

Asymmetric Cryptosystems

Brian A. Carter, Ari Kassin, and Tanja Magoc

September 27, 2007

1 Introduction

Throughout the history of cryptography, hundreds of cryptosystems have been developed. The earliest ones, as well as many later ones, relied on the complete secrecy in transferring keys between the sender and recipient. These kinds of systems, called secret key cryptosystem, have just a single key which is used for both encryption and decryption; therefore, these systems are more frequently known as symmetric cryptosystems. In contrast to a symmetric cryptosystem, an asymmetric cryptosystem is associated with a pair of keys; therefore, not everything related to the cryptosystem needs to be kept secret. This kind of a cryptosystem is also called public-key cryptosystem. A public key cryptosystem consists of a public key, which is used for encryption, and a private key, used for decryption. Therefore, anybody can encrypt plaintext since the encryption key is available to everyone, but only the holders of the private key corresponding to the public key used for encryption can decrypt the ciphertext [2].

The first asymmetric cryptosystems appeared in 1970s to address the problem of key management in symmetric cryptography. In 1976, Whitfield Diffie and Martin Hellman published a key management scheme that was the predecessor to the first public-key cryptosystems; it later became known as the Diffie-Hellman Key Exchange Protocol. Later, several asymmetric cryptosystems were developed, including ElGamal, Rabin, and RSA cryptosystems ([2], [1]).

2 Asymmetric Cryptosystems

The public-key cryptosystems rely heavily on mathematical functions known as trapdoor functions; these are easy to apply in one direction but extremely difficult to apply in the inverse, which is required for decryption of the ciphertext [1]. The major difference between the main asymmetric cryptosystems is the function used to produce the public key. Many of them rely on factoring a large number into two large prime numbers that compose the given number. This is not a trivial task, especially when each of the primes generally have more than 100 digits, which is usually the case in well-designed cryptosystems. However, there are other mechanisms used to derive public keys that we will briefly explain in the following sections.

2.1 The Diffie-Hellman Key Exchange Protocol

The Diffie-Hellman Key Exchange Protocol, often shortened to simply Diffie-Hellman, was invented in 1976 through the collaboration of Whitfield Diffie and Martin Hellman ([9], [1]). Diffie-Hellman is not a cryptosystem; rather, it is a means for two parties to jointly produce a shared secret key over an insecure communications channel with minimal risk of a third party obtaining the key. In fact, Diffie-Hellman was the first such method. Diffie-Hellman accomplishes this by taking advantage of the properties of multiplicative groups of integers modulo n —that is, \mathbb{Z}_n^* . Specifically, we select a prime number for n , the modulo value of the group, and we select a prime number as the generator of the group. These two conditions ensure that \mathbb{Z}_n^* will always be a cyclic group—this is the property needed for Diffie-Hellman to work as expected.

To use Diffie-Hellman, Alice and Bob communicate various information over an insecure communications channel. A typical session is described here:

Step 1. Alice and Bob select a large prime number, p , and another prime number, g . The multiplicative group used in the process will be modulo p ; g will be a primitive root mod p (as is the case with any prime numbers).

Step 2. Alice selects a random secret integer a that she keeps privately. Alice calculates $g^a \bmod p$ and transmits the result to Bob.

Step 3. Bob selects a random secret integer b that he keeps privately. Bob calculates $g^b \bmod p$ and transmits the result to Alice.

Step 4. Alice calculates $(g^b \bmod p)^a \bmod p$. This number is the shared secret.

Step 5. Bob calculates $(g^a \bmod p)^b \bmod p$. This number is the shared secret.

The key to Diffie-Hellman, of course, is that Alice and Bob end up with the same shared secret key in steps 4 and 5 of the above algorithm. This shared secret can be used to establish a secure communications channel. Considering this allows us to understand that the Diffie-Hellman Key Exchange Protocol is an excellent tool that can be used to securely establish a secure communications channel when only insecure means of communication are available.

