MAPLE IMPLEMENTATION OF CERTAIN CRYPTOGRAPHIC METHODS

Manmohan Kaur
Benedictine University
5700 College Road
Lisle IL-60532
mkaur@ben.edu

Cryptology is an effective tool to spark an undergraduate student's interest in mathematics. Technology allows them to see encryption and decryption in action, thus adding to the fascination. We introduce some public key cryptographic methods based on the Discrete Logarithm Problem, and present their implementation in the computer algebra system Maple.

Introduction

Cryptology, the science of sending and receiving secret messages, is at the intersection of mathematics and computer science, and encompasses every aspect of modern life – online financial transactions, digital signatures, cloud computing; the list goes on. Modern cryptology relies almost entirely on mathematics for its integrity. Today's cryptosystems convert messages into numbers and perform computations to encrypt those numbers. The crucial element of this method is the use of operations that cannot be undone without infeasible, time-consuming computation, unless some extra information is known to the receiving party. Some of the commonly used cryptographic methods are easily accessible to undergraduates, and can kindle a long-term interest in mathematics and its applications.

Cryptology has been of interest to mankind since centuries. One of the oldest known cryptographic methods is the Scytale, used by the ancient Spartans to communicate during military campaigns. The Scytale was first mentioned by the Greek poet Archilochus in the 7th century B.C., and is described by Plutarch (50-120 AD) [5]:

> *The dispatch-scroll is of the following character. When the ephors send out an admiral or a general, they make two round pieces of wood exactly alike in length and thickness, so that each corresponds to the other in its dimensions, and keep one themselves, while they give the other to their envoy. These pieces of wood they call scytalae. Whenever, then, they wish to send some secret and important message, they make a scroll of parchment long and narrow, like a leathern strap, and wind it round their scytale, leaving no vacant space thereon, but covering its surface all round with the parchment. After doing this, they write what they wish on the parchment, just as it lies wrapped about the scytale; and when they have written their message, they take the parchment off and send it, without the piece of wood, to the commander. He, when he has received it, cannot otherwise get any meaning out of it,--since the letters have no*

> *connection, but are disarranged,--unless he takes his own scytale and winds the strip of parchment about it, so that, when its spiral course is restored perfectly, and that which follows is joined to that which precedes, he reads around the staff, and so discovers the continuity of the message. And the parchment, like the staff, is called scytale, as the thing measured bears the name of the measure.* - Plutarch, *Lives* (Lysander 19), ed. Bernadotte Perrin.

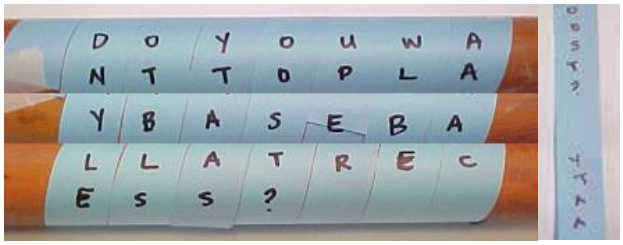

Figure 1. Scytale

Another famous historic cryptosystem is the Shift Cipher, which was used by Julius Caesar to protect messages of military significance. The following example depicts the Caesar Cipher, utilizing a shift of three:

Plaintext: *the quick brown fox jumps over the lazy dog*
Ciphertext: *WKH TXLFN EURZQ IRA MXPSV RYHU WKH ODCB GRJ*

These traditional cryptosystems are examples of *symmetric* cryptosystems. In these methods, the sender and the receiver are required to have a shared private key, which is used both to encrypt and decrypt the messages. Further, once it is known which particular symmetric cryptosystem is being used, encrypted messages can be decrypted either by brute force or by clever cryptanalysis. So in practice it becomes necessary to not let all parties know which symmetric cryptosystem is being used. The relatively recent concept of 'public key' cryptosystems, on the other hand, is revolutionary, in that it does not require the two parties to meet in order to exchange a key, and is based on the assumption that all parties know which method of encryption is being used.

