Signcryption and Its Applications in Efficient Public Key Solutions *

Yuliang Zheng

Monash University, McMahons Road, Frankston, Melbourne, VIC 3199, Australia
Email: yuliang@mars.fcit.monash.edu.au
URL: http://www-pscit.fcit.monash.edu.au/~yuliang/

Abstract. Signcryption is a new paradigm in public key cryptography that *simultaneously* fulfills both the functions of digital signature and public key encryption in a logically single step, and with a cost *significantly* lower than that required by the traditional "signature followed by encryption" approach. This paper summarizes currently known construction methods for signcryption, carries out a comprehensive comparison between signcryption and "signature followed by encryption", and suggests a number of applications of signcryption in the search of efficient security solutions based on public key cryptography.

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.

Public key cryptography discovered nearly two decades ago [9] 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 the 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.

* An invited lecture at the 1997 Information Security Workshop (ISW'97), Lecture Notes in Computer Science, Vol.1397, pp.291-312, Springer-Verlag, 1998.

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 [30], we addressed a question on the cost of secure and authenticated message delivery, 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.* In the same paper, we also presented a positive answer to the question. In particular, we discovered 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. More specifically, it has been shown in [30] that 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 following section is an exposition on how signcryption can be implemented by the use of so-called shortened ElGamal based digital signature schemes.

2 Digital Signcryption — A More Economical Approach

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.

The rest of this section is devoted to seeking for concrete implementations of signcryption.

Since its publication in 1985, ElGamal digital signature scheme [11] has received extensive scrutiny by the research community. In addition, it has been generalized and adapted to numerous different forms (see for instance [26, 4, 23, 25] and especially [14] where an exhaustive survey of some 13000 ElGamal based signatures has been carried out.)

In [30], a method for shortening an ElGamal based signature is shown. Applying the shortening method to Digital Signature Standard (DSS) yields two different shortened signature schemes. These two schemes are summarized in Table 1, and are denoted by SDSS1 and SDSS2 respectively. 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).

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 1. Examples of Shortened and Efficient Signature Schemes

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 2. Parameters for Signcryption

2.1 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 (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 1. 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 [22] and SPEED [31]. 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 2.

As shown in Table 3, the signcryption and unsigncryption algorithms are remarkably simple. The signcrypted version of a message m is composed of three parts c, r and s from which the recipient can recover the original message. Note that in the table, $\in_R$ indicates an operation that chooses an element uniformly at random from among a set of elements.

Signcryption of m by Alice the Sender		**Unsigncryption of (c, r, s) by Bob the Recipient**
$x \in_R [1, \ldots, q-1]$ $(k_1, k_2) = hash(y_b^x \bmod p)$ $c = E_{k_1}(m)$ $r = KH_{k_2}(m)$ $s = x/(r + x_a) \bmod q$ if SDSS1 is used, or $s = x/(1 + x_a \cdot r) \bmod q$ if SDSS2 is used.	$\Rightarrow c, r, s \Rightarrow$	$(k_1, k_2) = hash((y_a \cdot g^r)^{s \cdot x_b} \bmod p)$ if SDSS1 is used, or $(k_1, k_2) = hash((g \cdot y_a^r)^{s \cdot x_b} \bmod p)$ if SDSS2 is used. $m = D_{k_1}(c)$ Accept m only if $KH_{k_2}(m) = r$

Table 3. Example Implementations of Signcryption

With the signcryption algorithm described in the left column of the table, the output of the one-way hash function $hash$ used in defining $(k_1, k_2) = hash(y_b^x \bmod p)$ should be sufficiently long, say of at least 128 bits, which guarantees that both k_1 and k_2 have at least 64 bits. Also note that in practice, (k_1, k_2) can be defined in a more liberal way, such as $(k_1, k_2) = y_b^x \bmod p$ and $(k_1, k_2) = fd(y_b^x \bmod p)$, where fd denotes a folding operation.

The unsigncryption algorithm works by taking advantages of the property that $g^x \bmod p$ can be recovered from r, s, g, p and y_a by Bob.

In the following we use SCS1 and SCS2 to denote the two signcryption schemes constructed from SDSS1 and SDSS2 respectively.

2.2 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

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

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 committment/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 defining $r = KH_{k_2}(m, y_b, etc)$.

2.3 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 [16]. In addition, Lenstra's new method for constructing sub-groups based on cyclotomic polynomials [17] 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 re-define 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 $k = hash((y_a^{(s+1)\cdot x_b}) \cdot (g^{r \cdot s \cdot x_b}) \bmod p)$.

3 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:

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

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 4 summarizes the advantage of SCS1 and SCS2 over discrete logarithm based signature-then-encryption, while Table 5 summarizes that over RSA based signature-then-encryption.

security parameters			saving	saving in
$\|p\|$	$\|q\|$	$\|KH.(\cdot)\|$	average comp. cost	comm. overhead
512	144	72	58%	70.3%
1024	160	80	58%	81.0%
1536	176	88	58%	85.3%
2048	192	96	58%	87.7%
4096	256	128	58%	91.0%
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 4. Saving of Signcryption over Signature-Then-Encryption Using Schnorr Signature and ElGamal Encryption

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

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 several exponentials with the same modulo which has been attributed to Shamir (see [11] as well as Algorithm 14.88 on Page 618 of [21]), $g^s \cdot y_a^r \bmod p$ 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

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%
1024	160	80	32.3%	88.3%
1536	176	88	50.3%	91.4%
2048	192	96	59.4%	93.0%
4096	256	128	72.9%	95.0%
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 5. Advantage of Signcryption over Signature-Then-Encryption based on RSA with *Small Public Exponents*

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.

