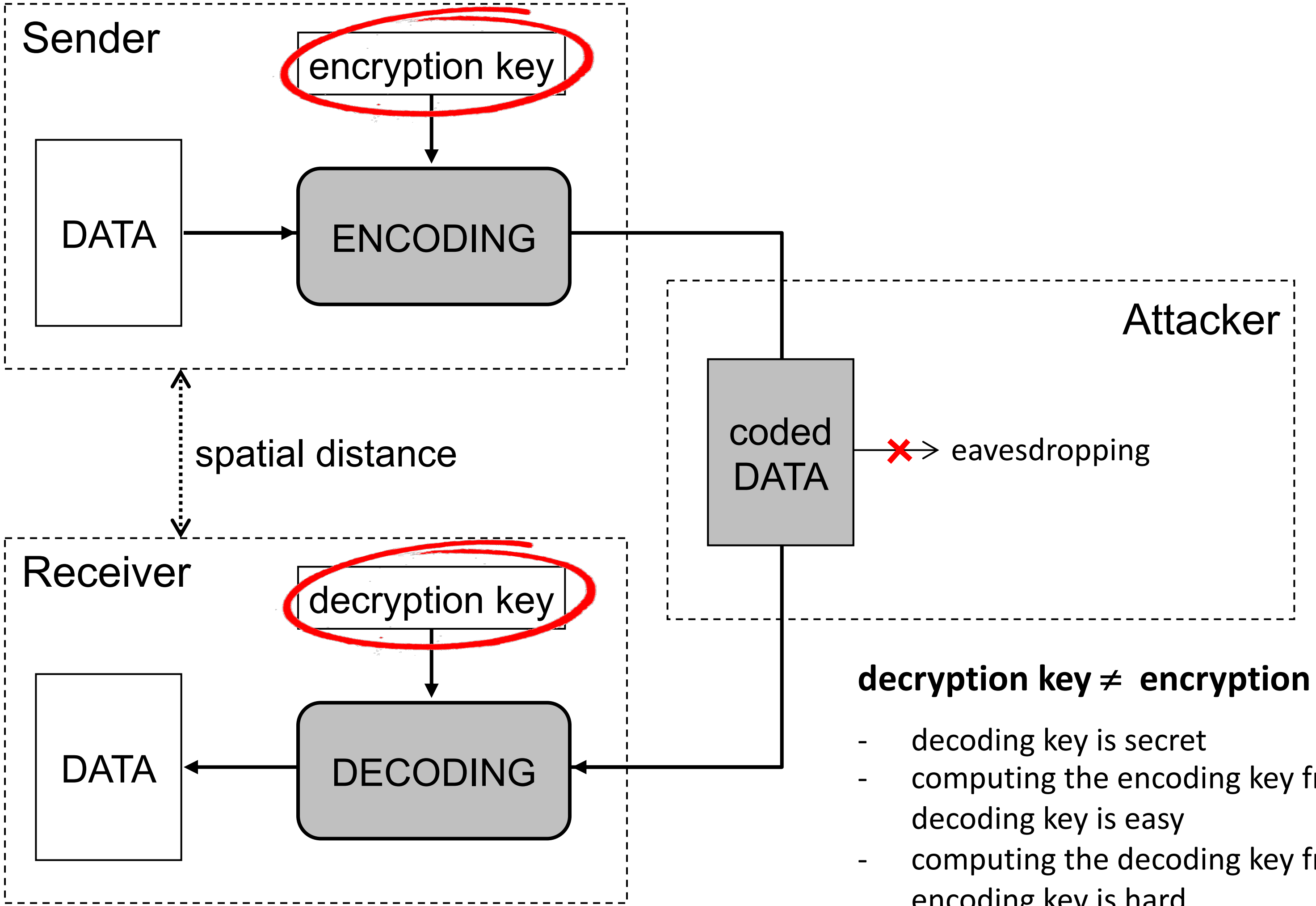
Public Key Cryptography

Levente Buttyán
CrySyS Lab, BME
buttyan@crysys.hu

Model of asymmetric key encryption



decryption key ≠ encryption key

- decoding key is secret
- computing the encoding key from the decoding key is easy
- computing the decoding key from the encoding key is hard
- encoding key can be made **public** (public-key crypto)

Public-key encryption schemes

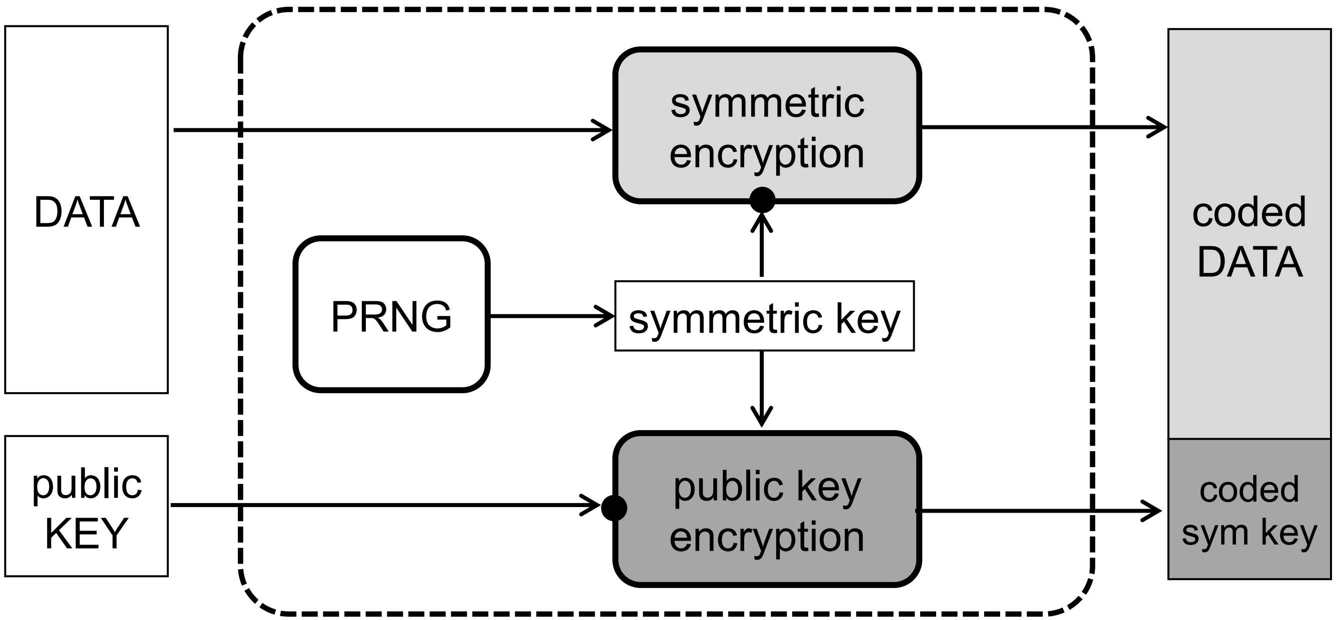
- functions (algorithms) and terminology:
  - key-pair generation function $G( ) = (K^+, K^-)$
    - K^+ – public key
    - K^- – private key
  - encryption function $E(K^+, X) = Y$
    - X – plaintext
    - Y – ciphertext
  - decryption function $D(K^-, Y) = X$

