Public-Key Cryptosystems Based on Hard Problems

B.Sc. Thesis

by

Viktória Németh

Mathematics B.Sc., Mathematical Analyst

Supervisor:

Viktória Ildikó Villányi

Assistant Professor

at the Department of Operations Research

Eötvös Loránd University

Budapest 2015.

Acknowledgement

I would like to thank heartly all my teachers who have thought me.
I am especially thankful for my supervisor, Viktória Villányi, who helps me a lot with her advices and proficiency.
This thesis wouldn't have been accomplished without the support of my parents.

Contents

1 Introduction

We can not image our life without sending an e-mail, paying with a credit card, making a phone call or connecting to a wireless LAN. Although, we do not know what welter in the background of these daily routines. Most of us do not even hear about crypthography.

Until the twenties century, cryptography was applied mainly in diplomatic, military and government applications. But with the developement of the telecommunication industries, it become more and more essential. It is widespread mostly in the United Kingdom, the United States, Germany and France.

The greatest milestone in the history of cryptography was the breaking of the Enigma Code during the World War II. Alan Turing developed a machine which was able to dechiper the Germans' messages using mathematical techniques. A movie, The Imitation Game is paying homage to Turing whose work shortend the war with two years. In january of 2015, the prime minister of the United Kingdom proposed to ban the end-to-end encryption in messages. This law was suggested against terrorist, but it raised dust. Just name only a few up to date happenings.

Cryptography is one of the two main branches of cryptology. It is the science which aim is to hiding the meaning of a message. Meanwhile the other one, cryptoanalysis deals with breaking cryptosystems.

Cryptography is also can be splitted into three main parts: symmetric algortihms, public-key algorithms and cryptograpic protocols. Symmetric cryptography has been used since ancient times, meanwhile the others are quite new. In symmetric algorithms there is only one key, which is used for encryption and decryption as well. On the other hand, public-key algorithms - as the name suggest - use not just a private-key, but also a public-key. Furthermore, protocols include the rules which determine the duties with the received message.

The main focus of my thesis is on public-key cryptosystems. My first chapter is about the mathematical theories which are essential for understanding the crypto algorithms. Because modern cryptoraphy is heavily based on mathematical theories, my first chapter is about those which are essential for understanding the mechanisms.

The main body of this dissertation includes two important asymmetric algorithms and their attributes. In the last chapter, I introduce one of the most important applications of this two cryptosystem.

2 Basic Mathematical Needs for Understanding Public-Key Cryptosystems

In cryptography, the two most-significant mathematical fields is number theory and algebra. Number theory deals with the properties of whole numbers, eminently prime numbers. There is lots of algorithms and theories in connection with this type of positive numbers. Large primes are essential for key generation. Moreover, some fundamentals of abstract algebra is introduced in this section.

2.1 Algebra

First and foremost, we need the definition of group, as the basic object.

Definition 2.1. A group (G) is a set of elements together with an operation ($\circ$) which combines two elements of G. A group has the following properties:
1. The group operation $\circ$ is closed. (for all $a, b \in G$ $a \circ b = c \in G$)
2. The group operation is associative. ($a \circ (b \circ c) = (a \circ b) \circ c$ for all $a, b, c \in G$)
3. There is an element $1 \in G$, named the identity element. ($a \circ 1 = 1 \circ a = a$ for all $a \in G$)
4. There is an element a^{-1} for each $a \in G$ called the inverse of a. ($a \circ a^{-1} = a^{-1} \circ a = 1$)
5. G is Abelian group, if $a \circ b = b \circ a$ for all $a, b \in G$.

However, in crypthography we use finite groups. The most important type of groups used in this science is $\mathbb{Z}_n^*$.

Theorem 2.1. *The set $\mathbb{Z}_n^*$ consisting of all integers $i = 0, 1, \ldots n-1$ for which $gcd(i, n) = 1$ forms an Abelin group under multiplication modulo n. The identity element is* 1.

Example 2.1. Let n=8, then $\mathbb{Z}_8^* = \{1, 3, 5, 7\}$.
For show all properties, we use a multiplication table.

	1	3	5	7
1	$1 \cdot 1 \pmod 8 = 1$	3	5	7
3	$3 \cdot 1 \pmod 8 = 3$	1	7	5
5	$5 \cdot 1 \pmod 8 = 5$	7	1	3
7	$7 \cdot 1 \pmod 8 = 7$	5	3	1

It can be easily seen that the table only consists elements from $\mathbb{Z}_8^*$. Furthermore, because of the symmetry, commutativity is satisfied.

2.1.1 Cyclic Groups and Subgroups

Many cryptographic schemes uses groups, mostly cyclic groups and subgroups. Before introducing them, we need some other definition.

Definition 2.2. A group (G) is finite if it has a finite number of elements. We denote the cardinality of the group by |G|.

Definition 2.3. The order (ord(a)) of an element a of a group (G, $\circ$) is the smallest positive integer (k) such that

$$a^k = a \circ a \circ \ldots \circ a = 1,$$

where 1 is the identity element of G.

Example 2.2. Let a=4 and the group is $\mathbb{Z}_7^*$!

$$\begin{aligned} a^1 &\equiv 4 \pmod 7 \\ a^2 &\equiv 2 \pmod 7 \\ a^3 &\equiv 1 \pmod 7 \\ a^4 &\equiv 4 \pmod 7 \\ a^5 &\equiv 2 \pmod 7 \\ a^6 &\equiv 1 \pmod 7 \\ a^7 &\equiv 4 \pmod 7 \\ &\vdots \end{aligned}$$

So the ord(4)=3.

