Advanced Cryptology

Asymmetric Cryptography

prof. Ing. Róbert Lórencz, CSc., Ing. Josef Kokeš

České vysoké učení technické v Praze, Fakulta informačních technologií
Katedra počítačových systémů

Příprava studijních programů Informatika pro novou fakultu ČVUT je spolufinancována Evropským sociálním fondem a rozpočtem Hlavního města Prahy v rámci Operačního programu Praha — adaptabilita (OPPA) projektem CZ.2.17/3.1.00/31952 – „Příprava a zavedení nových studijních programů Informatika na ČVUT v Praze".
Praha & EU: Investujeme do vaší budoucnosti

Tato přednáška byla rovněž podpořena z prostředků projektu č. 347/2013/B1a Fondu rozvoje vysokých škol Ministerstva školství, mládeže a tělovýchovy

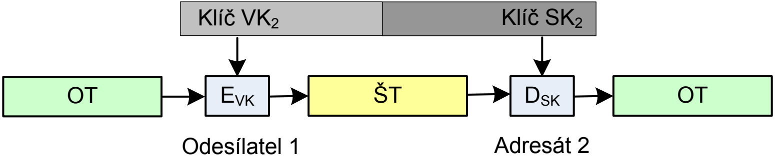
Lecture Contents

- RSA
- Principles of public key (PK) cryptography
- Cryptographic systems with PK
- El Gamal
- Digital signature with El Gamal
- Comparison of RSA and El Gamal

Asymmetric and symmetric cryptography (1)

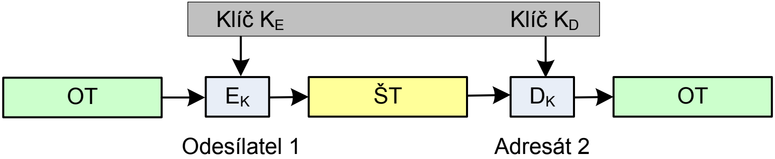
Asymetrické šifrování

SK_2 z VK_2 nelze schůdně vypočítat



Symetrické šifrování

K_E a K_D stejné nebo snadno převoditelné



Odesílatel 1 může i dešifrovat

RSA (1)

Introduction

- Securing communication in a network $\Rightarrow$ every communicating pair must use a different encryption key
- If the encryption key is known, it is easy to generate the decryption key with just a small number of operations.
- A public key (PK) encryption system solves the problem of generating a key for a secured communication.
- The PK encryption system has a two-part key – public key *PK* and private (secret) key *SK*.
  - It is difficult to calculate the decryption key from encryption key.
  - The PK can establish secured network communication beyween several subjects.
  - Every subject has his own *PK* and *SK* for the encryption system.
  - Every subject keeps a certain private information, used to construct *SK*, to himself.
- The list of keys $PK_1, PK_2, \ldots, PK_n$ is public.

RSA (2)

- Subject 1 is sending a message m to subject 2:
  - Message $\rightarrow$ (usually) one block of a certain length, with a related CT block, letters $\rightarrow$ numeric equivalents.
  - Subject 2 uses a decryption transformation to decrypt the CT block.
- Requirement: It must not be possible to find the decryption transformation in a reasonable time by onyone else than subject 2 $\Rightarrow$ an unauthorized subject cannot decrypt the message without knowing the key.

The principle of the RSA encryption system

- Created by Rivest, Shamir and Adleman in 1970.
- RSA is a PK encryption system based on modular exponentiation.
- Pair (e, n) forms the PK; e - exponent and n - modulus.
- $n \rightarrow$ a product of two primes p and q, i.e. $n = pq$, where $\gcd(e, \Phi(n)) = 1$.

RSA (3)

- Encryption: letters $\rightarrow$ numeric equivalents, we are creating the largest blocks we can (with an even number of digits).
- We use the following formula to encrypt the message m to CT c:

$$E(m) = c = |m^e|_n, \quad 0 < c < n.$$

- When decrypting, we use the knowledge of d, the inverse of e modulo $\Phi(n)$, $\gcd(e, \Phi(n)) = 1 \Rightarrow$ the inverse exists. It follows:

$$D(c) = |c^d|_n = |m^{ed}|_n = |m^{k\Phi(n)+1}|_n = |(m^{\Phi(n)})^k m|_n = |m|_n,$$

where $ed = k\Phi(n) + 1$ for some integer k ($|ed|_{\Phi(n)} = 1$) and the Euler's theorem ensures that $|m^{\Phi(n)}|_n = 1$, if $\gcd(m, n) = 1$.

