Section 4, and a comparison with regard to other aspects ranging from static key management, forward secrecy, past recovery to repudiation settlement is carried out in Section 5. Section 6 is devoted to signcryption for multiple recipients, while Section 7 suggests the significant potential of signcryption in reducing the cost of many widespread applications. This is followed by Section 8 where unforgeability, non-repudiation and confidentiality of the proposed signcryption schemes are discussed. Finally the paper is concluded with a few remarks in Section 9. A brief summary of notable public key digital signature and encryption schemes is provided in Appendix A.

[1] Methods for secure and authenticated communications can also be directly used for achieving confidentiality and authenticity of stored information.

2 The Traditional Signature-Then-Encryption Approach

As we mentioned earlier, public key cryptography invented by Diffie and Hellman [12] makes it a reality for one (1) to digitally sign a message, and (2) to send a message securely to another person with whom no common encryption key has been shared. In Appendix A, we provide a short summary on some of the most important public key digital signature/encryption schemes, these being RSA encryption and signature scheme, ElGamal encryption and signature scheme, and two signature schemes derived from ElGamal, namely Schnorr signature scheme and Digital Signature Standard (DSS).

In conjunction with other cryptographic technologies, such as secure and fast private key ciphers, one can now send electronic mail messages to his friends or partners in such a way that even an all-powerful attacker cannot learn the contents of a message or masquerade him in producing a valid digital signature. In doing so one is following essentially the same decades-old two-step approach, namely, one signs a message by attaching to it a digital signature by the use of a piece of secret information known only to himself, followed by encrypting the message using an encryption algorithm.

Software packages for secure and authenticated message delivery are now readily available. Some packages, such as PGP [50], are in the public domain and hence freely available. Regardless whether key-escrow, trusted third party (TTP) or schemes of a similar nature will be adopted by governments or not, cryptography is destined to find its widespread use in society. In particular, it can be envisaged that in the foreseeable future, security enhanced electronic transactions/message delivery, such as digital cash payments and EDI (electronic data interchange), will become an indispensable part of a person's private life and business activities. Therefore, analogously to fast "sorting" methods that are used by virtually all computer based systems, any technology that reduces the cost for secure and authenticated communications will bring potentially enormous saving in cost to society.

Currently, the standard approach for a user, say Alice, to send a secure and authenticated message to another user Bob is signature-then-encryption. The best example that follows the two-step approach is PEM, a standard for secure electronic mail on Internet [28]. The same approach is also followed by PGP. Part (c) of Figure 2 shows the format of a ciphertext in a signature-then-encryption based on discrete logarithm, while Part (b) of the figure shows the format based on RSA. The reader will notice that a notation $EXP = N_1 + N_2$ is used in the figure. N_1 indicates the number of modular exponentiations carried out by a sender, and N_2 indicates the number by a recipient.

To compare the efficiency of two different methods for secure and authenticated message delivery, we examine two types of "cost" involved: (1) computational cost, and (2) communication overhead (or storage overhead for stored messages). The *computational cost* indicates how much computational effort has to be invested both by the sender and the recipient of a message. We estimate the computational cost by counting the number of dominant operations involved. Typically these operations include private key encryption and decryption, hashing, modular addition, multiplication, division (inversion), and more importantly, exponentiation. In addition to computational cost, digital signature and encryption based on public key cryptography also require extra bits to be appended to a message. We call these extra "redundant" bits the *communication overhead* involved. We say that a message delivery method is superior to another if (the aggregated value of) the computational cost and communication overhead required by the former is less than that required by the latter.

The first part of Table 1 indicates the computational cost and communication overhead

of "RSA signature-then-RSA encryption" against that of "DSS-then-ElGamal encryption" and "Schnorr signature-then-ElGamal encryption". Note that, although not shown in the table, other combinations such as "Schnorr signature-then-RSA encryption" and "RSA signature-then-ElGamal encryption" may also be used in practice. As discussed in Appendix A, with the current state of the art, computing discrete logarithm on $GF(p)$ and factoring a composite n of the same size are equally difficult. This simplifies our comparison of the efficiency of a cryptographic scheme based on RSA against that based on discrete logarithm, as we can assume that the moduli n and p are of the same size.

It should be noted that to use RSA signature in a provably secure way, more extra computational effort (not counted in Table 1) has to be invested in signature generation and verification [5]. Similarly, to employ RSA and ElGamal encryptions in a provably secure fashion, more computational effort and communication overhead is required [49, 45, 4].

We close this section by examining the increasingly disproportionate cost for secure and authenticated message delivery in the currently standard signature-then-encryption approach, with an example text of 10000 bits (which corresponds roughly to a 15-line electronic mail message). For current and low security level applications, when RSA is used, the computational cost is centered around the execution of four (4) exponentiations modulo 512-bit integers, and the communication overhead is 1024 bits. When Schnorr signature and ElGamal encryption are used, the computational cost consists mainly of six (6) exponentiations modulo 512-bit integers, and the communication overhead is about 750 bits.

However, if the contents of the text are highly sensitive, or a text of the same length will be transmitted in 2010, then very large moduli, say of 5120 bits, might have to be employed. In such a situation, if RSA is used, four (4) exponentiations modulo (very large!) 5120-bit integers have to be invested in computation [2], and the communication overhead is 10240 bits, which is now longer than the original 10000-bit text ! If, instead, Schnorr signature and ElGamal encryption is used, then the computational cost is six (6) exponentiations modulo (again very large!) 5120-bit integers, and the communication overhead of about 5560 bits is more than half of the length of the original message. From this example, one can see that in the signature-then-encryption approach, the cost, especially communication overhead, for secure and authenticated message delivery, is becoming disproportionately large for future, or current but high-level security, applications. This observation serves as further justification on the necessity of inventing a new and more economical method for secure and authenticated message delivery.

3 Digital Signcryption — A More Economical Approach

With a hand-written letter to be delivered via ordinary postal services, the originator of the letter is physically (in time and space) bound to the signature-then-seal approach, although (un-realistically) with a highly personalized envelope the signing step may be omitted. Most people do not feel inconvenient with the signature-then-seal practice, as the time and cost involved is in most cases regarded as being marginal.

As discussed in the previous section, in the case of electronic message delivery, signing

[2] The number of bit operations required by exponentiation modulo an integer is a cubic function of the size of the modulo. Thus, if exponentiation modulo a 512-bit integer takes 0.1 second, then using the same computing device, exponentiation modulo a 5120-bit integer would theoretically take $(\frac{5120}{512})^3 \times 0.1 = 100$ seconds (although in practice a slightly smaller amount of time may suffice).

Various schemes	Computational cost	Communication overhead (in bits)
signature-then-encryption based on RSA	EXP=2, HASH=1, ENC=1 (EXP=2, HASH=1, DEC=1)	$\|n_a\| + \|n_b\|$
signature-then-encryption based on "DSS + ElGamal encryption"	EXP=3, MUL=1, DIV=1 ADD=1, HASH=1, ENC=1 (EXP=2.17, MUL=1, DIV=2 ADD=0, HASH=1, DEC=1)	$2\|q\| + \|p\|$
signature-then-encryption based on "Schnorr signature + ElGamal encryption"	EXP=3, MUL=1, DIV=0 ADD=1, HASH=1, ENC=1 (EXP=2.17, MUL=1, DIV=0 ADD=0, HASH=1, DEC=1)	$\|hash(\cdot)\| + \|q\| + \|p\|$
signcryption SCS1	EXP=1, MUL=0, DIV=1 ADD=1, HASH=2, ENC=1 (EXP=1.17, MUL=2, DIV=0 ADD=0, HASH=2, DEC=1)	$\|KH.(\cdot)\| + \|q\|$
signcryption SCS2	EXP=1, MUL=1, DIV=1 ADD=1, HASH=2, ENC=1 (EXP=1.17, MUL=2, DIV=0 ADD=0, HASH=2, DEC=1)	$\|KH.(\cdot)\| + \|q\|$

where
EXP = the number of modular exponentiations (a fractional number indicates an average cost),
MUL = the number of modular multiplications,
DIV = the number of modular division (inversion),
ADD = the number of modular addition or subtraction,
HASH = the number of one-way or keyed hash operations,
ENC = the number of encryptions using a private key cipher,
DEC = the number of decryptions using a private key cipher,
Parameters in the brackets indicate the number of operations involved in "decryption-then-verification" or "unsigncryption".

Table 1: Cost of Signature-Then-Encryption v.s. Cost of Signcryption

and encrypting a message consumes computational time and also results in added "redundant" bits or communication overhead. Unlike the case of a hand-written letter, however, the cost for signing and encrypting an electronic message can not be regarded as being marginal. The main reason is that digital signature and encryption employ computationally expensive operations including exponentiation modulo a large integer.

Over the past two decades since public key cryptography was invented, signature-then-encryption has been a standard method for one to deliver a secure and authenticated message of arbitrary length, and apparently no researcher has ever questioned whether it is absolutely necessary for one to use the sum of the cost for signature and that for encryption to achieve both contents confidentiality and origin authenticity.

Having posed a question that is of fundamental importance both from a theoretical and a practical point of view, we now proceed to tackle it. We will show how the question can be answered positively by the use of a new cryptographic primitive called "signcryption" whose definition follows.

Intuitively, a digital *signcryption* scheme is a cryptographic method that fulfills both the functions of secure encryption and digital signature, but *with a cost smaller than that required by signature-then-encryption*. Using the terminology in cryptography, it consists of a pair of (polynomial time) algorithms (S, U), where S is called the *signcryption algorithm*, while U the *unsigncryption algorithm*. S in general is probabilistic, while U is most likely to be deterministic. (S, U) satisfy the following conditions:

1. *Unique unsigncryptability* — Given a message m of arbitrary length, the algorithm S *signcrypts* m and outputs a *signcrypted text* c. On input c, the algorithm U *unsigncrypts* c and recovers the original message un-ambiguously.

2. *Security* — (S, U) fulfill, simultaneously, the properties of a secure encryption scheme and those of a secure digital signature scheme. These properties mainly include: confidentiality of message contents, unforgeability, and non-repudiation.

3. *Efficiency* — The computational cost, which includes the computational time involved both in signcryption and unsigncryption, and the communication overhead or added redundant bits, of the scheme is *smaller* than that required by the best currently known signature-then-encryption scheme with comparable parameters.

A direct consequence of having to satisfy both the second and third requirements is that "signcryption $\neq$ signature-then-encryption". These two requirements also justify our decision to introduce the new word *signcryption* which clearly indicates the ability for the new approach to achieve both the functions of digital signature and secure encryption in a logically single operation.

The rest of this section is devoted to seeking for concrete implementations of signcryption. We first identify two (types of) efficient ElGamal-based signature schemes. Then we show how to use a common property of these schemes to construct signcryption schemes.

3.1 Shortening ElGamal-Based Signatures

ElGamal digital signature scheme [14] involves two parameters public to all users:

1. p: a large prime.

2. g: an integer in $[1, \ldots, p-1]$ with order $p-1$ modulo p.

User Alice's private key is an integer x_a chosen randomly from $[1, \ldots, p-1]$ with $x_a \not| (p-1)$ (i.e., x_a does not divide $p-1$), and her public key is $y_a = g^{x_a} \bmod p$.

Alice's signature on a message m is composed of two numbers r and s:

$$\begin{aligned} r &= g^x \bmod p \\ s &= (hash(m) - x_a \cdot r)/x \bmod (p-1) \end{aligned}$$

where $hash$ is a one-way hash function, and x is chosen independently at random from $[1, \ldots, p-1]$ with $x \not| (p-1)$ every time a message is to be signed by Alice. Given (m, r, s), one can verify whether (r, s) is Alice's signature on m by checking whether $g^{hash(m)} = y_a^r \cdot r^s \bmod p$ is satisfied.

Since its publication in 1985, ElGamal signature has received extensive scrutiny by the research community. In addition, it has been generalized and adapted to numerous different forms (see for instance [41, 6, 33, 35] and especially [18] where an exhaustive survey of some 13000 ElGamal based signatures has been carried out.) For most ElGamal based schemes, the size of the signature (r, s) on a message is $2|p|$, $|q| + |p|$ or $2|q|$, where p is a large prime and q is a prime factor of $p-1$. The size of an ElGamal based signature, however, can be reduced by using a modified "seventh generalization" method discussed in [18]. In particular, we can change the calculations of r and s as follows:

1. Calculation of r — Set $r = hash(k, m)$, where $k = g^x \bmod q$ (or $k = g^x \bmod (p-1)$ if the original r is calculated modulo $(p-1)$), x is a random number from $[1, \ldots, q-1]$ (or from $[1, \ldots, p-1]$ with $x \not| (p-1)$), and $hash$ is a one-way hash function such as Secure Hash Standard or SHS [34] and HAVAL [48].

2. Calculation of s — For an *efficient* ElGamal based signature scheme, the calculation of (the original) s from x_a, x, r and optionally, $hash(m)$ involves only simple arithmetic operations, including modular addition, subtraction, multiplication and division. Here we assume that x_a is the private key of Alice the message originator. Her matching public key is $y_a = g^{x_a} \bmod p$. We can modify the calculation of s in the following way:

   (a) If $hash(m)$ is involved in the original s, we replace $hash(m)$ with a number 1, but leave r intact. The other way may also be used, namely we change r to 1 and then replace $hash(m)$ with r.

   (b) If s takes the form of $s = (\cdots)/x$, then changing it to $s = x/(\cdots)$ does not add additional computational cost to signature generation, but may reduce the cost for signature verification.

To verify whether (r, s) is Alice's signature on m, we recover $k = g^x \bmod p$ from r, s, g, p and y_a and then check whether $hash(k, m)$ is identical to r.

To illustrate how to shorten ElGamal based signatures, now we consider Digital Signature Standard (DSS). It should be stressed that many other ElGamal based signature schemes, in particular those defined on a sub-group of order q (see for example [18, 35]), can be shortened in the same way and are all equally good candidates for signcryption.