We have seen in the previous example that the results return in terms. This behavior motivates the definition of cyclic groups.

Definition 2.4. A group (G) which contains an element (α) which order is the same as the group cardinality (ord(α)=|G|) is said to be cyclic. These elements are called generators.

This definition says that every element of a group can be written as $\alpha^i \pmod n$, where α is a generator and n is from $\mathbb{Z}_n^*$.

The following theorems give us the most important properties of cyclic groups.

Theorem 2.2. *For every prime (p), $\mathbb{Z}_p^*$ is an Abelian finite cyclic group.*

Theorem 2.3. *Let G be a finite group. Then for every $a \in G$:*
1. $a^{|G|} = 1$
2. ord(a) divides |G|.

Theorem 2.4. *Let G be a finite cyclic group. Then:*
1. The number of generators of G is $\phi(|G|)$.
2. If |G| is prime, then all elements $a \neq 1 \in G$ are generators.

Subgroups are subsets of cyclic groups and they are groups themselves. We can make subgroups with the help of the following theorems.

Theorem 2.5. *Let $(G, \cdot)$ be a cyclic group. Then every element $a \in G$ with the ord(a)=s is the generator of a cyclic subgroup with s elements.*

Theorem 2.6. *Let H be a subgroup of G. Then |H| divides |G|.*

The last theorem describe the subgroups of a finite cyclic group unequivocally.

Theorem 2.7. *Let G be a finite cyclic group of order n and let α be a generator of G. Then for every integer k that divides n there exists exactly one cyclic subgroup H of G of order k. This subgroup is generated by $\alpha^{\frac{n}{k}}$. H consists exactly of the elements $a \in G$ which satisfy the condition $a^k = 1$. This H is the only one.*

2.2 Number Theory

2.2.1 Euclidean Algorithm

Computing the greatest common factor for small positive integers is not so demanding exercise. In this case, you can easily calculate by factorizing the numbers and finding the highest number that divides both of them. For large numbers used in public-key schemes, this is sometimes impossible. So we need an efficient algorithm. Eucledian algorithm reduce the problem for finding the gcd of two smaller number.

The basic observation of Eucledian algorithm is

$$gcd(r_0, r_1) = gcd(r_0 - r_1, r_1),$$

where $r_0 > r_1$ and both of them are positive integers.

We can use this process iteratively.

$$gcd(r_0, r_1) = gcd(r_0 - r_1, r_1) = gcd(r_0 - 2r_1, r_1) = ... = gcd(r_0 - mr_1, r_1)$$

while $(r_0 - m * r_1) > 0$. If we choose the m properly, then the algorithm will finish after few steps. It will occure when we calculate

$$gcd(r_0, r_1) = gcd(r_0 \bmod r_1, r_1).$$

Remark 2.1. If $(r_0 \bmod r_1) < r_1$, then we just change them.

$$gcd(r_0, r_1) = gcd(r_1, r_0 \bmod r_1).$$

The last step is calculating $gcd(r_l, 0)$. In this case, r_l is the greatest common factor of r_0 and r_1.

$$gcd(r_0, r_1) = ... = gcd(r_l, 0) = r_l$$

Example 2.3. Let r_0=1524 and r_1=678.
$1524 = 2 * 678 + 168$

$$gcd(1524, 678) = gcd(678, 168)$$

$678 = 4 * 168 + 6$

$$gcd(678, 168) = gcd(168, 6)$$

$168 = 28 * 6 + 0$

$$gcd(168, 6) = gcd(6, 0) = 6$$

So $gcd(1524, 678) = 6$.

2.2.2 Extended Euclidean Algorithm

In the previous subsection, we have seen the computation of the gdc by recursively reducing the operands. Although, the main application in Euclidean algorithm is not finding the gcd. In public-key cryptography modular inverses have big importance. Beside calculating the gcd, the extended Euclidean algorithm aim is to get the following form:

$$gcd(r_0, r_1) = s \cdot r_0 + t \cdot r_1$$

where s and t are integers. First of all, we execute the Euclidean algorithm, but we express the remainder in every step:

$$r_i = s_i \cdot r_0 + t_i \cdot r_1.$$

At the end, we get:

$$r_l = gcd(r_0, r_1) = s_l \cdot r_0 + t_l \cdot r_1 = s \cdot r_0 + t \cdot r_1.$$

Namely, $s_l = s$ and $t_l = t$.

Example 2.4. Let's use the same values as the previous example: $r_0 = 1524$ and $r_1 = 678$!