2.2 ElGamal

The ElGamal encryption system was developed by Taher Elgamal in 1984 ([8], [1]). It is based heavily on the same principle as Diffie-Hellman: a cyclic multiplicative group modulo some prime number. The ElGamal algorithm is described as follows:

Step 1. Alice selects a random large prime number p and creates the multiplicative group \mathbb{Z}_p^* . Alice selects a random generator element g of \mathbb{Z}_p^* .

Step 2. Alice selects a random prime number $x \in \mathbb{Z}_{p-1}^*$ —this is her private key.

Step 3. Alice computes her public key $y = g^x \bmod p$.

Step 4. Alice publishes the 3-tuple (p, g, y) as her public key—she makes certain to retain x as her private key.

Step 5. Bob picks an element of \mathbb{Z}_{p-1}^* to represent the plaintext message m he wishes to transmit to Alice. This, consequently, limits the number of possible messages to $p - 1$. Bob denotes the element of \mathbb{Z}_{p-1}^* that represents his plaintext message as k .

Step 6. Bob encrypts the message: he lets $c_1 = g^k \pmod{p}$ and $c_2 = y^k m \pmod{p}$. The pair (c_1, c_2) becomes the ciphertext. Bob transmits the ciphertext to Alice.

Step 7. Alice decrypts the ciphertext: $m = c_2 / c_1^x \pmod{p}$.

Generally speaking, of course, the number of possible messages will far exceed $p - 1$. To remedy this, ElGamal is generally used multiple times on several pieces of the message. One obvious disadvantage of ElGamal, consequently, is that any ciphertext generated using ElGamal is twice as large as the plaintext. Another disadvantage is that $p - 1$ must generally be larger than the number of possible messages, simply because this number is unlikely to be prime.

It is easy to verify that ElGamal is indeed a valid cryptosystem: $c_1^x \equiv (g^k)^x \equiv (g^x)^k \equiv y^k \equiv c_2 / m \pmod{p}$.

2.3 Rabin Cryptosystem

The Rabin cryptosystem is another example of an asymmetric cryptosystem. It was created by Michael O. Rabin in 1979. Rabin's work has theoretical importance because it provided the first provable security for public-key cryptosystems. It can be proven that recovering the entire plaintext from the ciphertext has same complexity as factoring large numbers [1].

Rabin's algorithm is stated as follows:

- Each user chooses two primes p and q each equal to 3 modulo 4, and forms the product $n = p \times q$.
- Public key: the number n .
- Private key: the numbers p and q .
- Encryption: to encrypt a message m , form the ciphertext $c = m^2 \pmod{n}$.
- Decryption: given ciphertext c , use the formulas below to calculate the four square roots modulo n of c : m_1, m_2, m_3 , and m_4 . One of them is the original message m , one square root is $n - m$, and the other two roots are negatives of one another, but otherwise random-looking. Somehow, a decrypting person needs to distinguish the original message from the other three roots (see below).
- In the special case in which both primes when divided by 4 give remainder 3, there are simple formulas to find the four roots. We need to calculate the following:

Step 1. a and b satisfying $a \cdot p + b \cdot q = 1$, which can be done using the extended greatest common divisor algorithm.

Step 2. $r = c \cdot ((p + 1)/4) \pmod{p}$.

Step 3. $s = c \cdot ((q + 1)/4) \pmod{q}$.

Step 4. $x = (a \cdot p \cdot s + b \cdot q \cdot r) \pmod{n}$.

Step 5. $y = (a \cdot p \cdot s - b \cdot q \cdot r) \pmod{n}$.

- Now, the four square roots are $m_1 = x$, $m_2 = -x$, $m_3 = y$, and $m_4 = -y$. In case that m , and hence c , has p or q as a divisor, the formulas will yield only two solutions, each with p or q as a factor. However, for the large primes used in an instance of Rabin, there is a vanishingly small chance of this happening. (Picture the chances that a 512-bit random prime number happens to divide evenly into a message!) [7].

