LSU Master's Theses   Graduate School

2007

Oblivious transfer for secure communication

Abhishek Parakh
Louisiana State University and Agricultural and Mechanical College

OBLIVIOUS TRANSFER FOR SECURE COMMUNICATION

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

in

The Department of Electrical and Computer Engineering

by
Abhishek Parakh
B. Tech., Dr. B R Ambedkar National Institute of Technology, Jalandhar, India,
2005
December 2007

Acknowledgements

I sincerely thank Dr Subhash Kak for providing me with invaluable guidance, inspiring talks and enthusiastic support throughout the course of my work. He has been very patient and equally reciprocative in sharing my excitement during my research. He has not only been helpful in research but also a mentor in all other realms of life.

I am indebted to Dr Suresh Rai and Dr Hsiao-Chun Wu for accepting my request to be on the advisory committee. Dr Rai has spared his invaluable time in guiding me through the process of thesis writing and Dr Wu has been an excellent teacher throughout my stay in the department.

Also, I would like to thank my family. Without their support I would not have been able to embark on this journey of academic excellence.

Table of Contents

Abstract

Over the past four decades, computational power and algorithmic strategies have advanced tremendously resulting in an enormous increase in the key sizes required for secure cryptosystems such as RSA. At the same time, the electronic devices have grown smaller and portable requiring algorithms running on them to be optimized in size and efficiency while providing security, at least, equivalent to that provided on a typical desktop computer.

As a result, the industry is moving towards newer cryptosystems such as ECC and NTRU that are well suited for resource constrained environments. While, ECC claims to provide security equivalent to that of RSA for a fraction of key size, NTRU is inherently suited for embedded systems technology.

However, implementation of new cryptosystems requires the development of protocols analogous to those developed using older cryptosystems. In this thesis, we fulfill a part of this requirement by providing protocols for Oblivious Transfer using ECC and NTRU. Oblivious Transfer, in turn, has applications in simultaneous contract signing, digital certified mail, simultaneous exchange of secrets, secure multiparty computations, private information retrieval, etc.

Furthermore, we introduce the idea of basing Oblivious Transfer on public-key exchange protocols. The presentation in the thesis uses Diffie-Hellman Key Exchange, but the scheme is generalizable to any cryptosystem that has a public-key exchange strategy. In fact, our proposal may especially be suited for Quantum Cryptography where the security of key exchange protocols has been proven.

Chapter 1

Introduction

Cryptography is closely related to information theory**.** It is the art of making a piece of information unintelligible to certain people while letting some others make sense of it. Cryptography has been used for thousands of years, mainly in wars and conspiracies where the outcomes have been significantly affected by its proper use. The Second World War was a monstrous proof of this fact. However, today almost every person in a slightly technologically well to do society uses cryptography knowingly or unknowingly, the "simplest" examples being emails, credit cards and phone conversations. Banks rely heavily on their ability to securely communicate with their clients. Further, cryptography has applications in secure storage of information/data, secret-sharing, digital copy-rights management, exchanging secrets, simultaneous contract singing, online games and so on.

If you are thinking, "I am not concerned with any of the above mentioned applications of cryptography, never used a credit card, never used Internet and do not even know the meaning of a computer; therefore, I do not need cryptography," you cannot be further away from the truth. Every time when you leave your home, you certainly lock it, and only those who possess the correct key can enter. The key represents the secret information that is needed to open the lock. It may be in the form of a secret number (for a number lock) or 'jagged' combination of teeth on a conventional key. This is a form of cryptography. The only ways to open the lock (other than using the right key) are to prop it open (a brute force attack) or let a lock-smith try his skills (a form of cryptanalysis).

1.1 Concepts

The field of cryptography can be divided into two subfields: the symmetric key cryptography (secret-key cryptography) and the asymmetric key cryptography (public-key cryptography). While the symmetric key cryptography has existed for thousands of years, the asymmetric key cryptography is a product of the computer age that dawned upon us about four decades ago; the former still remains the most secure form of encryption.

In a symmetric key encryption scheme, given an encryption/decryption key pair (e, d), it is computationally "easy" to determine d knowing only the value of e, and to determine e from d. Therefore, both encryption and decryption keys need to be kept secret. Since $e = d$ in most practical cases, the terms: single-key, one-key, conventional encryption and most importantly secret-key encryption can be found in the literature [22].

Even though symmetric key cryptography is extremely secure, it suffers from two fundamental problems: key-distribution and the extremely large number of keys required. The task of distribution has been traditionally accomplished by secure couriers. However, in the electronic age, with literally millions of users trying to establish secure sessions separated by thousands of miles, couriers become impossible. Moreover, the users need to change the shared session key from time to time.

These practical difficulties brought upon the scientists the need to develop asymmetric key ciphers, often termed as Public-Key Ciphers (PKC). In the PKC, given an encryption key e it is computationally infeasible to determine the corresponding decryption key d. Therefore, the encryption key can be made public while keeping the decryption key secret [22]. This eliminates the problem of key-distribution, as the encryption key can be posted on standard forums, and

significantly reduces the number of keys required in order to communicate between a given group of players.

Public-Key Cryptography (also abbreviated PKC) seems very useful but it is much weaker in security (for a given key size) and order of magnitudes slower than symmetric-key cryptography. In other words, PKC requires much larger key sizes to achieve the same level of security as the secret-key cryptography. Therefore, public-key cryptographic schemes have come to be used for secret key transports and once the shared secret keys are delivered, secure sessions can be established using the conventional methods of encryption. Apart from key distribution, the other applications of PKC include Digital Signatures, Digital Certificates, Data Integrity Check, Oblivious Transfer, etc [30].

1.1.1   Public Key Cryptography

The Public Key Cryptography had its inception in the 1970's when Ralph Merkle [6] showed that it was possible to create problems of controllable difficulty using puzzles. Though Merkle's idea laid the foundations of public-key cryptography, it was little appreciated until Whitfield Diffie and Martin Hellman, published their paper, 'New Directions in Cryptography' [5] that finally catapulted the idea of public-key cryptography to new heights.

The two major public-key schemes that emerged in the 1970s were the Diffie-Hellman scheme for key agreement in 1975 [5] and the key transport and digital signature scheme proposed by Rivest, Shamir and Adleman (RSA [29]) in 1977. The security of the Diffie-Hellman scheme is based on the hardness of the discrete logarithm problem while the RSA scheme *assumes* that its security to be based upon the difficulty of integer factorization.

In 1979, Michael O Rabin proposed a cryptosystem [27] based on the square transformation which was prohibited from use in the RSA scheme. The major advantage of Rabin's

3

cryptosystem was that he could prove its security to be equivalent to integer factorization, which in the case of RSA is only an assumption.

Even though Rabin's cryptosystem has a formal proof of its security being equivalent to integer factorization, it has found little application because of the additional overhead needed to provide disambiguation of the decrypted message.

An alternate sequence of development of public-key cryptography can be found in [7].

1.2 Oblivious Transfer

In 1981, Rabin discovered an interesting application for his cryptosystem and he called this new scheme as 'Oblivious Transfer' [28]. An oblivious transfer scheme is a protocol in which a sender sends a message to a receiver with some fixed probability between 0 and 1 without the sender knowing whether or not the receiver received the message.

Rabin presented the idea of OT applying it to solve the problem of mutual exchange of secrets; his protocol worked only for honest parties; Fischer, Micali and Rackoff presented a protocol [12] which used the concept of zero-knowledge proofs to make the "exchange of secrets" protocol secure against dishonest players. Blum [1] applied the notion to coin flipping, secret exchange and certified electronic mail.

In 1983, Even, Goldreich and Lempel [9] introduced the idea of 1-out-of-2 oblivious transfer where the sender (Alice) sends two secrets S_0 and S_1 and the receiver's (Bob's) input is choice bit b; the latter then learns S_b but gets no information about the other secret S_{1-b}. In turn, Alice does not know which of the two secrets Bob has retrieved.

Kilian [20] proved that the notion of 1-out-of-2 oblivious transfer was enough to develop all secure two party schemes for oblivious circuit evaluation. The idea of Private Information Retrieval was introduced in [3, 4, 21].

1.3 The Need for New Public Key Cryptosystems

Although the above developments were all based on 'exponential modular arithmetic', other cryptosystems were being developed in parallel. Most prominent were the McEliece cryptosystem based on algebraic coding theory and the ElGamal public-key encryption and signature scheme in 1984. The ElGamal scheme was seen as a rival to the RSA scheme and was based on the discrete logarithm problem rather than integer factorization [6].

The Elliptic curve cryptography (ECC) was discovered by Victor Miller and Neal Koblitz in 1985 again based on the discrete logarithm problems. In 1996, J. Hoffstein, J. Pipher and J.H. Silverman, presented the NTRU (Number Theorists aRe Us) cryptosystem which is based on the ring of truncated polynomials [24]. NTRU has come to be known for its efficiency in creating public/private key pairs which makes it feasible to change keys often as well as its encryption and decryption speed.

Two reasons for the upcoming of new cryptosystems are: first, due to the advancement in factorization techniques and increase in computational power, RSA requires at least a 1024 bit key for secure encryption. Second, the electronics industry has seen devices grow smaller. Handheld PDAs, cell-phones with built in internet capabilities, sensor networks that gather crucial military data, embedded systems, etc. have very limited battery life and limited processing power. The algorithms used on them need to be optimized not only in size (due to limited memory) but also in speed (efficiency). Further, Internet access on PDAs and cell-phones means that large number of people use these media for bank transactions, emails access, private

chats, etc. Therefore, security provided in them has to be at power if not greater than that which

is available on a desktop computer.

According to the National Institute of Standard and Technology (NIST) the keys in the

public key cryptosystems must match in strength with the symmetric cryptosystems such as

Advanced Encryption Standard (AES) [6]. This is because AES uses public-key cryptosystems

to provide digital signatures for non-repudiation and key agreement techniques that greatly

simplifies key management. A 128-bit AES key requires a 3072-bit RSA key and a 256-bit key

requires an RSA of 15,360 bits. On the other hand, cryptosystems such as ECC scale linearly

with AES and maintain comparable key sizes. Therefore, a 512-bit key size for ECC suffices for

a 256-bit AES algorithm [6]. ECC is showing up in standardization efforts, including the IEEE

P1363 Standard for public-key cryptography [32].