$1524 = 2 * 678 + 168$

$$168 = [1]r_0 + [-2]r_1$$

$678 = 4 * 168 + 6$

$$6 = 678 + [-4]168 = r_1 + [-4](r_0 + [-2]r_1) = r_1 + [-4]r_0 + [8]r_1 = [-4]r_0 + [9]r_1$$

$168 = 28 * 6 + 0$

So we get: gcd(1524, 678)=6, s=(-4) and t=9.
Verifying:

$$gcd(1524, 678) = 6 = [-4]r_0 + [9]r_1 = [-4]1524 + [9]678 = (-6096) + 6102.$$

The extended Euclidean algorithm is also used for computing the inverse modulo of an integer. If we would like to calculate the inverse of r_1 modulo r_0 ($r_1 < r_0$), we have to check the $gcd(r_0, r_1)$. Because the inverse exists only if the $gcd(r_0, r_1) = 1$. So r_0 and r_1 have to be relatively prime. Applying the extended Euclidean algortihm, we obtain:

$$s \cdot r_0 + t \cdot r_1 = 1$$

Taking this modulo r_0 :

$$s \cdot 0 + t \cdot r_1 \equiv 1 \bmod r_0$$
$$r_1 \cdot t \equiv 1 \bmod r_0$$

The last equation is the same as

$$t = r_1^{-1} \bmod r_0 \ .$$

So t is the inverse of r_1.
Subsequent upon, computing an inverse a^{-1} modulo p is the same as using the extended Euclidean algorithm with the parameter a and p. The calculated t will be the inverse.

Example 2.5. What is the value of 13^{-1} mod 25?

$gcd(25, 13) = 1$, so we can apply the extended Euclidean algorithm. In this case, $r_0 = 25$ and $r_1 = 13$.

Using the algorithm:
$25 = 1 \cdot 13 + 12$

$$12 = [1] \cdot 25 + [-1] \cdot 13$$

$13 = 1 \cdot 12 + 1$

$$1 = [1] \cdot 13 + [-1] \cdot 12 = [1] \cdot 13 - [1] \cdot ([1] \cdot 25 + [-1] \cdot 13) = [-1] \cdot 25 + [2] \cdot 13$$

The linear combination is the following:

$$1 = [-1] \cdot 25 + [2] \cdot 13 \ .$$

From this equation, the inverse of 13 is

$$13^{-1} \equiv 2 \bmod 25 \ .$$

Verifying:

$$2 \cdot 13 = 26 \equiv 1 \bmod 25 \ .$$

2.2.3 Euler's Phi Function

Definition 2.5. The number of integers in $\mathbb{Z}_m$ relatively prime to m is denoted by $\phi(m)$.

Example 2.6. Let m=8, then $\mathbb{Z}_8 = 0, 1, 2, 3, 4, 5, 6, 7$.

gcd(0, 8)=8 gcd(1, 8)=1
gcd(2, 8)=2 gcd(3, 8)=1
gcd(4, 8)=4 gcd(5, 8)=1
gcd(6, 8)=2 gcd(7, 8)=1

1, 3, 5 and 7 are relatively prime to 8, so $\phi(8) = 4$.

However, this method can be too slow for larger numbers. But if we know the prime factorization of m, then using the next theorem can ease our work.

Theorem 2.8. *Let m have the following canonical factorization*

$$m = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_n^{e_n},$$

where the p_i are distinct prime numbers and e_i are positive integers, then

$$\phi(m) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_{i-1}}).$$

Example 2.7. Let m=1512.

$m = 1512 = 2{\cdot}756 = 2{\cdot}2{\cdot}378 = 2^2{\cdot}2{\cdot}189 = 2^3{\cdot}3{\cdot}63 = 2^3{\cdot}3{\cdot}3{\cdot}21 = 2^3{\cdot}3^2{\cdot}3{\cdot}7 = 2^3{\cdot}3^3{\cdot}7$

Therefore, $n = 3$ and the Euler's phi function is

$$\phi(1512) = (2^3 - 2^2) \cdot (3^3 - 3^2) \cdot (7^1 - 7^0) = 4 \cdot 18 \cdot 6 = 432.$$

So 432 integers in $\mathbb{Z}_{1512}$ are coprime to $m = 1512$.

2.2.4 Fermat's Little Theorem and Euler's Theorem

Extended Euclidean algorithm show the computation of the inverse modulo an integer. Fortunately, there are other ways to get the result. The following theorems are about this.

Theorem 2.9. *Let a be an integer and p be a prime, then*

$$a^p \equiv a \pmod{p}.$$

Rearrange the equation we get:

$$a^{p-1} \equiv 1 \pmod{p}.$$

From this, we have a formula for inverting an integer modulo a prime:

$$a^{-1} \equiv a^{p-2} \pmod{p}.$$

Euler's Theorem is the generalization of Fermat's Little Theorem to any integer moduli.

Theorem 2.10. *Let a and m be integers with gcd(a, m)=1, then*

$$a^{\phi(m)} \equiv 1 \pmod{m}.$$

3 The Public-Key Encryption Schemes

Public-key cryptography or asymmetric cryptography is a quite new branch of cryptology. It was introduced in 1976 by Whitfield Diffie, Martin Hellman and Ralph Merkle. Their basic idea was the following: there is no need for a secure channel for key establishment. However, the messages have to be private. Moreover, the recipient has to be sure about the message origin. These two properties are extremely important in electronic communication.

In this type of cryptosystems, users have two types of keys: a public and a private. The public key - which is known by everyone - is used only for encryption. On the other hand, messages can be decrypted with the private one. The private key is secret and only the recipient owns it.

First of all, we should encipher our message to get the ciphertext. For this procedure, everyone can use the same method. So, we need a function which can be computed in polynomial time. But it's inverse has to be computationally infeasible. This type of functions is called one-way function.

The encryption and decryption procedures have to suit the following properties.
1. Get the original message if you encipher and then decipher.
2. Both procedures should be easy to compute.
3. Deciphering ought to be easy if you know the secret key. Without this, it would be a hard task.

