Introduction to Cryptography for the Mathematically Challenged

Leo Liberti[*]
Centre for Process Systems Engineering
Imperial College of Science, Technology and Medicine

October 31, 2000

1 Introduction

The need for cryptography is historically linked to the business of spying and warmaking. If an illegal agent has to communicate with his embassy, he must do it in such a way that even though the message might fall into the police hands, he and the embassy staff must not be compromised. During any war, if two regiments have to communicate in order to synchronize an attack on enemy lines, they must make sure their message is not intercepted by the enemy; and if it is, at least that it is not understood. Since the beginning of the industrial revolution and capitalistic society, there is also one other widespread application of cryptography: that of protecting corporate secrets like manufacturing process descriptions or private business transactions. Even more recently, with the advent of a technologically advanced information infrastructure, sending messages has become very easy, but so has interception. Cryptography is nowadays needed by most everyone who uses electronic and digital communications.

The literal meaning of "Cryptography" is "unintelligible writing" — this is quite different from "hidden writing" ("steganography"). Cryptography is not about preventing the enemy from intercepting a message, but rather about preventing the enemy from being able to read the message. Thus, devices like invisible ink, tattoing the message on the scalp or any other form of information smuggling are not in the realm of cryptography.

In the field of cryptography, any message that is clearly readable by both ally and enemy is called a "clear text message". After the message has undergone the process of encryption it is called "encrypted message". In what follows, when we talk about "clear text message" we will always consider a message written in the english language with the 26-letters English alphabet (A B C D E F G H I J K L M N O P Q R S T U V W X Y Z), the ten arabic digits (0 1 2 3 4 5 6 7 8 9) and one punctuation sign – the word separator, the blank (or space)[1]. For simplicity we will suppress all other punctuation signs.

Having said that, any message which does not use this alphabet or this language is unintelligible and hence encrypted, but this is not considered cryptography. Hiring two australian Aborigines as radiotelegraphists for two armies that need to communicate could probably be very effective in terms of hiding information to the enemy, but only if the enemy itself was not in the position of hiring australian Aborigines radiotelegraphists. Cryptographic processes tend to be "general" – that is, to hold in every similar situation.

2 Letter Substitution

The cryptographic scheme that is easiest to implement (and to break) is at the level of letters: we change the order of the letters of our alphabet (that is called a "permutation" of the alphabet) and then we map A to the first letter in the new ordering, B to the second letter in the new ordering, and so on. Since in this kind of encryption the process is always substituting a letter with another letter, it is also called "letter substitution".

Since there are 26 letters, ten digits and one punctuation sign in our alphabet - that is, 37 symbols, we can permute them in any of $37! - 1$ ways (that is read "thirtyseven factorial minus one" and it means $1 \times 2 \times 3 \times \cdots \times 36 \times 37 - 1$, that is 13763753091226345046315979581580902399999999 ways). Why this number? Let's take it one step at

[*]`l.liberti@ic.ac.uk`

[1]Usually denoted with the symbol "␣".

a time. if we had only one letter in our alphabet, there would only be one possible permutation: the only letter is mapped to itself. This permutation is called “identity”. If we had two letters in our alphabet (say A and B), we would have either the identity (A and B remain the same) or one other possible permutation (A becomes B and B becomes A). If our clear text message is “ABBA”, the encrypted message using the identity is still “ABBA”, whereas using the other permutation gathers “BAAB”. It should now become clear why in counting letter permutations used for encryption we always have to subtract one from the total number: we never want to use the identity permutation. If we had three letters in our alphabet, we have a total of six permutations:

$$\begin{array}{lll} A \to A & B \to B & C \to C \\ A \to B & B \to A & C \to C \\ A \to C & B \to B & C \to A \\ A \to A & B \to C & C \to B \\ A \to B & B \to C & C \to A \\ A \to C & B \to B & C \to A \end{array}$$

With four letter, we have 24 possible permutations, and with 5 letter 120 permutations. Notice $2 = 1 \times 2$, $6 = 1 \times 2 \times 3$, $24 = 1 \times 2 \times 3 \times 4$ and $120 = 1 \times 2 \times 3 \times 4 \times 5$. In short, with n letters we would have $1 \times 2 \times \cdots \times (n-1) \times n$. In mathematical notation, the multiplication of all the numbers up to (and including) n is indicated with $n!$ and is called “n factorial”. It is the number of different permutations of n distinct objects.