The effectiveness of this algorithm has been questioned mainly because of the fact that it produces 4 results, one correct and 3 false. Identifying the correct one is usually performed with a guess. If the plaintext is intended to represent a text message, guessing is not difficult; however, if the plaintext is intended to represent a numerical value, this issue becomes a problem that must be solved with some kind of disambiguation scheme. It is possible to choose plaintexts with special structures, or to add padding, to eliminate this problem. The most common way of implementing disambiguation is known as Blum disambiguation, which is based on characteristics of pseudorandom number generator [6].

The efficiency of the Rabin cryptosystem is at least as good as that of the RSA cryptosystem. In Rabin, encryption is computed by calculating a square modulo n . This is more efficient than RSA, which requires the calculation of at least a cube. For decryption, the results of congruence are applied through the Chinese remainder theorem, along with two modular exponentiations. Here the efficiency is comparable to RSA. Disambiguation introduces additional computational costs, which is the main reason preventing the Rabin cryptosystem from finding widespread practical use.

The main advantage of the Rabin cryptosystem is that a random plaintext can be recovered successfully from the ciphertext if and only if the codebreaker is capable of efficiently factoring the public key n . Even

though this is a very weak level of security, there exist extended versions of the Rabin cryptosystem achieving stronger levels of security.

It has been proven that decoding the Rabin cryptosystem is equivalent to the integer factorization problem, which is rather different from that of RSA. Thus, the Rabin system is “more secure” in this sense than RSA, and will remain so until a general solution for the factorization problem is discovered, or until the RSA problem is discovered to be equivalent to factorization. (This assumes that the plaintext was not created with a specific structure to ease decoding.)

Since the solution to the factorization problem is being sought on many different fronts, any solution (outside of classified research organizations, such as the NSA) would rapidly become available to the whole scientific community. However, a solution has been long in coming, and the factorization problem has been, thus far, practically unsolvable. Without such an advance, an attacker would have no chance today of breaking the code. This cryptosystem is provably secure against chosen plaintext attacks.

2.4 RSA

RSA is the most commonly used cryptosystem these days. It was invented in 1977 by Ron Rivest, Adi Shamir, and Len Adleman, where the name RSA comes from [4].

The most important part of asymmetric algorithms is the key generation. In RSA, the keys are formed in the following way:

Step 1. Pick two large prime numbers, p and q , of approximately same size such that their product, $n = pq$, is of the required bit length. Most commonly, the required length of n is 1024 bits. However, sometimes 2048 or even 4096 bits are used to give higher security. However, the higher security is usually gained by proper choice of random prime numbers than the longer keys.

Step 2. Calculate n and the Euler’s number, $\phi(n) = (p - 1)(q - 1)$.

Step 3. Choose the public exponent, which is an integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.

Step 4. Compute the secret exponent, which is an integer d such that $1 < d < \phi(n)$ and $ed \equiv 1 \pmod{\phi(n)}$. This is done by applying the Extended Euclidean Algorithm.

Step 5. The pair (n, e) is the public key, and the pair (n, d) is the private key. The values of p , q , and $\phi(n)$ should also be kept secret.

Once the keys are generated, the encrypting party can proceed with enciphering the text. The encryption follows the algorithm below:

Step 1. Encrypting party receives the public key from the decrypting party, (n, e)

Step 2. The message to be encrypted is represented as an integer m .

Step 3. The ciphertext, c , is produced as $c = m^e \pmod{n}$.

Step 4. The ciphertext is sent to the recipient.

Finally, the decrypting party receives the ciphertext and does the following to decipher the message:

Step 1. Calculates the integer message, $m = c^d$.

Step 2. Extract the plaintext message from m .

The security of RSA is based on difficulty of factoring large prime numbers. Therefore, the choice of primes is very crucial, which raises the biggest issue. The problem with sufficiently large primes is to check their primality, to compute $\gcd(e, \phi(n))$ to make sure they are relatively prime, and to compute the private key d satisfying $ed \equiv 1 \pmod{\phi(n)}$. There are several algorithms that can speed up some calculations required for encryption and decryption of a message using the RSA algorithm, which make this cryptosystem very likely to be used these days for transmitting secure information ([4], [5]).