The probability of m and n not being coprime is extremely small. But what happens if m and n are coprime?!

RSA (4)

Proof

- Suppose that $\gcd(m, n) = \gcd(m, pq) \neq 1$
- Because p and q are primes, it follows that:

  $$m = \alpha p \text{ or } m = \beta q,$$

  where α and β are integer numbers such that $\alpha < q$ and $\beta < p$.
- Assume that $m = \alpha p$ and thus $\gcd(m, \beta) = 1$. From the Euler's theorem follows that:

  $$1 \equiv 1^k \equiv (m^{\Phi(q)})^k \pmod{q},$$

  where k is a positive integer.
- We can express $(m^{\Phi(n)})^k$ as:

  $(m^{\Phi(n)})^k \equiv (m^{(p-1)(q-1)})^k \equiv ((m^{\Phi(q)})^k)^{(p-1)} \equiv 1^{(p-1)} \equiv 1 \pmod{q}$
- Then $(m^{\Phi(n)})^k = 1 + \gamma q$, where γ is an integer. By multiplying this equation with m we get:

  $$m(m^{\Phi(n)})^k = m + m\gamma q = m + (\alpha p)\gamma q = m + \alpha\gamma(pq) = m + \alpha\gamma n$$

Proof 2

- It follows:

$$m(m^{\Phi(n)})^k \equiv m \pmod{n} \text{ and}$$

$$D(c) = |c^d|_n = |m^{ed}|_n = |m^{k\Phi(n)+1}|_n = |(m^{\Phi(n)})^k m|_n = |m|_n,$$

Square-and-Multiply algorithm for modular exponentiation

Input: the base element m, the exponent $H = \sum_{i=0}^{t} h_i 2^i$, where $h_i \in \{1, 0\}$ and $h_t = 1$, and the modulus n.
Output: m^H mod n
Initialization: $r = m$
Algorithm:

1. FOR $i = t - 1$ DOWNTO 0
   1. $r = r^2 \bmod n$
   2. IF $h_i = 1$ THEN $r = rm \bmod n$
2. RETURN (r)

Example

- Encryption modulus is the product of primes 43 and 59. We get $n = 43 \cdot 59 = 2537$ as the modulus.
- $e = 13$ is the exponent, where $\gcd(e, \Phi(n)) = \gcd(13, 42 \cdot 58) = 1$, because $\Phi(2537) = (43 - 1) \cdot (59 - 1) = 42 \cdot 58 = 2436$.
- To encrypt the message

  PUBLIC KEY CRYPTOGRAPHY,
- we convert the letters of PT into their numeric equivalents $\Rightarrow$ we form 4-digit blocks (n is four-digit!) and get:

  1520 0111 0802 1004 2402 1724 1519 1406 1700 1507 2423,

  The letter X $= 23$ is used as padding.
- We use the following formula to encrypt PT plocks into CT: $c = |m^{13}|_{2537}$. The first OT block 1520 is encrypted into

$$c = |(1520)^{13}|_{2537} = 95.$$

RSA (5)

- After encrypting all PT blocks we get
  0095 1648 1410 1299 0811 2333 2132 0370 1185 1457 1084.
- To decrypt a message which was encrypted with the RSA cipher, we need to find the invese of $e = |13^{-1}|_{\Phi(n)}$, where $\Phi(n) = \Phi(2537) = 2436$.
- By means of the Euclidean algorithm we receive $d = 937$, which is the multiplicative inverse of 13 modulo 2436.
- We use
  $$m = |c^{937}|_{2537}, \;\; 0 \leq m \leq 2537,$$
  to decrypt block c of the CT. The formula is valid because
  $$|c^{937}|_{2537} = |(m^{13})^{937}|_{2537} = |m \cdot (m^{2436})^5|_{2537} = m,$$
  where we used the Euler's theorem
  $$|m^{\Phi(2537)}|_{2537} = |m^{2436}|_{2537} = 1,$$
  provided that $\gcd(m, 2537) = 1$, and that is true for every block of message m.

RSA (6)

RSA - generating the keys

Ouput: a public key PK = (n, e), private key SK = (d)

1. Choose two large primes p and q.
2. Calculate $n = pq$ and $\Phi(n) = (p-1)(q-1)$.
3. Choose a public exponent $e \in \{1, 2, \ldots \Phi(n) - 1\}$such that
$$\gcd(e, \Phi(n)) = 1$$
4. Calculate SK d such that
$$de \equiv \pmod{\Phi(n)}.$$

We call the pair $PK = (n, e)$ the public key (and publish it), the pair $SK = (n, d)$ the private key.