Table 2 shows two shortened versions of DSS, which are denoted by SDSS1 and SDSS2 respectively. Here are a few remarks on the table: (1) the first letter "S" in the name of a scheme stands for "shortened", (2) the parameters p, q and g are the same as those for DSS, (3) x is a random number from $[1, \ldots, q-1]$, x_a is Alice's private key and $y_a = g^{x_a} \bmod p$ is her matching public key, (4) $|t|$ denotes the size or length (in bits) of t, (5) the schemes have

the same signature size of $|hash(\cdot)| + |q|$, (6) SDSS1 is slightly more efficient than SDSS2 in signature generation, as the latter involves an extra modular multiplication.

Recently Pointcheval and Stern [37, 38] have proven that Schnorr signature is unforgeable by any adaptive attacker who is allowed to query Alice's signature generation algorithm with messages of his choice [16], in a model where the one-way hash function used in the signature scheme is assumed to behave like a random function (the random oracle model). The core idea behind the unforgeability proof by Pointcheval and Stern is based on an observation that the signature has been converted from a 3-move zero-knowledge (ZK) protocol (for proof of knowledge) with respect to a honest verifier. With such a signature scheme, unforgeability against a non-adaptive attacker who is not allowed to possess valid message-signature pairs follows from the soundness of the original protocol. Furthermore, as the protocol is zero-knowledge with respect to a honest verifier, the signature scheme converted from it can be efficiently simulated in the random oracle model. This implies that an adaptive attacker is not more powerful than a non-adaptive attacker in the random oracle model.

Turning our attention to SDSS1 and SDSS2, both can be viewed as being converted from a 3-move zero-knowledge protocol (for proof of knowledge) with respect to a honest verifier. Thus Pointcheval and Stern's technique is applicable also to SDSS1 and SDSS2. Summarizing the above discussions, both SDSS1 and SDSS2 are unforgeable by adaptive attackers, under the assumptions that (1) the one-way hash function behaves like a random function, and (2) discrete logarithm is hard (with respect to g chosen uniformly at random).

As a side note, both SDSS1 and SDSS2 are preferable to DSS in the sense that they admit a shorter signature and provable security (albeit under a strong assumption on hash functions).

Shortened schemes	Signature (r, s) on a message m	Verification of signature	Length of signature
SDSS1	$r = hash(g^x \bmod p, m)$ $s = x/(r + x_a) \bmod q$	$k = (y_a \cdot g^r)^s \bmod p$ check whether $hash(k, m) = r$	$\|hash(\cdot)\| + \|q\|$
SDSS2	$r = hash(g^x \bmod p, m)$ $s = x/(1 + x_a \cdot r) \bmod q$	$k = (g \cdot y_a^r)^s \bmod p$ check whether $hash(k, m) = r$	$\|hash(\cdot)\| + \|q\|$

p: a large prime (public to all),
q: a large prime factor of $p - 1$ (public to all),
g: an integer with order q modulo p chosen randomly from $[1, \ldots, p - 1]$ (public to all),
$hash$: a one-way hash function (public to all),
x: a number chosen uniformly at random from $[1, \ldots, q - 1]$,
x_a: Alice's private key, chosen uniformly at random from $[1, \ldots, q - 1]$,
y_a: Alice's public key ($y_a = g^{x_a} \bmod p$).

Table 2: Examples of Shortened and Efficient Signature Schemes

3.2 Implementing Signcryption with Shortened Signature

An interesting characteristic of a shortened ElGamal based signature scheme obtained in the method described above is that although $g^x \bmod p$ is not explicitly contained in a signature

Parameters public to all:
p — a large prime
q — a large prime factor of $p-1$
g — an integer with order q modulo p chosen randomly from $[1,\ldots,p-1]$
$hash$ — a one-way hash function whose output has, say, at least 128 bits
KH — a keyed one-way hash function
(E,D) — the encryption and decryption algorithms of a private key cipher
Alice's keys:
x_a — Alice's private key, chosen uniformly at random from $[1,\ldots,q-1]$
y_a — Alice's public key ($y_a = g^{x_a} \bmod p$)
Bob's keys:
x_b — Bob's private key, chosen uniformly at random from $[1,\ldots,q-1]$
y_b — Bob's public key ($y_b = g^{x_b} \bmod p$)

Table 3: Parameters for Signcryption

(r,s), it can be recovered from r, s and other public parameters. This motivates us to construct a signcryption scheme from a shortened signature scheme.

We exemplify our construction method using the two shortened signatures in Table 2. The same construction method is applicable to other shortened signature schemes based on ElGamal. As a side note, Schnorr's signature scheme, without being further shortened, can be used to construct a signcryption scheme which is slightly more advantageous in computation than other signcryption schemes from the view point of a message originator.

In describing our method, we will use E and D to denote the encryption and decryption algorithms of a private key cipher such as DES [32] and SPEED [46]. Encrypting a message m with a key k, typically in the cipher block chaining or CBC mode, is indicated by $E_k(m)$, while decrypting a ciphertext c with k is denoted by $D_k(c)$. In addition we use $KH_k(m)$ to denote hashing a message m with a keyed hash algorithm KH under a key k. An important property of a keyed hash function is that, just like a one-way hash function, it is computationally infeasible to find a pair of messages that are hashed to the same value (or collide with each other). This implies a weaker property that is sufficient for signcryption: given a message m_1, it is computationally intractable to find another message m_2 that collides with m_1. In [1] two methods for constructing a cryptographically strong keyed hash algorithm from a one-way hash algorithm have been demonstrated. For most practical applications, it suffices to define $KH_k(m) = hash(k,m)$, where $hash$ is a one-way hash algorithm.

Assume that Alice has chosen a private key x_a from $[1,\ldots,q-1]$, and made public her matching public key $y_a = g^{x_a} \bmod p$. Similarly, Bob's private key is x_b and his matching public key is $y_b = g^{x_b} \bmod p$. Relevant public and private parameters are summarized in Table 3.

The signcryption and unsigncryption algorithms constructed from a shortened signature are remarkably simple. For Alice to signcrypt a message m to be sent to Bob, she carries out the following operations:

Signcryption of m by Alice the Sender

1. Pick x uniformly at random from $[1, \ldots, q-1]$, and let $k = hash(y_b^x \bmod p)$. Split k into k_1 and k_2 of appropriate length.
2. $r = KH_{k_2}(m)$.
3. $s = x/(r + x_a) \bmod q$ if SDSS1 is used, or
   $s = x/(1 + x_a \cdot r) \bmod q$ if SDSS2 is used instead.
4. $c = E_{k_1}(m)$.
5. Send to Bob the signcrypted text (c, r, s).

Note that the output of the one-way hash function $hash$ used in defining $k = hash(y_b^x \bmod p)$ should be sufficiently long, say of at least 128 bits, which guarantees that both k_1 and k_2 have at least 64 bits. The main purpose of involving $hash$ in the derivation of k is to simplify our discussions on message confidentiality in Section 8. In practice, k can be defined in a more liberal way, such as $k = y_b^x \bmod p$ and $k = fd(y_b^x \bmod p)$, where fd denotes a folding operation.

The unsigncryption algorithm works by taking advantages of the property that $g^x \bmod p$ can be recovered by Bob from r, s, g, p. On receiving (c, r, s) from Alice, Bob unsigncrypts it as follows:

Unsigncryption of (c, r, s) by Bob the Recipient

1. Recover k from r, s, g, p, y_a and x_b:
   $k = hash((y_a \cdot g^r)^{s \cdot x_b} \bmod p)$ if SDSS1 is used, or
   $k = hash((g \cdot y_a^r)^{s \cdot x_b} \bmod p)$ if SDSS2 is used.
2. Split k into k_1 and k_2.
3. $m = D_{k_1}(c)$.
4. accept m as a valid message originated from Alice only if $KH_{k_2}(m)$ is identical to r.

The signcryption scheme that employs the shortened signature scheme SDSS1 is called SCS1, and the signcryption scheme that employs the shortened signature scheme SDSS2 is called SCS2. For convenience, the two signcryption schemes are detailed in Table 4.

The format of the signcrypted text of a message m is depicted in Part (a) of Figure 2. It should be pointed out that in some applications, part of a message m may not need to be encrypted and the creation of the signature part (r, s), especially r, may involve other data in addition to m. A similar observation can be made with the signature-then-encryption approach.

The two signcryption schemes share the same communication overhead $(|hash(\cdot)| + |q|)$. Although SCS1 involves one less modular multiplication in signcryption than does SCS2, both have a similar computational cost for unsigncryption. Table 1 shows the number of

dominant operations involved in the signcryption schemes, against those for two representative signature-then-encryption schemes.

Finally, we note that signature schemes with message recovery proposed in [35] do not provide message confidentiality, and hence they are not signcryption schemes. Although our concrete signcryption schemes shown above bear some resemblance to "authenticated encryption" discussed in [49, 19, 35, 27], the former differentiate itself from the latter in that it addresses all the following three aspects: (1) to minimize both the number of modular exponentiations and the overhead in communication, (2) to provide a explicit set of repudiation settlement procedures, and (3) to be able to handle messages of arbitrary length.

3.3 Name Binding

In some applications such as electronic cash payment protocols, the names/identifiers of participants involved may need to be tightly bound to messages exchanged. This can be achieved by explicitly including their names into the contents of a message. Alternatively, data related to participants' names, such as public keys and their certificates, may be included in the computation of r in the signcryption algorithm. Namely, we may define

$$r = KH_{k_2}(m, \mathit{bind\_info})$$

where *bind_info* may contain, among other data, the public keys or public key certificates of both Alice the sender and Bob the recipient. The corresponding unsigncryption algorithm can be modified accordingly. Compared with an exponentiation modulo a large integer, the extra computational cost invested in hashing *bind_info* is negligible.

Involving the recipient's public key y_b or his public key certificate in the computation of r is particularly important. To see this point, let (c, r, s) be a signcrypted text of m (from Alice to Bob) where the computation of r does not involve identification information on Bob the recipient, and consider a situation where m represents a commitment/statement for Alice to transfer a certain amount of money (or valuable goods) to the recipient of the message. Assume that a third participant Cathy has x_c as her private key and $y_c = g^{x_c} \bmod p$ as her matching public key. Furthermore, assume that Bob and Cathy are a pair of collusive and dishonest friends, and that their private keys are related by $x_b = w \cdot x_c \bmod q$. Then a modified text (c, r, s^*), where $s^* = w \cdot s \bmod q$, may represent a perfectly *valid* message from Alice to Cathy, and hence it might be obligatory for Alice to pay the same of amount money to both Bob and Cathy ! Clearly, such a collusive attack can be easily thwarted by involving information on Bob's identity in the creation of r, say by defining $r = KH_{k_2}(m, y_b, etc)$.

3.4 Extensions

Signcryption schemes can also be derived from ElGamal-based signature schemes built on other versions of the discrete logarithm problem such as that on elliptic curves [23]. In addition, Lenstra's new method for constructing sub-groups based on cyclotomic polynomials [24] can also be used to implement signcryption even more efficiently.

There is also a marginally less efficient version of signcryption schemes in which Alice's private key x_a participates in the computation of k. Taking SCS1 as an example, we can redefine the computation of k by Alice in the signcryption algorithm as $k = hash(y_b^{x+x_a} \bmod p)$, and correspondingly, the computation of k by Bob in the unsigncryption algorithm as $k = hash((y_a^{(s+1)\cdot x_b}) \cdot (g^{r \cdot s \cdot x_b}) \bmod p)$.

Finally, we point out that the one-way hash function used in deriving k, namely $hash$ in $k = hash(y_b^x \bmod p)$ and $k = hash(y_b^{x+x_a} \bmod p)$, can be replaced by a hash function h chosen uniformly at random from a class of $(1/2^{|k|} + 1/2^{|p|})$-almost strongly universal hash functions [44]. The Leftover Hash Lemma of [20] ensures that k is statistically very close to a truly random key. Note that h needs to be sent to a receiver as part of a signcrypted text, which slightly increases the communication overhead.

3.5 Working with Signature-Only and Encryption-Only Modes

Not all messages require both confidentiality and integrity. Some messages may need to be signed only, while others may need to be encrypted only. For the two digital signcryption schemes SCS1 and SCS2, when a message is sent in clear, they degenerate to signature schemes with direct verifiability by the recipient only. As will be argued in Section 5.4, limiting direct verifiability to the recipient only still preserves non-repudiation, and may represent an advantage for some applications where the mere fact that a message is originated from Alice needs to be kept secret. Furthermore, if Alice uses g instead of Bob's public key y_b in the calculation of k, the schemes becomes corresponding shortened ElGamal based signature schemes with universal verifiability.

To work with the encryption-only mode,

In an application where a message requires confidentiality only, one may either switch to a public key encryption scheme such as the ElGamal scheme, or stick to a signcryption scheme. The latter is more efficient than the former, even though authenticity may not be a concern in such an application.

