ΥΣ13 - Computer Security

Public-Key Cryptography

Κώστας Χατζηκοκολάκης
Context

• **Goal**
  - Confidentiality
  - Alice wants to send a message \( P(\text{plaintext}) \) to Bob
  - Only Bob should be able to read it
Context

• **Goal**
  - Confidentiality
  - Alice wants to send a message $P$ (plaintext) to Bob
  - Only Bob should be able to read it

• **Solution**: symmetric encryption
  - wait, we’ve done this lecture before!
  - we known about block ciphers, DES, AES, …
• **Goal**
  - Confidentiality
  - Alice wants to send a message $P(plaintext)$ to Bob
  - Only Bob should be able to read it

• **Solution**: symmetric encryption
  - wait, we’ve done this lecture before!
  - we known about block ciphers, DES, AES, …
  - but *are we satisfied with the solution?*
Goal

- Confidentiality
- Alice wants to send a message $P$ (plaintext) to Bob
- Only Bob should be able to read it

Solution: symmetric encryption

- wait, we’ve done this lecture before!
- we known about block ciphers, DES, AES, …
- but are we satisfied with the solution?
- Alice and Bob need to share a key
  - $n$ users : $n^2$ keys
- Can we share keys safely?
First solution: **Trusted Third Party**

- shares keys with every user \((K_A, K_B, \ldots)\)
  - \(n\) users : \(n\) keys

- When Alice wants to communicate to Bob
  - TTP generates a new key \(K_{AB}\)
  - Sends it to both Alice and Bob

Problems:

- Availability: TTP needs to be online
- Trust
First solution: **Trusted Third Party**

- shares keys with every user \((K_A, K_B, \ldots)\)
  - \(n\) users : \(n\) keys

- When Alice wants to communicate to Bob
  - TTP generates a new key \(K_{AB}\)
  - Sends it to both Alice and Bob

- Single communication to TTP
  - TTP → A : \(\{A, B, K_{AB}\}_{K_A}, \{A, B, K_{AB}\}_{K_B}\)
  - A → B : \(\{A, B, K_{AB}\}_{K_B}, \{M\}_{K_{AB}}\)
Context

First solution: **Trusted Third Party**

- shares keys with every user \((K_A, K_B, \ldots)\)
  - \(n\) users : \(n\) keys

- When Alice wants to communicate to Bob
  - TTP generates a new key \(K_{AB}\)
  - Sends it to both Alice and Bob

- Single communication to TTP
  - TTP → A : \(\{A, B, K_{AB}\}_{K_A}, \{A, B, K_{AB}\}_{K_B}\)
  - A → B : \(\{A, B, K_{AB}\}_{K_B}, \{M\}_{K_{AB}}\)

- **Problems?**
First solution: **Trusted Third Party**

- shares keys with every user \((K_A, K_B, \ldots)\)
  - \(n\) users : \(n\) keys

- When Alice wants to communicate to Bob
  - TTP generates a new key \(K_{AB}\)
  - Sends it to both Alice and Bob

- Single communication to TTP
  - TTP \(\rightarrow\) A : \(\{A, B, K_{AB}\}_{K_A}, \{A, B, K_{AB}\}_{K_B}\)
  - A \(\rightarrow\) B : \(\{A, B, K_{AB}\}_{K_B}, \{M\}_{K_{AB}}\)

- **Problems?**
  - Availability: TTP needs to be online
  - Trust
Better solution: **establish a new key**

- **No** shared secret
- Communication over a **public** channel
- **Is this possible?**
  - The adversary has exactly **the same information** as Alice and Bob!
Better solution: **establish a new key**

- **No** shared secret

- Communication over a **public** channel

- Is this possible?
  - The adversary has exactly **the same information** as Alice and Bob!

- **Key insight**
  - Make the adversary **work (much) harder** than Alice and Bob
Merkle’s puzzles (1978)

- Alice generates $n$ keys, hides each $K_i$ in a “puzzle”
  - Sends them to Bob
- Each puzzle needs $n$ steps to solve
  - Eg. use block cipher with a small key
- Each puzzle has an id $x_i$ contained in the puzzle
Merkle’s puzzles (1978)

- Bob selects random *j*, solves the *j*-th puzzle
  - obtains *x*ₖ and *k*ₖ
- Sends *x*ₖ to Alice
- Alice and Bob use *k*ₖ as their established key
Merkle’s puzzles (1978)

- Bob selects random $j$, solves the $j$-th puzzle
  - obtains $x_j$ and $k_j$
- Sends $x_j$ to Alice
- Alice and Bob use $k_j$ as their established key
- Is this secure?
Merkle’s puzzles (1978)

Is this secure?

