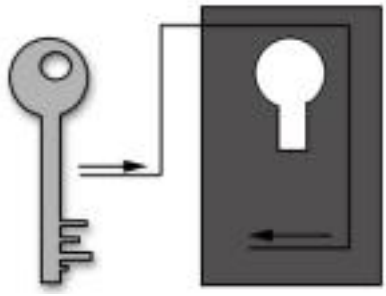
Crossroads
The ACM Student Magazine

Public Key Cryptography

By ***Pradosh Kumar Mohapatra***

Introduction



E-commerce on the world-wide web is a rapidly growing and proliferating field. But there are several differences between commerce in the real world and on the Internet, and perhaps the most fundamental issue is that of trust and security. When a consumer walks in to a store to buy goods, she presents herself, her identity, and a payment method. But on the Internet, both the buyer and the seller have difficulties proving each other's identity. How can the buyer convince herself to transmit sensitive information to the seller? How does the seller know about a legitimate purchase order? How do both parties become aware if an uninvited third party copies or modifies transaction information? These questions and many others describe the issues affecting commercial transactions over the Internet. In order to build secure E-commerce applications, we need to establish a definition of various security requirements. The following four requirements have been identified as the framework for secure E-commerce [**7**]:

- **Confidentiality:** Protecting the data from all but the intended receiver(s).
- **Authentication:** Proving one's identity.
- **Integrity:** Ensuring no unauthorized alteration of data.
- **Non-repudiation:** Preventing an entity from denying previous commitments or actions.

The universal technique for providing confidentiality of transmitted data is conventional cryptography. But as we will see later on, conventional cryptosystems do not satisfy the requirements of authentication, integrity, and non-repudiation. Public key cryptography is the first truly revolutionary advance in cryptography that satisfies these requirements. This paper discusses public key cryptographic algorithms and how

they are used in practical E-commerce applications.

The basic functionality of cryptography is to hide information. Its operation typically includes two processes: **encryption** as the process of transforming information so that it is unintelligible to an intruder, and **decryption** as the process of transforming the encrypted information so that it is intelligible again. The information in its original form is known as **plain text**, and the encrypted message is called **cipher text**. Figure 1 illustrates these processes.