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.

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, $hash$ is a one-way hash function and KH is a keyed hash function. 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.

3.2 A Comparison with Signature-Then-Encryption Using RSA

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 [8, 7]). 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 with the same modulo 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.

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 [28]. 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 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 [19].

4.1 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 [9].)

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".

4.2 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-encrytpion", 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 [10]. 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 [27] 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.

4.3 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-encrytpion" does not.

In terms of past recovery, one may view "signature-then-encrytpion" 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).

4.4 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 5.3.

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 [6] 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.

4.5 "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 [19]. Such restrictions seem to be inherent with cryptographic protocols based on the Diffie-Hellman public key cryptosystem [9]. A recent example of such protocols is an Internet key agreement protocol based on ISAKMP and Oakley [13].

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

Various Dimensions	Signcryption	Sign-then-Enc with a Static Key	Sign-then-Enc
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 4.5)	Yes

Table 6. Other Aspects of Signcryption v.s. Signature-then-Encryption

4.6 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 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.

5 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 have access to Alice's signcryption algorithm and 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) to masquerade Alice in creating a 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 (who may be any party other than Alice and Bob) to gain any partial information on the contents of a signcrypted text.

The following sub-sections are devoted to discussions of the security of the signcryption schemes SCS1 and SCS2.

5.1 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.

SDSS1 and SDSS2 can be shown to be unforgeable. Therefore we conclude that both signcryption schemes SCS1 and SCS2 are unforgeable against adaptive attacks, under the assumption that the keyed hash function behaves like a random function.

5.2 Confidentiality

Next we consider the confidentiality of message contents. We use SCS1 as an example, as discussions for SCS2 are similar. Given the signcrypted text (c, r, s) of a message m from Alice, an 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 [29, 3]. (See also earlier work [33].) Using a similar argument as that in [29, 3], we can show in the following that for C_{kh}, it is computationally infeasible for an adaptive attacker to gain any partial information on m.

5.3 Non-repudiation

As discussed in Section 4, 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 three possible repudiation settlement procedures, each requiring a different level of trust on the judge's side.

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.

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 [5].)

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.

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 [2] 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.

[2] 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.

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:

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

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. An example of such protocols is a 4-move protocol recently proposed in [2].

Properties of such a zero-knowledge repudiation settlement procedure 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 [15].

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

A zero-knowledge argument suffices for most cryptographic applications, including repudiation settlement in signcryption.

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 [18] 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".

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 7 details how to modify SCS1 into a multi-recipient signcryption scheme which we call SCS1M. SCS2M is constructed similarly from SCS2, and hence not shown in the Table. 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.

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 in the computation of r_i (see Section 2.2).

Next we examine the efficiency of the schemes.

Comparison with a Discrete Logarithm Based Scheme — We compare SCS1M and SCS2M with the signature-then-encryption for multiple recip-

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 $1 \leq i \leq t$, 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 $1 \leq i \leq t$, 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 7. SCS1M — A Signcryption Scheme for Multiple Recipients

ients based on Schnorr signature and ElGamal encryption. 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 eveluation 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.

Comparison with RFC1421 — RFC1421 [18] relies on RSA. As the discrete logarithm and factorization problems are of equal complexity with our current knowledge, we assume that $|n_a| = |n_b| = |p|$. First, two observations on computational costs can be made:

(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 ommitted here.

7 Applications of Signcryption

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. Currently, we are working on signcryption based efficient solutions to the following problems: (1) secure and authenticated key establishment in a single small data packet [32], (2) secure multicasting over the Internet [20], (3) authenticated key recovery [24], (4) secure ATM networks [12], and (5) secure and light weight electronic transaction protocols.

Acknowledgment

Part of this work was completed while I was on sabbatical leave at the University of Tokyo. I would like to take this opportunity to thank Professor Hideki Imai for his hospitality.

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 communcations.

