COMP3028
Lecture 7 - Cryptography IV

Public key Cryptography &

Key Management

Private-Key Cryptography

- traditional **private/secret/single key** cryptography uses **one** key
- shared by both sender and receiver
- if this key is disclosed communications are compromised
- also is **symmetric**, parties are equal
- hence does not protect sender from receiver forging a message & claiming is sent by sender

Public-Key Cryptography

- probably most significant advance in the 3000 year history of cryptography
- uses **two** keys – a public & a private key
- **asymmetric** since parties are **not** equal
- uses clever application of number theoretic concepts to function
- complements **rather than** replaces private key crypto

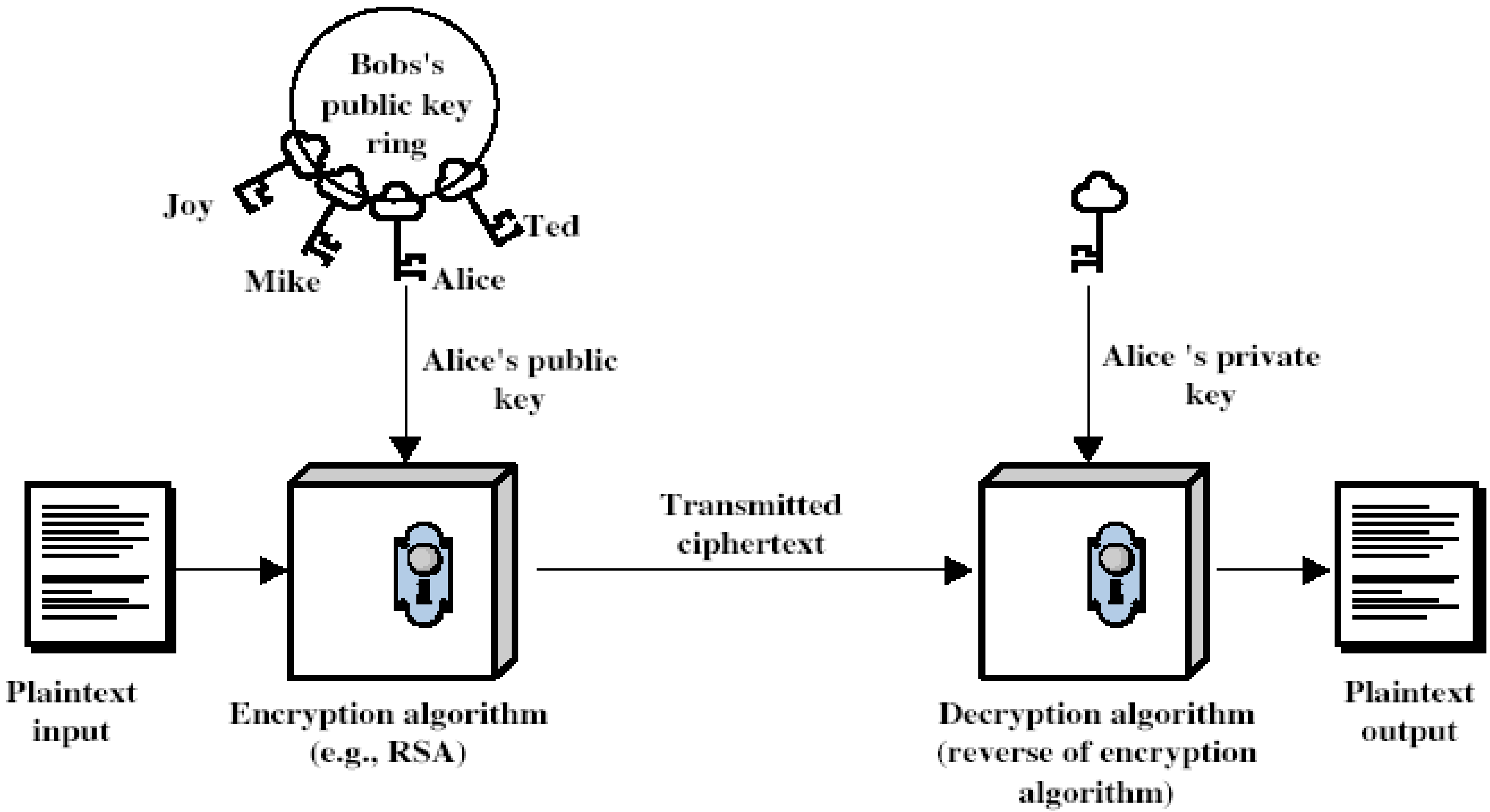
Why Public-key Cryptography?