The condition $\gcd(e, \Phi(n)) = 1$ ensures that the inverse of e modulo $\Phi(n)$ exists and is equal to the private exponent d of the private key.

RSA (6)

We can calculate the exponent d using the Extended Eucleidian Algorithm with input values n and e, which describes the equation:

$$\gcd(\Phi(n), e) = s\Phi(n) + te.$$

If $\gcd(\Phi(n), e) = 1$, we know that e is a valid public key. We also know that the parameter t calculated by EEA is the desired inverse of e, that is,

$$d = t \bmod \Phi(n).$$

If e and $\Phi(n)$ are not co-prime, we choose a new e and repeat the calculation of $\gcd(\Phi(n), e)$.

RSA (6)

The algorithm for generating *PK* and *SK*:

- Each subject finds, in a reasonable time, two random large odd integers p and q with 100 decimal digits.
- From the prime number theorem we know that the probability that these numbers are prime is $\approx 2/\log(10^{100})$.
- To find a prime we need on average $1/(2/\log(10^{100})) \approx 115$ primality tests of such integers.
- We can use Rabin-Miller's probabilistic primality test to determine whether those integers are primes.
- A 100-digit odd integer is tested for "100"witnesses.
- Then the probability that the number being tested is composite is $\approx 10^{-60}$.
- Each subject needs to perform this calculation only twice.

RSA (7)

- As soon as the primes p and q are found $\Rightarrow$ choose an encryption exponent e such that $\gcd(e, \Phi(pq)) = 1$.
- Recommendation: choose some prime $> p$ and q as e.
- If $2^e > n = pq \Rightarrow$ prevents a simple discovery of m by sequential exponentiation of integer c, where $c = |m^e|_n, 0 < c < n$, without performing the reduction modulo n.
- The requirement $2^e > n$ ensures that every block of m has to be reduced modulo n after performing the encrypting exponentiation.

RSA (8)

The security of RSA

- The modular exponentiation needed to encrypt a message with RSA can be performed, for *PK* and *m* of $\approx$ 200 decimal digit size, in a few seconds of computer time.
- If we know *p* and *q* ($\Phi(n) = \Phi(pq) = (q-1)(r-1)$), we can use the Euclidean algorithm to find the decryption key *d*, where $|de|_{\Phi(n)} = 1$, in plynomial time.
- In order to understand why the knowledge of the encryption key (e, n), which is public, doesn't lead easily to the discovery of the decryption key (d, n), we need to realize that finding *d* as the inverse of *e* modulo $\Phi(n)$ requires the knowledge of *p* and *q*, which allow us to easily calculate $\Phi(pq) = (p-1)(q-1)$.

If we don't known the values of *p* and *q*, then finding $\Phi(n)$ is approximately as difficult as the factorization of integer *n*.

RSA (9)

The factorization problem and RSA (1)

- If p and q are 100-digit primes $\Rightarrow$ n is 200-digit.
- The fastest known factorization algorithms need $\approx 10^6$ years to factorize such a number.
- On the other hand, if we know d, but not $\Phi(n)$, it is easy to factorize n, because we know $ed - 1$ is an integer multiple of $\Phi(n)$.
- For this task we have special algorithms for factorization of n using a multiple of $\Phi(n)$.
- No decryption of a message encrypted by RSA has yet been shown without the factorization of n or knowledge of d!
- $\Rightarrow$ if there is no method for RSA decryption without performing the factorization of modulus n, then the RSA system is based on the factorization problem.