A comparison of the key size requirements is given in Table 1.1.

Table 1.1. NIST guidelines for key sizes to maintain equivalent strengths across
various cryptosystems [6].

Security (bits)	Symmetric encryption algorithm	Minimum Size of Public Keys (bits)		
		DSA/DH	RSA	ECC
80	-	1024	1024	160
112	3DES	2048	2048	224
128	AES-128	3072	3072	256
192	AES-192	7680	7680	384
256	AES-256	15360	15360	512

If ECC claims smaller key sizes, NTRU provides faster encryption even with key sizes

comparable to that of RSA. Specifically, NTRU claims that for a key consisting of N bits, RSA

and ECC systems require on the order of N^3 operations to encrypt or decrypt a message, while

NTRU requires only on the order of N^2 operations [24]. As a result NTRU is making an impact

on the embedded systems industry and is being considered for numerous standards.

1.3.1 Problem Formulation and Layout of the Thesis

From the previous sections, we can conclude that there is a need for implementation of new cryptosystems that are suitable for low power devices with limited computational power. Also, we have seen that oblivious transfer is an important idea and is a basis for a number of "high level" protocols. Therefore, the development of protocols using cryptosystems such as ECC and NTRU, that are analogous to those that have been developed using exponential arithmetic, is needed. This will provide a smooth transition from present technology to the new ones.

In Chapter 2, we describe the "traditional" oblivious transfer scheme but present its implementation using a new class of transformation introduced in [16]. Chapter 3 provides protocols for oblivious transfer using Elliptic Curve Cryptography. We will follow setup similar to that described in [28] and provide solution for mutual exchange of secrets and 1-out-of-2 oblivious transfer. In Chapter 4 the solution to the above two problems is presented using the NTRU cryptosystem. In chapter 5, we develop an oblivious transfer protocol based on the idea of public-key exchange. Even though we explicitly provide a scheme that is based on Diffie-Hellman transformation, the idea is to introduce new implementation strategy for OT based on key exchange, different from those used before. This has the advantage of generalizing the scheme to all the new cryptosystems, since all of them have an implicit key exchange strategy in them. Chapter 6 discusses the conclusions and future work.

Chapter 2

Implementing Oblivious Transfer

Public-Key Cryptography involves the use of trapdoor functions. Some of these are one-to-one functions, while the others are many-to-one functions. The idea of OT was first developed using the latter type of function. Rabin described OT using the square transformation of the form $c = m^2 \bmod n$, $n = p \times q$, p and q are primes, which results in two or four messages being mapped to a single cipher. Hence, Alice would convey the factors of n_A (assuming Alice is using a public key encryption method of the form $c = m^e \bmod n_A$, where e is the encryption exponent) without knowing for sure whether Bob received the factors or not. In other words, Bob may or may not receive the factors, each happening with probability one-half. Now when Alice sends her secret to Bob encrypting it using her public-key encryption function, Bob will be able to decrypt it only if he has the key (a 50-50 chance).

Another interesting class of many-to-one functions was introduced by Kak [16]. He discussed the application of the transformation of the type $c = m^3 \bmod n$; where, n is a product of two primes p and q such that the Euler's Totient function $\phi(n)$ is divisible only by 3 and not 9. This case was overlooked in [27], where the author had implicitly assumed that $\phi(n)$ is divisible by 9.

The restriction on $\phi(n)$ [16] implies that only one of the factors of n, say p, is of the form $3k + 1$, while the $GCD(\phi(q), 3) = 1$; (GCD stands for Greatest Common Divisor). Such an arrangement causes only three clear-text messages to map to a single cipher; one message less

than the square transformation. Also, Kak's cubic transformation gives rise to a two-third probability of transfer when used to implement oblivious transfer.

In the following sections, we first briefly review Kak's scheme and then discuss its application to oblivious transfer.

2.1 The Cubic Transformation

Consider the transformation $c = m^3 \bmod p$, where $p = 3k + 1$ and $p = 3 \bmod 4$. The condition $p = 3 \bmod 4$ simplifies calculations; in general, one may use other values of p. If $p - 1$ is not divisible by 9, the inverse $c^{1/3}$ can be obtained by using:

$$c^{1/3} = \begin{cases} c^{\frac{p+2}{9}} & \text{if } p - 1 \bmod 9 = 6 \\ c^{\frac{2p+2}{9}} & \text{if } p - 1 \bmod 9 = 3 \end{cases} \tag{2.1}$$

The reason for this is the fact that $c^{p-1} \equiv 1$ by Fermat's Little Theorem and $c^{a(p-1)+3} \equiv c^3$. Hence, $c^{a(p-1)+3/9} \equiv c^{1/3}$. If $p - 1 \bmod 9 = 6$, then $p + 2$ is divisible by 9; if $p - 1 \bmod 9 = 3$, then $2(p - 1) + 3$ is divisible by 9. Consequently, the result in equation 2.1.

It turns out that the cube roots of 1 are 1, α and α^2. Therefore, the cube roots of an arbitrary c are m, $m\alpha$ and $m\alpha^2$. Hence, it suffices to fix a single value for α, between Alice and Bob.

The three cube roots of 1 may be obtained by solving the equation: $\alpha^3 - 1 = 0$. Apart from 1, the other two roots can be obtained by solving $\alpha^2 + \alpha + 1 = 0$, which is easily possible if the square root of $\sqrt{p - 3}$ exits. By Euler's criterion, b is a square modulo p if and only if $b^{(p-1)/2} \equiv 1 \bmod p$. Since $p = 3 \bmod 4$, the square root $a^{1/2} = a^{p+1/4}$, we can write

$$\alpha = \frac{1}{2}\left(-1 \pm (p-3)^{\frac{p+1}{4}}\right)$$
(2.2)

Now, consider the transformation $c = m^3 \bmod n$, where $p = 3k+1$ and $q-1$ is relatively prime to 3. This makes only 3 messages m to map to a single cipher c. To find the inverse we may use:

$$c^{1/3} = \begin{cases} c^{\frac{\phi(n)+3}{9}} & \text{if } \phi(n) \bmod 9 = 6 \\ c^{\frac{2\phi(n)+3}{9}} & \text{if } \phi(n) \bmod 9 = 3 \end{cases}$$
(2.3)

Since, we deal with computations modulo $n = p \cdot q$, we can solve the equation $\alpha^3 - 1 = 0$ modulo the two primes factors of n separately and then combining the results to generate an answer modulo n.

Example 2.1: Set $p = 7$ and $q = 11$. Our first task is to find the solution to the cube root of 1. We can do so by using the Chinese Remainder Theorem (CRT) and solving: $\alpha^3 - 1 = 0$ separately, modulo 7 and 11.

For $p = 7$, we note that $p - 3 = 4$ is square modulo 7, and therefore, $\alpha_p = 2, 4$ by equation (2.2). However, we note that $q - 3 = 8$ is not a square modulo 11, therefore equation (2.2) cannot be used. Hence, we use $\alpha_q = 1$ only; the other values can be found using probabilistic algorithms however, they are not needed. Combining the result using CRT:

$$\alpha = \left(\alpha_p \times q \times \left(q^{-1} \bmod p\right) + \alpha_q \times p \times \left(p^{-1} \bmod q\right)\right) \bmod n$$
(2.4)

We obtain the two values $\alpha_1 = 23$ and $\alpha_2 = 67$. Alice and Bob may, beforehand, agree to use $\alpha = 23$. (Note that $\alpha_2 = \alpha_1^2 \bmod 77$).

Returning to our example, if $m = 12$, then other values that will map to the same cipher are $m\alpha = 34$ and $m\alpha^2 = 45$. Bob may send the cipher along with side information that helps Alice decide on which of the possible cube roots is the message.

2.2 Oblivious Transfer

Suppose Alice is using an encryption function of the form $c = m^3 \bmod n_A$, where $n_A = p_A \cdot q_A$, $p_A = 3k + 1$, $\mathrm{GCD}(q_A - 1, 3) = 1$ and $p = 3 \bmod 4$, to encrypt her secret and that she wants to send this encrypted secret to Bob, obliviously. For Bob to decrypt the secret he must have the decryption key, which in this case is equivalent to having the factors of n_A. However, in an oblivious scheme the recipient should obtain the secret only with a fixed probability between 0 and 1. Thus, the problem of oblivious transfer of secret reduces to the problem of oblivious transfer of the decryption key.

The encryption exponent 3 and n_A are public. The transfer proceeds as follows:

1. Bob randomly chooses an $x < n_A$ and sends to Alice $c_B = x^3 \bmod n_A$.

2. Alice computes $x_1 = \sqrt[3]{c_B} \bmod n_A$ (since she knows the factors of n_A) and sends it to Bob.

3. Bob evaluates $y = GCD((x - x_1), n_A)$.

After step 3, $y = p$ or q with a two-third probability. Consequently, Bob can compute the decryption key with a probability of two-third.

To see the working of the algorithm consider the example below:

Example 2.2: Let $p_A = 13$ and $q_A = 3$, $n_A = p_A \cdot q_A = 13 \cdot 3 = 39$. Let Bob's choice be $x = 11$. And he sends $c_B = 11^3 \bmod 39 = 5$ to Alice. Alice then computes the cube root of 5

modulo 39. Note that $11^3 \equiv 8^3 \equiv 20^3 \equiv 5 \bmod 39$. Hence, Alice may send either 11, 8 or 20; each with equal probability, since She does not know what was Bob's choice.

If Alice sends 11, Bob computes $y = GCD((11-11), 39) = 0$. He does not receive the factors.

If Alice sends 8, Bob computes $y = GCD((11-8), 39) = GCD(3,39) = 3$. He receives q_A.

If Alice sends 20, Bob computes $y = GCD((20-11), 39) = GCD(9,39) = 3$. He receives q_A.

Therefore, we see that Bob receives the factors with a probability two-third. He can consequently compute $\phi(n_A)$ and the decryption key. Alice is completely unaware as to whether Bob received the factors or not. Her knowledge is limited to the apriori probability of two-third. Hence, we have achieved our goal of oblivious transfer.

2.2.1 Comments

Rabin's protocol was built on the square transformation giving rise to an oblivious transfer probability of one-half. Thus the protocol for mutual exchange of secrets, implemented by Rabin, has a non-termination probability of one-quarter for every iteration. However, using Kak's cubic function a mutual exchange of secrets protocol would have a non-termination probability of only one-ninth.

