Department of Electronics

Cryptography

Fall 2019

Hasan Mahmood

hasan@qau.edu.pk

Week 7 (16 & 17 October 2019)

The RSA Algorithm

- Alice wants to send message to Bob
- No previous contact, not pre-communications key exchange
- All information will be potentially obtained by Eve
- Is it still possible to send that is not visible to Eve
- Alice has to send a key which Eve would intercept
- She could then decrypt all the subsequent messages
- Public Key Cryptosystem introduced by Diffie and Hellman [Diffie-Hellman]
- "Factorization of integers into their prime factors hard" is used, proposed by Rivest, Shamir and Adleman in 1977 aka RSA algorithm

RSA algorithm

- Bob chooses two distinct large prime p and q and multiplies them together to form

$$n = pq.$$

- He also chooses an encryption exponent e such that

$$\gcd(e, (p-1)(q-1)) = 1.$$

- He sends the pair (n, e) to Alice but keeps the values of p and q secret
- Alice never needs to know p and q to send her message to Bob securely

The RSA algorithm

- Alice writes her message as a number m
- If m is larger then n, she breaks the message into blocks
- Now each message has length less then n (m<n)
- Alice computes

$$c \equiv m^e \pmod{n}$$

- Alice send c to Bob
- Bob knows p and q, he can compute (p-1)(q-a) to find decryption coefficient d, with

$$de \equiv 1 \pmod{(p-1)(q-1)}.$$

- Objective:

$$m \equiv c^d \pmod{n},$$

Example

- Encryption
- (5, 14) is the public key
- Take text, for example B->2
- $2^5 \pmod{14} = 32 \pmod{14} = 4 \pmod{14}$ =ciphertext=D

- Decryption
- (11, 14) My decipher key
- $4^{11} \pmod{14} = 4194304 \pmod{14} = 2 \pmod{14}$ =plaintext

Encryption

- Pick two prime numbers, p=2, q=7
- N=14, becomes mod in encryption and decryption key
- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
- Exclude all even numbers, remove 7
- Remaining numbers: 1, 3, 5, 9, 11, 13: co prime with 14
- Phi (N)=(p-1)(q-1)
- Choose a number for e, 1<e<Phi(N), Coprime with n, Phi(N)
- e=5, lock: (5, 14)

Decryption

- Choose d: de (mod phi(N))=1, 5xd (mod 6)=1
- 5, 10, 15, 20…., in mod 6: 5, 4, 3, 2, 1, 0: d=11, 5x11=55=1 (mod 6)

The RSA Algorithm

1. Bob chooses secret primes p and q and computes $n = pq$.
2. Bob chooses e with $\gcd(e, (p-1)(q-1)) = 1$.
3. Bob computes d with $de \equiv 1 \pmod{(p-1)(q-1)}$.
4. Bob makes n and e public, and keeps p, q, d secret.
5. Alice encrypts m as $c \equiv m^e \pmod{n}$ and sends c to Bob.
6. Bob decrypts by computing $m \equiv c^d \pmod{n}$.

Example, large numbers

- Choose p and q as:

$$p = 885320963, \quad q = 238855417.$$

- then,

$$n = p \cdot q = 211463707796206571$$

- Let the encryption key be:

$$e = 9007.$$

- The values of *n* and *e* are sent to Alice

Example, large numbers

- Alice computes

$$c \equiv m^e \equiv 30120^{9007} \equiv 113535859035722866 \pmod{n}$$

- She sends *c* to Bob, since Bob knows p and q, he knows (p-1)(q-1), he computes d, such that,

$$de \equiv 1 \pmod{(p-1)(q-1)}.$$

$$d = 116402471153538991.$$

$$c^d \equiv 113535859035722866^{116402471153538991} \equiv 30120 \pmod{n}$$

Treaty Verification

- Countries A and B have signed a nuclear test ban treaty
- Each wants to make sure the other doesn't test any bombs
- Country A is going to use seismic data to monitor country B
- Country A wants to put sensors in B, which then send data back to A
- Two problems