Let's be the two participants A (Alice) and B (Bob). Both of them have different encryption (E_A, E_B) and decryption (D_A, D_B) keys. E_A and E_B are available in the public file. If Bob would like to send a private message to Alice, he needs to do the forthcoming things.
1. He obtains E_A which is inserted in the public file.
2. Then, he enciphers his message with the use of E_A and send it to Alice.
3. Finally, Alice deciphers the message with D_A. The important part of this procedure is that Alice can only decipher those messages which was enciphered with E_A.

Of course, if Alice would reply, she has to do the same.

We have seen that, a secure channel is not needed to establish private communication. The users must share only his encryption key in the public file and he can receive secret message. Although, participants can also establish private communication over an insecure channel without the use of a public file. In this way, they must send directly the encryption key to the other. An eavesdropper will not be able to decipher any message, because deriving the decryption key from the encryption key is impossible.

Besides privacy, the other major property of public-key cryptosystems is the signing. The receiver must be sure about that the message originated from the real sender. Everyone has to have an own digital signature. They are used to avoid two important thing. Later the sender can not deny that the message was sent by him. In addition, the recipient can not modify the message.

An electronic message can be signed in the following way. Let's use the same situation as before.
1. First, Bob has to compute his digital signature for the message. He uses his signing key (D_B) for this.
2. Then, he encrypts the signature with the message using Alice's enciphering key (E_A).
3. After that, Alice should decrypts the text with D_A and verify the signature on it. Now she knows who sent it.

Nowadays, only three types of public-key algorithms are widely used. They are classified by their underlying computational problem. These families have practical relevance. The first is the Integer-Factorization Schemes. It uses the fact that it is difficult to factor large integers. RSA is the most popular from this group. The next one is the Discrete Logarithm Schemes. These algorithms operate in finite fields. For example, the Digital Signature Algorithm. The third one is the Elliptic Curve Schemes. This is the latest proposal. It is the generalization of the Discrete Logarithm Schemes. The Elliptic Curve Digital Signature Algorithm is the most popular from this family.

All of them can be used for key establishement, digital signatures and encryption.

If the operand and key lengths are chosen carefully, all of them are secure. In this case, we are unable to break them.

The algorithms based on numbertheoretic functions. They operate with very long operands and keys. If we use longer operands and keys, we can get more secure algorithms. The security level is the most often used property to compare the cryptosystems. The table below shows the bit length which are needed for different security levels.

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

Figure 1: Bit lengths of public-key algorithms for different security levels [4]

We can see that algorithms which are based on the integer factorization or the Digital Signature Algorithm require long operands and keys. Meanwhile, for elliptic curve schemes we need shorter keys to reach the same security level.

3.1 The RSA

The RSA was invented in 1977 by Ronald Rivest, Adi Shamir and Leonard Adleman. One year after Whitfield Diffie and Martin Hellman proposed public-key cryptography, they inroduced a method for realizing this type of encryption. Nowadays, the RSA is the most widely used asymmetric algorithm. We most often use it for encrypting small private keys. and for digitatal signatures.

The one-way function on which RSA is based is the integer factorization. Because the multiplication of two large primes is computationally easy, but factoring a long integer is nearly impossible. So, number theory plays an important role in RSA. RSA also has influenced the number theoretic research. Researchers are trying to find new algorithms to break RSA.

3.1.1 Key Generation

We have seen above that for public-key cryptosystems two keys are needed. Both of them are a pair of positive integers: the encryption key is (e, n) and the decryption key is (d, n). The encryption one is public, while the decryption key has to be kept private. The value e is named as the encryption exponent and d is the decryption exponent. For an RSA system, we can compute them applying the following steps.
1. First, choose two large primes: p and q which are part of the secret key.
2. Compute n as the product of them: $n = p \cdot q$. As n is part of the public key, p and q must be very large. In this case, factoring n will cause difficulty.
3. After that, compute $\phi(n)$. From *Theorem 2.8.* we get: $\phi(n) = (p-1) \cdot (q-1)$.
4. Select the public exponent: e. This $e \in \{1, 2, \ldots, \phi(n) - 1\}$ and $gcd(e, \phi(n)) = 1$. So e and $\phi(n)$ are relatively prime.
5. At last, compute a part of the secret key: d such that $d \cdot e \equiv 1 \mod \phi(n)$.

To sum up, the public key is consits of e and n, while the public key contains d, p and q. d and e can be computed at once with the use of the extended Eucledian algorithm. But first, e should be selected and it must be satisfied the given conditions. Now we can apply the extended Eucledian algorithm with n and e.

$$gcd(\phi(n), e) = s \cdot \phi(n) + t \cdot e$$

That t is the inverse of e. So:

$$d \equiv t \mod \phi(n).$$

Remark 3.1. For sufficient security level, p and q should be greater than 2^{512}.

3.1.2 Encryption and Decryption

Encryption and decryption procedures are done in the integer ring $\mathbb{Z}_n$. All computation is accomplished modular. Our goal is to get the message as an integer between 0 and $(n-1)$. For this, we can use any standard representation. If our message is in a numeric form, then we can encrypt this plaintext (x). It is done by raising x to the eth power modulo n. e and n are in the public key. As a result, it gives the ciphertext (c).

$$c \equiv x^e \pmod{n}$$

For decryption, we use the decryption exponent (d) from the private key. In this case, the ciphertext (c) is raised to the dth power modulo n. The outcome is the plaintext (x).

$$x \equiv c^d \pmod{n}$$

Remark 3.2. $x \in \mathbb{Z}_n$ and $c \in \mathbb{Z}_n$ as well.