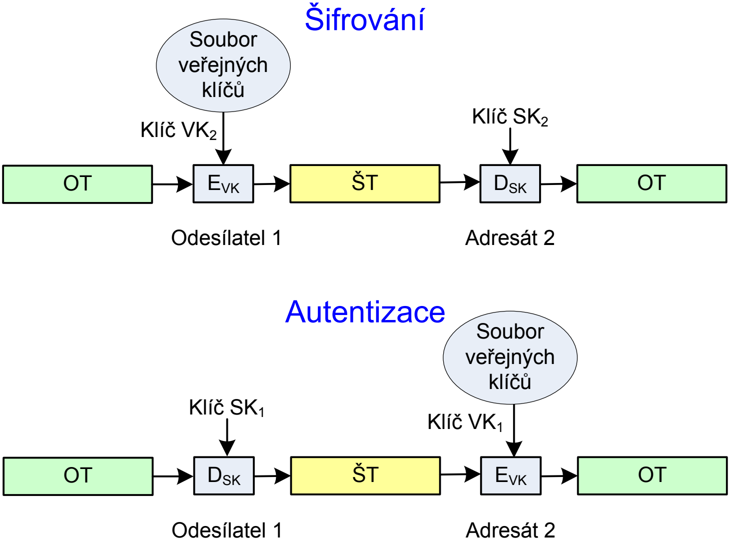
RSA (10)

The factorization problem and RSA (2)

The computational complexity grows with the size of the modulus.

- The messages encrypted by RSA become vulnerable to attacks at the precise moment when the factorization of n becomes feasible in "real world"conditions!
- That means we need to pay particular attention to the choice of primes p and q, to ensure the security of messages which must be secured for decades or centuries.
- We need to protect against fast techniques which can factorize special cases of $n = pq$. For example, the values $p - 1$ and $q - 1$ should have a large prime factor, i.e. $\gcd(p - 1, q - 1)$ should be small and the decimal representation of p and q should have approximately the same length in digits.

The public key cryptography schemes



Digital signatures and RSA (1)

- The RSA encryption system can be used for digitally signing messages.
- The use of signature can assure the recipient, that the message was sent by an authorized sender. This knowledge can be objectively established based on an unbiased test.
- This assurance is useful for e-mail, e-banking, e-commerce etc.

The principle

- Let subject 1 send a signed message m to subject 2.
- The subject 1 calculates for m the PT

$$S = D_{SK_1}(m) = |m^{d_1}|_{n_1},$$

where $SK_1 = (d_1, n_1)$ is the secret decryption key for subject 1.
- If $n_2 > n_1$, where $PK_2 = (e_2, n_2)$ is the public encryption key for subject 2, subject 1 encrypts S by

$$c = E_{PK_2}(S) = |S^{e_2}|_{n_2}, \quad 0 < c < n_2 .$$

Digital signatures and RSA (2)

- If $n_2 < n_1$, subject 1 splits S into blocks of size smaller than n_2 and encrypts each block with an encryption transformation E_{PK_2}.
- To decipher the message, subject 2 first uses his private decryption transformation D_{SK_2} to discover S, because

$$D_{SK_2}(c) = D_{SK_2}(E_{PK_2}(S)) = S\ .$$

- To find the PT m, we will assume that c was sent by subject 1. Subject 2 will now use the public encryption transformation E_{PK_1}, because

$$E_{PK_1}(S) = E_{PK_1}(D_{SK_1}(m)) = m.$$

Here we used the identity $E_{PK_1}(D_{SK_1}(m)) = m$, which follows from

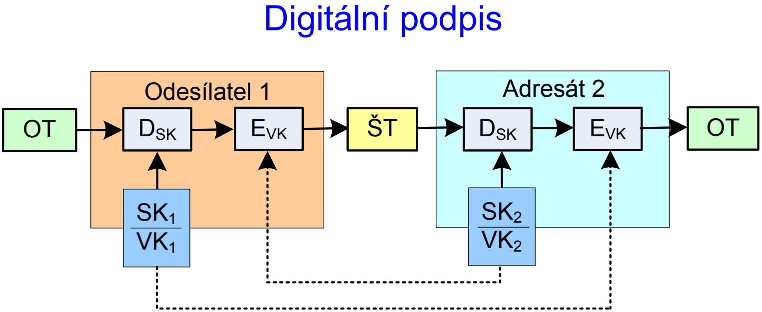
$$E_{PK_1}(D_{SK_1}(m)) = |(m^{d_1})^{e_1}|_{n_1} = |m^{d_1 e_1}|_{n_1} = m,$$

because

$$|d_1 e_1|_{\Phi(n_1)} = 1.$$

Digital signatures and RSA (3)

- The combination of OT m and the signed version S assures subject 2 that the message was sent by subject 1, because no-one else knows the decryption transformation $DSK_1(m)$.
- Furthermore, the subject 1 can not repudiate sending the message, because no other subject than 1 can generate the signed message S from the original message m.