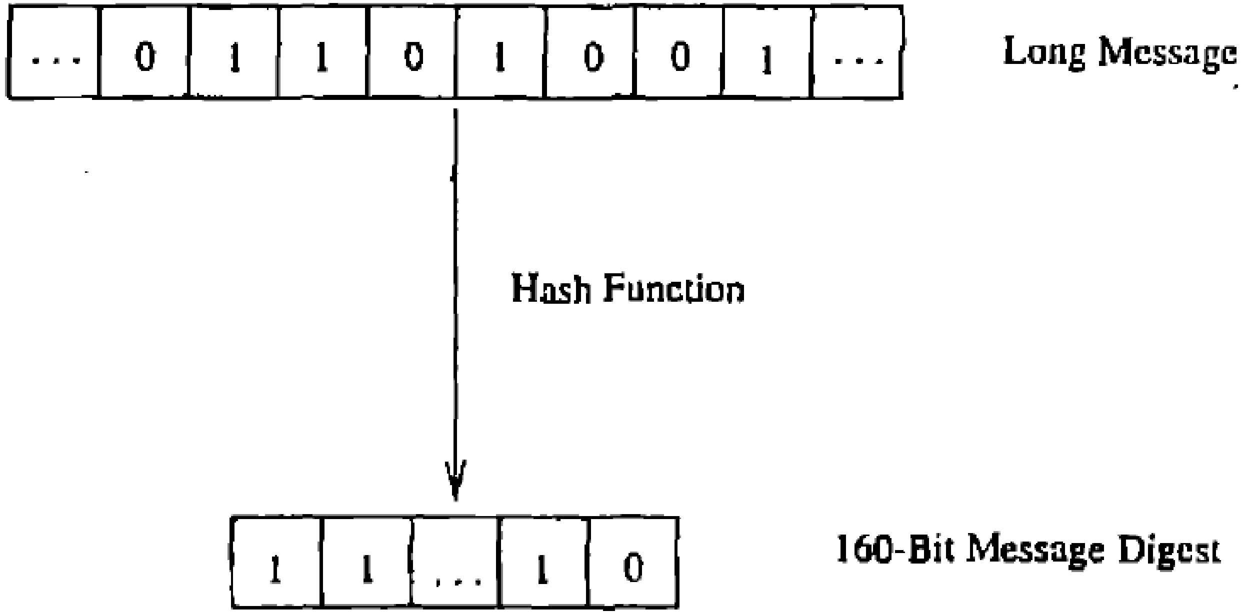
1. Country A wants to be sure that Country B doesn't modify the data.
2. Country B wants to look at the message before it's sent to be sure that nothing else, such as espionage data, is being transmitted.

Treaty Verification

- Reversing RSA
- A chooses *n=pq*, the product of two large primes, determines *e* and *d*
- *The* numbers n and e are given to B, but p, q and d are kept secret
- Sensor is temper proof, buried deep, collets data x
- Sensors uses *d* to encrypt *x* to $y=x^d$ (mod n)
- Both *x* and *y* are sent first to country *B,* which checks $y^e=x$ (mod n)
- If so, it knows that the encrypted message y corresponds to the data x
- Forwards the pair x, y to A
- A checks yat $y^e=x$ (mod n)
- If so, A is sure that the number x is not modified
- If x is chosen, then solving y^ex (mod n) for y is the same as decrypting the RSA message x.

The Hash Functions

- Non-invertibility properties
- A cryptographic hash function h takes as input a message of arbitrary length and produces as output a message digest of fixed length
- For example, 160 bits



Hash function properties

1. Given a message m, the message digest $h(m)$ can be calculated very quickly.

2. Given a y, it is computationally infeasible to find an m' with $h(m') = y$ (in other words, h is a **one-way**, or **preimage resistant**, function). Note that if y is the message digest of some message, we are not trying to find this message. We are only looking for some m' with $h(m') = y$.

3. It is computationally infeasible to find messages m_1 and m_2 with $h(m_1) = h(m_2)$ (in this case, the function h is said to be **strongly collision-free**).

