University of Salford MANCHESTER

SCHOOL OF SCIENCE, ENGINEERING & ENVIRONMENT

Privacy and Network Security

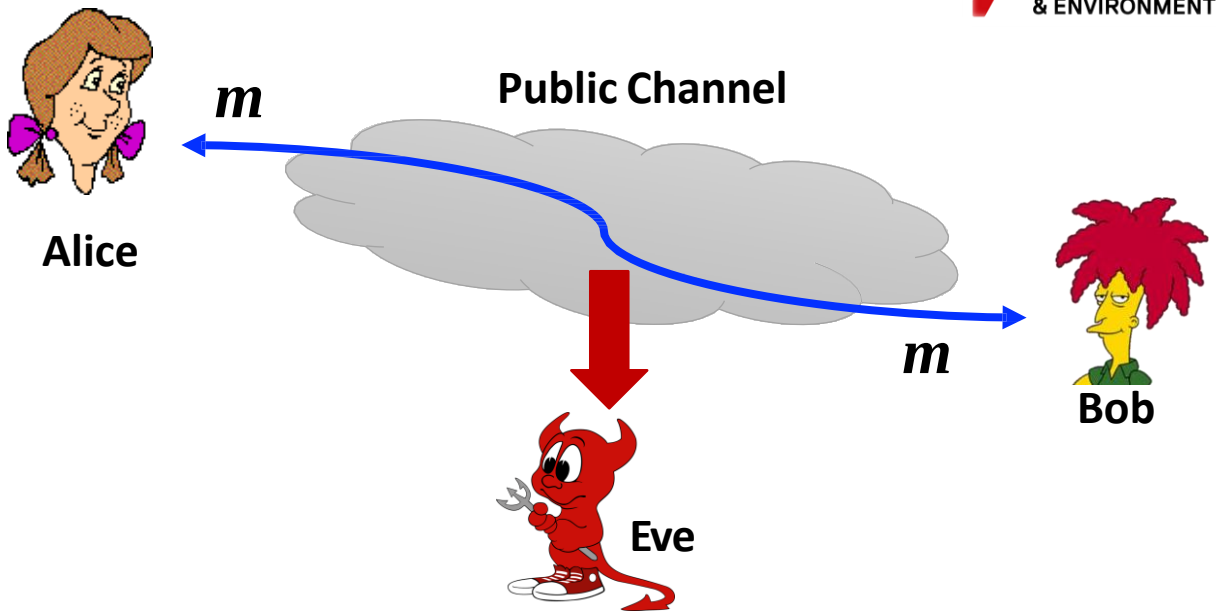
Lecture 5 – Applied Cryptography Part 2

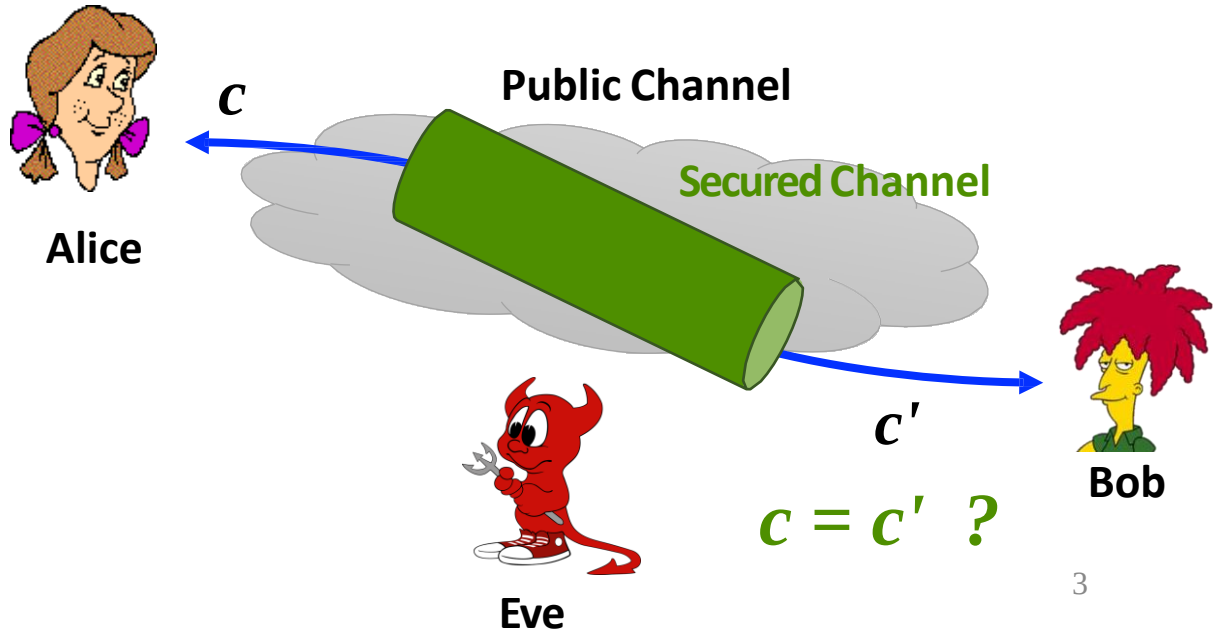
Learning Objectives

- After this lecture you will be able to:
  - Discuss asymmetric encryption algorithms
  - Discuss the applications of asymmetric cryptography in key exchange
  - Discuss digital signature

Chapter 9, 10, 13, 14, from Cryptography and Network Security: Principles and Practice

- **Actors:** Alice, Bob, Eve
- Elements of the **Cryptosystem**:
  - Plaintext: *m*
  - Ciphertext: *c*
  - Set of keys: *K*
    - *Encryption key:* k_e
    - *Decryption key :* k_d
  - Encryption function: *E*
    - $E(m , k_e) = c$
  - Decryption function: *D*
    - $D(c , k_d) = m$





Asymmetric Cryptography

- First publicly proposed by Diffie and Hellman in 1976
- Based on mathematical functions rather than on simple operations on bit patterns

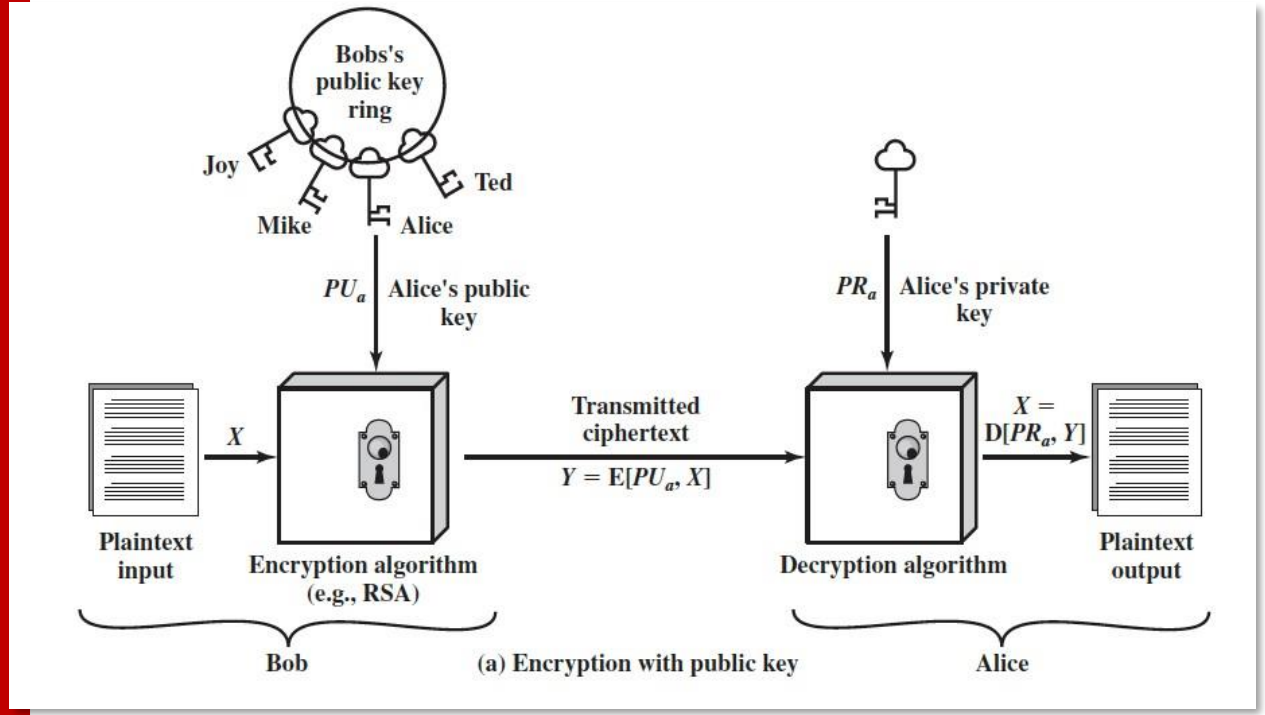
Public-Key Cryptography

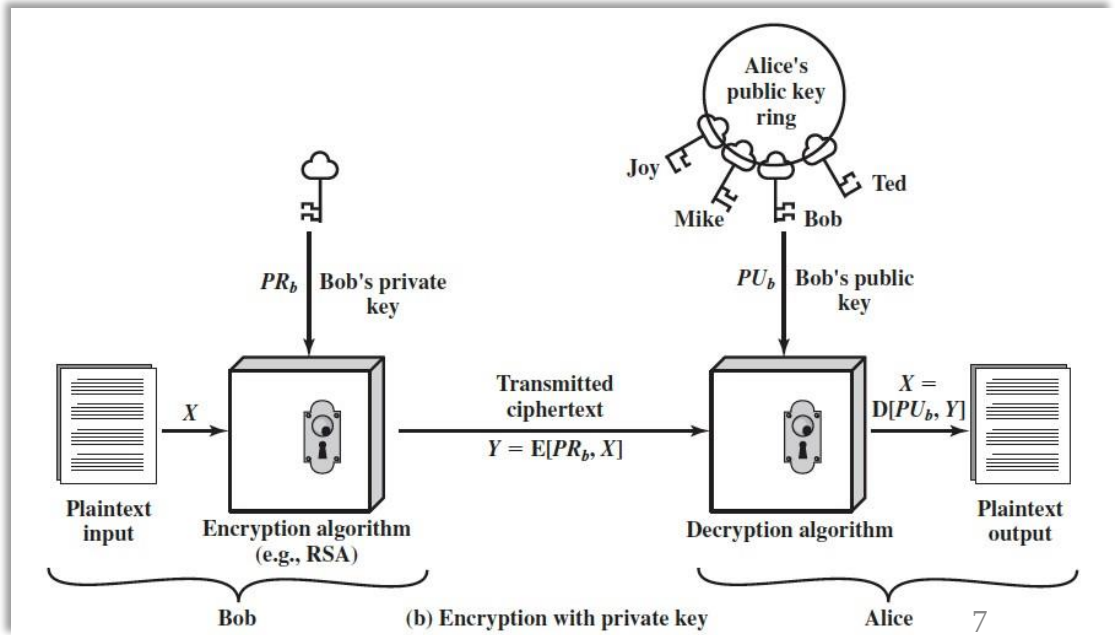
- **Ingredients:**
  - **Plaintext**: This is the readable message or data that is fed into the algorithm as input.
  - **Encryption algorithm:** The encryption algorithm performs various transformations on plaintext.
  - **Public and private key:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input.
  - **Ciphertext**: 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.
  - **Decryption algorithm**: This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

Public-Key Cryptography

- **Steps:**
  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 another accessible file. This is the public key. The companion key is kept private. Each user maintains a collection of public keys obtained from others.
  3. If Bob wishes to send a private message to Alice, Bob encrypts the message using Alice's public key.
  4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

Public-Key Cryptography





Application of Public-Key Cryptosystem

1. Encryption/Decryption:
   - Sender encrypts a message with the recipient's public key.

2. 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.

3. Digital Signature:
   - The sender "signs" a message with its 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.

Requirements for Public-Key Cryptography

1. It is computationally easy for a party B to generate a pair (public key PU_b, private key PR_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 ciphertext:

$$C = \mathrm{E}(PU_b, M)$$

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

$$M = \mathrm{D}(PR_b, C) = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

4. It is computationally infeasible for an opponent, knowing the public key, PU_b, to determine the private key, PR_b.
5. It is computationally infeasible for an opponent, knowing the public key, PU_b, and a ciphertext, C, to recover the original message, M.
6. Either of the two related keys can be used for encryption, with the other used for decryption.

$$M = \mathrm{D}[PU_b, \mathrm{E}(PR_b, M)] = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

How Secure is a System?

- How secure is a system against cryptanalysis when the enemy has **unlimited time and manpower available** for the analysis of intercepted cryptograms?

- **Computationally Secure**
- The cost of breaking the cipher exceeds the value of the encrypted information.
- The time required to break the cipher exceeds the useful lifetime of the information.

Computationally?

- To satisfy the aforementioned requirements, it is required to have a one-way function: it is easy to calculate this function, but it is hard to calculate its inverse.

How hard it is to solve a problem?

- Computational complexity
- Hard problem assumptions

Commonly used hard problems:

- Large integer factorization -> RSA cryptosystem
- (computational/decisional) Diffie-Hellman problem -> ElGamal Cryptosystem
- Discrete logarithm problem
- Elliptic curve D-H problem -> elliptic curve cryptography

RSA Public-Key Encryption

- Proposed by Ron Rivest, Adi Shamir, and Len Adleman at MIT in 1977
- RSA is a block cipher in which the plaintext and ciphertext are integers between 0 and n-1 for some n.
- Encryption and decryption:

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

$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

- Sender and Receiver must know values of n and e
- Public key = $\{e, n\}$, Private key = $\{d, n\}$

RSA Requirements

1. It is possible to find values of e, d, n such that $M^{ed} \bmod n = M$ 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.

This is only possible for large values of *e* and *n*

RSA Algorithm

Key Generation	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$ Extended GCD
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod n$

Decryption	
Ciphertext:	C
Plaintext:	$M = C^d \pmod n$

- Alice publishes her public key, and Bob wants to send message M to Alice
- Bob calculates C and sends to Alice

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

- Alice decrypts the message:

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

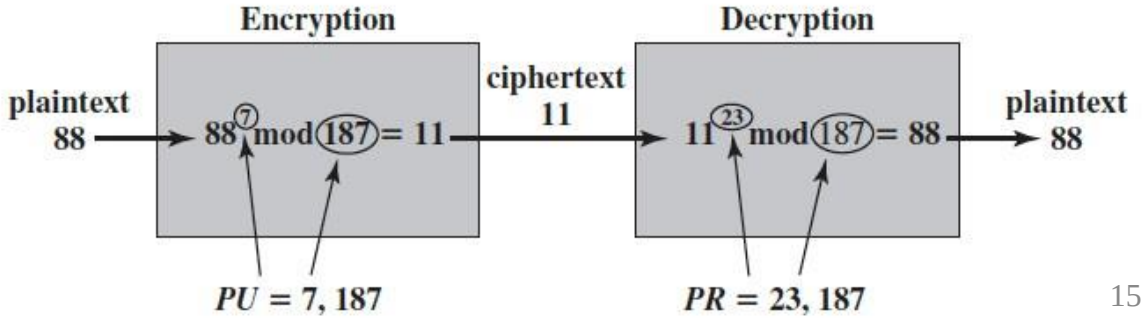
Q1: Prove correctness! Hint: use Euler's theorem or Fermat's little theorem
Q2: Is RSA IND-CPA?

RSA Example

1. Select two prime numbers, p =17 and q =11
2. Calculate $n = pq = 17 \times 11 = 187$
3. Calculate $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e=7$.
5. Determine d such that de mod 160=1 and d < 160.The correct value is $d=23$, because 23 × 7= 161 = (1 × 160) +|1.

Q2: Try an encryption and decryption:
With the keys above, encrypt m = 14

$$c = m^e \bmod n, m = c^d \bmod n$$



Attacks to RSA

- **Brute-force approach**:
  - Try all possible private keys. Thus, the larger the number of bits in *e* and *d*, the more secure the algorithm
  - However, because the calculations involved (both in key generation and in encryption/decryption) are complex, the larger the size of the key, the slower the system will run.

- **Cryptanalysis of *n*:**
  - 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

Discussions

- Is public key encryption more secure than symmetric key encryption?
- How do you decide which type of encryption to use?

Key Exchange

- Recap of symmetric crypto:
  - What is the fundamental issue?

- What security requirements are needed, to ensure two users can exchange a secret key securely that then can be used for subsequent encryption of messages

Key exchange

- Symmetric key distribution using symmetric encryption
- Symmetric key distribution using asymmetric encryption (key encapsulation)
- Diffie-Hellman key exchange

Diffie-Hellman Problem, Discrete Logarithm

DH problem: Given an element g and the values of g^x and g^y, what is the value of g^{xy}?

The difficulty of answering this question is equivalent to finding discrete logarithms, e.g., given g and g^x , find x.

The problem is defined over a group of a prime order p, with modular arithmetic.

Discrete Logarithm

g is a generator of the group, or, a primitive root of the prime number p.

The powers of g modulo p generate all the integers from 1 to p.

For any integer b and a primitive root g of prime number p, we can find a unique exponent i such that

$$b \equiv g^i \ (mod\ p)$$

where $0 \leq i \leq (p-1)$

Diffie-Hellman Algorithm

We have Alice and Bob

we need to set up some public parameters:

p: a prime number and

α : an integer parameter (a primitive root of p).

Given these two numbers, **Alice sets its keys:**

a = private key = K_{prA} in {2,3, . . . *p*-2}

A = public key = $K_{pubA} = \alpha^a mod\ p$

Alice sends its public key portion to Bob

Bob does the same: first select his private key then computes and sends its public key portion to Alice!

b = private key = K_{prB} in {2,3, . . . *p*-2}

B = public key = $K_{pubB} = \alpha^b mod\ p$

Bob sends its public key portion to Alice

Diffie-Hellman Algorithm

Then both Alice and Bob do the same thing to compute their (equivalent) private key:

Bob → $A^b \bmod p$ $(AlicePublic^{BobPrivate} \bmod p)$

Alice → $B^a \bmod p$ $(BobPublic^{AlicePrivate} \bmod p)$

Surprising and awesome fact:

$A^b \bmod p = B^a \bmod p = K_{AB}$

Private key!!!

Symmetric in this case

This was a breakthrough, as lead to public key crypto!

Now Alice encrypts a message x using e.g. AES and her public key K_{AB} :

$y = AES(K_{AB}, x)$ and sends y to Bob

$x = AES^{-1}(K_{AB}, y)$ Bob reconstructs x using his private key K_{AB} (the same)

Diffie-Helman Example

Let's pick our prime p = 353 and a "primitive root" of 353, in this case α = 3.

Alice and Bob select secret keys a = 97 and b = 233, respectively. Each computes its public key portion:

Alice computes $\alpha^a \bmod p = 3^{97} \bmod 353 = 40$.

Bob computes $\alpha^b \bmod p = 3^{233} \bmod 353 = 248$.

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

Alice computes $K_{AB} = B^a \bmod 353 = 248^{97} \bmod 353 = 160$.

Bob computes $K_{AB} = A^b \bmod 353 = 40^{233} \bmod 353 = 160$.

Diffie-Helman Example

The attacker Dennis has available the following information:

$q = 353;$

$\alpha = 3;$

$Y_A = 40$ (Alice public key portion)

$Y_B = 248$ (Bob public key portion)

Why can't Dennis decrypt the secret key and so the msg?

Diffie-Helman Example

In this simple example, Dennis could use brute force to determine the secret key 160. In particular, he can determine the common key by discovering a solution to the equation:

$3^a \bmod 353 = 40$ or the equation

$3^b \bmod 353 = 248.$

The brute-force approach is to calculate powers of 3 modulo 353, stopping when the result equals either 40 or 248.

The desired answer is reached with the exponent value of 97, which provides $3^{97} \bmod 353 = 40$.

With larger numbers, the problem becomes impractical.

ElGamal Cryptosystem

- Has a temporary key for each encryption
- Temporary key is established with Diffie-Hellman key exchange
- Not secure under chosen-ciphertext attack

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{XA} \bmod q$
Public key	$PU = \{q, \alpha, Y_A\}$
Private key	X_A

Both of the public key and the private key belong to Alice
Public key is publicly known, private key is only known to Alice

ElGamal, continued

Encryption by Bob with Alice's Public Key

Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

K is established as the temporary key, using Diffie-Hellman key exchange

Decryption by Alice with Alice's Private Key

Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{XA} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

Achieves confidentiality of M, sent by Bob to Alice => only Bob and Alice know the content

Elliptic curve cryptography

- Uses the Elliptic Curve Discrete Logarithm (ECDL) or Elliptic Curve Diffie-Hellman (ECDH) problem
- Elliptic Curve Arithmetic: a set of operations to compute points on elliptic curves

Analog to Discrete Logarithm in group

- Pick a group
- Pick a generator of the group, g
- DL: it is easy to calculate g^a, but it is hard to find a with g^a and g

- Pick a curve
- Pick a base point $G = (x, y)$ on the curve, of order n
- ECDL: It is easy to calculate $a \cdot G$, but it is hard to find a from $a \cdot G$ and G.

Security of ECC

- Depends on …
  - The curve
  - Key length

In practice, ECC uses a shorter key length to achieve the same level of security provided by RSA or ElGamal

Digital Signature

- Motivation:
  - Man-in-the-middle attacks

Motivation

- Motivation:
  - Denial or dispute

- Dennis orders a pink car from the car salesmen Bob
- Dennis seeing the pink car states that he has never ordered it:
- How can Bob prove towards a judge that Dennis has ordered a pink car? (And that he did not fabricate the order himself)

  ⇒ Symmetric cryptography fails because both Dennis and Bob can be malicious

Idea of Standard Signature

Goal: Signature-like function for the electronic world

Conventional (paper) signature:

proof of authenticity of a sender

Let's try the same digitally:

00001101010 PDF Document

==========

11101001011 Append digital signature

Would this work as a digital signature?

Digital Signature Basic Idea

Let's use cryptography for that:

You can sign only if you have the key!

How do we sign a document x?

With a signature function:

We plug x into the signature(x) = s

- Sign an element from the message space x:
  $sign_{Kpri}(x) = s$ (signature s)

- Send both doc x and signature y over the channel
  Alice ← (x,s) ← Bob

- Now Alice (receiver) applies a ***Verification function***
  $ver_{Kpub}(x,s)$ = ???

Example of DS - RSA

Alice Bob

K_{prA}, K_{pubA} ----> K_{pubA} ---->

(Alice sends K_{pubA})

$s = sign_{KprA}(x)$ ---> (x,s) ---->

(Alice signs with private key
and sends both message and signature to Bob)

In RSA D-S: the big question is:
what is the function sign()?
what is the function ver()?

$ver_{KpubA}(x,s)$ = True/false?

Bob verifies with the public key

To generate the private and public key:

Use the same key generation as RSA encryption.

To generate the signature: Sign

- "encrypt" the message x with the private key

$$s = sig_{Kpriv}(x) = x^d \bmod n$$

- Append s to message x

To verify the signature: Verify

- "decrypt" the signature with the public key:

$$x' = ver_{Kpub}(s) = s^e \bmod n$$

- If x=x', the signature is valid

RSA Digital Signature Protocol

Alice

Bob

K_{pub} ←

$K_{pr} = d$
$K_{pub} = (n, e)$

Compute signature:
$s = sig_{k_{pr}}(x) \equiv x^d \bmod n$

(x,s) ←

Verify signature:
$x' \equiv s^e \bmod n$
If $x' \equiv x \bmod n$ → valid signature
If $x' \not\equiv x \bmod n$ → invalid signature

In practice, signature is not generated by encrypting the whole message, because it is computationally expensive to encrypt a large block of data. In application, digital signature is generated by encrypting a 'digest' of the whole message, while the 'digest' is shorter, and is highly likely to be unique to each input message. The 'digest' is typically generated with cryptographic hash functions.

Security services provided by digital signature

Which ones apply here?

1. **Confidentiality**: info is kept secret for all but the authorized part
2. **Message Authentication:** Sender of a message is authentic
3. **Message Integrity:** message has not been modified during TX
4. **Non-repudiation:** Sender of a message cannot deny its creation

References

- William Stallings, “Network Security Essentials: Applications and Standards”
- William Stallings, “Cryptography and Network Security: Principles and Practice”
- Workshop code and lab manual
- https://github.com/meng-ya-li/PNS2023crypto