3 Weaknesses in Asymmetric Cryptosystems

Even though asymmetric cryptosystems are relatively secure, there are a few issues that do not guarantee the complete safety and invulnerability of these kinds of cryptosystems. One of the main problems in public-key cryptography is to prove that the public key is authentic, i.e., it has not been replaced with another key by a malicious user. There are two main approaches to address this problem. The first approach relies on a third party, which acts as a certificate authority that certifies the ownership of key. The other approach is the “web of trust” to which a user has access through a password [2].

As any other cryptosystem, the asymmetric one is vulnerable to brute-force attacks. However, with a large enough key, the exhaustive search attack is not practical, and therefore, this type of weakness can be reduced by enlarging the size of the key [2]. Also, many asymmetric cryptosystems rely on random choices for a few numbers do generate the private key. Poor choices of these numbers can lead to easier attack on the system.

Another more serious attack is the side channel attack, which is based on information gained from the physical implementation of a cryptosystem rather than theoretical weaknesses of the algorithm. It is known that a different time is necessary to encrypt a 0 than to encrypt a 1. Thus, a careful time measurement of encryption of ciphertext is a helpful tool when trying to decrypt a ciphertext [2].

Perhaps the most vulnerable attack on an asymmetric cryptosystem is the man-in-the-middle attack, in which a third malicious party intercepts and modifies the public key. This third party has to intercept and modify all the responses between the two communicating parties to give both parties a false sense of security. This attack can be implemented through insecure media such as public Internet. To address the problem, a trusted third party can be used to act as a certificate authority, which ensures the identity of parties using the system. The issue with this approach is that the third party has to be a trusted party, which is always questionable [2].

Another weakness of a public-key cryptosystem is that it takes much more time to apply encryption and decryption keys to plaintext than is the case in symmetric systems. The main reason for this fact is that heavy mathematical calculations are required for use of asymmetric cryptosystems [2].

4 Key Management

Key management is an important part of cryptography since it deals with the generation, distribution, and storage of keys. Only well-designed key management can guarantee the security of a cryptosystem. When dealing with asymmetric cryptosystems, we need to consider the distribution of both public and private keys.

Among techniques to distribute private keys, the most often used ones are key layering and the use of trusted server. Key layering is based on a hierarchy of keys, where each higher level is used to protect the lower levels. The master key is the highest level of hierarchy and is the level that is not cryptographically protected. They can be distributed manually or through a physical or electronic secure device. The next level in hierarchy is the key-encryption key which is used to transport or store other keys. These keys can themselves be protected by another key. The final level is the data key, which is the private key used in an cryptosystem to decipher the ciphertext. The reason for having a hierarchy in key distribution and key storage is to make possible attacks on a cryptosystem more difficult and more time consuming [3].

The other method of distributing private key of an asymmetric cryptosystem is through the key translation center which acts as a trusted server between parties that do not directly share the key between them. Each party shares a different key with the key translation center. The way this sharing information works is that one person sends an encrypted message to the trusted server using its own key, and the center translates it to the message that can be decrypted using the other party’s key. To accomplish this, the key transfer center maintains a secure database of users’ secret keys [3].

Distribution of public keys can be accomplished through different media. One of the ways to distribute a public key is point-to-point delivery over a trusted channel. In this case, the key is delivered directly

by the user or through a direct channel originating at that user, which guarantees the key's authenticity. This method is a good choice if it is not used often or if it used in small closed systems.