Hash function

- Collision free (weakly)
- Preimage resistance
- Requirement 3 is the hardest one to satisfy
- In 2004, Wang, Feng, Lai, and Yu fond many examples of collisions for the popular hash functions MD4, MD5, HAVAL-128 and RIPEMD
- This means that a valid digital signature on one certificate is also valid for the other certifite.
- SHA-1 collision can be determined with around 2^69 calculations

Hash Example

- Efficient in computational requirements
- Start with a message *m* of arbitrary length *L*
- Break the message *m* into *b*-bit blocks, where *n* << *L*
- *Denote* these n-bit blocks by m_j, $m = [m_1, m_2, \cdots, m_l]$
- The length, *l*=ceil[*L*/*n*], last block is padded with zeroes
- We write the *j*th block m_j as row vector

$$m_j = [m_{j1}, m_{j2}, m_{j3}, \cdots, m_{jn}]$$

Example

- Stack these row vectors to form an array

$$h_i = m_{1i} \oplus m_{2i} \oplus \cdots \oplus m_{li}.$$

$$\begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{l1} & m_{l2} & \cdots & m_{ln} \end{bmatrix}$$

$$\begin{matrix} \Downarrow & \Downarrow & \Downarrow & \Downarrow \\ \oplus & \oplus & \oplus & \oplus \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow \end{matrix}$$

$$\begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} = h(m).$$

Hash example

- Input is arbitrary length message
- Output is n-bit message digest
- It is not considered cryptographically secure
- Practical cryptographic hash functions typically make use of several other bit level operations
- Need to avoid collision
- Bit rotation is used, similar to DES

Simple Hash with rotation operation

$$
\begin{bmatrix}
m_{11} & m_{12} & \cdots & m_{1n} \\
m_{22} & m_{23} & \cdots & m_{21} \\
m_{33} & m_{34} & \cdots & m_{32} \\
\vdots & \vdots & \ddots & \vdots \\
m_{ll} & m_{l,l+1} & \cdots & m_{l,l-1}
\end{bmatrix}
$$

$$
\begin{matrix}
\Downarrow & \Downarrow & & \Downarrow & \Downarrow \\
\oplus & \oplus & & \oplus & \oplus \\
\Downarrow & \Downarrow & & \Downarrow & \Downarrow
\end{matrix}
$$

$$
\begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} = h(m).
$$

The Secure Hash Algorithm (SHA)

- SHA-1 produces 160-bit hash
- The original message is broken into a set of fixed size blocks
- Last block is padded to fill out the block
- Message blocks are processed gvia sequence of rounds that use a comp[ression function h’
- Currend block is combined with the result of previous rounds
- In a good compression function, makes each input bit effect as many output bits as possible.

SHA-1

Take original meswsage and paddi it with a 1 bit followind by a sequence of 0 bits

Enough 0 bits are appended to make the mew message 64 bits shout of the next highest multiple of 512 bits in length

We append the 64 bit reprenesntation of the length T of the message

For example, if the original message has 2800 bits, we add a 1 and 207 0s to obtain a new message of length 3008=6x512-64

Since 2800=101011110000, we append 52 0s followed by this number

Message length is 3072, broken down into six blocks of length 512

SHA operations

1. $X \wedge Y$ = bitwise "and", which is bitwise multiplication mod 2, or bitwise minimum.
2. $X \vee Y$ = bitwise "or", which is bitwise maximum.
3. $X \oplus Y$ = bitwise addition mod 2.
4. $\neg X$ changes 1s to 0s and 0s to 1s .
5. $X + Y$ = addition of X and Y mod 2^{32}, where X and Y are regarded as integers mod 2^{32}.
6. $X \hookleftarrow r$ = shift of X to the left by r positions (and the beginning wraps around to the end).

SHA operations

$$f_t(B,C,D) = \begin{cases} (B \wedge C) \vee ((\neg B) \wedge D) & \text{if } 0 \le t \le 19 \\ B \oplus C \oplus D & \text{if } 20 \le t \le 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & \text{if } 40 \le t \le 59 \\ B \oplus C \oplus D & \text{if } 60 \le t \le 79 \end{cases}$$

Define constants $K_0, \ldots, K_{79}$ as follows:

$$K_t = \begin{cases} \texttt{5A827999} & \text{if } 0 \le t \le 19 \\ \texttt{6ED9EBA1} & \text{if } 20 \le t \le 39 \\ \texttt{8F1BBCDC} & \text{if } 40 \le t \le 59 \\ \texttt{CA62C1D6} & \text{if } 60 \le t \le 79 \end{cases}$$

The SHA-1 Algorithm

$$
\begin{aligned}
H_0 &= 67452301\\
H_1 &= EFCDAB89\\
H_2 &= 98BADCFE\\
H_3 &= 10325476\\
H_4 &= C3D2E1F0.
\end{aligned}
$$

The SHA-1 Algorithm

1. Start with a message m. Append bits, as specified in the text, to obtain a message y of the form $y = m_1 \| m_2 \| \cdots \| m_L$, where each m_i has 512 bits.
2. Initialize $H_0 = 67452301$, $H_1 = EFCDAB89$, $H_2 = 98BADCFE$, $H_3 = 10325476$, $H_4 = C3D2E1F0$.
3. For $i = 0$ to $L - 1$, do the following:

(a) Write $m_i = W_0 \| W_1 \| \cdots \| W_{15}$, where each W_j has 32 bits.

(b) For $t = 16$ to 79, let $W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \hookleftarrow 1$

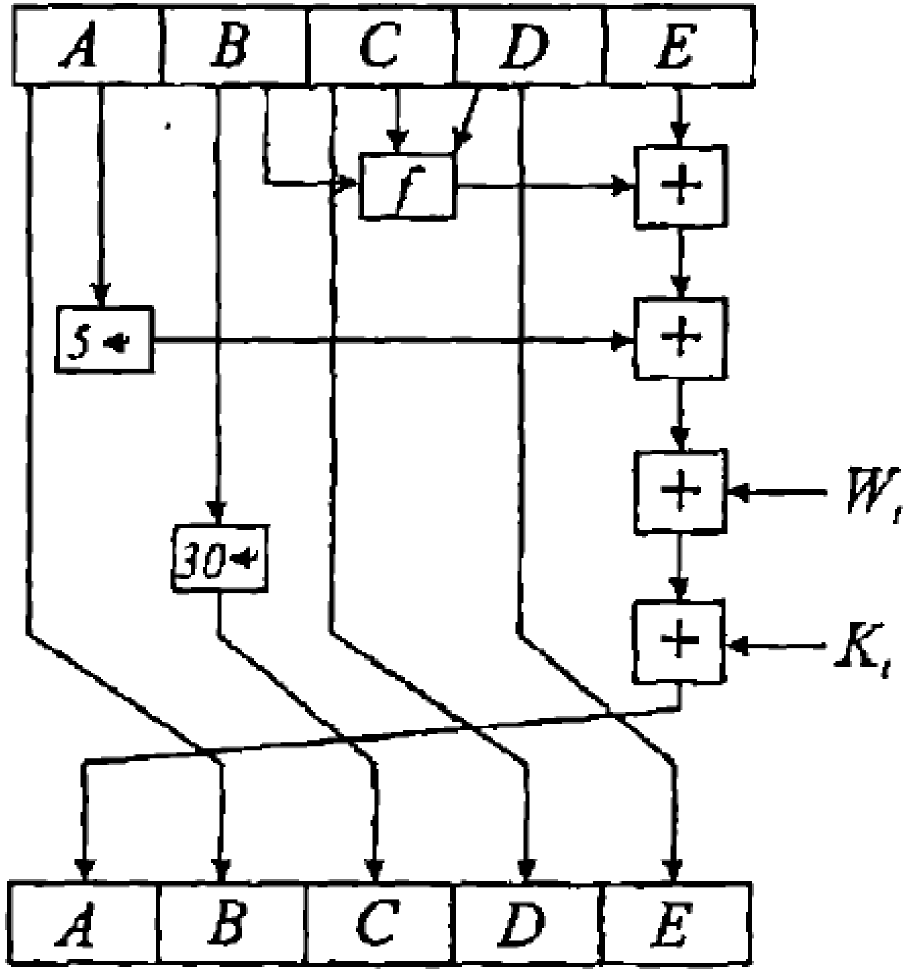
(c) Let $A = H_0$, $B = H_1$, $C = H_2$, $D = H_3$, $E = H_4$.

