Public Key Cryptography and Key Distribution

S.R. Subramanya

School of Engineering and Computing
National University
Elements of Number Theory
Fermat’s Theorem

If $p$ is prime and $a$ is a positive integer not divisible by $p$, then

$$a^{p-1} \equiv 1 \mod p$$

Alternatively,

$$a^p \equiv a \mod p$$
Euler’s Totient Function

Euler’s Totient Function $\Phi(n)$ is the number of positive integers less than $n$ and relatively prime to $n$. 
Euler’s Theorem

For every $a$ and $n$ that are relatively prime,

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

Alternatively,

$$a^{\phi(n)+1} \equiv a \pmod{n}$$
Primitive Root

- Primitive Root of a prime number \( p \) is one whose powers generate all integers from 1 to \( p-1 \).
- If \( a \) is a primitive root of the prime number, \( p \) then the numbers \( a \mod p, a^2 \mod p, \ldots, a^{p-1} \mod p \) are distinct and consist of integers from 1 to \( (p-1) \) in some permutation.
Discrete Logarithm

For any integer ‘\(b\)’ and a primitive root ‘\(a\)’ of prime number ‘\(p\)’, one can find a unique exponent ‘\(i\)’ such that

\[ b = a^i \mod p \text{ where } 0 \leq i \leq (p-1) \]

The exponent ‘\(i\)’ is called the Discrete Logarithm or index of \(b\) for the base \(a \mod p\).

Given \(a\), \(i\), and \(p\), it is straightforward to compute \(b\).

Given \(a\), \(b\), and \(p\), it is computationally infeasible to compute the discrete logarithm \(i\).
Diffie–Hellman Key Exchange
Diffie–Hellman Key Exchange

First published public-key algorithm (1976)
Based on difficulty of computing Discrete Logarithms
Enables two users to exchange a key securely to be used for subsequent message encryption
Several commercial products based on this technique
Diffie – Hellman Key Exchange

$q$: Prime number

$\alpha$: $\alpha < q$ and is primitive root of $q$

$q, \alpha$ are required to be known ahead of time (or $A$ could pick $q$ and $\alpha$ and include in the first message)
Diffie–Hellman Exchange Example

Key exchange is based on the use of prime number q=97 and a primitive root of 97, in this case $\alpha = 5$. A and B select secret keys $X_A=36$ and $X_B=58$, respectively.

Each computes its public key:

$$K = (Y_B)^{X_A} \mod 97 = 44^{36} = 75 \mod 97$$
$$K = (Y_A)^{X_B} \mod 97 = 50^{58} = 75 \mod 97$$

From [50,44], an attacker cannot easily compute 75.
Confidentiality and Authentication
Using Public-Key Cryptography
Confidentiality Using Public-key System
Authentication Using Public-key System
Confidentiality and Authentication Using Public-key System

Source A

Message Source \( X \) Encryption Algorithm Y Encryption Algorithm Z Decryption Algorithm Y Decryption Algorithm X Message Dest.

Destination B

Key Pair Source

Key Pair Source

\( KU_a \) \( KR_a \) \( KU_b \) \( KR_b \)
RSA Cryptosystem


- RSA is a block cipher
- The most widely implemented
RSA Algorithm: Basics

Block Cipher
Block has binary value < n => Block Size ≤ log₂(n)
Block Size k bits: 2^k < n ≤ 2^{k+1}

M: message; C: ciphertext;
{e,n}: public key; {d,n}: private key
Both Sender and receiver know n;
Sender knows e, receiver knows d

C = M^e mod n
M = C^d mod n = (M^e)^d mod n = M^{ed} mod n

Requirements:
Possible to find e,d,n such that M^{ed} = M mod n for all M<n
Relatively easy to compute M^e and C^d for all M<n
Infeasible to determine d given e and n
Relationship between \( d \) and \( e \)

**Required:** \( M^{ed} \equiv M \mod n \)

**Corollary to Euler’s theorem (eq. 7.7):**

Given two prime numbers \( p \) and \( q \) and two integers \( m \) and \( n \) such that \( n = pq \) and \( 0 < m < n \) and an arbitrary integer \( k \),

\[
m^{k \Phi(n) + 1} = m^{k(p-1)(q-1) + 1} \equiv m \mod n,
\]

where \( \Phi(n) \) is the Euler Totient Function.

From the above, \( ed = K \Phi(n) + 1 \) satisfies the requirement

\[
\rightarrow \text{ed} = 1 \mod \Phi(n)
\]

\[
d \equiv e^{-1} \mod \Phi(n)
\]

\[
\rightarrow e \text{ and } d \text{ are multiplicative inverses } \mod \Phi(n)
\]
# The RSA Algorithm – Key Generation

1. **Select** \( p, q \) \( p \) and \( q \) both prime
2. **Calculate** \( n = p \times q \)
3. **Calculate** \( \Phi(n) = (p - 1)(q - 1) \)
4. **Select integer** \( e \) \( \gcd(\Phi(n), e) = 1; 1 < e < \Phi(n) \)
5. **Calculate** \( d \) \( d = e^{-1} \mod \Phi(n) \)
6. **Public Key** \( \text{KU} = \{e, n\} \)
7. **Private key** \( \text{KR} = \{d, n\} \)
The RSA Algorithm – Encryption

<table>
<thead>
<tr>
<th>Plaintext:</th>
<th>( M &lt; n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ciphertext:</td>
<td>( C = M^e \pmod{n} )</td>
</tr>
</tbody>
</table>
The RSA Algorithm – Decryption

Ciphertext: \( C \)

Plaintext: \( M = C^d \pmod{n} \)
Key Management Using Public-key Encryption
Public Key Distribution Using public Key Authority