- developed to address two key issues:
  - key distribution – how to have secure communications in general without having to trust a KDC with your key
  - digital signatures – how to verify a message comes intact from the claimed sender

Public-Key Cryptography

- **public-key/two-key/asymmetric** cryptography involves the use of **two** keys:
  - a **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
  - a **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**
- is **asymmetric** because
  - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

Public-Key Cryptography



Public-Key Characteristics

- Public-Key algorithms rely on two keys with the characteristics that it is:
  - computationally infeasible to find decryption key knowing only algorithm & encryption key
  - computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - either of the two related keys can be used for encryption, with the other used for decryption (for some algorithms)

Public-Key Applications

- can classify uses into 3 categories:
  - **encryption/decryption** (provide secrecy)
  - **digital signatures** (provide authentication)
  - **key exchange** (of session keys)
- some algorithms are suitable for all uses, others are specific to one

Security of Public Key Schemes

- like private key schemes brute force **exhaustive search** attack is always theoretically possible
- but keys used are too large (> 512bits)
- security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalyse) problems
- more generally the **hard** problem is known, its just made too hard to do in practise
- requires the use of **very large numbers**
- hence is **slow** compared to private key schemes

RSA

- by Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- based on exponentiation in a finite (Galois) field over integers modulo a prime
  - nb. exponentiation takes $O((\log n)^3)$ operations (easy)
- uses large integers (eg. 1024 bits)
- security due to cost of factoring large numbers
  - nb. factorization takes $O(e^{\log n \log \log n})$ operations (hard)

RSA Key Setup

- each user generates a public/private key pair by:
- selecting two large primes at random: p, q
- computing their system modulus $n = p.q$
  - note $\varnothing(n) = (p-1)(q-1)$
- selecting at random the encryption key e
  - where $1 < e < \varnothing(n)$, $GCD(e, \varnothing(n)) = 1$
- solve following equation to find decryption key d
  - $e.d = 1 \mod \varnothing(n)$ and $0 \le d \le n$
- publish their public encryption key: $KU = \{e,n\}$
- keep secret private decryption key: $KR = \{d,p,q\}$

RSA Use

- to encrypt a message M the sender:
  - obtains **public key** of recipient $KU=\{e,n\}$
  - computes: $C = M^e \bmod n$, where $0 \le M < n$
- to decrypt the ciphertext C the recipient:
  - uses their private key $KR=\{d,p,q\}$, $n = p.q$
  - computes: $M = C^d \bmod n$
- note that the message M must be smaller than the modulus n (block if needed)

RSA Example – Key Setup

1. Select primes: $p = 17$ & $q = 11$
2. Compute $n = pq = 17 \times 11 = 187$
3. Compute $ø(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select e: $1 < e < 160$, $GCD(e,160) = 1$; choose $e = 7$
5. Determine d: $de = 1 \bmod 160$ and $d < 160$
   Value is $d = 23$ since $23 \times 7=161= 10 \times 160+1$
6. Publish public key $KU=\{7,187\}$
7. Keep secret private key $KR=\{23,17,11\}$

RSA Example – En/Decryption

- sample RSA encryption/decryption is:
- given message $M = 88$ (nb. $88 < 187$)
- encryption:

  $C = 88^7 \bmod 187 = 11$
- decryption:

  $M = 11^{23} \bmod 187 = 88$

Exponentiation

- can use the Square and Multiply Algorithm
- a fast, efficient algorithm for exponentiation
- concept is based on repeatedly squaring base
- and multiplying in the ones that are needed to compute the result
- look at binary representation of exponent
- only takes $O(\log_2 n)$ multiples for number n
  - eg. $7^5 = 7^4.7^1 = 3.7 = 10 \mod 11$
  - eg. $3^{129} = 3^{128}.3^1 = 5.3 = 4 \mod 11$

Square and Multiply Algorithm

- Compute $a^e \bmod n$:
  - Convert e to binary: $b_{i-1}...b_1b_0$

    z=1

    for k = i-1 downto 0

        do $z = z^2 \bmod n$

        if $b_k = 1$

          then $z = (z^2 \times a) \bmod n$

    return z

S & M Algorithm - Example

$11^{23} \bmod 187 = 88$

i	b	z mod n
4	1	$1^2 \times 11 = 11$
3	0	$11^2 = 121$
2	1	$121^2 \times 11 = 44$
1	1	$44^2 \times 11 = 165$
0	1	$165^2 \times 11 = 88$

