

Chapter FIVE

Public Key

Dr .Saad Abdual Aize
2016

Chapter five

Public Key Cryptosystem

Public – Key Cryptography Principle:

Public key encryption finds use in message authentication and key distribution:

Public-Key encryption ,first publicly proposed by Diffie and Hellman in 1976,is the first truly revolutionary advance in encryption in literally thousand of years.

Public- Key algorithms are based on mathematical functions rather than an simple operations on bit patterns. More important ,public-key encryptions is asymmetric, involving the use of two separate keys,in contrast to the symmetric conventional encryption ,which uses only one key. The use of two keys has profound consequences in the areas of confidentiality, key distribution ,and authentication.

The security of any encryption scheme depends on :

- 1- The length of the key.
- 2- The computational work involved in breaking a cipher.

A public key encryption scheme has five ingredients:

- 1-**Plaintext** :This is the readable message or data that is led into the algorithm as input.
- 2-**Encryption algorithm**: The encryption algorithm performs various transformations on the plaintext.
- 3-**Public and private key**: This is a pair of keys that have been selected so that if one is used for encryption the other is used for decryption .The exact transformation performed by the encryption algorithm depends on the public or private key that is provided as input.
- 4- **Cipher text**: This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message ,two different keys will produce two different ciphertexts.

5- Decryption algorithm: This algorithm accepts the ciphertext and the watching key and produces the original plaintext.

A general purpose public-key cryptographic algorithm relies on one key for encryption and a different but related key for decryption.

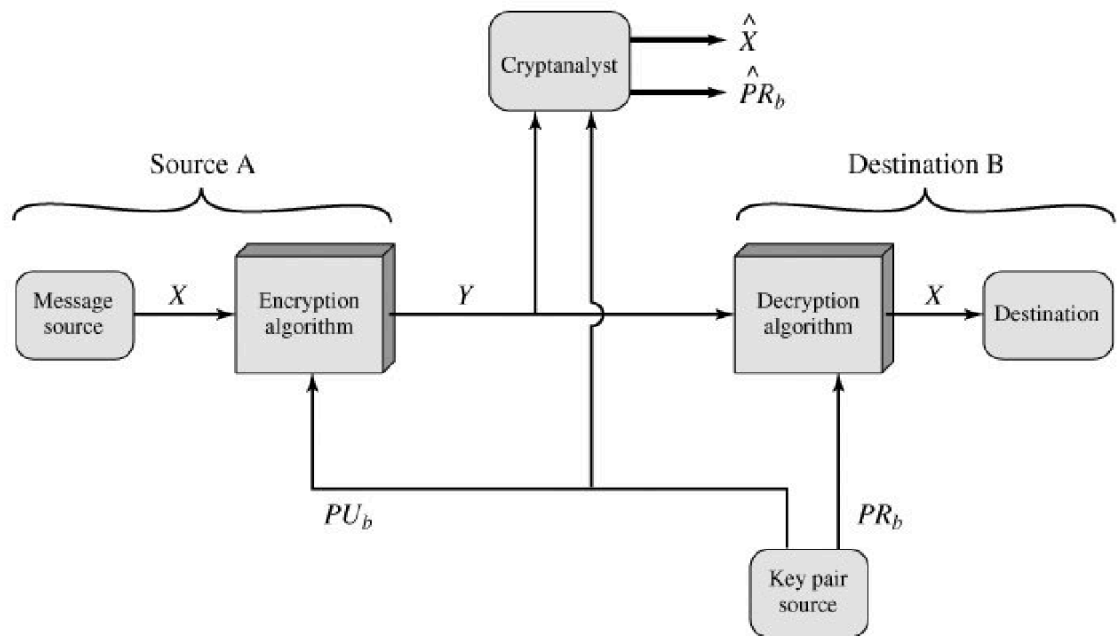


Figure : Public-Key Cryptosystem: Secrecy

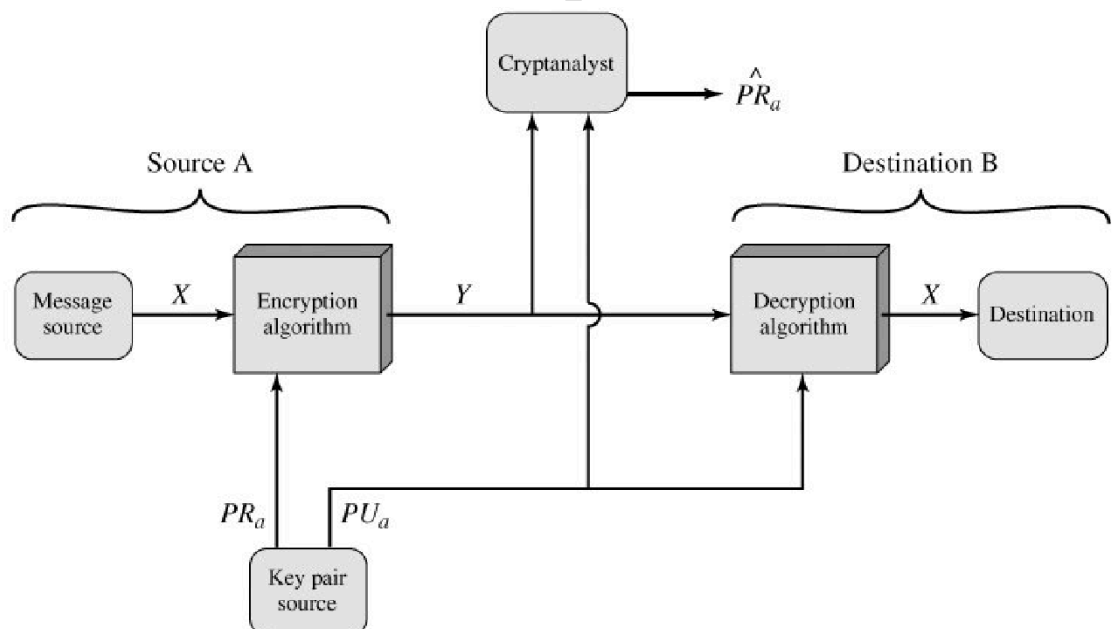


Figure : Public-Key Cryptosystem: Authentication

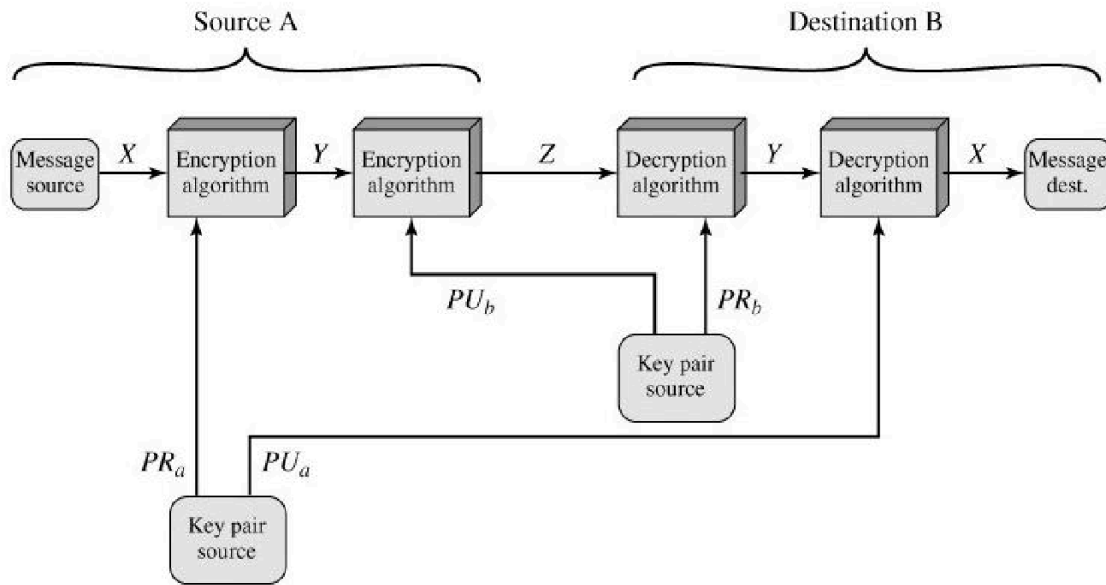


Figure:Public-Key Cryptosystem: Authentication and Secrecy

Different between conventional encryption and public key

1- Same algorithm with same key is used for encry and decry	1- One algorithm used for encry and decry with pair of key ,one for encry and anther for decry.
2- The key must be kept secret	2 – one o the two key must be secret

The essential steps are as follows:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key . The companion key is kept private.
3. If (A) wishes to send a private message to (B). (A) encrypts the message using (B) public key.
4. When (B) receives the message, (B) decrypts it using his/her private key. No other recipient can decrypt the message because only (B) knows his/her private key.

