CRYPTOGRAPHY LECTURE, 2022

Computer Science and Algorithmics, PWr

Mirosław Kutyłowski

Asymmetric Encryption Schemes

Asymmetric encryption

- Alice holds a pair of keys:
  - a private key SK (kept secret by Alice)
  - a public key PK (available to some other people)

Asymmetric encryption

Goal: Bob holding PK can send a message M to Alice so that only M can read it

- $C := \mathrm{Enc}_{\mathrm{PK}}(M)$
- C sent to Alice
- Alice calculates: $M := \mathrm{Dec}_{\mathrm{SK}}(C)$

Conditions:

- without SK it should be infeasible to learn anything about M
- so in particular:
  - infeasible to derive SK from PK
  - Enc must be a randomized function(otherwise test whether a given C fulfils the equality $\mathrm{Enc}_{\mathrm{PK}}(U) = C$ for a candidate U for a plaintext)

Security - formal requirement

- generate a pair of keys $(\mathrm{PK}, \mathrm{SK})$ according to the scheme
- choose plaintexts m_0, m_1
- then for any efficient algorithm $\mathcal{A}$ in the following game the adversary has a negligible advantage

  i. choose $b \in \{0, 1\}$ at random

  ii. present $\mathrm{Enc}_{\mathrm{PK}}(m_b), \mathrm{PK}, m_0, m_1$ to $\mathcal{A}$

  iii. $\mathcal{A}$ outputs b' and wins if $b' = b$

Attention:

the length of the plaintext cannot be fully hidden,

so in the above definition Enc encrypt the messages of a fixed size

Exercise:

you may consider a game with two sequences of messages:

$m_0^1, m_0^2, \ldots, m_0^k$ and $m_1^1, m_1^2, \ldots, m_1^k$ and challenge $\mathcal{A}$ with the ciphertexts

$$\mathrm{Enc}_{\mathrm{PK}}(m_b^1), \mathrm{Enc}_{\mathrm{PK}}(m_b^2), \ldots, \mathrm{Enc}(m_b^k)$$

and ask to guess b

It turns out that this definition is equivalent to the previous one

ElGamal public key encryption

based on a group where DDH assumption holds, e.g.

- take $\mathbb{Z}_p$ for a large prime number p,
- $\mathbb{Z}_p$ has order $p-1=2\cdot q$, choose p so that q is prime
- find $g\in\mathbb{Z}_p$ of order q (take g_0 at random, set $g=g_0^2 \bmod p$ provided that $g\neq 1$

Key generation:

i. choose $x<q$ at random

ii. put $\mathrm{SK}=x$ and $\mathrm{PK}=g^x$

ElGamal public key encryption

Encryption of m

i. choose k at random

ii. $C := (\mathrm{PK}^k \cdot m, g^k)$

Decryption of $C = (A, B)$

calculate $m := A / B^{\mathrm{SK}}$

correctness: $\frac{A}{B^{\mathrm{SK}}} = \frac{\mathrm{PK}^k \cdot m}{(g^k)^{\mathrm{SK}}} = \frac{\mathrm{PK}^k \cdot m}{(g^{\mathrm{SK}})^k} = \frac{\mathrm{PK}^k \cdot m}{\mathrm{PK}^k} = m$

Security of ElGamal

indistinguishable distributions:

- $H_0 = \{(g^x, g^r, g^{x \cdot r} \cdot m_0) : x, r - \text{random}\}$
- $H_1 = \{(g^x, g^r, g^z \cdot m_0) : x, r, z - \text{random}\}$
- $H_2 = \{(g^x, g^r, g^z) : x, r, z - \text{random}\}$
- $H_3 = \{(g^x, g^r, g^z \cdot m_1) : x, r, z - \text{random}\}$
- $H_4 = \{(g^x, g^r, g^{x \cdot r} \cdot m_1) : x, r - \text{random}\}$

ElGamal properties

- **reencryption:** given $(A, B) = (\mathrm{PK}^k \cdot m, g^k)$ one can get another ciphertext of the same m:

$$(A \cdot \mathrm{PK}^\delta, B \cdot g^\delta) \quad (=(\mathrm{PK}^{k+\delta} \cdot m, g^{k+\delta}))$$