Over the past few years, our students have implemented some private and public key cryptosystems, such as the substitution ciphers, the German Enigma, and others. Watching the encryption and decryption of messages in real-time during class is exciting for the students. Moreover, this direct and active interaction with the cryptosystems facilitates a greater understanding of the underlying mathematics.

Many private key cryptosystems are easily available on the web, including many good examples on Simon Singh's website, *The Black Chamber* [6]. In this paper, we introduce some public key cryptosystems, whose security is based on the Discrete Logarithm Problem, and present their Maple implementations. These Maple implementations also attempt to bridge the gap between theoretical cryptography, whose main concern is security, and practical cryptography, which is guided by efficiency.

Public Key Cryptography

The basic problem of Public Key Cryptography (PKC) is the following: Alice wants to send a secret message to Bob. Alice and Bob cannot meet in person to decide which method to use, or to exchange the key. Evil Eve is 'eavesdropping', and wants to know the secret message. The method of encryption is public knowledge. Evil Eve has high computation power, and is very smart. How can Alice send the secret message to Bob, without Eve being able to decipher it?

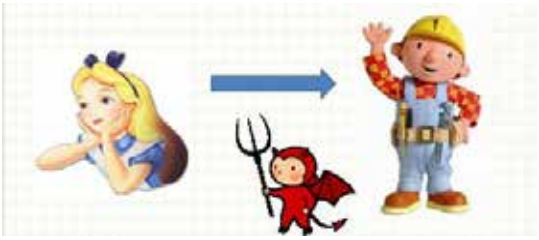

Figure 2. Public Key Cryptography

The basis of PKC is the concept of One-Way Functions. These are functions that are easy to apply, but their inverse is difficult to calculate with limited time and memory resources. This allows for easy encryption, but decryption is hard, unless some other (trapdoor) information is available. The one-way function is used by Alice to encrypt her message, but only Bob, who has the trapdoor information, can decrypt her message. Eve can see the encrypted message, but is not able to decrypt it, since she does not have the trapdoor information.

Figure 3. One-Way Function

In the modern day scenario, Alice wants to send her credit card information to Amazon.com through the internet, and the hacker Eve has access to the encrypted message as it goes through the internet channel. However, despite having access to advanced computational resources, Eve should not be able to decrypt Alice's message to get the credit card information. The only party that should be able to do this is Amazon.com, because they have some additional trapdoor information regarding to the encryption algorithm.

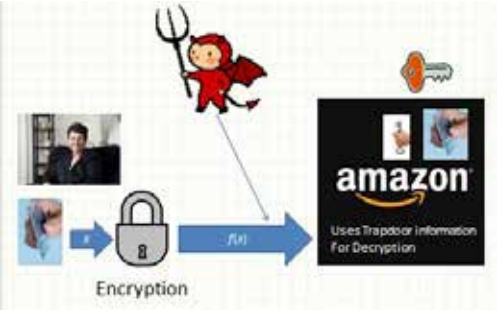

Figure 4. PKC Modern Day Scenario

Discrete Logarithm Problem

Let p be a large prime number, and let g be a *primitive element* (or *generator*) of the group $\mathbb{F}_p$. Then every non-zero element of $\mathbb{F}_p$ is equal to some power of g. By Fermat's Little Theorem, $g^{p-1}=1$. So we can completely describe the multiplicative group $\mathbb{F}_p^* = \{1, g, g^2, ..., g^{p-2}\}$. For example, the set of integers modulo 5, $\mathbb{Z}_5 = \{0,1,2,3,4\}$, is a group with respect to the operations of addition and multiplication modulo 5, and has a primitive element, or generator, 2:

$$\mathbb{Z}_5^* = \{2^0 \equiv 1,\ 2^1 \equiv 2,\ 2^2 \equiv 4,\ 2^3 \equiv 8 \equiv 3\}.$$

The operations of addition and multiplication modulo 5 in $\mathbb{Z}_5$ are shown in the following tables:

+	**0**	**1**	**2**	**3**	**4**
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

*	**0**	**1**	**2**	**3**	**4**
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

The Discrete Logarithm Problem (DLP) seeks to find x, given $f(x) = h = g^x \pmod p$. The integer x is between 0 and p – 1, and is written as $\log_g(h)$. Thus, the discrete logarithm is the function $\log_g : \mathbb{F}_p^* \to \mathbb{Z}_{p-1}$.

