CHAPTER 3
CRYPTOGRAPHY SYSTEMS

3.1. Cryptography Basics

Cryptography is about the design and analysis of mathematical techniques that enable secure communications in the presence of malicious adversaries, achieving the following fundamental objectives of secure communications:

1. **Confidentiality:** keeping data secret from all but those authorized to see it.
2. **Data integrity:** ensuring that data has not been altered by unauthorized means.
3. **Data origin authentication:** corroborating the source of data.
4. **Entity authentication:** corroborating the identity of an entity.
5. **Non-repudiation:** preventing an entity from denying previous commitments or actions.

cryptographic systems can be broadly divided into two kinds; *symmetric – key schemes*, and a*symmetric – key schemes* [4].

In the symmetric – key schemes, the communicating entities first agree upon keying material that is both secret and authentic. Subsequently, may use a symmetric-key encryption scheme such as DES, RC4, or AES to achieve confidentiality. Also may use MAC algorithm such as HMAC to achieve data integrity and data origin authentication.

The major advantage of symmetric-key cryptography is high efficiency; however, there are significant drawbacks to these systems. One primary drawback is *the key distribution problem,* the requirement for a channel that is both secret and authenticated for the distribution of keying material.

In some applications this distribution may be conveniently done by using a physically secure channel such as a trusted courier. Another way is to use the services of an on – line trusted third-party who initially establishes secret keys with all the entities in a network and subsequently uses these keys to securely distribute keying material to communicating entities when required. Solutions such as these may be well-suited to environments where there is an accepted and trusted CA, but are clearly impractical in applications such as email over the Internet.

A second drawback is *the key management problem*, in a network of N entities, each entity may have to maintain different keying material with each of the other $N - 1$ entities. This problem can be alleviated by using the services of an on – line trusted third party that distributes keying material as required, thereby reducing the need for entities to securely store multiple keys. Since keying material is shared between two (or more) entities, also symmetric-key techniques cannot be used to devise elegant *digital signature schemes,* that provide non-repudiation services. This is because it is impossible to distinguish between the actions taken by the different holders of a secret key [4,9].

In this way, public-key cryptography provides elegant solutions to the three problems with symmetric – key cryptography, namely key distribution, key management, and the provision of non – repudiation. It must be pointed out that, although the requirement for a secret channel for distributing keying material has been eliminated, implementing a Public – key Infrastructure (PKI) for distributing and managing public keys can be a formidable challenge in practice. Also, public-key operations are usually significantly slower than their symmetric-key counterparts. Hence, hybrid systems that benefit from the efficiency of symmetric-key algorithms and the functionality of Public – key algorithms are often used.

3.2. Public-key Cryptography

Public-key cryptography was introduced in 1975 by Diffie, Hellman and Merkle to address the aforementioned shortcomings of symmetric-key cryptography. In contrast to symmetric – key schemes, public-key schemes require only that the communicating entities exchange keying material that is authentic (but not secret). Each entity selects a single key pair (e, d) consisting of a public key e, and a related private key d (that the entity keeps secret). The keys have the property that it is computationally infeasible to determine the private key only from knowledge of the public key [9]. Number – theoretic problems whose intractability form the basis for the security of commonly used Public – key schemes are:

1. The Integer Factorization Problem, whose hardness is essential for the security of RSA public-key encryption and signature schemes.
2. The Discrete Logarithm Problem, whose hardness is essential for the security of the ElGamal Public – key encryption and signature schemes and their variants such as DSA.
3. The Elliptic Curve Discrete Logarithm Problem, whose hardness is essential for the security of all elliptic curve cryptographic schemes.

3.2.1. RSA Algorithm

The scheme developed in 1977 by Rivest, Shamir, and Adleman makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n where a typical size of n is 1024 bits, or 309 decimal digits, then n is less than 2^{1024}. That is, the block size must be less than or equal to $log_2(n)$; in practice, the block size is i bits, where $2^i < n \leq 2^{i+1}$. Encryption and decryption are of the following form, for some plaintext block m and ciphertext block c:

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

$$m = c^d \bmod n = (m^e)^d \bmod n = m^{ed} \bmod n$$

both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d. Thus, this is a public-key encryption algorithm with a public key of $PU = \{e, n\}$ and a private key of $PU = \{d, n\}$. For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:

1. It is possible to find values of e, d, n such that $m^{ed} \bmod n = m$ for all $m < n$.

2. It is relatively easy to calculate mod $m^e mod\ n$ and c^d for all values of $m < n$.
3. It is infeasible to determine d given e and n.