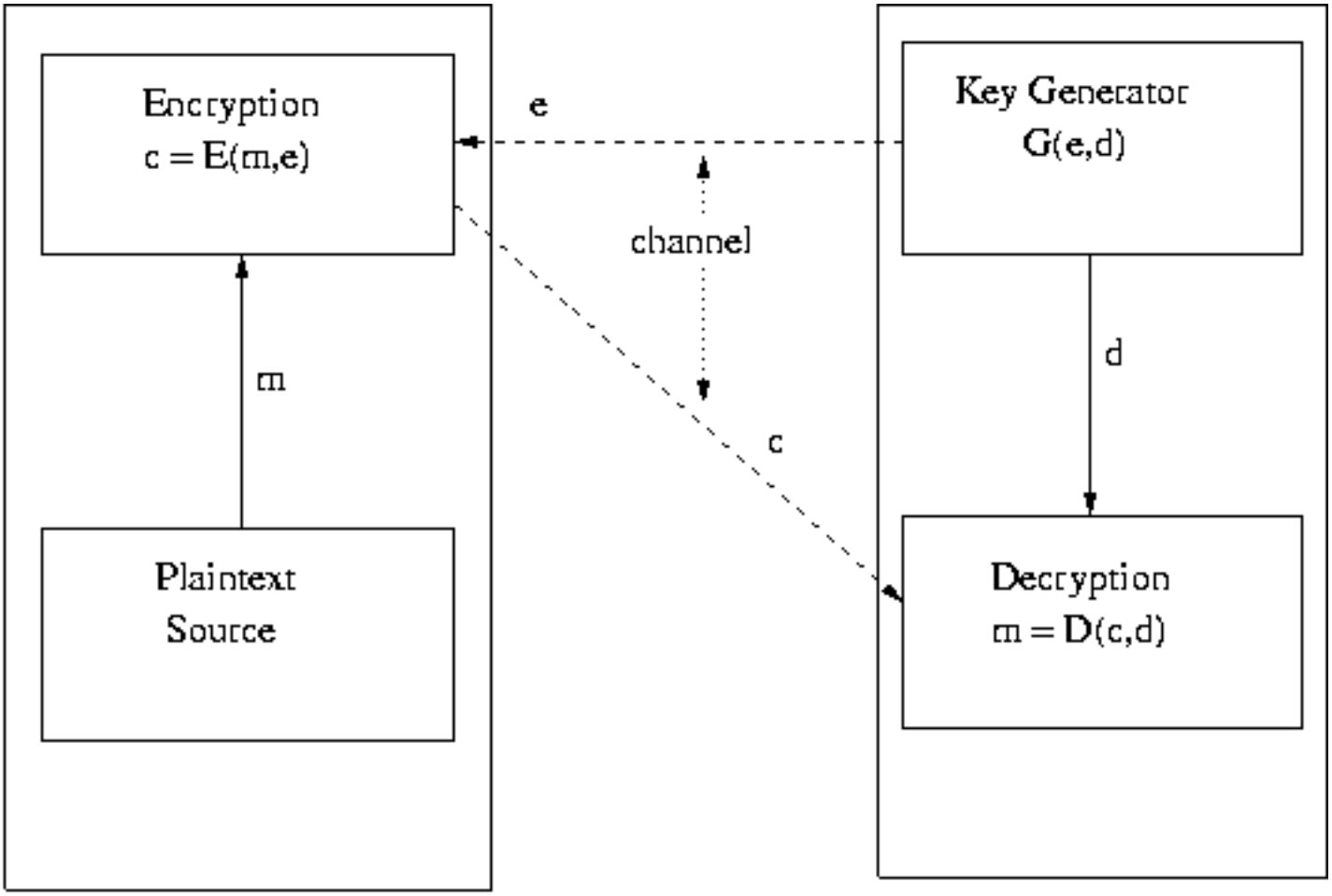


FIGURE 1: A Generic Cryptosystem

Suppose Alice and Bob want to have a secure communication. They first secretly choose or exchange a key pair **(e,d)**. At a subsequent point of time, if Alice wishes to send some secret data **m** to Bob, she applies a mathematical function involving the key **e** to compute the cipher text **c = E(m, e)** and transmits **c** to Bob. Upon receiving **c**, Bob applies the inverse mathematical function with the other key **d** to compute **m = D (c, d)** and hence recovers the original data **m**. The security lies in the fact that the mathematical function and the key are only bound to the sender and receiver, not to anybody else.

One fundamental question arises as to why keys are necessary. (Why not just choose one encryption function and its corresponding decryption function?) Having functions associated with keys means that if the functions are revealed, then the scheme does not need to be redesigned, but simply the keys can be changed. In fact, keys are very critical to the functionality of cryptographic algorithms and it is sound cryptographic practice to change keys frequently.

Primitives for Cryptography

Fundamentally, cryptography is a subject for mathematicians and, in particular, number theorists. As is evident from **figure 1**, the basic objective of cryptography should be to keep the process of transforming plain text to cipher text relatively easy. On the other hand, the reverse process of converting cipher text to plain text should be effectively impossible. Such a process is called a **one-way function** and is based on the difficulty of solving a precisely formulated mathematical problem. Cryptographers have found that problems from group theory are often the best suited for this purpose. **Appendix A** discusses some facts concerning the basics of group theory and number theory that will be necessary to understand cryptographic algorithms.

Motivation for the Public Key Cryptography

Traditional cryptosystems (commonly known as symmetric systems or secret key systems) require the sender and the receiver to share a key known only to them. Knowledge of the key allows the decipherment of the cipher text, hence the need for secrecy. Referring to **figure 1**, this system requires that **e = d**. Secret key cryptography is perceived to have an extensive history. The most common algorithm used for secret key systems is the Data Encryption Algorithm (DEA) defined by the Data Encryption Standard (DES). Other schemes [**14**, **15**] include Triple DES, IDEA, RC4, RC5, Blowfish, and Twofish. Though these systems provide strong security, they suffer from many disadvantages including:

- **Key distribution/exchange**: In a two-party communication, the key must remain secret and must be known to both the sender and receiver before the transaction. Thus the two parties must take great care in exchanging the key so that an eavesdropper does not obtain it.
- **Key management:** In a large network, there are many key pairs to be managed. Moreover, to guarantee security, the key should be changed

frequently, perhaps for each communication session.

Thus, classical secret key systems create problems of trust. Moreover, the requirements of authentication, integrity, and non-reputability discussed above are impossible to achieve in such cryptosystems [**14**]. The breakthrough came in 1976 with the invention of public key cryptography by Diffie and Hellman [**5**] (described in section **Diffie-Hellman Key Exchange**). In addition to easing the key distribution and key management problems, public key cryptosystems have plenty of other advantages over the symmetric systems. Moreover, they adequately satisfy the aforementioned four requirements. The following section describes this cryptosystem in detail.

Public Key Cryptography

In contrast to symmetric key systems, public key systems require each user **A** to have two keys: a public key, $\mathbf{K_{pub}(A)}$ that is published and a private key $\mathbf{K_{pri}(A)}$ that is kept secret. A message encrypted (with a ruleset **E**) using one key can be decrypted (using ruleset **D**) using *only* the other key. This eliminates the need to share a key secretly. Alice wishing to send a message to Bob uses Bob's public key to encrypt the message (denoted as $\mathbf{E_{Kpub(Bob)}(m)}$). Bob, upon receipt of the encrypted message, can use his private key to decrypt it (denoted as $\mathbf{D_{Kpri(Bob)}(c)}$). To understand the concept, we can take an analogy of a snail mail system. We can send mail to anyone knowing only his/her address. Once the mail is in the recipient's mailbox, only the recipient can read it (assuming that only the receiver can open the mailbox).