Moreover, probabilities of transfer other than one-half may be useful in certain applications such as lottery draws, where the probability of draw of prizes of larger value must be lower than the probability of draw for the prizes of lower value.

Chapter 3

Oblivious Transfer Using Elliptic Curves*

As seen in the Chapter 1, Elliptic Curve Cryptography (ECC) has started to show up as a serious competitor to RSA due to the tremendous increase in key length required for a secure RSA putting a heavier processing load on applications implementing it. ECC offers equal security compared to RSA for much smaller key sizes, thereby reducing processing overhead.

Rabin [28] presented the notion of oblivious transfer while developing a solution to the problem of mutual exchange of secrets between two distrustful parties. We develop a similar protocol for oblivious transfer using ECC and apply it to provide a solution for mutual exchange of secrets and also implement the 1-out-of-2 oblivious transfer.

3.1 Background

ECC is a public-key cipher that is based on the use of an abelian group. Another example of a well known cipher based on the abelian group is the Diffie-Hellman key exchange where the encryption keys are generated by exponentiation over the group. For elliptic curve cryptography, addition over the group is used and multiplication is replaced by repeated addition. For example, $v \times k = (v + v + \ldots + v)$, where the addition is performed over an elliptic curve [8, 32].

In general, an elliptic curve is defined by an equation containing two variables, with coefficients belonging to the set of real numbers. However, for cryptographic purposes the variables and coefficients are restricted to elements in a finite field:

$$y^2 \bmod p = (x^3 + ax + b) \bmod p \tag{3.1}$$

where a and b are integer constants and p is a prime . The set of points $E_p(a,b)$ is a set (x, y) of all x and y satisfying (3.1).

The order r of a point $T = (x_1, y_1)$ on an elliptic curve is defined as the smallest positive integer r such that $rT = 0$; the evaluation of rT is a computation done on elliptic curve [8]. A point G is called a base point in $E_p(a,b)$ and is picked such that its order r is a very large value.

A finite abelian group can be defined based on the set $E_p(a,b)$ provided that $(x^3 + ax + b) \bmod p$ has no repeated factors, which is equivalent to the condition $(4a^3 + 27b^2) \bmod p \neq 0 \bmod p$. The rules for arithmetic stated below hold good for both the elliptic curves defined over real numbers as well as the finite field [32]. For any points $P, Q \in E_p(a,b)$:

1. O serves as the additive identity. Thus $P + O = P$ and $O = -O$.

In the following we assume that $P \neq O$ and $Q \neq O$,

2. If $P = (x_P, y_P)$ then the point $(x_P, -y_P)$ is the negative of P, denoted as $-P$.

3. Multiplication is defined as repeated addition; i.e. $4P = P + P + P + P$.

4. If $P = (x_P, y_P)$, $Q = (x_Q, y_Q)$ and $P \neq -Q$, then $R = P + Q = (x_R, y_R)$ is determined by the following rules:

$$x_R = (\lambda^2 - x_P - x_Q) \bmod p$$

$$y_R = (\lambda(x_p - x_R) - y_P) \bmod p$$

Where,

$$\lambda = \begin{cases} \left(\dfrac{y_Q - y_P}{x_Q - x_P} \right) \bmod p & \text{if } P \neq Q \\[2em] \left(\dfrac{3x_P^2 + a}{2y_P} \right) \bmod p & \text{if } P = Q \end{cases}$$

Thus the addition operation on elliptic curves is analogous to modular multiplication in RSA and the repeated addition is the analogous to modular exponentiation. The security of ECC arises from the fact that for $Q = kP$, where $Q, P \in E_p(a,b)$ and $k < p$, it is easy to calculate Q given the values of k and P, but it is relatively very hard to determine k given the values of Q and P; this is the discrete logarithm problem over the elliptic curves [32].

A standard elliptic curve transfer proceeds as follows. Encode the plain text message m to be sent as an (x, y)- point P_m. It is the point P_m that will be encrypted as cipher-text and subsequently decrypted. We cannot simply encode the message as the x or y coordinate at a point, because not all such coordinates are in $E_p(a,b)$. User Alice selects a private key n_A and generates a public key $U_A = n_A \times G$. Similarly, Bob generates a public key U_B. To encrypt and send a message P_m to Bob, Alice chooses a random positive integer k and produces a cipher-text C_m consisting of the pair of points

$$C_m = \{ kG ; \ P_m + kU_B \}$$

Alice has used Bob's public key U_B. To decrypt the cipher-text, Bob multiplies the first point in the pair with his own secret key and subtracts the result from the second point:

$$P_m + kU_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$$

Note that Alice has masked the message P_m by adding kU_B to it. Nobody but Alice knows the value of k, so even though U_B is public, nobody can remove the mask kU_B. Reader may refer to [8] for further background on elliptic curve cryptography.

With the above background on Elliptic Curve Cryptography, we set forth on the task of developing the protocol for oblivious transfer.

3.2 Key Observation

If we compare the square transformation in [28] and the elliptic curve equation given by (3.1), we can rewrite (3.1) as

$$y^2 \bmod p = X \qquad\qquad (3.2)$$

where $X = (x^3 + ax + b) \bmod p$. It should be clear to the reader that for every x coordinate there are two possible y coordinates. However, unlike in the square transformation, here neither x nor y can be substituted for a message, because not all values of x and y are permissible in ECC.

3.3 The Proposed Algorithm

Our aim is to allow exchange of secret S_A and S_B between two parties Alice and Bob without using a trusted third party and without simultaneous exchange. Here, we do not go into the details of signing the messages using ECC and take it for granted that all the messages are signed.

Both Alice and Bob select a common elliptic curve $E_q(a,b)$. This information is public. They then decide upon one x-coordinate. Let the two points corresponding to this x-coordinate be P_1 and P_2, whereupon by symmetry $P_1 = -P_2$. The x- coordinate is also public knowledge.

16

Since, Alice and Bob have not decided upon which y - coordinate to use, we will denote Alice's choice of point as P_A and Bob's choice as P_B, such that

$$P_A = P_1 \qquad \text{or} \qquad P_A = P_2$$

Similarly, $\qquad P_B = P_1 \qquad \text{or} \qquad P_B = P_2$.

Even though the x - coordinate is common, neither party knows what is the final point chosen by the other because there are two possible y - coordinates to choose from.

Now, let Alice choose a secret key n_A, which she wishes to use for encryption of her messages, with the aim of obliviously conveying this secret key n_A to Bob. Also, we assume that a procedure for mapping of n_A to a point on elliptic curve has been pre-decided. We call the point on our elliptic curve, corresponding to n_A, as P_{n_A}. Thus, if a person knows P_{n_A}, he can deduce n_A from it. Similar, arrangement is made on Bob's side too.

With the above assumptions, the oblivious transfer of the secret key proceeds as follows:

1. Alice sends to Bob : $n_A P_A$

2. Bob sends to Alice : $\{\ n_B P_B\ ;\ n_B(n_A P_A) + R\ ;\ n_B R\ \}$

   where, n_B is Bob's secret key
   R is randomly chosen point by Bob, belonging to the group $E_q(a,b)$.

3. Alice computes : $n_A [\ n_B(n_A P_A) + R - n_A(n_B P_B)\ ]\ =\ Q$

4. Alice sends to Bob : $\{\ n_A(n_B P_B) + Q\ ;\ n_A(n_B R) + P_{n_A}\ \}$

5. Bob computes :

   a. $n_A(n_B P_B) + Q - n_B(n_A P_A)\ =\ K$

   b. $n_A(n_B R) + P_{n_A} - n_B K\ =\ Z_B$

The sequence of steps presented above achieves our goal of oblivious transfer. The two cases that arise in such a transfer are $P_A = P_B$ and $P_A \neq P_B$. We discuss these two cases below and show how the algorithm given above achieves our goal.

The difference between the two cases arises in step 3. Hence, we analyze them step 3 onwards.

Case I: $P_A = P_B$

3. Alice computes : $n_A [\, n_B(n_A P_A) + R - n_A(n_B P_B) \,] = n_A R$

4. Alice send to Bob : $\{ \, n_A(n_B P_B) + n_A R \; ; \; n_A(n_B R) + P_{n_A} \, \}$

5. Bob computes :

    a. $n_A(n_B P_B) + n_A R - n_B(n_A P_A) = n_A R$

    b. $n_A(n_B R) + P_{n_A} - n_B(n_A R) = P_{n_A}$

Case II: $P_A \neq P_B$

In this case, we note that $P_A = -P_B$. Therefore, the results are as follows:

3. Alice computes : $n_A [\, n_B(n_A P_A) + R - n_A(n_B P_B) \,] = n_A [\, 2 \times n_B(n_A P_A) + R \,]$

4. Alice send to Bob : $\{ \, n_A(n_B P_B) + n_A [\, 2 \times n_B(n_A P_A) + R \,] ; \; n_A(n_B R) + P_{n_A} \, \}$

5. Bob computes :

    a. $n_A(n_B P_B) + n_A [\, 2 \times n_B(n_A P_A) + R \,] - n_B(n_A P_A) = K$

    b. $n_A(n_B R) + P_{n_A} - n_B(K) \neq P_{n_A}$

Note: K is never equal to $n_A R$ unless $P_A = P_B$ or $P_A = P_B = (0, 0)$. Hence, the users are not allowed to choose $P_A = (0, 0)$ and $P_B = (0, 0)$ in our algorithm.

Once the receiver knows P_{n_A}, he can deduce n_A from it. Therefore, this point forward we refer to P_{n_A} as n_A.

However, it is to be noted that no matter what calculations are performed by Bob in step 5, he cannot get n_A if $P_A \neq P_B$. The problem is equivalent to the discrete log problem in case of $P_A \neq P_B$.

Since, $P_A = P_B$ with probability one-half, Bob receives the secret key n_A with probability one-half.

Returning to our algorithm, Bob can verify the value Z_B he has obtained from step 5, whether it is n_A or not, by doing $Z_B \times P_1$ and $Z_B \times P_2$ and checking if one of them is equal to $n_A P_A$ sent to him by Alice in the first step.

