ΥΣ13 - Computer Security

Public-Key Cryptography

Κώστας Χατζηκοκολάκης

Context

- **Goal**
  - Confidentiality
  - Alice wants to send a message P (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, …

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, …
  - but are we satisfied with the solution?

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, …
  - 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?

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

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 $\rightarrow$ A : $\{A, B, K_{AB}\}_{K_A}, \{A, B, K_{AB}\}_{K_B}$
  - A $\quad\rightarrow$ 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 $\rightarrow$ 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?

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 $\rightarrow$ 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

Context

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!

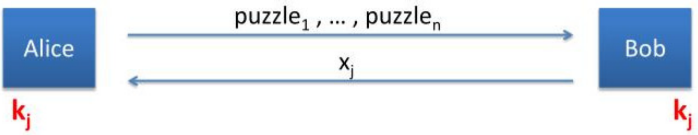
Context

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

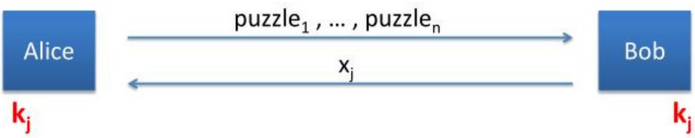
Merkle's puzzles (1978)

- 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



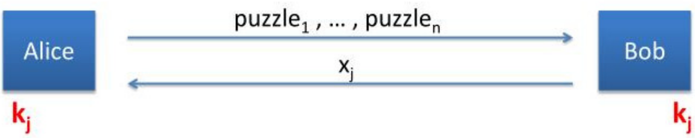
Merkle's puzzles (1978)

- Bob selects random j , solves the j-th puzzle
  - obtains x_j and k_j
- Sends x_j to Alice
- Alice and Bob use k_j as their established key



Merkle's puzzles (1978)

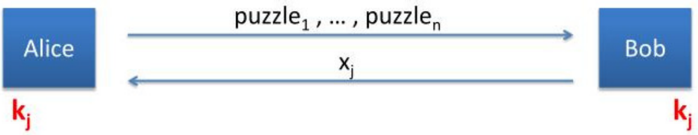
- Bob selects random j , solves the j-th puzzle
  - obtains x_j and k_j
- Sends x_j to Alice
- Alice and Bob use k_j as their established key
- Is this secure?



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



Encyption without shared keys

- Can we do better?
- Use problems that are
  - polynomial for Alice, Bob
  - exponential for the adversary

Encyption 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

Encyption 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
- Morover: a cyclic group
  - g a (small) number such that
  - $g^k \bmod p \qquad k = 1..p-1$
  - is a permutation of $\mathbb{Z}_p^*$

Discrete logarithm

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

Discrete logarithm

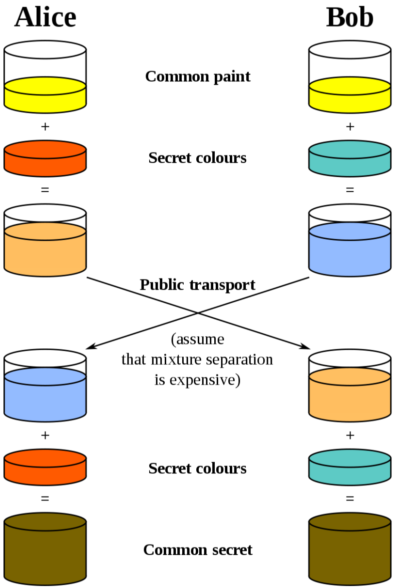
- **Exponentiation**
  - $x \mapsto g^x \bmod p$
  - Easy : exponentiation by squaring
    - $$x^n = \begin{cases} x\,(x^2)^{\frac{n-1}{2}}, & \text{if } n \text{ is odd} \\ (x^2)^{\frac{n}{2}}, & \text{if } n \text{ is even.} \end{cases}$$