While computing the modular exponentiation is easy, the discrete logarithm problem is hard; it takes exponential time by brute force. So the modular exponentiation is a one-way function. The graph below shows that $\log_{101} : \mathbb{F}_{1307}^* \to \mathbb{Z}_{1306}$ is a hard problem, since there the exponential function $f(x) = 101^x \bmod(1307)$ for $1 \le x \le 1306$ does not display any decipherable pattern.

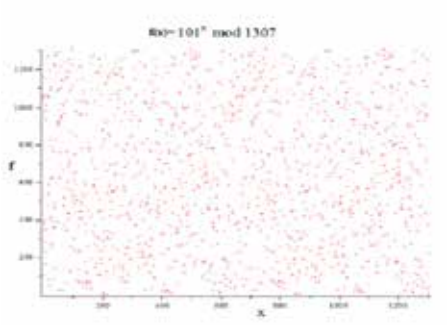

Figure 5. $101^x \bmod(1307)$ for $1 \le x \le 1306$

Diffie-Hellman Key Exchange and its Maple Implementation

This method allows Alice and Bob to exchange a key without meeting in person, through an insecure channel. Alice and Bob start with a large prime number p, say with a hundred digits, and a generator g of $\mathbb{Z}_p$, and place them in public domain. Here g is an integer which is less than, and coprime to p. They then pick their own secret keys x and y, and keep them to themselves. Alice calculates $A = g^x \bmod p$ and sends it over to Bob over the insecure channel. Bob calculates $B = g^y \bmod p$ and sends it over to Alice over the insecure channel. Alice then calculates $B^x \bmod p$ and Bob calculates $A^y \bmod p$. Since

$$A^y \bmod p = \left(g^x\right)^y \bmod p = g^{xy} \bmod p = \left(g^y\right)^x \bmod p = B^x \bmod p,$$

Alice and Bob now have a shared secret key.

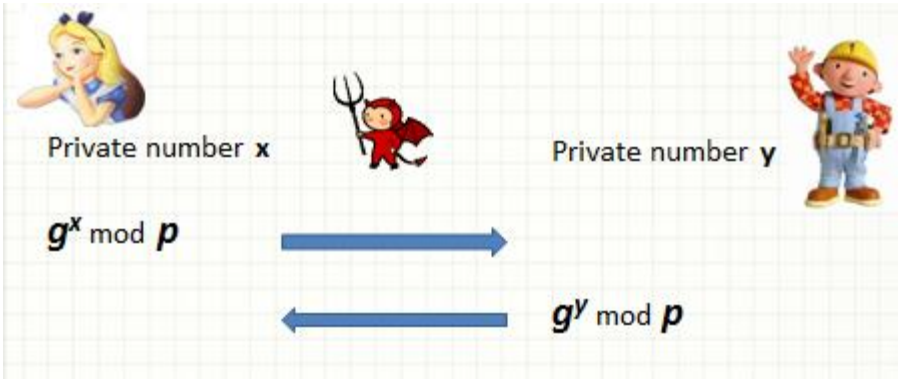

Figure 6. Diffie-Hellman Key Exchange

The protocol relies on the hardness of the Discrete Logarithm Problem. Since modular exponentiation is easy, Alice and Bob can easily calculate the secret key. On the other hand, even though Eve knows p, g, g^x, $g^y \bmod p$, she is unable to determine g^{xy}. The protocol also requires generation of large prime numbers.