In a similar manner, Bob transfers his secret key n_B to Alice with probability one-half. Once this transfer has been achieved we can follow steps similar to those proposed in [28] in order to prevent cheating by either of the parties during exchange of information. Here, we present these steps, adapting them to suit elliptic curve transfers. We define the state of knowledge of the secret keys as follows:

$$W_A = \begin{cases} M, & \text{if Alice knows Bob's secret key} \\ \overline{M}, & \text{if Alice does not know Bob's secret key} \end{cases}$$

Similarly,

$$W_B = \begin{cases} M, & \text{if Bob knows Alice's secret key} \\ \overline{M}, & \text{if Bob does not know Alice's secret key} \end{cases}$$

where M is a constant and $\overline{M}$ is the bit wise complement of M.

After the transfer of keys according the algorithm presented in this chapter and having defined the state of knowledge of keys as above,

Alice sends to Bob: $W_A \oplus S_A$

Bob sends to Alice: $W_B \oplus S_B$

Note: Reader may be reminded from the start of Section 3.3 that S_A is Alice's secret which Bob wishes to know and S_B is Bob's secret which Alice wishes to know.

Note that the above two steps do not provide either party any information about the other's secret. Now, Alice may transfer her secret to Bob using an elliptic curve cryptographic transfer. However, in our scheme, Alice will encode her secret using her secret key and not the public key of Bob. (From section 3.1) We take G to be the base point with large order.

Alice sends to Bob : $S_A + n_A G$

Bob computes (assuming he knows n_A) : $S_A + n_A G - n_A G = S_A$

Bob transfers his secret to Alice in the next step in a similar manner. However, suppose, at the last step Bob were to cheat and not pass on his secret S_B to Alice, then the fact that Bob has cheated Alice implies that Bob has n_A, i.e. $W_B \oplus S_B = M \oplus S_B$. Here $\oplus$ denotes Exclusive-OR operation.

Thus, Alice can do $M \oplus S_B \oplus M = S_B$ and thus obtain S_B. The probability, when the protocol is completed, that neither one knows the other's secret is one-quarter.

Example 3.1: Let Alice and Bob choose an elliptic curve $E_{23}(9, 21)$. The equation corresponding to this curve is $y^2 \bmod 23 = (x^3 + 9x + 21) \bmod 23$. Now, both parties, Alice and Bob, decide upon a common x-coordinate, say 7. The two points corresponding to this x-

coordinate are $P_1 = (7, 6)$ and $P_2 = (7, 17)$. From the properties of elliptic curves, we have $P_1 = -P_2$.

Let Alice choose a secret number $n_A = 5$. We do not explore the details of mapping of n_A to a point on the elliptic curve and just refer to it as P_{n_A}. In turn, let Bob choose a secret number $n_B = 3$ and a random point $R = (2, 1)$. Now we execute our algorithm by considering the two cases separately:

Case 1: $P_A = (7, 6)$ and $P_B = (7, 6)$

1. Alice sends to Bob: $n_A P_A = 5(7, 6) = (11, 18)$.

2. Bob sends to Alice:

$$\{ n_B P_B; \ n_B(n_A P_A) + R; \ n_B R \} = \{ 3(7, 6); \ 3(11, 18) + (2, 1); \ 3(2, 1) \}$$
$$= \{ (1, 10); \ (11, 5); \ (14, 19) \}$$

3. Alice computes:

$$n_A[\ n_B(n_A P_A) + R - n_A(n_B P_B) \ ] = Q$$
$$= 5[\ (11, 5) - 5(1, 10) \ ]$$
$$= 5[\ (11, 5) - (13, 9) \ ]$$
$$= 5[\ (2, 1) \ ] = (7, 17)$$

4. Alice sends to Bob:

$$\{ n_A(n_B P_B) + Q; \ n_A(n_B R) + P_{n_A} \} = \{ (13, 9) + (7, 17); \ 5(14, 19) + P_{n_A} \}$$
$$= \{ (15, 9); (1, 13) + P_{n_A} \}$$

5. Bob computes:

   a) $n_A(n_B P_B) + Q - n_B(n_A P_A) = K$
   $$= (15, 9) - 3(11,18)$$
   $$= (15, 9) - (13, 9)$$
   $$= (7, 17)$$

   b) $n_A(n_B R) + P_{n_A} - n_B(K) = (1, 13) + P_{n_A} - 3(7, 17)$

21

$$= (1, 13) + P_{n_A} - (1, 13)$$
$$= P_{n_A}$$

Case 2: $P_A = (7, 6)$ and $P_B = (7, 17)$

1. Alice sends to Bob: $n_A P_A = 5\,(7, 6) = (11, 18)$.

2. Bob sends to Alice:

$$\{\, n_B P_B; \ n_B (n_A P_A) + R; \ n_B R \,\} = \{\, 3\,(7, 17); \ 3\,(11, 18) + (2, 1); \ 3\,(2, 1) \,\}$$
$$= \{\, (1, 13); \ (11, 5); \ (14, 19) \,\}$$

3. Alice computes:

$$n_A [\, n_B (n_A P_A) + R - n_A (n_B P_B) \,] \ = Q$$
$$= 5\,[\,(11, 5) - 5\,(1, 13)\,]$$
$$= 5\,[\,(11, 5) - (13, 14)\,]$$
$$= 5[\,(3, 11)\,] = (9, 7)$$

4. Alice sends to Bob:

$$\{\, n_A (n_B P_B) + Q; \ n_A (n_B R) + P_{n_A} \,\} = \{\, (13, 14) + (9, 7); \ 5\,(14, 19) + P_{n_A} \,\}$$
$$= \{\, (17, 2); (1, 13) + P_{n_A} \,\}$$

5. Bob computes:

   a) $n_A (n_B P_B) + Q - n_B (n_A P_A) = K$
   $$= (17, 2) - 3(11,18)$$
   $$= (17, 2) - (13, 9)$$
   $$= (2, 22)$$

   b) $n_A (n_B R) + P_{n_A} - n_B (K) = (1, 13) + P_{n_A} - 3\,(2, 22)$
   $$= (1, 13) + P_{n_A} - (14, 4)$$
   $$\neq P_{n_A}$$

3.4 Chosen One-out-of-two Oblivious Transfer

The chosen one-out-of-two oblivious transfer [9], $\binom{2}{1}$- OT for short, is an important application of the basic oblivious transfer protocol. In this transfer, the sender sends two secrets S_0 and S_1 and the receiver's input is choice bit c; the latter then learns S_c but gets no information about the other secret S_{1-c}.

This transfer has been implemented using exponentiations. Here we show that the one-out-of-two oblivious transfer can be implemented using the algorithm we presented.

We assume that both parties are willing to take part in the protocol honestly, i.e. Alice is willing to disclose one out of two secrets that she has to Bob, but Bob does not want Alice to know which one secret he wants to know. Also, Bob should learn only the one secret he wants to know and nothing about the other.

In other words, Alice is said to have two secrets S_0 and S_1. Alice associates different secret keys to each secret. These secret keys will be used to encrypt S_0 and S_1 when transferring them to Bob. Bob must be able to retrieve only one of these two secrets and Alice should not come to know what Bob has extracted.

Let Alice associate keys n_{A_0} with S_0 and n_{A_1} with S_1 for encryption. Bob's task is to retrieve one of these two keys; i.e., retrieve n_{A_1} if he wants to know S_1. The retrieval should be accomplished in such a manner that Alice should not be able to determine which key Bob has retrieved and Bob should not gain any information about the key he could not retrieve.

23

Recall, from the previous section that every x-coordinate yields two points P_1 and P_2 such that $P_1 = -P_2$. Alice declares that she is associating secret S_0 with point P_1 and secret S_1 with point P_2. The transfer of the secrets then proceeds as follows:

1. Alice sends to Bob : $\{\ n_{A_0} P_1\ ;\ \ n_{A_1} P_2\ \}$

2. Bob sends to Alice : $\{\ n_B P_B\ ;\ \ n_B\left(n_{A_0} P_1\right) + R\ ;\ \ n_B\left(n_{A_1} P_2\right) + R\ ;\ \ n_B R\ \}$

3. Alice computes :

   $$n_{A_0}\,[\ n_B\left(n_{A_0} P_1\right) + R - n_{A_0}\left(n_B P_B\right)\ ] = H_1\ ;$$

   $$n_{A_1}\,[\ n_B\left(n_{A_1} P_2\right) + R - n_{A_1}\left(n_B P_B\right)\ ] = H_2\,.$$

4. Alice sends to Bob :

   $$\{\ n_{A_0}\left(n_B P_B\right) + H_1\ ;\ n_{A_0}\left(n_B R\right) + P_{n_{A_0}}\ ;\ \ \ n_{A_1}\left(n_B P_B\right) + H_2\ ;\ n_{A_1}\left(n_B R\right) + P_{n_{A_1}}\ \}$$

   Note: $P_{n_{A_0}}$ and $P_{n_{A_1}}$ is the mapping of secret keys n_{A_0} and n_{A_1} to points on the elliptic curve.

   Bob must have chosen P_B in the second step such that $P_B = P_1$ if Bob wants secret S_0 and $P_B = P_2$ if Bob wants secret S_1. Therefore after step 4, Bob picks up only one of the two pairs of points sent to him by Alice which will yield the secret key he wants.

   For example, if Bob has chosen $P_B = P_1$ then the first pair of points in step 4, i.e. $\{\ n_{A_0}\left(n_B P_B\right) + H_1\ ;\ n_{A_0}\left(n_B R\right) + P_{n_{A_0}}\ \}$, will yield n_{A_0} in the following manner :

5. Bob computes :

   a) $\ n_{A_0}\left(n_B P_B\right) + H_1 - n_B\left(n_{A_0} P_1\right) = H_1 = n_{A_0} R$

   b) $\ n_{A_0}\left(n_B R\right) + P_{n_{A_0}} + n_B\left(n_{A_0} R\right) = P_{n_{A_0}}\,.$

24

From $P_{n_{A_0}}$, Bob can easily calculate n_{A_0}. The second pair of points will not yield any key. Thus, Bob can get only one of the two secret keys and Alice remains oblivious to the fact that which of the two keys did Bob retrieve.