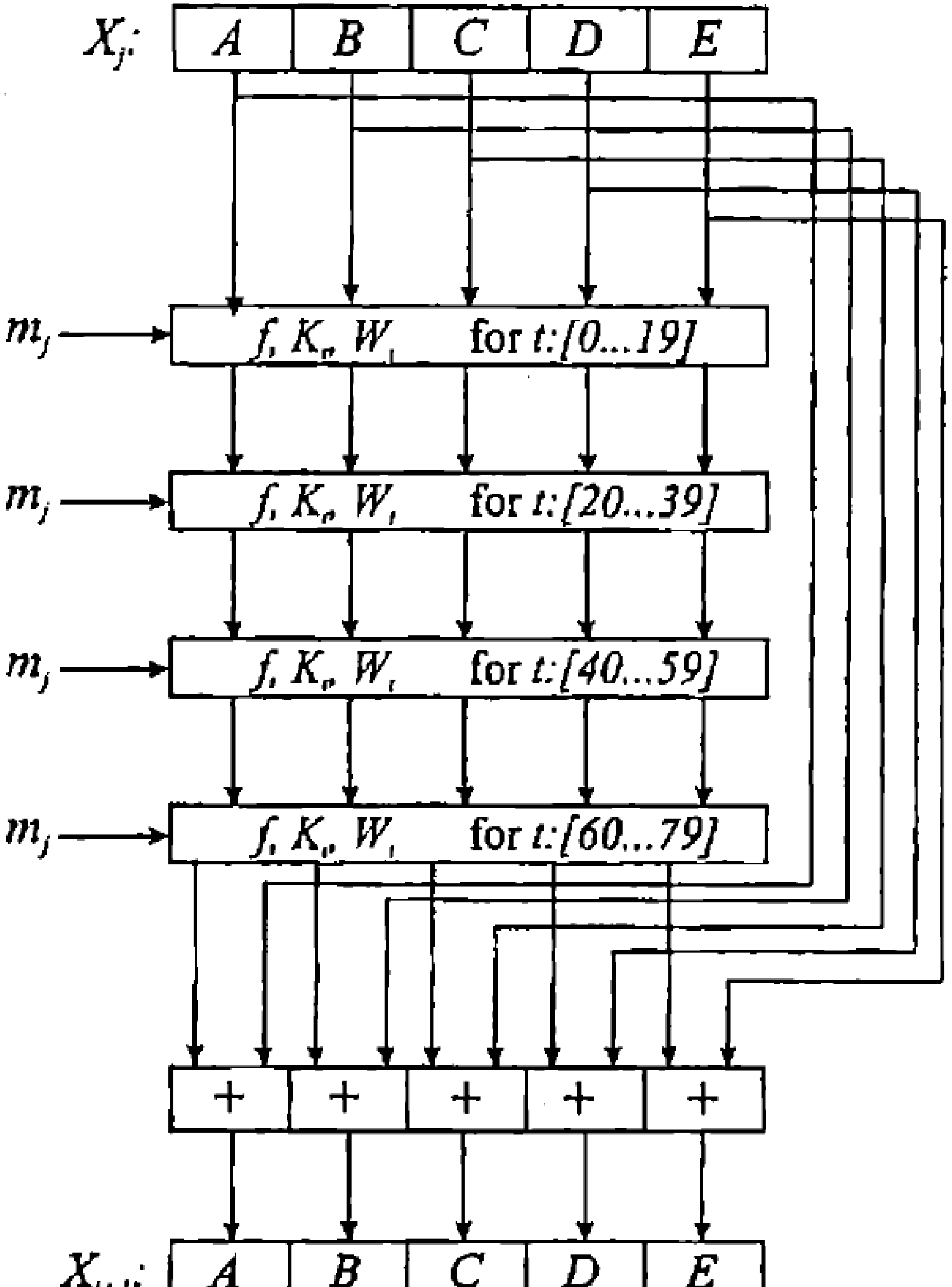
(d) For $t = 0$ to 79, do the following steps in succession: $T = (A \hookleftarrow 5) + f_t(B, C, D) + E + W_t + K_t$, $E = D$, $D = C$, $C = (B \hookleftarrow 30)$, $B = A$, $A = T$.

(e) Let $H_0 = H_0 + A$, $H_1 = H_1 + B$, $H_2 = H_2 + C$, $H_3 = H_3 + D$, $H_4 = H_4 + E$.

4. Output $H_0 \| H_1 \| H_2 \| H_3 \| H_4$. This is the 160-bit hash value.

SHA-1 Step 3


A B C D E
f
5
30
W_t
K_t
A B C D E


X_j: A B C D E
m_j
f, K_t, W_t for t:[0...19]
f, K_t, W_t for t:[20...39]
f, K_t, W_t for t:[40...59]
f, K_t, W_t for t:[60...79]
A B C D E