RSA Key Generation

The public key consists of a pair of integers (n, e) where the RSA *modulus* n is a product of two randomly generated and secret primes p and q of the same bitlength where $n = pq$. The *encryption exponent* e is an integer satisfying $1 < e < \varphi$ and $gcd\,(e, \varphi) = 1$ where $\varphi = (p-1)(q-1)$. The private key d, also called the *decryption exponent*, is the integer satisfying $1 < d < \varphi$ and $ed \equiv 1\,(mod\ \varphi)$. the problem of determining the private key d from the public key (n, e) is computationally equivalent to the problem of determining the factors p and q of n; this problem is known as the *integer factorization problem* (IFP) [9].

RSA Encryption Scheme

RSA encryption and signature schemes use the fact that

$$m^{ed} \equiv m\,(mod\ n)$$

for all integers m. The encryption and decryption procedures for the (basic) RSA Public – key encryption scheme are presented as ciphertext $c \equiv m^e (mod\ n)$. Decryption works because $c^d \equiv (m^e)^d \equiv m\,(mod\ n)$.The security relies on the difficulty of computing the plaintext m from the ciphertext $c = m^e (mod\ n)$ and the public parameters n and e. This is the problem of finding e^{th} roots modulo n and is assumed to be as difficult as the integer factorization problem [4,9].

RSA Signature Scheme

The signer of a message m first computes its MD $h = H(m)$ using a cryptographic *hash function* H, where h serves as a short fingerprint of m. Then, the signer uses his private key d to compute the e^{th} root s of h modulo n: $s = h^d\ mod\ n$. Note that $s^e \equiv (h^d)^e\ mod\ n \equiv h\,(mod\ n)$. The signer transmits the message m and its signature s to a verifying party. Then this party recomputes the MD $h = H(m)$, recovers a message digest $h' = s^e\ mod\ n$ from s, and accepts the signature as being valid for m provided that $h = h'$. The security relies on the inability of a forger to compute e^{th} roots modulo n.

3.2.2. DL Systems

The first DL system was the key agreement protocol proposed by DH in 1976. In 1984, ElGamal described DL public-key encryption and signature schemes. Then, many variants of these schemes have been proposed. Here presenting the basic ElGamal Public – key encryption scheme and DSA [9].

DL Key Generation

In discrete logarithm systems, a key pair is associated with a set of public domain parameters (p, q, g). Here, p is a prime, q is a prime divisor of $p-1$, and $g \in [1, p-1]$ has order q. A private key is an integer x that is selected uniformly at random from the

interval $[1, q-1]$, and the corresponding public key is $y = g^x \bmod p$. The problem of determining x given domain parameters (p, q, g) and y is the DLP.

DL Encryption Scheme

In the encryption and decryption procedures for the basic ElGamal public key encryption scheme, if y is the intended recipient's public key, then a plaintext m is encrypted by multiplying it by $y^k \bmod p$ where k is randomly selected by the sender. The sender transmits this product $c_2 = my^k \bmod p$ and also $c_1 = g^k \bmod p$ to the recipient who uses her private key to compute

$$c_1^x \equiv g^{kx}(\bmod p) \equiv y^k \ (\bmod p)$$

and divides c_2 by this quantity to recover m

$$m = c_2 c_1^{-x} \ (\bmod p)$$

an eavesdropper who wishes to recover m needs to calculate $y^k \bmod p$. This task of computing $y^k \bmod p$ from the domain parameters $(p, q, g), y$, and $c_1 = g^k \bmod p$ is called the Diffie – Hellman Problem. The DHP is assumed to be as difficult as the Discrete Logarithm Problem [4,9,15].

DL signature scheme

The DSA was proposed in 1991 by the U.S. NIST and was specified in a U.S. Government Federal Information Processing Standard (FIPS – 186) called the Digital Signature Standard.

An entity A with private key x signs a message by selecting a random integer k from the interval $[1, q-1]$, and computing $T = g^k \bmod p, r = T \bmod q$ and

$$s = k^{-1}(h + xr) \bmod q \tag{3.1}$$

where $h = H(m)$ is the message digest. A's signature on m is the pair (r, s). To verify the signature, an entity must check that (r, s) satisfies equation (3.1). Since the verifier knows neither A's private key x nor k, this equation cannot be directly verified. Note, however, that equation is equivalent to

$$k \equiv s^{-1}(h + xr) \bmod q \tag{3.2}$$

rising g to both sides of equation (3.2) yields the equivalent congruence

$$T \equiv g^k \ (\bmod p) \equiv g^{hs^{-1}} \ y^{rs^{-1}} \bmod p$$

the verifier can therefore compute T, recompute r' and then check the $r' = T \bmod q$ with the received r, if $r = r'$ accept the signature.