Another way to access public keys is through direct access to a trusted party public file, called a public key registry, which contains names and the authentic public key of each system user. Users get keys directly from this registry. To guarantee security against active attackers, a secure channel is required for remote access to the register, which is often done through an authentication tree. An authentication tree is a binary tree which uses a hash function to traverse from requested data to the root of the tree. The root of the tree is the only value that goes under the authentication process. Leaves of the tree are the public keys, and the paths going upward from the leaves all the way to the root are the hash functions applied to the previous nodes to determine the value of the next node. To access a public key, a party needs to know the location of the key and all the hash values on the way to the root. The advantage of authentication trees is the smaller storage requirement, which is visible when a party wants to store more than one public key. In this case, if any technique is used other than an authentication tree, it requires registering each key individually. However, the authentication tree requires only one registration since all public keys associated with the same party can be stored under the same root [3].

Another way to deliver public keys is the use of public key certificates, which consist of a data part and a signature part. The data part contains at least the name of the party (or another string identifying the party) and the public key. It can also contain other information such as a validity period of the key, an algorithm used to create the key, network address of the party, etc. The signature part is the digital signature of a certification authority, which acts as a trusted third party. The certificate authority has its own public key, which is given to every registered user and is a tool for users to access the certified party to obtain requested information [3].

Use of systems implicitly guaranteeing authenticity of public parameters is the final method of distributing public keys. In such systems, including identity-based system, users do not have an explicit public key but rather the public key is constructed from a user's publicly available identity information such as name and/or network or street address, which must uniquely identify the user. The user's key to access the public key is computed from this identity-based public key by applying a private key of a trusted third party. The trusted party calculates the access key and provides it to the user via secure channels. Another example of systems implicitly guaranteeing authenticity is the system using self-certified public keys, which is exactly the same as the identity-based public key with the exception that the manipulations of the public key are done by the individual user rather than the trusted third party [3].

5 Conclusion

Asymmetric cryptosystems have become a popular way of encrypting data since their first appearance in the 1970s. The main reason for development of these types of cryptosystems lies in the security of key management and the vulnerability under malicious attacks. The asymmetric cryptosystem contains a pair of keys, an encryption key, which is available to public, and a decryption key, which is private. Even with a known public key, with a large key space and a large key size, a public-key cryptosystem is usually more secure than symmetric systems. The reason mostly lies in the fact that the public-key cryptosystems are based on mathematical functions that are easily applied in one direction, but very hard to calculate in the inverse. The most commonly used function is the product of two large prime numbers, which is a simple calculation, but going backwards from this large product number to the two primes used to produce it is not an easy task. This adds to the security of the system, which is almost invulnerable to brute-force attacks if correct steps are taken to prevent poor choices for inputs into a public key. On the other hand, the complex modular arithmetic, which is often used in these types of cryptosystems, adds to the time necessary to apply these kinds of algorithms. The biggest concern in using the asymmetric cryptosystems is that the public key can be intercepted and modified by a malicious party during its delivery to the designated party. The most common way of addressing this problem is through the use of the trusted third party, which issues the public key certificates. Even though there are a few weakness

that asymmetric cryptosystems express, they are still the most commonly used in transmitting the secure information.

References

- [1] Mao, W., *Modern Cryptography: Theory & Practice*, Upper Saddle River, NJ: Prentice Hall PTR, 2004.
- [2] “Public-key cryptography,” *Wikipedia*, <http://en.wikipedia.org/wiki/Public-key_cryptography>
- [3] Menezes, A., van Oorschot, P., and Vanstone, S. *Handbook of applied cryptography*, CRC Press, 1996.
- [4] “RSA algorithm,” <http://www.dimgt.com.au/rsa_alg.html>
- [5] “RSA cryptosystem,” <<http://ww3.algorithmdesign.net/handouts/RSA.pdf>>
- [6] “Rabin cryptosystems,” *Wikipedia*, <http://en.wikipedia.org/wiki/Rabin_cryptosystem>
- [7] N. R. Wagner, *The laws of cryptography: Rabin’s version of RSA*, <<http://www.cs.utsa.edu/wagner/laws/Rabin.html>>
- [8] “ElGamal encryption,” *Wikipedia*, <http://en.wikipedia.org/wiki/ElGamal_encryption>
- [9] “Diffie-Hellman key exchange,” *Wikipedia*, <http://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange>