Public Key Cryptography

Instructor: Sencun Zhu

Public Key Algorithms

• Public key algorithms
  – RSA: encryption and digital signature
  – Diffie-Hellman: key exchange
  – DSA: digital signature
  – Elliptic curve: encryption and digital signature
• Number theory underlies most of public key algorithms
  – Prime numbers
  – The Chinese Remainder Theorem (CRT)
  – Discrete Logarithms

Applications

• Encryption/decryption
  – The sender encrypts a message with the receiver’s public key
  – Only the receiver can decrypt the message.
• Digital signature
  – The sender signs a message with its private key.
  – Authentication and non-repudiation
• Key exchange
  – Two sides cooperate to exchange a session key.
  – Secret key cryptosystems are often used with the session key.
Requirements

- It is **computationally easy** to generate a pair of public key and private key.
- It is **computationally easy** to generate a ciphertext using the public key.
- It is **computationally easy** to decrypt the ciphertext using the private key.
- It is **computationally infeasible** to determine the private key from the public key.
- It is **computationally infeasible** to recover the message from the ciphertext and the public key.

Trapdoor One-Way Function

- One-way function $f$
  - $Y = f(X)$: easy
  - $X = f^{-1}(Y)$: infeasible
- Trapdoor one-way function
  - $Y = f_k(X)$: easy if $k$ and $X$ are known
  - $X = f_k^{-1}(Y)$: easy if $k$ and $Y$ are known
  - $X = f_k^{-1}(Y)$: infeasible if $Y$ is known but $k$ is unknown.
- Designing public-key algorithm is to find appropriate trapdoor one-way function

Public-Key Cryptanalysis

- Brute-force attack
  - Try all possible keys
- Derivation of private key from public key
  - Try to find the relationship between the public key and the private key and compute the private key from the public one
- Probable-message attack
  - The public key is known
  - Encrypt all possible messages
  - Try to find a match between the ciphertext and one of the above encrypted messages
A Little Background

• Theorem 1 (Euclid):
  – There are an infinite number of primes
• In the neighborhood of a number \( n \), approximately one in every \( \ln(n) \) numbers is prime
  – \( \ln(n) \approx 0.7 \log_2(n) \)
  – In the neighborhood of a number of 2000 bits \( (2^{1999} \sim 2^{2000}) \), there is about one prime in every 1386 numbers

A Little More Background

• Generate Large Prime
  – Input: lower/upper bound \([l, u]\) of range in which prime should lie
  – Output: a random prime \( p \) in \([l, u]\)
  – Method: randomly choose a number in the range and test its primality until success
• Generate a secure prime
  – The range \([l, u]\) should be large enough so that there are at least a certain number (e.g., \(2^{128}\)) of primes in the interval
  – Why?
• Do not compare the size of a symmetric key to that of a public key

RSA (Rivest, Shamir, Adleman)

• The most popular one
• Support both public key encryption and digital signature
• Assumption/theoretical basis
  – Factorization of large primes is hard
• Variable key length
  – 1024, 2048, 4096 bits (512 bits was broken)
• Variable plaintext block size
  – Plaintext must be “smaller” than the key
  – Ciphertext block size is the same as the key length
RSA Algorithm

• Key pair generation
  – Pick large primes $p$ and $q$
  – Let $n = p \cdot q$, keep $p$ and $q$ to yourself!
  – For public key, choose $e$ that is relatively prime to $\phi(n) = (p-1)(q-1)$, let pub = $<e,n>$
  – For private key, find $d$ that is the multiplicative inverse of $e \mod \phi(n)$, i.e., $e \cdot d = 1 \mod \phi(n)$, let pri = $<d,n>$

• In practice
  – small public exponents $e = 3, 5, 17, \text{ or } 65537$
  – Very large $n$, at least 1024 bits

How Does RSA Work?

• Given pub = $<e, n>$ and priv = $<d, n>$
  – encryption: $c = m^e \mod n$, $m < n$
  – decryption: $m = c^d \mod n$
  – signature: $s = m^d \mod n$, $m < n$
  – verification: $m = s^e \mod n$

• Correctness
  – Decryption: $m = c^d \mod n = m^{e \cdot d} \mod n = m \mod n$ (since $m < n$)
  – Signature similar

An Example

• Choose $p = 7$ and $q = 17$.
• Compute $n = p \cdot q = __$.  
• Compute $\phi(n) = (p-1)(q-1) = __$.
• Select $e = 5$, which is relatively prime to $\phi(n)$.
• Compute $d = __77\_\_$ such that $e \cdot d = 1 \mod \phi(n)$.
• Public key: $<__\_, \_\_\_\_>$
• Private key: $<__\_, \_\_\_\_>$
• Encryption: $19^5 \mod 119 = 66$
• Decryption: $66^{77} \mod 119 = 19$.  

The Security of RSA

- **Attacks against RSA**
  - **Brute force:** Try all possible private keys
  - Can be defeated by using a large key space
- **Mathematical attacks**
  - Factor $n = pq$.
  - Determine $\phi(n)$ directly; equivalent to factoring $n$.
  - Determine $d$ directly: at least as difficult as factoring $n$.
- **Timing attacks**
  - Recover the private key according to the running time of the decryption algorithm.