3.2.3. ECC

In 1985, Victor Miller and N. Koblitz [1,8] independently, produced a public key cryptography analogue of the ElGamal schemes in which the group Z_p^* is replaced by the group of points on an elliptic curve defined over finite field. The main attraction of ECC over competing technologies such as RSA and DSA is that the best algorithm known for solving the underlying hard mathematical problem in ECC, the Elliptic Curve Discrete Logarithm Problem (ECDLP) takes fully exponential time on reverse the hard mathematical problems in RSA and DSA (the integer factorization problem, and the discrete logarithm problem) take sub – exponential time [9,15].

ECDLP

The security of ECC depends on the difficulty of ECDLP. Let P and Q be two points on an elliptic curve such that $kP = Q$, where k is a scalar. Given P and Q, it is computationally infeasible to obtain k, if k is sufficiently large. k is the discrete logarithm of Q to the base P. Hence the main operation involved in ECC is point multiplication. i.e. multiplication of a scalar k with any point P on the curve to obtain another point Q on the curve [16].

EC Key Generation

Let E be an elliptic curve defined over a finite field F_P. Let G be a point in $E(F_P)$, and suppose that G has prime order n. Then the cyclic subgroup of $E(F_P)$ generated by G is

$$\langle G \rangle = \{\infty, G, 2G, 3G, \ldots, (n-1)G\}.$$

the prime p, the equation of the elliptic curve E, and the point G and its order n, are the public domain parameters. A private key is an integer d that is selected uniformly at random from the interval $[1, n-1]$, and the corresponding public key is $Q = dG$. The problem of determining d given the domain parameters and Q is the ECDLP [9].

EC Encryption Scheme

The encryption and decryption procedures for the elliptic curve are analogue of the basic ElGamal encryption scheme. A plaintext m is first represented as a point M on the elliptic curve (see Appendix A), and then encrypted by adding it to kQ where k is a randomly selected integer from the interval $[1, n-1]$, and Q is the intended recipient's public key.

The sender transmits the points $C_1 = kG$ and $C_2 = M + kQ$ to the recipient who uses her private key d to compute

$$dC_1 = d(kG) = k(dG) = kQ$$

and thereafter recovers $M = C_2 - kQ$. An eavesdropper who wishes to recover M needs to compute kQ. This task of computing kQ from the domain parameters, Q, and $C_1 = kP$, is the elliptic curve analogue of the Diffie-Hellman problem [4,9].

3.2.4. Elliptic Curve Digital Signature Algorithm (ECDSA)

The ECDSA [8] is the elliptic curve analogue of the DSA. It is the most widely standardized elliptic curve – based signature scheme, appearing in the ANSI X9.62, FIPS 186 – 2, Institute of Electrical and Electronics Engineers (IEEE) 1363 – 2000 and ISO / IEC 15946 – 2 tandards [15,16]. In the following steps to sign the message m:

Step 1: Select a Random number $k \in [1, n-1]$

Step 2: Compute Point $kG = (x, y)$ and $r = x \bmod n$, if $r = 0$ then goto Step 1

Step 3: Compute $t = k^{-1} \bmod n$

Step 4: Compute $e = \text{SHA}-1(m)$, where $\text{SHA}-1$ denotes the 160 bit hash function

Step 5: Compute $s = k^{-1}(e + d_a.r)$ mod n, if $s = 0$ go to Step 1

Step 6: The signature of message m is the pair (r, s)

Step 7: The message m and the signature (r, s)

to verify the signature, the recipient does the following (Note that the recipient knows the domain parameters and sender's public key Q)

Step 1: Verify r and s are integers in the range $[1, n-1]$

Step 2: Compute $e = \text{SHA}-1\,(m)$

Step 3: Compute $w = s^{-1} \bmod n$

Step 4: Compute $u_1 = e.w$ and $u_2 = r.w$

Step 5: Compute Point $X = (x_1, y_1) = u_1 G + u_2 Q$

Step 6: If $X = 0$, then reject the signature Else compute $v = x_1 \bmod n$

Step 7: Accept the signature if $v = r$

3.3. Key Size Comparisons

The parameter sizes providing comparable levels of security for cryptographic systems, The parameter sizes, also called key sizes, that provide equivalent security levels for RSA, DSA, and ECC systems and symmetric-key encryption as are listed in table 3.1. By a *security level* of k bits that means the best algorithm known for breaking the system takes approximately 2^k steps [9].