Applications for Public-Key Cryptosystem:

Depending on application, the sender uses either the sender's private key or the receiver's public key, or both, to perform some type of cryptographic function. In broad terms, we can classify the use of public-key cryptosystems into three categories:

1. Encryption/Decryption: The sender encrypts a message with the recipient's public key.
2. Digital Signature: The sender "signs" a message with his/her private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
3. Key Exchange: Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

NOTE:

Some algorithms are suitable for all three applications, whereas others can be used only for one or two of these applications. The following table indicates the applications supported by the algorithms RSA, and Diffie-Hellman. The table also includes the digital signature standard (DSS) and Elliptic-Curve cryptography.

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

Requirements for Public-Key Cryptography:

Diffie and Hellman postulated this system without demonstrating that such algorithms exist. However, they did lay out the conditions that such algorithms must fulfill:

1. It is computationally easy for a party B to generate a pair (public key KU_b , private key KR_b).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding cipher text:

$$C = E_{KU_b}(M)$$

3. It is computationally easy for the receiver B to decrypt the resulting cipher text using the private key to recover the original message:
 $M = D_{KR_b}(C) = D_{KR_b}[E_{KU_b}(M)]$
4. It is computationally infeasible for an opponent, knowing the public key, KU_b , to determine the private key KR_b .
5. It is computationally infeasible for an opponent, knowing the public key, KU_b , and a cipher text, C, to recover the original message, M.

We can add a sixth requirements that, although useful, is not necessary for all public key applications.

6. Either of the two related keys can be used for encryption, with other key used for decryption:

$$M = D_{KR_b}[E_{KU_b}(M)] = D_{KU_b}[E_{KR_b}(M)]$$

Public-Key Cryptography Algorithms:

The two most widely used public-key algorithms are RSA and Diffie-Hellman. We will discuss the two algorithms:

The RSA Public-Key Encryption Algorithm.

One of the first public-key schemes was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adelman at MIT and first published in 1978.

RSA is a block cipher in which the plain text and cipher text are integers between 0 and $n-1$ for some n . Encryption and decryption are of the following form, for some plain text block M and cipher text 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 values of n and e , and only the receiver knows of d . This is a public key encryption algorithm with a public key of $KU = \{e, n\}$ and a private key of $KR = \{d, n\}$. For this algorithm to be satisfactory for public key encryption, the following requirements must be met:

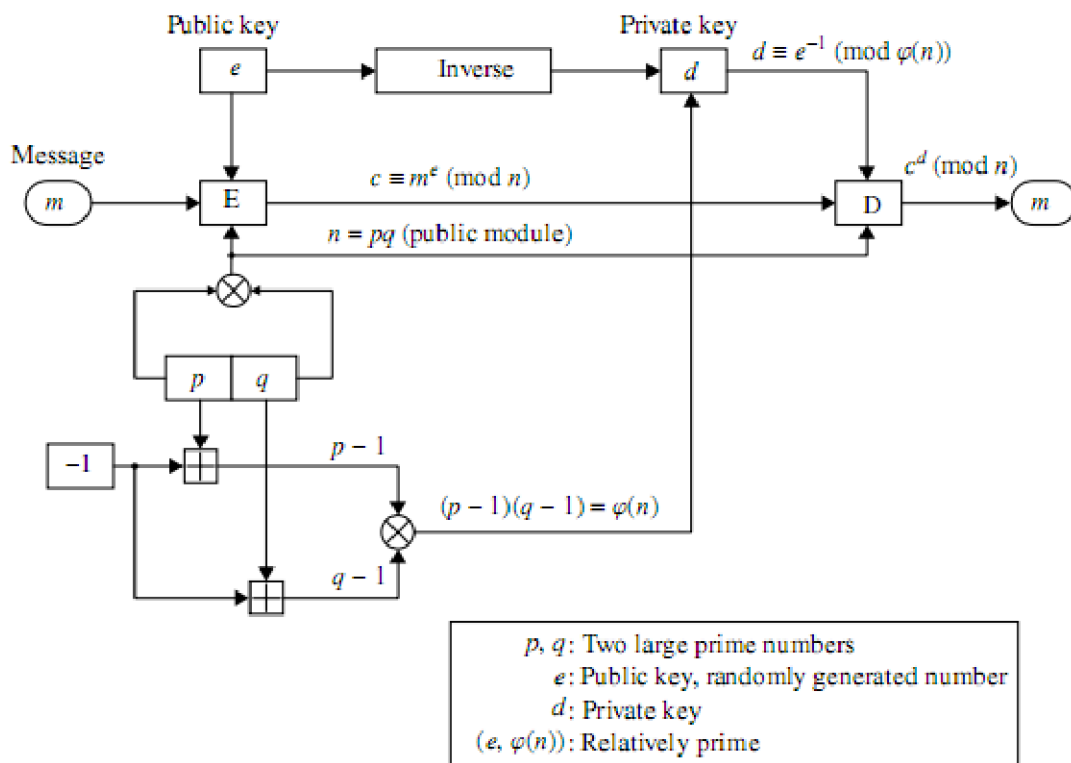
1. It is possible to find values of e, d, n such that:

$$M^{ed} = M \bmod n \text{ for all } M < n.$$

2. It is relatively easy to calculate M^e and C^d for all values of $M < n$.
3. It is infeasible to determine d given e and n .

NOTE:

The first two requirements are easily met. The third requirement can be met for a large values of e and n .



.Figure: RSA public-key cryptosystem for encryption/decryption

KEY GENERATION

Select p, q	p and q are both prime
Calculate $n = p * q$	
Calculate $\Phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\Phi(n), e) = 1; 1 < e < \Phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\Phi(n)}$
Public Key	$KU = \{e, n\}$
Private Key	$KR = \{d, n\}$

ENCRYPTION

Plain text = $M < n$

Cipher text = $C = M^e \pmod{n}$

DECRYPTION

Plain text = $M = C^d \pmod{n}$

Most discussions of the cryptanalysis of RSA have focused on the task of Factoring (n) into its two prime factors. For a large (n) with large prime factors, factoring is a hard problem, but not as hard as it used to be. Larger key sizes must be used. Currently a lazy bit key size (about 300 decimal digits) is considered strong enough.