The implementation of Diffie-Hellman Key Exchange requires the following steps:

- Generate public modulus p and public base g
- Generate a key pair for Alice and another for Bob
- Give Bob's public key to Alice and Alice's public key to Bob
- Each computes the shared secret with the public key received

Our implementation of the Diffie-Hellman key exchange in Maple follows the algorithm described. First, public information is selected, using the modulus length specified by the user. This information is then used in determining Alice and Bob's public and private keys. Finally, Alice and Bob independently compute the shared secret.

Although Maple can perform modular exponentiation, it does not do so efficiently. It first performs exponentiation through repeated multiplications, and then reduces modulo p. When dealing with large exponents, this requires computing and storing a very large number before the reduction modulo p. In order to decrease the computation time, numbers must be reduced as often as possible, instead of waiting until the end. An additional optimization is done through the use of the following trick:

$$x^n = \begin{cases} \left(x^2\right)^{\frac{n}{2}}, & \text{if } n \text{ is even} \\ x \cdot \left(x^2\right)^{\frac{n-1}{2}}, & \text{if } n \text{ is odd} \end{cases}$$

This trick drastically reduces the exponent into half, and thus allows for much faster computation.

Below is a sample program execution yield:

- >performExchange(10)
  Public modulus: 7214268101
  Public base: 7108591800
- Alice's private key: 698412101
  Alice's public key: 3461922501
- Bob's private key: 511091197
  Bob's public key: 3052477831
- Shared secret, as calculated by
  Alice: 4109225554
  Bob: 4109225554

While the Diffie-Hellman Key Exchange allows two parties to publicly share a secret key, it is just a key exchange, and not a cryptosystem. Once Alice and Bob have a shared secret key, a symmetric encryption algorithm can be used to securely communicate a chosen message.

El Gamal Cryptosystem

A cryptosystem must allow the secure communication of chosen message, not just randomly generated data, as in the Diffie-Hellman key exchange. The El Gamal Public Key Cryptosystem allows Alice to send a secret message to Bob. Its security is also based on the hardness of the discrete logarithm problem.

Alice wants to send a message m to Bob. There are three steps in the protocol: key generation, encryption, and decryption. For key generation, Bob, or another trusted party, chooses a large prime number p, say with a hundred digits, and a generator g of $\mathbb{Z}_p$, and places them in public domain. As before, g is an integer which is less than, and coprime to p, and has a large prime order. Bob chooses a secret number (private key) b, and keeps it to himself, calculates $B = g^b \bmod p$ and sends it over to Alice over the insecure channel. For encryption, Alice choses a random number k, calculates

$$c_1 \equiv g^k \pmod{p}$$
$$c_2 \equiv mB^k \pmod{p}$$

and sends these over to Bob. For decryption, Bob computes $x \equiv c_1^{\,b} \pmod{p}$, and multiplies it's modular inverse to c_2 to obtain the message $c_2 x^{-1} \equiv mB^k \left(B^k\right)^{-1} \equiv m$.

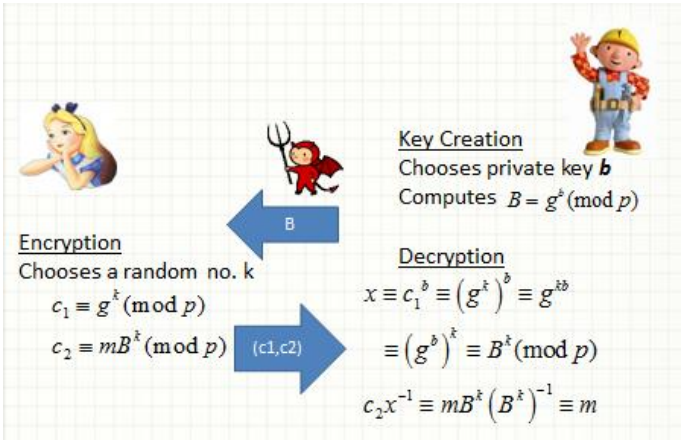

Figure 7. El Gamal Cryptosystem

The implementation of the El Gamal Cryptosystem requires the following:

- Key generation by Bob
  - generate public modulus p and public base g
  - generate Public key B and private key b