Theoretically, a public-key system can be constructed by a special case of the one-way function known as a **trapdoor one-way function**. Mathematically, a one-way function **f** is one for which **f(x)** is easy to compute for any input **x**, but $\mathbf{f^{-1}}$ is hard to compute (i. e., it is difficult to solve the equation **f(x) = y** when **y** is known). A trapdoor one-way function is one for which the equation **f(x) = y** becomes easy to solve if additional information (the trapdoor) is available. The following two problems are considered the most likely candidates for creating trapdoor one-way functions:

- **Integer Factorization Problem:** INSTANCE: a composite number **n** that is the product of two large prime numbers. QUESTION: find the prime numbers **p** and **q** such that **n = pq**. While finding large prime numbers is a relatively easy task, the problem of factoring the product of two large primes is considered computationally infeasible if those primes are carefully selected. However, if **phi**

(n) is known, **p** and **q** can easily be determined. This additional information is the trapdoor.

- **Discrete Logarithm Problem:** INSTANCE: a prime **p**, a **generator** **g** of $Z_p^{>*}$, and an element **a** of Z_p^*. QUESTION: find the unique integer **i**, **0 <= i < p-1**, such that $a \equiv g^i \pmod p$. The utility of discrete logarithm problem in cryptography is that finding discrete logs is difficult. The brute force method of computing $g^j \bmod p$ for **1 < j <= p-1** is not practical for large **p**. However, the inverse operation of exponentiation can be computed efficiently. So $g^i \bmod p$ is a one-way function for suitable **p**.

Most public key cryptosystems are based on the hardness of the above two problems. The following sections describe two of the most widely used public key systems based on the above problems.

Diffie-Hellman Key Exchange

This was the first ever public key cryptosystem to be published and was meant to address the issue of key exchange in secret-key systems. The protocol is based on the difficulty of determining **discrete log** in group Z_p^*. The formal ruleset of this scheme is illustrated in **figure 2**.

INSTANCE: a prime **p**, **generator** **g** of Z_p^*.

ASSUMPTION: **p** and **g** are publicly known.

ALGORITHM:

1. Alice chooses **a** at random, **0 <= a < p-1**.
2. Bob chooses **b** at random, **0 <= b < p-1**.
3. Alice computes $u = g^a \bmod p$ and sends to Bob.
4. Bob computes $v = g^b \bmod p$ and sends to Alice.
5. Alice computes $K = (g^b)^a \bmod p$ upon receipt of message from Bob.
6. Bob computes $K = (g^a)^b \bmod p$ upon receipt of message from Alice.

FIGURE 2: Diffie-Hellman Key Exchange Protocol

At the end of the ruleset, both Alice and Bob would have computed the same value $g^{ab} \bmod p$. They can use this as the secret key for the next set of transactions and use symmetric cryptography to encrypt/decrypt data. An eavesdropper, Eve, tapping onto the transaction gets the values of $g^a \bmod p$ and $g^b \bmod p$, but not $g^{ab} \bmod p$. So the security of the system lies in the following question:

"Given the values of $g^a \bmod p$ and $g^b \bmod p$ (but not **a** or **b**), is it feasible to compute the value of $g^{ab} \bmod p$?"

This problem is known as the diffie-hellman problem. If Eve could determine **a** from $g^a \bmod p$ or **b** from $g^b \bmod p$, then she could compute $K = g^{ab} \bmod p$. But both these are instances of the discrete log problem. So it is conjectured that the diffie-hellman problem is equivalent to discrete log problem [**15**].

RSA Cryptosystem

The RSA scheme is one of the most popular public key cryptosystems. It was published by R.L. Rivest, A. Shamir, and L.M. Adleman in 1978 [**12**]. The system is based on the hardness of the **integer factorization problem** in group Z_n. It can be described as a two step process as follows:

1. **RSA setting**: to generate a public/private key pair.

1. Bob generates two large primes, **p** and **q**.
2. He computes **n = pq** and **phi(n) = (p-1)(q-1)**. **n** will be used as the modulus.
3. He chooses a random **e** (**0 < e < phi(n)**) such that **e** is **relatively prime** to **phi(n)**. **e** used as the public exponent.
4. He computes **d** as the inverse of **e modulo phi(n)** (i.e., **d** such that **ed ºd phi(n))**). **d** will be used as the private exponent.
5. He publishes **(n,e)** as his public key and keeps **(n,d)** as his private key. The primes **p** and **q** must be kept secret or destroyed.

FIGURE 3:/tableating Public-Private Key Pair for RSA

The condition to notice here is that **e** should be **relatively prime** to **phi(n)**. Otherwise, there will be no modular inverse and we can not find any private exponent **d**. Note also that the step 4 for computing **d** can be easily carried out by **solving the modular linear equation** **ed °od phi(n))** as described before.

2. **RSA algorithm**: to encrypt and decrypt data.

INSTANCE: group $\mathbf{Z_n}$ and the set **(n,e,d)**: **n = pq**, **p** and **q** primes. **ed °1 (mod phi (n))**.

ASSUMPTION: Alice knows Bob's public key **(n,e)**, but does not know Bob's private key **(n,d)**.