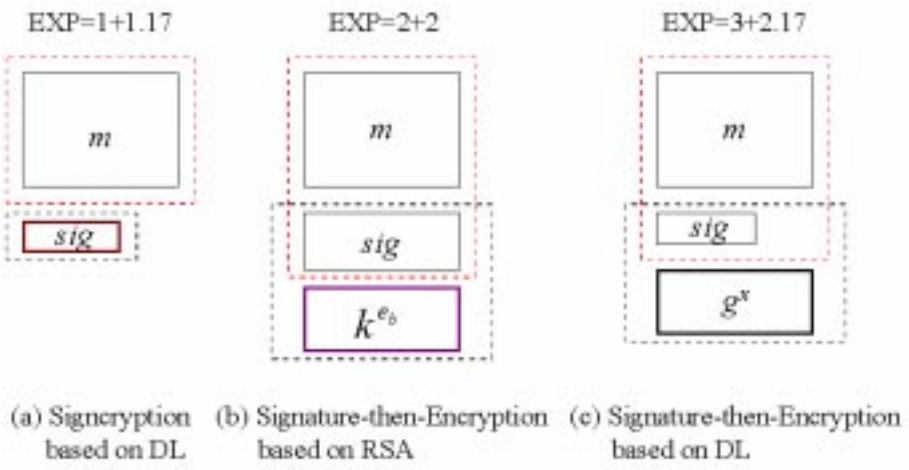

Figure 2: Output Formats of Signcryption and Signature-then-Encryption for a Single Recipient

4 Cost of Signcryption v.s. Cost of Signature-Then-Encryption

The most significant advantage of signcryption over signature-then-encryption lies in the dramatic reduction of computational cost and communication overhead which can be symbolized by the following inequality:

Signcryption schemes	Signcrypted text (c, r, s) of a message m (by Alice)	Recovery of $k = hash(g^{x \cdot x_b} \bmod p)$ (by Bob)
SCS1 (from SDSS1)	$c = E_{k_1}(m)$ $r = KH_{k_2}(m)$ $s = x/(r + x_a) \bmod q$	$k = hash((y_a \cdot g^r)^{s \cdot x_b} \bmod p)$
SCS2 (from SDSS2)	$c = E_{k_1}(m)$ $r = KH_{k_2}(m)$ $s = x/(1 + x_a \cdot r) \bmod q$	$k = hash((g \cdot y_a^r)^{s \cdot x_b} \bmod p)$

On Alice's side, x is a number chosen independently at random from $[1, \ldots, q-1]$, k is obtained by $k = hash(y_b^x \bmod p)$, k_1 and k_2 are the left and right halves of k respectively. Bob can recover k from r, s, g, p, y_a and x_b, and decipher c by $m = D_{k_1}(c)$. He accepts m as a valid message from Alice only if r can be reconstructed from $KH_{k_2}(m)$.

Table 4: Two Signcryption Schemes

$$\text{Cost(signcryption)} < \text{Cost(signature)} + \text{Cost(encryption)}.$$

With SCS1 and SCS2, this advantage is identifiable in Table 1. The purpose of this section is to examine the advantage in more detail. The necessity of such an examination is justified by the facts that the computational cost of modular exponentiation is mainly determined by the size of an exponent, and that RSA and discrete logarithm based public key cryptosystems normally employ exponents that are quite different in size. For readers who are not interested in technical details in the comparison, Table 5 summarizes the advantage of SCS1 and SCS2 over discrete logarithm based signature-then-encryption, while Table 6 summarizes that over RSA based signature-then-encryption.

4.1 A Comparison with Signature-Then-Encryption Using Schnorr Signature and ElGamal Encryption

4.1.1 Saving in computational cost

With the signature-then-encryption based on Schnorr signature and ElGamal encryption, the number of modular exponentiations is three, both for the process of signature-then-encryption and that of decryption-then-verification.

Among the three modular exponentiations for decryption-then-verification, two are used in verifying Schnorr signature. More specifically, these two exponentiations are spent in computing $g^s \cdot y_a^r \bmod p$. Using a technique for fast computation of the product of multiple exponentials with the same modulo which has been attributed to Shamir (see Appendix B), $g^s \cdot y_a^r \bmod p$ can be computed, on average, in $(1 + 3/4)|q|$ modular multiplications. Since a modular exponentiation can be completed, on average, in about $1.5|q|$ modular multiplications when using the classical "square-and-multiply" method, $(1 + 3/4)|q|$ modular multiplications is computationally equivalent to 1.17 modular exponentiations. Thus with "square-and-multiply" and Shamir's technique, the number of modular exponentiations involved in decryption-then-verification can be reduced from 3 to 2.17. The same reduction techniques, however, cannot be applied to the sender's computation. Consequently, the combined computational cost of the sender and the recipient is 5.17 modular exponentiations.

security parameters			saving	saving in
$\|p\|$	$\|q\|$	$\|KH.(\cdot)\|$	average comp. cost	comm. overhead
512	144	72	58%	70.3%
768	152	80	58%	76.8%
1024	160	80	58%	81.0%
1280	168	88	58%	83.3%
1536	176	88	58%	85.3%
1792	184	96	58%	86.5%
2048	192	96	58%	87.7%
2560	208	104	58%	89.1%
3072	224	112	58%	90.1%
4096	256	128	58%	91.0%
5120	288	144	58%	92.0%
8192	320	160	58%	94.0%
10240	320	160	58%	96.0%

saving in average comp. cost $= \frac{(5.17-2.17) \text{ modular exponentiations}}{5.17 \text{ modular exponentiations}} = 58\%$

saving in comm. cost $= \frac{|hash(\cdot)|+|q|+|p|-(|KH.(\cdot)|+|q|)}{|hash(\cdot)|+|q|+|p|}$

Table 5: Saving of Signcryption over Signature-Then-Encryption Using Schnorr Signature and ElGamal Encryption

security parameters			advantage in	advantage in
$\|p\|(=\|n_a\|=\|n_b\|)$	$\|q\|$	$\|KH.(\cdot)\|$	average comp. cost	comm. overhead
512	144	72	0%	78.9%
768	152	80	14.2%	84.9%
1024	160	80	32.3%	88.3%
1280	168	88	43.1%	90.0%
1536	176	88	50.3%	91.4%
1792	184	96	55.5%	93.1%
2048	192	96	59.4%	93.0%
2560	208	104	64.8%	94.0%
3072	224	112	68.4%	94.5%
4096	256	128	72.9%	95.0%
5120	288	144	75.6%	96.0%
8192	320	160	83.1%	97.0%
10240	320	160	86.5%	98.0%

advantage in average comp. cost $= \frac{0.375(|n_a|+|n_b|)-3.25|q|}{0.375(|n_a|+|n_b|)}$

advantage in comm. cost $= \frac{|n_a|+|n_b|-(|KH.(\cdot)|+|q|)}{|n_a|+|n_b|}$

Table 6: Advantage of Signcryption over RSA based Signature-Then-Encryption with *Small Public Exponents*

In contrast, with SCS1 and SCS2, the number of modular exponentiations is one for the process of signcryption and two for that of unsigncryption respectively. Since Shamir's technique can also be used in unsigncryption, the computational cost of unsigncryption is about 1.17 modular exponentiations. The total average computational cost for signcryption is therefore 2.17 modular exponentiations. This represents a

$$\frac{5.17 - 2.17}{5.17} = 58\%$$

reduction in average computational cost.

4.1.2 Saving in communication overhead

The communication overhead measured in bits is $|hash(\cdot)| + |q| + |p|$ for the signature-then-encryption based on Schnorr signature and ElGamal encryption, and $|KH.(\cdot)| + |q|$ for the two signcryption schemes SCS1 and SCS2, where $|x|$ refers to the size of a binary string, KH is a keyed hash function and $hash$ is a one-way hash function (used in Schnorr signature, but not the one used in signcryption). Hence the saving in communication overhead is

$$\frac{|hash(\cdot)| + |q| + |p| - (|KH.(\cdot)| + |q|)}{|hash(\cdot)| + |q| + |p|}$$

Assuming that the one-way hash function $hash$ used in the signature-then-encryption scheme and the keyed hash function KH used in the signcryption scheme share the same output length, the reduction in communication overhead is $|p|$. For the minimum security parameters recommended for use in current practice: $|KH.(\cdot)| = |hash(\cdot)| = 72$, $|q| = 144$ and $|p| = 512$, the numerical value for the saving is 70.3%. One can see that the longer the prime p, the larger the saving.

Comparing Table 2 (shortened signature schemes) with Table 4 (signcryption schemes), we can see that

$$\text{Cost(signcryption)} \approx \text{Cost(shortened signature)}$$

for the two signcryption schemes SCS1 and SCS2.

4.2 A Comparison with Signature-Then-Encryption Using RSA

4.2.1 Advantage in computational cost

With RSA, it is a common practice to employ a relatively small public exponent e for encryption or signature verification, although cautions should be taken in light of recent progress in cryptanalysis against RSA with an small exponent (see for example [10, 9]). Therefore the main computational cost is in decryption or signature generation which generally involves a modular exponentiation with a *full size* exponent d, which takes on average 1.5ℓ modular multiplications using the "square-and-multiply" method, where ℓ indicates the size of the RSA composite involved. With the help of the Chinese Remainder Theorem, the computational expense for RSA decryption can be reduced, theoretically, to a quarter of the expense with a full size exponent, although in practice it is more realistic to expect the factor to be between 1/4 and 1/3. To simplify our discussion, we assume that the maximum speedup is achievable, namely the average computational cost for RSA decryption is $\frac{1.5}{4}\ell = 0.375\ell$ modular multiplications.

For the signature-then-encryption based on RSA, four (4) modular exponentiations are required (two with public exponents and the other two with private exponents). Assuming small public exponents are employed, the computational cost will be dominated by the two modular exponentiations with full size private exponents. When the Chinese Remainder Theorem is used, this cost is on average $0.375(|n_a| + |n_b|)$ modular multiplications, where n_a and n_b are the RSA composites generated by Alice and Bob respectively.

As discussed earlier, the two signcryption schemes SCS1 and SCS2 both involve, on average, 2.17 modular exponentiations, or equivalently $3.25|q|$ modular multiplications, assuming the "square-and-multiply" method and Shamir's technique for fast computation of the product of exponentials are used. This shows that the signcryption schemes represent an advantage of

$$\frac{0.375(|n_a| + |n_b|) - 3.25|q|}{0.375(|n_a| + |n_b|)}$$

in average computational cost over the RSA based signature-then-encryption. For small security parameters, the advantage is less significant. This situation, however, changes dramatically for large security parameters: consider $|n_a| = |n_b| = |p| = 1536$ and $|q| = 176$ which are recommended to be used for long term (say more than 20 years) security, the signcryption schemes show a 50.3% saving in computation, when compared with the signature-then-encryption based on RSA.

The advantage of the signcryption schemes in computational cost will be more visible, should large public exponents be used in RSA.

4.2.2 Advantage in communication overhead

The signature-then-encryption based on RSA expands each message by a factor of $|n_a|+|n_b|$ bits, which is multiple times as large as the communication overhead $|KH.(\cdot)| + |q|$ of the two signcryption schemes SCS1 and SCS2. Numerically, the advantage or saving of the signcryption schemes in communication overhead over the signature-then-encryption based on RSA is as follows:

$$\frac{|n_a| + |n_b| - (|KH.(\cdot)| + |q|)}{|n_a| + |n_b|}$$

For $|n_a| = |n_b| = 1536$, $|q| = 176$ and $|KH.(\cdot)| = 88$, the advantage is 91.4%. The longer the composites n_a and n_b, the larger the saving by signcryption.

Note that we have chosen not to compare the signcryption schemes with unbalanced RSA recently proposed by Shamir [43]. The main reason is that while the new variant of RSA is attractive in terms of its computational efficiency, its security has yet to be further scrutinized by the research community.

4.3 Remarks on the Comparison

The above comparison between signcryption and signature-then-encryption should only be interpreted as a rough indicator for the relative efficiency of the two different paradigms. The comparison has been based on the assumption that only basic modular exponentiation techniques are used, these being the "square-and-multiply" method, Shamir's method for fast evaluation of the product of several exponentials with the same modulo, and in the case of RSA, the Chinese Remainder Theorem.

Instead of "square-and-multiply", other fast exponentiation methods such as sliding-window exponentiation may be used. For the RSA cryptosystem, signature generation and

decryption can be sped up by adopting fast algorithms for fixed-exponent exponentiation. A notable example of such algorithms is addition chain exponentiation. On the other hand, discrete logarithm based cryptosystems, including the signcryption schemes proposed in this work, can be made more efficient using a number of strategies. Examples of these strategies are (1) elliptic curves, (2) Lenstra's new sub-groups based on cyclotomic polynomials [24], and (3) fast algorithms for fixed-base exponentiation. When all these techniques are used, the resultant comparative indicator for the relative efficiency of signcryption and signature-then-encryption may oscillate around the values shown above.

A final remark is that our comparison has been carried out in terms of the *absolute* number of bits and computational operations that can be saved by signcryption. Comparisons can also be made in terms of savings *relative* to the size of an entire data packet which may include a (scrambled) message, the identifiers of a sender and a recipient, signatures, public key certificates, time-stamps, and so on. A problem with relative comparisons is that their indicators decrease when the size of a message increases, which may render obscure the significant advantages of signcryption over signature-then-encryption.

4.4 How the Parameters are Chosen

Advances in fast computers help an attacker in increasing his capability to break a cryptosystem. To compensate this, larger security parameters, including $|n_a|$, $|n_b|$, $|p|$, $|q|$ and $|KH.(\cdot)|$ must be used in the future. From an analysis by Odlyzko [36] on the hardness of discrete logarithm, one can see that unless there is an algorithmic breakthrough in solving the factorization or discrete logarithm problem, $|q|$ and $|KH.(\cdot)|$ can be increased at a smaller pace than can $|n_a|$, $|n_b|$ and $|p|$. Thus, as shown in Tables 5 and 6, the saving or advantage in computational cost and communication overhead by signcryption will be more significant in the future when larger parameters must be used.

The selection of security parameters $|p|$, $|q|$, $|n_a|$ and $|n_a|$ in Tables 5 and 6, has been partially based on recommendations made in [36]. The parameter values in the tables, however, are indicative only, and can be determined flexibly in practice. We also note that choosing $|KH.(\cdot)| \approx |q|/2$ is due to the fact that using Shank's baby-step-giant-step or Pollard's rho method, the complexity of computing discrete logarithms in a sub-group of order q is $O(\sqrt{q})$ (see [25]). Hence choosing $|KH.(\cdot)| \approx |q|/2$ will minimize the communication overhead of the signcryption schemes SCS1 and SCS2. Alternatively, one may decide to choose $KH.(\cdot) \in [1, \ldots, q-1]$ which can be achieved by setting $|KH.(\cdot)| = |q| - 1$. This will not affect the computational advantage of the signcryption schemes, but slightly increase their communication overhead.