- Encryption by Alice
  - Alice splits the message into blocks and converts each block into numbers. Individual characters are converted to numbers based on the ASCII numerical

code
- Choose a different random number k for each character block, and calculate (c_1, c_2)

- Decryption by Bob
  - Decrypt each encrypted block
  - Convert each decrypted block into a series of characters
  - Concatenate all blocks together into the original plaintext string

The number, k must be used to encrypt only one message block. While the plaintext consists of one integer, m, the ciphertext consists of two integers, (c_1, c_2), and all three are between 2 and $p - 1$. Eve knows the public values, p and g, and also the value of $B = g^b \bmod p$, however she is unable to decrypt the message, since she does not know b.

During text encoding, characters are manipulated according to their ASCII values. The user can decide the number of digits in the keys. Below is a sample program execution, encrypting and decrypting the message "Test message", and specifying that 10 digit keys should be used:

```
> performEncryption("Test message", 10)
Key generation: 0.008000 sec.
Encryption: 0.016000 sec.
Decryption: 0.019000 sec.
Public modulus: 8297478979
Public base: 3350272820
Bob's keys:
Private: 595387459
Public: 6023553422
Encrypted message:
[2022194496, 5664659242]
[5118996000, 2717054611]
[8218117505, 618022799]
Decrypted message: Test message
```

The code timing at the beginning of the output indicates that the encryption and decryption are fast. Increasing the key and message length makes the program slower. Modular systems allow the computer to restrain the magnitude of the numbers down to a manageable size.

Conclusions

Cryptology today is more than just sending secret messages. Today, secure communications play an unprecedented role – from defense systems, to e-banking, to e-commerce, to telecommunications, to online course management systems, and so on. Current cryptographic protocols include key exchange; digital signatures, and many

others. For example, we are concerned with authentication (is the message really from Alice, and was not corrupted by Eve on the way, how can Alice make sure that Bob received the message, how can Bob make sure that Alice cannot deny having sent the message), zero-knowledge proofs (how can Alice convince Bob that she has a certain piece of information, without revealing the information to Bob), secret sharing (Alice, Bob, Carly, . . . , Yenjin, and Zake each have a piece of information that is part of a commonly held secret S. If N or more of them meet and combine their knowledge, then S can be reconstructed, but if less than N of them meet, they cannot reconstruct S). All of these protocols are in common usage in computer networks today. It is rewarding to show to the student how mathematics forms the basis of cryptographic systems.

While some of the more traditional cryptosystems, like the Caesar's Cipher, are widely implemented, and easily available on the web, the authors have described the implementation of some of the newer cryptosystems, based on the discrete logarithm problem.

In the future it would be worthwhile to explore the cryptanalysis of these ciphers, and implement various methods that solve the discrete logarithm problem, for example, index calculus method, Shank's baby step giant step algorithm, and the Pohlig Hellman algorithm. By analyzing the weaknesses that these methods can possibly cause, the cryptosystems can be made more secure.

Acknowledgments

We would like to acknowledge our students, most notably Jonathan Schram, who wrote the two codes described in this paper.

References

1. Gomez, Pardo Jose Luis: *Introduction to Cryptography with Maple.* Heidelberg: Springer-Verlag, 2013
2. Hoffstein, Pipher and Silverman: *An Introduction to Mathematical Cryptography.* (1 ed., Vol. 1) New York Springer 2008
3. Koblitz, Neal: *A Course in Number Theory and Cryptography*. Springer 1994
4. Maple help website, http://www.maplesoft.com/support/help/index.aspx Retrieved June 2014
5. Plutarch's comments on Scytale http://crypto.di.uoa.gr/ec13puzzle/Eurocrypt_2013_-_puzzle_page/Eurocrypt_2013_puzzle_page_files/plut-lysandros-scytale2.pdf Retrieved May 2016
6. Singh, S.: *Black Chamber,* http://www.simonsingh.net/The_Black_Chamber/ Retrieved May 2016
7. Trappe, W. and Washington, L.: *Introduction to Cryptography with Coding* (2 ed.) Pearson, 2005