ALGORITHM:

1. Alice encrypts message **m** by computing $\mathbf{c = m^e \bmod n}$.
2. Alice sends **c** to Bob.
3. Upon receipt, Bob decrypts **c** by computing $\mathbf{c^d \bmod n}$ and gets back **m**.

FIGURE 4: RSA Algorithm Execution: Alice wants to send message m to Bob

As can be seen from the figure, the only arithmetic operation required for encryption and decryption is modular exponentiation, i.e., computation of a function of the form $\mathbf{x^y \bmod n}$. There is a well-known method to compute this, which takes polynomial time as a function of the number of bits in the binary representation of **x** [**3**]. Note that the encryption and decryption are inverse operations. A proof of this (or more specifically the fact that step 3 in the RSA algorithm of **Figure 4** is true) is given in **Figure 5**.

	GIVEN:		**ed º 1 (mod phi(n))**
	=>	**ed = k * phi(n) + 1** for some **k**.	
c^d mod n	**º**	**$(m^d)^e$ mod n**	
	º	**m^{ed} mod n**	
	º	**$m^{k * phi(n)+1}$ mod n**	
	ºstrong>	**$(m^{phi(n)})^k m$ mod n**	
	º	**$1^k m$ mod n**	
	º	**m mod n**	

FIGURE 5: Proving that RSA works

The security of this scheme is based on the hope that the encryption function **$c = m^e$ mod n** is one-way, so it will be computationally infeasible for the opponent, Eve, to decrypt **c**. It is necessary for Eve,to know the value of **d** in order to compute **$m = c^d$ mod n**. As discussed above, **e** and **d** are related by the equation:

ed º 1 (mod phi(n)).

So she can compute **d** from **e** provided he has knowledge of **phi(n)**. For **n = pq**, where **p** and **q** are primes, **phi(n)** is given by:

phi(n) = (p-1)(q-1).

>So it has been conjectured that breaking RSA (which is in fact knowing **phi(n)**) is equivalent to factoring **n** to get **p** and **q** [**15**].

Implementing RSA: To implement the RSA system, we convert the text into numeric form. This can be done by removing all spaces and punctuations and assigning

numbers from 0 to 25 for the letters **A** through **Z**. The message **M** so obtained is then divided into blocks $\mathbf{M_i}$ such that $\mathbf{0 <= M_i < n}$. It is necessary to have $\mathbf{M_i < n}$, otherwise $\mathbf{M_i^{ed} \bmod n}$ (as shown in **figure 5**) will not be equal to $\mathbf{M_i}$. Then the algorithm given in **figure 4** is applied to each $\mathbf{M_i}$. If **n** has **k** digits, then we can guarantee that $\mathbf{M_i < n}$ by letting $\mathbf{M_i}$ have at most **k-1** digits.

Example RSA: Suppose p = 1733, q = 2347. n = 4067351. phi(n) = 1732*2346=4063272. Example e (public key) can be 31 as 31 is relatively prime to phi (n)=4063272. The decryption exponent d is computed by solving ed º 1 (mod phi(n)). This yields d = 3145759. So the encryption function is $E(m) = m^{31} \bmod 4067351$ and the decryption function is $D(c) = c^{3145759} \bmod 4067351$.

Suppose we want to encrypt the phrase "public key cryptography." The encoding routine is:

a - 00, b - 01, c - 02, d - 03, e - 04, f - 05, g - 06, h - 07, i - 08, j - 09, k - 10, l - 11, m - 12
n = 13, o - 14, p - 15, q - 16, r - 17, s - 18, t - 19, u - 20, v - 21, w - 22, x - 23, y - 24, z - 25

That yields the following encoding of the phrase:

p - 15, u - 20, b - 01, l - 11, i - 08, c - 02, k - 10, e - 04, y - 24
c - 02, r - 17, y - 24, p - 15, t - 19, o - 14, g - 06, r - 17, a - 00, p - 15, h - 07, y - 24

As n has 7 digits, we can divide the message into blocks, M_i of six digits each to ensure M_i <; n. That gives the following:

M_1	M_2	M_3	M_4	M_5	M_6	M_7
152001	110802	100424	021724	151914	061700	150724

Encrypting each block using E results in the following ciphers:

$E(M_1)$	$E(M_2)$	$E(M_3)$	$E(M_4)$	$E(M_5)$	$E(M_6)$	$E(M_7)$
2721372	3969831	2416419	1795753	2110079	0242624	0889174

Applying the decryption function to these cipher texts recovers the original plain text stream.

Elliptic Curve Cryptography

The strength of public key systems is based on the hardness of intractable computational problems like the **discrete logarithm problem** and the **factoring** problem. They are defined over some finite group, mainly over $\mathbf{Z_n}$ and $\mathbf{Z_p^*}$. But theoretically, we can consider any group while defining these problems. Apart from experimental reasons, there are primarily two reasons why we should consider a variety of groups. First, the custom-defined operation in some groups may be easier to implement in software or hardware than the operation in other groups. Second, the computational problems that form the base of cryptosystems may be harder in another group. Consequently, one can use a group **G** that is smaller than $\mathbf{Z_n}$ or $\mathbf{Z_p^*}$, but has same or greater level of security. This results in smaller key sizes, and faster implementations. Elliptic curves [**6**, **8**, **15**] defined over a **finite field** produce one such group where the **discrete log problem** is intractable. They were first proposed for cryptographic use independently by Neal Koblitz and Victor Miller [**8**] in 1985.