- **homomorphic:** $(\mathrm{PK}^k \cdot m, g^k) \cdot (\mathrm{PK}^{k'} \cdot m', g^{k'})$ equals a ciphertext of $m \cdot m'$:

$$(\mathrm{PK}^{k+k'} \cdot (m \cdot m'), g^{k+k'})$$

- **manipulating plaintext** of $(A, B) = (\mathrm{PK}^k \cdot m, g^k)$:

  $(A \cdot u, B)$ is a ciphertext of $m \cdot u$

RSA encryption

- based on RSA numbers: $n = p \cdot q$, where p and q are large prime numbers
- factorization is generally a hard problem if prime factors are large
- take $\mathbb{Z}_n^*$ - the numbers invertible modulo n (that is, coprime with p and q)
- $\mathbb{Z}_n^*$ is a group with multiplication $\bmod n$, with $\phi(n) = (p-1) \cdot (q-1)$ elements

RSA specification

i. find different large primes p, q of bitlength 1024 (or 2048, ...)

   (how to do it??)

   preferably p and q are strong: $(p-1)/2$ and $(q-1)/2$ are prime

ii. $n := p \cdot q$

iii. take e coprime with $(p-1)(q-1)$

iv. compute d such that $e \cdot d = 1 \bmod (p-1)(q-1)$ (Extended Euclidean Algorithm)

Keys:

– $\mathrm{SK} = d$

– $\mathrm{PK} = (n, e)$

Encryption of m

1. $m_0 := \text{encode}(m)$ - get a number $m_0 < n$ (from binary representation via some padding)
2. $\text{Enc}_{n,e}(m) = m_0^e \bmod n$

Decryption of c

1. compute $m_0 := c^d \bmod n$
2. $m := \text{encode}^{-1}(m_0)$

Magic

$$c^d = (m_0^e)^d = m_0^{e \cdot d} = m_0^{1+i\cdot(p-1)(q-1)} = m_0 \cdot m_0^{i(p-1)(q-1)} = m_0$$

the last equality follows from the fact that

– $\mathbb{Z}_n^*$ has $(p-1)(q-1)$ elements

– if a group has k elements, then $a^k = 1$ for each element a from the group (Euler's Theorem)

RSA Assumption

computing the eth root of c is infeasible

(unless you know d such that $e \cdot d = 1 \bmod (p-1)(q-1)$)

Observations

i. if you have d then you may compute p and q:

a. $e \cdot d - 1 = i \cdot (p-1)(q-1) = i \cdot (n + 1 - (p+q))$

b. you may easily estimate i and later find $z = p + q$

c. solve equation $n = x \cdot (z - x)$

ii. so two users must not share the same n

iii. breaking an RSA ciphertext is not necessarily via finding d

(there is a similar scheme - Rabin - where it is equivalent)

Properties of RSA

i. $u^e \cdot v^e = (u \cdot v)^e \bmod n$ so depending on the encoding it might be the case that $\mathrm{Enc}_{n,d}(u) \cdot \mathrm{Enc}_{n,d}(v) = \mathrm{Enc}_{n,d}(u \cdot v)$

ii. due to the size of n the ciphertexts are quite long (e.g. 2K)

iii. computation intensive on long integers (however exponentiation implemented in a clever way)

Hybrid encryption

dividing into block and encrypting each block with RSA would be tedious

Hybrid encryption of a long file D:

i. choose a symmetric key K at random

ii. $C := \mathrm{RSA}-\mathrm{Enc}_{n,d}(K)$

iii. $S := \mathrm{AES}-\mathrm{Enc}_K(D)$

iv. output (C, S)

decryption in the reverse order

Malicious Application - Ransomware:

- ransomware program R installed on a computer
- R runs:
  - i applies a one way-function F to compute a symmetric key K, namely $K = F(D)$ where D is the data to be encrypted (in practice, F is a hash function)
  - ii encrypts D on the disk: replaces D with $\mathrm{Enc}_K(D)$ (symmetric scheme) and attaches $R = \mathrm{RSA} - \mathrm{Enc}_{\mathrm{PK}}(K)$, where PK is the public key
  - iii leaves a message: "pay ... bitcoins to get the decryption key"
- the victim pays ransom,
- the criminal holding the secret key corresponding to PK decrypts R to get K and sends to the victim
- the victim decrypts the ciphertext $\mathrm{Enc}_K(D)$