RSA-CRT (1)

Speeding-up the encryption

- To speed-up the encryption, a set of encryption exponents e has been recommended.
- These exponents have a low Hamming weight $\Rightarrow$
- encryption is performed quickly in several steps (see modular exponentiation).
- For example $e = 11_2, 1011_2, 10001_2, 2^{16} + 1, \ldots$

VK e	binary representation of e	#MUL + #SQ
3	11	3
17	10001	5
$2^{16}+1$	1 0000 0000 0000 0001	17

RSA-CRT (1)

Speeding-up the decryption

- In order to speed-up the decryption, we use the separation according to the Chinese remainder theorem - RSA-CRT
- This separation allows us to calculate the decryption with half-length numbers $\Rightarrow$ speed-up of 4 to 8-times compared to the original decryption calculation.

RSA-CRT application

- Let p and q be prime.
- Calculate $n = pq$, $\Phi(n) = (p-1)(q-1)$.
- Choose e, $1 < e < n$, $\gcd(e, \Phi(n)) = 1$ and calculate $d = \left|e^{-1}\right|_{\Phi(n)}$.
- Calculate $d_p = |d|_{p-1}$, $d_q = |d|_{q-1}$, $q_{inv} = \left|q^{-1}\right|_p$.
- The pair $PK = (n, e)$ is the public key and can be published, the sextuplet $SK = (n, p, q, d_p, d_q, q_{inv})$ is the private key.

Encryption and decryption

- Encryption uses the same formula as in RSA: $c = |m^e|_n$.
- To use the RSA-CRT decryption, the following congruence must hold for d_p and d_q:

$$
\begin{aligned}
ed_p &\equiv 1 \bmod (p-1) \\
ed_q &\equiv 1 \bmod (q-1)
\end{aligned}
$$

RSA-CRT (3)

Decryption

1. Calculate
$$m_1 = |c^{d_p}|_p$$
$$m_2 = |c^{d_q}|_q$$
2. Calculate
$$h = \left|\left|q^{-1}\right|_p (m_1 - m_2)\right|_p$$
3. Calculate
$$m = m_2 + hq$$

- Step 1 is the computationally most complex one. But we use half-length digits than with RSA. The calculation is data-independent and can be performed in parallel (or in advance).
- Step 2 uses a computationally simple multiplication of the difference of m_1 and m_2 and the pre-computed constant $\left|q^{-1}\right|_p$, followed by a reduction modulo p.
- The last step requires the least computationally complex

El Gamal (1)

Úvod

- El Gamal - Taher ElGamal
- Another public key cryptography algorithm
- Based on the Diffie-Hellman key exchange, or the discrete logarithm problem
- Like RSA, El Gamal provides encryption and digital signatures

El Gamal (2)

Diffie-Hellman revisited

- Alice (A) and Bob (B) publicly agree on primes g and m relatively co-prime, $1 < g < m$ (more exactly: a group of order m and its generator g)
- A randomly selects a number x such that $0 < x < m$, calculatedd$c = |g^x|_m$ and sends it to B.
- B randomly selects a number y such that $0 < y < m$, calculates $d = |g^y|_m$ and sends it to A.
- A and B both calculate the shared key $k = |d^x|_m = |(g^y)^x|_m = |g^{xy}|_m = |(g^x)^y|_m = |c^y|_m$.
- The attacker cannot calculate $k = |g^{xy}|_m$ from $|g^x|_m$ a $|g^y|_m$ (the Diffie-Hellman problem, DHP).
- DHP is no more complex that the discrete logarithm problem (DLP): If the attacker could solve DLP, he could calculate x from $|g^x|_m$ and then trivially calculate $|g^{xy}|_m$ from x and $|g^y|_m$.
- Is DHP easier than DLP? We don't know, but it seems it isn't.

El Gamal (3)

El Gamal - key preparation

- El Gamal is a modification of DH:
- Alice (A) ~~and Bob (B) publicly agree on~~ selects primes g and m relatively co-prime, $1 < g < m$ (more exactly: a group of order m and its generator g)
- A randomly selects number x such that $0 < x < m$, calculates $c = |g^x|_m$ ~~and sends it to B~~.
- A publishes the triplet (m, g, c) as her public key. x is her private key.

El Gamal (4)

El Gamal - encryption

- Bob wants to send message p to Alice:
- B randomly selects number y such that $0 < y < m$, calculates $d = |g^y|_m$ ~~and sends it to A~~.
- ~~A and B both calculate~~B calculates the shared key $k = |d^x|_m = |(g^y)^x|_m = |g^{xy}|_m = |(g^x)^y|_m = |c^y|_m$
- B encrypted message p as $e = |p \cdot k|_m$
- B sends the pair (d, e) to Alice.