- typically, the plaintext (and the ciphertext) consists of a few hundred bits → operation is similar to symmetric-key block ciphers

- examples: RSA, ElGamal, NTRU

Hybrid encryption (digital envelop)

- public-key encryption schemes use large number arithmetics, and hence, they are several orders of magnitude slower than the best known symmetric key ciphers (on the same platform)
- to overcome this problem, the following hybrid approach is used in practice:



Security of public key crypto schemes

- security is usually related to the difficulty of some problems that are widely believed to be hard to solve (i.e., for which no polynomial time solution exists today), such as
  - factoring:

    given a positive integer N, find its prime factors
  - computing discrete logarithm:

    given a prime p, a generator g of Z_p^*, and an element y in Z_p^*, find the integer x, $0 \leq x \leq p-2$, such that $g^x \bmod p = y$

- sometimes it can even be rigorously proven that breaking the encryption scheme would mean that there exists an efficient solution to the related hard problem (reduction)
  - although widely used practical schemes have no complete proofs

Semantic security

- an adversary should not be able to choose two plaintexts X and X' and later distinguish between the encryptions $E_K(X)$ and $E_K(X')$ of these messages
  - in case of public-key encryption, the adversary can compute $E_K(X)$ and $E_K(X')$ using the public key K and trivially determine that $E_K(X)$ is the encryption of X and $E_K(X')$ is the encryption of X'
  - How about symmetric-key encryption?

- the solution is ***probabilistic encryption***
  - the ciphertext should depend on some random input that is kept secret
  - after decryption, the original plaintext van be recovered unambiguously
  - some public-key encryption schemes are probabilistic by design (e.g., ElGamal)
  - others need pre-formatting of messages which involves the addition of some randomness (e.g., RSA uses PKCS #1 formatting)

RSA

The (textbook) RSA cryptosystem

- key-pair generation algorithm:
  - choose two large primes p and q (easy)
  - $n = pq$, $\phi(n) = (p-1)(q-1)$ (easy)
  - choose e, such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$ (easy)
  - compute the inverse d of e mod $\phi(n)$, i.e., $ed \bmod \phi(n) = 1$ (easy if p and q are known)
  - output **public key: (e, n)** (made public after key-pair generation)
  - output **private key: d (and p, q)** (kept secret after key-pair generation)

- encryption algorithm:
  - represent the plaintext message as an integer $m \in [0, n-1]$
  - compute the ciphertext $\mathbf{c = m^e \bmod n}$

- decryption algorithm:
  - compute the plaintext from the ciphertext c as $\mathbf{m = c^d \bmod n}$
  - this works, because $c^d \bmod n = m^{ed} \bmod n = m^{k\phi(n)+1} \bmod n = m \bmod n = m$

Proof of RSA decryption

- $c^d \bmod n = m^{ed} \bmod n = m^{k\,\phi(n) + 1} \bmod n = m\, m^{k(p-1)(q-1)} \bmod n$
- since $m < n$, it is enough to prove that $m\, m^{k(p-1)(q-1)} \equiv m \pmod n$
- Fermat theorem
  - if r is a prime and $\gcd(a, r) = 1$, then $a^{r-1} \equiv 1 \pmod r$
- if $\gcd(m, p) = 1$
  - $m^{p-1} \equiv 1 \pmod p$
  - $m\, m^{k(p-1)(q-1)} \equiv m \pmod p$
- if $\gcd(m, p) = p$
  - $p \mid m$
  - $m\, m^{k(p-1)(q-1)} \equiv m \equiv 0 \pmod p$
- for all m, $m\, m^{k(p-1)(q-1)} \equiv m \pmod p$
- similarly, for all m, $m\, m^{k(p-1)(q-1)} \equiv m \pmod q$
- $p, q \mid m\, m^{k(p-1)(q-1)} - m$ --» $pq \mid m\, m^{k(p-1)(q-1)} - m$
- $m\, m^{k(p-1)(q-1)} \equiv m \pmod{pq}$

Security of RSA

- factoring integers is believed to be a hard problem
  - given a composit integer n, find its prime factors
  - true complexity is unknown
  - it is believed that no polinomial time algortihm exists to solve it

- computing d from (e, n) is equivalent to factoring n

- computing m from c and (e,n) (known as the RSA problem) may not be equivalent to factoring n
  - if the factors p and q of n are known, then one can easily compute d, and using d, one can also compute m from c
  - we don't know if one could factor n, given that he can efficiently compute m from c and (e,n)
  - nevertheless, the RSA problem is believed to be a hard problem

- **textbook RSA is not semantically secure (encryption is deterministic) and malleable (due to its homomorphic property)**
  - in practice, textbook RSA needs to be extended with message formatting (PKCS #1)

RSA in practice – special messages

- unconcealed messages
  - a message is unconcealed if it encrypts to itself
    - i.e., if $m^e \bmod n = m$
  - trivial examples for unconcealed messages are $m = 0$, $m = 1$, and $m = n-1$
  - exact number of unconcealed messages is
    - $(1 + \gcd(e-1, p-1))(1 + \gcd(e-1, q-1))$
    - in practice, the number of unconcealed messages is negligibly small

