An Survey of Cryptographic schemes

Ilya Hekimi

September 4, 2005

Abstract

This text is a survey of various methods to send secure, private messages. The origins are these methods are their applications will be discussed. They involve many different subjects in number theory and are quite interesting to study.

1 What is Cryptography?

Cryptography is the science of sending messages structured in such a way that only intended recipients are able to understand their content. Some simple methods have been used throughout history to send these kinds of messages. Interest in the subject has dramatically increased with the developement of computer technology and the internet. Massive amounts of sensitive information, such as financial transactions or personal information, are being transmitted through phone lines and optical cable, and keeping them confidential is very important.

Throughout the text, I will present various systems that have been developed to send these disguised message. These systems are usually called *cryptosystems*. I will also assume that the reader has knowledge of modular arithmetic, finite fields and some basic notions of elliptic curves.

2 The Two-way System

2.1 Basic notions

Suppose that two people want to communicate with each other by sending disguised messages. The actual message sent is called the *plaintext* and when it is disguised, it is called the *ciphertext*. Both the plaintext and ciphertext are written is some pre-chosen *alphabet* that consists of N *letters*, or characters. For example, if these two people wish to send written messages, the common English alphabet could be used. Each message is split up into *message units*, which are made from blocks of letters. The set of all possible plaintext and ciphertext messages (which can be the same) is called the *message space*, and is usually denoted P and C, respectively. Transforming a plaintext message unit into a ciphertext one can be seen as a function f from P to C and is called the *encryption*. The inverse function f^{-1} is called the *decryption*, and retrieves the plaintext from the ciphertext. This function is always assumed to have a 1-to-1 correspondence between the 2 message spaces.

When one wants to design a cryptosystem, the most important consideration is to be able to put a label on the elements of plaintext and ciphertext message spaces, such that invertible functions can be easily constructed. The labels most commonly in use are integers in some range. Other ones are also used, such as vectors or points on some curve. For example, if one uses single-letter message units, and an alphabet consisting of all characters that can be represented on a computer, the unicode values of these characters could be used as labels. Of course, the label must be unique for each element of the message space.

2.2 Some Simple Systems

For simplicity in the following example, I will assume that there is single letter message unit using the same N-letter alphabet for both the plaintext and ciphertext. They will be labeled by $0, 1, 2, ..., N-1$. Any function based on these assumptions is a permutation of the set. The easiest way to visualize this is to think of the set as the additive group Z/nZ, and f as some isomorphism on it. The letters m and c will denote the plaintext and ciphertext letters, respectively. If an alphabet consisting of letters $A - Z$ is used, they can be labeled with the elements of $Z/26Z$ in the obvious way. Consider the following function:

$$f(p) = \begin{cases} p+5 & if \quad x < 21 \\ p-5 & if \quad x \geq 23 \end{cases}$$

f is simply adding 5 modulo 26 ($f(m) \equiv m + 5 \bmod 26$). For example, the word 'DOG' is encrypted into 'ITL', by adding 5 to the numerical equivalent of each letter.

This example is easily generalizable into an N-letter alphabet, by fixing an integer b and defining the encryption $c = f(m) \equiv m + b \bmod N$. The system shown above was the case $N = 26$ and $b = 3$.

Now suppose an outsider, who does not know the encryption and decryption, but wants to be able to read the coded messages. He will need a way to extract plaintext messages from the ciphertext. This is called *breaking the code*, and the study of code-breaking is called *cryptanalysis*. Of course, given only a string of numbers, it is impossible to know any information about the message. Two different types of information must be obtained. One information that is vital for breaking a system is understanding its structure, or *nature*. Information such as which message spaces and which general function are used is part of the nature of the system. Another thing is a spefific parameter, or *key*, which will

usually be used to set a particular function. In the previous example, a code-breaker had to know that the group $Z/26Z$ was being used for representing the letters $A - Z$, and that a letter-shift was being used. The key of the system was the number b. It must be assumed that the structure of system is always known, as people do not usually change systems, and this information could be leaked out. Therefore, only the key protects a system from eavesdroppers.

One way to break many systems is to use *frequency analysis*. This works by analyzing long strings of ciphertext, and looking for recurring message units. For example, if someone is using the system explained above, and one knows that 'E' is the most common letter in English, then by looking at the ciphertext and finding the most frequently re-occurring letter, one can identify it as 'E' and extract the secret key. For example, one intercepts the message 'IFMMPUIFSF'. Then, 'F', labeled '5', is the most common letter so it is associated with 'E', labeled '4'. So $b = 1$, and we can know figure out the whole message: 'HELLOTHERE'. In a more complicated system (this one is quickly broken), it could take days to analyze the ciphertext and find the key, so exchanging keys regularly prevents anyone from breaking the code.