5 More on Signcryption v.s. Signature-then-Encryption

In the previous section we concentrated on saving in computation and communication offered by signcryption schemes. A natural question is why signcryption schemes can achieve the savings. To search for a possible answer to the question, we have further compared signcryption with "signature-then-encryption" and "signature-then-encryption-with-a-static-key", in terms of key management, forward secrecy, past recovery, repudiation settlement and users' "community" or world orientation.

We use the following encryption algorithm as an example of "signature-then-encryption-with-a-static-key": (c, r, s) where $c = E_k(m)$, $k = KH_{SV}(r, s)$, SV is a static key shared between Alice and Bob, and (r, s) is Schnorr's signature on m. Typical examples of SV

include (a) a pre-shared random string between Alice and Bob, (b) the Diffie-Hellman key $g^{x_a x_b} \bmod p$, and (c) a shared key generated by an identity-based key establishment scheme such as the key pre-distribution scheme [29].

5.1 Static Key Management

We focus narrowly on the way a static key SV between two users is generated and stored. If SV is defined as a pre-shared random string between Alice and Bob, then first of all there is a cost associated with distributing the key before a communication session takes place. In addition, storing it in secure memory incurs a burden to a user, especially when the number of keys to be kept securely is large. (These problems contributed to the motivation for Diffie and Hellman to discover public key cryptography [12].)

On the other hand, if SV is defined as the Diffie-Hellman key $g^{x_a x_b} \bmod p$, then prior to using the value, a modular exponentiation is required on both Alice and Bob's sides. Alice and Bob may save the exponentiation by computing $SV = g^{x_a x_b} \bmod p$ and storing it in secure memory. But then they face the same problem with secure storage as that for a pre-shared random string. Similar discussions apply to the case where SV is defined as a shared key using the key pre-distribution scheme.

Now it becomes clear that static key generation/storage is a problem for "signature-then-encryption-with-a-static-key", but not for signcryption or "signature-then-encryption".

5.2 Forward Secrecy

A cryptographic primitive or protocol provides forward secrecy with respect to a long term private key if compromise of the private key does not result in compromise of security of previously communicated or stored messages.

With "signature-then-encryption", since different keys are involved in signature generation and public key encryption, forward secrecy is in general guaranteed with respect to Alice's long term private key. (Nevertheless, loss of Alice's private key renders her signature forgeable.) In contrast, with the signcryption schemes, it is easy to see that knowing Alice's private key alone is sufficient to recover the original message of a signcrypted text. Thus no forward secrecy is provided by the signcryption schemes with respect to Alice's private key. A similar observation applies to "signature-then-encryption-with-a-static-key" with respect to Alice's shared static key.

Forward secrecy has been regarded particularly important for session key establishment [13]. However, to fully understand its implications to practical security solutions, we should identify (1) how one's long term private key may be compromised, (2) how often it may happen, and (3) what can be done to reduce the risks of a long key being compromised. In addition, the cost involved in achieving forward secrecy is also an important factor that should be taken into consideration.

There are mainly three causes for a long term private key being compromised: (1) the underlying computational problems are broken; (2) a user accidentally loses the key; (3) an attacker breaks into the physical or logical location where the key is stored.

As a public key cryptosystem always relies on the (assumed) difficulty of certain computational problems, breaking the underlying problems renders the system insecure and useless. Assuming that solving underlying computational problems is infeasible, an attacker would most likely try to steal a user's long term key through such a means as physical break-in.

To reduce the impact of signcryption schemes' lack of forward secrecy on certain security applications, one may suggest users change their long term private keys regularly. In addition, a user may also use techniques in secret sharing [42] to split a long term private key into a number of shares, and keep each share in a separate logical or physical location. This would significantly reduce the risk of a long term key being compromised, as an attacker now faces a difficult task to penetrate in a larger-than-a-threshold number of locations in a limited period of time.

5.3 Past Recovery

Consider the following scenario: Alice signs and encrypts a message and sends it to Bob. A while later, she finds that she wants to use the contents of the message again.

To satisfy Alice's requirement, her electronic mail system has to store some data related to the message sent. And depending on cryptographic algorithms used, Alice's electronic mail system may either (1) keep a copy of the signed and encrypted message as evidence of transmission, or (2) in addition to the above copy, keep a copy of the original message, either in clear or encrypted form.

A cryptographic algorithm or protocol is said to provide a past recovery ability if Alice can recover the message from the signed and encrypted message alone using her private key.

Obviously a cryptographic algorithm or protocol provides past recovery if and only if it does *not* provide forward secrecy with respect to Alice the sender's long term private key.

Thus both signcryption and "signature-then-encryption-with-a-static-key" provide past recovery, while "signature-then-encryption" does not.

In terms of past recovery, one may view "signature-then-encryption" as an information "black hole" with respect to Alice the sender: whatsoever Alice drops in the "black hole" will never be retrieval to her, unless a separate copy is properly kept. Therefore signcryption schemes are more economical with regard to secure and authenticated transport of large data files. It is even more so when Alice has to broadcast the same message to a large number of recipients. (See also Section 6 for more discussions on broadcasting).

5.4 Repudiation Settlement

Now we turn to the problem of how to handle repudiation. With signature-then-encryption, if Alice denies the fact that she is the originator of a message, all Bob has to do is to decrypt the ciphertext and present to a judge (say Julie) the message together with its associated signature by Alice, based on which the judge will be able to settle a dispute.

With digital signcryption, however, the verifiability of a signcryption is in normal situations limited to Bob the recipient, as his private key is required for unsigncryption. Now consider a situation where Alice attempts to deny the fact that she has signcrypted and sent to Bob a message m. Similarly to signature-then-encryption, Bob would first unsigncrypt the signcrypted text, and then present the following data items to a judge (Julie): q, p, g, y_a, y_b, m, r, and s. One can immediately see that the judge cannot make a decision using these data alone. To solve this problem, Bob and the judge have to engage in an interactive zero-knowledge proof/argument protocol. Details will be discussed in Section 8.3.

At the first sight, the need for an interactive repudiation settlement procedure between Bob and the judge may be seen as a drawback of signcryption. Here we argue that interactive repudiation settlement will not pose any problem in practice and hence should not be an obstacle to practical applications of signcryption. In the real life, a message sent

to Bob in a secure and authenticated way is meant to be readable by Bob only. Thus if there is no dispute between Alice and Bob, direct verifiability by Bob only is precisely what the two users want. In other words, in normal situations where no disputes between Alice and Bob occur, the full power of universal verifiability provided by digital signature is never needed. (For a similar reason, traditionally one uses signature-then-encryption, rather than encryption-then-signature. See also [8] for potential risks of forgeability accompanying encryption-then-signature.) In a situation where repudiation does occur, interactions between Bob and a judge would follow. This is very similar to a dispute on repudiation in the real world, say between a complainant (Bob) and a defendant (Alice), where the process for a judge to resolve the dispute requires in general interactions between the judge and the complainant, and furthermore between the judge and an expert in hand-written signature identification, as the former may rely on advice from the latter in correctly deciding the origin of a message. The interactions among the judge, Bob the recipient and the expert in hand-written signature identification could be time-consuming and also costly.

5.5 "Community" or World Orientation

With the signcryption schemes, both Alice and Bob have to use the same p and g. So they basically belong to the same "community" defined by p and g. Such a restriction does not apply to "signature-then-encryption".

Similar restrictions apply to "signature-then-encryption-with-a-static-key" where the static key is derived from the Diffie-Hellman key $g^{x_a x_b} \bmod p$, or a key pre-distribution scheme [29]. Such restrictions seem to be inherent with cryptographic protocols based on the Diffie-Hellman public key cryptosystem [12]. A recent example of such protocols is an Internet key agreement protocol based on ISAKMP and Oakley [17].

In the case where a static key is a pre-shared random string between Alice and Bob, whether or not Alice and Bob belong to the same "community" will be determined by the underlying protocol for distributing the pre-shared random string.

In theory, the requirement that both Alice and Bob belong to the same "community" does limit the number of users with whom Alice can communicate using a signcryption scheme. In reality, however, all users belong to several "communities", and they tend to communicate more with users in the same group than with outsiders: users (including banks and individuals) of a certain type of digital cash payment system, employees of a company and citizens of a country, to name a few. Therefore the "community" oriented nature of signcryption schemes may not bring much inconvenience to their use in practice.

Table 7 summarizes all the comparisons we have carried out in this section.

5.6 Why Can Signcryption Save ?

Now we come back to the question of why signcryption has a cost similar to that of Schnorr signature. At the first sight, one might think that a possible answer would lie in the fact that with signcryption, forward secrecy is lost with respect to the sender's long term private key. However, signcryption offers past recovery which cannot be achieved by "signature-then-encryption". In other words, past recovery is not something for free. So perhaps loss of forward secrecy does not directly contribute to the low cost of signcryption. Rather, one may consider that the cost for forward secrecy has been somehow transformed to achieve past recovery.

It seems more likely that the loss of non-interactive repudiation settlement, together

Various Dimensions	Signcryption	Signature-then-Encryption with a Static Key	Signature-then-Encryption
Cost in Comp. & Comm.	$\approx$ Cost(signature)	$\approx$ Cost(signature)	Cost(signature) + Cost(encryption)
Static key Management	N/A	Distribution, Derivation, Secure storage	N/A
Forward Secrecy	No	No	Yes
Past Recovery	Yes	Yes	No
Repudiation Settlement	Interactive	Non-interactive	Non-interactive
World Orientation	No	Yes & No (see Section 5.5)	Yes

Table 7: Other Aspects of Signcryption v.s. Signature-then-Encryption

with the fact that users are all confined to the same "community" defined by p and g, has contributed to the low cost of signcryption.

6 Signcryption for Multiple Recipients

So far we have only discussed the case of a single recipient. In practice, broadcasting a message to multiple users in a secure and authenticated manner is an important facility for a group of people who are jointly working on the same project to communicate with one another. In this scenario, a message is broadcast through a so-called multi-cast channel, one of whose properties is that all recipients will receive an identical copy of a broadcast message. Major concerns with broadcasting to multiple recipients include security, unforgeability, non-repudiation and consistency of a message. Here consistency refers to that all recipients recover an identical message from their copies of a broadcast message, and its aim is to prevent a particular recipient from being excluded from the group by a dishonest message originator.

With the traditional signature-then-encryption, the standard practice has been to encrypt the message-encryption key using each recipient's public key and attach the resulting ciphertext to the signed and also encrypted message. RFC1421 [28] details a standard based on RSA. A similar scheme for multiple recipients can be defined using cryptographic schemes based on the discrete logarithm problem, such as "Schnorr signature-then-ElGamal encryption". In Parts (b) and (c) of Figure 3, the format of a ciphertext for multiple recipients in a Discrete Logarithm based approach is shown against that in an RSA based approach.

Now we show that a signcryption scheme can be easily adapted to one for multiple recipients. We assume that there are t recipients R_1, R_2, ..., R_t. The private key of a recipient R_i is a number x_i chosen uniformly and independently at random from $[1, \ldots, q-1]$, and his matching public key is $y_i = g^{x_i} \bmod p$.

Table 8 details how to modify SCS1 described in Table 4 into a multi-recipient signcryption scheme which we call SCS1M. SCS2M is constructed similarly from SCS2, and hence not shown in the Table. See also Part (a) of Figure 3 for the format of the signcrypted text of a message for multiple recipients. The basic idea is to use two types of keys: the

first type consists of only a single randomly chosen key (a message-encryption key) and the second type of keys include a key chosen independently at random for each recipient (called a recipient specific key). The message-encryption key is used to encrypt a message with a private key cipher, while a recipient specific key is used to encrypt the message-encryption key.

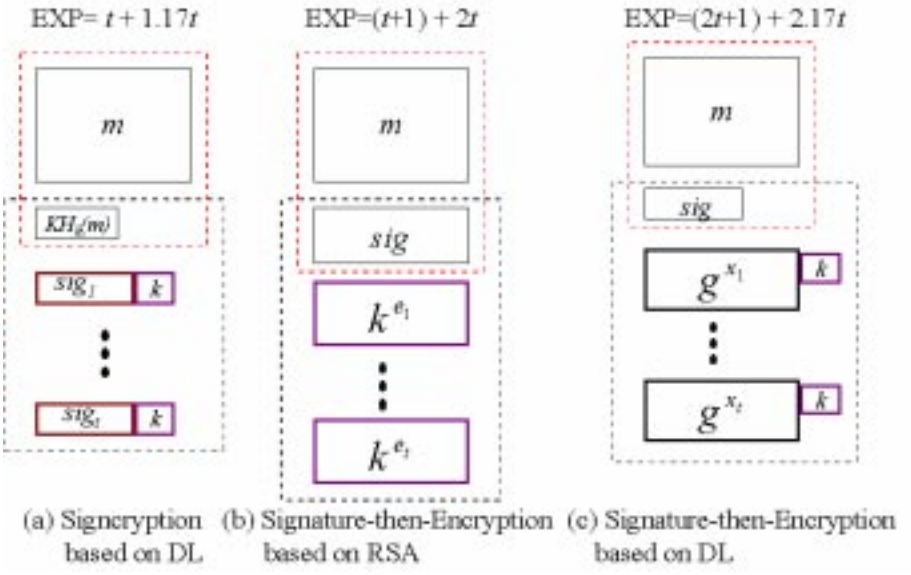

Figure 3: Output Formats of Signcryption and Signature-then-Encryption for Multiple Recipients

Having specified SCS1M, a signcryption for multiple recipients, next we proceed to examining other major issues with the scheme: message consistency, confidentiality, unforgeability, non-repudiation and efficiency.