Definition: Let $\mathbf{p > 3}$ be a prime. The elliptic curve over $\mathbf{Z_p}$ (written as $\mathbf{E(Z_p)}$) is the set of solutions $\mathbf{(x,y)}$ in $\mathbf{Z_p x\ Z_p}$ to the congruence:

$$\mathbf{y^2 \equiv x^3 + ax + b \pmod p}$$

where **a**, **b** are in $\mathbf{Z_p}$ and the discriminant $\mathbf{4a^3 + 27b^2 <> 0 \pmod p}$, together with a special point **O** called the **point at infinity**. Varying the values of **a** and **b** gives us different elliptic curves over $\mathbf{Z_p}$.

The elliptic curve $\mathbf{E(Z_p)}$ can be made into an **abelian group** by defining the addition

operation on its points. An example of the addition operation over an elliptic curve is shown in **figure 6** ([2]).

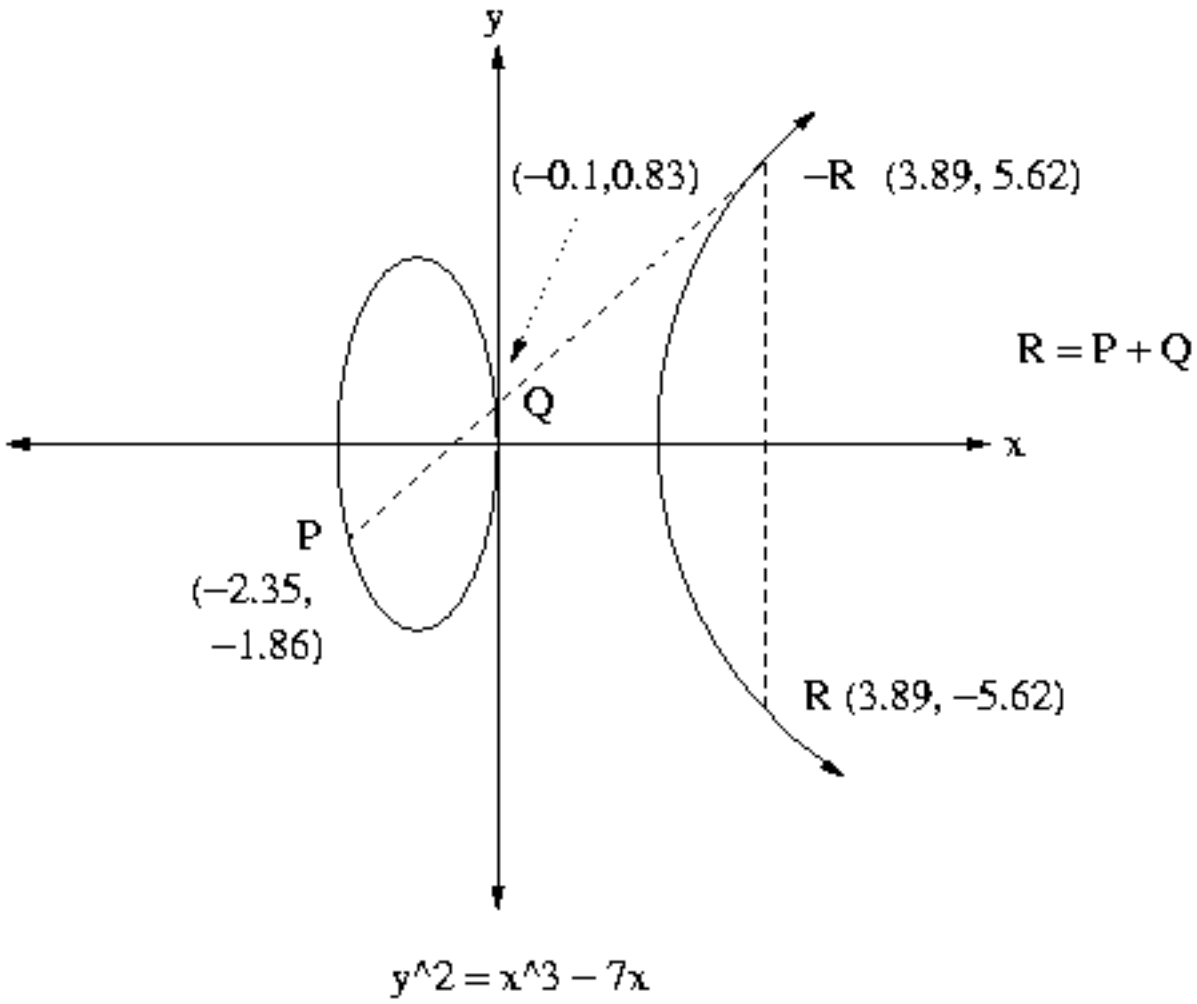


FIGURE 6: Elliptic Curve Addition Geometrically=

Like the other groups, we can define **exponentiation** operation on **$E(Z_p)$** of a point **P**, which is written as multiples of **P** (e.g., **iP**) because the group operation is additive. So the discrete log problem reduces to:

Definition: Given an elliptic curve **E** defined over **Z_p**, a point **P** in **$E(Z_p)$** of order **n**, and a point **Q** in **$E(Z_p)$**, determine the integer **i**, **0 < i < n-1**, such that **Q = iP**, provided that such an integer exists.