• $x_j$ cannot be easily associated to $j$

• The adversary needs to solve all puzzles

• Computation time
  - Alice, Bob: $O(n)$ time
  - Adversary: $O(n^2)$

• Not good enough by modern standards
Encryption without shared keys

• Can we do better?

• Use problems that are
  - *polynomial* for Alice, Bob
  - *exponential* for the adversary

• Such problems do exist!
  - *Discrete logarithm*
  - *Factorization*

• Major breakthroughs
  - 1976, Diffie & Hellman: key exchange protocol
  - 1978, Rivest, Shamir & Adleman: public key encryption

• Both discovered previously by GCHQ (British intelligence agency)
• Can we do better?

• Use problems that are
  - *polynomial* for Alice, Bob
  - *exponential* for the adversary

• Such problems do exist!
  - Discrete logarithm
  - Factorization

Major breakthroughs:
- 1976, Diffie & Hellman: key exchange protocol
- 1978, Rivest, Shamir & Adleman: public key encryption
- Both discovered previously by GCHQ (british intelligence agency)
Encryption without shared keys

• Can we do better?

• Use problems that are
  - polynomial for Alice, Bob
  - exponential for the adversary

• Such problems do exist!
  - Discrete logarithm
  - Factorization

• Major breakthroughs
  - 1976, Diffie & Hellman: key exchange protocol
  - 1978, Rivest, Shamir & Adleman: public key encryption
  - Both discovered previously by GCHQ (British intelligence agency)
Discrete logarithm

- $p$: large prime (say 2048 bits)

- $\mathbb{Z}_p^* = \{1, \ldots, p - 1\}$: a group under multiplication modulo $p$
Discrete logarithm

- \( p \): large prime (say 2048 bits)

- \( \mathbb{Z}_p^* = \{1, \ldots, p - 1\} \): a group under multiplication modulo \( p \)

- Moreover: a cyclic group
  - \( g \) a (small) number such that
  - \( g^k \mod p \quad k = 1..p - 1 \)
  - is a permutation of \( \mathbb{Z}_p^* \)
Discrete logarithm

- $p$: large prime (say 2048 bits)
- $\mathbb{Z}_p^* = \{1, \ldots, p - 1\}$: a group under multiplication modulo $p$
- Moreover: a cyclic group
  - $g$ a (small) number such that
  - $g^k \mod p \quad k = 1..p - 1$
  - is a permutation of $\mathbb{Z}_p^*$
- In other words
  - each $a \in \mathbb{Z}_p^*$ can be written as
  - $g^k \mod p$ for some $k$
Discrete logarithm

• Exponentiation
  - $x \mapsto g^x \mod p$
  - **Easy**: exponentiation by squaring
    $x^n = \begin{cases} 
      x(x^2)^{\frac{n-1}{2}}, & \text{if } n \text{ is odd} \\
      (x^2)^{\frac{n}{2}}, & \text{if } n \text{ is even.} 
    \end{cases}$
Discrete logarithm

• Exponentiation
  - \( x \mapsto g^x \mod p \)
  - Easy: exponentiation by squaring
    \[
    x^n = \begin{cases} 
    x \left( x^2 \right)^{\frac{n-1}{2}}, & \text{if } n \text{ is odd} \\
    \left( x^2 \right)^{\frac{n}{2}}, & \text{if } n \text{ is even.}
    \end{cases}
    \]

• Discrete logarithm
  - \( a = g^x \mod p \mapsto x \)
  - Hard
• **Goal**
  - Establish a shared key

• **Basic idea**
  - use secrets that can be “mixed”
  - but not “unmixed”
Diffie-Hellman

Alice

agree $p, g$

$a \leftarrow \mathbb{Z}_p^*$

$A \leftarrow g^a$

Bob

$b \leftarrow \mathbb{Z}_p^*$

$B \leftarrow g^b$

$K \leftarrow B^a = g^{ab}$

$K \leftarrow A^b = g^{ab}$
Diffie-Hellman

Why is this secure?