It's easy to see how these systems can become extremely complicated and very hard to break. For more examples, consult [1]. Unfortunately, systems like these have many drawbacks. A few problems arise, and will be discussed in the next section.

3 Public Key

3.1 Problems of the two-way system

A cryptosystem is made up of 2 sets, or message spaces, and an invertible function. The sets and the function are usually defined by a choice of parameters, called the key. The function f, used to encrypt messages, will be denoted E, and the function f^{-1}, will be denoted D (It is useful to use a different notation then just f and f^{-1}, as both functions, are kept separate in a public key system. This will be explained later in the section). A few problems arise:

- The key must be sent through physical means. It must be exchanged in person or by a trusted courier. This takes a lot of time compared to the actual encryption and decryption of messages, and it unfeasible if one has to communicate with thousands of people.

- Once the encryption is knows, it is very easy to figure out the decryption and vice-versa. Therefore, if you are corresponding with more then one individual, even if you only give them the encryption function they can figure out how to decrypt and read the messages you send to others. The only way to solve this problem is to use different keys for each individual, which is very time-consuming, as noted above.

- If someone sends you an encrypted message, it is not possible to know if anyone else has gotten his hand on the encryption and it sending you false information. There is no way to prove one's identity.

These problems were always present until 1976, when Diffie and Hellman developed the public key cryptosystem. The idea of such a system is to design a function f which is very hard to invert without knowing some extra information

(i.e. the key). Such a function is called *trapdoor*. The attribute of 'trapdoorness' is not clearly defined from a mathematical standpoint, as advances in computer technology and algorithm design could make such a function easier to invert, making it lose its trapdoor status. It may be possible to prove that such a function is hard to invert, by showing that some lower bound exists on the number of operations to invert it, but no one has ever proved such a theorem on any of the trapdoor functions which are in use. An added difficulty to prove anything is to take into account that someone could compute an arbitrary number of pairs $(m, f(m))$, and get some information on the function somehow. The security of the functions currently known to work well is based on empirical results.

I will now describe how such a system would work. Assume 2 people, Heloise and Abelard, are trying to send each other messages, and that they can each generate a trapdoor function E and its inverse D. We also have the same message space M for the plaintext and ciphertext (for simplicity), and $\forall\ m \in M$, we have $D(E(m)) = p$ and $E(D(m)) = m$. A public key system works in the following way:

1. Both Heloise and Abelard put the encryption function, E_H and E_A, respectively, into a public file accessible to all. They also keep their respective decryption functions, D_H and D_A, private.

2. If Abelard want to send a message m to Heloise, he computes $c = E_H(m)$ and sends it. She in turn computes $m = D_H(c)$ and retrieves the original message.

Suppose someone named Fulbert wants to eavesdrop on Abelard's message. The only things he has access to are E_H, and c. Because E_H is very hard to invert, she is not able to figure out the decryption and therefore cannot read the original message. Another important part of any message is its *signature*. Various meth-

ods have been used to authenticate the sender of any message or letter. Methods such as seals or the traditional signature have been in use for a very long time. In electronic communication, where one cannot have a physical signature, other ways have been developed, such as divulging personal information that an impostor would not be likely to know. In a public key cryptosystem, there is a simple way to authenticate a message. As before, Abelard wants to send a message to Heloise, and therefore sends an encrypted message c, where $c = E_H(m)$. Let l be Abelard's signature, and he can include additional information, such as when the message was sent, etc.. He cannot just send $E_A(l)$ as anyone can send that. So Abelard then transmits $(D_A(l))$ and adds it to the beginning or the end of c. When Heloise receives the message, she first computes $D_H(c + D_A(l))$ and finds out what the message is, except for a small section of gibberish. Since the message it supposed to come from Abelard, she applies E_A (which is public information) to that section and uncovers the signature l. One problem with public key cryptography is that it is often slower to implement then a two-way system, and it may cause time-related problems if one needs to send vast quantities of data. One way to solve this is to use public key to exchange an encryption and decryption function, which can then be used to in the faster, two-way system (a system to do this is explained later in the section).