Example: Confidentiality (Secrecy)

Let $p = 7$, $q = 11$, so $n = 77$ and $\phi(n) = 60$

Alice's keys: public: (17, 77);

$d = 17^{\phi(60)-1} \pmod{60}$

$d = 17^{15} \pmod{60} = 53$

Bob wants to send Alice secret message HELLO (07 04 11 11 14)

$07^{17} \pmod{77} = 28$

$04^{17} \pmod{77} = 16$

$11^{17} \pmod{77} = 44$

$11^{17} \pmod{77} = 44$

$14^{17} \pmod{77} = 42$

Bob sends 28 16 44 44 42 to Alice

Alice receives 28 16 44 44 42

Alice uses private key, $d = 53$, to decrypt message:

$$28^{53} \bmod 77 = 07$$

$$16^{53} \bmod 77 = 04$$

$$44^{53} \bmod 77 = 11$$

$$44^{53} \bmod 77 = 11$$

$$42^{53} \bmod 77 = 14$$

Then Alice translates message to letters to read HELLO

Example: Integrity/Authentication

Let $p = 7$, $q = 11$, so $n = 77$ and $\phi(n) = 60$

Alice chooses $e = 17$, making $d = 53$

Alice wants to send Bob message HELLO (07 04 11 11 14) so Bob knows it is what Alice sent (no changes in transit, and authenticated)

$$07^{53} \bmod 77 = 35$$

$$04^{53} \bmod 77 = 09$$

$$11^{53} \bmod 77 = 44$$

$$11^{53} \bmod 77 = 44$$

$$14^{53} \bmod 77 = 49$$

Alice sends 35 09 44 44 49 to Bob

Bob receives 35 09 44 44 49

Bob uses Alice's public key, $e = 17$, $n = 77$, to decrypt message:

$$35^{17} \bmod 77 = 07$$

$$09^{17} \bmod 77 = 04$$

$$44^{17} \bmod 77 = 11$$

$$44^{17} \bmod 77 = 11$$

$$49^{17} \bmod 77 = 14$$

Bob translates message to letters to read HELLO

Example: confidentiality and authenticity

Alice's keys: public (17, 77); private: 53

Bob's keys: public: (37, 77); $d = 37^{15} \bmod 60 = 13$, $d = 13$

Alice enciphers HELLO (07 04 11 11 14):

$$(07^{53} \bmod 77)^{37} \bmod 77 = 07$$

$$(04^{53} \bmod 77)^{37} \bmod 77 = 37$$

$$(11^{53} \bmod 77)^{37} \bmod 77 = 44$$

$$(11^{53} \bmod 77)^{37} \bmod 77 = 44$$

$$(14^{53} \bmod 77)^{37} \bmod 77 = 14$$

Alice sends 07 37 44 44 14

Example

Let $p = 7$, $q = 11$, $e = 17$ so

$$n = p * q = 7 * 11 = 77$$

$$\phi(n) = (p-1) * (q-1) = 6 * 10 = 60$$

public key $(e, n) = (17, 77)$

$$d = e^{-1} \bmod \phi(n)$$

$$\phi(60) = 2^2 * 3^1 * 5^1 = 2(2-1) * (3-1) * (5-1) = 2 * 2 * 4 = 16$$

$$d = 17^{-1} \bmod 60$$

$$d = 17^{15} \bmod 60 = 53$$

encryption

M (message) = HE (07 04)

$$C = M^e \bmod n$$

$$C(H) = 07^{17} \bmod 77 = 28$$

$$C(E) = 04^{17} \bmod 77 = 16$$

Decryption

$$M = C^d \bmod n$$

$$E(28) = 28^{53} \bmod 77 = 07$$

$$E(16) = 16^{53} \bmod 77 = 04$$

KEY MANAGEMENT:

One of the major roles of public-key encryption has been to address the problem of key distribution. There are actually two distinct aspects to the use of public-key cryptography in this regards:

- The distribution of public keys.
- The use of public-key encryption to distribute secret key.

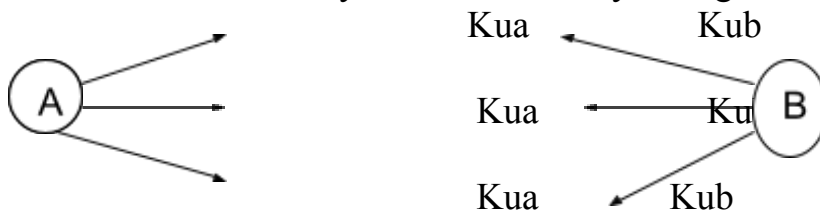
Distribution of public keys:

Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into the following general schemes:

1. Public Announcement.
2. Publicly available directory.
3. Public-key authority.
4. Public-key Certificate.

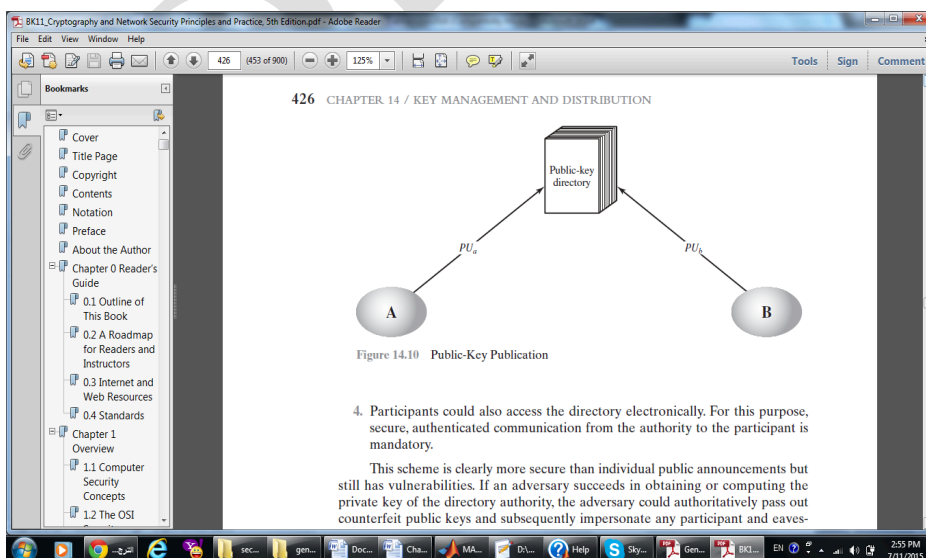
1. Public Announcement:

If there is some broadly accepted public-key algorithm, such as RSA, any participant can send his/her public key to any other participant or broadcast the key to the community at large.



2. Publicly Available Directory:

- a- The authority maintains a directory with (name, key,) entry for each participant .
- b- Each participant registers public key with the directory authority.
Registration would have to be in person or by some form of secure authenticated communication.
- c- A participant may replace the existing key with a new one at any time ,either because of the desire to replace a public key that has private key has been compromise in some way .
- d- periodically the authority publishes the entire directory or updates to the directory ,for example , a hard –copy version must like a telephone book could be published , or updates could be listed in the widely.



3. Public key Authority:

Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. A central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key.

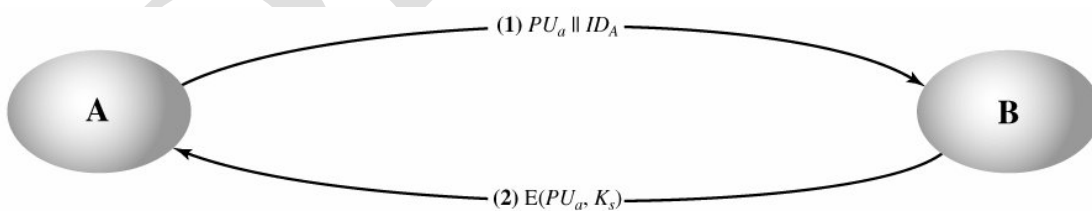
4. Public-key Certificate:

The directory of names and public key maintained by the authority is vulnerable to tampering. An alternative approach is to use certificate that can be used by participants to exchange keys without contacting a public-key authority, in away that is reliable as if the keys were obtained directly from a public-key authority. Each certificate contains a public key and other information, is created by a certificate authority, and is given to the participant with the matching private key. A participant conveys its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority.