As we discussed earlier, a message delivery scheme for multiple recipients is said to be consistent if messages recovered by the recipients are identical. Such a requirement is essential in the case of multiple recipients, as otherwise Alice the sender may be able to exclude a particular recipient from the group of recipients by deliberately causing the recipient to recover a message different from the one recovered by other recipients. With a signature-then-encryption scheme for multiple recipients, message consistency is not a problem in general. With SCS1M message consistency is achieved through the use of two techniques: (1) a message m is encrypted *together with the hashed value* $h = KH_k(m)$, namely $c = E_k(m, h)$. (2) m and k are both involved in the formation of r_i and s_i through $r_i = KH_{k_{i,2}}(m, h)$. These two techniques effectively prevent a recipient from being excluded from the group by a dishonest message originator.

Similarly to the case of a single recipient, identification information on each recipient R_i can be tied to a signcrypted text by involving R_i's public key or certificate in the computation of r_i, namely re-defining r_i as $r_i = KH_{k_{i,2}}(m, h, y_i, etc)$. (see Section 3.3).

Next we examine the efficiency of the schemes.

Signcryption by Alice the Sender for Multi-Recipients

An input to this signcryption algorithm for multi-recipients consists of a message m to be sent to t recipients $R_1, \ldots, R_t$, Alice's private key x_a, R_i's public key y_i for all $1 \leqq i \leqq t$, q and p.

1. Pick a random message-encryption key k, calculate $h = KH_k(m)$, and encrypt m by $c = E_k(m, h)$.
2. Create a signcrypted text of k for each recipient $i = 1, \ldots, t$:
   (a) Pick a random number v_i from $[1, \ldots, q-1]$ and calculate $k_i = hash(y_i^{v_i} \bmod p)$. Then split k_i into $k_{i,1}$ and $k_{i,2}$ of appropriate length.
   (b) $c_i = E_{k_{i,1}}(k)$.
   (c) $r_i = KH_{k_{i,2}}(m, h)$.
   (d) $s_i = v_i/(r_i + x_a) \bmod q$.

Alice then broadcasts to all the recipients $(c, c_1, r_1, s_1, \ldots, c_t, r_t, s_t)$.

Unsigncryption by Each Recipient

An input to this unsigncryption algorithm consists of a signcrypted text $(c, c_1, r_1, s_1, \ldots, c_t, r_t, s_t)$ received through a broadcast channel, together with a recipient R_i's private key x_i where $1 \leqq i \leqq t$, Alice's public key y_a, g, q and p.

1. Find out (c, c_i, r_i, s_i) in $(c, c_1, r_1, s_1, \ldots, c_t, r_t, s_t)$.
2. $k_i = hash((y_a \cdot g^{r_i})^{s_i \cdot x_i} \bmod p)$. Split k_i into $k_{i,1}$ and $k_{i,2}$.
3. $k = D_{k_{i,1}}(c_i)$.
4. $w = D_k(c)$. Split w into m and h.
5. check if h can be recovered from $KH_k(m)$ and r_i recovered from $KH_{k_{i,2}}(w)$.

R_i accepts m as a valid message originated from Alice only if both $h = KH_k(m)$ and $r_i = KH_{k_{i,2}}(w)$ hold.

Table 8: SCS1M — A Signcryption Scheme for Multiple Recipients

6.1 Comparison with a Discrete Logarithm Based Scheme

We compare SCS1M and SCS2M with the signature-then-encryption for multiple recipients based on Schnorr signature and ElGamal encryption. As can be seen in Table 9, saving by SCS1M (and by SCS2M) in computational cost and communication overhead can be summarized as follow: the number of modular exponentiations is reduced (1) for Alice the sender, from $2t+1$ to t (i.e., by a factor of larger than 50%), and (2) for each recipient, from 2.17 to 1.17 (i.e., by a factor of 45.2% on average, assuming Shamir's fast evaluation of the product of exponentials is used), while the communication overhead measured in bits is reduced from $t \cdot (|k| + |p|) + |hash(\cdot)| + |q|$ to $t \cdot (|k| + |KH.(\cdot)| + |q|) + |KH.(\cdot)|$. As $|p|$ is in general far larger than $|KH.(\cdot)| + |q|$ (compare $|p| = 512$ with $|KH.(\cdot)| = 72$ and $|q| = 144$), the saving in communication overhead is significant. To summarize the above discussion, SCS1M and SCS2M are far more efficient than the signature-then-encryption based on Schnorr signature and ElGamal encryption, both in terms of computational cost and communication overhead.

6.2 Comparison with an RSA Based Scheme

Now we compare SCS1M and SCS2M with RFC1421 [28] in which an RSA based signature-then-encryption for multiple recipients is specified. As the discrete logarithm and factorization problems are of equal complexity with our current knowledge (see Appendix A), we assume that $|n_a| = |n_b| = |p|$. First, two observations on computational costs can be made (see also Table 9):

1. For Alice the sender — The number of modular exponentiations is $t+1$ with RFC1421, as against t with SCS1M and SCS2M. Among the $r+1$ exponentiations with RFC1421, one is for RSA signature generation which involves a full length exponent, and the remaining are for RSA public key encryption which generally only involves small exponents. The t exponentiations with SCS1M and SCS2M all involve exponents from $[1, \ldots, q-1]$. In addition, both SCS1M and SCS2M involve more hashing, modular multiplications and additions. Hence it is fair to say that from Alice the sender's point of view, neither SCS1M nor SCS2M shows an advantage in computational cost over CFR1421.

2. For a recipient R_i — The number of modular exponentiations is two with RFC1421, and on average 1.17 with SCS1M and SCS2M. Since one of the two exponentiations with RFC1421 is invested in RSA decryption which involves a full size exponent, SCS1M and SCS2M are faster than RFC1421 from R_i's point of view.

A significant advantage of SCS1M and SCS2M over RFC1421, however, lies in its low communication overhead: RFC1421 expands a message by $|n_a| + \sum_{i=1,\ldots,t} |n_i|$ bits, which is a number of times larger than $t \cdot (|k| + |KH.(\cdot)| + |q|) + |KH.(\cdot)|$, the communication overhead of SCS1M and SCS2M. In conclusion, the following can be said: SCS1M and SCS2M share a similar computational cost with the scheme in RFC1421, but they have a significantly lower communication overhead than RFC1421.

A final note follows: comparisons between the new schemes and RSA or discrete logarithm based schemes in other aspects, including key management, forward secrecy, past recovery, repudiation settlement and users' group or world orientation, are similar to the case of a single recipient, and hence are committed here.

Schemes for Multiple Recipients	Computational cost	Communication overhead (in bits)
RFC1421 (signature-then-encryption for multi-recipients based on RSA)	EXP=$t+1$, HASH=1, ENC=1 (EXP=2, HASH=1, DEC=1)	$\|n_a\| + \sum_{i=1,\ldots,t} \|n_i\|$
signature-then-encryption for multi-recipients based on "Schnorr signature + ElGamal encryption"	EXP=$2t+1$, MUL=1, ADD=1 HASH=1, ENC_{short}=t, ENC=1 (EXP=2.17, MUL=1 HASH=1, DEC_{short}=1, DEC=1)	$t \cdot (\|k\| + \|p\|)+$ $\|hash(\cdot)\| + \|q\|$
SCS1M (signcryption for multi-recipients)	EXP=t, DIV=t, ADD=t HASH=$2t+1$, ENC_{short}=t, ENC=1 (EXP=1.17, MUL=2 HASH=3, DEC_{short}=1, DEC=1)	$t \cdot (\|k\| + \|KH.(\cdot)\| + \|q\|)+$ $\|KH.(\cdot)\|$
SCS2M (signcryption for multi-recipients)	EXP=t, MUL=t, DIV=t, ADD=t HASH=$2t+1$, ENC_{short}=t, ENC=1 (EXP=1.17, MUL=2 HASH=3, DEC_{short}=1, DEC=1)	$t \cdot (\|k\| + \|KH.(\cdot)\| + \|q\|)+$ $\|KH.(\cdot)\|$

where
EXP = the number of modular exponentiations (a fractional number indicates an average cost),
DIV = the number of modular divisions,
MUL = the number of modular multiplications,
ADD = the number of modular addition or subtraction,
HASH = the number of one-way or keyed hash operations,
ENC = the number of encryptions using a private key cipher,
ENC_{short} = the number of short encryptions using a private key cipher,
DEC = the number of decryptions using a private key cipher,
DEC_{short} = the number of short decryptions using a private key cipher,
$|k|$ = the size of a message-encryption key,
Parameters in the brackets indicate the number of operations involved in "decryption-then-verification" or "unsigncryption" by each participant.

Table 9: Cost Saving of Signcryption for Multiple Recipients

7 Applications of Signcryption

As discussed in the introduction, a major motivation of this work is to search for a more economical method for secure and authenticated transactions/message delivery. If digital signcryptions are applied in this area, the resulting benefits are potentially significant: for every single secure and authenticated electronic transaction, we may save 50% in computational cost and 85% in communication overhead.

The proposed signcryption schemes are compact and particularly suitable for smart card based applications. We envisage that they will find innovative applications in many areas including digital cash payment systems, EDI and personal heath cards. Of particular importance is the fact that signcryption may be used to design more efficient digital cash transaction protocols that are often required to provide with both the functionality of digital signature and encryption.

Another surprising property of the proposed signcryption schemes is that it enables us to carry out fast, secure, unforgeable and non-repudiatable key transport *in a single block whose size is smaller than* $|p|$. In particular, using the two signcryption schemes defined in Table 4, we can transport highly secure and authenticated keys in a single ATM cell (48 byte payload + 5 byte header). A possible combination of parameters is $|p| \geqq 512$, $|q| = 160$, and $|KH.(\cdot)| = 80$, which would allow the transport of an unforgeable and non-repudiatable key of up to 144 bits (see Figure 4). In the figure, (key, TQ) indicates materials on a key to be transported and TQ may contain a time-stamp or a nonce. The data item denoted by *other* in $r = KH_{k_2}(key, TQ, other)$ may contain name-binding data such as a recipient's public key.

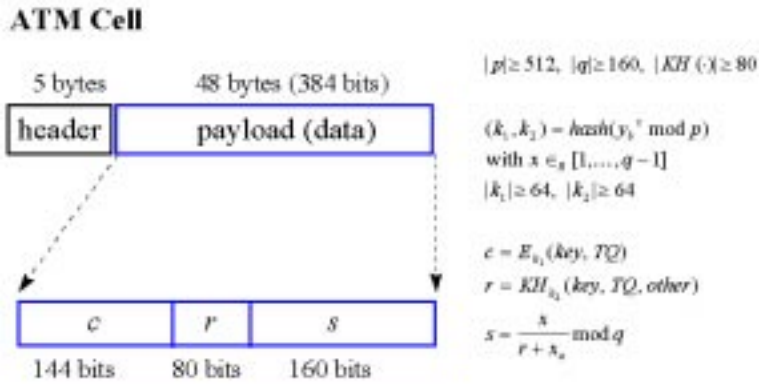

Figure 4: Secure and Unforgeable Key Transport in a Single ATM Cell (Based on SCS1)

Advantages of such a key transport scheme over interactive key exchange protocols such as those proposed in [13] are obvious, both in terms of computational efficiency and compactness of messages. Compared with previous attempts for secure, but un-authenticated, key transport based on RSA (see for example [21]), our key transport scheme has a further advantage in that it offers both unforgeability and non-repudiation. In a similar way, a multi-recipient signcryption scheme can be used as a very economical method for generating conference keys among a group of users. The reader is directed to [47] for more technical details.

8 Unforgeability, Non-repudiation and Confidentiality of Signcryption

Like any cryptosystem, security of signcryption in general has to address two aspects: (1) to protect what, and (2) against whom. With the first aspect, we wish to prevent the contents of a signcrypted message from being disclosed to a third party other than Alice, the sender, and Bob, the recipient. At the same time, we also wish to prevent Alice, the sender, from being masquerade by other parties, including Bob. With the second aspect, we consider the most powerful attackers one would be able to imagine in practice, namely adaptive attackers who are allowed to query Alice's signcryption algorithm and/or Bob's unsigncryption algorithm.

We say that a signcryption scheme is secure if the following conditions are satisfied:

1. Unforgeability — it is computationally infeasible for an adaptive attacker, who may be a dishonest Bob and allowed to query Alice's signcryption algorithm, to masquerade Alice in creating an authentic signcrypted text.

2. Non-repudiation — it is computationally feasible for a third party to settle a dispute between Alice and Bob in an event where Alice denies the fact that she is the originator of a signcrypted text with Bob as its recipient.

3. Confidentiality — it is computationally infeasible for an adaptive attacker to gain any *partial information* [15] on the contents of a signcrypted text. The adaptive attacker may be any party other than Alice and Bob. In addition to Bob's unsigncryption algorithm, the attacker is also allowed to query Alice's signcryption algorithm, as her secret key is involved in the creation of a signcrypted text.

A formal definition for unforgeability of digital signature can be found in [16]. As for confidentiality, a formal definition for security against a passive attacker, who does not query a decryption algorithm, was first introduced in [15], while a matching definition for security against an adaptive attacker was first provided in [39].

Before proceeding to the proof of the security of the signcryption schemes SCS1 and SCS2 defined in Table 4, let us first discuss the assumptions that are required by the proof.

8.1 Assumptions

Examining the procedures for signcryption and unsigncryption, we can see that they involve three basic operations that are related to assumptions: (1) private key (symmetric) encryption and decryption, (2) one-way hashing and keyed hashing, and (3) modular exponentiation. In the following we discuss relevant assumptions.

Assumption 1 *The private key cipher (E, D) hides all partial information on a message.*

Assumption 2 *The hash function hash behaves like a random function (or random oracle).*

As the keyed hash function KH can be constructed from *hash*, Assumption 2 implies that for every secret key k, $KH_k(\cdot)$ behaves like a random oracle. The above assumption also implies that both *hash* and KH hide all partial information on their input.

With the discrete exponentiation, there are three different, but closely related, computational problems in a sub-group of order q modular p, where p is prime and q is a prime factor of $p-1$. Recall that g is an integer in $[1,\ldots,p-1]$ with order q modulo p.

1. **Discrete Logarithm (DL) Problem.** Given (p,q,g,y), where $y \in [1,\ldots,p-1]$, find out an $x \in [1,\ldots,q-1]$ such that $y = g^x \bmod p$.

2. **Computational Diffie-Hellman (CDH) Problem.** Given (p,q,g,y_a,y_b), where $y_a, y_b \in [1,\ldots,p-1]$, find out a $y \in [1,\ldots,p-1]$ such that $y = g^{x_a x_b} \bmod p$, where $x_a, x_b \in [1,\ldots,q-1]$ satisfying $y_a = g^{x_a} \bmod p$ and $y_b = g^{x_b} \bmod p$. (Note that it is not required to actually find out x_a and x_b.)

3. **Decisional Diffie-Hellman (DDH) Problem.** Given (p,q,g,y_a,y_b,y_c), where $y_a, y_b, y_c \in [1,\ldots,p-1]$, decide whether $x_c = x_a x_b \bmod q$, where $x_a, x_b, x_c \in [1,\ldots,q-1]$ satisfying $y_a = g^{x_a} \bmod p$, $y_b = g^{x_b} \bmod p$, and $y_c = g^{x_c} \bmod p$. (As in the computational Diffie-Hellman problem, it is not required to actually find out x_a, x_b and x_c.)

Intuitively, a problem A is said to be reducible to another problem B if an algorithm for solving B can be converted into one for solving A. Put it in another way, saying that "A is reducible to B" is roughly the same as saying that "A is not harder than B". The current state of the art in relation to the above three problems is that the decisional Diffie-Hellman problem is reducible to the computational Diffie-Hellman problem, and the computational Diffie-Hellman problem is reducible to the Discrete Logarithm problem. It is not known whether the opposite direction is also true. The fastest algorithm currently available for solving any of the three problems takes an exponential amount of time. This leads to the following assumptions:

Assumption 3 The Discrete Logarithm assumption. *There is no efficient algorithm for solving the Discrete Logarithm problem.*

Assumption 4 The computational Diffie-Hellman assumption. *There is no efficient algorithm for solving the computational Diffie-Hellman problem.*

Assumption 5 The decisional Diffie-Hellman assumption. *There is no efficient algorithm for solving the decisional Diffie-Hellman problem.*

From the reductions discussed above, we can see that the decisional Diffie-Hellman assumption implies the computational Diffie-Hellman assumption, and the latter implies the Discrete Logarithm assumption. In other words, the decisional Diffie-Hellman assumption is the strongest among the three. The decisional Diffie-Hellman assumption has been very useful in cryptography. Our discussions on the forgeability of the signcryption schemes will be based on the Discrete Logarithm assumption, while discussions on the confidentiality of the schemes will have to rely on the stronger decisional Diffie-Hellman assumption. Some recent applications of the decisional assumption can be found in [11, 31].

Finally we note that all the assumptions can be stated in a precise and formal way using the terminology from the theory of computation. Such formalizations, however, are not critical to our discussions on the security of signcryption, and hence omitted.

8.2 Unforgeability

Regarding forging Alice's signcryption, a dishonest Bob is in the best position to do so, as he is the only person who knows x_b which is required to directly verify a signcrypted text from Alice. In other words, the dishonest Bob is the most powerful attacker we should look at. Given the signcrypted text (c, r, s) of a message m from Alice, Bob can use his private key x_b to decrypt c and obtain $m = D_{k_2}(c)$. Thus the original problem is reduced to one in which Bob is in possession of (m, r, s). The latter is identical to the unforgeability of SDSS1 or SDSS2.

As shown in Section 3.1, SDSS1 and SDSS2 are unforgeable. Therefore we conclude that both signcryption schemes SCS1 and SCS2 are unforgeable against adaptive attacks, under the assumptions that the (keyed) hash function behaves like a random function, and that the Discrete Logarithm is hard to compute (Assumptions 1 and 2).

8.3 Non-repudiation

As discussed in Section 5, signcryption requires a repudiation settlement procedure different from the one for a digital signature scheme is required. In particular, the judge would need Bob's cooperation in order to correctly decide the origin of the message. In what follows we describe four possible repudiation settlement procedures, each requiring a different level of trust on the judge's side.

8.3.1 With a Trusted Tamper-Resistant Device

If a tamper-resistant device is available, a trivial settlement procedure starts with the judge asking Bob to provide the device with q, p, g, y_a, y_b, m, c, r, s and his private key x_b, together with certificates for y_a and y_b. The tamper-resistant device would follow essentially the same steps used by Bob in unsigncrypting (c, r, s). It would output "yes" if it can recover m from (c, r, s), and "no" otherwise. The judge would then take the output of the tamper-resistant device as her decision. Note that in this case, Bob puts his trust completely on the device, rather than on the judge.

8.3.2 By a Trusted Judge

If the judge is trusted, achieving correct repudiation settlement by the judge is again trivial: Bob simply presents to the the judge x_b together with other data items. Note that both in this case and in the case of a tamper-resistant device, Bob's presenting k, rather than x_b, to the judge or the device without convincing her or the device that k satisfies the condition of $k = hash(u^{x_b} \bmod p)$, would open up a door for a dishonest Bob to frame Alice by providing a message made up by himself, where x_b is Bob's private key, $u = (y_a \cdot g^r)^s \bmod p$ for SCS1, and $u = (g \cdot y_a^r)^s \bmod p$ for SCS2.

Note that once being given x_b, the judge can do everything Bob can with x_b.

8.3.3 By a Less Trusted Judge

Another possible solution would be for Bob to present $v = u^{x_b} \bmod p$, rather than x_b, to the judge. Bob and the judge then engage in a zero-knowledge interactive/non-interactive proof/argument protocol (with Bob as a prover and the judge as a verifier), so that Bob can convince the judge of the fact that v does have the right form. (A possible candidate protocol is a 4-move zero-knowledge proof protocol developed in [7].)

Bob has to be aware of the fact that with this repudiation settlement procedure, the judge can obtain from v, r, s and y_b the Diffie-Hellman shared key between Alice and Bob, namely $k_{DH,ab} = g^{x_a x_b} \bmod p$ ($= v^{1/s} y_b^{-r} \bmod p$ for SCS1). With $k_{DH,ab}$, the judge can find out v^* for other communication sessions between Alice and Bob, and hence recover the corresponding messages ($v^* = k_{DH,ab}^{s^*} y_b^{r^* \cdot s^*} \bmod p$ for SCS1). Therefore Bob may not rely on this repudiation settlement procedure if the judge is not trusted by either Alice or Bob.

8.3.4 By any (Trusted/Untrusted) Judge

Now we describe a repudiation settlement procedure that works even in the case when the judge corrupts and is not trusted. The procedure uses techniques in zero-knowledge proofs/arguments [3] and guarantees that the judge can make a correct decision, with no useful information on Bob's private key x_b being leaked out to the judge.

First Bob presents following data to the judge: q, p, g, y_a, y_b, m, c, r, s and certificates for y_a and y_b. Note that Bob does not hand out x_b, k or $v = u^{x_b} \bmod p$. The judge then verifies the authenticity of y_a and y_b. If satisfied both with y_a and y_b, the judge computes $u = (y_a \cdot g^r)^s \bmod p$ when SCS1 is used, and $u = (g \cdot y_a^r)^s \bmod p$ when SCS2 is used instead. Bob and the judge then engage in a zero-knowledge interactive protocol, with Bob as a prover and the judge as a verifier.

The goal of the protocol is for Bob to convince the judge of the fact that he knows a satisfying assignment $z = x_b$ to the following Boolean formula φ:

$$\begin{aligned} \varphi(z) \quad = \quad & (g^z \bmod p == y_b) \wedge \\ & (D_{k_1}(c) == m) \wedge \\ & (KH_{k_2}(D_{k_1}(c)) == r) \end{aligned}$$

where k_1 and k_2 are defined by $(k_1, k_2) = hash(u^z \bmod p)$, and $==$ denotes equality testing.

φ is clearly a satisfiable Boolean formula in the class of NP. There are a large number of zero-knowledge proof/argument protocols for NP statements in the literature. Some zero-knowledge protocols are based on assumptions on specific computational problems such integer factorization and discrete logarithm, while others work with general complexity assumptions such as the existence of one-way functions. Here we only mention one of such protocols recently proposed in [2]. This zero-knowledge argument protocol assumes that both Bob (the prover) and the judge (the verifier) are polynomially bounded in computing time, which matches our cryptographic setting. In other words, it is a zero-knowledge argument protocol. It consists of only four moves of messages between Bob and the judge, and any one-way function whose input and out are of equal length can be used to build a so-called bit-commitment scheme used in the protocol. In practice, the one-way function can be instantiated with a one-way hash function or a secure block cipher. After an execution of the protocol, the judge announces that (c, r, s) is originated from Alice to Bob only if she is convinced that the Boolean formula φ is satisfiable.

To summarize the above discussion on interactions between Bob and the judge, one (such as an implemented of a signcryption scheme) would first correctly figure out the Boolean formula φ. Then one would select a suitable one-way function. And finally one would code the four-move protocol specified in [2].

[3] The main difference between a proof and an argument in the context of zero-knowledge protocols is that, while an argument assumes that a prover runs in polynomial time, a proof works even if a prover has unlimited computational power. A zero-knowledge argument suffices for most cryptographic applications, including repudiation settlement in signcryption.

Properties of the protocol include: (1) the judge always correctly announces that (c, r, s) is originated from Alice when it is indeed so; (2) the probability is negligibly small for the judge to declare that (c, r, s) is originated from Alice when in fact it is not; (3) no useful information on Bob's private key x_b is leaked to the judge (or any other parties).

Two remarks on the interactive repudiation settlement procedure follow. First, the message m may be dropped from the data items handed over to the judge, if Bob does not wish to reveal the contents of m to the judge. Second, Bob may include k into the data handed over to the judge if k is defined as $k = hash(y_b^x \bmod p)$ in which a one-way hash function $hash$ is involved. This will reduce the computation and communication load involved in the interactions without compromising the security of x_b, especially when $hash$ is a cryptographically strong function that does not leak information on its input.

Finally we note that if Bob and the judge share a common random bit string, then the number of moves of messages between Bob and the judge can be minimized to 1, by the use of a non-interactive zero-knowledge proof protocol such as the one proposed in [22].

8.4 Confidentiality

Finally we consider the confidentiality of message contents against an adaptive attacker who is allowed to query both the signcryption and unsigncryption algorithms. We have already proved in Section 8.2 that the signcryption schemes are unforgeable by any attacker who is allowed to query the signcryption algorithm (and even has access to Bob's secret key), assuming that the hash function is a random oracle and that the discrete logarithm problem is hard. This implies that every signcrypted text that can be produced by such an attacker, during or after mounting an attack, must have been obtained by querying the signcryption algorithm. This in turn implies that the attacker already knows the corresponding plain text that the attacker must present to the signcryption algorithm for a query.

As the attacher is already aware of the plain text, being allowed to query the unsigncryption algorithm does *not* help the attacker in any way. Thus we only have to examine the security of the signcryption schemes against a passive attacker who is allowed to query the signcryption algorithm, but not the unsigncryption algorithm.

We use SCS1 as an example in proving security against a passive attacker, as discussions for SCS2 are similar. Given the signcrypted text (c, r, s) of a message m from Alice, a passive attacker can obtain $u = (y_a \cdot g^r)^s = g^x \bmod p$. Thus to the attacker, data related to the signcrypted text of m include: q, p, g, $y_a = g^{x_a} \bmod p$, $y_b = g^{x_b} \bmod p$, $u = g^x \bmod p$, $c = E_{k_1}(m)$, $r = KH_{k_2}(m)$, and $s = x/(r + x_a) \bmod q$.

We wish to show that it is computationally infeasible for the attacker to find out any partial information on the message m from the related data listed above. We will achieve our goal by reduction: we will reduce the confidentiality of another encryption scheme to be defined shortly (called C_{kh} for convenience) to the confidentiality of SCS1.

The encryption scheme C_{kh} is based on ElGamal encryption scheme. With this encryption scheme, the ciphertext of a message m to be sent to Bob is defined as ($c = E_{k_1}(m)$, $u = g^x \bmod p$, $r = KH_{k_2}(m)$) where (1) x is chosen uniformly at random from $[1, \ldots, q-1]$, and (2) $(k_1, k_2) = k = hash(y_b^x \bmod p)$. It turns out C_{kh} is a slightly modified version of a scheme that has received special attention in [45, 3]. (See also earlier work [49].) To prove that C_{kh} hides all partial information from a passive attacker, we note that being given $r = KH_{k_2}(m)$ does not help the attacker at all, since the keyed hash function hides all partial information on its input. Thus the problem can be translated into a new one in which we wish to prove the security of $(c = E_{k_1}(m),\ u = g^x \bmod p)$ against the passive

attacker. It turns out that ($c = E_{k_1}(m)$, $u = g^x \bmod p$) does hide all partial information, due to the following facts: (1) as q is prime and hence y_b has order q modular p, when x is chosen uniformly at random from $[1, \ldots, q-1]$, $y_b^x \bmod p$ is a random element in the sub-group of order q; (2) Thus $k = hash(y_b^x \bmod p)$ is a random key, assuming that the decisional Diffie-Hellman problem is hard and that the hash function behaves like a random oracle.