3.2 The Discrete Logarithm

The discrete logarithm is defined as follows:

Definition: If G is a finite group and $g \in G$, and $g^a = y$ is given by applying the group operation a times on g, then the *discrete logarithm* of y is a.

It it easy to compute g^a by using the repeated squaring method (computing $g, g^2, g^4, ...$ and adding them up), but given y that we know to be of the form g^a,

it is computationally hard to compute a. This is called the discrete logarithm problem and gives rise to a very useful trapdoor function.

One method of finding discrete logarithms is trial multiplication, which is a very simple brute force method. It is obvious why such a method is not feasible when the group G becomes very large. I will describe an algorithm that has exponential running time called Baby Step, Giant Step and was developed by Shanks. It is one of the fastest methods that can be used in arbitrary groups.

Baby Step, Giant Step

Input:A cyclic group G of order n, with a generator g and an element y. The idea of this algorithm is that if we know that $g^x = y$, we want to rewrite x as $im + j$, keeping m constant and changing i, j. So we get $y * g^{mi} = g^j$. We write down a table with g^j's and cycle the i's until we find a match.

Output:A value x such that $y = g^x$.

1. Choose $m > \sqrt{n}$
2. For all j such that $0 \leq j \leq m$, compute g^j and store the pair (g^j, j) in a table.
3. Compute a^{-m}
4. Set $y = g$
5. For $i = 0$ to $i = m - 1$
   (a) Check to see if $y = g^j$ for some j in the table.
   (b) If so, return $im + j$
   (c) If not set $y = y * a^{-m}$

It is possible to search through the table in constant time, making the algorithm have a running time of $O(e^{\log n})$.

There are other algorithms for finding discrete logarithms, the most notable one being the Index-Calculus algorithm. It is important because it has subexponential running time $O(e^{\log n \log\log n})$, but only works on finite fields. Most of the original cryptosystem based on discrete logarithms used finite fields, making them use larger keys (i.e. increase the size of the group) to get the same security, making the actual encryption and decryption process slower. This is why elliptic curve variants of these algorithms were introduced, as the Index-Calculus algorithm cannot be applied to them. I will not describe the algorithm, but for more information consult [1], or any other book on cryptography.

3.3 RSA

This first public key cryptosystem to be introduced was invented by Rivest, Shamir and Adleman, and is called the 'RSA' cryptosystem. This system is based on the difficulty of factoring. I will describe the procedure for a single person to generate a set of functions E and D, and then receive message from others.

First of all, one must choose two very large primes p and q, and then set $n = pq$. How does one generate random prime numbers? One possibility is to first generate a random number q. If q is even, replace it by $q + 1$. Then apply primality tests to see if the number is a suitable prime (since most primality tests are probabilistic). If not, replace q by $q+2$ and repeat until you find a prime. By the Prime Number Theorem, one will find a prime after testing $O(\log m)$ numbers. There is a lot of literature on random number generation and primality testing.

With this information, one can now compute $\phi(n) = (p-1)(q-1)$, where

/*phi* is the Euler totient function. Now, a number e must be selected such that $1 \leq e \leq (\phi(n))$ and $\gcd(e, \phi(n))$. Then, one computes the multiplicative inverse $d = e^{-1} \bmod \phi(n)$, using the Extended Euclidean Algorithm. The encryption key is the pair (n, e) and the decryption key is the pair (n, d).

Encryption: Given a plaintext message $m \in M$, compute $c = m^e \bmod n$.

Decryption: Given c, compute $c^d \bmod n$, giving us $m^e d \equiv m^{l\phi(n)} m^1,\ l \in Z$, which is equal to $m \bmod n$ by Euler's Theorem and therefore recovers the original message.

The security of the system is based on the difficulty of factoring. Obviously, if one can factor n, one can break RSA. Conversely, breaking RSA gives rise to a probabilistic algorithm to factor n (described in [3]). Therefore, inverting the encryption is computationally hard. Unfortunately, there is no way to predict what information someone can get by analyzing any intercepted ciphertext. As I mentioned before, none of the public key systems are provably secure, but most are conjectured to be and experimental evidence also suggests so.

3.4 Other public key cryptosystems

Diffie-Hellman key exchange

The idea of this cryptosystem is to use use a random element in the field F_q as a key for a two-way cryptosystem. Again, we have Heloise and Abelard that want to communicate the secret key to each other. The method is as follows:

1. Heloise and Abelard both agree on a generator $g \in {F_q}^*$.
2. Heloise chooses a secret integer a between 1 and $q - 1$, and computes g^a and sends it to Abelard.