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 signryption 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.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In Advances in Cryptology - CRYPTO'96 (Berlin, New York, Tokyo, 1996) vol. 1109 of Lecture Notes in Computer Science Springer-Verlag pp. 1–15.
2. Bellare, M., Jakobsson, M., Yung, M.: Round-optimal zero-knowledge arguments based on any one-way function. In Advances in Cryptology - EUROCRYPT'97 (Berlin, Tokyo, 1997) vol. 1233 of Lecture Notes in Computer Science Springer-Verlag pp. 280–305.
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In Proceedings of the First ACM Conference on Computer and Communications Security (New York, November 1993) The Association for Computing Machinery pp. 62–73.
4. Brickell, E., McCurley, K.: Interactive identification and digital signatures. AT&T Technical Journal (1991) 73–86.
5. Chaum, D.: Zero-knowledge undeniable signatures. In Advances in Cryptology - EUROCRYPT'90 (Berlin, New York, Tokyo, 1990) vol. 473 of Lecture Notes in Computer Science Springer-Verlag pp. 458–464.
6. Chen, M., Hughes, E.: Protocol failures related to order of encryption and signature: Computation of discrete logarithms in RSA groups April 1997. (Draft).
7. Coppersmith, D.: Finding a small root of a univariate modular equation. In Advances in Cryptology - EUROCRYPT'96 (Berlin, Tokyo, 1996) vol. 1070 of Lecture Notes in Computer Science Springer-Verlag pp. 153–165.
8. Coppersmith, D., Franklin, M., Patarin, J., Reiter, M.: Low-exponent RSA with related messages. In Advances in Cryptology - EUROCRYPT'96 (Berlin, Tokyo, 1996) vol. 1070 of Lecture Notes in Computer Science Springer-Verlag pp. 1–9.
9. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory **IT-22** (1976) 472–492.
10. Diffie, W., van Oorschot, P., Wiener, M.: Authentication and authenticated key exchange. Designs, Codes and Cryptography **2** (1992) 107–125.

11. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **IT-31** (1985) 469–472.
12. Gamage, C., Zheng, Y.: Secure high speed networking with ABT and signcryption 1997. (submitted for publication).
13. Harkins, D., Carrel, D.: The resolution of ISAKMP with Oakley February 1997. Internet-draft (draft-ietf-ipsec-isakmp-oakley-03.txt).
14. Horster, P., Michels, M., Petersen, H.: Meta-ElGamal signature schemes. In Proceedings of the second ACM Conference on Computer and Communications Security (New York, November 1994) ACM pp. 96–107.
15. Kilian, J., Petrank, E.: An efficient non-interactive zero-knowledge proof system for NP with general assumption. Electronic Colloquium on Computational Complexity **Reports Series** (1995).
16. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation **48** (1987) 203–209.
17. Lenstra, A.: Using cyclotomic polynomials to construct efficient discrete logarithm cryptosystems over finite fields. In Information Security and Privacy – Proceedings of ACISP'97 (Berlin, New York, Tokyo, 1997) vol. 1270 of Lecture Notes in Computer Science Springer-Verlag pp. 127–138.
18. Linn, J.: Privacy enhancement for internet electronic mail: Part I: Message encryption and authentication procedures. RFC 1421 IETF 1993.
19. Matsumoto, T., Imai, H.: On the key predistribution systems: A practical solution to the key distribution problem. In Advances in Cryptology - CRYPTO'87 (Berlin, New York, Tokyo, 1987) vol. 239 of Lecture Notes in Computer Science Springer-Verlag pp. 185–193.
20. Matsuura, K., Zheng, Y., Imai, H.: Analysis of and improvements on CBT multicast key-distribution 1997. (submitted for publication).
21. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press 1996.
22. National Bureau of Standards:. Data encryption standard. Federal Information Processing Standards Publication FIPS PUB 46 U.S. Department of Commerce January 1977.
23. 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.
24. Nishioka, T., Matsuura, K., Zheng, Y., Imai, H.: A proposal for authenticated key recovery system. In Proceedings of 1997 Joint Workshop on Information Security and Cryptography (JW-ISC'97) (Seoul, 1997) KIISC (Korea) pp. 189–196.
25. Nyberg, K., Rueppel, R.: Message recovery for signature schemes based on the discrete logarithm problem. Designs, Codes and Cryptography **7** (1996) 61–81.
26. Schnorr, C. P.: Efficient identification and signatures for smart cards. In Advances in Cryptology - CRYPTO'89 (Berlin, New York, Tokyo, 1990) vol. 435 of Lecture Notes in Computer Science Springer-Verlag pp. 239–251.
27. Shamir, A.: How to share a secret. Communications of the ACM **22** (1979) 612–613.
28. Shamir, A.: RSA for paranoids. CryptoBytes **1** (1995) 1–4.
29. Zheng, Y.: Improved public key cryptosystems secure against chosen ciphertext attacks. Technical Report 94-1 University of Wollongong Australia January 1994.
30. Zheng, Y.: Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption). In Advances in Cryptology - CRYPTO'97

(Berlin, New York, Tokyo, 1997) vol. 1294 of Lecture Notes in Computer Science Springer-Verlag pp. 165–179.
31. Zheng, Y.: The SPEED cipher. In Proceedings of Financial Cryptography'97 (Berlin, New York, Tokyo, 1997) vol. 1318 of Lecture Notes in Computer Science Springer-Verlag.
32. Zheng, Y., Imai, H.: Compact and unforgeable session key establishment over an ATM network. In Proceedings of IEEE Infocom'98 IEEE.
33. Zheng, Y., Seberry, J.: Immunizing public key cryptosystems against chosen ciphertext attacks. IEEE Journal on Selected Areas in Communications **11** (1993) 715–724.

This article was processed using the LaTeX macro package with LLNCS style