RSA Key Generation

- users of RSA must:
  - determine two primes at random: p, q
  - select either e or d and compute the other
- primes p, q must not be easily derived from modulus $n = p.q$
  - means must be sufficiently large
  - typically guess and use probabilistic test
- exponents e, d are inverses, so use inverse algorithm to compute the other

RSA Security

- possible approaches to attacking RSA:
  - brute force key search (infeasible given size of numbers)
  - mathematical attacks (based on difficulty of computing $ø(n)$, by factoring modulus n)
  - timing attacks (on running of decryption)
  - chosen ciphertext attacks (given properties of RSA)

Factoring Problem

- mathematical approach takes 3 forms:
  - factor n = p.q, hence find ø(n) and then d
  - determine ø(n) directly and find d
  - find d directly
- currently believe all equivalent to factoring
  - have seen slow improvements over the years
    - as of Feb-20 best is 250 decimal digits (829 bit) with GNFS
  - currently assume 1024+ bit RSA is secure
    - ensure p, q of similar size and matching other constraints

Timing Attacks

- developed by Paul Kocher in mid-1990's
- exploit timing variations in operations
  - eg. multiplying by small vs large number
  - or IF's varying which instructions executed
- infer operand size based on time taken
- RSA exploits time taken in exponentiation
- countermeasures
  - use constant exponentiation time
  - add random delays
  - blind values used in calculations

Key Management

- public-key encryption helps address key distribution problems
- have two aspects of this:
  - distribution of public keys
  - use of public-key encryption to distribute secret keys

Public-Key Distribution of Secret Keys

- use previous methods to obtain public-key
- can use for secrecy or authentication
- but public-key algorithms are slow
- so usually want to use private-key encryption to protect message contents
- hence need a session key
- have several alternatives for negotiating a suitable session

Diffie-Hellman Key Exchange

- first public-key type scheme proposed
- by Diffie & Hellman in 1976 along with the exposition of public key concepts
  - note: now know that James Ellis (UK CESG) secretly proposed the concept in 1970
- is a practical method for public exchange of a secret key
- used in a number of commercial products

Diffie-Hellman Key Exchange

- a public-key distribution scheme
  - cannot be used to exchange an arbitrary message
  - rather it can establish a common key
  - known only to the two participants
- value of key depends on the participants (and their private and public key information)
- based on 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
  - α, primitive root mod q
- each user (eg. A) generates their key
  - chooses a secret key (number): $x_A < q$
  - compute their public key: $y_A = \alpha^{x_A} \bmod q$
- each user makes public that key y_A

Diffie-Hellman Key Exchange

- shared session key for users A & B is K_{AB}:

  $K_{AB} = \alpha^{x_A . x_B} \bmod q$

  $= y_A^{x_B} \bmod q$ (which B can compute)

  $= y_B^{x_A} \bmod q$ (which A can compute)
- K_{AB} is used as session key in private-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 needs an x, must solve discrete log

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:
  - $y_A=3^{97} \bmod 353 = 40$ (Alice)
  - $y_B=3^{233} \bmod 353 = 248$ (Bob)
- compute shared session key as:

$K_{AB}= y_B^{x_A} \bmod 353 = 248^{97} = 160$ (Alice)

$K_{AB}= y_A^{x_B} \bmod 353 = 40^{233} = 160$ (Bob)

Man in the Middle Attack

- Charles chooses a secret key, $x_c = 3$
- He intercepts $q=353$, $\alpha=3$, $y_A=40$, $y_B=248$
- He computes public key $y_c = 3^3 \bmod 353 = 27$ and sends it to Alice and Bob
- He then computes $K_{AC} = 40^3 \bmod 353 = 107$ and $K_{BC} = 248^3 \bmod 353 = 215$
- Not knowing the public key is from Charles:
  - Alice computes $K = 27^{97} \bmod 353 = 107$
  - Bob computes $K = 27^{233} \bmod 353 = 215$

El Gamal Public-Key Encryption Scheme

- a variant of the Diffie-Hellman key distribution scheme
- allowing secure exchange of messages
- published in 1985 by ElGamal:
- *T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", IEEE Trans. Information Theory, vol IT-31(4), pp469-472, July 1985.*
- like Diffie-Hellman its security depends on the difficulty of computing discrete logarithms
- major disadvantage is that it doubles the size of info sent

El Gamal Setup

- Key Generation
  - select a large prime p, and α, a primitive element mod p
  - recipient Bob chooses a secret number x_B ($0 < x_B < p$) and computes $y_B = \alpha^{x_B} \bmod p$

