Lecture Note 5

PUBLIC-KEY CRYPTOGRAPHY

Sourav Mukhopadhyay

Cryptography and Network Security - MA61027

- Modern/Public-key cryptography started in 1976 with the publication of the following paper.
  - W. Diffie and M.E.Hellman. "New directions in cryptography". IEEE Transactions on Information Theory, 22 (1976) 644-654.
- Right up to modern times all cryptosystems are based on the elementary tools of substitution and permutation.
- Public-key algorithms are based on mathematical functions and are asymmetric in nature, involving the use of two keys, as opposed to conventional single key encryption.

- Several misconceptions are held about p-k:
  1. That p-k encryption is more secure from cryptanalysis than conventional encryption. In fact the security of any system depends on key length and the computational work involved in breaking the cipher.
  2. That p-k encryption has superseded single key encryption. This is unlikely due to the increased processing power required.
  3. That key management is trivial with public key cryptography, this is not correct.

- The concept of P-K evolved from an attempt to solve two problems, *key distribution* and the development of *digital signatures*.
- In 1976 Whitfield Diffie and Martin Hellman achieved great success in developing the conceptual framework.
- For conventional encryption the same key is used for encryption and decryption - not a necessary condition.
- Instead, possible to develop a cryptographic system that relies on one key for encryption and a different but related key for decryption.

- These algorithms have the following important characteristic:

  It is *computationally infeasible* to determine the decryption key given only knowledge of the algorithm and the encryption key.

- In addition, some algorithms such as RSA, also exhibits the following characteristics:

  Either of the two related keys can be used for encryption, with the other used for decryption.

- Fig 1 illustrates the P-K process. The steps are:
  1. Each system generates a pair of keys.
  2. Each system publishes its encryption key (public key) keeping its companion key private.
  3. If A wishes to send a message to B it encrypts the message using B's public key.
  4. When B receives the message, it decrypts the message using its private key. No one else can decrypt the message because only B knows its private key.

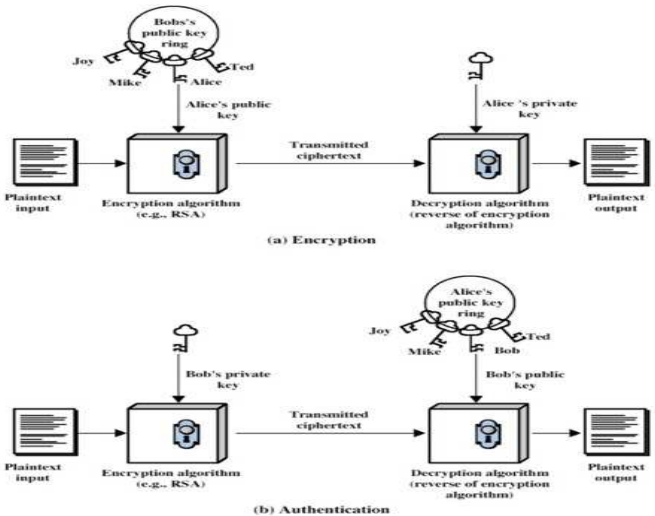

Figure 1: Public Key Cryptography.

- Considering P-K in more detail we have a source A that produces plaintext X destined for B (figure 2).
- B generates a pair of keys KU_b (a public key) and KR_b (a private key).
- With X and KU_b as inputs, A forms the ciphertext Y:

$$Y = E_{KU_b}(X)$$

- The intended receiver B is able to invert the transformation with his private key:

$$X = D_{KR_b}(Y).$$

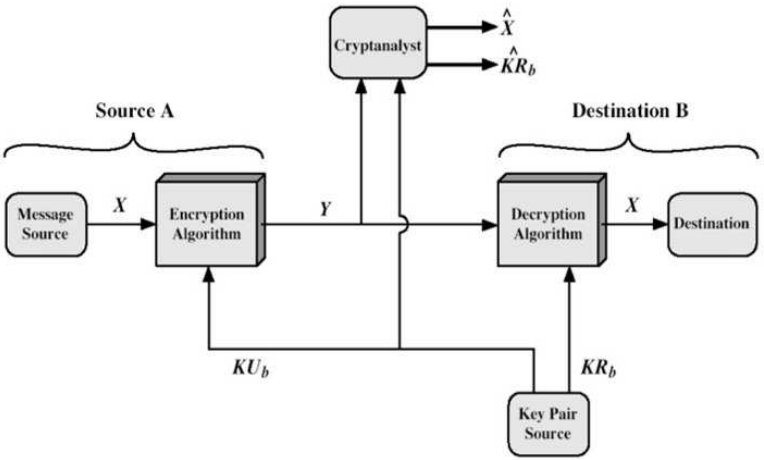

Figure 2: Public Key Cryptography: Secrecy.