So, if we raise the plaintext to the eth power during encryption and then if we rais this ciphertext to the dth power, the result will be the plaintext.

$$D(c) = D(E(x)) = (x^e)^d \equiv x^{e \cdot d} \equiv x \pmod{n}$$

This is the substantial idea of RSA.

3.1.3 Proof of Correctness

Proving the correctness of RSA scheme, we need to show that encryption (E) is the inverse function of decryption (D).

From the process of key generation, we know the following:

$$d \cdot e \equiv 1 \mod \phi(n).$$

This is the same as:

$$d \cdot e = 1 + t \cdot \phi(n),$$

where t is an integer.

Then we get:

$$D(c) \equiv x^{d \cdot e} \equiv x^{1+t \cdot \phi(n)} \equiv x^{t \cdot \phi(n)} \cdot x^1 \equiv (x^{\phi(n)})^t \cdot x \pmod{n}.$$

Now we have to prove that:

$$x \equiv (x^{\phi(n)})^t \cdot x \pmod{n}.$$

By using the Euler's theorem (*Theorem 2.10.*):

$$1 \equiv 1^t \equiv (x^{\phi(n)})^t \pmod{n}.$$

If x and n are relatively prime: gcd(x, n)=1, then with the use of the Euler's theorem we are ready:

$$D(C) \equiv (x^{\phi(n)})^t \cdot x \equiv 1 \cdot x \equiv x \pmod{n}.$$

Else if, when the $gcd(x, n) = gcd(x, p \cdot q) \neq 1$, then p or q must be a factor of x:

$$x = r \cdot p \text{ or } x = s \cdot q,$$

where r, s are integers and $r < q$, $s < p$.

Let us assume that $x = r \cdot p$. Then $gcd(x, q) = 1$. So:

$$1 \equiv 1^t \equiv (x^{\phi(n)})^t \pmod{q}.$$

We also know that:

$$(x^{\phi(n)})^t \equiv (x^{(q-1)\cdot(p-1)})^t \equiv ((x^{\phi(q)})^t)^{p-1} \equiv 1^{(p-1)} \equiv 1 \pmod{q}.$$

This is equivalent to:

$$(x^{\phi(n)})^t = 1 + u \cdot q,$$

where u is an integer.

If we multiply this by x:

$$x \cdot (x^{\phi(n)})^t = x + x \cdot u \cdot q = x + (r \cdot p) \cdot (u \cdot q) = x + (r \cdot u) \cdot (p \cdot q) = x + (r \cdot u) \cdot n.$$

Then:

$$x \cdot (x^{\phi(n)})^t \equiv x \pmod{n}.$$

Finally,

$$D(c) \equiv (x^{\phi(n)})^t \cdot x \equiv x \pmod{n}.$$

That which was to be demonstrated.

An eavesdropper only knows the public exponent (e), the modulus (n) and the ciphertext (c). His aim is to get the private exponent (d), since he knows it, he will be able to decrypt the ciphertext and read the message. There is one relationship between e, d and n:

$$e \cdot d \equiv 1 \mod \phi(n),$$

where $\phi(n)$ is not known by the attacker.

To reveal the value of $\phi(n)$, n have to be decompose into two primes: p and q. If someone can do this, then he can calculate d easily. By the way, the modulus is very large, 1024 or more bit length. Fortunately, factoring a large number is a difficult task. Even with excellent algorithms, it can be last for hundreds of years. So, we should choose the extent of parameters properly to reach long-term security.

The other possible way is computing $\phi(n)$ without factoring n.
In this case, $(p+q)$ can be obtained from n and $\phi(n) = n - (p+q) + 1$.
$(p+q)$ is equal to $\sqrt{(p+q)^2 - 4 \cdot n}$.
Finally, $q = \dfrac{(p+q) - (p-q)}{2}$.
We know that n is the composite of p and q. If n would be prime, then computing $\phi(n)$ will be trivial. Now we can say that it is no easier than factoring n.

There is an other idea for attack RSA such that determining d without factoring n or computing $\phi(n)$. If the d is known, than $e \cdot d - 1$ can be calculated, because this is a multiple of $\phi(n)$. However, this approach is also not effective.

3.1.4 RSA Padding

That RSA system which is described above has some weaknesses. Hence, a padding scheme has to be used for proper implementation of RSA. Without this padding scheme, the execution of RSA may be insecure.

Some different thing can cause the default of the system.
1. If an attacker has seen several pairs of plaintext-ciphertext, then he could derive informations from an other ciphertext which is encrypted with the same key.
2. Without padding, small public exponents and small plaintexts can cause problem.
3. RSA is malleable. It means that, the attacker is able to transform the ciphertext into an other one. Thus, he can get a known transformation of the plaintext.

To avoid these problems, we use padding. It is founded on a random structure which is embeded into the plaintext. For padding RSA messages, we use the Optimal Asymmetric Encryption Padding (OAEP). It was set up by Bellare and Rogaway in 1994. OAEP is standardized in Public Key Cryptography Standard.

For this type of padding, we need for a one-way permutation (f) and it's inverse (g). This f is k-bit length. We also need two parameters: k_0 and k_1 such that $k_0 + k_1 < k$. The scheme uses two cryptographic hash functions -G and H- fixed by the protocol. n will be the number of bits in the RSA modulus and m will be the plaintext message which is $(n - k_0 - k_1)$-bit length. That f is play the role of the public key and g is the private key.