Similar observations on the security of other relevant encryption schemes against a passive attacker have been made in [11, 31]. To summarize the above discussions, C_{kh} hides all partial information from a passive attacker, under the assumptions that the private key cipher (E, D) is secure and hides all partial information, that the (keyed) hash function behaves like a random oracle, and that the decisional Diffie-Hellman problem is hard to solve.

Now we assume that there is an attacker for SCS1. Let us call this attacker A_{SCS1}. We show how A_{SCS1} can be translated into one for C_{kh}, called $A_{C_{kh}}$. Recall that for a message m, the input to A_{SCS1} includes q, p, g, $y_a = g^{x_a} \bmod p$, $y_b = g^{x_b} \bmod p$, $u = g^x \bmod p$, $c = E_{k_1}(m)$, $r = KH_{k_2}(m)$, and $s = x/(r + x_a) \bmod q$. With the attacker $A_{C_{kh}}$ for C_{kh}, however, its input includes: q, p, g, $y_b = g^{x_b} \bmod p$, $u = g^x \bmod p$, $c = E_{k_1}(m)$, and $r = KH_{k_2}(m)$. One immediately identifies that two numbers that correspond to y_a and s which are needed by A_{SCS1} as part of its input are currently missing from the input to $A_{C_{kh}}$. Thus, in order for $A_{C_{kh}}$ to "call" the attacker A_{SCS1} "as a sub-routine", $A_{C_{kh}}$ has to create two numbers corresponding to y_a and s in the input to A_{SCS1}. Call these two yet-to-be-created numbers y'_a and s'. y'_a and s' have to have the right form so that $A_{C_{kh}}$ can "fool" A_{SCS1}. It turns out that such y'_a and s' can be easily created by $A_{C_{kh}}$ as follows: (1) pick a random number s' from $[1, \ldots, q-1]$. (2) let $y'_a = u^{1/s'} \cdot g^{-r} \bmod p$.

From the reduction, we see that if there is a passive attacker that finds any partial information on a message in SCS1, then the attacker can be used to find partial information on a message in C_{kh}. As we have already proven the confidentiality of C_{kh}, we conclude that no attacker for SCS1 can find any partial information on a message, under the assumptions (1) that the private key cipher (E, D) is secure and hides all partial information, (2) that the (keyed) hash function behaves like a random oracle and hence hides all partial information on its input, and (3) that the decisional Diffie-Hellman problem is hard to solve (Assumptions 1, 2 and 5).

We note that the proof techniques discussed in this section can also be used for other signcryption schemes, provided that the underlying digital signature schemes are provably secure. In addition, the techniques can be extended to prove the security of the signcryption schemes for multiple recipients (SCS1M and SCS2M) described in Section 6.

9 Conclusion

We have introduced a new cryptographic primitive called signcryption for secure and authenticated message delivery, which fulfills all the functions of digital signature and encryption, but with a far smaller cost than that required by the current standard signature-then-encryption methods. Security of the signcryption schemes has been proven, and extensions of the schemes to multiple recipients has been carried out. We believe that the new primitive will open up a number of avenues for future research into more efficient security solutions.

The signcryption schemes proposed in this paper have been based on ElGamal signature and encryption. We have not been successful in searching for a signcryption scheme

employing RSA or other public key cryptosystems. Therefore it remains a challenging open problem to design signcryption schemes based factorization or other computationally hard problems.

Our signcryption schemes all involve a zero-knowledge interactive protocol in repudiation settlement. There seems to exist a trade-off overhead in computation and communication and other dimensions such as forward secrecy, past recovery and complexity of repudiation settlement. Thus an even more challenging task would be to find out, say from an information theoretic point of view, exactly how the trade-off varies, with a view to design a signcryption scheme that is efficient and admits a less complicated repudiation settlement procedure.

Acknowledgment

Many revisions on this paper have been carried out while the author was on sabbatical at the University of Tokyo. The very idea of combining signature with encryption can be partially traced many years back when I was still Professor Imai's PhD student and learnt the beauty of Imai-Hirakawa scheme, a revolutionary invention that combines modulation with error-correcting codes with an aim to achieve more reliable and efficient communications.

Thanks also go to Dr Minghua Qu who pointed out the potential risk of double-payment by Alice to a pair of collusive friends Bob and Cathy, to Dr Burt Kaliski who first noticed a lack of forward secrecy with the signcryption schemes, and to Drs Markus Michels and Holger Petersen who pointed out the problem with using Chaum's 4-move zero-knowledge protocol in repudiation settlement by a dishonest judge.

Comments from anonymous referees for Crypto'97 have also been helpful in improving the presentation of this paper.

References

[1] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology - CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15, Berlin, New York, Tokyo, 1996. Springer-Verlag.

[2] M. Bellare, M. Jakobsson, and M. Yung. Round-optimal zero-knowledge arguments based on any one-way function. In *Advances in Cryptology - EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 280–305, Berlin, New York, Tokyo, 1997. Springer-Verlag.

[3] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 62–73, New York, November 1993. The Association for Computing Machinery.

[4] M. Bellare and P. Rogaway. Optimal asymmetric encryption — how to encrypt with RSA. In *Advances in Cryptology - EUROCRYPT'93*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Berlin, New York, Tokyo, 1994. Springer-Verlag.

[5] M. Bellare and P. Rogaway. The exact security of digital signatures — how to sign with RSA and Rabin. In *Advances in Cryptology - EUROCRYPT'96*, volume 1070 of

Lecture Notes in Computer Science, pages 399–416, Berlin, New York, Tokyo, 1996. Springer-Verlag.

[6] E. Brickell and K. McCurley. Interactive identification and digital signatures. *AT&T Technical Journal*, pages 73–86, November/Decmber 1991.

[7] D. Chaum. Zero-knowledge undeniable signatures. In *Advances in Cryptology - EUROCRYPT'90*, volume 473 of *Lecture Notes in Computer Science*, pages 458–464, Berlin, New York, Tokyo, 1990. Springer-Verlag.

[8] M. Chen and E. Hughes. Protocol failures related to order of encryption and signature: Computation of discrete logarithms in RSA groups, April 1997. (Draft).

[9] D. Coppersmith. Finding a small root of a univariate modular equation. In *Advances in Cryptology - EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 153–165, Berlin, New York, Tokyo, 1996. Springer-Verlag.

[10] D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter. Low-exponent RSA with related messages. In *Advances in Cryptology - EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 1–9, Berlin, New York, Tokyo, 1996. Springer-Verlag.

[11] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology - CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, Berlin, New York, Tokyo, 1998. Springer-Verlag.

[12] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):472–492, 1976.

[13] W. Diffie, P. van Oorschot, and M. Wiener. Authentication and authenticated key exchange. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[14] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[15] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[16] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptively chosen message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[17] D. Harkins and D. Carrel. The resolution of ISAKMP with Oakley, March 1998. Internet-draft (draft-ietf-ipsec-isakmp-oakley-07.txt).

[18] P. Horster, M. Michels, and H. Petersen. Meta-ElGamal signature schemes. In *Proceedings of the second ACM Conference on Computer and Communications Security*, pages 96–107, New York, November 1994. The Association for Computing Machinery.

[19] P. Horster, M. Michels, and H. Petersen. Meta-message recovery and meta-blind signature schemes based on the discrete logarithm problem and their applications. In *Advances in Cryptology - ASIACRYPT'94*, volume 917 of *Lecture Notes in Computer Science*, pages 224–237, Berlin, New York, Tokyo, 1995. Springer-Verlag.

[20] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–253, 1989.

[21] D. Johnson and S. Matyas. Asymmetric encryption: Evolution and enhancements. *CryptoBytes*, 2(1):1–6, 1996. (available at `http://www.rsa.com/`).

[22] J. Kilian and E. Petrank. An efficient non-interactive zero-knowledge proof system for NP with general assumption. *Electronic Colloquium on Computational Complexity*, Reports Series(TR95-038), 1995. (available at `http://www.eccc.uni-trier.de/eccc/`).

[23] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[24] A. Lenstra. Using cyclotomic polynomials to construct efficient discrete logarithm cryptosystems over finite fields. In *Information Security and Privacy – Proceedings of ACISP'97*, volume 1270 of *Lecture Notes in Computer Science*, pages 127–138, Berlin, New York, Tokyo, 1997. Springer-Verlag.

[25] A. K. Lenstra and H. W. Lenstra. *Algorithms in Number Theory*, volume A of *Handbook in Theoretical Computer Science*. Elsevier and the MIT Press, 1990.

[26] A. K. Lenstra and H. W. Lenstra. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, 1993.

[27] C. H. Lim and P J. Lee. Directed signatures and applications to threshold cryptography. In *Security Protocols*, volume 1189 of *Lecture Notes in Computer Science*, pages 131–138, Berlin, New York, Tokyo, 1997. Springer-Verlag.

[28] J. Linn. Privacy enhancement for internet electronic mail: Part I: Message encryption and authentication procedures. Request for Comments RFC 1421, IETF, 1993.

[29] T. Matsumoto and H. Imai. On the key predistribution systems: A practical solution to the key distribution problem. In *Advances in Cryptology - CRYPTO'87*, volume 239 of *Lecture Notes in Computer Science*, pages 185–193, Berlin, New York, Tokyo, 1987. Springer-Verlag.

[30] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[31] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *Proceedings of the 38-th Annual Symposium on Foundations of Computer Science*, pages 458–467, 1997.

[32] National Bureau of Standards. Data encryption standard. Federal Information Processing Standards Publication FIPS PUB 46, U.S. Department of Commerce, January 1977.

[33] National Institute of Standards and Technology. Digital signature standard (DSS). Federal Information Processing Standards Publication FIPS PUB 186, U.S. Department of Commerce, May 1994.

[34] National Institute of Standards and Technology. Secure hash standard. Federal Information Processing Standards Publication FIPS PUB 180-1, U.S. Department of Commerce, April 1995.

[35] K. Nyberg and R. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. *Designs, Codes and Cryptography*, 7(1/2):61–81, 1996.

[36] A. Odlyzko. The future of integer factorization. *CryptoBytes*, 1(2):5–12, 1995. (available at `http://www.rsa.com/`).

[37] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Advances in Cryptology - EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398, Berlin, New York, Tokyo, 1996. Springer-Verlag.

[38] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 1998. (to appear).

[39] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen-ciphertext attacks. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, Berlin, New York, Tokyo, 1992. Springer-Verlag.

[40] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–128, 1978.

[41] C. P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–251, Berlin, New York, Tokyo, 1990. Springer-Verlag.

[42] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[43] A. Shamir. RSA for paranoids. *CryptoBytes*, 1(3):1–4, 1995. (available at `http://www.rsa.com/`).

[44] M. Wegman and J. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.

[45] Y. Zheng. Improved public key cryptosystems secure against chosen ciphertext attacks. Technical Report 94-1, University of Wollongong, Australia, January 1994.

[46] Y. Zheng. The SPEED cipher. In *Proceedings of Financial Cryptography'97*, volume 1318 of *Lecture Notes in Computer Science*, pages 71–89, Berlin, New York, Tokyo, 1997. Springer-Verlag.

[47] Y. Zheng and H. Imai. Compact and unforgeable session key establishment over an ATM network. In *Proceedings of IEEE INFOCOM'98*, pages 411–418, San Francisco, 1998. IEEE.

[48] Y. Zheng, J. Pieprzyk, and J. Seberry. HAVAL - a one-way hashing algorithm with varialbe length of output. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology - AUSCRYPT'92*, volume 718 of *Lecture Notes in Computer Science*, pages 83–104, Berlin, New York, Tokyo, 1993. Springer-Verlag.

[49] Y. Zheng and J. Seberry. Immunizing public key cryptosystems against chosen ciphertext attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):715–724, June 1993.

[50] P. Zimmermann. *PGP Source Code and Internals.* MIT Press, Cambridge, Mass., 1995.

A RSA and Discrete Logarithm Based Signature and Encryption

This appendix is intended as a concise summary of the factorization problem, the discrete logarithm problem, RSA encryption and signature [40], ElGamal signature and encryption [14], Schnorr signature [41], and Digital Signature Standard (DSS) [33]. For more technical details, the reader is directed to the relevant original papers and documents.

To assist our description of the various schemes, we consider a situation where a user (say Alice) wishes to deliver a message to another user (say Bob) over an open insecure communication network such as Internet. In addition we use $hash$ to denote a one-way hash algorithm such as SHS [34] and HAVAL [48]. We also use E and D to denote the encryption and decryption algorithms of a private key cipher such as DES [32] and SPEED [46]. Encrypting a message m with a key k, typically in the cipher block chaining or CBC mode, is indicated by $E_k(m)$, while decrypting a ciphertext c with k is denoted by $D_k(c)$.

A.1 Hardness of Factorization and Discrete Logarithm

The security of RSA is based on the difficulty of factoring large composite numbers, while the security of other schemes described in this appendix is based on the difficulty of computing discrete logarithm on large finite fields. Currently, the fastest algorithm for factorization is the number field sieve (see Page 50 of [26]) whose running time is asymptotically

$$e^{(1.923+o(1))((\ln |n|)^{1/3}(\ln \ln |n|)^{2/3})}$$

where n is a composite to be factored, $|n|$ refers to the size of n (in bits), and $o(1)$ denotes a function of $|n|$ that approaches 0 as $|n| \to \infty$. Similarly the fastest algorithm for computing discrete logarithm on $GF(p)$ for a prime p is also based on the number field sieve (see Page 13 of [26]) with the same asymptotic running time as that for factoring, namely

$$e^{(1.923+o(1))((\ln |p|)^{1/3}(\ln \ln |p|)^{2/3})}$$

Some cryptographic schemes such as DSS and Schnorr signature rely on the difficulty of computing discrete logarithms in a sub-group of order q. One may compute discrete logarithms in the sub-group either by solving the general discrete logarithm problem or working on the sub-group directly. Currently the best algorithms that work directly on the sub-group are Shank's baby-step-giant-step and Pollard's rho methods, whose running times are both in the order of $O(\sqrt{q}) = e^{0.345|q|+O(1)}$. (See [25] for a survey on this and other related topics.) The two algorithms become ineffective for sufficiently large q, say $|q| \geqq 144$.