For Distribution of public keys for directory to users A and B

$ID_A, ID_B$: Id of A(B); $KU_a, KU_b$: Public key of A(B);
$KRauth$: Private key of authority; $N_1, N_2$: nonce

1. A sends timestamped request for $KUb$
2. Authority sends $KUb$ together with timestamped request encrypted using $KRauth$ to A
2b. A decrypts using $KUauth$ and stores $KUb$
3. A sends $ID_A$ and $N_1$ encrypted using $KUb$ to B
4,5. B requests for and receives $KUa$ from the authority (Similar to 1,2,2b)
6. B sends $N1||N2$ encrypted using $KUa$ to A
7. A returns $N2$ encrypted using $KUb$ to B

Steps 1-5: Keys have been delivered
Public Key Distribution using Key Authority
Public Keys Certificates

Overcomes bottleneck in Public Key Authority
   Each user requesting authority for the public key of every other user

Certificates facilitate exchange of keys without contacting key authority

Certificate created by certificate authority

Certificate contains Public Key plus some other information

Certificate given to user with matching private key

Certificate: Timestamp, ID, Public Key encrypted using certificate authority’s Private key
Exchange of Public Key Certificates

- Certificate: \( C_A = E_{K^{\text{Auth}}} [T, ID_A, KU_a] \)
- Verification: \( D_{K^{\text{Uauth}}} [C_A] = D_{K^{\text{Uauth}}} [E_{K^{\text{Auth}}} [T, ID_A, KU_a]] = (T, ID_A, KU_a) \)
- Decryption of Certificate using Public key of authority provides authentication
Distribution of Secret keys using Public-keys

Simple Scheme (by Merkle)

Key Distribution with confidentiality and Authentication (Needham and Schroeder)
Merkle’s Secret Key Distribution Scheme

1. A generates a public/private key pair \([K_{UA}, K_{RA}]\) and transmits a message to B consisting of \(K_{UA}\) and an identifier of A, \(ID_A\).

2. B generates a secret key, \(K_S\), and transmits it to A, encrypted with A’s public key.

3. A computes \(D_{KR_a}[E_{K_{UA}}[K_S]]\) to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of \(K_S\).

4. A discards \(K_{UA}\) and \(KR_a\) and B discards \(K_{UA}\).
Needham & Schroeder Scheme 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 of A encrypted with KU_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 A.

3. A selects a secret key K_S and sends M=E_{KUb}[E_{KRa}[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.

4. B Computes D_{KUa}[D_{KRb}[M]] to recover the secret key.
Needham & Schroeder Scheme with Confidentiality and Authentication

Initiator A

Responder B

1. $E_{KUb}[N_1 \parallel ID_A]$

2. $E_{KUb}[N_1 \parallel N_2]$

3. $E_{KUb}[N_2]$

4. $E_{KUb}[E_{KRa}[K_s]]$
Digital Signatures
Digital Signatures

Digital Signature Creation

Digital Signature Verification
Digital Envelopes

Digital Envelope Creation

Opening Digital Envelope
Digital Envelopes Containing Signed Messages

Creation of Digital Envelope Carrying Signed Message

Opening Digital Envelope Carrying Signed Message and Verifying Signature
A Few Miscellaneous Security Protocols
Computation of Salary Averages

Group of $N$ people

Need to accurately compute the average of their salaries

No person should reveal his/her salary to any other person

No person should try to steal the salary information from any other person(s)

Should be done in $O(N)$ time
Zero–Knowledge Proofs

Zero knowledge proofs, are proofs that yield no information apart from the validity of the claim we wanted to prove.

Given any input $x$, anything that the verifier can compute efficiently after the interaction with prover on $x$, could also be computed before the interaction.

Showing a protocol is zero knowledge guarantees a high level of security for the protocol, since no matter what the verifier does, he does not get any new information about the prover’s secrets.
Where’s Waldo?
Zero-Knowledge Protocols

Challenge and Response Protocol

Prover knows a secret $S$

Prover needs to demonstrate to Verifier his knowledge of $S$, without revealing $S$ (or any information about $S$, from which $S$ can be derived)

Outcome of a Zero-Knowledge Protocol:

Verifier is convinced that Prover knows $S$

Verifier gains no information about $S$

Most Z-K Protocols operate iteratively
This process can be repeated as many times as the Verifier wishes.

 IF the Prover is just lucky guessing, then the probability of being correct every time in:
 10 trials is less than 1 in ~1,000
 20 trials is less than 1 in a ~Million
 30 trials is less than 1 in a ~Billion
 So, in about 20 or 30 trials the Verifier is convinced that the Prover knows the password (upon no incorrect outcomes)
ZKP Using $k$-Colorability of (Arbitrary) Graphs

Given an arbitrary graph $G$, 
“Is $G$ $k$-colorable?” is a hard question

Suppose the Prover has colored a large graph $G$ using “$k$” colors. It can be the basis of a Zero-knowledge protocol as follows:
The colored vertices are covered.
The Verifier gets to choose an edge arbitrarily
The corresponding vertices are uncovered to reveal the colors
   If they are of the same color, then the Prover’s claim is false
ZKP Using \( k \)-Colorability of (Arbitrary) Graphs (2)

The Prover then chooses a different set of colors for the vertices (and they are covered).

Again, the Verifier gets to choose an arbitrary edge.

The process is repeated an arbitrarily large number of iterations, until the Verifier is convinced.

However, the Verifier cannot get (reconstruct) any information about the \( k \)-coloring of \( G \).
In Conclusion …

- Diffie–Hellman solved an age old problem of secure key distribution
- RSA introduced a radically new way of encryption and authentication
- Public key system is also used in distribution of keys of symmetric key encryption
- Public key certificates provide efficient way of public key distribution
- Digital signatures provide authenticity and non–repudiation
Thank You