Alice may send both the secrets to Bob in the following manner:

Alice sends to Bob: { $P_{s_0} + n_{A_0}G$; $P_{s_1} + n_{A_1}G$ }, where P_{s_0} is the mapping of secret S_0 to the elliptic curve and P_{s_1} is the mapping of secret S_1 to the elliptic curve.

Bob will be able to retrieve only P_{s_0} in our example because he has only n_{A_0} and hence obtain S_0. He will not be able to get any information from the second half of the message about secret S_1. Alice does not know which of the two secrets Bob obtained. We have achieved our goal of chosen one-out-of-two oblivious transfers.

3.5 Conclusions

This chapter has described the idea of oblivious transfer using elliptic curves and presented an algorithm for its implementation. Also we describe the application of our protocol to the solution of the problem of $\binom{2}{1}$- OT.

The algorithm presented here may be expressed in different variants. The key contribution is the introduction of oblivious transfer to ECC. The one-out-of-two oblivious transfer may be further modified in order to obtain 1-out-of-n oblivious transfer.

In the next chapter, we will develop a protocol for mutual exchange of secrets using NTRU cryptosystem. As mentioned before, though NTRU does not provide any decrease in key sizes, it is inherently faster in implementation and especially suited for embedded technology.

Chapter 4

Exchanging Secrets Using NTRU

In this chapter we will discuss the NTRU cryptosystem and develop the idea of oblivious transfer using NTRU. The application of the idea to the problem of mutual exchange of secrets is presented in section 4.2. While the section 4.3 presents a protocol for 1-out-of-2 oblivious transfer.

4.1 Background

NTRU is a cryptosystem based on the use of polynomials of degree $N-1$ with integer coefficients; e.g.,

$$a = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_{N-1} x^{N-1}$$

Any two polynomials a and b are added together in the usual manner of polynomial addition by summing up the coefficients of similar powers of x. Example, $a + b = (a_0 + b_0) + (a_1 + b_1)x + (a_2 + b_2)x^2 + \ldots + (a_{N-1} + b_{N-1})x^{N-1}$. Similarly, the multiplication laws hold true for the polynomials provided the powers of x are suitably mapped to $N-1$ by replacing x^N by 1, x^{N+1} by x and so no. The set of all such polynomials is denoted by R [24].

The above addition and multiplication rules make R into a ring, which is called the Ring of Truncated Polynomials. In terms of abstract algebra we work in the ring $R = \dfrac{Z[x]}{x^N - 1}$.

The NTRU public-key cryptosystem uses the ring of truncated polynomials along with modular arithmetic. This has further the effect of reducing the coefficients of the polynomial

modulo a number q. Note here that q may not be a prime. Thus the expression a (modulo q) represents the reduction of coefficients of the polynomial a modulo q.

The inverse modulo q of a polynomial a is a polynomial $\overline{A}_q$ with the property that:

$$a * \overline{A}_q \equiv 1 \bmod q$$

Even though not all polynomials have an inverse modulo q, it is easy to determine if a polynomial has an inverse and to compute if it exists. The details of a fast algorithm for computing the inverse is described in [15].

4.1.1   The NTRU Cryptosystem

The NTRU cryptosystem is characterized by the following parameters [24]:

1. N - the polynomials used are of degree $N-1$.

2. q - is a large modulus. (Not necessarily prime.)

3. p - is a small modulus. (Not necessarily prime.)

Table 4.1. Typical values for the above parameters N, q and p [24].

Security	N	q	p
Moderate	167	128	3
Standard	251	128	3
High	347	128	3
Highest	503	256	3

a) *Key Generation:* Alice generates an encryption/decryption key pair (or a public/private key pair). In order to do so she chooses two polynomials f and g belonging to the ring R. The polynomial f should have inverses modulo two numbers q and p. In general, q is large and p is small and that q and p do not have any common factors.

The inverses of polynomial f modulo q and modulo p is denoted by F_q and F_p, i.e. $F_q * f \equiv 1 \bmod q$ and $F_p * f \equiv 1 \bmod p$.

Alice now computes her public key $h \equiv pF_q * g \pmod{q}$. Her private key is the polynomial f.

The polynomials f and g are chosen to be "small" relative to a random polynomial modulo q. In a random polynomial the coefficients are randomly distributed mod q where as in a "small" polynomial, the coefficients are much smaller than q.

b) *Encryption:* When Bob wants to send a message to Alice, he first encodes his message as a polynomial m of degree $N-1$ with coefficients lying between $-p/2$ to $p/2$. He also chooses a small random polynomial r. The encrypted message sent to Alice is of the form:

$$c = r * h + m \pmod{q}$$

c) *Decryption:* Alice decrypts c to obtain the original message polynomial m (or equivalently the message) as follows: Alice computes: $a = f * c \pmod{q}$, where the coefficients of the resultant polynomial a is chosen to lie between $-q/2$ and $q/2$.

Finally, Alice computes: $m' = F_p * a \pmod{p}$.

The resultant m' in the last step is the same as m with an extremely large probability for appropriately chosen parameter values. In some cases, the decryption may fail to yield the correct message and therefore check bits need to be included. For the detailed description of the method for choosing appropriate parameters refer to [15].

The encryption/decryption scheme work because of the following:

$$\begin{aligned} a = f * c \pmod{q} &= f * (r * h + m) \pmod{q} \\ &= f * (r * pF_q * g + m) \pmod{q} \\ &= pr * g + f * m \pmod{q} \end{aligned}$$

and $m' = F_p * a \ (\text{mod } p) = F_p * (pr * g + f * m) \ \text{mod } p = m \ \text{mod } p$.

With this background we discuss, in the next section, the implementation of OT using NTRU.

4.2   The Proposed Technique

Suppose Alice and Bob wish to exchange secrets S_A and S_B, however they do not trust each other.   The problem is to develop a protocol without trusted third party and without a simultaneous exchange. Here we propose a protocol to do so using the NTRU encryption. We have outlined the advantages of using NTRU in Chapter 1. We take for granted that every message exchanged between Alice and Bob is signed.

In order to conduct the exchange, Alice generates a two secret encryption keys h_{A_1}, h_{A_2} and their corresponding decryption keys f_{A_1}, f_{A_2}. She is going to use one of these encryption keys to encrypt her original secret S_A when transferring it to Bob. However, for Bob to decrypt the cipher to obtain the secret, he needs to know the correct decryption key, which will be transferred to him "obliviously" with a probability of one-half. Bob generates two secret encryption/decryption key pairs h_{B_1}/f_{B_1} and h_{B_2}/f_{B_2}. These will be used for the transfer of S_B from Bob to Alice. For all the computations Alice and Bob agree upon the parameters to be used publically, i.e. N, p and q, where they have their usual meaning. The method of key generation, encryption and decryption remains the same as described in section 4.1.

Now, Alice generates another pair of encryption/decryption key h_A/f_A which she will use to conduct the protocol. Bob generates h_B/f_B. These pairs are also kept secret.

Therefore, all the keys generated by either party are kept secret. Only one of the main decryption keys (f_{A_1}, f_{A_2}) will be transferred obliviously (vice versa for f_{B_1}, f_{B_2} in the second half of the protocol) . The protocol proceeds as follows (appropriate modulus operations are taken to be implicit in all the steps):

1. Alice generates a random polynomial r_A and using the encryption key h_A, she encrypts the two secret keys, f_{A_1}, f_{A_2} and sends them to Bob.

   Alice sends to Bob: $c_1 = r_A * h_A + f_{A_1}$ and $c_2 = r_A * h_A + f_{A_2}$.

2. Bob randomly chooses one of the ciphers, c_1 and c_2 and encrypts it using his own secret key h_B and a randomly chosen polynomial r_B.

   Bob sends to Alice: $c = r_B * h_B + r_A * h_A + f_{A_1}$ or $c = r_B * h_B + r_A * h_A + f_{A_2}$.

3. Alice decrypts the cipher sent by Bob by subtracting $r_A * f_A$ from c and sends it back to Bob.

   Alice sends to Bob: $c' = r_B * h_B + f_{A_1}$ or $c' = r_B * h_B + f_{A_2}$ depending on what she receives.

4. Bob can now subtract $r_B * h_B$ from c' and obtains one of the keys f_{A_1} or f_{A_2}. Alice remains oblivious to which of the two keys Bob has received.

Steps 1 through 4 achieve oblivious transfer of encryption keys; the following steps are for the exchange of secrets.

5. Bob sends to Alice $M_1 \oplus S_B$ if he has received f_{A_1} else he sends $M_2 \oplus S_B$.

Here, M_1 and M_2 are pre-agreed random constants. Alice at this point does know which of the two keys Bob has received and hence she does not know if Bob has sent $M_1 \oplus S_B$ or $M_2 \oplus S_B$ to her.

6. Alice randomly chooses one of the two encryption keys h_{A_1} or h_{A_2} and another random polynomial r.

    Alice sends to Bob: $c_{S_A} = r * h_{A_1} + S_A$ or $c_{S_A} = r * h_{A_2} + S_A$.

    where we have assumed that secret S_A is represented in the polynomial form with coefficients of the polynomial chosen to lie between $-p/2$ and $p/2$.

Bob will receive the secret S_A if he can successfully decrypt c_{S_A}, which will depend on his chances of having the decryption key for the encryption function that Alice has randomly chosen. Since, there are two possible encryption functions Alice can choose from, Bob's probability of receiving the secret after step 6 is one-half. Alice will remain oblivious to whether Bob has received the secret.

Similar transfer of secret keys and exchange of secret takes place from Bob to Alice.

Step 6 is a purely NTRU PKCS transfer involving encryption and decryption in the NTRU system.

If after receiving S_A, Bob was to cheat and not send his secret to Alice then Alice will know which of the two XORs ($M_1 \oplus S_B$ or $M_2 \oplus S_B$) had Bob sent in step 5 and therefore deduce S_B.

4.3 One-out-of-two Oblivious Transfer

A situation in which Alice has two secrets S_1 and S_2 such that Bob wants one of these without Alice knowing which one of the secrets he has retrieved is called 1-out-of-2 OT. On the

other hand, Alice is concerned that Bob must get only the secret he chooses and no information about the other secret. We had presented a protocol for the solution of this problem in chapter 3. Here we solve the problem using NTRU encryption functions.