By way of an example of this kind of encryption, we take our alphabet,

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_
```

we “shift” it to the left by one position putting the leftmost character (A) at the end, and we write the permutation:

```
BCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_A
```

We thus obtain the mapping of letters: $A \to B$, $B \to C$ and so on. If the clear text message is

```
CAESAR IS ATTACKING
```

the encrypted message is

```
DBFTBSAJTABUUBDLJOH
```

This particular type of permutation is called a “cyclic” permutation because it just shifts the whole alphabet but it doesn’t change the relative order of the letters, save for the first and last. In fact cyclic permutations are extremely easy to implement and they were used by the Romans in their conquest wars.

The mapping between the alphabet and the permutation which permits the encryption of the message is called the “encryption key”. More generally, whatever permits the encryption of a message is an encryption key, and whatever permits the decryption of a message (that is, the process that transforms an encrypted message back to the clear text message) is called the “decryption key”. In the case of letter cryptography these keys are the same: one needs the permutation in order to encrypt a message (as we have already seen); and to decrypt it, all one does is reading the mapping backwards. So the same information used for encryption is also used for decryption.

Since the point of encrypting a message is making sure it is not intelligible by anybody save those people who possess the decryption key, it is crucial that there is no way of obtaining the key by unauthorised means (this is called “breaking a key”); or else, that it takes the enemy a longer time to break the key than it takes the allies to change the old key with a new one. In the case of letter cryptography by substitution it is actually quite easy to break any key in a very short time indeed. Substitution keys are usually broken by frequency analysis. This is based on the fact that the relative frequencies of the mapped letters does not change. Mapping the word “attacking” with the key given above, we got the word “buubdljoh”, and we can observe that the repetition of “a” and “t” in the clear text word has been transformed into a repetition of “b” and “u”, but the frequency structure of the encrypted word has not changed. All the enemy must do in order to perform frequency analysis is to compile frequency tables in which to every letter in the alphabet there corresponds the average frequency of appearance of that letter in the English language (or the language used to write the clear text message). Thereafter, the enemy only has to compute the frequency with which each letter appears in the encrypted message to find out what the encryption key is. With modern calculators working on average length messages, this takes a fraction of a second. And the longer the message, the more accurate the frequency tables turn out to be (length of message is not an issue: even it the encrypted text is forty billion letters long, the enemy need only perform frequency analysis on the first 500 or 1000 letters to retrieve the key; and once the key is known, the whole message can be decrypted very quickly).

The way to defy frequency analysis is to change the permutation used to encrypt the message for each letter in the clear text message. This is called “one time pad” cryptography because when it was first introduced after World War I each of the allies had to have a copy of one-time pads booklet. Each of the pads contained a key which was totally random and long enough to encrypt simple messages. By way of an example, if we have three different permutations:

```
BCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_A
CIBEF05DJKAM_OTQRSGLVWUXZH1234G6789NY
ACEGIKMOQSUWY0123987456_BDFHJLNPRTVXZ
```

we can safely encrypt (that is, safely from frequency analysis) clear text messages of three letters: the message `MUM` would become `N_Y`. As we can see, the frequency of the letter M (two thirds of the message) is not kept constant in the encrypted message; hence no frequency analysis can be performed. The trouble with one-time pad cryptography is that the key to be transmitted to the allies is three times as long as with simple permutation methods, and that is just to encrypt/decrypt three letters words! In general, in order to encrypt an n-letter message written in an m-letter alphabet in this fashion one needs to use keys which are $n \times m$ letters long! This is excessive: broadcasting keys needs a secure communication channel (if the key is intercepted the all messages can be understood), but the lack of secure channels is exactly the reason why cryptography is used. One-time pad cryptography is absolutely secure against frequency analysis but it is, in practice, very difficult to implement.

3 Public Key Cryptography

In order to solve the problem of the difficulty of broadcasting long keys, mathematicians have worked on a way to separate encryption from decryption, thus having two separate keys with the following crucial property: it must be practically impossible to obtain the decryption key by knowing the encryption key. If the allies possess such a method, the way to go about it is this. Each ally uses this technique to build an encryption key and a decryption key. Each ally then sends its encryption key to all other allies by using whatever channel is available: it is assumed that the enemy can intercept all the encryption keys. If an ally wants to send a message to another ally, it encrypts the message using the recipient’s (public) encryption key, which it can do since the encryption keys are public. Since only the recipient will have the (private) decryption key needed to decrypt the message, nobody else but the intended recipient will be able to read the message.

The difficult part, then, is how to produce a pair of keys (encryption/decryption) where the decryption key cannot be calculated just by knowing the encryption key. The mathematical operation involved to build such a pair must be a reversible operation (we need both keys) but one direction must be reasonably easy whereas the opposite direction must be extremely difficult. Such an operation can be provided by integer multiplication and its reverse, prime factorization. A “prime number” is an integer number that cannot be divided precisely by any other number save 1 and itself. Prime numbers are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 and so on. 21 is not prime because we can write $21 = 3 \times 7$. Multiplying two prime numbers is very fast even when the primes are very large. The converse, that is, given a number, finding two primes such that their product is the given number, is extremely difficult. For primes composed of 1000 digits or so, even using the fastest computer on earth could not calculate two primes given their product before a century of continuous computing.

Now we need an encryption scheme based on a product of primes and a decryption scheme based on the two prime factors. Such schemes have been first proposed in the ’70s by three researchers: Riverst, Shamir and Adleman. Taking after their surname initials, the encryption/decryption algorithms have been named “RSA”.

To understand how the RSA encryption scheme works, one has to understand the basics of number theory and modular arithmetic. Apart from the notion of prime number, explained above, there is also a weaker notion of two numbers being “coprime”. This means that the largest integer dividing exactly both numbers is 1. Let us now see some modular arithmetic: given two integers n, m the notation $n \mod m$ (read “n modulo m”) is the remainder of the integer division of n by m. For example, if $m = 5$ we have $0 \mod 5$, $1 \mod 5 = 1$, $2 \mod 5 = 2$, $3 \mod 5 = 3$ and $4 \mod 5 = 4$ because every number smaller than 5 gives itself as a remainder on division by 5. We then have $5 \mod 5 = 0$ as $\frac{5}{5} = 1$ with remainder of 0. The following table illustrates the results of the operation $\mod 5$ for a few more values.

0	1	2	3	4	5	6	7	8	9	10	11
0	1	2	3	4	0	1	2	3	4	0	1

Computing "modulo n", loosely speaking, is akin to making a normal computation, then taking the remainder of the division of the result by n. For example, $5 \times 4 + 3 \mod 7$ is equal to $23 \mod 7$ which is equal to 2 (because $23 = 3 * 7 + 2$). In fact modular arithmetic is much faster to perform than normal arithmetic because usually it is possible to reduce intermediate results. For example, instead of computing $5 \times 4 + 3$ and then reducing the result modulo 7, we could have computed $5 \times 4 = 20$, then reduced $20 \mod 7 = 6$, then computed $6 + 3 = 9$ and finally $9 \mod 7 = 2$. When large numbers are involved, this speeds up computations enormously. This is especially true when computing powers. For example, the number $1027^4 5907$ is unmanageably large, but if we are only interested in that number modulo 5, we can always reduce all intermediate results modulo 5 (so that in each new multiplication the operands are necessarily less than 5 — that is, quite small). We only need one more theoretical notion in order to understand RSA. That of a modular inverse. Let m and n be integers. Then $m^{-1} \mod n$ is the number t such that $t \times m \mod n$ is equal to 1. For example, $2^{-1} \mod 5 = 3$ because $2 * 3 \mod 5 = 6 \mod 5 = 1$.

