CS 333 Project Report

Egemen İşcan
egemen.iscan@ozu.edu.tr

Hasan Erdem Bilgin
hasan.bilgin@ozu.edu.tr

Yamaç Demirkan Yılmaz
demirkan.yilmaz@ozu.edu.tr

Peker Çelik
peker.celik@ozu.edu.tr

June 2021

Contents

I. Introduction

In this article, we will discuss a new way of transmitting and receiving any kind of information in an electronic environment with a simple but very secure encryption method. It is very crucial to preserve the validity of a message when transmitting it to its correspondent. Paper messages have two important qualifications. The first one is that they are private, covered in an envelope and they are sealed, so nobody except the correspondent cannot read them. The second is that they are signed by an authority, therefore the correspondent would know that this message is actually coming from the intended sender. All our effort is to protect these properties of a message while transmitting and receiving them in an electronic system by developing a new encryption method.

To ensure these two crucial properties of a message, we have come up with a public key cryptography implementation, which was originally proposed by Diffie and Hellman [1]. Public key cryptography means that each user, for instance, the sender and the receiver, stores a unique encryption method in a publicly visible file. Again, each user would have to come up with a unique decryption method to decrypt the messages, but this time, they will store their decryption methods in a secret place.

Different from Diffie and Hellman's proposal, we are going to present an implementation of the system in action in the upcoming sections.

II. Background

In this section, we will briefly describe the Public-Key Cryptosystems and the advantages that provide us like privacy and signature.

A. Public-Key Cryptosystems

Public-Key Cryptosystems have two main procedures. Those are encryption (E) and decryption (D). Encryption procedure (E) is published publicly

for each user. However, the decryption procedure (D) must be kept private. These procedures have the following four properties:

a. Deciphering an encrypted message (M') results in getting the original message. In math notation:
$$D(E(M)) = M \tag{1}$$
$$D(M') = M$$

b. Both E and D are computed easily.

c. Revealing E publicly does not cause any vulnerability because revealing E does not get computed D easier. This means that only the receiver who has D can see the messages encrypted with E.

d. Because of the mathematical concept of Public-Key Cryptosystems also, E and D is reverse, following notation is also correct.
$$E(D(M)) = M \tag{2}$$

Procedures have a general method and a secret key. The method encrypts the message (M) to the form called ciphertext (C). Everybody can use the same method unless the key is revealed. If so, vulnerability occurs. Presenting E does not provide any practical approach to find out D based on the workload of the computation.

Any function that satisfies (a) - (c) is known as the "trap-door one-way function". Trap-door functions are the functions that cannot be reversed. Therefore the only way of revealing the function is brute force, which is impractical. The property (d) is necessary for signing the message. In the next chapters, we will show some scenarios that we suppose Alice (A) and Bob (B) use this cryptosystem to communicate and they use E_A, D_A, E_B, D_B.

B. Signatures

A signature is a stamp on a message that uniquely identifies the sender. As we mentioned in the previous section, Bob sends Alice a message. The requirements that we need to satisfy are that the message needs to be encrypted,

only Alice must be able to decrypt the message and Alice must be sure that the sender is Bob. So let's see how it is done.

Bob encrypts the message M with his Decrypt function D_B and creates signature S. Then Bob encrypts his signature S with the encrypt function of Alice E_A to get M. Then Alice receives the encrypted message. First Alice uses E_B to get the signature after that uses his private D_A to get the message. Since if the sender was not Bob E_B would not get us the signature. In mathematical terms:

$$S = D_B(M)$$

$$E_B(S) = E_B(D_B(M)) = M$$

Bob computes $E_A(D_B(M))$ to get the encrypted version of the signature, which includes the message at the same time. And all Alice need to do is decrypt the message with Her decrypt method D_A and use the public method of Bob E_B to transform the signature to a readable message.

C. Privacy

Encryption is the standard means of rendering a communication private. Let's call Charlie and listen to the conversation between Alice and Bob. The aim of this system is to make Charlie read garbage, in other words, encrypted messages which actually include the context and signature at the same time. Unless any private key or Decrypt method is leaked, the encryption is quite unbreakable based on the computation necessary and one-way trap function that we use due to Diffie and Hellman. Using two encryption that Bobs signature and Alice's Encryption function consists of the sender Bobs identified correctly and only Alice can read the message. Also changing the keys and using different keys per conversation increases the level of privacy.