Pallier scheme

Properties:

- homomorphic scheme: $\mathrm{Enc}_{\mathrm{PK}}(m) \cdot \mathrm{Enc}_{\mathrm{PK}}(m') = \mathrm{Enc}_{\mathrm{PK}}(m + m')$
- based on RSA modulus n and computations modulo n^2
- basic observation: $(1+n)^m = 1 + m + n^2(\ldots) = 1 + m \cdot n \bmod n^2$
- secret information: factors p, q of $n = p \cdot q$

Encryption of m:

let $g = n + 1$

i. choose $r < n$ at random

ii. $c := g^m \cdot r^n \bmod n^2$

Pallier decryption

Decryption keys:

- $\lambda = \mathrm{lcm}\,(p-1, q-1)$,
- $\mu = L(g^{\lambda} \bmod n^2)^{-1} \bmod n$, where $L(x) = \frac{x-1}{n}$

Decryption

$$m := L(c^{\lambda} \bmod n^2) \cdot \mu \bmod n$$

Why it works?

$$c^{\lambda} = (g^m \cdot r^n)^{\lambda} = g^{m \cdot \lambda} \cdot r^{n \cdot \lambda} \bmod n^2$$

order of the group is $\phi(n^2) = p(p-1) \cdot q(q-1)$, but from the structure of the group it follows that the order of each element divides $n \cdot \lambda$ so $r^{n \cdot \lambda} = 1 \bmod n^2$

$$c^{\lambda} = g^{m \cdot \lambda} = 1 + n \cdot m \cdot \lambda \bmod n^2$$

$$L(c^{\lambda} \bmod n^2) = m \cdot \lambda \bmod n$$

Cramer-Shoup encryption

resistant to manipulations,

the same group as for ElGamal, cyclic group with q elements, DDH hard, g_1, g_2 are random generators

key generation:

i. choose $x_1, x_2, y_1, y_2, z < q$ independently at random

ii. $\mathrm{SK} = (x_1, x_2, y_1, y_2, z)$

iii. $c := g_1^{x_1} \cdot g_2^{x_2}$, $d := g_1^{y_1} \cdot g_2^{y_2}$, $h := g_1^{z}$

iv. $\mathrm{PK} = (c, d, h)$ together with parameters g_1, g_2

Cramer-Shoup Encryption of m

i. choose k at random

ii. $u_1 := g_1^k$, $u_2 := g_2^k$

iii. $e := h^k \cdot m$

iv. $\alpha := H(u_1, u_2, e)$

v. $v := c^k d^{k \cdot \alpha}$

vi. output (u_1, u_2, e, v)

Cramer-Shoup Decryption of (u_1, u_2, e, v)

i. (e, u_1) is in fact an ElGamal ciphertext $(h^k \cdot m, g_1^k)$, so m can be derived as before

ii. integrity check:

   a. $\alpha := H(u_1, u_2, e)$

   b. check whether $u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^\alpha = v$

Padding

Problems:

1. the schemes like RSA enable manipulations of the plaintext by manipulations on a ciphertext
2. plaintexts are sometimes too short

To show: well-designed padding can solve the problem

RSA OEAP

– the concept somewhat similar to the Feistel network

– padding transformation before application of the RSA exponentiation

– all-or-nothing concept

OEAP-Optimal Asymmetric Encryption Padding

parameters:

- m is the length of the RSA modulus
- k_0, k_1 are fixed parameters $< m$
- G and H are hash functions: output of G has $m - k_0$ bits, output of H has k_0 bits
- input message of length $m - k_0 - k_1$

padding:

i. add k_1 zeroes to M: $M00\ldots0$

ii. choose a random string r of length k_0

iii. $X := M00\ldots0 \oplus G(r)$

iv. $Y := r \oplus H(X)$

output: $X \| Y$

OEAP

padding:

i. add k_1 zeroes to M: $\quad M00\ldots0$

ii. choose a random string r of length k_0

iii. $X := M00\ldots0 \oplus G(r)$

iv. $Y := r \oplus H(X)$

Reverse operation:

i. $r := Y \oplus H(X)$

ii. calculate $X \oplus G(r)$

iii. if no k_1 zeroes at the end then abort (manipulation detected!)

iv. otherwise truncate k_1 zeroes