Let p and q two large primes (as large as it is needed to make the encryption secure), let $N = p \times q$ be their product, let $n = (p-1)(q-1)$ and let e be any number coprime to n. The two numbers (e, N) form the public key. This is how the encryption process works: first we transform the clear text message in a number M (this can be done in a number of ways: the most natural — but not necessarily the most effective — is to assign a number to each letter of the alphabet, for example: A=10, B=11, C=12 and so on. The message "CAB" is then transformed in the number 121011), making sure that M is coprime to N (if it is not, it is always possible to add a few spurious letters to the clear text message in order that its numeric encoding M is coprime to N). We then compute $E = M^e \mod N$: E is the encrypted message. There are two more requirements: that M^e has to be larger[2] than N, and that M has to be smaller than N. If M^e turns out to be smaller than N, it can be multiplied by a fixed constant (remembering that the decryption process needs to be modified to take this into account). If M turns out to be bigger than N, it can be split in chunks and each of the chunk can be encrypted separately.

Now let us see the decryption process. The decryption key is given by $d = e^{-1} \mod n$ (recall $n = (p-1)(q-1)$). Of course to calculate d one needs to know what n is, or equivalently, what the numbers p and q are. In this sense the private decryption key is based on the large primes p and q, and knowing the public key (e, N) does not help in calculating p and q unless one can factor N into p and q (but if p and q are large enough, this is a very difficult task). The ally which receives the message E (encrypted using the ally's own public key) can decrypt the message using its own private key d by calculating $D = E^d \mod N$. It is quite easy to show that $D = M$, and we include a sketch of the proof in a footnote[3], to be skipped by the uninitiated.

Let us see an example of RSA encryption and decryption process. We shall use very small primes for simplicity, although this means that the encryption is not secure at all. Let $p = 11$ and $q = 7$. We have $N = 77$ and $n = (11-1)(7-1) = 60$. The encryption key e must be coprime to n, and we choose $e = 7$ (because the largest integer dividing exactly both 5 and 24 is 1). The decryption key d is given by $7^{-1} \mod 60$, that is, 43 (because $7 \times 43 \mod 60 = 301 \mod 60 = 1$). Suppose the ally A has to send a "yes" answer to a question another ally B had sent to it previously. The ally A wants to send B a "Y" in encrypted form. According to the mapping A=10, B=11, etcetera, Y is mapped to the number $M = 35$. To encode M, the ally A looks up the public key of ally B $(7, 77)$ and calculates $35^7 \mod 77 = 7$, thus the encrypted message

[2]If this were not the case, we would have a situation $m \mod n$ where $m < n$ and we have seen with the example modulo 5 above that in this case $m \mod n$ is just m; that is, the encrypted message would be the same as the clear text message, and we clearly do not want that.

[3]Sketch of proof that $D = M$.

$$\begin{aligned}
D &= E^d \mod N \\
&= (M^e \mod N)^d \mod N \\
&= M^{ed} \mod N \\
&= M^{e(e^{-1} \mod n)} \mod N \\
&= M^{1 \mod n} \mod N \\
&= M^{1+kn} \mod N \\
&= M(M^n)^k \mod N \\
&= M(M^n \mod N)^k \mod N \\
&= M \times 1^k \mod N \\
&= M \mod N \\
&= M
\end{aligned}$$

QED.

is 7. Ally B, on receiving 7, sets out to decript it with its private key: all it has to do is calculating $7^43 \mod 77$, which is again 35. The same numeric code is used to transform 35 into the letter "Y".

RSA has been considered the most important breakthrough in cryptography because it dispenses for the need of a secure channel to communicate keys. Public key cryptography can be generalized to provide secure electronic signatures in a straightforward way: suppose allies B wants to be sure that the message really comes from ally A. After all anybody could have looked up the public key of ally B and encrypted a forged message. This is how it goes: ally A first encrypts the message using B's public key, and then it encrypts it again using its own private key (notice that encryption and decryption are just modular operations so a private decryption key can be used to encrypt a message as well as decrypt it). When B receives the doubly-encrypted message, it first decrypts it using A's public key: thus B is sure that it was really A sending the message, for nobody else could have had the private key corresponding to A's public key; and then it decrypts the outcome a second time using its own private key to retrieve the clear text message.

It has been said that RSA would be the ultimate cryptographic challenge because it offers scalable security. Whenever computers get so powerful that integers with thousands of digits can be factorized in a few seconds, allies can just pick primes p and q with millions of digits, and so on — the important thing is that multiplying p by q takes an amount of time proportional to the length in digits of p and q, whereas factoring their product without knowing p or q takes an amount of time proportional to an exponential of the length in digits. This means that the two orders of magnitudes are not even comparable. As long as nobody finds a method for factoring an integer which takes an amount of time proportional to the length in digits of the integer, RSA is theoretically secure.

4 Quantum Cryptanalysis