- small messages
  - if $m < n^{1/e}$, then $m^e < n$, and hence $c = m^e \bmod n = m^e$
  - in such a case, m can be computed from c by taking the e^{th} root of c
  - to prevent this, m needs to be pre-formatted (setting most significant bits) to ensure that what is raised to e is not too small (see PKCS #1 formatting)

RSA in practice – small encryption exponent e

- a group of entities may use the same exponent e, but different moduli n_1, n_2, …
- if a message m is sent to at least e recipients and e is small (e.g., 3), then an attacker may find a plaintext m efficiently:
  - assume that the attacker observes $c_i = m^3 \bmod n_i$ ($i = 1,2,3$)
  - let $x = m^3$
  - the attacker must solve for x the following system of congruences:

    $x \equiv c_1 \pmod{n_1}$

    $x \equiv c_2 \pmod{n_2}$

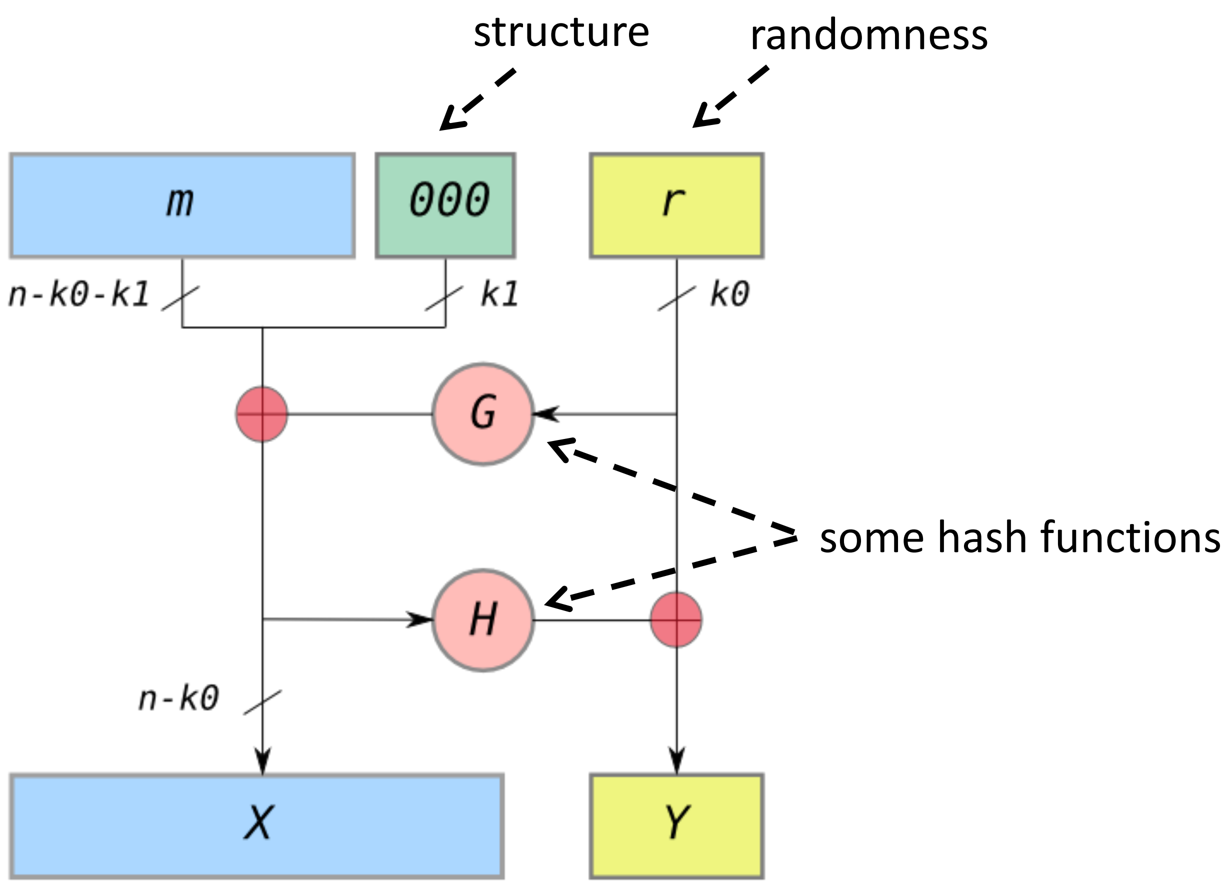
    $x \equiv c_3 \pmod{n_3}$
  - Chinese remainder theorem: if n_1, n_2, …, n_k are pairwise relatively primes, then such a congruence system has a unique solution (mod $n_1 \cdot n_2 \cdot \ldots \cdot n_k$)
  - since $m^3 < n_1 \cdot n_2 \cdot n_3$, the solution found must be m^3
  - the attacker then computes the cube root of m^3 to obtain m

RSA in practice – homomorphic property

- if m_1 and m_2 are two plaintext messages and c_1 and c_2 are the corresponding ciphertexts, then the encryption of $m_1 m_2$ mod n is $c_1 c_2$ mod n
  – $(m_1 m_2)^e \equiv m_1^e m_2^e \equiv c_1 c_2 \pmod n$
- this leads to an adaptive chosen-ciphertext attack on RSA
  – assume that the attacker wants to decrypt $c = m^e \bmod n$
  – assume that an oracle decrypts arbitrary ciphertexts for the attacker, except c
  – the attacker can select a random number r and submit $c \cdot r^e \bmod n$ to the oracle
  – since $(c \cdot r^e)^d \equiv c^d \cdot r^{ed} \equiv m \cdot r \pmod n$, the attacker will obtain $m \cdot r \bmod n$
  – he then computes m by multiplication with $r^{-1} \pmod n$
- we say that textbook RSA is ***malleable***
  – valid new ciphertexts can be constructed from other known ciphertexts
  – this can be circumvented by imposing some structural constraints on plaintext messages → see PKCS #1 formatting

PKCS #1

- v1 – vulnerable to adaptive chosen ciphertext attacks (Bleichenbacher)
- v2 – Optimal Asymmetric Encryption Padding (OAEP) (Bellare-Rogaway)
  http://www.emc.com/collateral/white-papers/h11300-pkcs-1v2-2-rsa-cryptography-standard-wp.pdf