• **Diffie-Hellman** problem (DH)
  - Given $g, g^a, g^b$, compute $g^{ab}$

• **Discrete Logarithm** problem (DL)
  - Given $g, g^x$, compute $x$

• Both believed to be hard
  - DH is no harder than DL
  - Whether the converse holds is unknown!
• Generalized Diffie-Hellman
  - Exactly the same thing, on some other finite cyclic group!
  - Works as long as exponentiation is easy by logarithm is hard

• Elliptic curves
  - Points on a curve with a group operation
  - Advantage: no specialized discrete logarithm algorithms (in contrast to $\mathbb{Z}_p^*$)
  - So: harder problem, shorter keys!
Diffie-Hellman

• We have established a key with whoever has the matching $b$
  - How do we know that this is Bob?
Diffie-Hellman

• We have established a key with whoever has the matching $b$
  - How do we know that this is Bob?
  - We don’t!

![Diagram showing the Diffie-Hellman key exchange process](Diagram.png)
Public-Key Cryptography

- Use **pairs** of keys
  - public key \( pk \) : can be sent in clear
  - secret key \( sk \) : kept private

- **Operations**
  - Encryption : \( C = Enc(pk, P) \)
  - Decryption : \( P = Dec(sk, C) \)

- **Correctness**
  - \( Dec(sk, Enc(pk, P)) = P \) for any plaintext \( P \)
Public-Key Cryptography

From DH to PK Encryption

• Keys
  - secret key: \( sk = a \)
  - public key: \( pk = g^a \quad (g, p \text{ public}) \)

• Encryption
  - \( Enc(pk, P) = (k_e, AES_{enc}(k_m, P)) \quad \text{where} \ b \ \text{random}, \ k_e = g^b, k_m = pk^b \)

• Decryption
  - \( Dec(sk, (k_e, C)) = AES_{dec}(k_m, C) \quad \text{where} \ k_m = k_e^{sk} \)
Public-Key Cryptography

From DH to PK Encryption

• Keys
  - secret key: $sk = a$
  - public key: $pk = g^a$ (g, p public)

• Encryption
  - $Enc(pk, P) = (k_e, AES_{enc}(k_m, P))$ where $b$ random, $k_e = g^b$, $k_m = pk^b$

• Decryption
  - $Dec(sk, (k_e, C)) = AES_{dec}(k_m, C)$ where $k_m = k_e^{sk}$

• Can we do it without a symmetric encryption?
  - Elgamal!
Alice
choose $p, g$

$sk \leftarrow \mathbb{Z}_p^*$

$pk \leftarrow g^{sk}$

Bob

$p, g, pk$

$b \leftarrow \mathbb{Z}_p^*$

$k_e \leftarrow g^b$

$k_m \leftarrow pk^b$

$y \leftarrow x \cdot k_m$

$k_m \leftarrow k_e^{sk}$

$x \leftarrow y \cdot k_m^{-1}$
If we have PK encryption we can easily perform key exchange

Alice
generate pk, sk

Bob

\[ k \leftarrow \{0, 1\}^m \]

\[ y = Enc(pk, k) \]

\[ k \leftarrow Dec(sk, y) \]
Factorization

• $p, q$: large primes

• **Multiplication**
  - $p, q \mapsto pq$
  - Easy

• **Factorization**
  - $pq \mapsto p, q$
  - Hard
• Initialization
  - Select $p, q$: large random primes (eg 2048 bits), $n = pq$
  - Select $e$: small prime

• Public key
  - $pk = (n, e)$
RSA

- **Initialization**
  - Select $p, q$: large random primes (eg 2048 bits), $n = pq$
  - Select $e$: small prime

- **Public key**
  - $pk = (n, e)$

- **Private key**
  - $sk = d = e^{-1} \mod \Phi(n)$ where $\Phi(n) = (p - 1)(q - 1)$
  - We can show that: $\forall x: x^{ed} = x \mod n$
RSA

• Initialization
  - Select $p, q$: large random primes (eg 2048 bits), $n = pq$
  - Select $e$: small prime

• Public key
  - $pk = (n, e)$

• Private key
  - $sk = d = e^{-1} \mod \Phi(n)$ where $\Phi(n) = (p - 1)(q - 1)$
  - We can show that: $\forall x: x^{ed} = x \mod n$

• Encryption: $y = x^e \mod n$

• Decryption: $x = y^d \mod n$
RSA

Alice

choose $p$, $q$, $e$

$n \leftarrow pq$

$sk \leftarrow e^{-1} \mod \Phi(n)$

Bob

$pk = (n, e)$

$y \leftarrow x^e \mod n$

$x \leftarrow y^{sk} \mod n$
Why is this secure?