The comparisons in table 3.1 demonstrate that smaller parameters can be used in ECC than with RSA and DL systems at a given security level. The difference in parameter sizes is especially pronounced for higher security levels. The advantages that can be gained from smaller parameters include speed (faster computations) and smaller keys and certificates. In particular, Public – key operations (such as signature generation and decryption) for ECC are many times more efficient than RSA and DL Public – key operations. Public – key operations (such as signature verification and encryption) for ECC are many times more efficient than for DL systems. Public – key operations for RSA are expected to be somewhat faster than for ECC if a small encryption exponent e is selected for RSA. The

advantages offered by ECC can be important in environments where processing power, storage, bandwidth, or power consumption is constrained [9].

Table 3.1 Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis

Symmetric Scheme (key size in bits)		ECC-Based Scheme (size of n in bits)	RSA/DSA (modulus size in bits)
(SKIPJACK)	56	112	512
(Triple – DES)	80	160	1024
(AES – Small)	112	224	2048
(AES – Medium)	128	256	3072
(AES – Large)	92	384	7680

3.4. Signcryption Scheme

3.4.1. Introduction

In order to send a confidential letter in a way that it cannot be forged, it has been a common practice for the sender of the letter to sign it, put it in an envelope and then seal it before handing it over to be delivered.

Discovering Public key cryptography has made communication between people who have never met before over an open and insecure network, in a secure and authenticated way possible. Before sending a message, the sender has to do the following:

1. Sign it using a Digital Signature scheme.
2. Encrypt the message and the signature using a private key encryption algorithm under randomly chosen message encryption key.
3. Encrypt the random message encryption key using the receivers public key.
4. Send the message.

this approach is knows as signature – then – encryption. The main disadvantage of this approach is that, digitally signing a message and then encrypting it, consumes more machine cycles and bloats the message by introducing extended bits to it. Hence, decrypting and verifying the message at the receivers end, a lot of computational power is used up. Thus you can say that the cost of delivering a message using signature – then – encryption is in effect the sum of the costs of both digital signatures and public key encryption.

In 1997, Zhen proposed a new cryptography primitive called *signcryption* [17], Signcryption is a new paradigm in public key cryptography that simultaneously fulfils both

signature and encryption in a logically single step depending on DLP, and guaranteeing unforgeability, non – repudiation, integrity and confidentiality of data with communication overhead and computational costs significantly lower than that required by the traditional approach signature followed by encryption. So Signcryption is a combination of the encryption algorithms (have been discussed in the previous sections) and a digital signature algorithm.

3.4.2. Shortening ElGamal Based Signatures

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 3.2, and are denoted by Shorten Digital Signature Standard (SDSS1 and SDSS2) respectively [18].

Table 3.2 Examples of Shortened and Efficient Signature Schemes

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

Where :

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 $[1, .., p - 1]$ (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 ($y_a = g^{x_a} mod\ p$).

3.4.3. Implementing Signcryption with Shortened Signature

An interesting characteristic of a shortened ElGamal based signature scheme is that although $g^x\ mod\ p$ is not explicitly contained in a signature (r, s), it can be recovered from r, s and other public parameters [17].

Table 3.3 Parameters for Signcryption

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 ($y_a = g^{x_a} mod\ p$)
Bob's keys: x_b-Bob's private key, chosen uniformly at random from $[1,\ldots.,q-1]$ y_b- Bob's public key ($y_b = g^{x_b} mod\ p$)

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.

Encrypting a message m with a key k, typically in the cipher block chaining, is indicated by $E_k(m)$, while decrypting a ciphertext c with k is denoted by $D_k(c)$. In addition, 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, a one-way hash function, it is computationally infeasible to find a pair of messages that are hashed to the same value. 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. 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 [17,18].

Signcryption Implementation

Alice has chosen a private key x_a from$[1,\ldots.,q-1]$, and made public her matching public key $y_a = g^{x_a} mod\ p$. Similarly, Bob's private key is x_b and his matching public key is $y_b = g^{x_b} mod\ p$. Alice has chosen a random integer x from $[1,\ldots.,q-1]$ (private value) and computing the key using hash function and Bob's public key

$$k = (k_1, k_2) = Hash({y_b}^x mod\ p)$$

should be sufficient length at least 128 bits, split the key into k_1 and k_2 ,using k_1 for encryption $c = E_{k_1}(m)$, and k_2 to define $r = KH_{k_2}(m)$. Computing the value of s according to the shortening method to Digital Signature Standard,

if using SDSS1 $\quad s = x/(r + x_a) mod\ q$.

if using SDSS2 $\quad s = x/(1 + x_a.r) mod\ q$.