ElGamal and ECC

ElGamal encryption scheme

- key-pair generation
  - domain parameters: p, q, g
    - » p is a large prime (defines a multiplicative group over {1, 2, ..., p-1})
    - » q is a prime divisor of p-1
    - » g in [1, p-1] is an element of order q
      (the smallest positive t satisfying $g^t = 1 \bmod p$ is t = q)
  - **private key: uniformly randomly selected x from [1, q-1]**
  - **public key: $y = g^x \bmod p$**

- encryption
  - input: domain params p, q, g; public key y; message m in [0, p-1]
  - choose uniformly random k from [1, q-1]
  - compute $\mathbf{c_1 = g^k \bmod p}$ **and** $\mathbf{c_2 = my^k \bmod p}$
  - output: $\mathbf{(c_1, c_2)}$

- decryption
  - input: domain params p, q, g; private key x; ciphertext (c_1, c_2)
  - output: $\mathbf{c_2 c_1^{-x} \bmod p} = my^k g^{-xk} \bmod p = mg^{xk} g^{-xk} \bmod p = m$

Notes on ElGamal encryption

- efficiency issues
  - encrypted message is twice as long as the plaintext (message expansion)
  - encryption requires two modular exponentiations, whereas decryption requires only one, but ...
  - all entities in a system may choose to use the same prime p and generator g
    - we can speed up encryption by pre-computation
    - size of the public key is reduced (no need to contain domain parameters)

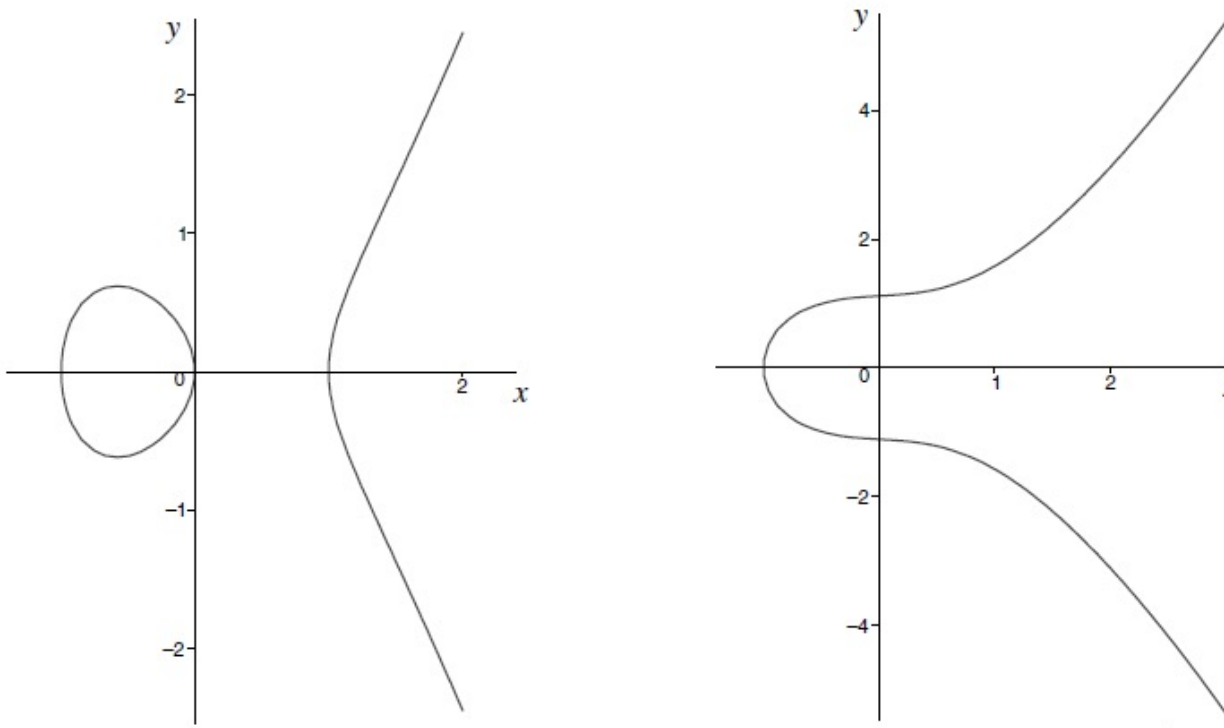
- relation to hard problems
  - computing the private key from the public key is equivalent to the discrete logarithm problem
  - semantic security of the ElGamal scheme is based on the hardness of the so-called Decisional Diffie-Hellman problem, that is *at most* as hard as the discrete logarithm problem
  - recovering m given p, q, g, y, c_1, c_2 is equivalent to solving the Computational Diffie-Hellman problem

The idea of elliptic curve crypto

- ElGamal is essentially defined over a multiplicative cyclic group
  - elements: {1, 2, ..., p-1}
  - group operation: mod p multiplication
- fact: any two cyclic groups of the same order are essentially the same (isomorph)
  - i.e., they have the same structure even though the elements may be represented differently and the group operations may be different
- ElGamal over cyclic subgroups of elliptic curve groups → elliptic curve cryptography
  - elements: points on an elliptic curve
  - group operation: point addition

Elliptic curves over real numbers

- an elliptic curve (over real numbers) is a plane curve defined by an equation of the form $y^2 = x^3 + ax + b$ (Weierstrass equation)
- examples:



(a) $E_1 : y^2 = x^3 - x$ (b) $E_2 : y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$