Authentication

- As previously mentioned, either key may be used for encryption with the other used for subsequent decryption. This facilitates a different form of scheme as shown in figure 3.
- In this case A prepares a message to B using his *private key* to encrypt and B can decrypt it using A's *public key.*

$$Y = E_{KR_a}(X)$$

$$X = D_{KU_a}(Y).$$

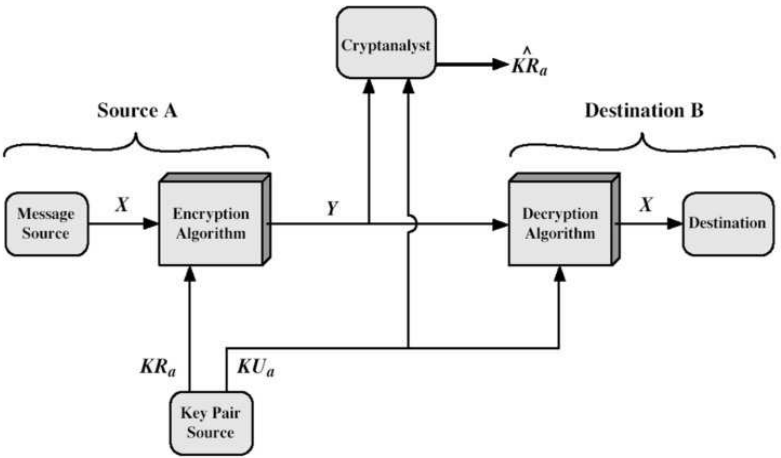

Figure 3: Public Key Cryptography: Authentication.

Authentication

- As the message was prepared using A's private key it could only have come from A therefore the entire message serves as a **digital signature**.
- This scheme does not provide confidentiality because everyone has access to A's public key.
- It is not efficient because B must maintain/store both the ciphertext (as proof of authenticity) and the decoded plaintext (for practical use of the document).

- A more efficient way of achieving the same result is to encrypt a small block of bits that are a function of the document.
- This block, called an **authenticator**, must have the property that it is infeasible to change the document without changing the authenticator.
- If the authenticator is encrypted using the senders private key then it serves as a signature that verifies the origin, content and sequencing of the document.

Confidentiality and Authentication

- If both are required, the double use of the public key scheme (figure 4) facilitates this.
- Here we have: $Z = E_{KU_b}[E_{KR_a}(X)]$

$$X = D_{KU_a}[D_{KR_b}(Z)] \qquad (1)$$

- In this case the message is first encrypted using the senders private key, providing the digital signature.
- Then a second encryption is performed using the receivers public key, which delivers confidentiality.
- The disadvantage with this scheme is that the public-key algorithm which is complex must be used four times.

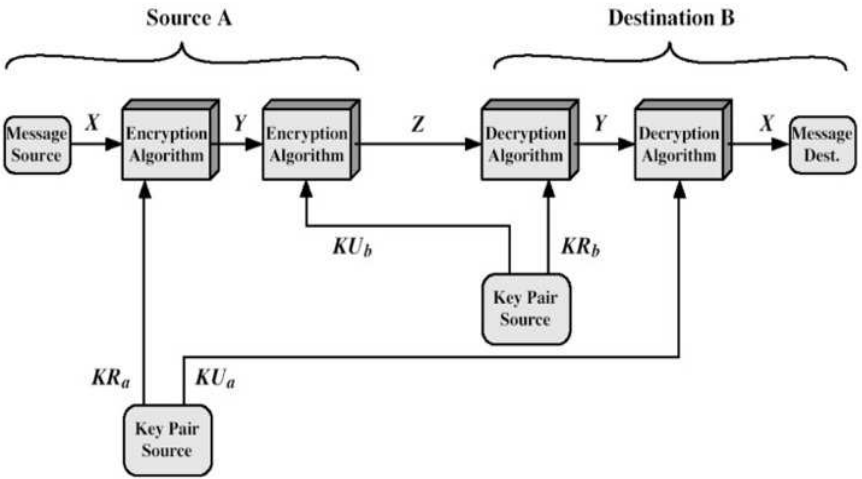

Figure 4: Public Key Cryptography: Secrecy and Authentication.

Applications for P-K cryptosystems

- In broad terms, we can classify the use of public-key cryptosystems into three categories:
  1. Encryption/decryption: where the sender encrypts the message with the receivers public key.
  2. Digital signature: where the sender "signs" a message with his private key.
  3. Key exchange: several approaches later.
- However, not all algorithms are suitable for all three applications. Some can only be used for say digital signature. RSA however can be used for all three as will be seen.

Requirements of the algorithm

- The requirements of any P-K system were laid out by Diffie and Hellman:
  1. It is computationally easy for party B to generate a key pair (public (KU) and private (KR)).
  2. It is computationally easy for sender A knowing KU_b and the message to be encrypted to generate the corresponding ciphertext $C = E_{KU_b}(M)$.
  3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using his private key (KR_b) to recover the original message.
     $M = D_{KR_b}(C) = D_{KR_b}[E_{KU_b}(M)]$.

4. It is computationally infeasible for an opponent, knowing the public key KU_b, to determine the private key KR_b.
5. It is computationally infeasible for an opponent, knowing KU_b and C to recover the plaintext message M.
6. A sixth requirement that, although useful, is not necessary for all public-key applications - the encryption and decryption can be applied in either order: $M = E_{KR_b}[D_{KR_b}(M)]$.