The encoding procedure includes the following steps:
1. First, the message (m) is padded with k_1 zeroes. So m will be $n - k_0$ bits length.
2. We choose a random k_0-bit string: (r).
3. Then we use G to expand r to $n - k_0$ bits.
4. We compute X: $X = m00\ldots0 \oplus G(r)$.
5. This X is reduced to k_0 bits by H.
6. Let $Y = r \oplus H(X)$.

The output consists of two blocks: $X \mid\mid Y$.

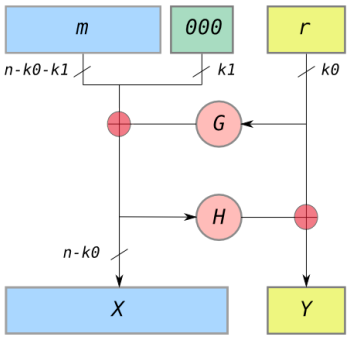

Figure 2: OAEP Diagram

[1]

For decoding:

1. We first have to recover the random string: $r = Y \oplus H(X)$.
2. After that, we can compute the message as $m00\ldots0 = X \oplus G(r)$.

3.1.5 Attacks

To determine whether an encryption scheme is secure or not is a difficult question. We can not be sure about it. There is just one way: trying to attack the system in all possible way. And if it resists, then we can maybe say so.

Since RSA was invented, there have been several test for breaking it. All of them was an experiment against the implementation of RSA. Three types of attacks can be distinguish:

1. protocol,
2. mathematical and
3. side-channel attacks.

[1] http://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding

3.2 The Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is the newest one of the public-key algorithms. It was introduced in 1985 by Victor Miller and Neil Koblitz. ECC uses shorter signatres end keys than RSA, however it provides the same security level. On the other hand, RSA with short keys is faster. Breaking this system needs more effort as it based on the discrete logarithm problem, but operates on elliptic curves.

Definition 3.1. For a Generalized Discrete Logarithm Problem a finite cyclic group (G) with the group operation $\circ$ and cardinality n is given. Let α be a primitive element in G and β be an other element in G. The discrete logarithm problem is finding the integer x, where $1 \leq x \leq n$ such that:

$$\beta = \alpha \circ \alpha \circ \ldots \circ \alpha = \alpha^x.$$

On an elliptic curve we have sets of points which are fulfill a polynomial equation. We have to remark that elliptic curves are not ellipses. In cryptography, we operate with them over a prime field. So the operations are performed modulo a prime. The points on the elliptic curve compose an abelian group.

Definition 3.2. The elliptic curve over $\mathbb{Z}_p$, where $p > 3$ prime, is the set of all pairs $(x, y) \in \mathbb{Z}_p$ such that

$$y^2 \equiv x^3 + a \cdot x + b \pmod{p}$$

together with an imaginary point of infinity θ, where $a, b \in \mathbb{Z}_p$ and

$$4 \cdot a^3 + 27 \cdot b^2 \neq 0 \pmod{p}.$$

Because we work with groups, we have to identify it's properties. First, a set of elements is needed. The elements are those points which are fulfill the overhead equation. Furthermore, we must define the group operations as well. For ease our life, we name it addition, although it is a quite arbitrary choice. We apply coordinate geometry for the addition operation. We have two different cases. The first is when we would like to add two diverse points, this is the point addition. The other one is the addition of a point to itself, which is named as point doubling.

Definition 3.3. Let's have two point on the elliptic curve: $P = (x_1, y_1)$ and $Q = (x_2, y_2)$. The coordinates of the third point, which are the results of point addition or point doubling, will be the following:

$$x_3 = s^2 - x_1 - x_2 \pmod{p}$$
$$y_3 = s \cdot (x_1 - x_3) - y_1 \pmod{p}$$

where

$$s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \pmod{p} & \text{if } P \neq Q \\ \frac{3 \cdot x_1^2 + a}{2 \cdot y_1} \pmod{p} & \text{if } P = Q. \end{cases}$$

Remark 3.3. If $P \neq Q$, then s will be the slope of the line through P and Q. In the case of point doubling, when $x_1 = x_2$ ($x_2 - x_1 = 0$), we use the derivative of the curve as s.

What we still have to establish is the identity element, a θ such that:

$$P + \theta = P$$

where P is a point on the elliptic curve. However, there is not any point which fulfills that equation. Instead of this, an abstract point at infinity (θ) is used for primitive element.

We also need an inverse element:

$$P + (-P) = \theta.$$

The coordinates of $(-P) = (x_1, -y_1)$ if $P = (x_1, y_1)$. So $(-P)$ is the reflect of P along the x-axis. It is true, because

$$-y_1 \equiv p - y_1 \pmod{p}.$$

Theorem 3.1. *The points on an elliptic curve together with θ have cyclic subgroups. Under certain conditions all points on an elliptic curve form a cyclic group, namely an Abelian group.*

To set up a cryptosystem, the order of the group is needed. Although we can not determine the number of points on a curve correctly, we can appreciate it.