As it turns out, physicists have already designed a computer which is capable of factoring an integer quickly, i.e. taking a time proportional to the length in digits. The only thing is, they can't build it yet. So for the time being RSA is really the most secure cryptographic technology around. But this may become false in the future. The marvellous computer capable of performing such a task is called a "quantum computer" because it is based on a property of quantum mechanics which is unique. There is no corresponding feature in the classical model of physics. The theory of quantum mechanics says that energy and matter in the universe come in chunks, rather than in continuous quantities. From this postulate stem quite a lot of truly counterintuitive notions which are nonetheless thought to be true, like, for example, the superposition principle. Until an observable entity is not observed, it is in fact in a "superposition" of possible states. Only when a measurement is taken (that is, when the observable entity is actually observed) the observable entity collapses in the observed state. For example: an electron has an observable called "spin". The spin can be either "up" or "down". Until the moment we observe the spin we don't know whether the spin is up or down: we have 50% chance of observing the spin being up and 50% chance of the spin being down. In fact, quantum mechanics say that this is the physical truth: the spin is in a state $\frac{1}{\sqrt{2}}(|up\rangle + |down\rangle)$. Some explanations about the notation: $|state\rangle$ just indicates the state the quantum entity is in. The sum means "superposition". We multiply both terms by $\frac{1}{\sqrt{2}}$ because the probability of seeing a state is the square of the coefficient of that state. The coefficient of a state in a superposition is called the amplitude. In our example, since the probability of seeing each state is 0.5, we look for the number x such that $x^2 = 0.5$, that is, $\frac{1}{\sqrt{2}}$.

The reason why this feature of quantum mechanics (superposition) can help build a quantum computer capable of factoring integers quickly is that it is in theory possible to build a device that changes the amplitude of a state without actually observing the observable entity. In short, superpositions can be manipolated to a certain extent without causing the collapse of the superposition of states on a definite state. The algorithm for quantum factorization was invented by Peter Shor in 1995. It is based on a theorem of number theory that states that one can factor an integer of N by finding the period of the function $f(x) = a^x \mod N$, where a is any integer coprime to N. Any function which has the same set of values repeated in the same order over and over again as its variable varies is called a "periodic function", and the length of the sequence of values before the first repetition is called the period. For example, if $N = 6$ and $a = 2$ we see that the function $g(x) = 2^x \mod 15$ has the values:

0	1	2	3	4	5	6	7	8	9	...
1	2	4	8	1	2	4	8	1	2	...

We can see that the period of g is 4 as its values are 1,2,4,8 repeated in this order indefinitely. This theorem does not matter much if classical computation (as opposed to quantum computation) is taken into account, because finding the period of a function like $f(x)$ above is just as hard a problem as factoring N directly. But seeing the factorization problem as a period-finding problem makes it easier for a quantum computer to tackle it. A "quantum register" is a device that can hold a number of quantum states. For example, the spin of the electron is a register that can hold two values. The quantum computer that calculates the period of arbitrary functions is built around two registers X and Y. The registers are big enough to hold all the values that the variable x can take (x can take values from 0 to N). X Initially the quantum computer is in the state $|0\rangle|0\rangle$, i.e. each register holds the value 0.

1. Initialize the X register with an equiprobable superposition of all values the variable x can take:
$$\frac{1}{\sqrt{n}}\sum_{x=0}^{N}|x\rangle|0\rangle$$

2. Initialize the Y register with all the values of $f(x)$:
$$\frac{1}{\sqrt{n}}\sum_{x=0}^{N}|x\rangle|f(x)\rangle$$

3. Observe the second register, and suppose the value f_0 is obtained: since f is periodic, there are many values of x such that $f(x) = f_0$. This triggers a change in the amplitude coefficients of states so that all x's such that $f(x) \neq f_0$ have their coefficients set to 0, whilst all other amplitude coefficients are emphasized:
$$\left(\sum_{x:f(x)=f_0}\alpha_x|x\rangle\right)|f_0\rangle$$
where (α_x are amplitude coefficients such that their squares — the probabilities associated with the quantum states — add up to 1).

4. From now on we are only interested in the first register, so we can ignore the second register. The first register now holds all values of x such that $f(x) = f_0$. Their associated amplitudes are roughly the same — we can assume for simplicity they are the same: suppose we have k such x's, then:
$$\frac{1}{\sqrt{k}}\sum_{x:f(x)=f_0}|x\rangle$$

5. We apply a special quantum operator which measures the distance between two consecutive x's — i.e. the period of f — and gets the right answer with over 50% probability.

This allows us to calculate the period of f: the algorithm just needs to be run a sufficient number of times to ensure a sufficient success confidence.

The reason why quantum computers can't be built yet is that it is extremely difficult to keep a physical quantum register stable for a period of time long enough to perform a similar calculation.

5 Quantum Cryptography

Just when it seems that all the enemy needs is a quantum computer in order to be able to break allies' encryption schemes, it turns out that allies can actually use quantum technology at their own advantage. It is possible to build a quantum-secured communication line which not only offers the total security of the one-time pad cryptography, but it also detects if the line is being tapped. If it is, both the enemy and the recipient ally only receive garbage. The technical details of this kind of quantum cryptography go beyond the scope of these notes. However, quantum cryptography seems to have a time advantage over quantum cryptanalys, as in 1988 an experiment was carried out where the first quantum-encrypted communication line was designed and implemented by Charles Bennett. In 1995, researchers at the University of Geneva succeeded in implementing quantum cryptography in an optic fibre that stretched 23 km from Geneva to Nyon.