El Gamal - encryption - example

- $m = 2543, g = 5, c = |g^x|_m = 505$ (for $x = 10$, not known to B).
- $y = 123$ (random choice)
  $\rightarrow d = |g^y|_{2543} = 308, k = |c^y|_{2543} = 1883$!! Simplification for the demonstration purposes, y must be different for different blocks, otherwise the cipher gets broken !! See notes below !!
- p = "`ELGAMAL RULES`"$\rightarrow$ `0511, 0701, 1301, 1118, 2111, 0519`
- $e = |p \cdot k|_{2543} \rightarrow$ `0959, 0166, 0874, 2133, 0304, 0765`
- B sends pairs $(308, 959), (308, 166), (308, 874)$... to Alice

El Gamal (6)

El Gamal - dešifrování

- Alice dostala od Boba zprávu (d, e)
- A si spočítá sdílený klíč $k = |d^x|_m = |(g^y)^x|_m = |g^{xy}|_m$.
- A si spočítá $|k^{-1}|_m$ (Eukleidův rozšířený algoritmus)
- A dešifruje zprávu $p' = |e \cdot k^{-1}|_m = |p \cdot k \cdot k^{-1}|_m = |p|_m = p$

El Gamal (7)

El Gamal - decryption - example

- Alice received message $(308, 959)$, that is, $d = 308$, $e = 959$
- A calculates the shared key $k = |d^x|_m = |308^{10}|_{2543} = 1883$.
- A calculates $|1883^{-1}|_{2543} = 1337$ (via the extended Euclidean algorithm)
- A decrypts the message $p' = |959 \cdot 1337|_{2543} = 511 \rightarrow$ "EL"
- Similarly for the other blocks of the message

El Gamal (8)

El Gamal - notes

- We can apply a different invertible operation in place of the multiplication in $e = |p \cdot k|_m$, e.g. an addition modulo m or a xor. That may be beneficial in terms of e.g. speed (of both encryption and decryption).
- Note that the encrypted message is twice the size of the plaintext p.
- Decryption of the communication between A and B is no more difficult than solving the DHP, because if we could solve the DHP, we could calculate $k = |g^{xy}|_m$ from $c = |g^x|_m$ and $d = |g^y|_m$.
- Is the decryption of simpler than solving the DPH? We don't know for sure, but we think it isn't.

El Gamal - poznámky

- El Gamal is not secure against chosen ciphertext attacks, because if the attacker captured a valid encrypted message (d, e), he can generate another valid encrypted message (d, e') with predictable plaintext, e.g. by applying $e' = |2 \cdot e|_m$, where (unknown) $p' = |2 \cdot p|_m$ (consider the case when p represents a price or an account number).
- For every encryption we must choose a different ephemeral key y. If we use the same y for two different messages p_1, p_2, then $|\frac{e_1}{e_2}|_m = |\frac{p_1}{p_2}|_m$ ... by dividing the two ciphertexts we get a fraction of the plaintexts, the key was completely eliminated! If we know one plaintext, we can directly calculate the other; if we know the language of one plaintext, we can use the knowledge of characteristics of this language to decode both plaintexts. Try it on the example above, where we used a constant y.

Digital signature and El Gamal (1)

El Gamal - generating a signature

- Alice wants to sign the message p in such a way that anyone can verify the signature
- The public key is identical to the encryption public key, i.e. the triplet $(m, g, c = |g^x|_m)$
  - Alice randomly chooses a non-repeating y such that $0 < y < m - 1$ and calculates:
  - $r = |g^y|_m$
  - $s = |(p - x \cdot r) \cdot y^{-1}|_{m-1}$
- Repeat these steps until $s \neq 0$.
- The pair (r, s) is the desired signature for message p.

Digital signature and El Gamal (2)

El Gamal - generating a signature - example

- From the private key: $x = 10$
- From the public key: $m = 2543, g = 5, c = 505$
- The message being signed: $p = 1234$
- Alice randomly chose $y = 1111$, which has never been used yet
- $|y^{-1}|_{2542} = 1835$
- $r = |g^y|_m = |5^{1111}|_{2543} = 1567$
- $s = |(p - x \cdot r) \cdot y^{-1}|_{m-1} = |(1234 - 10 \cdot 1567) \cdot 1835|_{2542} = 122$
- The pair $(1567, 122)$ is the signature for message 1234.