Diffie-Hellman Key Exchange

- **A public-key distribution scheme**
  - can establish a common key known only to the two participants
  - cannot be used to exchange an arbitrary message
- **Value of key depends on the participants**
  (and their private and public key information)
- **Theoretical basis**
  - Exponentiation in a finite (Galois) field (modulo a prime or a polynomial) - easy
  - Security relies on the difficulty of computing discrete logarithms (similar to factoring) – hard

Diffie-Hellman Setup

- **All users agree on global parameters:**
  - large prime integer or polynomial $q$
  - $\alpha$ a primitive root mod $q$
- **Each user (eg. A) generates two keys**
  - chooses a secret key (number): $x_A < q$
  - compute their public key: $y_A = \alpha^{x_A} \mod q$
- **Each user makes public the key $y_A$**
Diffie-Hellman Key Exchange

- Shared session key for users A & B is $K_{AB}$:
  $$K_{AB} = y_a^x \mod q$$
  $$= y_b^x \mod q \quad \text{(which B can compute)}$$
  $$= y_b^x \mod q \quad \text{(which A can compute)}$$
- $K_{AB}$ is used as session key in symmetric-key encryption scheme between Alice and Bob
- If Alice and Bob subsequently communicate, they will have the same key as before, unless they choose new public-keys
- Attacker must solve discrete log to obtain $x$

Diffie-Hellman Example

- users Alice & Bob who wish to swap keys:
- agree on prime $q=353$ and $\alpha=3$
- select random secret keys:
  - A chooses $x_A=97$, B chooses $x_B=233$
- compute public keys:
  - $y_A=3^{97} \mod 353 = 40$ (Alice)
  - $y_B=3^{233} \mod 353 = 248$ (Bob)
- compute shared session key as:
  $$K_{AB}= y_B^{x_A} \mod 353 = 248^{97} = 160 \quad \text{(Alice)}$$
  $$K_{AB}= y_A^{x_B} \mod 353 = 40^{233} = 160 \quad \text{(Bob)}$$

Security of DH Scheme

- Security factors
  - Discrete logarithm very difficult
  - Shared key (the secret) itself never transmitted
- Security vulnerability
  - No authentication of partners
    - Man-in-the-middle attack
  - Session key provides no non-repudiation, so you cannot sign anything
Man-In-The-Middle Attack

<table>
<thead>
<tr>
<th>Alice</th>
<th>Eve</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g^{Sa} = 123$</td>
<td>$g^{Se} = 654$</td>
<td>$g^{Sb} = 255$</td>
</tr>
<tr>
<td>123 → 654</td>
<td>654 → 255</td>
<td></td>
</tr>
<tr>
<td>$654^{Se} = 123^{Sb}$</td>
<td>$255^{Se} = 654^{Sb}$</td>
<td></td>
</tr>
</tbody>
</table>

• Eve impersonates Bob to Alice and Alice to Bob

Phone Book Mode

• DH is subject to active man-in-the-middle attack because their public key-component may be intercepted and substituted
• Phone book mode allows everyone to generate the public key-component in advance and publish them through other reliable means
• All communicating parties agree on their common $<g, p>$
• Essential requirement: authenticity of the public key.

An Example Application

• Connected to lab-218.cse.psu.edu:22.
• Exchanging SSH version...Done.
• Server: SSH-2.0-OpenSSH_3.8.1p1.
• Client: SSH-2.0-Xssh_1.0.
• SSH2 is enabled.
• Negotiating algorithms...
• Initiating key exchange...Done.
• Exchanging Diffie-Hellman complete message...Done.
• Waiting for new key message...Received.
• Precompute: 0124 36 db/9 13 ad 80 65 3a 73 da/6e/ce/be 1a bb Verifying host key...
• Algorithm negotiation has been finished.
• CS Cipher: 3des-cbc, SC Cipher: 3des-cbc
• CS Compress: zlib, SC Compress: zlib
• CS MAC: hmac-sha1, SC MAC: hmac-sha1
• Trying to login as szhu.
• Trying password authentication...Done.
• Requesting session channel...Done.
• Requesting X11 forwarding...Done.
• Executing remote command...Done.
• X11 channel request arrived.
• Establishing X11 connection (x=2)...Done.
Public Key Infrastructure (PKI)

- An infrastructure that allows you to recognize which public key belongs to whom
  - Binding a public key to an identity
  - Involving a trusted certificate authority (CA)
  - CA signs the public key of a user
    - “I, the CA, have verified that public key $P_A$ belongs to Alice”
  - Everybody knows the public key of the CA
  - Often a hierarchy providing multilevel certificates (or certificate chain)

Certificates

- Content of a certificate
  - A unique sequence number
  - Identity of principal (who uses it)
  - Corresponding public key
  - Timestamp
    - when issued and when to expire
  - Other information
    - Can this principle issue certificates to others
    - What is the purpose of the public key
      - Encryption or signature

Certificate Revocation

- The hardest problem to solve in PKI
- The key issue: how do you know if a presented certificate is still valid?
- Why revocation before expiration?
  - Public/private key compromise
  - Change your company / fired
  - CA made a mistake
  - And so on…
**Solutions**

- **Fast expiration**
  - Use a short expiration time, say minutes or hours
  - Ask for a new certificate after it is expired
  - Problem?
- **CRL**
  - Publish the list of the revoked certificates
  - Signed by CA
  - Stored in some directory services
  - Not necessary to be trusted

**CRL**

- **How to sign CRL**
  - Entire list
  - Every individual
  - Certificate revocation tree (CRT)
- **How to distribute CRL**
  - Push
  - Pull
  - Online/offline

**PKI Difficulties**

- **What is a name?**
  - Hard to have a unique name
- **Who has the authority?**
  - Government?
- **Whom to trust?**
  - No one is trusted by everybody in the world
- **What are authorized?**
  - Keys, not identities