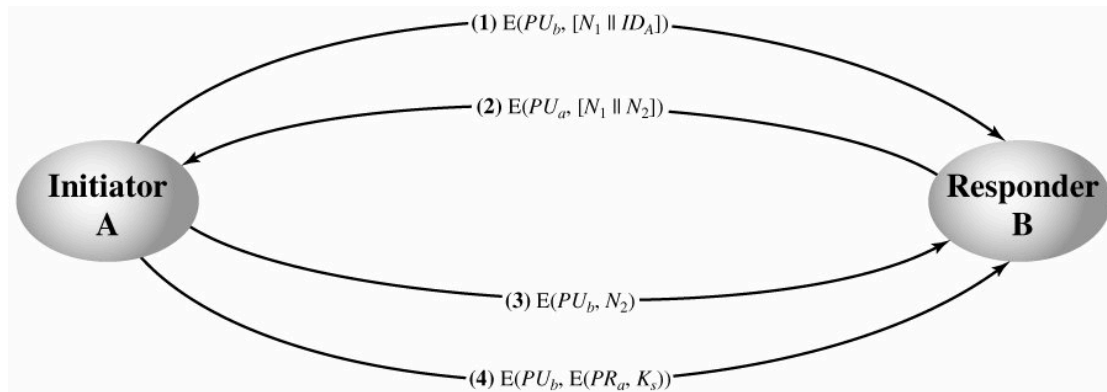
Public-key Distribution of secret keys:

Once public keys have been distributed or have become accessible, secure communication that thwarts eavesdropping tampering, or both is possible.

1- Simple secret key Distribution.



2-Secret Key Distribution with Confidentiality and Authentication:



1. A uses B's public key to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.
2. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2). Because only B could have decrypted message (1), the presence of N_1 in message (2) assures A that the correspondent is B.
3. A returns N_2 encrypted using B's public key, to assure B that its correspondent is A.
4. A selects a secret key K_s and sends $M = E(PU_b, E(PR_a, K_s))$ to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.
5. B computes $D(PU_a, D(PR_b, M))$ to recover the secret key.

3- A Hybrid Scheme

This scheme retains the use of a key distribution centre (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key. A public key scheme is used to distribute the master keys.

DIFFIE-HELLMAN KEY EXCHANGE:

The purpose of the algorithm is to enable two users to exchange a secret key securely that can be used for subsequent encryption of messages. The algorithm itself is limited to the exchange of the keys.

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.

We can define the discrete logarithm in the following way:

- 1- We define a primitive root of a prime number P as one whose powers generate all integers from 1 to $P-1$. That is, if a is a primitive root of the prime number P , then the numbers:

Ex:

Prove 3 is primitive root of 7

$$3^1 \bmod 7 = 3$$

$$3^2 \bmod 7 = 2$$

$$a \bmod P, a^2 \bmod P, \dots, a^{P-1} \bmod P$$

For any integer b and a primitive root of the prime number p , one can find a unique exponent I such that:

$$B = a^i \bmod p \quad \text{where } 0 \leq i \leq (p-2)$$

Or

$$B = a^i \bmod p \quad \text{where } 1 \leq i \leq (p-1)$$

For this scheme, there are two publicly known numbers: a prime number q and an integer α that is a primitive root of q . Suppose the users A and B wish to exchange a key. User A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{X_A} \bmod q$. Similarly, user B independently selects a random integer $X_B < q$ and computes $Y_B = \alpha^{X_B} \bmod q$. Each side keeps the X value private and makes the Y value available publicly to the other side. User A computes the key as $K = (Y_B)^{X_A} \bmod q$ and user B computes the key as

$K = (Y_B)^{X_A} \bmod q$. These two calculations produce identical results:

$$K = [(Y_B)^{X_A} \bmod q] \bmod q$$

$$\begin{aligned} K &= [(\alpha^{X_B} \bmod q)^{X_A}] \bmod q \\ &= [(\alpha^{X_B} \bmod q)^{X_A}] \bmod q \end{aligned}$$

$$= \alpha^{X_B X_A} \bmod q \quad \text{by the rules of modular arithmetic.}$$

$$= (\alpha^{X_A})^{X_B} \bmod q$$

$$= (\alpha^{X_A} \bmod q)^{X_B} \bmod q$$

$$= (Y_A)^{X_B} \bmod q$$

Thus, the two sides have exchanged a secret key. Because X_A and X_B are private, an opponent only has the following ingredients to work with: q , a , Y_A , Y_B . Thus, the opponent is forced to take a discrete logarithm to determine the key. For example, attacking the secret key of user B, the opponent must compute :

$$X_B = \text{ind}_{\alpha, q}(Y_B)$$

Example:

The prime number $q = 71$ and a primitive root of 71, in this case $\alpha = 7$. A and B select private keys $X_A = 5$ and $X_B = 12$, respectively. Each computes its public key:

$$Y_A = 7^5 \bmod 71 = 51 \bmod 71$$

$$Y_B = 7^{12} \bmod 71 = 4 \bmod 71$$

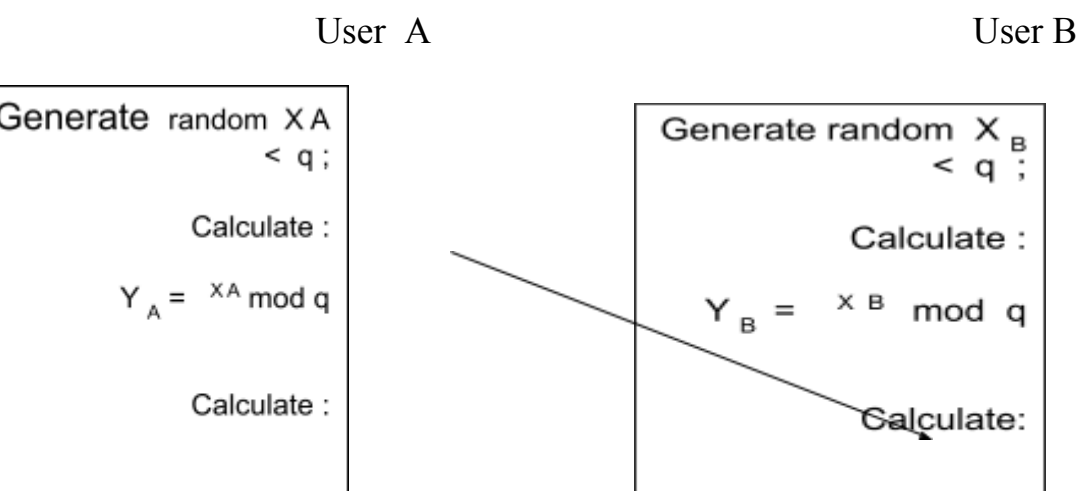
After they exchange the public keys, each can compute the common secret keys:

$$K = (Y_B)^{X_A} \bmod 71 = 4^5 \bmod 71 = 30 \bmod 71$$

$$K = (Y_A)^{X_B} \bmod 71 = 51^{12} \bmod 71 = 30 \bmod 71$$

From $[51, 4]$, an attacker can not easily compute 30.

The following figure shows a simple protocol that makes use of the Diffie-Hellman calculation:





- 1- Suppose that user A wishes to connect user B by using a secret key to encrypt messages.
- 2- User A can generate a one-time private key X_A .
- 3- Calculate Y_A and send that to user B.
- 4- User B responds by generating a private value X_B , calculating Y_B and sending Y_B to user A.
- 4- Both users can now calculate the key.
- 5- The necessary public values q and ∞ would need to be known ahead of time.
- 6- Alternatively, user A could pick values for q and ∞ and include those in the first message.

The following figure is summarized the Diffie- Hellman key Exchange.

Global Public Elements

Q

Prime Number

$< q$ and a primitive root of q .