Theorem 3.2. *Given an elliptic curve E modulo p. Hasse's bound states that the number of points on the curve ($\#E$) is bounded by:*

$$p + 1 - 2 \cdot \sqrt{p} \leq \#E \leq p + 1 + 2 \cdot \sqrt{p}.$$

This theorem is useful if we would like to get on elliptic curve with an exact number of elements. For example, if $\#E = 2^x$, then we have to use a x-bit length prime.

Now we have everything for establishing the Elliptic Curved Discrete Logarithm Problem, short for ECDLP. This is defined in the following definition.

Definition 3.4. Let E be an elliptic curve over a finite field. A primitive element (P) and an other point (Q) on the curve are given. The problem is finding an integer (d) such that

$$P + P + \ldots + P = d \cdot P = Q,$$

where $1 \leq d \leq \#E$.

That d plays the role of the private key and Q is the part of the public key in the Elliptic Curve Digital Signature Algorithm.

We have a starting point (P) and we compute $2 \cdot P, 3 \cdot P, \ldots, d \cdot P$. So we jump point for point on the curve. In this problem, P and Q are published. Meanwhile, the private key (d) is the number of jumps.

Nowadays, the Elliptic Curve Cryptosystem becomes more popular. There are many new applications which apply this scheme.

4 Digital Signatures

Digital signatures are widely used applications of cryptographic schemes. We use them to provide a method over insecure channels which ensure us about the origin of the message. It has mostly the same functions as the handwritten signatures.

Digital signatures operate like the public-key algorithms. The most influence of this is that we can differentiate who performed a certain cryptographic operation. Because in a symmetric set up both participants have the same keys, they can not do things which can only be connected with one of them. This is why digital signatures lie in public-key cryptography.

The important part of this algorithm is that the person who sends the digital message has to generate a valid signature as well. The sender uses his private key for this, while the recipient uses his public key in order to check the validity.

Let's ALice and Bob be the two participants again. If Bob would like to send a message to Alice with a digital signature, the following should be done.
1. Bob shares his public key.
2. Then he signs the message (x) with the part of his private key. The private key is only known by Bob, hence only Bob can sign in this way.
3. After that, he sends the signature (s) and the message (x) to Alice over a channel.
4. Alice uses a verification function which verify wheter the signature is valid or not. It's output is a binary statement because of the length of the signature and the message. The digital signature is about 2048 bits length. This function needs Bob's public key. We get the true value if the message (x) was signed with that private key which belongs to the given public key.

There are several security services which can be achieved with digital signatures. Messages can not be modified in transit. In addition to these, the sender should be authentic and later he can not deny the creation of the message. Proving the identity of an entity is also essential. As well as, protection against misuse of identity is needed.

Digital signatures can be constructed with each types of the public-key algorithms. So we have signature schemes which are based on the integer factorization or on the hardness of the discrete logarithm prolem in EC groups.

4.1 The Probabilistic Signature Standard

In practice, the most widespread digital signatures schemes are those which are based on RSA encryption. To keep off the potential attacks, we use padding to determine the validity of the message. The Probabilistic Signature Standard (PSS) is a padding scheme for RSA. The result is a combination of a signature and a verification with an encoded message. In practice, we sign the hashed version of the message with any length. A hash function computes a digital fingerprint for messages with fixed length.

The encoding procedure of this scheme is called Encoding Method for Signature with Appendix Probabilistic Signature Scheme. It consists of the following steps [4].

Let $|n|$ be the size of the RSA modulus in bits. The encoded message EM has a at most $|n|-1$ bits length.

1. Generate a random value: *salt*.

2. Form a string M' by concatenating a fixed padding ($padding_1$), the hash value ($mHash = h(M)$ and *salt*.

3. Compute a hash value H of the string M'.

4. Concatenate a fixed padding ($padding_2$) and the value *salt* to form a data block (DB).

5. Apply a mask generation function (MGF) to the string M' to compute the mask value ($dbMask$). In practice, a hash function such as SHA-1 is often used as MGF.

6. XOR the mask value $dbMask$ and the data block (DB) to compute $maskedDB$.

7. The encoded message EM is obtained by concatenating $maskedDB$, the hash value H and the fixed padding bc.

Remark 4.1. n should be at least 1024-bits long.

After that, the signature can be computed by applying the signing operation to EM.