Elliptic curves over finite fields

- let p be a prime and let F_p denote the field of integers modulo p (with the usual multiplication * and addition + operations)
- consider the elliptic curve E defined by equation

$$y^2 = x^3 + ax + b,$$

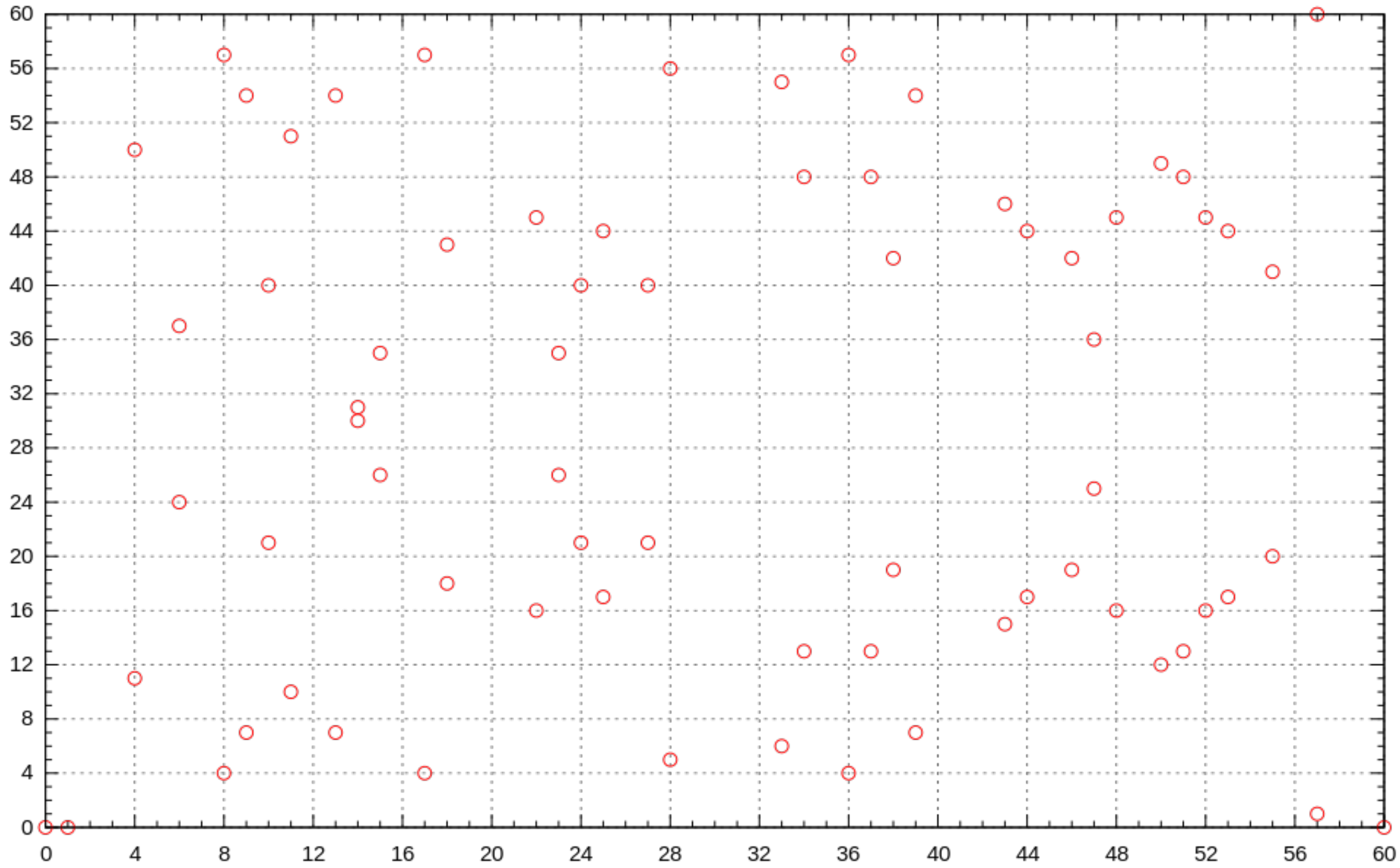
where $a,b \in F_p$ and the operations are the field operations

- $(x,y) \in F_p$ is a point on the curve if it satisfies the equation
- in addition, there is a distinguished point called *infinity* ∞
- the set of all the points on the curve E is denoted by $E(F_p)$

- example:
  - let p = 7 and $y^2 = x^3 + 2x + 4$
  - $E(F_7) = \{\infty, (0,2), (0,5), (1,0), (2,3), (2,4), (3,3), (3,4), (6,1), (6,6)\}$