El Gamal Encryption

- to encrypt a message M into ciphertext C:
- sender selects a random number n (sender's private key), $0 < n < p$
- and computes the message key, $K = (y_B)^n \bmod p$
- then computes the ciphertext pair: $C = \{C_1, C_2\}$, where

  $C_1 = \alpha^n \pmod p$

  $C_2 = K.M \pmod p$
- this is then sent to the recipient
- note that K should be destroyed after use and never knowingly used again

El Gamal Decryption

- to decrypt the message:
- extracts the message key K:

  $K = (C_1)^{x_B} \pmod{p}$
- extracts M by solving for M in the following equation:

  $M = C_2.K^{-1} \pmod{p}$

El Gamal Example

- given prime p=97 with primitive root α=5
- recipient Bob chooses secret key, x_B=58 & computes & publishes his public key, $y_B=5^{58}=44 \bmod 97$
- Alice wishes to send the message M=3 to Bob
- she obtains Bob's public key, y_B=44
- she chooses random n=36 (her private key) and computes the message key:
  $K=44^{36}=75 \bmod 97$
- she then computes the ciphertext pair:
  $C_1 = 5^{36} = 50 \bmod 97$
  $C_2 = 75.3 \bmod 97 = 31 \bmod 97$
  and send the ciphertext {50,31} to Bob
- Bob recovers the message key $K=50^{58}=75 \bmod 97$
- Bob computes the inverse $K^{-1} = 22 \bmod 97$
- Bob recovers the message $M = 31.22 = 3 \bmod 97$