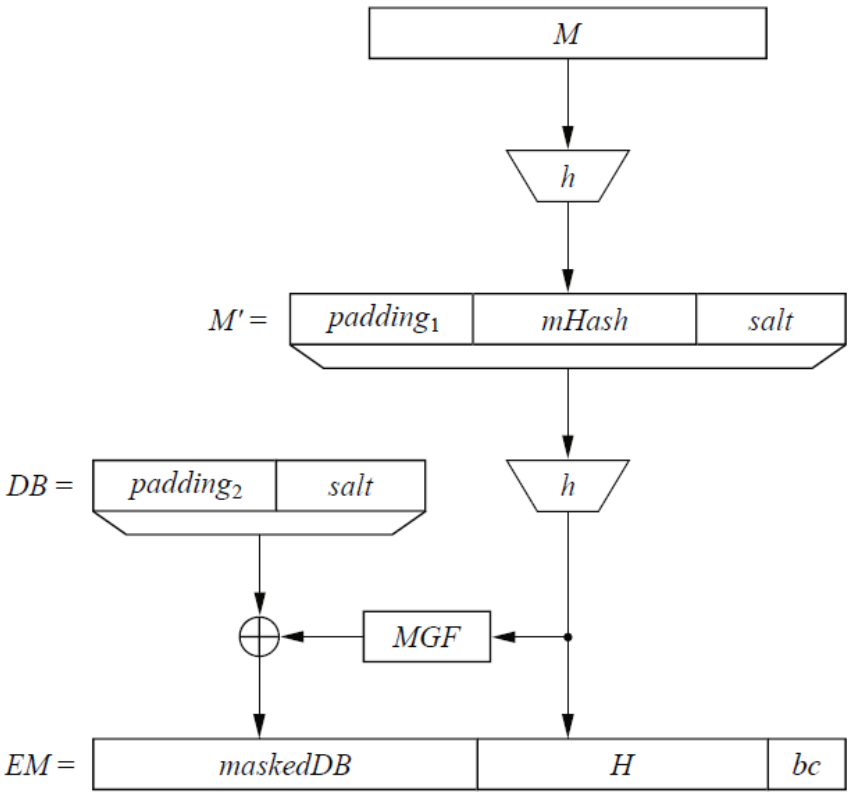

Figure 3: Principle of EMSA-PSS encoding [4]

$padding_1$ and $padding_2$ are available for the receiver and the H is the hashed message. If we add the *salt* to the $padding_2$, EM will become probabilistic. This operation ensures that we get different signatures for every encoding procedure.

For verification, the *salt* need to be recovered and we have to determine wether the encoding of the message is a valid transform of the hash value ($mHash$).

4.2 The Elliptic Curve Digital Signature Algorithm

Based one the hardness of the discrete logarithm problem in elliptic curve groups we can build a digital signature scheme. The Elliptic Curve Digital Signature Algorithm (ECDSA) was introduced in 1998. In this case, we can reach shorter signatures. Moreover, the sufficient security level can be obtained with 160-256 bits length modulus. The standard algorithm is over prime fields ($\mathbb{Z}_p$).

For key generation we have to set up a discrete logarithm problem in the following way.
1. Given an elliptic curve (E). The modulus is p, the coefficients are a and b, respectively there is a primitive element P on the curve which order is q.
2. Then we have to choose an integer(d) such that: $0 < d < q$.
3. After that, we can compute an other point Q with scalar multiplication: $Q = d \cdot P$.

Now we have the two keys. The public key consists of p, a, b, q, P and Q. Meanwhile, the private key is the d.

We need a pair of integers (r, s) for the signature. These two values are as long as q. First, we have to choose an integer k: $0 < k < q$. Then with a point multiplication we compute a point (R):

$$R = k \cdot P.$$

The x-coordinate will be the r: if $R(x_R, y_R)$, then $r = x_R$. After that, with the extended Euclidean algorithm we can compute the s:

$$s \equiv (h(x) + d \cdot r) \cdot k^{-1} \pmod{q},$$

where h is a hash function.

The last step is the verification process. For this, we have to compute three auxiliary value:

1. $w \equiv s^{-1} \pmod{q}$,
2. $u_1 \equiv w \cdot h(x) \pmod{q}$ and
3. $u_2 \equiv w \cdot r \pmod{q}$.

Then with two point multiplications we get a point (A) such that:

$$A(x_A, y_A) = u_1 \cdot P + u_2 \cdot Q.$$

If

$$x_A \equiv r \pmod{q},$$

then the signature is valid.
Else if

$$x_A \neq r \pmod{q},$$

then the signature is invalid.

5 Summary

In the previous sections we see that these public-key algorithms ensure a secure private communication. Todays, there are several projects for building a Quantum Computer. If it turns out, RSA will be undermined because of Peter Shor's [2]algorithm. This can factor big integers in polynomial time. A variant of Shor's algorithm can also be applied for discrete logarithm problems. So elliptic curve cryptosystems are also in danger.

"Every single security function out there is using something called public-key cryptography. It's a specific set of algoriths and they all share one common property – they absolutely spill their guts and fall apart under a quantum computing attack," said Brian Snow, who was a technical director of the Associate Directorate for Education and Training at the National Security Agency.

[2]Peter Williston Shor is an American professor of applied mathematics at the Massachusetts Institute of Technology.

References

[1] Freud Róbert, Gyarmati Edit, *Számelmélet.*
Nemzeti Tankönyvkiadó, Budapest, 2000
ISBN 693-19-0784-8

[2] Buttyán Levente, Vajda István, *Kriptográfia és alkalmazásai.*
Typotex, Budapest, 2004
ISBN 963-9548-13-8

[3] Ronald Linn Rivest, Adi Shamir, Leonard Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.*
Communications of the ACM, 1978

[4] Christof Paar, Jan Pelzl, *Understanding Cryptography.*
Springer-Verlag Berlin Heidelberg, 2010
DOI 10-1007-978-3-642-04101-3-6

[5] Victor Shoup, *OAEP Reconsidered*
IBM Zurich Research Lab, Switzerland, 2001

[6] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern, *RSA-OAEP is secure under the RSA assumption.*
In Advances in Cryptology–Crypto, 2001

[7] Kristin E. Lauter and Katherine E. Stange, *The elliptic curve discrete logarithm problem and equivalent hard problems for elliptic divisibility sequences*
Selected Areas in Cryptography (Pages 309 - 327)
Springer-Verlag Berlin, Heidelberg, 2009
ISBN 978-3-642-04158-7

[8] Kiss Emil, *Bevezetés az algebrába*
Typotex Kiadó, Budapest, 2007
ISBN 978-963-9664-48-7