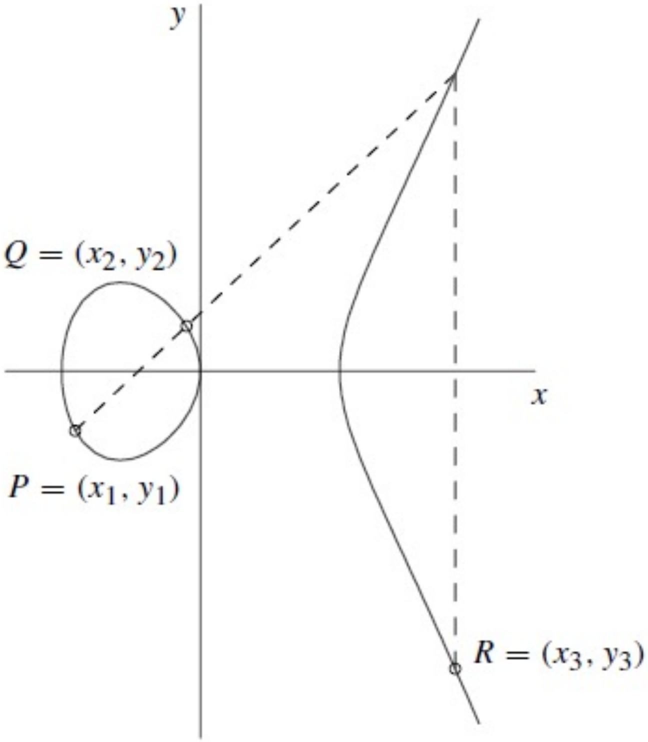
Elliptic curves over finite fields

- example: elliptic curve $y^2 = x^3 - x$ over finite field F_{61}

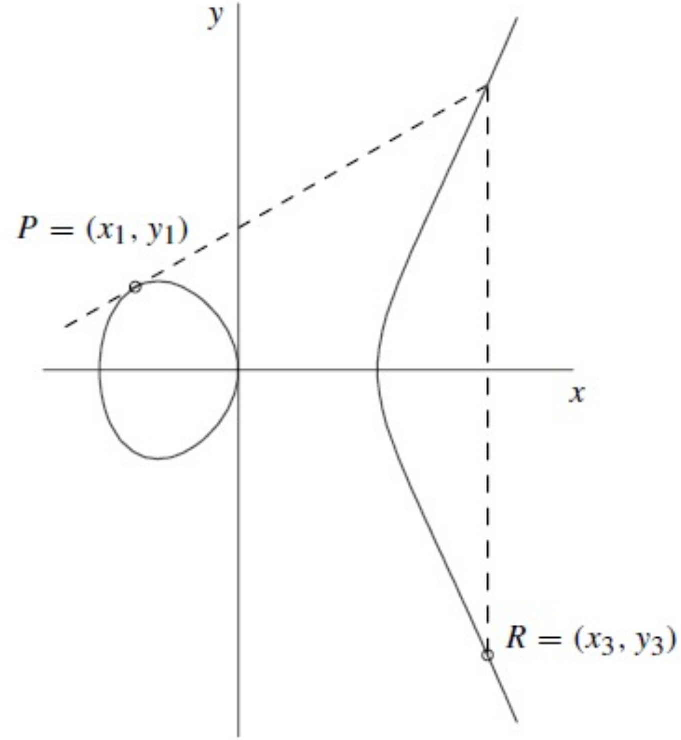


Elliptic curve groups

- we define an addition operation over the points of the curve:
  - illustrative examples (in case of ECs over real numbers):



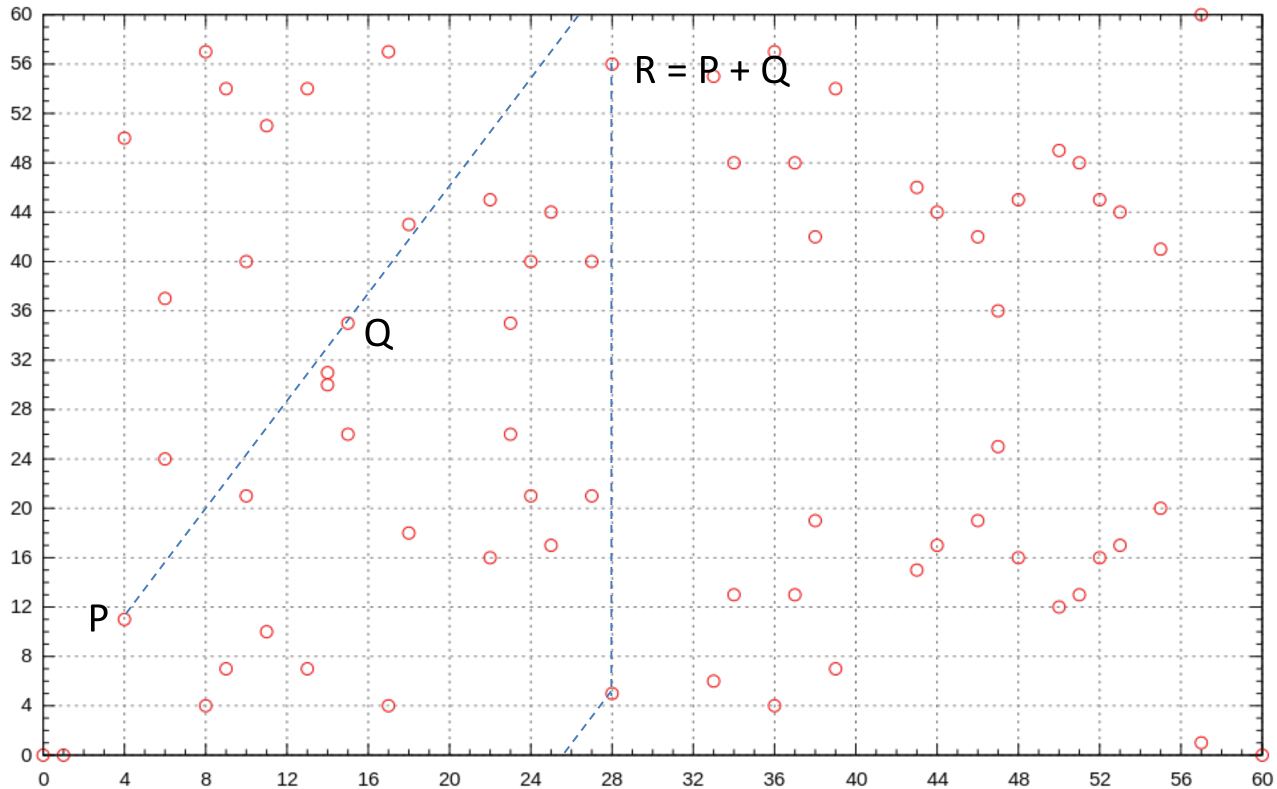
(a) Addition: $P + Q = R$.



(b) Doubling: $P + P = R$.

Elliptic curve groups

- illustrative example in case of an EC over a finite field



- with this addition operation, the set of points $E(F_p)$ form an (additive) abelian group with ∞ serving as the identity element

Cyclic subgroups of elliptic curve groups

- let E be an elliptic curve defined over a finite field F_p
- let P be a point in $E(F_p)$ with prime order n
- then the cyclic subgroup of $E(F_p)$ generated by P is

$$\{ \infty, P, 2P, 3P, \ldots, (n-1)P \}$$

- such cyclic subgroups of elliptic curve groups can be used to implement discrete logarithm systems!