III. Our Encryption and Decryption Methods

We assign the letter M for the message to be encrypted, and the pair of integers (e, n) signifies the public key, where $e \geq 0$, $n \geq 0$. If message M is larger than n, it is required to break into smaller parts where each message is between 0 and n-1. The encryption and decryption algorithms may then be computed as follows:

Raising M to the power e modulo n:

$$C \equiv E(M) \equiv M^e$$

Then you may raise C from the previous part to power d modulo n, in order to compute the decryption algorithm D:

$$D(C) \equiv C^d$$

Keep in mind that an encrypted message is not larger than the original. The encryption key is the pair of (e, n) and the decryption key is the pair of (d, n). For each user, the encryption key is left public while the corresponding decryption key is kept private.

In order to choose the encryption and decryption keys, we should determine two very large prime numbers and assign their product to the public variable n.

$$n = p \cdot q$$

Since the primes p and q are large, this will make them difficult to reveal because factoring n is computationally impractical. This also obscures the method of deducing d from e.

Then pick an integer d, preferably a large random integer, which satisfies the following condition:

$$gcd(d,\ (p-1)\cdot(q-1)) = 1 \qquad (gcd\ standing\ for\ "greatest\ common\ divisor")$$

The reciprocal for d modulo (p-1).(q-1) is the integer e which is derived from p, q and d:

$$e \cdot d \equiv 1\ (mod\ (p-1)\cdot(q-1)) \tag{3}$$

IV. The Underlying Mathematics

Fermat-Euler theorem (or Euler's totient theorem) states that $a^{\phi(n)} \equiv 1\ (mod\ n)$ if a is coprime to the modulus n, where ϕ is Euler's totient function. The output of this function is the number of integers greater than 0 that are relatively prime to n.

We use this theorem to show that the deciphering algorithm is correct by plugging M in place of a where M is relatively prime to n,

$$M^{\phi(n)} \equiv 1\ (mod\ n) \tag{4}$$

Therefore, for prime numbers p,

$$\phi(p) = p-1$$

If n = p · q, by a fundamental property of the totient function we have

$$\phi(n) = \phi(p)\cdot\phi(q)$$

Using both of the properties above

$$\begin{aligned}\phi(n) &= (p-1)\cdot(q-1) \\ &= n-(p+q)+1 \end{aligned} \tag{5}$$

As we have already demonstrated (3) d·e is relatively prime to (p-1) · (q-1) which were also shown to be equal to ϕ(n).

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

We can now prove that (1) and (2) hold

$$D(E(M)) \equiv (E(M))^d \equiv (M^e)^d \ (mod \ n) = M^{e\cdot d} \ (mod \ n)$$

$$E(D(M)) \equiv (D(M))^e \equiv (M^d)^e \ (mod \ n) = M^{e\cdot d} \ (mod \ n)$$

And so

$$M^{e\cdot d} \equiv M^{k\cdot\phi(n)+1} \ (mod \ n)$$

Observe the two following equations

$$M^{\phi(n)} \equiv 1 \ (mod \ n)$$

$$M^{p-1} \equiv 1 \ (mod \ p)$$

From which we derive the below equation since we have shown in (5) that (p-1) divides ϕ(n).

$$M^{k\cdot\phi(n)+1} \equiv M \ (mod \ p)$$

When $M \equiv 0 \ (mod\ p)$ is true, it implies that the above statement is also true. Therefore this equality is valid for all M. When we apply the same reasoning to q, we get

$$M^{k\cdot\phi(n)+1} \equiv M \ (mod\ q)$$

Here, the last two equations combined mean that for all M

$$M^{e\cdot d} \equiv M^{k\cdot\phi(n)+1} \equiv M \ (mod\ n)$$

We can draw the conclusion that consequently, (1) and (2) hold for all M where $0 \leq M < n$. Hence, D and E are an example of reciprocal permutations.

V. Algorithms

In this section, we will try to explain how to encrypt and decrypt the message in the least time consuming way allowed by the public key cryptography algorithm.

A. How to Encrypt and Decrypt Efficiently

Computing M^e (mod n) will require at most $2 * \log_2 e$ multiplications and $2 * log_2 e$ divisions using the following methodology:

Step 1: Let $e_k e_{k-1} e_{k-2} ... e_1 e_0$ be the binary representation of e.
Step 2: Set the variable C to 1.
Step 3: Repeat steps 4 and 5 for $i = k, k-1, k-2, ..., 0$.
Step 4: Set C to C^2 (mod n).
Step 5: If $e_i = 1$, then set C to $C * M$ (mod n).
Step 6: Stop, C is the encrypted form of M.

Decryption can be done by following the same methodology using d instead of e.

This methodology is called "exponentiation by repeated squaring and multiplication". There are more efficient procedures than this technique. The reason we are using this approach is, since the encryption and decryption is identical to each other, the whole algorithm can be implemented on a few special-purpose integrated circuit chips.

High-speed machines can encrypt a 200-digit message M in just a few seconds. Special-purpose machines can perform the same task even more quickly. The encryption time per block grows at the same rate as the number of digits in n cubed.

B. How to Find Large Prime Numbers

For every single user, we have to append two randomly chosen large numbers p and q, ideally 100 digits each, to serve as a pair of encryption and decryption keys. Since the numbers are large it would be difficult to compute their product $n = p * q$ that's included in the public file.

One solution to coming up with a 100-digit prime number is generating odd 100-digit numbers until a prime number is found. Approximately $(ln10^100)/2 =$ 115 numbers will be generated before a prime is found, according to the prime number theorem.

In order to find out if a large number b is prime, the following "probabilistic algorithm" of Solovay and Strassen* is proposed. It works by picking a random number a between 1 and $b-1$ and tests if

$$gcd(a,\ b = 1)\ and\ J(a,b) = a^{(b-1)/2}\ (mod\ b),$$

Where $J(a,b)$ is the Jacobi symbol. The above statement always holds if b is prime. In the case of b being composite, the statement would be false with a 50% probability. We should check if the statement is true for a 100 random numbers for a, this means that b is most likely prime. Let's consider that instead

of a prime number, a composite was used in the algorithm. This would not pose a great problem since the receiver would determine that there was a problem with the decryption process.

The Jacobi symbol $J(a,b)$ has a value in -1, 1 where b is odd, $a \leq b$ and $gcd(a,b) = 1$. It can be computed by the pseudocode

$$J(a,b) = \textbf{if } a = 1 \textbf{ then } 1 \textbf{ else}$$
$$\textbf{if } a \textit{ is even } \textbf{then } J(a/2,\ b) \cdot (-1)^{(b^2-1)/8}$$
$$\textbf{else } J(b\ (mod\ a),\ a)\ \cdot (-1)^{(a-1)\cdot(b-1)/4}$$

Notice that this algorithm does not try to factor a number to test its primality.

Further security may be provided if the following steps are followed. $(p-1)$ and $(q-1)$ should both contain large prime factors, $gcd(p-1,\ q-1)$ should be small and p and q should have different lengths by a few digits.

In order to come up with a prime number p where $(p-1)$ has a large prime factor, first generate a large prime number u, then have p as the first prime in the sequence $i \cdot u + 1\ with\ i = 2,\ 4,\ 6, ...$ Even more security can be achieved by making sure that $(u-1)$ also has a large prime factor.

It is relatively easy for a modern computer to determine whether a 100-digit number is prime or not, and it would be a matter of minutes for the computer to determine the first prime after a given point.

Another method for finding large prime numbers is to test the primality of a number whose preceding number's factorization is known. If a prime p is found, the factorization of $p-1$ can be used to prove that it is truly prime.

C. How to Choose d

We can choose any d if and only if the number we choose must be a number which is relatively prime to $\phi(n)$. There is an important factor that we should consider while we are choosing d. It must be chosen from a dense possibility space to ensure a cryptanalyst cannot find it on first try.

D. How to Compute e from d and ϕ(n)

Use the following variation of Euclid's algorithm for computing the greatest common divisor of $\phi(n)$ and d in order to compute e. Calculate $gcd(\phi(n), n)$ by computing the series $x_0, x_1, x_2, ...$, where $x_0 \equiv \phi(n)$, $x_1 = d$, *and* $x_{i+1} \equiv x_{i-1} (mod\, x_i)$. If you find a k such that $x_k = 0$, halt. At this step $gcd(x_0, x_1)$ should be equal to x_{k-1}. Then for each x_i find two integers a_i and b_i which satisfies the equation $x_i = a_i \cdot x_0 + b_i \cdot x_1$. Finally, check if $x_{k-1} = 1$. If so, b_{k-1} is the multiplicative inverse of x_1 $mod(x_0)$.

If e is not less than $\log_2 n$, choose another value for d.

VI. Security of the Method: Cryptanalytic Approaches

The only way to ensure that an encryption algorithm is secure is to try to break the algorithm. The NBS (National Bureau of Standards) standard was verified in this way. If a method successfully resisted so many different attacks, then it may be considered secure. But, there is still a controversy on this NBS standard approach [2].

In the next sections, we will show that trying to break this algorithm is hard as factoring n. Fermat and Legendre developed different factoring techniques, some of today's more efficient algorithms based on Legendre's technique. As we will see in section A, no one did develop an algorithm to factor n in an implementable amount of time. We can certify our algorithm as partially certified, since factoring n is not possible in a reasonable amount of time in today's knowledge.

In the following sections, we consider the ways a cryptanalyst may try to find the private decryption key from the public encryption key. We do not think of ways of protecting the decryption keys from the theft physically. Normal security measures must satisfy our needs in terms of protecting the private keys.

A. Factoring n

An attacker cryptanalyst might "crack" our technique by factoring n. Using the factors of n, he could compute $\phi(n)$ and hence d. Luckily, it appears that factoring a number is far more difficult than identifying if it is prime or composite.

There are plenty of factoring algorithms. There are several factoring algorithms. Knuth [3] does a good job of presenting several of them. Pollard describes a method for factoring an integer n in $O(n^{1/4})$ time. Richard Schroeppel (unpublished) developed the quickest factoring method known to the authors; it can factor n in approximately

$$\begin{aligned} exp\sqrt{ln(n) \cdot ln(ln(n))} &= n^{\sqrt{lnln(n)/ln(n)}} \\ &= (ln(n))^{\sqrt{ln(n)/ln(ln(n))}} \end{aligned}$$

steps (here ln denotes the natural logarithm function). The table below shows the number of operations necessary to factor n using Schroeppel's approach, as well as the time necessary if each operation takes one microsecond (in decimal digits), for various lengths of the number n.

Table

Digits	Number of operations	Time
50	1.4×1010	3.9 hours
75	9.0×1012	104 days
100	2.3×1015	74 years
200	1.2×1023	3.8×109 years
300	1.5×1029	4.9×1015 years
500	1.3×1039	4.2×1025 years

We suggest that n be between 200. Depending on the relative significance of encryption speed and security in the targeted application, longer or shorter lengths can be applied. An 80-digit n offers considerable security against current technological attacks; employing 200 digits gives a margin of safety against novel advances.

B. Computing ϕ(n) Without Factoring n

Let's assume that $\phi(n)$ has been computed. Computing d as the reciprocal for e modulo $\phi(n)$. This method has the same level of complexity as factoring n because it would imply the factorization of n using $\phi(n)$. We could do this by calculating $(p+q)$ from n and $\phi(n) = n-(p+q)+1$. Then $(p-q)$ is equal to $\sqrt{(p+q)^2-4n}$. q may be expressed as $((p+q)-(p-q))/2$.

This is not a practical approach and is as difficult as factoring.

C. Determining d Without Factoring n or Computing $\phi(n)$

We state that factoring n is not harder than computing d for a cryptanalyst. If d is revealed then n can be factored without any trouble, this is essential. n can be factored by a cryptanalyst when the he knows d. Since $e \cdot d - 1$ is a multiple of $\phi(n)$. On the other hand if n is large enough then factoring n and determining d is equally likely hard.

If a cryptanalyst gets lucky to find a d' (equivalent of d) then bruteforce may break the system. Yet, the possibility of finding such d' is not more than factoring n. The mathematical background saved us in this case.

D. Computing D in Some Other Way

There are some problems known to be difficult such as factoring and although "computing e-th roots modulo n without factoring n" may not be one of those famous problems, it still may not be easy-to-compute. If we could prove that any algorithm which causes an exception to our system, can be simplified to a factoring problem; this would imply that breaking this system is just as hard to solve a factoring problem. Since it hasn't been proved yet, this is not the case for now.

E. Conclusions

In this paper, we proposed a new technique for implementing a public-key cryptosystem whose reliability and security mainly depends on the difficulty of factoring very large prime numbers. If the security of this algorithm proves to be reasonable and adequate, it allows secure communication between the sender and the receiver without the need of carrying keys with a courier, and it also allows the messages to be digitally signed with a unique structure. This was our main goal at the beginning of the paper.

The main security point, which is the factoring of the large numbers, needs to be examined in more detail. If the system successfully resists all the attacks for a reasonable amount of time, then it may be used with a reasonable amount of confidence.

References

[1] Diffie, W., and Hellman, M. New directions in cryptography. *IEEE Trans. Inform. Theory IT-22y*, Nov. 1976.

[2] Knuth, D. E. *The Art of Computer Programming, Vol 2: Seminumerical Algorithms*, Addison-Wesley, Reading, Mass., 1969.

[3] Diffie, W., and Hellman, M. Exhaustive cryptanalysis of the NBS data encryption standard. *Computer 10*, (June 1977), 74-84