It is not difficult to see the solution since we have discussed the protocol for exchange of secrets in the previous section. With a few modifications to the already presented protocol we can achieve our goal of 1-out-of-2 oblivious transfer. The protocol is as follows:

1. Alice declares that she is sending secrets S_1 and S_2 in order (represented in appropriate polynomial form).

   Alice send to Bob: $c_1 = r_A * h_A + S_1$ and $c_2 = r_A * h_A + S_2$.

2. Bob chooses c_1 if he wants to retrieve secret S_1 and c_1 if he wants to retrieve secret S_2.

   He then encrypts his choice using his secret key h_B and a randomly chosen polynomial r_B.

   Bob sends to Alice: $c = r_B * h_B + r_A * h_A + S_1$ or $c = r_B * h_B + r_A * h_A + S_2$.

3. Alice decrypts the cipher sent by Bob by subtracting $r_A * h_A$ from c and sends it back to Bob.

   Alice sends to Bob: $c' = r_B * h_B + S_1$ or $c' = r_B * h_B + S_2$ depending on what she receives.

4. Bob can now subtract $r_B * h_B$ from c' and deduce one of the secrets, depending on his choice. Alice remains oblivious to which of the two secrets Bob has received.

At the end of the protocol, Bob will deduce only one of the two secrets and not be able to get any information about the other. Alice will not know which of the two secrets Bob received.

4.4   Conclusions

In the chapter we have provided schemes for mutual exchange of secrets and 1-out-of-2 oblivious transfer using NTRU. The protocol may be generalized to 1-out-of-n oblivious transfer and non-interactive oblivious transfer may be possible to implement.

In this next chapter we will discuss the broader idea of basing the protocol for oblivious transfer on key-exchange protocols. This is an important idea since almost every public-key cryptosystem has a method for public-key exchange which can be modified to implement oblivious transfer.

Chapter 5

Oblivious Transfer Based on Key Exchange*

In this chapter we construct a protocol for oblivious transfer using key exchange similar to the Diffie-Hellman (DH) protocol [5], which is a popular method for establishing a shared key between two parties over an insecure channel. We modify the Diffie-Hellman protocol such that the two communicating parties will succeed or fail in establishing a shared key each with a probability of one-half. However, the party sending the secret will not know if the receiver has the same key as he/she does.

The disadvantage of the protocol described by Rabin in [28] is that it is valid only when the encryption key is factorization dependent. In other words, Rabin's protocol works only for "RSA- type" encryption schemes. However, the advantage of using the idea of oblivious transfer based on key exchange protocols is that after the keys are exchanged obliviously, the players can use any mutually agreed encryption scheme to encrypt their secrets.

Further, with the advent of quantum computers, all the present protocols will be needed to be developed using quantum cryptography. Quantum key exchange has been proven to be secure and therefore an oblivious transfer scheme may be possible to develop using ideas similar to those presented in this chapter.

There have been implementations [23] of 1-out-of-n OT based on the Decision Diffie-Hellman (DDH) problem [2]. However, our protocol differs from previous ones in the sense that - firstly, we describe a scheme for mutual exchange of secrets based on DH. Secondly, in the previous implementations the 1-out-of-n OT use the DDH for the transfer itself, i.e. applies the Diffie-Hellman exponentiation for the encryption of secrets directly. Here we administer the idea

of the oblivious key exchange. Once the keys are exchanged (obliviously), the parties may use any mutually agreed encryption method for the actual transfer / exchange of secrets.

5.1 Preliminary

Diffie-Hellman key exchange was the first published public-key algorithm and a number of commercial products still employ this exchange technique [32]. The purpose of the algorithm is to enable two users to exchange a key securely that can be used for subsequent encryption of messages. The algorithm itself is limited to the exchange of the keys. As in the case of Elliptic Curve Cryptography, the Diffie-Hellman algorithm also depends for its effectiveness on the difficulty of computing discrete logarithms. Briefly, we can define the discrete logarithm in the following way.

A number x is said to be a primitive root of a prime number p if consecutive exponentiations from $x \bmod p$, $x^2 \bmod p$,..., $x^{p-1} \bmod p$ generate distinct numbers lying between 1 and $p-1$ in some order. Given $b \equiv x^i \bmod p$ where x is a primitive root of p then the task of finding a unique exponent i where $0 \leq i \leq (p-1)$ satisfying the equation is known as taking the discrete logarithm of b for the base x, $\bmod p$.

With this background we proceed with the description of the Diffie-Hellman key exchange algorithm. The publicly known numbers are: a prime number p and an integer x that is a primitive root of p.

1. Alice selects a random and secret integer $X_A < p$ and computes $Y_A = x^{X_A} \bmod p$.

2. Bob independently selects a random and secret integer $X_B < p$ and computes
   $Y_B = x^{X_B} \bmod p$.

3. The values of Y are available publically.

35

4. Alice and Bob can communicate by computing a shared key $K = (Y_B)^{X_A} \bmod p$ and

$K = (Y_A)^{X_B} \bmod p$, respectively.

The two calculations in step 4 produce the same key as follows:

$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod p \\
&= \left(x^{X_B} \bmod p\right)^{X_A} \bmod p \\
&= \left(x^{X_B}\right)^{X_A} \bmod p \\
&= \left(x^{X_A}\right)^{X_B} \bmod p \\
&= \left(x^{X_A} \bmod p\right)^{X_B} \bmod p \\
&= (Y_A)^{X_B} \bmod p
\end{aligned}
$$

At the end of execution of the algorithm the two participants have exchanged a secret key. The opponent has x, p, Y_A and Y_B available to him and in order to break the algorithm, he is forced to take a discrete logarithm to determine the key.

In practice, the participants choose numbers p and x, such that p is a large prime on the order of at least 300 decimal digits (1024 bits), $p-1$ has a large prime factor and x is a generator of order $p-1$ in the multiplicative group Z_p (a generator is a primitive root of p). This ensures the security of the protocols not only against eavesdroppers but also against the opposing party, which is to be considered as an adversary as well in our protocol. Since we will be working only in Z_p, we often do not state it explicitly.

5.2   Mutual Exchange of Secrets – The Proposed Protocol

Suppose Alice and Bob possess secrets S_A and S_B respectively, which they wish to exchange, however, they do not trust each other. We would like to complete the exchange without a trusted third party and without a procedure for simultaneous exchange of secrets; the

latter being practically impossible to implement when the parties are geographically far apart. Both parties are assumed to have an appropriate mechanism to digitally sign every message they send.

Let the secrets S_A and S_B be passwords to files that Bob and Alice want to access such that if a wrong password is used then the files will self-destruct. This prevents the parties from trying random passwords. The protocol is based on the oblivious exchange of encryption keys.

In the protocol, we exploit the fact that there exist $g_1, g_2 \in Z_p$, $g_1 \neq g_2$ such that they map to a single cipher c, where $c \equiv g_1^2 \bmod p \equiv g_2^2 \bmod p$. Let K_A denote the key that Alice uses to encrypt her secret, while Bob uses K_B to encrypt his secret. With these assumptions, the protocol proceeds as follows:

1. Alice and Bob agree upon a prime p, a number $x \in Z_p$ as the generator and c such that $c \equiv g_1^2 \bmod p \equiv g_2^2 \bmod p$ (Alice and Bob both know g_1 and g_2).

2. Alice privately chooses $g_A = g_1$ or $g_A = g_2$ and two random numbers N_{A_1} and N_{A_2}.

3. Bob secretly decides on g_B, such that $g_B = g_1$ or $g_B = g_2$ and a random number N_B.

4. Alice sends to Bob: $x^{g_A + N_{A_1}} \bmod p$ and $x^{N_{A_2}} \bmod p$.

5. Bob sends to Alice: $\left(\dfrac{x^{g_A + N_{A_1}}}{x^{g_B}} \right)^{N_B} \bmod p$ and computes $K_A' = \left(x^{N_{A_2}} \right)^{N_B} \bmod p$ for himself.

6. Alice computes: $K_A = \left[\left(\dfrac{x^{g_A + N_{A_1}}}{x^{g_B}} \right)^{N_B} \right]^{\frac{N_{A_2}}{N_{A_1}}} \bmod p$.

7. Bob chooses a random message M and sends $C = f(M, K_A')$ to Alice.

8. Alice sends back $Y = f^{-1}(C, K_A)$ to Bob.

Here $f(m,k)$ is a function known to both Alice and Bob, where m is the input, k is the key, and knowing $c = f(m,k)$ does not reveal the key used. Therefore, f may be an encryption function using a secret key and f^{-1} is the decryption function.

Two cases arise from the above sequence, namely $g_A = g_B$ and $g_A \neq g_B$. If $g_A = g_B$ then $K'_A = K_A$, else $K'_A \neq K_A$. Hence, Bob receives K_A with probability one-half. Steps 7 and 8 help Bob check if he has K_A by comparing Y and M.

Similarly, exchange of K_B takes place from Bob to Alice.

Define states,

$$U_b = \begin{cases} K, & \text{if Bob received } K_A. \\ \overline{K}, & \text{if Bob did not receive } K_A. \end{cases}$$

where, $K \in Z_p$ and $\overline{K}$ is the bitwise complement of K. U_a is similarly defined.

Mutual agreement:
1. Prime p such that $p-1$ has a large prime factor.
2. a number $x \in Z_p$ that is a primitive root of p.
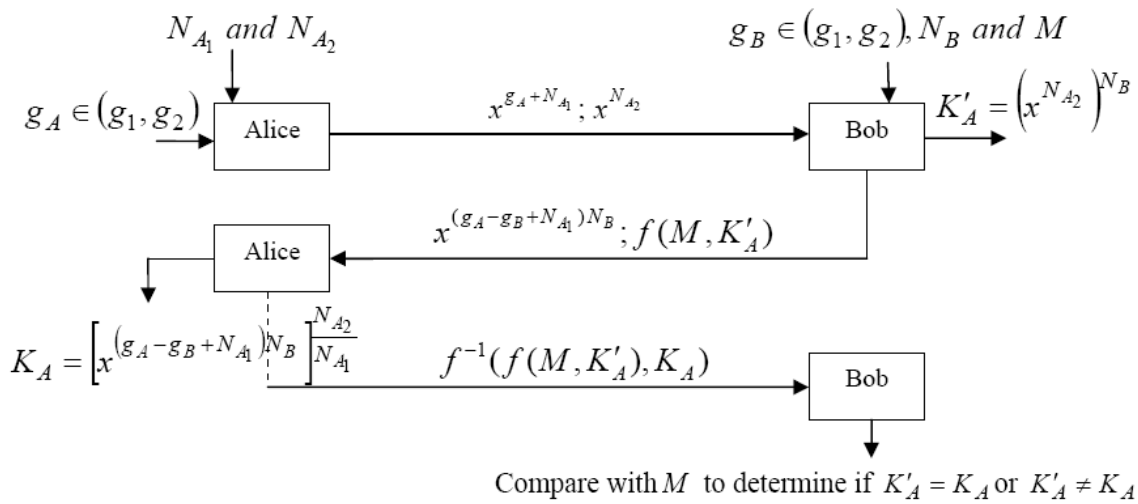3. a number c, where $c = g_1^2 \bmod p = g_2^2 \bmod p$.