Given these facts, we can design a simple cryptosystem by generating public/private pairs as follows:

- Select a suitable point **P** in **$E(Z_p)$** of **order** **n**.

- Select a unique and unpredictable integer **i**, **0 < i < n-1**.
- Compute **Q = iP**.
- Publish the public key as **(E,P,n,Q)** and keep the private key as **i**.

and redefining the earlier cryptosystems under this scheme. In fact, this is the scheme followed in Elliptic Curve Digital Signature Algorithm (ECDSA) that is currently being standardized within ANSI X9F1 and IEEE P1363 standard committees. As a comparison, an ECDSA system with a point P whose order is a 160-bit prime offers approximately the same level of security as a RSA system with 1024-bit modulus n.

Applications of Public-key Cryptosystems

Modern e-commerce, as popularized on the Internet, runs on open networks. Consequently, there are significantly higher risks than with older forms of e-commerce that run on closed networks such as electronic data interchange (EDI) or electronic fund transfer (EFT). Open networks pose many security challenges including the integrity of the information being transmitted, the confidentiality of private or personal information, the authenticity of the communicating parties, and the assurance that the communicating parties have the authority to enter into the transactions. Not surprisingly, new e-commerce applications are appearing every day from researchers and industries. Some of the simple applications and their underlying technology are:

- secure mail supported by a protocol called SMIME,
- secure web browsing and purchasing supported by secure sockets layer (SSL) protocol, developed by Netscape, and transport layer security (TLS) proposed by IETF,
- secure credit card transactions supported by secure electronic transactions (SET) protocol, designed by VISA and MasterCard.

Needless to say, public-key cryptography is the technological cornerstone of all these systems. It enables the design of secure e-commerce engines by providing the following infrastructures:

Authentication

Authentication is the process of verifying one's identity. Public key cryptosystems make it possible for an easy way of authenticating a person. This is very useful in e-commerce applications involving a client and a server, where the client wants to

confirm a server's identity before sending a credit card number over the network. As an example, the following is a way SSL uses public key cryptography techniques for authentication (where Alice wants to authenticate Bob):

1. Alice sends some random message **r** to Bob.
2. Bob encrypts **r** using his private key and sends to Alice.
3. Alice decrypts the data sent by Bob using Bob's public key and compares it with **r** (that she sent) to find out the identity of Bob.

Because Bob's private key is known only to Bob, he could only have encrypted the message **r** if the comparison at Alice's end was successful.

Key Exchange

Key exchange was the first application of public key cryptography. The requirement is to share a key, which can later be used to encrypt and decrypt messages by a secret key system like DES. The following process can be used:

1. Alice generates a random number **k** to be used as the session key.
2. Alice encrypts **k** using Bob's public key and sends it to Bob.
3. Upon receipt, Bob decrypts the message using his private key and gets back **k**.

Because Bob's private key is known only to Bob, an eavesdropper, Eve, listening to the transaction can only get the encrypted message, but not the key **k**.

Confidentiality

Confidentiality means protecting data from eavesdroppers. A public key cryptosystem can be used in much the same way as above to encrypt messages and send across. But in practice, this type of a scheme is not used as secret key cryptosystems are 100-200 times faster than any public-key system. So public-key cryptosystems are used to set up the framework for a secret key system by exchanging the session key.

Non-repudiation and Digital Signatures

This requirement comes from the issue of denying previous actions. Public Key systems provide digital signatures that can not be repudiated. The concept is much similar to the real world process of signing a document to make it legal and to specify the person responsible for the document. A digital signature scheme consists of two components:

a *signing algorithm* and a *verification algorithm*. The process is very similar to that of authentication.

1. Alice encrypts the message **m** with her private key ($E_{Kpri(Alice)}(m)$).
2. Alice encrypts the resulting data using Bob's public key and sends to Bob ($c = E_{Kpub(Bob)}(E_{Kpri(Alice)}(m))$).
3. Bob recovers **m** by doing $m = D_{Kpub(Alice)}(D_{Kpri(Bob)}(c))$.

Since Bob is able to recover **m** using Alice's public key, he can verify that Alice signed it with her private key. Also, the signature depends on the contents of the message, hence no one can use the signature with another document.

