Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

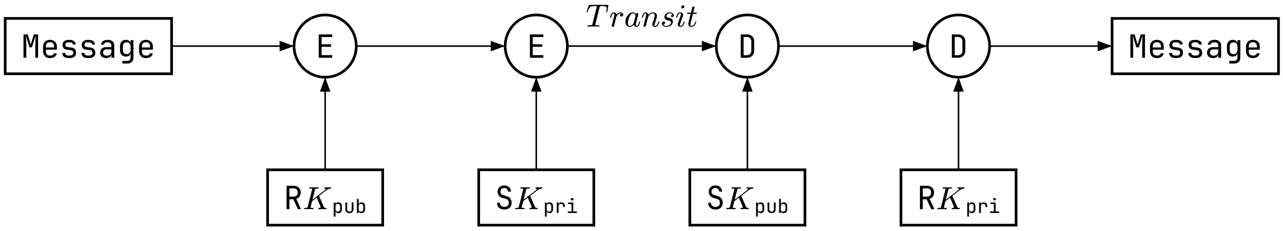
B.Tech. Winter Semester 2024-25
School Of Computer Science and Engineering (SCOPE)

Notes
Cryptography and Network Security

Apurva Mishra: 22BCE2791
Date: CAT - II

Contents

1 Module 3: Asymmetric Encryption Algorithm and Key Exchange



1.1 Principles

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes

1.2 RSA

1.2.1 Steps

1. Choose two large primes:

$$\begin{gathered} P, Q \\ N = P * Q \end{gathered} \tag{1}$$

2. Choose public and private key:

$$\begin{gathered} K_{\text{pub}} \mid K_{\text{pub}} \text{ is not factor of } \phi(N) \\ K_{\text{pri}} \mid (K_{\text{pri}} * K_{\text{pri}}) \bmod \phi(N) = 1 \end{gathered} \tag{2}$$

3. Encrypt:

$$CT = PT^{K_{\text{pub}}} \bmod N \tag{3}$$

4. Decrypt:

$$PT = CT^{K_{\text{pri}}} \bmod D \tag{4}$$

1.3 ElGamal

1. Choose public numbers such that:
   - α, q are prime
   - α is primitive root of q

$$\alpha, q \tag{5}$$

2. A: Compute
   - Private Key: X_{A}
   - Public Key : $\{q, \alpha, Y_{\text{A}}\}$

$$
\begin{aligned}
& X_{\mathrm{A}} \mid X_{\mathrm{A}} \in (1, q-1) \\
& Y_{\mathrm{A}} = \alpha^{\mathrm{X}_{\mathrm{A}}} \bmod q
\end{aligned} \tag{6}
$$

3. B
   - Message: $M \mid M \in [1, q-1]$
   - Random : $k \mid k \in [1, q-1]$
4. Encrypt (C_1, C_2):

$$
\begin{aligned}
& C_1 = \alpha^k \bmod q \\
& C_2 = KM \bmod q
\end{aligned} \tag{7}
$$

5. A: Decrypt

$$
\begin{aligned}
& K = {C_1}^{X_{\mathrm{A}}} \bmod q \\
& M = C_2 K^{-1} \bmod q
\end{aligned} \tag{8}
$$

> If a message must be broken up into blocks and sent as a sequence of encrypted blocks, a unique value of k should be used for each block. If k is used for more than one block, knowledge of one block M1 of the message enables the user to compute other blocks as follows. Let

1.4 Elliptic Curve cryptography

1.5 Homomorphic Encryption and Secret Sharing

1.6 Key distribution and Key exchange protocols

1.7 Diffie-Hellman Key Exchange

1. Choose public numbers such that:
   - g is primitive root of n
   - g, n are primes

$$
g, n \tag{9}
$$

2. Choose private numbers:

$$
\begin{aligned}
& x_{\mathrm{A}} \mid x < n \\
& y_{\mathrm{B}} \mid y < n
\end{aligned} \tag{10}
$$

3. New public values:

$$
\begin{aligned}
& A = g^x \bmod n \\
& B = g^y \bmod n
\end{aligned} \tag{11}
$$

4. Generate Keys User side:

$$
\begin{aligned}
K_{\mathrm{A}} &= B^{x} \bmod n \\
K_{\mathrm{B}} &= A^{y} \bmod n \\
K_{\mathrm{A}} &== K_{\mathrm{B}}
\end{aligned}
\tag{12}
$$

1.8 Man-in-the-Middle Attack

2 Module 4: Message Digest and Hash Functions

2.1 Requirements for Hash Functions

A Hash Function H accepts a variable length block of data M as input and produces a fixed size result $h = H(M)$ referred to as a **hash value** or **hash code**.