Figure 5.1 Illustration of proposed algorithm to achieve oblivious exchange of encryption key (all computations performed in Z_p).

In order to prevent cheating by either party, Alice sends $U_a \oplus S_A$ to Bob and Bob sends $U_b \oplus S_B$ to Alice. Since, neither party knows other's state of knowledge of the secret key, this step does not provide either party with any knowledge of other's secret.

Finally, Alice and Bob exchange their secrets encrypting them using K_A and K_B, respectively.

If at the last step, after Alice sends her encrypted secret to Bob, Bob was to cheat and not send his secret to Alice, then the fact that Bob cheated implies that Bob received K_A and $U_b = K$ and that Bob had previously sent $U_b \oplus S_B = K \oplus S_B$. Alice can retrieve S_B by computing $K \oplus S_B \oplus K = S_B$.

The probability, after the protocol is complete, that neither party knows other's secret key is one-fourth.

Example 5.1: Alice and Bob wish to exchange secrets S_A and S_B. They agree upon $p = 23$, $x = 5$ and $c = 9$. Therefore, $\sqrt{c} \bmod p = g_1 = 3$ and $\sqrt{c} \bmod p = g_2 = 20$. We examine the two cases arising in step 2 of the algorithm.

Case I: $g_A = g_B$

2. Alice chooses: $g_A = g_1 = 3$ and two random numbers $N_{A_1} = 5$ and $N_{A_2} = 15$.

3. Bob chooses: $g_B = g_1 = 3$ and $N_B = 17$.

4. Alice sends to Bob: $x^{g_A + N_{A_1}} \bmod p \equiv 5^{3+5} \bmod 23 \equiv 16$ and

$$x^{N_{A_2}} \bmod p \equiv 5^{15} \bmod 23 \equiv 19.$$

5. Bob sends to Alice:

$$\left(\frac{x^{g_A+N_{A_1}}}{x^{g_B}}\right)^{N_B} \mod p \equiv \left(\frac{5^{3+5}}{5^3}\right)^{17} \mod 23 \equiv \left(\frac{16}{125}\right)^{17} \mod 23$$

$$\equiv \left(16 \times 125^{-1}\right)^{17} \mod 23$$

$$\equiv \left(16 \times 7\right)^{17} \mod 23 \equiv 7$$

and computes for himself: $K'_A \equiv \left(x^{N_{A_2}}\right)^{N_B} \mod p \equiv 19^{17} \mod 23 \equiv 21$.

6. Alice computes: $K_A \equiv \left[\left(\frac{x^{g_A+N_{A_1}}}{x^{g_B}}\right)^{N_B}\right]^{\frac{N_{A_2}}{N_{A_1}}} \mod p \equiv (7)^{\frac{15}{5}} \mod 23 \equiv 21$.

Bob may encrypt a random message with the key that he has generated and ask Alice to decrypt it using her key to determine if he has K_A. Since Alice and Bob have chosen $g_A = g_B = 3$, then $K'_A = K_A = 21$. The choice $g_A = g_B = 20$ gives similar results.

Case II: $g_A \neq g_B$

2. Alice chooses: $g_A = g_1 = 3$ and two random numbers $N_{A_1} = 5$ and $N_{A_2} = 15$.

3. Bob chooses: $g_B = g_2 = 20$ and $N_B = 17$.

4. Alice sends to Bob: $x^{g_A+N_{A_1}} \mod p \equiv 5^{3+5} \mod 23 \equiv 16$ and

$$x^{N_{A_2}} \mod p \equiv 5^{15} \mod 23 \equiv 19.$$

5. Bob sends to Alice:

$$\left(\frac{x^{g_A+N_{A_1}}}{x^{g_B}}\right)^{N_B} \mod p \equiv \left(\frac{5^{3+5}}{5^{20}}\right)^{17} \mod 23 \equiv \left(\frac{16}{12}\right)^{17} \mod 23$$

$$\equiv \left(16 \times 12^{-1}\right)^{17} \mod 23$$

$$\equiv \left(16 \times 2\right)^{17} \mod 23 \equiv 9$$

and computes for himself: $K'_A \equiv \left(x^{N_{A_2}}\right)^{N_B} \bmod p \equiv 19^{17} \bmod 23 \equiv 21$.

6. Alice computes: $K_A \equiv \left[\left(\dfrac{x^{g_A+N_{A_1}}}{x^{g_B}}\right)^{N_B}\right]^{\frac{N_{A_2}}{N_{A_1}}} \bmod p \equiv (9)^{\frac{15}{5}} \bmod 23 \equiv 16$.

In this case, Alice and Bob have chosen $g_A \neq g_B$, hence $K'_A \neq K_A$. The alternate choice, $g_A = 20$ and $g_B = 3$ yields similar results.

In none of the cases can Bob predict beforehand what choice Alice has made, so the protocol remains fair.

Security issues: The protocol breaks down if Bob is able to compute both $\left(x^{N_{A_2}}\right)^{N_B} \bmod p$

and $\left[x^{(g_A-g_B+N_{A_1})N_B}\right]^{\frac{N_{A_2}}{N_{A_1}}} \bmod p$. We see that Bob can deduce $x^{N_{A_1}}$ and $x^{N_{A_2}}$, which he may compute

$\dfrac{x^{N_{A_2}}}{x^{N_{A_1}}} \equiv x^y \bmod p$. Given $y = N_{A_2} - N_{A_1}$, deducing y is a DLP. If we assume that "somehow"

Bob is able to deduce y, then in order for him to compute the ratio $\dfrac{N_{A_2}}{N_{A_1}}$, he still needs to know

either N_{A_1} or N_{A_2}, which is again equivalent to a DLP. Based on the assumption that a Discrete

Log Problem is difficult to solve, the protocol remains secure.

5.3 One-out-of-two Oblivious Transfer

One of the most powerful primitives that have led to the invention of numerous cryptographic schemes is the one-out-of-two oblivious transfer. It may conceptually be described as a black box where Alice puts in two secrets, S_1 and S_2, such that Bob can only retrieve one of them while getting no information about the other. Bob is concerned that Alice should not know which secret he retrieved.

A situation may be such that a spy wishes to sell one out of two secrets that he possesses, while the buyer does not wish the spy to know which information he wants. In such a situation the 1-out-of-2 oblivious transfer can be employed. It is assumed that the party possessing the two secrets is willing to disclose one and only one of these to the other.

The procedure of choosing prime p, generator number x and $c \equiv g_1^2 \bmod p \equiv g_2^2 \bmod p$ remains identical to that described before. However, this time Alice uses secret keys K_1 and K_2 to encrypt secrets S_1 and S_2, respectively. She announces to Bob that she is associating key K_1 with g_1 and key K_2 with g_2. With these initial conditions the protocol follows:

1. Alice secretly chooses N_{A_1} and sends to Bob: $x^{g_1 + N_{A_1}} \bmod p$.

2. Bob chooses $g_B = g_1$ (if he wants secret S_1) or $g_B = g_2$ (if he wants secret S_2) and secret numbers N_B and N_{B_1}.

3. Bob sends to Alice: $\left(\dfrac{x^{g_1 + N_{A_1}}}{x^{g_B}} \right)^{N_B N_{B_1}} \bmod p$ and $x^{N_B} \bmod p$.

4. Alice chooses a number N_{A_2} and sends to Bob: $\left[\left(\dfrac{x^{g_1 + N_{A_1}}}{x^{g_B}} \right)^{N_B N_{B_1}} \right]^{N_{A_2}} \bmod p$.

5. Bob computes: $K_B \equiv \left[\left(\dfrac{x^{g_1 + N_{A_1}}}{x^{g_B}} \right)^{N_B N_{B_1} N_{A_2}} \right]^{\frac{1}{N_{B_1}}} \bmod p \equiv \left(\dfrac{x^{g_1 + N_{A_1}}}{x^{g_B}} \right)^{N_B N_{A_2}} \bmod p$.

6. Alice computes: $K_1 \equiv x^{N_B N_{A_1} N_{A_2}} \bmod p$ and $K_2 \equiv \left(x^{N_B (g_1 - g_2 + N_{A_1})} \right)^{N_{A_2}} \bmod p$.

7. Alice encrypts secret S_1 using K_1 and secret S_2 using K_2 and sends them to Bob.

From the above sequence we see that if Bob chooses $g_B = g_1$, then $K_B = K_1$ and if Bob chooses $g_B = g_2$, then $K_B = K_2$. Hence, Bob will only be able to retrieve one of the two secrets depending upon his choice, while Alice will not be able to determine which secret Bob has retrieved.

Security issues: In order for Bob to cheat, he needs to compute both K_1 and K_2. His best option is to determine one of the keys honestly and using that, try to deduce the other key. For instance, if Bob honestly computes K_1, then he will have access to $x^{N_{A_1}}$ and $\dfrac{x^{N_{A_1} N_{A_2}}}{x^{N_{A_1}}}$. But this does not provide him with any information about N_{A_1} and N_{A_2} which he needs to compute K_2. Similarly, he cannot calculate K_1 from K_2. The problem is again equivalent to efficiently solving a DLP.

5.4   Coin-Flipping Protocols

A couple may decide on which restaurant to go to or whether they should take a vacation or buy a car for their next anniversary, by tossing a coin. In this case flipping a coin is a trivial matter since both parties are present at the same place physically. However, problems arise when the participants are geographically separated over large distances. How are they supposed to fairly flip a coin when both of them cannot see the outcome simultaneously? Many business transactions require such an arrangement or a simple game of gambling over the Web may need a fair coin-toss.