- Elliptic Curve Discrete Log Problem (ECDLP):

  given an elliptic curve group $E(F_p)$, a generator P of a cyclic subgroup of prime order n of $E(F_p)$, and a point Q in that subgroup, find the integer d, $1 \leq d \leq n-1$, such that $dP = Q$

ElGamal over elliptic curves

- EC ElGamal key generation:
  - domain parameters:
    - » prime p
    - » equation defining an elliptic curve E (e.g., $y^2 = x^3 - x$)
    - » point P that defines a cyclic subgroup of $E(F_p)$
    - » the prime order n of the subgroup
  - **private key: uniformly randomly selected integer d from [1, n-1]**
  - **public key: Q = dP**

- EC ElGamal encryption:
  - input: domain params (p, E, P, n); public key Q; message m
  - represent m as a point M in $E(F_p)$
  - uniformly randomly choose k from [1, n-1]
  - compute $\mathbf{C_1 = kP}$ **and** $\mathbf{C_2 = M + kQ}$
  - output: $\mathbf{(C_1, C_2)}$

- EC ElGamal decryption:
  - input: domain params (p, E, P, n); private key d; ciphertext (C_1, C_2)
  - compute $\mathbf{C_2 - dC_1} = M + kQ - dkP = M + kdP - dkP = M$
  - output: extract m from M

Original vs. EC ElGamal

- underlying group
  - cyclic subgroup of prime order q of Z_p* for some prime p
  - generator g
- group operation
  - mod p multiplication, exponentiation (repeated multiplication)
- computations
  - public key: $y = g^x$
  - encryption: $c_1 = g^k$; $c_2 = my^k$
  - decryption: $m = c_2 c_1^{-x}$

- underlying group
  - cyclic subgroup of prime order n of $E(F_p)$ for some prime p and elliptic curve E over F_p
  - generator P
- group operation
  - point addition, point/scalar multiplication (repeated addition)
- computations
  - public key: $Q = dP$
  - encryption: $C_1 = kP$ and $C_2 = M + kQ$
  - decryption: $M = C_2 – dC_1$

Why elliptic curve crypto?

- smaller parameters in ECC provide the same level of security as in traditional schemes (RSA, ElGamal (discrete log – DL)):

	Security level (bits)				
	80 (SKIPJACK)	112 (Triple-DES)	128 (AES-Small)	192 (AES-Medium)	256 (AES-Large)
DL parameter q EC parameter n	160	224	256	384	512
RSA modulus n DL modulus p	1024	2048	3072	8192	15360

- faster operations:
  - private-key operations for ECC are many times more efficient than RSA and DL private-key operations
  - public-key operations for ECC are many times more efficient than those for DL systems
  - public-key operations for RSA are expected to be somewhat faster than for ECC if a small encryption exponent (such as $e = 3$ or $e = 2^{16} + 1$) is selected for RSA

Digital Signature Schemes

Digital signature schemes

- similar to MACs but they are
  - unforgeable by the receiver
  - verifiable by a third party
- services:
  - **message authentication and integrity protection:** after successful verification of the signature, the receiver is assured that the message has been generated by the sender and it has not been altered
  - **non-repudiation of origin:** the receiver can prove this to a third party (hence the sender cannot repudiate)
- examples: RSA, DSA, ECDSA

Functions and terminology

- key-pair generation function $G() = (K^+, K^-)$

  K^+ – public key

  K^- – private key

- signature generation function $S(K^-, m) = s$

  m – message

  s – signature

- signature verification function $V(K^+, m, s)$ = accept or reject

Security of digital signature schemes

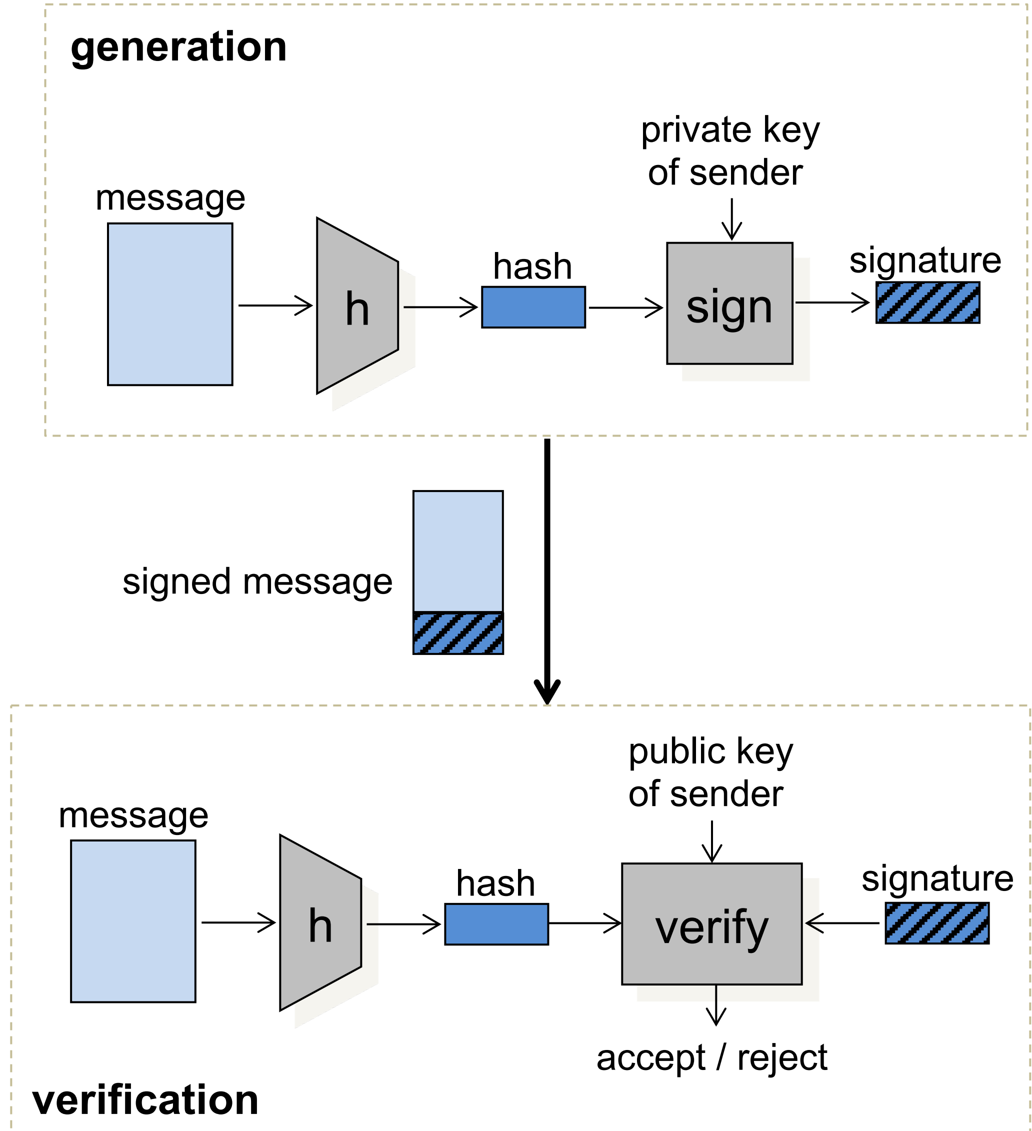
- as in the case of public-key encryption, security is usually related to the difficulty of solving the underlying hard problems