User A Key Generator

Select Private Key X_A

$$X_A < q$$

Calculate public Y_A

$$Y_A = X_A^A \mod q$$

User B Key Generator

Select Private Key X_B

$$X_B < q$$

Calculate public Y_B

$$Y_B = X_B^B \mod q$$

Generation of secret key by user A

$$K = (Y_B)^{X_A} \mod q$$

Generation of secret key by user B

$$K = (Y_A)^{x_B} \mod q$$

Message Authentication and Hash Functions.

Message Authentication Requirements:

In the context of communications across a network, the following attacks can be identified:

- 1- Disclosure: Release of message contents to any person or process not possessing the appropriate Cryptographic key.
- 2- Traffic Analysis: Discovery of the pattern of traffic between parties.
- 3- Masquerade: Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity
- 4- Content Modification: Changes to the content of message including insertion, deletion, transposition, and modification.
- 5- Sequence Modification: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
- 6- Timing Modification: Delay or replay of messages.
- 7- Source Repudiation: Denial of transmission of message by source.
- 8- Destination Repudiation: Denial of receipt of messages by destination.

Authentication Functions:

Any message authentication or digital signature mechanism can be viewed as having fundamentally two levels. At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message.

There are several types of function to produce an authenticator. These may be grouped into three classes as follows:

- 1- Message Encryption: The cipher text of the entire message serves as its authenticator.
- 2- Message Authentication Code (MAC): A public function of the message and a secret key that produces a fixed length value that serves as the authenticator.
- 3- Hash Function: A public function that maps a message of any length into a fixed length hash value, which serves as the authenticator.

MESSAGE AUTHENTICATION.

Encryption protects against passive attack (eavesdropping). A different requirement is to protect against active attack (falsification of data and transactions). Protection against such attacks is known as message authentication.

Message authentication is a procedure that allows communicating parties to verify that received messages are authentic.

Message timeliness (it has not been artificially delayed and replayed) and sequence relative to other messages flowing between two parties.

1- Message Encryption:

a. Authentication Using Conventional Encryption.

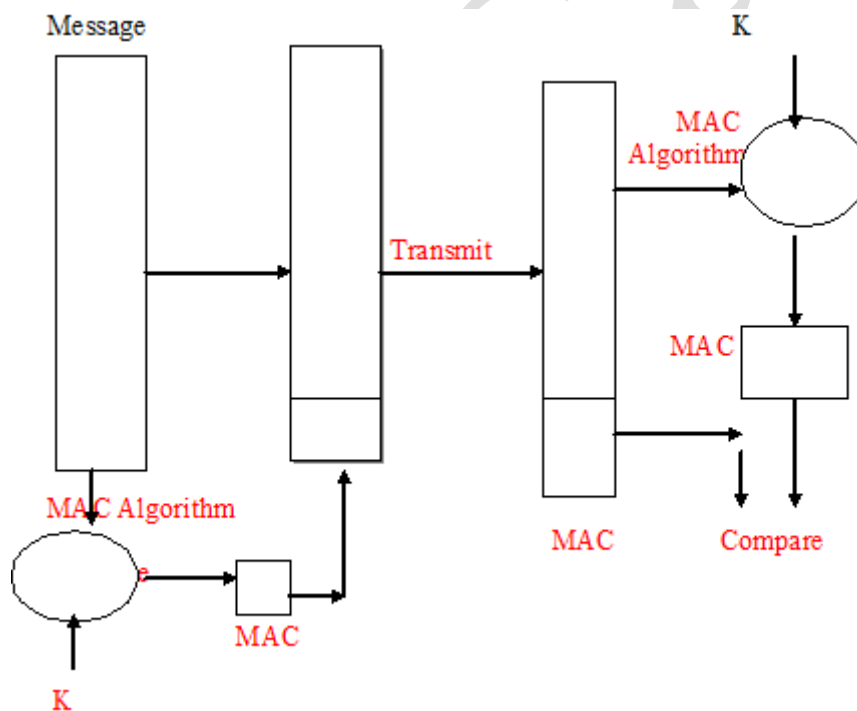
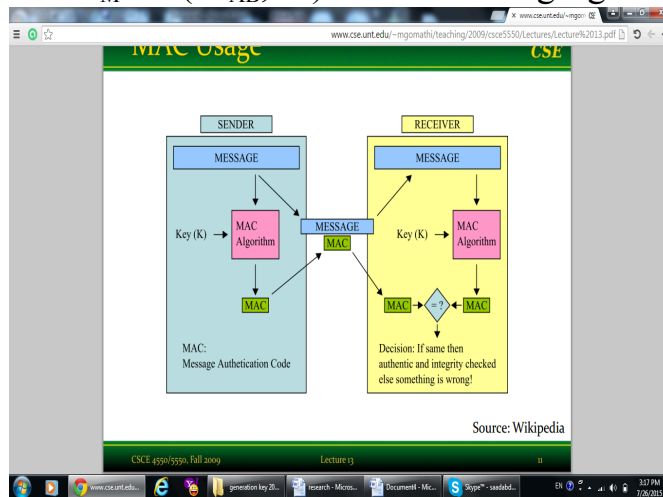
If we assume that only the sender and receiver share a key (which is as it should be), then only the genuine sender would be able to encrypt a message successfully for the other participant. If the message also includes a timestamp, the receiver is assured that the message has not been delayed beyond that normally expected for network transit.

b. Message Authentication without Message Encryption.

In all of these approaches, an authentication tag is generated and appended to each message for transmission. The message itself is not encrypted and can be read at the destination independent of the authentication function at the destination.

2- Message Authentication Code (MAC).

One authentication technique involves the use of a secret key to generate a small block of data, known as a message authentication code, that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K_{AB} . When A has a message to send to B, it calculates the message authentication code as a function of the message and the key: $MAC_M = F(K_{AB}, M)$. The following figure is explaining MAC.



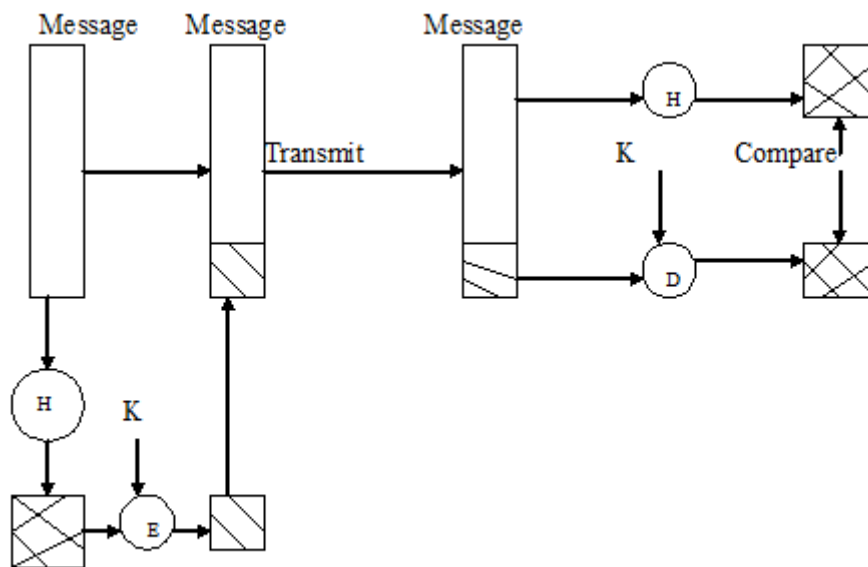
The process just described is similar to encryption. One difference is that the authentication algorithm need not be reversible, as it must for decryption.

3- One-Way Hash Function.