At Alice:

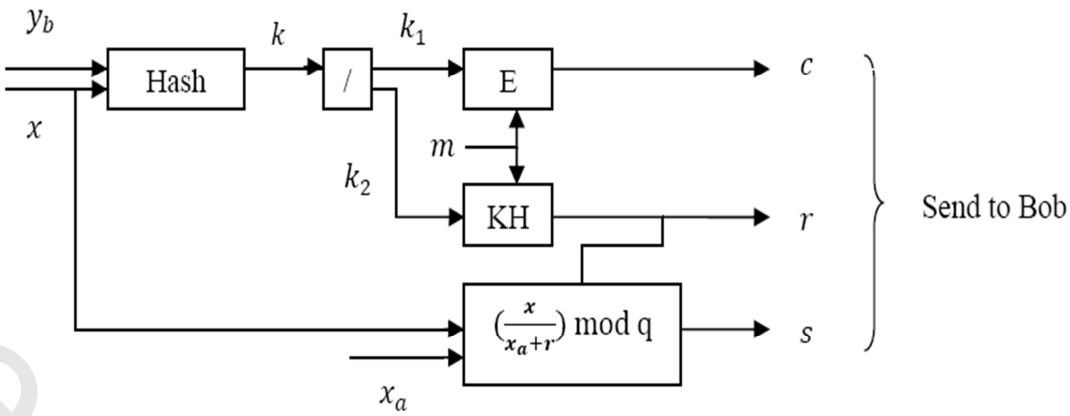

Figure 3.1 Signcryption Block Diagram

Unsigncryption Implementation

The unsigncryption algorithm works by taking advantages of the property that $g^x mod\ p$ can be recovered from r, s, g, p and y_a by Bob's private key [18]. Bob computes the key,

if using SDSS1 $k = (k_1, k_2) = Hash((y_a . g^r)^{s.x_b}\ mod\ p)$.

if using SDSS2 $k = (k_1, k_2) = Hash((g . y_a{}^r)^{s.x_b}\ mod\ p)$.

split k_1 and k_2 using k_1 for decryption $m = D_{k_1}(c)$. To check the integrity of decrypted message, using k_2 to recomputed $r' = KH_{k_2}(m)$ and compare it with the received, if $r' = r$ accept m.

At Bob:

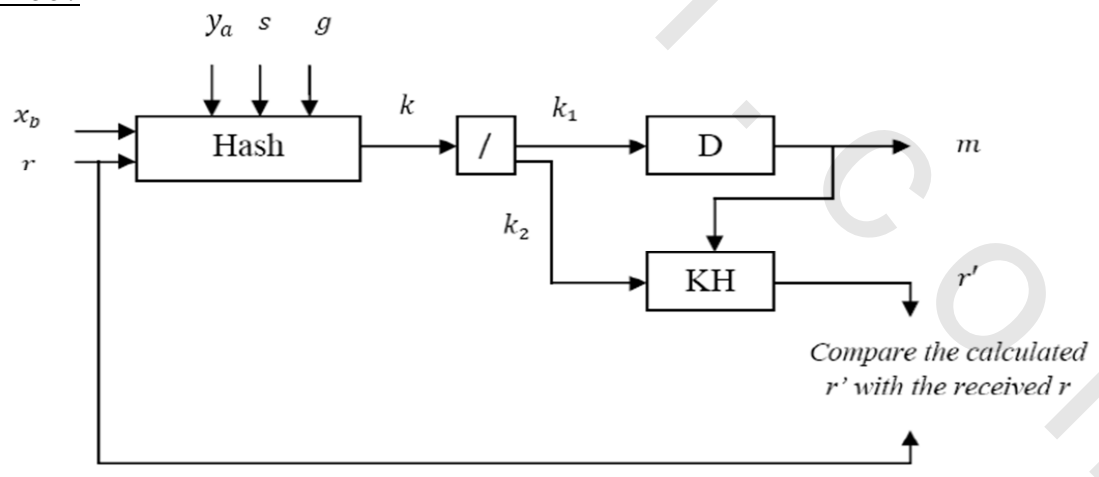

Figure 3.2 Unsigncryption Block Diagram

3.4.4. Cost of Signcryption vs. 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:

$$Cost(signcryption) < Cost(signature) + Cost(encryption).$$

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 [17].