In practice, however, a little different scheme is used. Because with the above protocol, anybody can be Alice. All you need is a public key and a private key. Then you can lie to Bob, say you are Alice and provide your public key instead of Alice's. So the practical systems like SSL [**9**] use an object called a **certificate**. **Certificate authorities (CAs)** are entities that validate identities and issue certificates. They can be either independent third parties (e.g., Verisign) or organizations using their own certificate-issuing server software (e.g., Netscape Certificate Server). A certificate, among other things, contains the certificate issuer's name, the entity for whom the certificate is being issued (the subject), the public key of the subject, and some timestamps. The certificate is signed using the certificate issuer's (a trusted authority) private key. Certificates are a standard way of binding a public key to a name. Then, typically in a client-server transaction, the server can send its certificate to the client instead of just the public key. The client can first decrypt the certificate using the trusted CA's public key, and confirm the server's identity.

Refer to sites [**4**, **9**, **10**, **13**] for more information on these concepts and the associated protocols.

Data Integrity

Digital signatures can also be used to validate the integrity of signed data. SSL uses a mathematical function called a **one-way hash** (also called a **message digest**). A one-way hash is a number of fixed length computed from any message text **m** with the following characteristics:

- The value of the hash is unique for the hashed data. Any change in the data, even deleting, or altering a single character, results in a different value.
- The content of the hashed data can not, for all practical purposes, be deduced from the hash, which is why it is called one-way.

Using the one-way hash, the following process can be used to protect data integrity:

1. Alice computes a one-way hash of the message **m** and encrypts it with her private key.
2. Alice sends both the text **m** and the encrypted hash to Bob.
3. Upon receiving, Bob extracts **m** and computes the same one-way hash on **m**. He also extracts the encrypted hash and decrypts it using Alice's public key.
4. Bob then compares the computed hash value with the received hash value. If they are identical, then he validates data integrity.

If the two hashes match in the above protocol, then Bob can confirm that the data has not changed since it was signed.

Conclusions

There is no gainsaying the fact that cryptography plays an essential role in protecting the privacy of electronic information against threats from a variety of potential attackers. Public key cryptography, in particular, is the most important technology in modern cryptographic schemes to address issues like key management, authentication, non-repudiation and digital signatures. However, it is important to note that technology readily available today makes smart attacks against these systems both fast and cheap. As a result, cryptosystems with smaller key lengths offer virtually no security. Considering that, symmetric-key systems offer an advantage over the public-key systems. Private keys in public-key systems are much larger (e.g. 1024 bits for RSA) than secret keys in symmetric cryptosystems (e.g. 64 or 128 bits). This is because the most efficient attack on symmetric-key systems is an exhaustive key search, but all known public-key systems are subject to "short-cut" attacks (e.g., factoring in RSA) that are more efficient than exhaustive search. As the technology advances, more efficient public key systems will be in practice as is evident by the recent interest in elliptic curve cryptography. There has also been a vast amount of research going on in the field of quantum cryptography [1]: applying laws of quantum physics (uncertainty principle, entanglement and no-clone properties of subatomic particles) to the field of cryptography. Though it is still in the experimental stage, we hope to see a lot of it in

future.

Appendix A: Primitives for Cryptography

name=con-gcd>Greatest Common Divisor (gcd): If **d** is a divisor of **a** and also a divisor of **b**, then **d** is a **common divisor** of **a** and **b**. The greatest common divisor of two integers **a** and **b**, not both zero, is the largest of the common divisors of **a** and **b**. One of the first ever algorithms of computational science, Euclid's algorithm, finds the gcd of two integers by recursively applying the formula **gcd(a,b) = gcd(b, a mod b)**. Consequently, we can extend the Euclid's algorithm to compute the integer coefficients **x** and **y** such that **ax + by = d**. This is known as the *extended Euclid's algorithm*.

Relatively Prime Integers: Two integers **a**, **b** are said to be relatively prime if **gcd(a, b) = 1** (e.g., 10 and 21 are relatively prime). Consequently, an integer **a** is **prime** if **a** is relatively prime to all integers **j: 0 < j < a**.

Congruence: The concept of congruence is to do arithmetic with the remainders obtained upon division by different integers. If **a**, **b**, and **n** are integers, we say that **a** is congruent to **b** modulo **n** [denoted by **a º b (mod n)**] if **n** divides **(a-b)** or in more specific terms, **a = b + kn**, for some integer **k**, called a multiple. **n** is called the modulus. Modular math means that the only numbers under consideration are the non-negative integers less than the modulus, **n**. Another aspect of modular math is the concept of **"modular inverse."** Two numbers are modular inverse of each other if their product(mod n) equals 1 (e.g., 7 * 343 = 2401, but if the modulus is 2400, the result is 7*343 mod 2400 = 1 (mod 2400). Here 7 is called the inverse of 343 modulo 2400 or the other way).

Solving Modular Linear Equations: Finding solutions to the equation **ax º b (mod n)** is an important practical problem, which will be used heavily in cryptosystems (like **RSA**). A well-known theorem in number theory states that [**2**]:

"The equation **ax º b (mod n)** is solvable for the unknown **x** if and only if **gcd(a,n)** divides **b**."