Digital signature and El Gamal (3)

El Gamal - verifying the signature

- A valid signature satisfies: $|g^p|_m = |c^r \cdot r^s|_m$
- Why:
  - The equation for s can be rewritten as $p = |xr + sy|_{m-1}$
  - From the Fermat's little theorem: $|a|_{m-1} = |b|_{m-1} \Rightarrow |c^a|_m = |c^b|_m$ for all c. Thus:
  - $|g^p|_m = |g^{xr} \cdot g^{sy}|_m = |(g^x)^r \cdot (g^y)^s|_m = |c^r \cdot r^s|_m$
- We know m, g, c from the public key, p, r, s we received as the message and its signature.
- Nobody but Alice could create the signature, because nobody else knows x.

Digital signature and El Gamal (4)

El Gamal - verifying the signature - example

- From the public key: $m = 2543, g = 5, c = 505$
- The message: $p = 1234$
- The signature: $r = 1567, s = 122$
- Left side: $|g^p|_m = |5^{1234}|_{2543} = 2009$
- Right side: $|c^r \cdot r^s|_m = |505^{1567} \cdot 1567^{122}|_{2543} = 2009$
- Left side = Right side $\Rightarrow$ The signature is valid

Digital signature and El Gamal (5)

El Gamal - notes

- Attempting to recover x, y from $|g^p|_m = |c^r \cdot r^s|_m$ requires solving the discrete logarithm problem, because both x and y appear in the exponent.
- It is essential that y never be repeated: The attacker can use n captured messages $p_1 ... p_n$ and their signatures $(r_1, s_1), ..., (r_n, s_n)$ form a system of n linear congruencies $p_i = |x \cdot r_i + y_i \cdot s_i|_{m-1}$ with $n+1$ unknowns (one x, n different y_i). This system has too many valid solutions to check them all. But if some y was used twice, then the system would only have one solution and by calculating it, the attacker would learn the value of the private key x. See an example in the next slide.

Digital signature and El Gamal (6)

El Gamal - example of using y twice

- Unknown values used for signing: $x = 10, y = y_1 = y_2 = 1111$.
- Known values from the public key: $m = 2543, g = 5, c = 505$
- Known signed messages (p_i, r_i, s_i): $(1234, 1567, 122)$, $(2323, 1567, 425)$
- Note that for $y_1 = y_2$ it necessarily follows that $r_1 = r_2$ (because $r_i = |g^{y_i}|_m$) $\Rightarrow$ it is easy to detect this case.
- We know that $p = |xr + sy|_{m-1}$. Thus in our case: $1234 = |1567x + 122y|_{2542}$, $2323 = |1567x + 425y|_{2542}$. Subtract the first equation from the second: $1089 = |303y|_{2542}$. $|303^{-1}|_{2542} = 797$, and thus $y = |797 \cdot 1089|_{2542} = 1111$. Insert this value into the first equation: $1567x = |1234 - 122 \cdot 1111|_{2542} = 418$. Because $|1567^{-1}|_{2542} = 73$, we get $x = |73 \cdot 418|_{2542} = 10$. We really found Alice's private key!!
- Note: If the required inversions do not exist, it's more difficult to find the solution - but still easier than solving the DLP.

Digital signature and El Gamal (7)

El Gamal - notes

- To generate a false signature (find r, s such that $|g^p|_m = |c^r \cdot r^s|_m$) requires the solving of DLP, because the left side is constant, c^r is determined by (random) choice of r, and s, which needs to be calculated, appears in the exponent. If we choose a s with the idea of calculating the r leads to equation $A = |r^s \cdot B^r|_m$, which is thought to be as difficult as DLP.
- There is an attack (which can also be applied to e.g. RSA signatures) which can use a valid triplet (p, r, s) to generate other valid triplets (P, R, S), but it doesn't allow to choose the value of P (i.e. we can generate fake signatures, but the messages will be nonsensical). Details in [1] and the following slide.