Discrete logarithm

- **Exponentiation**
  - $x \mapsto g^x \bmod p$
  - 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**
  - $a = g^x \bmod p \mapsto x$
  - Hard

Diffie-Hellman

- **Goal**
  - Establish a shared key
- Basic idea
  - use secrets that can be “mixed”
  - but not “unmixed”



Diffie-Hellman

Alice		**Bob**
	agree p, g	
$a \xleftarrow{\$} \mathbb{Z}_p^*$		
$A \leftarrow g^a$		
	$\xrightarrow{\quad A \quad}$	
		$b \xleftarrow{\$} \mathbb{Z}_p^*$
		$B \leftarrow g^b$
	$\xleftarrow{\quad B \quad}$	
$K \leftarrow B^a = g^{ab}$		$K \leftarrow A^b = g^{ab}$

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!

Diffie-Hellman

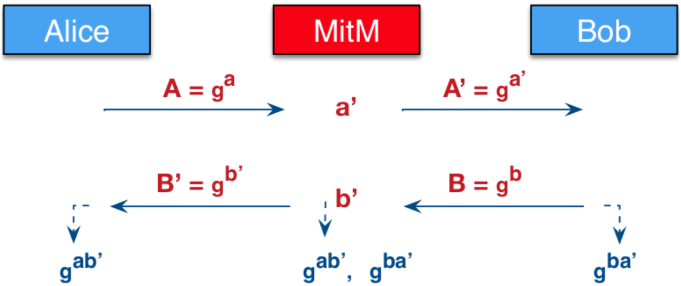
- **Generalized Diffie-Hellman**
  - Exactly the same thing, on some other finite cyclic group!
  - Works as long as exponentiation is easy by logarithm is hard
- **Elliptic curves**
  - Points on a curve with a group operation
  - Advantage: no specialized discrete logarithm algorithms (in contrast 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?

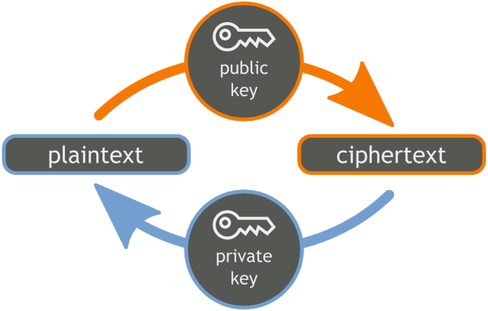
Diffie-Hellman

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



Public-Key Cryptography

- Use pairs of keys
  - public key pk : can be sent in clear
  - secret key sk : kept private
- Operations
  - Encryption : $C = Enc(\mathsf{pk}, P)$
  - Decryption : $P = Dec(\mathsf{sk}, C)$
- Correctness
  - $Dec(\mathsf{sk}, Enc(\mathsf{pk}, P)) = P$
    for any plaintext P



Public-Key Cryptography

From DH to PK Encryption

- Keys
  - secret key : $\mathsf{sk} = a$
  - public key : $\mathsf{pk} = g^a$ $\quad(g, p$ public)
- Encryption
  - $Enc(\mathsf{pk}, P) = (\mathsf{k}_e, AES_{enc}(\mathsf{pk}^b, P))$ $\quad$ where b random, $\mathsf{k}_e = g^b$,
- Decryption
  - $Dec(\mathsf{sk}, (\mathsf{k}_e, C)) = AES_{dec}(\mathsf{k}_e^{\mathsf{sk}}, C)$

Public-Key Cryptography

From DH to PK Encryption

- Keys
  - secret key : $\mathsf{sk} = a$
  - public key : $\mathsf{pk} = g^a$ (g, p public)
- Encryption
  - $Enc(\mathsf{pk}, P) = (\mathsf{k}_e, AES_{enc}(\mathsf{pk}^b, P))$ where b random, $\mathsf{k}_e = g^b$,
- Decryption
  - $Dec(\mathsf{sk}, (\mathsf{k}_e, C)) = AES_{dec}(\mathsf{k}_e^{\mathsf{sk}}, C)$
- Can we do it without a symmetric encryption?
  - Elgamal!

Elgamal

Alice		**Bob**
choose p, g		
$\mathsf{sk} \leftarrow_{\$} \mathbb{Z}_p^*$		
$\mathsf{pk} \leftarrow g^{\mathsf{sk}}$		
	$\xrightarrow{p,\, g,\, \mathsf{pk}}$	
		$b \leftarrow_{\$} \mathbb{Z}_p^*$
		$\mathsf{k}_e \leftarrow g^b$
		$\mathsf{k}_m \leftarrow \mathsf{pk}^b$
		$y \leftarrow x \cdot \mathsf{k}_m$
	$\xleftarrow{\mathsf{k}_m,\, y}$	
$\mathsf{k}_m \leftarrow \mathsf{k}_e^{\mathsf{sk}}$		
$x \leftarrow y \cdot \mathsf{k}_m^{-1}$		

From PK Encryption to Key Exchange

If we have PK encryption we can easily perform key exchange

Alice		**Bob**
generate pk, sk		
	$\xrightarrow{\mathsf{pk}}$	
		$\mathsf{k} \leftarrow_{\$} \{0,1\}^m$
	$\xleftarrow{y = Enc(\mathsf{pk}, \mathsf{k})}$	
$\mathsf{k} \leftarrow Dec(\mathsf{sk}, y)$		

Factorization

- p, q : large primes
- **Multiplication**
  - $p, q \mapsto pq$
  - Easy
- **Factorization**
  - $pq \mapsto p, q$
  - Hard

RSA

- Initialization
  - Select p, q : large random primes (eg 2048 bits), $n = pq$
  - Select e : small prime
- **Public key**
  - $\mathrm{pk} = (n, e)$

RSA

- Initialization
  - Select p, q : large random primes (eg 2048 bits), $n = pq$
  - Select e : small prime
- **Public key**
  - $\text{pk} = (n, e)$
- **Private key**
  - $\text{sk} = d = e^{-1} \bmod \Phi(n)$ where $\Phi(n) = (p-1)(q-1)$
  - We can show that : $\forall x : x^{ed} = x \bmod n$

RSA

- Initialization
  - Select p, q : large random primes (eg 2048 bits), $n = pq$
  - Select e : small prime
- **Public key**
  - $\mathrm{pk} = (n, e)$
- **Private key**
  - $\mathrm{sk} = d = e^{-1} \bmod \Phi(n)$ where $\Phi(n) = (p-1)(q-1)$
  - We can show that : $\forall x : x^{ed} = x \bmod n$
- **Encryption** : $y = x^e \bmod n$
- **Decryption** : $x = y^d \bmod n$

RSA

Alice **Bob**

choose p, q, e

$n \leftarrow pq$

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

$$\xrightarrow{\mathsf{pk} = (n, e)}$$

$y \leftarrow x^e \bmod n$

$$\xleftarrow{y}$$

$x \leftarrow y^{\mathsf{sk}} \bmod n$

RSA

Why is this secure?

- RSA problem (e-th root)
  - Given $n = pq$, e, $x^e \bmod n$
  - 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!

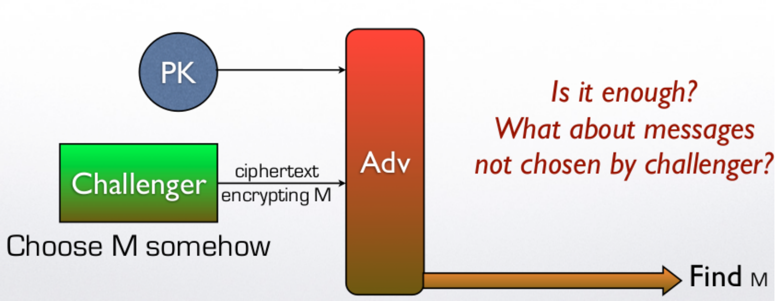
Key sizes

- The security of each cryptosystem is estimated based on the best known algorithms
- Current records
  - Factorization : 768 bits
  - Discrete logarithm : 768 bits

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

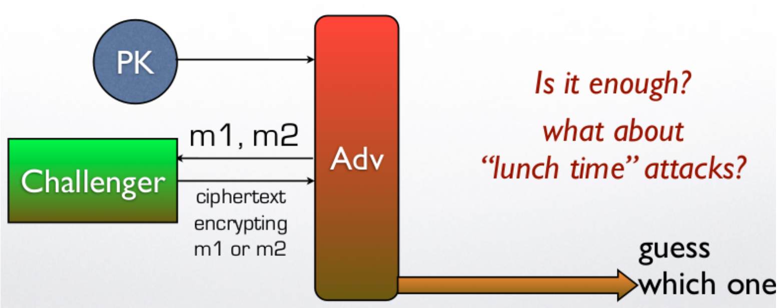
Security models

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



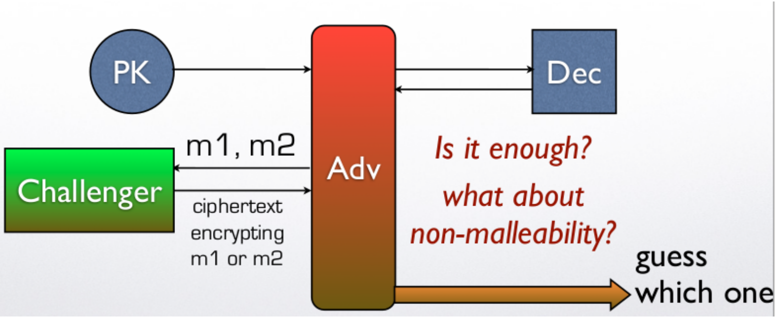
Security models

- A game modeling the the adversary's goal and capabilities
  - Chosen plaintext (IND-CPA)



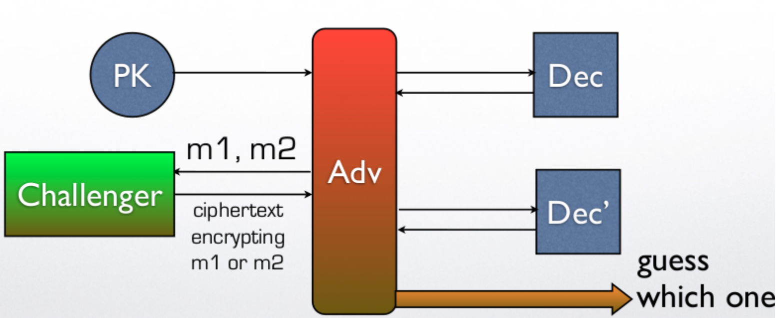
Security models

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



Security models

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



Security models

- Is (schoolbook) RSA IND-CCA2 secure?

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		**Bob**
choose p, q, e		
$n \leftarrow pq$		
$\mathsf{sk} \leftarrow e^{-1} \bmod \Phi(n)$		
$s \leftarrow x^{\mathsf{sk}} \bmod n$		
	$\xrightarrow{x,\, s,\, \mathsf{pk} = (n, e)}$	
		check $x = s^e \bmod n$

RSA signatures

Alice **Bob**

choose p, q, e

$n \leftarrow pq$

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

$s \leftarrow x^{\mathsf{sk}} \bmod n$

$$\xrightarrow{x, s, \mathsf{pk} = (n, e)}$$

check $x = s^e \bmod n$

RSA signatures

- Are (schoolbook) RSA signatures secure?

RSA signatures

- 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 $x = s^e \bmod n$!

RSA signatures

- 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 $x = s^e \bmod n$!
- Solution
  - Random padding

References

- 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
- RL Rivest, A Shamir, L Adleman, ''A Method for Obtaining Digital Signatures and Public-Key Cryptosystems'', in Communications of the ACM v 21 no 2 (Feb 1978) pp 120–126