Comparing the running time of the two algorithms, we conclude that with the current state of the art, computing discrete logarithm on $GF(p)$ and factoring a composite n of the same size are equally difficult. This simplifies our comparison of the efficiency of a cryptographic scheme based on RSA against that based on discrete logarithm, as we can assume that n and p are of the same size.

A.2 RSA Signature and Encryption

The RSA scheme is based on the difficulty of factoring large composite numbers. To use RSA, Alice first has to choose two large random primes p_a and q_a. She then calculates the products $n_a = p_a q_a$ and $\varphi(n_a) = (p_a - 1)(q_a - 1)$. Next she selects two numbers e_a and

d_a from $(1, \ldots, n_a)$ such that $e_a d_a = 1 \bmod \varphi(n_a)$. Finally Alice publishes (e_a, n_a) as her public key in a public key file, while keeps d_a as her private key.

Alice's signature on a message m is defined as $s = hash(m)^{d_a} \bmod n_a$. Other users can verify whether s is Alice's valid signature on m by checking whether $hash(m)$ is identical to $s^{e_a} \bmod n_a$.

Similarly to Alice, user Bob can create his public key (e_b, n_b) and private key d_b. To send a (long) message m to Bob in a secure way, Alice picks a random message-encryption key k and sends to Bob $c_1 = E_k(m)$ and $c_2 = k^{e_b} \bmod n_b$. Upon receiving c_1 and c_2, Bob can retrieve k by calculating $c_2^{d_b} \bmod n_b$, with which he can decrypt c_1.

A.3 ElGamal Signature and Encryption

ElGamal digital signature and encryption schemes are based on the hardness of computing discrete logarithm over a large finite field. It involves two parameters public to all users:

1. p: a large prime.
2. g: an integer in $[1, \ldots, p-1]$ with order $p-1$ modulo p.

User Alice's private key is an integer x_a chosen randomly from $[1, \ldots, p-1]$ with $x_a \not| (p-1)$ (i.e., x_a does not divide $p-1$), and her public key is $y_a = g^{x_a} \bmod p$.

Alice's signature on a message m is composed of two numbers r and s which are defined as

$$\begin{aligned} r &= g^x \bmod p \\ s &= (hash(m) - x_a \cdot r)/x \bmod (p-1) \end{aligned}$$

where x is a random number from $[1, \ldots, p-1]$ with $x \not| (p-1)$. It should be stressed that x must be chosen independently at random every time a message is to be signed by Alice.

Given (m, r, s), one can verify whether $g^{hash(m)} = y_a^r \cdot r^s \bmod p$ is satisfied. (r, s) is regarded as Alice's signature on m only if the equation holds.

Now assume that Bob has also chosen his private key x_b randomly from $[1, \ldots, p-1]$ with $x_b \not| (p-1)$, and made public the matching public key $y_b = g^{x_b} \bmod p$. By using Bob's public key. Alice can send him messages in a secure way. To do this, Alice chooses, for each message m, a random integer x from $[1, \ldots, p-1]$ with $x \not| (p-1)$, calculates $k = y_b^x \bmod p$ and sends to Bob $c_1 = E_k(m)$ and $c_2 = g^x \bmod p$.

Upon receiving c_1 and c_2, Bob can recover k by $k = c_2^{x_b} \bmod p$. He can then use k to decrypt c_1 and obtain m.

Note that ElGamal encryption can also be achieved using parameters for Schnorr signature and DSS described below.

A.4 Schnorr Signature Scheme

Schnorr signature scheme, together with DSS to be described below, is a variant of ElGamal signature scheme. The main idea behind the two variants is to choose g to be an integer in $[1, \ldots, p-1]$ with order q modulo p for a prime factor q of $p-1$, instead of with order $p-1$ modulo p.

Schnorr signature scheme involves the following parameters:

1. Parameters public key to all users:

(a) p: a large prime, say $p \geqq 2^{512}$.

(b) q: a prime factor of $p-1$. The size of q would be at least 2^{144}.

(c) g: an integer in $[1, \ldots, p-1]$ with order q modulo p. In practice, g is obtained by calculating $g = h^{(p-1)/q} \bmod p$ where h is chosen uniformly at random from $[2, \ldots, p-2]$ and satisfies $h^{(p-1)/q} \bmod p > 1$.

2. Parameters specific to user Alice:

(a) Alice's private key: a number x_a drawn randomly from $[1, \ldots, q-1]$.

(b) Alice's public key: $y_a = g^{-x_a} \bmod p$.

With the above parameters, Schnorr suggests that Alice sign a digital document m by picking a random x from $[1, \ldots, q-1]$ and appending to m a pair of numbers (r, s) which are calculated as follows:

$$\begin{aligned} r &= hash(g^x \bmod p, m) \\ s &= x + x_a \cdot r \bmod q \end{aligned}$$

The procedure for other people to verify Alice's signature (r, s) on m is straightforward: checking whether r is identical to $hash((g^s \cdot y_a^r \bmod p), m)$.

If Alice publishes $y_a = g^{x_a} \bmod p$, instead of $y_a = g^{-x_a} \bmod p$, then s can be defined as $s = x - x_a \cdot r \bmod q$. Signature verification is the same.

A.5 Digital Signature Standard (DSS)

The public and private parameters involved in DSS are all the same as those in Schnorr signature scheme, except that for DSS, Alice's public key is $y_a = g^{x_a} \bmod p$, but not $y_a = g^{-x_a} \bmod p$ as is the case for Schnorr signature scheme. In addition, the standard suggests that, for current applications, $|p|$ be between 512 and 1024, $|q| = 160$, and SHS [34] whose output has 160 bits be used as the one-way hash function.

Alice's signature on a message m is composed of two numbers r and s which are defined as

$$\begin{aligned} r &= (g^x \bmod p) \bmod q \\ s &= (hash(m) + x_a \cdot r)/x \bmod q \end{aligned}$$

where x is a random number chosen from $[1, \ldots, q-1]$.

Given (m, r, s), one can verify whether (r, s) is indeed Alice's signature on m by the following steps:

1. calculates $v = (g^{hash(m)/s} \cdot y_a^{r/s} \bmod p) \bmod q$.

2. accepts (r, s) as valid only if $v = r$.

Table 10 shows the computational cost and communication overhead of the signature and encryption schemes. The table assumes that Shamir's method for fast computation of the product of multiple exponentials with the same modulo is employed (for technical details of the method, see Appendix B) Note that to use RSA signature in a provably secure way, more extra computational effort (not shown in the table) has to be invested in the signing process [5]. Similarly, to employ RSA and ElGamal encryptions in a provably secure fashion, more computational effort and communication overhead is required (see [4, 21] for provably secure RSA encryption and [49, 45] for provably secure ElGamal encryption).

Various Schemes	Computational cost	Communication overhead (in bits)
RSA encryption	EXP=1, ENC=1 (EXP=1, DEC=1)	$\|n_b\|$
ElGamal encryption	EXP=2, ENC=1 (EXP=1, DEC=1)	$\|p\|$
RSA signature	EXP=1, HASH=1 (EXP=1, HASH=1)	$\|n_a\|$
ElGamal signature	EXP=1, MUL=1, DIV=1 ADD=1, HASH=1 (EXP=1.25, MUL=1, DIV=0 ADD=0, HASH=1)	$2\|p\|$
Schnorr signature	EXP=1, MUL=1, ADD=1, HASH=1 (EXP=1.17, MUL=1, ADD=0, HASH=1)	$\|hash(\cdot)\| + \|q\|$
DSS	EXP=1, MUL=1, DIV=1 ADD=1, HASH=1 (EXP=1.17, MUL=1, DIV=2 ADD=0, HASH=1)	$2\|q\|$

where
EXP = the number of modular exponentiations (a fractional number indicates an average cost),
MUL = the number of modular multiplications,
DIV = the number of modular division (inversion),
ADD = the number of modular addition or subtraction,
HASH = the number of one-way or keyed hash operations,
ENC = the number of encryptions using a private key cipher,
DEC = the number of decryptions using a private key cipher,
Parameters in the brackets indicate the number of operations involved in verification or decryption.

Table 10: Cost of RSA, ElGamal, Schnorr, DSS

B Fast Computation of the Product of Multiple Exponentials with the Same Modulo

In unsigncryption, the most expensive part of computation is contributed by $g_0^{e_0} g_1^{e_1} \bmod p$, where g_0, g_1, e_0, e_1 and p are all large integers. Although the computation can be carried out in a straightforward way, namely computing $y_0 = g_0^{e_0} \bmod p$ and $y_1 = g_1^{e_1} \bmod p$ separately and then multiplying y_0 and y_1 together, it was observed by A. Shamir that as the product involves the same modulo, the final result can be obtained with a smaller computational cost by using a variant of the "square-and-multiply" method for exponentiation (see [14] as well as Algorithm 14.88 on Page 618 of [30]). The following algorithm embodies Shamir's technique to compute the product of k exponentials with the same modulo.

INPUT: Integers p, g_0, g_1, $\cdots$, g_{k-1} and e_0, e_1, $\cdots$, e_{k-1}, where the size of each e_i is t bits.

OUTPUT: $g_0^{e_0} g_1^{e_1} \cdots g_{k-1}^{e_{k-1}} \bmod p$.

1. Let the t-bit binary representation of e_j, where $0 \leqq j \leqq k-1$, be

$$e_j = e_{j,t-1} e_{j,t-2} \cdots e_{j,0}$$

   And let E be a $k \times t$ binary array whose jth row is the binary representation of e_j. Note that the most significant bits of e_j are stored in the left hand side of a row. Call E an exponent array.

$$\mathrm{E} = \begin{array}{|c|c|c|c|} \hline e_{0,t-1} & e_{0,t-2} & \cdots & e_{0,0} \\ \hline e_{1,t-1} & e_{1,t-2} & \cdots & e_{1,0} \\ \hline & & \vdots & \\ \hline e_{k-1,t-1} & e_{k-1,t-2} & \cdots & e_{k-1,0} \\ \hline \end{array}$$

2. Denote by I_{t-1}, I_{t-2}, ..., I_0 the t integers represented by the t columns of the exponent array E, with $e_{0,i}$ being the least significant bit, and $e_{k-1,i}$ the most significant bit, of I_i, where $t-1 \geqq i \geqq 0$. Namely, $I_i = e_{k-1,i} e_{k-2,i} \cdots e_{0,i}$, as indicated below.

I_{t-1}	I_{t-1}	$\cdots$	I_0
$e_{0,t-1}$	$e_{0,t-2}$	$\cdots$	$e_{0,0}$
$e_{1,t-1}$	$e_{1,t-2}$	$\cdots$	$e_{1,0}$
		$\vdots$	
$e_{k-1,t-1}$	$e_{k-1,t-2}$	$\cdots$	$e_{k-1,0}$

3. For ℓ from 0 up to $2^k - 1$, pre-compute $G_\ell = \prod_{j=0}^{k-1} g_j^{\ell_j} \bmod p$, where $\ell = (\ell_{k-1} \cdots \ell_0)_2$.

4. set $A = 1$.

5. For i from $t-1$ down to 0,

   (a) Let $A = A \cdot A \bmod p$,

   (b) If $I_i \neq 0$, let $A = A \cdot G_{I_i} \bmod p$.

6. Return A as the final result.

Now we analyze the computational complexity of the algorithm. First we note that for a small k, say $k \leqq 4$, the computational cost for pre-computing G_0, G_1, ..., G_{2^k-1} is marginal when compared to the total cost for computing the product. In other words, the total computational cost is dominated by $(t + v)$ modulo multiplications invested in updating A, where v is the number of non-zero columns in the exponent array E. For e_0, e_1, $\cdots$, e_{k-1} chosen independently at random, one expects that $(\frac{1}{2})^k t$ of the columns in E are zeros. Thus the expected number of modulo multiplications is $(2 - (\frac{1}{2})^k)t$.

For $k = 2$, the expected computational cost is $1.75t$ modulo multiplications. This is roughly equivalent to 1.17 modulo exponentiations when the standard "square-and-multiply" method is used.

Example 1 Let $k = 2$ and $t = 6$. Assume that we are given the following two exponents: $e_0 = 60 = (111100)_2$, and $e_1 = 20 = (010100)_2$. Then the exponent array E is:

E =

	I_5	I_4	I_3	I_2	I_1	I_0
e_0	1	1	1	1	0	0
e_1	0	1	0	1	0	0

Thus we have $I_0 = 0$, $I_1 = 0$, $I_2 = 3$, $I_3 = 1$, $I_4 = 3$, and $I_5 = 1$.

The pre-computation (Step 3) gives:

ℓ	0	1	2	3
G_ℓ	1	g_0	g_1	$g_0 g_1 \bmod p$

And iterations in Step 5, with $i = 5, 4, 3, 2, 1, 0$, update A by way of:

i	5	4	3	2	1	0
A	g_0	$g_0^3 g_1 \bmod p$	$g_0^7 g_1^2 \bmod p$	$g_0^{15} g_1^5 \bmod p$	$g_0^{30} g_1^{10} \bmod p$	$g_0^{60} g_1^{20} \bmod p$

The total number of modulo multiplications in Step 5 is therefore $6+4 = 10$. If we count the one modulo multiplication for computing G_3, the total number of modulo multiplications is 11. As a comparison, if the computation is done by calculating the two exponentiations separately, then the total number of modulo multiplications would be $(6+4)+(6+2)+1 = 19$, which is nearly twice as large as the number of modulo multiplications taken by the fast method.