- attacker models:
  - capabilities of the attacker:
    - » key-only attack (attacker knows only the signature verification key)
    - » known-message attack (attacker has message – signature pairs)
    - » (adaptive) chosen-message attack (attcker can choose a message and obtain its signature from an oracle)

  - objectives of the attacker:
    - » forgery
      - attacker is able to compute a valid signature for a message for which no signature has been obtained by observation or from an oracle
    - » key recovery
      - the attacker is able to deduce the private key

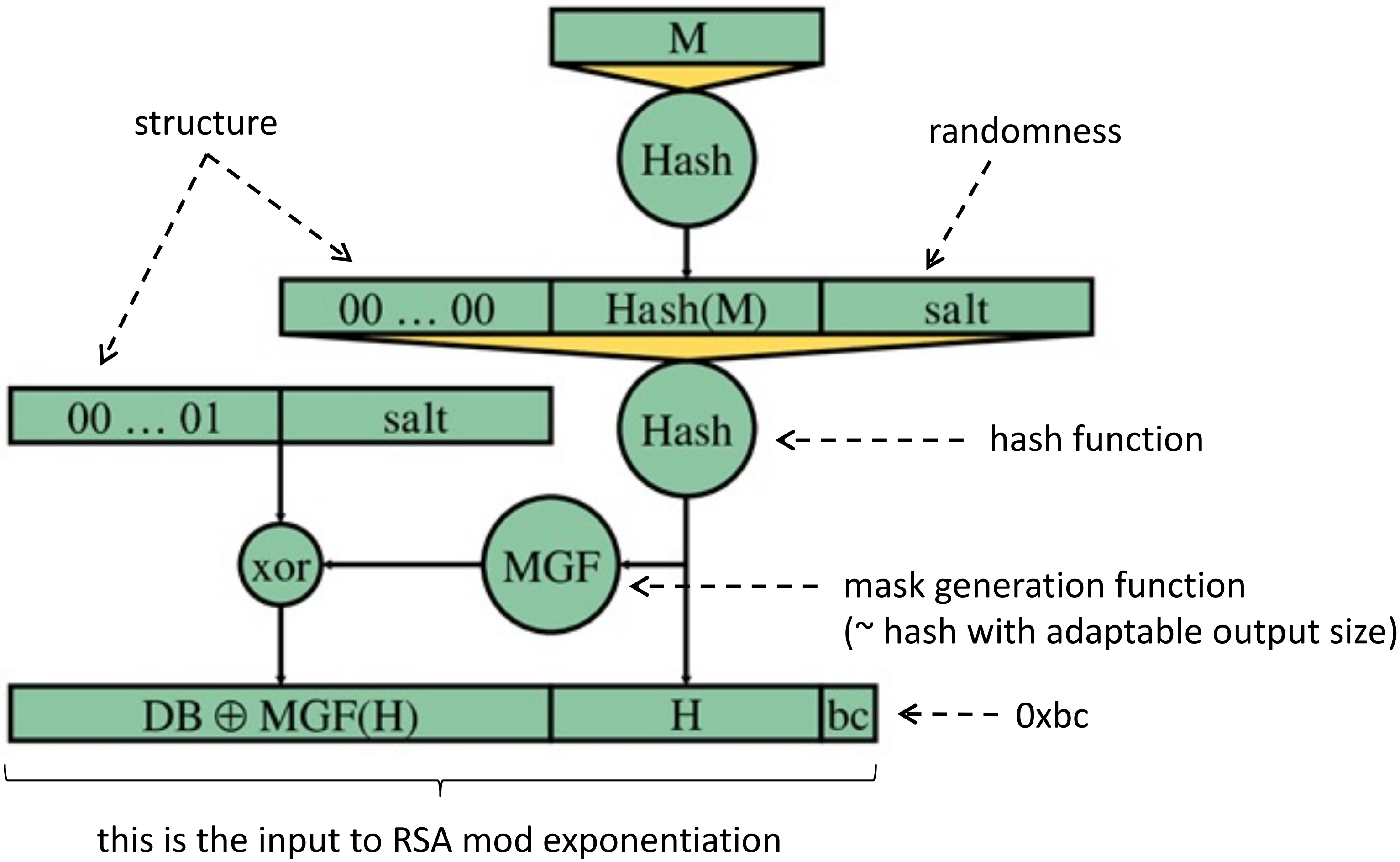
Hash-and-sign paradigm

- public/private key operations are slow
- increase efficiency by signing the hash of the message instead of the message
- it is essential that the hash function is collision resistant (why?)



PKCS #1

- v2 – Probabilistic Signature Scheme (PSS) (Bellare-Rogaway)



http://www.docstoc.com/docs/83431303/PKCS-_1-v21-RSA-Cryptography-Standard
http://rsapss.hboeck.de/rsapss.pdf

Control questions

- What is the basic idea of public-key cryptography?
- What is a digital envelop? (hybrid approach)
- What hard problems is the security of public-key crypto schemes related?
- What is semantic security? How to achieve it?
- How does the RSA algorithm work?
- Which hard problems RSA is related to?
- What are practical issues to consider in case of RSA?
  - unconcealed messages
  - small messages
  - small encryption exponent e
  - homomorphic property
- What is PKCS #1 ?

Control questions

- How does the ElGamal encryption work?
- What is Elliptic Curve Cryptography (ECC) in a nutshell?
- What are the advantages of ECC?
- How does the ElGamal algorithm work over elliptic curves?
- What is a digital signature scheme?
- What is the key difference between MAC functions and digital signatures? What additional security function do signatures provide?
- What attacker models do exist for digital signature schemes?
- What is the hash-and-sign paradigm?
  - Why is it used in practice?
  - What are the requirements on the hash function used?