• **RSA** problem (e-th root)
  - Given $n = pq$, $e$, $x^e \mod n$
  - compute $x$

• **Factorization** problem (DL)
  - Given $n = pq$
  - compute $p, q$

• Both **believed** to be hard
  - RSA is no harder than Factorization
  - Whether the converse holds is unknown!
Key sizes

- The security of each cryptosystem is estimated based on the best known algorithms
- Current records
  - Factorization: 829 bits
  - Discrete logarithm: 795 bits

<table>
<thead>
<tr>
<th>Algorithm Family</th>
<th>Cryptosystems</th>
<th>Security Level (bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>80</td>
</tr>
<tr>
<td>Integer factorization</td>
<td>RSA</td>
<td>1024 bit</td>
</tr>
<tr>
<td>Discrete logarithm</td>
<td>DH, DSA, Elgamal</td>
<td>1024 bit</td>
</tr>
<tr>
<td>Elliptic curves</td>
<td>ECDH, ECDSA</td>
<td>160 bit</td>
</tr>
<tr>
<td>Symmetric-key</td>
<td>AES, 3DES</td>
<td>80 bit</td>
</tr>
</tbody>
</table>
Security models

• A game modeling the adversary’s goal and capabilities
  - No choice of plaintext/ciphertext

Choose M somehow

PK

Challenger

Adv

Ciphertext encrypting M

Is it enough? What about messages not chosen by challenger?

Find M
Security models

- A **game** modeling the adversary’s **goal** and **capabilities**
  - Chosen plaintext (IND-CPA)
Security models

- A game modeling the adversary’s goal and capabilities
  - Chosen ciphertext (IND-CCA1)
Security models

- A game modeling the adversary’s goal and capabilities
  - Chosen ciphertext, adaptive (IND-CCA2)
Security models

• Is (schoolbook) RSA IND-CCA2 secure?

No!

Problems

- Deterministic
- Malleable

Solution

- Random padding
Security models

• Is (schoolbook) RSA IND-CCA2 secure?
  - No!

• Problems
  - Deterministic
  - Malleable
Security models

• Is (schoolbook) RSA IND-CCA2 secure?
  - No!

• Problems
  - Deterministic
  - Malleable

• Solution
  - Random padding
Digital signatures

- **Problem**
  - So far we assume an external adversary
  - What if Alice cannot be trusted?
  - With a shared key
    - any encrypted message can be generated by both Alice and Bob
Digital signatures

• Problem
  - So far we assume an external adversary
  - What if Alice cannot be trusted?
  - With a shared key
    • any encrypted message can be generated by both Alice and Bob

• Signatures
  - generated with the sk of Alice
  - verified with the pk of Alice
RSA signatures

Alice
choose $p$, $q$, $e$

$n \leftarrow pq$

$sk \leftarrow e^{-1} \mod \Phi(n)$

$s \leftarrow x^{sk} \mod n$

$x, s, pk = (n, e)$

Bob

check $x = s^e \mod n$
RSA signatures

Alice
choose $p, q, e$

$n \leftarrow pq$

$sk \leftarrow e^{-1} \mod \Phi(n)$

$s \leftarrow x^{sk} \mod n$

Bob

$x, s, pk = (n, e)$

check $x = s^e \mod n$
• Are (schoolbook) RSA signatures secure?

- Select $s \in \mathbb{Z}_p^*$
- This is a valid signature for $x = s e \mod n$
• Are (schoolbook) RSA signatures secure?

• The adversary can forge a signature of a random message
  - Select $s \leftarrow \mathbb{Z}_p^*$
  - This is a valid signature for $x = s^e \mod n$!
• Are (schoolbook) RSA signatures secure?

• The adversary can forge a signature of a random message
  - Select $s \leftarrow \mathbb{Z}_p^*$
  - This is a valid signature for $x = s^e \mod n$!

• Solution
  - Random padding
• Ross Anderson, Security Engineering, Sections 5.7

• W Diffie, ME Hellman, “New Directions in Cryptography”, in IEEE Transactions on information theory v 22 no 6 (Nov 76) pp 644–654