It turns out that any oblivious transfer scheme may be suitably modified to flip a coin and so can be the protocol for mutual exchange of secrets that we have presented. For instance, if Bob receives the same key as Alice then Bob wins the toss else Alice wins. After Bob declares the

key he has computed, Alice replies if he won or lost and reveals all the variables that she had chosen which Bob can use to verify Alice's claim. Bob need not disclose any of the variables of his choice.

Another approach to coin-tossing by telephone is using d-sequences [17]. This becomes possible because the digits of the d-sequence are generated by an exponentiation process [18], [19]. But this will not be discussed further in this thesis.

5.5   Conclusion

Our algorithms, in this chapter, open up the possibility of development of oblivious transfer schemes using key exchange protocols. Academically, it appears that such algorithms should have preceded Rabin's protocol. The existence of our algorithm shows that there are numerous variations on the implementation of OT protocols. Also, most OT schemes can be extended to coin flipping with minor modifications, in which case, only one sided transfer may take place and victory or loss depends on the opposing party being lucky enough to deduce the key.

Our protocol for mutual exchange of secrets is different from Rabin's protocol in the sense that the latter aims at obliviously transmitting the decryption key from the transmitter to the receiver whereas we establish a shared key between the transmitter and receiver with probability one-half. Higher exponents may be employed to generate transfer probabilities other than one-half. It turns out that the Diffie-Hellman protocol is a powerful primitive and can be used as a basis for implementing many cryptographic protocols that have been implemented via the RSA type transformations. This possibility had been overlooked.

Chapter 6

Conclusions

This thesis provides protocols to conduct mutual exchange of secrets using the idea of oblivious transfer and also provided schemes for one-out-of-two oblivious transfer. In Chapters 2 and 3, the schemes provided are based on ECC and NTRU, respectively that are currently competing against RSA and have certain advantages over the latter. Before this thesis, oblivious transfer had been implemented only using exponential arithmetic and hence the wide spread use of the new cryptosystems depends on the fast development of schemes analogous to those that have been previously implemented using the outgoing cryptosystems.

In the last chapter of the thesis, we have discussed an oblivious transfer scheme using the DH-key exchange for implementation. The scheme is not restricted to DH-key exchange and the basic idea is to emphasize the fact that oblivious transfer can be implemented using public-key exchange techniques.

As a passing note we should also mention that with the advent of quantum computing and the discovery of fast factorization algorithm and discrete logarithm algorithm, if and when a quantum computer is realized, the systems of classical cryptography will collapse and *may be* the only form of cryptography that will be secure will be quantum cryptography [31]. However, quantum cryptography has had its own share problems with the discovery of no-go theorems. Yet quantum key exchange has been proven to be secure and our idea of basing the oblivious transfer on key exchange schemes might turn out to be a useful notion in quantum cryptography.

Also, we would like to mention that even though the factorization and discrete logarithm algorithms have already been discovered in quantum computing, there is not yet a quantum algorithm for breaking the NTRU cryptosystem [24].

References

1. Blum, M. Three applications of the oblivious transfer: Part i: Coin flipping by telephone; part ii: How to exchange secrets; part iii: How to send certified electronic mail. Technical report, Department of EECS, University of California, Berkeley, CA, 1981.

2. Boneh, D. The Decision Diffie-Hellman Problem. Proceedings of the Third Algorithmic Number Theory Symposium. Springer-Verlag LNCS 1423, 1998, 48-63.

3. Chor, B., Goldreich, O., Kushilevitz, E. and Sudan, M. Private Information Retrieval. Journal of the ACM, Volume 45, Issue 6, 1998, 965-981.

4. Chor, B. and Gilboa, N. Computationally Private Information Retrieval. Proceedings 29th ACM Symposium on Theory of Computing, 1997, 304-313.

5. Diffie, W. and Hellman, M. E. New Directions in Cryptography, IEEE Transactions on Information Theory, vol. IT-22, Nov. 1976, 644-654.

6. Code and Cipher, Vol 1. No. 1. A copy can be found at www.certicom.com. September 9, 2007.

7. Ellis, J. H. The History of Non-Secret Encryption. 1987 (made public in December 1997)

8. Enge, A. Elliptic Curves and their Applications to Cryptography. Kluwer Academic, Boston, 1999.

9. Even, S., Goldreich, O. and Lempel, A. A randomized protocol for signing contracts. Proceedings Crypto '82, 205-210, 1983.

10. Feige, U., Fiat, A. and Shamir, A. Zero Knowledge Proofs of Identity. Proceedings of the 19th ACM Symposium. on Theory of Computing, May 1987, 210-217.

11. Fiat, A. and Shamir, A. How to prove yourself: Practical solutions to identification and signature problems, Advances in Cryptology - Crypto '86, Springer-Verlag (1987), 186-194.

12. Fischer, M., Micali, S. and Rackoff C. A secure protocol for the oblivious transfer. EuroCrypt 84, 1984.

13. Goldreieh, O., Micali, S. and Wigderson, A. "Proofs That Yield Nothing But Their Validity and a Methodology of Cryptographic Protocol Design", Proceedings of FOGS 1986, 174-187.

14. Goldwasser, S., Micali, S. and Rackoff, C. The knowledge complexity of interactive proof systems. ACM Symposium on Theory of Computing, ACM Press, New York, USA, 1985, 210-217.

15. Hoffstein, J., Pipher, J., Silverman, H. J. NTRU: A Ring-Based Public Key Cryptosystem. Lecture Notes in Computer Science 1423, Springer-Verlag, Berlin, 1998, 267-288.

16. Kak, S. A cubic public-key transformation, Circuits, Systems and Signal Processing, Volume 26, 2007, 353-359.

17. Kak, S. A new method for coin flipping by telephone, Cryptologia, vol. 13, pp. 73-78, 1989.

18. Kak, S. and Chatterjee, A. On decimal sequences, IEEE Transactions on Information Theory, IT-27: 647 – 652, 1981.

19. Kak, S. Encryption and error-correction coding using D sequences, IEEE Transactions on Computers, C-34: 803-809, 1985.

20. Kilian, J. Founding cryptography on oblivious transfer. Proceedings of the twentieth annual ACM symposium on Theory of computing, 1988, 20-31.

21. Kushilevitz, E. and Ostrovsky, R. Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval. Proceedings 38th IEEE Symposium on Foundations of Computer Science, 1997, 364-373.

22. Menezes, J. A., Oorschot, C. P., Vanstone, A. S. Handbook of Applied Cryptography, CRC, 1996.

23. Naor, M. and Pinkas, B. Efficient Oblivious Transfer Protocols, Proceedings of SODA 2001 (SIAM Symposium on Discrete Algorithms), January 7-9 2001, Washington DC.

24. NTRU Tutorials and FAQs. www.ntru.com/cryptolab. September 9, 2007.

25. Parakh, A. Oblivious Transfer using Elliptic Curves, Cryptologia, Volume 31, Issue 2 April 2007, 125-132.

26. Parakh, A. Oblivious Transfer based on Key Exchange, arXiv:0705.0178 (To appear in Cryptologia.)

27. Rabin, M. O. Digitalized signatures and public-key functions as intractable as factorization. MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.

28. Rabin, M. O. How to exchange secrets by oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.

29. Rivest, R., Shamir, A. and Adleman L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, Vol. 21 (2), pp.120–126. 1978. (Technical Report in 1977)

30. Schneier, B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. Wiley, 2 Edition, 1995.

31. Singh, S. The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. Anchor, 2000.

32. Stallings, W. Cryptography and Network Security: Principles and Practice. Third Edition. Prentice Hall. 2003.

Appendix: Copyright Agreement

Taylor & Francis
Taylor & Francis Group

TRANSFER OF COPYRIGHT AGREEMENT

The transfer of copyright from author to publisher must be clearly stated in writing to enable the publisher to assure maximum dissemination of the author's work. Therefore, the following agreement, executed and signed by the author, is required with each manuscript submission. (If the article is a "work made for hire" it must be signed by the employer.)

The article entitled **Oblivious Transfer Using Elliptic Curves**

is herewith submitted for publication in **Cryptologia**

It has not been published before, and it is not under consideration for publication in any other journals. It contains no matter that is scandalous, obscene, libelous, or otherwise contrary to law. When the article is accepted for publication, I, as the author (U.S. Government employees: see bottom of page), hereby agree to transfer to Taylor & Francis all rights, including those pertaining to electronic forms and transmissions, under existing copyright laws, except for the following, which the author(s) specifically retain(s):

1. The right to make further copies of all or part of the published article for my use in classroom teaching or for any other;
2. The right to reuse all or part of this material in a compilation of my own works or in a textbook of which I am the author;
3. The right to make copies of the published work for internal distribution within the institution that employs me.

I agree that copies made under these circumstances will continue to carry the copyright notice that appeared in the original published work. I agree to inform my co-authors, if any, of the above terms. I certify that I have obtained written permission for the use of text, tables, and/or illustrations from any copyrighted source(s), and I agree to supply such written permission(s) to Taylor & Francis upon request.

Abhishek   2/26/07
_____
(1)Signature and date

ABHISHEK PARAKH
_____
(1)Name and title

_____
(1)Institution or company (if appropriate)

_____
(2)Signature and date

_____
(2)Name and title

_____
(2)Institution or company (if appropriate)

Government Copyright

I certify that the above article has been written in the course of the author"s employment by the United States Government, so that it is not subject to U.S. copyright laws, or that it has been written in the course of the author"s employment by the United Kingdom Government (Crown Copyright).

_____
Signature   Date   Title

Note to U.S. Government Employees:

★   If the above article was not prepared as part of the employee's duties, it is not a U.S. Government work.
★   If the above article was prepared jointly, and any co-author is not a U.S. Government employee, it is not a U.S. Government work.

50

Vita

Abhishek Parakh was born in Khetri Nagar, a mining town in the state of Rajasthan, India. He completed his bachelor's degree in electronics and communications engineering from National Institute of Technology, Jalandhar, India.

He is currently with the Department of Electrical and Computer Engineering at Louisiana State University, Baton Rouge, Louisiana. He expects to receive his Master of Science in Electrical Engineering degree in Fall 2007.

His research interests include cryptography, signal processing and philosophy of science.