3. Abelard chooses a secret integer b between 1 and $q-1$, and computres g^b and sends it to Heloise.

4. Both Heloise and Abelard can now compute g^{ab}, and they use that as their secret key

An outsider wanting to break the code needs to be able to determine the key g^{ab}, but has only g, g^a, g^b. The security of this system is assumed by the *Diffie-Hellman assumption*, which states that it is computationally infeasable to computer g^{ab} knowing only g^a, g^b. Obviously, if one can compute discrete logarithms, then the assumptions fail. Conversely, it is conjectured that there is no possible way compute g^{ab} from g^a, g^b without first computing discrete logarithms, but not proven.

ElGamal Cryptosystem

For this cryptosystem, a large field F_q, must first be selected, and an element $g \in F_q^*$. If we keep to our usual set-up, Heloise chooses an element a between 0 and $q-1$. She publishes g^a as her encryption key. Abelard wants to send her a message m which we assume is an element of F_q. The system is as follows:

Encryption: Abelard chooses an integer k at random and computes (g^k, mg^{ak}). Abelard can compute g^{ak} by raising g^a to the k-th power.

Decryption: Heloise receives the pair of numbers and computes g^{ak}, and can now retrieve m by dividing the second element by it.

Again, if one can solve discrete logarithms, one can break the system. The converse also holds by the Diffie-Hellman assumption, as one needs to compute g^{ak} knowing only g^k, g^a to break the code.

3.5 Elliptic Curve Cryptography

It is assumed that the reader knows the group law on the points of an elliptic curve over a field. For simplicity, the elliptic curves will always be in Weierstrass form ($y^2 = x^3 + ax + b$) over F_q, and we will not be in fields of characteristic 2.

The first important thing is to be able to represent messages as points on the elliptic curve. We need to be able to generate points that are related to the message m in some way. One possible probabilistic way of doing this is the following:

1. It is assumed that $m \leq M$ and that $q > M\kappa$.

2. Choose a fairly large integer κ, such that we want the probability of failure to be $1/2^{\kappa}$.

3. Write down the integers from 1 to $M\kappa$ in the form $m\kappa + j$ where $q \leq j \leq \kappa$ and set up a 1-to-1 correspondence between these integers and elements of F_q. One way to do this is to write these integers as r-digit numbers to the base p, and then take the r digits (as elements of Z/pZ) to be the coefficients of an r-1 degree polynomial corresponding to an element of F_q (by considering the polynomial in the ring $Z/pz[X]$ mod an irreducible polynomial or r-degree).

4. Given m, for each $1 \leq j \leq \kappa$,we obtain an element $x \in F_q$ corresponding to $m\kappa+j$. We then compute the right-side of the equation $y^2 = x^3+ax+b$ and try to find a square root in F_q (there are probabilistic algorithms for finding square roots, one can be found in [1]). Once a root is found for a particular j, we use the point $P_m = (x, y)$ as a representative for our message m.

5. To retrive m from P_m, we just have to compute $\lfloor x - 1/\kappa \rfloor$. Since an element $x^3 + ax + b$ will be a square root $1/2$ of the time, there is only a $1/2^\kappa$ chance that this method will fail.

Now that we have a method for representing messages as points, I will describe another public key cryptosystem. One must note that the previous public key systems based on discrete logarithms in F_p can be generalized to arbitrary groups, and one can use the points on an elliptic curve to represent such groups. Here, the notation aP will represent adding P to itself a times (applying the group law a times).

Analog of Massey-Omura using Elliptic Curves

Assume there is an elliptic curve E over F_q, and that the number N of points on the curve has been computed and is publicly known. Again, Abelard wants to send a message m, represented as a point P_m, to Heloise.

1. Each user (Heloise and Abelard) secretly choose a random integer e between 1 and N such that $\gcd(e, N) = 1$ and using the Extended Euclidean Algorithm, compute its inverse $d \equiv e^{-1}$ mod N.

2. Abelard sends the message $e_a P_m$.

3. This means nothing to Heloise, who does not know e, so she computes $e_h e_a P_m$.

4. Abelard receives this and uncovers part of the message by computing $d_a e_h e_a P_m = e_h P_m$, and sends it back to Heloise.