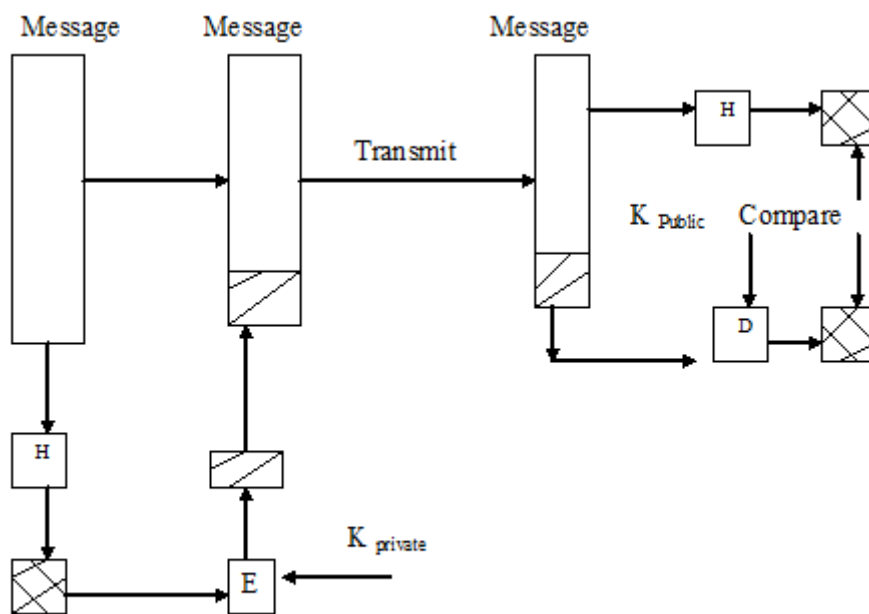
A variation on the message authentication code that has received much attention recently is the one-way hash function. As with a message authentication code, a hash function accepts a variable-size message M as input and produces a fixed-size message digest $H(M)$ as output. Unlike MAC, a hash function does not also take a secret key as input. To authenticate a message, the message digest is sent with the message in such a way that the message digest is authentic.

The following figure illustrates three ways in which the message can be authenticated.

Dr. Saad



(a) Using Conventional Encryption



(b) Using Public-Key Encryption

Ex using RSA in an uthentication

- 1- $P = 7$, $q = 5$, $e = 31$, $M = 4$
- 2- $N = p * q = 7 * 5 = 35$
- 3- $\phi(n) = (p-1)(q-1) = (7-1)(5-1) = 24$
- 4- $d \text{ mod } \phi(n) = 1$
 - $1 * 31 \text{ mod } 24 = 7$
 - $2 * 31 \text{ mod } 24 = 14$
 - $3 * 31 \text{ mod } 24 = 21$
 - $4 * 31 \text{ mod } 24 = 4$

$$5 * 31 \bmod 24 = 11$$

$$6 * 31 \bmod 24 = 18$$

$$7 * 31 \bmod 24 = 7$$

$$d = 7$$

$$kp = \{ 31, 35 \}$$

$$kR = \{ 7, 35 \}$$

encryption

$$\begin{aligned} C &= M^d \bmod n \\ &= 4^7 \bmod 35 = 4 \end{aligned}$$

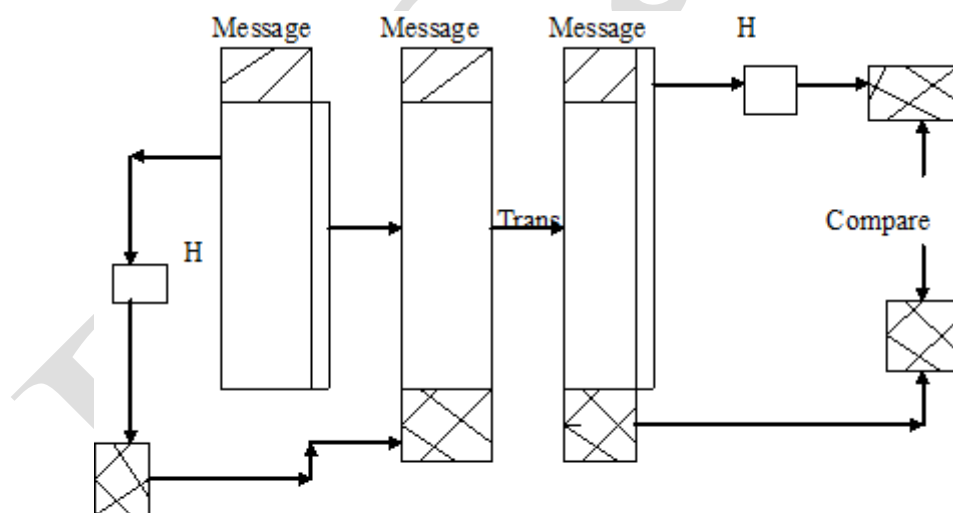
Decryption

$$\begin{aligned} M &= C^e \bmod n \\ &= 4^{31} \bmod 35 = 4 \end{aligned}$$

Note

$$4^3 \bmod 35 = 29$$

$$29 * 29 \bmod 35 = 1$$



(C) Using Secret Value

There are several reasons for developing a technique that avoids encryption:

1. Encryption software is quite slow. Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system.

2. Encryption hardware costs are no negligible. Low cost chip implementations of DES are available, but the cost adds up if all nodes in a network must have this capability.
3. Encryption hardware is optimized toward large data sizes.
4. Encryption algorithms may be covered by patents.
5. Encryption algorithms may be subject to export control. This is true of DES.

knapsacks

Key generation

In Merkle-Hellman, the keys are knapsacks. The public key is a 'hard' knapsack, and the private key is an 'easy', or superincreasing, knapsack, combined with two additional numbers, a multiplier and a modulus, which were used to convert the superincreasing knapsack into the hard knapsack. These same numbers are used to transform the sum of the subset of the hard knapsack into the sum of the subset of the easy knapsack, which is solvable in polynomial time.

Encryption

To encrypt a message, a subset of the hard knapsack is chosen by comparing it with a set of bits (the plaintext), equal in length to the key, and making each term in the public key that corresponds to a 1 in the plaintext an element of the subset, while ignoring the terms corresponding to 0 terms in the plaintext. The elements of this subset are added together, and the resulting sum is the ciphertext.

Decryption

Decryption is possible because the multiplier and modulus used to transform the easy, superincreasing knapsack into the public key can also be used to transform the number representing the ciphertext into the sum of the corresponding elements of the superincreasing knapsack. Then, using a simple greedy algorithm, the easy knapsack can be solved using $O(n)$ arithmetic operations, which decrypts the message.

Mathematical method

Key generation

To encrypt n -bit messages, choose a [superincreasing sequence](#)

$$1- w = (w_1, w_2, \dots, w_n)$$

2 -of n nonzero [natural numbers](#). Pick a random integer q , such that

$$q > \sum_{i=1}^n w_i,$$

and a random integer, r , such that $\gcd(r, q) = 1$ (i.e. r and q are [coprime](#)).

q is chosen this way to ensure the uniqueness of the ciphertext. If it is any smaller, more than one plaintext may encrypt to the same ciphertext. r must be [coprime](#) to q or else it will not have an inverse mod q . The existence of the inverse of r is necessary so that decryption is possible.

Now calculate the sequence

- 3- $\beta = (\beta_1, \beta_2, \dots, \beta_n)$ where
 $\beta_i = rw_i \bmod q$.

The public key is β , while the private key is (w, q, r) .

Encryption

To encrypt an n -bit message

- 4- $\mathbf{M} = \mathbf{a} = (a_1, a_2, \dots, a_n)$,
 5- where a_i is the i -th bit of the message and $a_i \in \{0, 1\}$, calculate

$$c = \sum_{i=1}^n a_i \beta_i.$$

The cryptogram then is c .

Decryption

In order to decrypt a ciphertext c a receiver has to find the message bits a_i such that they satisfy

$$c = \sum_{i=1}^n a_i \beta_i.$$

This would be a hard problem if the β_i were random values because the receiver would have to solve an instance of the subset sum problem, which is known to be NP-hard. However, the values β_i were chosen such that decryption is easy if the private key (w, q, r) is known.