Digital signature and El Gamal (8)

El Gamal - generating fake signatures

- We have a valid signed message (p, r, s), i.e. $|g^p|_m = |c^r \cdot r^s|_m$.
- Choose any A, B, C such that $gcd(A \cdot r - C \cdot s, m-1) = 1$.
- Let $R = |r^A \cdot g^B \cdot c^C|_m$, $S = |\frac{s \cdot R}{A \cdot r - C \cdot s}|_{m-1}$, $P = |\frac{R \cdot (A \cdot p + B \cdot s)}{A \cdot r - C \cdot s}|_{m-1}$
- Then (P, R, S) is also a valid signed message:

$$
\begin{aligned}
|c^R R^S|_m &= |c^R (r^A g^B c^C)^{\frac{sR}{Ar-Cs}}|_m && (1)\\
&= |(c^{R(Ar-Cs)+CsR} r^{AsR} g^{BsR})^{\frac{1}{Ar-Cs}}|_m && (2)\\
&= |(c^{RAr} r^{AsR} g^{BsR})^{\frac{1}{Ar-Cs}}|_m && (3)\\
&= |((c^r r^s)^{AR} g^{BsR})^{\frac{1}{Ar-Cs}}|_m && (4)\\
&= |((g^p)^{AR} g^{BsR})^{\frac{1}{Ar-Cs}}|_m && (5)\\
&= |g^{\frac{pAR+BsR}{Ar-Cs}}|_m && (6)\\
&= |g^P|_m && (7)
\end{aligned}
$$

- For $A = 0$ we can generate signatures without having the original plaintext p.

Comparison of RSA and El Gamal (1)

General properties

- Both RSA and El Gamal facilitate encryption, decryption, signatures and signature verification.
- Both RSA and El Gamal use one-way functions. In case of RSA this is a “trapdoor function” (one way function which can be inverted if we know a special information - here the value of $\Phi(n)$), in case of El Gamal it is a function with special property ($|(g^a)^b|_m = |(g^b)^a|_m$).
- Both RSA and El Gamal can be freely used now (RSA was patented until 2000).

Comparison of RSA and El Gamal (2)

Key preparation

- RSA: Requires generation of two strong primes, the choice of an encryption exponent and the calculation of decryption exponent.
- El Gamal: Requires a choice of a good group and preferably (though not necessarily) finding its generator.
- Conclusion: The preparations are easier for El Gamal.

Key size

- RSA: The public key is (n, e), private key is (d).
- El Gamal: The public key is $(m, g, c = g^x)$, private key is (x).

Comparison of RSA and El Gamal (3)

Security

- RSA: The security is based on the factorization problem.
- El Gamal: The security is based on the discrete logarithm problem.
- Conclusion: Both ciphers are considered secure, but both are broken by a quantum computer and for both we know efficient non-quantum algorithms which can solve some special cases (GNFS for RSA, Index Calculus for El Gamal).
- Assuming the same bit-size of the key and correctly chosen parameters, El Gamal is more secure because it has more valid values for a given set size (typically $n-1$ for El Gamal and $\frac{n}{\log n}$ for RSA).

Comparison of RSA and El Gamal (4)

Implementation

- RSA: All four operations are implemented by modular exponentiation (with different exponents). Encryption and verification can be sped up by choosing a good e, decryption and signature can be sped up using CRT.
- El Gamal: Each operation uses a different algorithm. Besides exponentiation we also use multiplication and modular inverse. A choice of exponent doesn't help much, but we can pre-calculate some values or choose different (faster) operations (XOR instead of the multiplication).
- Conclustion: RSA is much easier to implement and faster to execute.

Srovnání RSA a El Gamal (5)

Plaintext and ciphertext (PT, CT)

- RSA: $CT = PT^e$
- El Gamal: $CT = (g^y, c^y \cdot PT)$ where y is nonce.
- Conclusion: El Gamal's ciphertext is twice as long as RSA's.
- But also: RSA is a random oracle while El Gamal creates different ciphertexts for repeated plaintexts.

Comparison of RSA and El Gamal (6)

Dangerous values

- RSA: $PT = 0$ and $PT = 1$ lead to $CT = PT$. Low values of PT combined with a low e can lead to easily decrypted ciphertexts (using a regual root rather than modular one).
- El Gamal: Failure to choose a non-repeating y enables the attacker to recover the private key (for signatures) or decrypt ciphertexts (for encryption). The same is true for non-repeating but predictable value of y.

Literature

El Gamal - literature

- ElGamal, Taher: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. Advances in cryptology: Proceedings of CRYPTO 84. Lecture Notes in Computer Science 196. Santa Barbara, California, United States: Springer-Verlag. pp. 10–18. Dostupné z: `http://groups.csail.mit.edu/cis/crypto/classes/6.857/papers/elgamal.pdf`