As shown, table 3.4 summarizes the advantage of Signcryption Constructed form Shorten Digital Signature Standard (SCS1) using SDSS1 and SCS2 using SDSS2 over discrete logarithm based signature – then – encryption, while table 3.5 summarizes that over RSA based signature – then – encryption.

Table 3.4. Saving of Signcryption over Signature –Then–Encryption Using Schnorr Signature and ElGamal Encryption

Security parameters $\|p\|$	$\|q\|$	$\|KH(.)\|$	Saving in average computational cost	Saving in communication 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 computational cost $= \frac{(5.17-2.17)\text{modular exponentiations}}{5.17 \text{ modular exponentiation}} = 58\%$

Saving in communication cost $= \frac{|\text{hash}(.)|+|q|+|p|-(|\text{KH}(.)|+|q|)}{|\text{hash}(.)|+|q|+|p|}$

Table 3.5. Advantages of Signcryption over Signature–Then–Encryption based on RSA with Small Public Exponents

Security parameters $\|p\|(=\|n_a\|=\|n_b\|)$	$\|q\|$	$\|KH(.)\|$	Saving average computational cost	Saving in communication 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%

Saving in average computational cost = $\frac{(5.17-2.17)\text{modular exponentiations}}{5.17\text{ modular exponentiation}} = 58\%$

Saving in communication cost = $\frac{|\text{hash}(.)|+|q|+|p|-(|\text{KH}(.)|+|q|)}{|\text{hash}(.)|+|q|+|p|}$

3.5. Signcryption Scheme over Elliptic Curve

As saw Zheng's scheme was based on the DLP; in 1998, Zheng and Imai proposed another signcryption scheme based on the ECDLP that achieved similar functionality of signcryption scheme and named ECSCS1 and ECSCS2. They were based on shortened variants of the Elliptic Curve Digital Signature Standard (ECDSS) (SECDSS1 and SECDSS2) . Table 3.6 summarized an elliptic curve version of the Digital Signature Standard together with its shortened variants, and table 3.7 summarized the parameters for elliptic curve based on signcryption scheme [19].

Table 3.6. Elliptic Curve DSS and Its Shortened and Efficient Variants [19]

Shortened schemes	Signature (r, s) on a message m	Verification of signature	Length of signature
ECDSS	$r = (vG)\text{mod}q$ $s = ((hash(m) + v_a r)/v)\text{mod}\, q$	$K = s'(hash(m)G + rp_a)$ Where $s' = \frac{1}{s}$ Check whether $K \bmod q = r$	$2\|q\|$
SECDSS1	$r = hash(vG, m)$ $s = (v/(r + v_a)\text{mod}q$	$K = s(p_a + rG)$ Check whether $hash(K, m) = r$	$\|hash(.)\| + \|q\|$
SECDSS2	$r = hash(vG, m)$ $s = (v/(1 + v_a.r)\text{mod}q$	$K = s(G + rp_a)$ Check whether $hash(K, m) = r$	$\|hash(.)\| + \|q\|$

Table 3.7 Parameters Elliptic Curve Signcryption

Public parameters to all: C: an elliptic curve over $GF(p^m)$, either with $p > 2^{150}$ and $m = 1$ or $p = 2$ and $m > 150$ (public to all). q: a large prime whose size is approximately of $\|p^m\|$ (public to all). G: a point with order q, chosen randomly from the points on C (public to all). $hash$: a one-way hash function (public to all). v: a number chosen uniformly at random from $[1, \dots, q-1]$
Alice's keys: v_a: Alice's private key, chosen uniformly at random from $[1, \dots, q-1]$ P_a: Alice's public key ($P_a = v_a G$, a point on C).
Bob's keys: v_b: Bob's private key, chosen uniformly at random from $[1, \dots, q-1]$ P_b: Bob's public key ($P_b = v_b G$, a point on C).

Alice has a message m to send to Bob. Alice signcrypts m as follows so that the effect is similar to signature then encryption:

Signcryption Implementation

Alice implements the message m as a point on the elliptic curve M, choose a random value v from the interval $[1, \dots, q-1]$ (private value) and computing the key using hash function and Bob's public key

$$k = (k_1, k_2) = Hash(vP_b \bmod p)$$

should be sufficient length at least 128 bits, split the key into k_1 and k_2 ,using k_1 for encryption using elliptic curve cryptography scheme $c = E_{k_1}(M)$, and k_2 to define $r = KH_{k_2}(M)$. Computing the value of s according to the shortening method to Digital Signature Standard,

if using SECDSS1 $s = (v/(r + v_a) \bmod q.$

if using SECDSS2 $s = (v/(1 + v_a.r) \bmod q.$

send the values c, r, s to Bob.

Unsigncryption Implementation

Since Bob receive the values c, r, s, computes the key, evaluate $u = sv_b \bmod q$,

if using SECDSS1 $k = (k_1, k_2) = Hash(uP_a + urG).$

if using SECDSS2 $k = (k_1, k_2) = Hash(uG + urP_a).$

split k_1 and k_2 using k_1 for decryption $M = D_{k_1}(c)$. To check the integrity of decrypted message, using k_2 to recomputed $r' = KH_{k_2}(M)$. and compare it with the received, if $r' = r$ accept M.

3.5.1. Advantages of Signcryption over Signature-Then-Encryption based on ECC

When comparing signcryption over elliptic curve with signature – then – encryption, signcryption on the curves represents a 58% saving in computational cost, with ECSCS1 and ECSCS2,where the number of point multiples is one for the process of signcryption and two for that of unsigncryption respectively. Applying Shamir's technique, one reduces the computational cost of unsigncryption from 2 multiples to 1.17 on average [19,20]. The total average computational cost for signcryption is therefore 2.17 point multiples. This represents a

$$(5.17 - 2.17)/5.17 = 58\%$$

and reduction in average computational cost, and saving a 40% in communication overhead as follows

$$\frac{hash(.)| + 2|q| - (|KH(.)| + |q|)}{hash(.)| + 2|q|} = \frac{|q|}{\left(\frac{1}{2}\right)|q| + 2|q|} = 40\%$$

Note: Communication Overhead Calculation will be discussed later in Chapter 4.

3.6. Signcryption for Multiple Recipients

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 [17].

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 approach signature – then – encryption, the standard practice has been to encrypt the message by encryption key using each recipient's public key and attach the resulting ciphertext to the signed and also encrypted message. 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" as shown in the following procedures.

Key Generation

Alice needs to send message m for multiple recipients. Assume that there are t recipients $R_1, R_2, \ldots\ldots, R_t$. The private key of a recipient R_i is a number xi chosen uniformly and independently at random from $[1, \ldots\ldots, q-1]$, and his matching public key is

$$y_i = g^{x_i} \bmod p.$$

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.

Signcryption by Alice for Multi – Recipients

An input to this Signcryption algorithm for multi-recipients consists of a message m to be sent to t recipients $R_1, \ldots\ldots\ldots, R_t$ 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 from $[1, \ldots\ldots, q-1]$, calculate $h = KH_k(m)$, and encrypt m by $c = E_k(m, h)$.

2. Create a signcrypted text of k for each recipient $i = 1, \ldots\ldots, t$
   - Pick a random number v_i from $[1, \ldots\ldots\ldots, q-1]$ 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.
   - $c_i = E_{k_{i,1}}(k)$.

- $r_i = KH_{k_{i,2}}(m, h)$.
- $s_i = \frac{v_i}{r_i + x_a} mod\ q$.

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

Unsigncryption by each recipient

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

1. Find out (c, c_i, r_i, s_i) in $(c, c_1, r_1, s_1, \ldots\ldots, c_t, r_t, s_t)$.
2. $k_i = hash\ ((y_a . g^{r_i})^{s_i . x_i}\ mod\ p)$. 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.

3.6.1. Comparison between Signcryption Scheme and Signature – then – Encryption based on (DL and RSA)

The following figure shows the format of signcrypted text for multiple recipients.

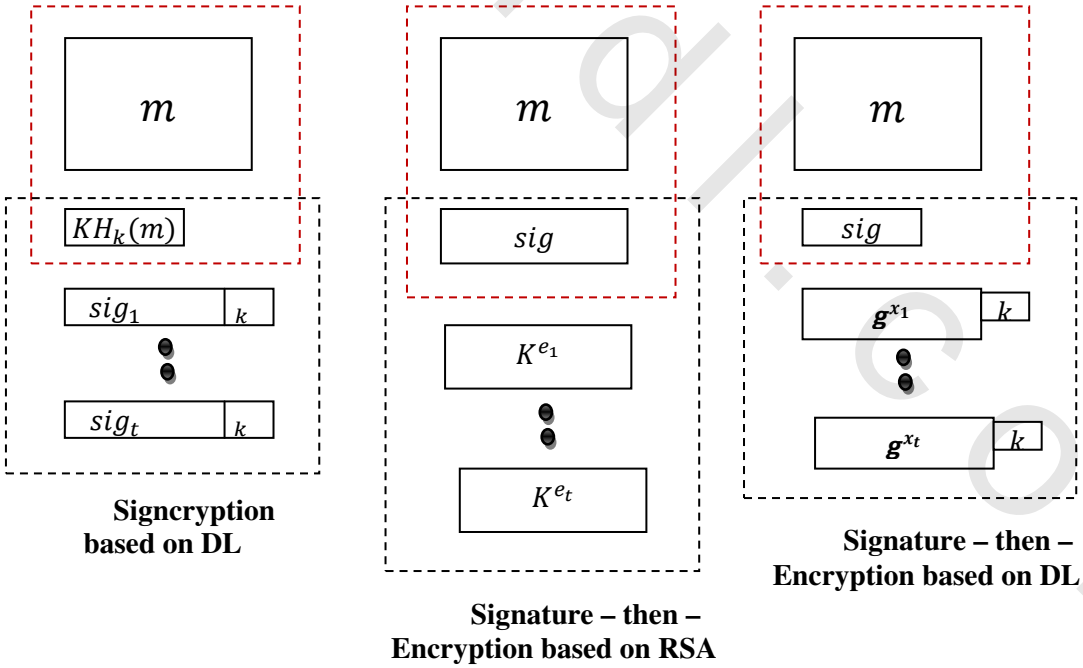

Figure 3.3 Output Formats of Signcryption and Signature-then-Encryption for Multiple Recipients [17]

Saving in Communication Overhead

The comparison between signcryption scheme for multiple recipients (SCS1M and SCS2M) with the traditional approach, signature – then – encryption based on DL. As can be seen in table 3.8, saving by SCS1M and by SCS2M in communication overhead is measured as the number of bits for the sender reduce from $t.(|k| + |p|) + |KH| + |q|$ to $t.(|k| + |KH| + |q|) + |KH|$ where $|p|$ is in general far larger than $|KH| + |q|$ (compare $|p|$ = 512 with $|KH|$ = 72 and $|q|$= 144), so, saving in communication overhead is significant when compared with signature – then – encryption based on DL [17,18].

When comparing between SCS1M and SCS2M and the traditional approach, signature – then – encryption based on RFC1421, the communication overhead of RFC1421 expands a message by $|n_a| + \sum_{i=1,\ldots t} |n_i|$ bits, which is a number of times larger than $t.(|k| + |KH| + |q|) + |KH|$, the communication overhead of SCS1M and SCS2M.

Table 3.8. Saving in Comm. Overhead of Signcryption for Multi-Recipients

Scheme	Communication Overhead
Signature – then – Encryption based on DL	$t.(\|k\| + \|p\|) + \|KH\| + \|q\|$
Signature – then – Encryption based on RSA (RFC1421)	$\|n_a\| + \sum_{i=1,\ldots t} \|n_i\|$
Signcryption (SCS1M – SCS2M)	$t.(\|k\| + \|KH\| + \|q\|) + \|KH\|$

Saving in Computational Cost

When comparing signcryption scheme for multiple recipients (SCS1M and SCS2M) with the signature – then – encryption for multiple recipients based on DL. As can be seen in table 3.9, saving by SCS1M and by SCS2M in computational cost can be summarized as the number of modular exponentiations is reduced for the sender from $2t + 1$ to t and for each recipient, from 2.17 to 1.17. And when comparing SCS1M and SCS2M with the traditional approach based on RFC1421, in which an RSA based signature – then – encryption for multiple recipients is specified. As seen in table 3.9, saving in computational costs for the sender is approximately the same and for each recipient reduce from 2 to 1.17.

Table 3.9. Saving in Comp. Cost of Signcryption for Multi-Recipients

Operation / Scheme	Participant	EXP	DIV	MUL	ADD	ENC	ENC_{short}	DEC	DEC_{short}	HASH
Signature – then – Encryption based on DL	**Alice**	$2t$+1	–	1	1	1	t	–	–	1
	Recipient R_i	2.17	–	1	–	–	–	1	1	1
Signature – then – Encryption based on RSA (RFC1421)	**Alice**	t+1	–	–	–	1	–	–	–	1
	Recipient R_i	2	–	–	–	–	–	1	–	1
Signcryption	**Alice**	t	t	–	t	1	t	–	–	$2t$+1
	Recipient R_i	1.17	–	2	–	–	–	1	1	3

Where

EXP : The number of Modular Exponentiations .

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.

Finally, from above cost comparison was found, the signcryption schemes for multi-recipients (SCS1M and SCS2M) [17] are more efficient than the signature – then – encryption based on DL in both terms communication over head and computational cost. But with respect to signature – then – encryption based on RSA, the SCS1M and SCS2M are share a similar computational cost but significantly lower communication overhead.