The key to decryption is to find an integer s that is the [modular inverse](#) of r modulo q . That means s satisfies the equation $sr \bmod q = 1$ or equivalently there exist an integer k such that $sr = kq + 1$. Since r was chosen such that $\gcd(r, q) = 1$ it is possible

to find s and k by using the [Extended Euclidean algorithm](#). Next the receiver of the ciphertext c computes

$$c' \equiv cs \pmod{q}.$$

Hence

$$c' \equiv cs \equiv \sum_{i=1}^n \alpha_i \beta_i s \pmod{q}.$$

Because of $rs \bmod q = 1$ and $\beta_i = rw_i \bmod q$ follows

$$\beta_i s \equiv w_i r s \equiv w_i \pmod{q}.$$

Hence

$$c' \equiv \sum_{i=1}^n \alpha_i w_i \pmod{q}.$$

The sum of all values w_i is smaller than q and hence $\sum_{i=1}^n \alpha_i w_i$ is also in the interval $[0, q-1]$. Thus the receiver has to solve the subset sum problem

$$c' = \sum_{i=1}^n \alpha_i w_i.$$

This problem is easy because w is a superincreasing sequence. Take the largest element in w , say w_k . If $w_k > c'$, then $a_k = 0$, if $w_k \leq c'$, then $a_k = 1$. Then, subtract $w_k \times a_k$ from c' , and repeat these steps until you have figured out a .

Example

First, a superincreasing sequence w is created

$$w = \{2, 7, 11, 21, 42, 89, 180, 354\}$$

This is the basis for a private key. From this, calculate the sum.

$$\sum w = 706$$

Then, choose a number q that is greater than the sum.

$$q = 881$$

Also, choose a number r that is in the range $[1, q)$ and is [coprime](#) to q .

$$r = 588$$

The private key consists of q , w and r .

To calculate a public key(β), generate the sequence β by multiplying each element in w by $r \bmod q$

$$B = w * r \bmod q$$

$$\begin{aligned} 2 * 588 \bmod 881 &= 295 \\ 7 * 588 \bmod 881 &= 592 \\ 11 * 588 \bmod 881 &= 301 \\ 21 * 588 \bmod 881 &= 14 \\ 42 * 588 \bmod 881 &= 28 \\ 89 * 588 \bmod 881 &= 353 \\ 180 * 588 \bmod 881 &= 120 \\ 354 * 588 \bmod 881 &= 236 \\ \beta &= \{295, 592, 301, 14, 28, 353, 120, 236\} \end{aligned}$$

The sequence β makes up the public key.

Say Alice wishes to encrypt "a". First, she must translate "a" to binary (in this case, using [ASCII](#) or [UTF-8](#))

$$\text{Message} = \text{'a'} = 01100001$$

She multiplies each respective bit by the corresponding number in β

$$\begin{aligned} \text{Message} = a &= 01100001 \\ &= 0 * 295 + 1 * 592 + 1 * 301 + 0 * 14 + 0 * 28 + 0 * 353 + 0 * 120 + 1 * 236 \\ &= 1129 \end{aligned}$$

She sends this to the recipient.

To decrypt, Bob multiplies 1129 by $r^{-1} \bmod q$ (See [Modular inverse](#))

$$1129 * 442 \bmod 881 = 372$$

Decryption

Now Bob decomposes 372 by selecting the largest element in w which is less than or equal to 372. Then selecting the next largest element less than or equal to the difference, until the difference is 0:

$$\begin{aligned} 372 - 354 &= 18 \\ 18 - 11 &= 7 \\ 7 - 7 &= 0 \end{aligned}$$

The elements we selected from our private key correspond to the 1 bits in the message

01100001

When translated back from binary, this "a" is the final decrypted message.

Example

W = (1,3,5,10)

1- $\sum w = 19$

2- Select $r = 7 \quad r \in [1, q)$

3- Select $q = 20 \quad q > \sum w$

4- $B = wr \bmod q$

5-
$$\begin{aligned} 1 * 7 \bmod 20 &= 7 \\ 3 * 7 \bmod 20 &= 1 \\ 5 * 7 \bmod 20 &= 15 \\ 10 * 7 \bmod 20 &= 10 \end{aligned}$$

6- $B = (7, 1, 15, 10)$

7- Plain text = n = 13 in binary = { 1101 }

$$c' = \sum_{i=1}^n \alpha_i w_i.$$

8- $C = (1 * 7 + 1 * 1 + 15 * 0 + 10 * 1 = 18$

9- **Decryption**

10- Calculate r^{-1}

11- $r * r^{-1} \bmod q = 1$

12-
$$\begin{aligned} 7 * 1 \bmod 20 &= 7 \\ 7 * 2 \bmod 20 &= 14 \\ 7 * 3 \bmod 20 &= 1 \end{aligned}$$

13- $r^{-1} = 3$

14- $Dr^{-1}(c) = D(18) = 18 * 3 \bmod 20 = 14$

15- Cipher = 14 = (1101) = 13 = N

Ex2

16- **W = (1,3,5,7)**

17- $\sum w = 16$

18- Select $r = 17 \quad r \in [1, q)$

19- Select $q = 5 \quad q > \sum w$

20- $B = wr \bmod q$

21-
$$\begin{aligned} 1 * 5 \bmod 20 &= 5 \\ 3 * 5 \bmod 20 &= 15 \\ 5 * 5 \bmod 20 &= 8 \\ 10 * 5 \bmod 20 &= 1 \end{aligned}$$

22- $B = (5, 15, 8, 1)$

23- Plain text = n = 13 in binary = { 1101 }

$$c' = \sum_{i=1}^n \alpha_i w_i.$$

$$24- C = (1 * 5 + 1 * 15 + 8 * 0 + 1 * 1 = 21$$

25- **Decryption**

26- **Calculate r^{-1}**

$$27- r * r^{-1} \bmod q = 1$$

$$28- \quad 5 * 1 \bmod 20 = 5$$

$$5 * 2 \bmod 20 = 10$$

$$5 * 3 \bmod 20 = 15$$

$$5 * 4 \bmod 20 = 0$$

$$5 * 5 \bmod 20 = 5$$

$$5 * 7 \bmod 20 = 1$$

$$29- r^{-1} = 7$$

$$30- Dr^{-1}(c) = D(21) = 21 * 7 \bmod 17 = 11$$

$$31- \text{Cipher} = 11 - 7 = 4 - 3 = 1 - 1 = 0 = (1101) = 13 = N$$

ElGamal public key cryptosystem

Alice chooses

Step1 : take prime number “p” and any number $g < p$, which are secretly shared among Ali and Bob. Therefore public key “y” is

Step 2 : A primitive element g modulo p

Step3: to generate secret key (at sender side)

Create secret key “x” as $x < p$, which shared among Alice and Bob

Alice public key (pa,ga,Y)

$$Y = g^x \bmod p$$

Encryption

1 Bob encrypts a short message M ($M < p$) and sends to ALice

2- Choose one-time random number “k” as it relative prime to p-1, such that $0 \leq k \leq (p-1)$ which is secret.

3- Create “c1 “ ,”c2”

$$C1 = g^k \bmod p$$

$$C2 = Y^k * M \bmod p$$

4- Bob send his encrypted message(c1,c2) to Alice

Decryption

1- Calculate $h = c1^x \bmod p$

2- Find h^{-1} (inverse of h) such that $h * h^{-1} \bmod p = 1$

3- $M = h^{-1} c2 \bmod p$

Ex:

prime $p = 139$, $g = 3$, $k = 52$, $x = 12$, $M = 100$

encryption

$$Y = g^x \bmod p$$

$$Y = 3^{12} \bmod 139 = 44$$

$$C1 = g^k \bmod p = 3^{52} \bmod 139 = 38$$

$$C2 = Y^k * M \bmod p = 44^{52} * 100 \bmod 139 = 80$$

$$(c1, c2) = (38, 80)$$

decryption

$$h = c1^x \bmod p = 38^{12} \bmod 139 = 112$$

$$h^{-1} = 112^{-1} \bmod 139 = 36$$

$$h^{-1} = 36$$

$$M = h^{-1} c2 \bmod p = 36 * 80 \bmod 139 = 100$$

Example

1. **Public Values:** Choose a prime number $p = 23$ and a generator $g = 5$.