A **Cryptographic Hash Function** for which it is computationally infesible to find:
1. M which maps to a predefined h
2. (M_1, M_2) which map to same h

2.2 Security of Hash Functions

2.3 Message Digest (MD5)

Setup

$$
\begin{aligned}
\text{Input} &: M \\
\text{Message Length} &: 2^{64} \text{ bits} \\
\text{Output} &: 128 \text{ bits}
\end{aligned}
\tag{13}
$$

Steps:
1. **Padding**

Padding bits P:

$$P = 1 \cdot \sum_{i}^{n} 0_i \tag{14}$$

Padding is **always** added, even if:

$$O = 512 \cdot x = M + 64 \text{ bits} \mid x \in [1, \infty) \tag{15}$$

Examples: $\{10, 100, 1000\}$

Output:

$$O_1 = M + P \tag{16}$$

2. **Append Length**

$$L = \mathrm{len}(M) \mid \text{expressed in 64 bits} \tag{17}$$

Then,

$$O_2 = O_1 + L \tag{18}$$

Output:

$$
\begin{aligned}
O_2 &= O_1 + L \\
O_2 &= M + P + L
\end{aligned}
\tag{19}
$$

3. **Divide into 512 bit blocks**

$$
\begin{aligned}
O_3 &= \sum_i^n a_i \mid \text{where len(a)} == 512 \\
O_3 &= \{a_1, a_2, .., a_n\}
\end{aligned}
\tag{20}
$$

4. **Initialize Chaining Variable**

Chaining variables: $\{A, B, C, D\}$ are initialised, each 32 bits.

A	01	23	45	67
B	89	AB	CD	EF
C	FE	BC	DA	98
D	76	54	32	10

5. **Process Block**

There are **4** rounds.

Let $\{a, b, c, d\} = \{A, B, C, D\}$

Divide 512 bits in sub-blocks of 32 bits each (16 sub-blocks):

$$
a_i = \sum_{j=1}^{16} b_j \mid \text{where len(b)} == 32 \text{ bits} \tag{21}
$$

Initialize constant *t*: [u32; 64]

Then round function:

$$
abcd` = f_r(abcd, \{b_1, .., b_{16}\}, t) \tag{22}
$$

2.4 Secure Hash Function (SHA)

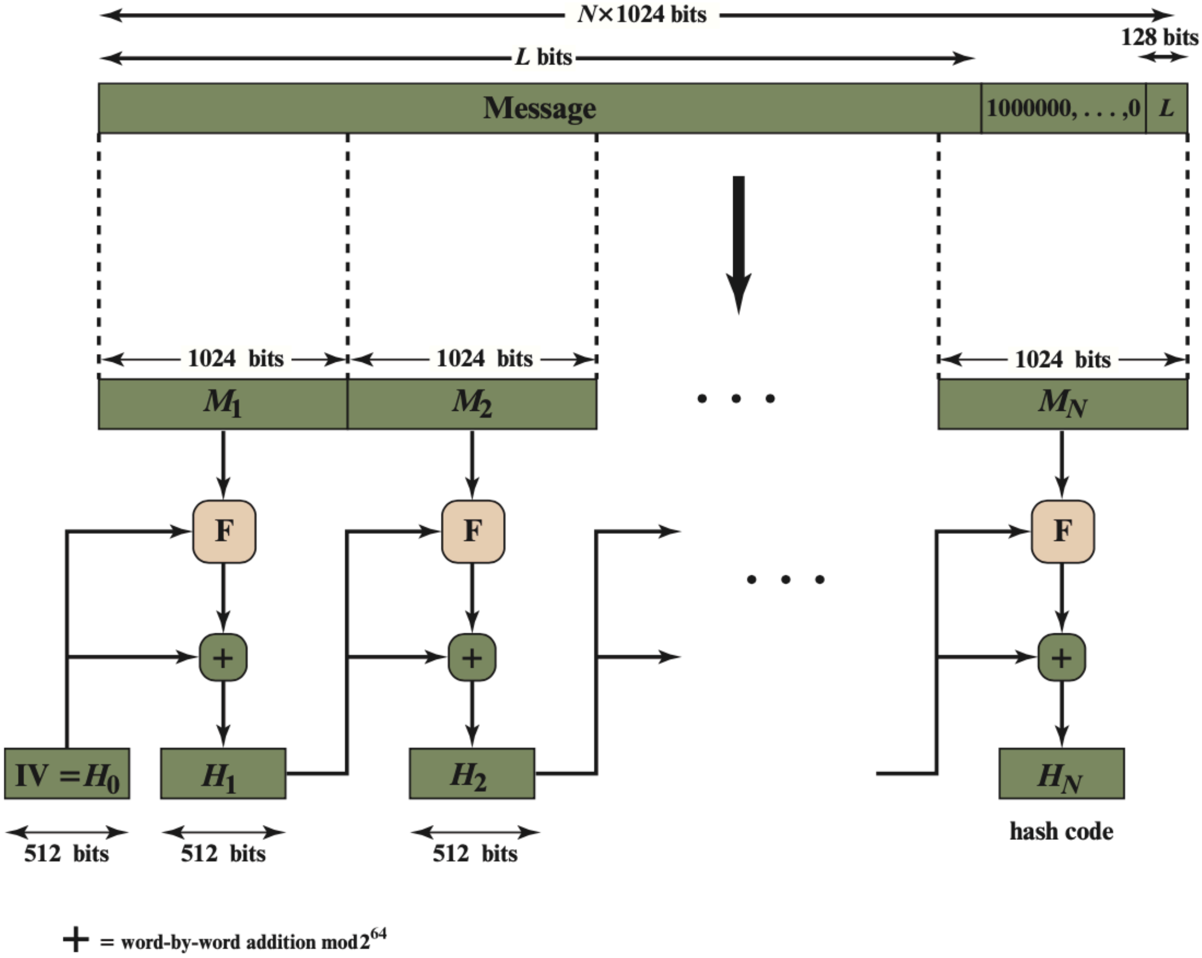


Figure 11.9 Message Digest Generation Using SHA-512

2.5 Birthday Attack

2.6 HMAC