5. Heloise can now compute $d_h e_h P_m = P_m$ and recover the original message.

Someone wanting to break the code only has access to $e_h P_m, e_a e_h P_m, e_a P_m$, and is not able to extract information, again by the Diffie-Hellman assumption.

4 Probabilistic Public Key

As mentioned above, many of the public key cryptosystems are not *provably* secure. For example, in the Diffie-Hellman Key exchange, we know that breaking the system is not harder then solving discrete logarithms, but we have no idea if a lower bound exists. Also, it is hard to prove security, as information could be retrieved from the ciphertext, by frequency analysis or any other method. The usefullness of these systems are based on conjectures and experimental data (no one has been able to break them).

4.1 The first cryptosystem

Probabilistic public key cryptosystems were developed because of the restriction of regular public key systems on proving security. These systems propose a cryptosystem with the following scheme:

Whatever is efficiently computable about the plaintext given the ciphertext, is also efficiently computable without the ciphertext.

The first such system was developed by S.Goldwasser and S.Micali, and is based on the difficulty of extracting quadratic residues modulo some composite n. This cryptosystem relies on the notion of *Unapproximable Trapdoor Predicates*. The idea is that one develops a function Q that is hard to compute without knowing some additional information, but given y, it is easy to find an element x such that $Q(x) = y$. I will not define exactly what Unapproximable Trapdoor predicate are, nor will I show the proof of the security of the system as it is a rather lengthy proof, but I direct the reader to [5] for more information.

A probabilistic public key cryptosystem based on quadratic residues

We first choose 2 primes p and q and set $n = pq$. When they choose a y, such that y is a non-residue modulo n and $\left(\frac{y}{n}\right) = 1$, where $\left(\right)$ is the Jacobi symbol. The pair (n, y) is made public and (p, q) kept secret. We also have a function Q such that

$$Q(x) = \begin{cases} 1 & if\ x\ is\ a\ quadratic\ residue\ mod\ \text{n} \\ 0 & if\ x\ is\ a\ quadratic\ non-residue\ mod\ \text{n} \end{cases}$$

Encryption: Suppose Abelard wants to send a message m to Heloise. We first represent $m = m_1 m_2 m_3 ... m_l$ as a binary string.

1. For $i = 1$ to $i = l$
   (a) Pick $x \in (Z/nZ)^*$ at random
   (b) If $m_i = 1$, set $e_i = yx^2 \bmod n$
   (c) else set $e_i = x^2 \bmod n$
2. Send the l-tuple $(e_1, e_2, ..., e_l)$

Decryption: We receive an l-tuple of numbers $(e_1, e_2, ..., e_l)$.

1. For $i = 1$ to $i = l$
   (a) Set $m_i = Q(e_i)$
2. Set $m = m_1 m_2 ... m_l$

4.2 Other Systems

There are other types of probabilistic cryptosystems that do not do a bit-by-bit encryption, but rather are able to encrypt whole messages. One such system is one developed by Paillier (see[6]) and based on composite residuosity classes. This system is quite famous, but unfortunately not implementable in practice,

as it posseses very long encryption times. They have been some modifications by various people, notably using elliptic curves (see [7]).

5 Conclusion

Cryptography is a very interesting field of study as it invovles many different notions from number theory and has many useful applications. Both code-breaking and devising new systems are intresting from a mathematical standpoint. The subject will no doubt become even more interesting if some technological breakthroughs happen, such as quantum computers (who are able to factor in polynomial time). I hope to have increased the reader's interest in the subject with this text.

6 References

1. Koblitz, Neal. A course in number theory and cryptography. Second edition. Graduate Texts in Mathematics, 114. Springer-Verlag, New York, 1994

2. Cohen, H. A Course in Computational Algebraic Number Theory, volume 138 of Graduate Texts in Mathematics. Springer-Verlag, 1993.

3. L. M. Aldeman, R. L. Rivest and A. Shamir, "A method for obtaining digital signatures and public key cryptosystems," *Communications of the ACM*, **21**, (1978), 120-126

4. T. ElGamal "A public key cryptosystem and a signature scheme based on discrete logarithms,"*IEEE Transactions on Information Theory IT-31*, (1985), 469-472

5. S.Goldwasser and S.Micali, *Probabilistic Encryption*, JCSS Vol.28 No 2, (1984), 270-299.

6. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. *EUROCRYPT99, LNCS 1592*,(1999), 223238.

7. S.Galbraith, Elliptic curve Paillier schemes, *Journal of Cryptology*, Vol. 15, No. 2 (2002) 129–138.