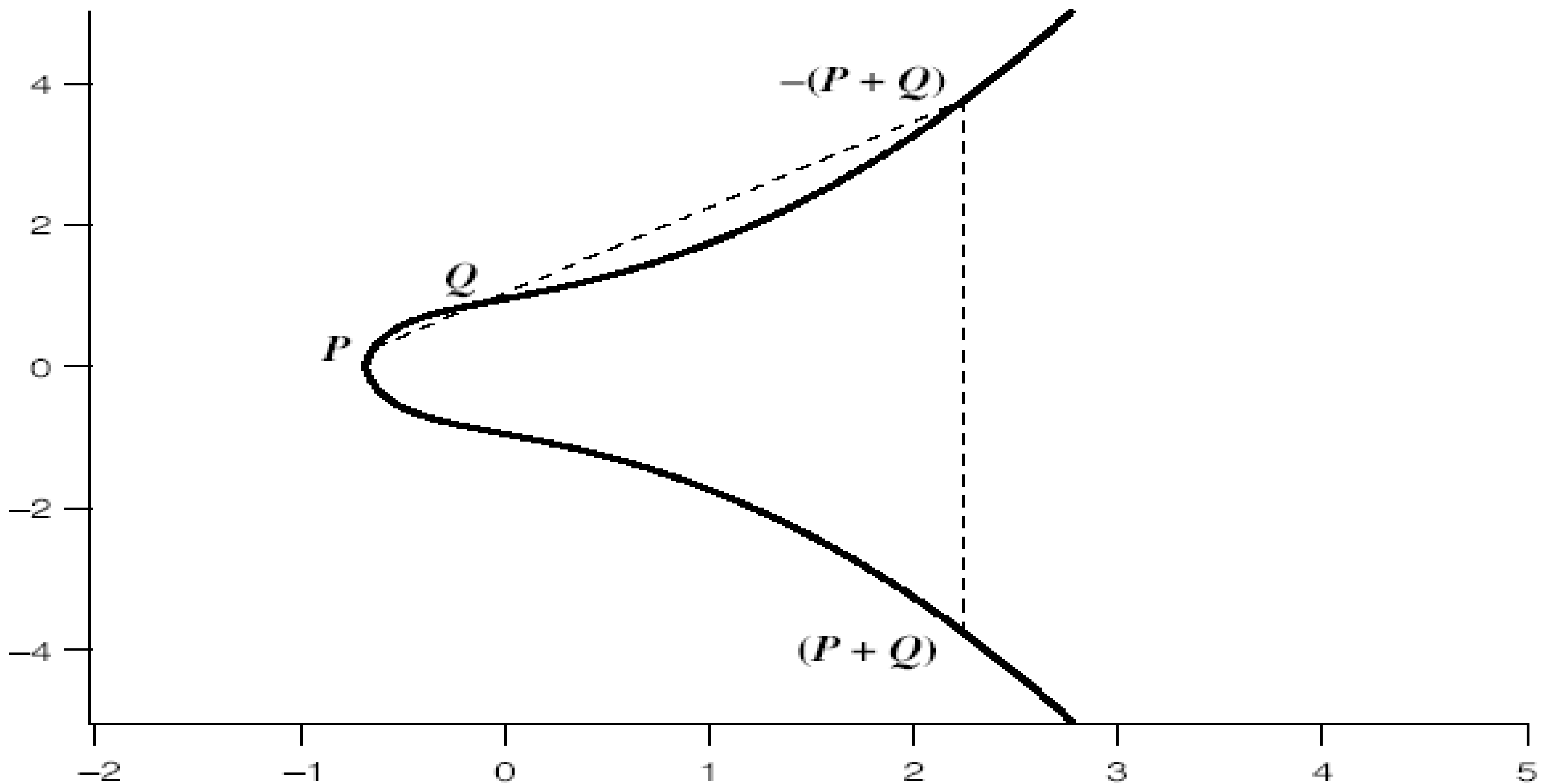
Elliptic Curve Cryptography

- majority of public-key crypto (RSA, D-H) use either integer or polynomial arithmetic with very large numbers/polynomials
- imposes a significant load in storing and processing keys and messages
- an alternative is to use elliptic curves
- offers same security with smaller bit sizes

Real Elliptic Curves

- an elliptic curve is defined by an equation in two variables x & y, with coefficients
- consider a cubic elliptic curve of form
  - $y^2 = x^3 + ax + b$
  - where x,y,a,b are all real numbers
  - also define point O (point at infinity)
- have addition operation for elliptic curve
  - geometrically sum of Q+R is reflection of intersection R

Real Elliptic Curve Example



(b) $y^2 = x^3 + x + 1$

Finite Elliptic Curves

- Elliptic curve cryptography uses curves whose variables & coefficients are finite
- have two families commonly used:
  - prime curves $E_p(a,b)$ defined over Z_p
    - use integers modulo a prime
    - best in software
  - binary curves $E_{2m}(a,b)$ defined over $GF(2^n)$
    - use polynomials with binary coefficients
    - best in hardware

Elliptic Curve Cryptography

- ECC addition is analog of modulo multiply
- ECC repeated addition is analog of modulo exponentiation
- need "hard" problem equiv to discrete log
  - Q=kP, where Q,P belong to a prime curve
  - is "easy" to compute Q given k,P
  - but "hard" to find k given Q,P
  - known as the elliptic curve logarithm problem
- Certicom example: $E_{23}(9,17)$
  - https://www.certicom.com/content/certicom/en/52-the-elliptic-curve-discrete-logarithm-problem.html

ECC Diffie-Hellman

- can do key exchange analogous to D-H
- users select a suitable curve $E_p(a,b)$
- select base point $G=(x_G,y_G)$ with large order n s.t. $nG = O$
- A & B select private keys $k_A<n$, $k_B<n$
- compute public keys: $P_A=k_AG$, $P_B=k_BG$
- compute shared key: $K_{AB}=k_AP_B=k_BP_A$
  - same since $K_{AB}=k_Ak_BG$

ECC Encryption/Decryption

- several alternatives, will consider simplest
- must first encode any message M as a point on the elliptic curve P_m
- select suitable curve & point G as in D-H
- each user chooses private key $k_A, k_B < n$
- and computes public key $P_A = k_A G$, $P_B = k_B G$
- to encrypt P_m : $C_m = \{k_A G, P_m + k_A P_B\} = \{C_1, C_2\}$
- decrypt C_m compute: $C_2 - k_B C_1 = P_m$

ECC Example

- Consider $E_{751}(-1,188)$ and $G = (0,376)$
- E is of order $n = 727$ s.t. $nG = O$
- Alice wishes to send the message $P_m = (562,201)$ to Bob
- she obtains Bob's public key $P_B = (201,5)$
- she chooses random k=386
- she then computes the ciphertext $C_m = \{kG, P_m + kP_B\}$ where $kG = 386(0,376)$ and $P_m + kP_B = (562,201) + 386(201,5) = (385,328)$
- and sends the ciphertext $\{(676,558),(385,328)\}$ to Bob

ECC Security

- relies on elliptic curve logarithm problem
- fastest method is "Pollard rho method"
- compared to factoring, can use much smaller key sizes than with RSA etc
- for equivalent key lengths computations are roughly equivalent
- hence for similar security ECC offers significant computational advantages

Comparable Key Sizes for Equivalent Security

Symmetric scheme (key size in bits)	ECC-based scheme (size of *n* in bits)	RSA/DSA (modulus size in bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

Summary

- Public-key cryptosystems
- RSA
- Diffie-Hellman key exchange
- El Gamal Scheme
- Elliptic Curve cryptography