If that is the case, we can find a solution to the equation in the following way: let **d = gcd(a,n)**. Apply **Extended Euclid's Algorithm** to find integer coefficients x' and y' such that $ax' + ny' = d$. Then the solution is given by $x_0 = x'(b/d) \bmod n$. This is an

integer, because **d** divides **b**.

Group: a set of elements with a custom-defined arithmetic operation. This operation along with the unique ruleset abided by the elements makes it possible for information security. Two most important groups used in cryptography are $\mathbf{Z_n}$, the additive group of integers modulo number **n**, and $\mathbf{Z_p^*}$, the multiplicative group of integers modulo a prime number **p**.

Abelian Group: a group **G** with the custom-defined arithmetic operation, **#**, commutative (i.e., for two elements **a** and **b**, **a # b = b # a**). Abelian groups are the groups used in every cryptosystem because of their cyclic property. Both the groups $\mathbf{Z_n}$ and $\mathbf{Z_p^*}$ are abelian groups.

Properties of Group Elements: There can be other operations defined on the group elements, and one of the most fundamental is **exponentiation**. Exponentiation is the repeated use of the operation **#** on the group element (e.g., for a group element **a**, $\mathbf{a^m}$ **= a # a # a... (m times)**). The **order** of an element **a** in **G** is the smallest positive integer **m** such that $\mathbf{a^m = 1}$. An element **g** in **G** having order **n** (where **n** is the number of elements in **G**) is called a **primitive element** (also called a **generator**) of **G**, as repeated exponentiation of **g** yields all the elements of **G** (i.e., **g** is a primitive element if and only if $\mathbf{\{g^i: 0 <= i < n-1\} = G}$). **Euler's Phi Function** (**phi(n)**) for a group $\mathbf{Z_n^*}$ is the number of integers less than **n** that are **relatively prime** to **n**. **Euler's theorem** [4] states that:

$$\mathbf{a^{phi(n)} \equiv 1 \pmod{n},\ 1 <;= a < n.}$$

If **p** is a prime, **phi(p) = p-1**. Another important property of Euler's phi function is its **"multiplicative"** nature. If **p** and **q** are relatively prime, then **phi(pq) = phi(p)*phi(q)**. Hence for primes **p** and **q** and **n = pq**, **phi(n) = (p-1)(q-1)**. This fact will be used in RSA algorithm.

Field: a set of elements with two custom-defined arithmetic operations, most commonly, addition and multiplication. A field is finite if it has a finite number of elements. The most commonly used finite field in cryptography is $\mathbf{F_p}$ (the combination of $\mathbf{Z_n}$ and $\mathbf{Z_p^*}$).

Refer to [3,6] for more information on these.

References

1

Bennett, H.C., Bessette, F., and Brassard, G. Experimental Quantum Cryptography. *EUROCRYPT'90*. Arhus, Denmark, 1990, pp. 253-265.

2

Certicom Corp. ECC Tutorials and Whitepapers. ***http://www.certicom.com/ecc/index.htm***.

3

Cormen, T.H., Leiserson, C.E., and Rivest, R.L. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

4

Cryptography Research Inc. Crypto Resources. ***http://www.cryptography.com/resources/index.html***

5

Diffie, W. and Hellman, M.E. New Directions in Cryptography. *IEEE Transactions in Information Theory.* IT-22(1976), pp. 644-654.

6

Kumanduri, R., and Romero, C. *Number Theory with Computer Applications*. Prentice Hall, Upper Saddle River, NJ, 1998.

7

Menezes, A., Oorschot, P.V., and Vanstone, S. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, 1996.

8

Miller, V. Uses of Elliptic Curves in Cryptography. *Advances in Cryptology CRYPTO'85*, Lecture Notes in Computer Science, 218(1986), Springer-Verlag, pp. 417-426.

9

Netscape Communications Corporation. Security Online Page. ***http://developer.netscape.com/tech/security***.

10

O'reilly Online Catalog. Web Security Landscape (Sample Chapter 1 from the Book "Web Security and Commerce" by Simson Garfinkel and Gene Spafford). ***http://oreilly.com/catalog/websec/chapter/ch01.html***, 1999.

11

Rivest, R. Cryptography and Security Page. ***http://theory.lcs.mit.edu/***

~rivest/crypto-security.html.

12

Rivest, R.L., Shamir, A., and Adleman, A. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM.* 21 (1978), pp. 120-126.

13

RSA Security Inc. Cryptography FAQ. ***http://www.rsa.com/rsalabs/faq/***.

14

Schneier, B. *Applied Cryptography*. 2nd Edition, John Wiley & Sons, New York, 1996.

15

Stinson, D.R. *Cryptography, Theory and Practice*. CRC Press, Boca Raton, FL, 1995.

Biography

Pradosh Kumar Mohapatra received his B.E. degree in computer science from Birla Institute of Technology and Science, Pilani, India and worked with Novell for two years. At Novell, he concentrated in areas like Network Device Drivers, Network File Systems, and Internet Security. Currently, he is a graduate student at the University of Illinois at Chicago. He can be reached at: **pmohapat@eecs.uic.edu**. Web site: **http://www.eecs.uic.edu/~pmohapat**.