2. **Key Generation:**

- Suppose Alice chooses private key $x = 6$.
- The public key $y = g^x \bmod p = 5^6 \bmod 23 = 8$.
- So Alice's public key is ($p = 23$, $g = 5$, $y = 8$) and the private key is $x = 6$.

3. **Encrypting a Message:**

- Suppose the message Bob wants to send is $\mathbf{M} = 10$.
- He chooses a random ephemeral key $\mathbf{k} = 15$ (this key should be different each time encryption is performed).
- He calculates the two parts of the ciphertext:
- $\mathbf{c1} = \mathbf{g}^{\mathbf{k}} \bmod \mathbf{p} = 5^{15} \bmod 23 = 19$.
- $\mathbf{c2} = \mathbf{M} \cdot \mathbf{y}^{\mathbf{k}} \bmod \mathbf{p} = 10 \cdot 8^{15} \bmod 23 = 7$.
- The ciphertext is then ($\mathbf{c1} = 19, \mathbf{c2} = 7$) and this is sent to Alice as (19, 7).

4. Decrypting the Message:

- Alice receives ($\mathbf{c1}, \mathbf{c2}$) = (19, 7)
- To decrypt, she uses her private key $\mathbf{x} = 6$.
- Calculate $\mathbf{M} = \mathbf{c2} \cdot (\mathbf{c1x})^{-1} \bmod \mathbf{p}$.
- First, calculate $\mathbf{c1x} \bmod \mathbf{p} = 19^6 \bmod 23 = 7$.
- Now find the modular inverse of 7 mod 23, which is 10.
- Finally, she computes $\mathbf{M} = 7 \cdot 10 \bmod 23 = 10$, which is the original message.
- Alice successfully recovers the plaintext $\mathbf{M} = 10$

homepages.math.uic.edu/~leon/mcs425-s08/handouts/PrimitiveElements.pdf

Microsoft Word - PrimitiveElements.doc

1 / 2 90%

$a \not\equiv 0 \pmod{p}$. Let us look at some examples

$p = 5$		$p = 7$		$p = 13$		$p = 23$		
$a = 2$		$a = 2$	$a = 3$	$a = 2$	$a = 3$	$a = 2$	$a = 3$	$a = 5$
$2^0 \equiv 1$		$2^0 \equiv 1$	$3^0 \equiv 1$	$2^0 \equiv 1$	$3^0 \equiv 1$	$2^0 \equiv 1$	$3^0 \equiv 1$	$5^0 \equiv 1$
$2^1 \equiv 2$		$2^1 \equiv 2$	$3^1 \equiv 3$	$2^1 \equiv 2$	$3^1 \equiv 3$	$2^1 \equiv 2$	$3^1 \equiv 3$	$5^1 \equiv 5$
$2^2 \equiv 4$		$2^2 \equiv 4$	$3^2 \equiv 2$	$2^2 \equiv 4$	$3^2 \equiv 9$	$2^2 \equiv 4$	$3^2 \equiv 9$	$5^2 \equiv 2$
$2^3 \equiv 3$		$2^3 \equiv 1$	$3^3 \equiv 6$	$2^3 \equiv 8$	$3^3 \equiv 1$	$2^3 \equiv 8$	$3^3 \equiv 4$	$5^3 \equiv 10$
$2^4 \equiv 1$		$2^4 \equiv 2$	$3^4 \equiv 4$	$2^4 \equiv 3$	$3^4 \equiv 3$	$2^4 \equiv 16$	$3^4 \equiv 12$	$5^4 \equiv 4$
		$2^5 \equiv 4$	$3^5 \equiv 5$	$2^5 \equiv 6$	$3^5 \equiv 9$	$2^5 \equiv 9$	$3^5 \equiv 13$	$5^5 \equiv 20$
		$2^6 \equiv 1$	$3^6 \equiv 1$	$2^6 \equiv 12$	$3^6 \equiv 1$	$2^6 \equiv 18$	$3^6 \equiv 16$	$5^6 \equiv 8$
				$2^7 \equiv 11$	$3^7 \equiv 3$	$2^7 \equiv 13$	$3^7 \equiv 2$	$5^7 \equiv 17$
				$2^8 \equiv 9$	$3^8 \equiv 9$	$2^8 \equiv 3$	$3^8 \equiv 6$	$5^8 \equiv 16$
				$2^9 \equiv 5$	$3^9 \equiv 1$	$2^9 \equiv 6$	$3^9 \equiv 18$	$5^9 \equiv 11$
				$2^{10} \equiv 10$	$3^{10} \equiv 3$	$2^{10} \equiv 12$	$3^{10} \equiv 8$	$5^{10} \equiv 9$
				$2^{11} \equiv 7$	$3^{11} \equiv 9$	$2^{11} \equiv 1$	$3^{11} \equiv 1$	$5^{11} \equiv 22$
				$2^{12} \equiv 1$	$3^{12} \equiv 1$	$2^{12} \equiv 2$	$3^{12} \equiv 3$	$5^{12} \equiv 18$
						$2^{13} \equiv 4$	$3^{13} \equiv 9$	$5^{13} \equiv 21$
						$2^{14} \equiv 8$	$3^{14} \equiv 4$	$5^{14} \equiv 13$
						$2^{15} \equiv 16$	$3^{15} \equiv 12$	$5^{15} \equiv 19$
						$2^{16} \equiv 9$	$3^{16} \equiv 13$	$5^{16} \equiv 3$
						$2^{17} \equiv 18$	$3^{17} \equiv 16$	$5^{17} \equiv 15$
						$2^{18} \equiv 13$	$3^{18} \equiv 2$	$5^{18} \equiv 6$
						$2^{19} \equiv 3$	$3^{19} \equiv 6$	$5^{19} \equiv 7$
						$2^{20} \equiv 6$	$3^{20} \equiv 18$	$5^{20} \equiv 12$
						$2^{21} \equiv 12$	$3^{21} \equiv 8$	$5^{21} \equiv 14$
						$2^{22} \equiv 1$	$3^{22} \equiv 1$	$5^{22} \equiv 1$

Fermat's Theorem.

2) For each prime p in the table, we can find some integer b (not divisible by p) such that $b^i \not\equiv 1 \pmod{p}$ for $0 < i < p-1$. In other words, $p-1$ is the *smallest* positive integer j such that $b^j \equiv 1 \pmod{p}$.

We call b a primitive root mod p .

2 is a primitive root mod 5, and also mod 13.
3 is a primitive root mod 7.
5 is a primitive root mod 23.

It can be proven that there exists a primitive root mod p for every prime p . (However, the proof isn't easy; we shall omit it here.)

3) For each primitive root b in the table, $b^0, b^1, b^2, \dots, b^{p-2}$ are all distinct in Z_p , and they constituted all the nonzero elements of Z_p .

Again, this is always true, and easy to prove. We know that b has an inverse since $b \not\equiv 0 \pmod{p}$. If

$$b^i \equiv b^k \pmod{p} \text{ for } 0 \leq i < k \leq p-2,$$

then

$$b^i(b^{k-i}) \equiv b^k(b^{-i}) \Rightarrow b^i b^{-i} \equiv b^k b^{-i} \Rightarrow 1 \equiv b^{k-i} \pmod{p}$$

and $0 < k-i \leq p-2$, which contradicts b being primitive.

Activate Windows
Go to Settings to activate Windows.