- These are formidable requirements as is evidenced by the fact that only one algorithm (RSA) has received widespread acceptance in over 20 years. The requirements boil down to the need for a **trapdoor one-way function**.

- A **one-way function** is a function that maps a domain into a range such that every function value has a unique inverse, with the condition that the calculation of the function is easy whereas the calculation of the inverse is infeasible:

$$\begin{aligned} Y &= f(X) && \text{easy} \\ X &= f^{-1}(Y) && \text{infeasible} \end{aligned}$$

- "Easy" is defined to mean a problem that can be solved in *polynomial time* as a function of input length (n).
- For example, the time to compute is proportional to n^a where a is a fixed constant.

- "Infeasible" is not as well defined however. Generally we can say that if the effort to solve is greater than polynomial time the problem is infeasible, e.g. if time to compute is proportional to 2^n.
- Trapdoor one-way functions are a family of invertible functions f_k such that
  - $Y = f_k(X)$ is easy if k and X known,
  - $X = f_k^{-1}(Y)$ is easy if k and Y are known, and
  - $X = f_k^{-1}(Y)$ is infeasible if Y is known but k is not known.
- The development of a practical public-key scheme depends on the discovery of a suitable trapdoor one-way function.

The Knapsack Algorithm

- Many algorithms have been proposed for P-K, and have subsequently been broken.
- The most famous of these was proposed by Merkle and Hellman.
- The problem deals with determining which of a set of objects are in a container, say a knapsack. Of the list of say six objects of different weights shown below, which subset is in the knapsack if it weighs S?

Object 1	455 g
Object 2	341 g
Object 3	284 g
Object 4	132 g
Object 5	82 g
Object 6	56 g

- Given that the weight of the knapsack is $S = 821$ grams, the problem is to determine which of the items are in the knapsack.
- The problem shown here is simple but when the number of items is increased (> 100) it becomes *computationally infeasible*.
- So what we have is six different objects with six different weights.
- The knapsack weighs nothing itself but with a selected number of objects in it weighs (say) 821 grams. Which objects does it contain?

Subset Sum Problem

- Merkle-Hellman Knapsack cryptosystem is based on the *Subset Sum* problem defined as follows.
- **Problem Instance:** $I = (a_1, a_2, \ldots, a_n, S)$, where $a_1, \ldots, a_n$ and S are positive integers. The a_i's are called sizes and S is called the target sum.
- **Question:** Is there a 0-1 vector $X = (x_1, x_2, \ldots, x_n)$ such that

$$\sum_{i=1}^{n} a_i x_i = S?$$

- Subset sum problem is a *decision problem* (i.e., we are required only to answer "yes" or "no").
- We now rephrase the problem slightly, so that in any instance where the answer is "yes" we are required to find the desired vector X (which may not be unique), this is called Subset Sum *search* problem.
- Subset sum problem is one of the so-called *NP-complete* problems, i.e., there is no polynomial-time algorithm that solves it.

- But even if a problem has no polynomial-time algorithm to solve it in general, this does not rule out the possibility that certain special cases can be solved in polynomial time.
- We define a list of sizes, $(a_1, \ldots, a_n)$ to be *superincreasing* if

$$a_j > \sum_{i=1}^{j-1} a_i$$

for $2 \leq j \leq n$. If the list of sizes in superincreasing, then Subset sum search problem can be solved in time $O(n)$, and a solution X (if it exists) must be unique.

Algorithm 1 for the subset sum problem

1. **for** $i = n$ **downto** 1 **do**
2. **if** $S \geq a_i$ **then**
3. $S = S - a_i$
4. $x_i = 1$
5. **else**
6. $x_i = 0$
7. **if** $S = 0$ **then**
8. $X = (x_1, \ldots, x_n)$ is the solution
9. **else**
10. there is no solution

- Merkle's contribution was to show how to turn the knapsack problem into a scheme for encryption and decryption.
- In other words how to incorporate "trapdoor" information which enabled the easy solution of the knapsack problem.

- Suppose we wish to send messages in blocks of n bits. We define the following:
  - Cargo vector: $a = (a_1, a_2, \ldots, a_n)$, where a_i is an integer.
  - Plaintext message block $x = (x_1, x_2, \ldots, x_n)$, where x_i is a binary digit.
  - Corresponding ciphertext S:

$$S = \mathbf{a} \cdot \mathbf{x} = \sum_{i=1}^{n} (a_i x_i).$$

- The vector $\mathbf{a}$ is considered to be a list of potential elements to be put into the knapsack with each vector element equal to each weight of the element.
- The message block $\mathbf{x}$ is considered to be a selection of elements of the cargo vector in the knapsack.
- Each element is set equal to 1 if the corresponding element is in the knapsack and 0 if it is not.
- The product S is simply the sum of the selected item's weights (i.e. the weight of the contents of the knapsack).

- As an example lets take a cargo vector as follows:

$$\begin{aligned} a &= (455,341,284,132,82,56) \\ x &= (x_1, x_2, x_3, x_4, x_5, x_6) \quad : \text{a six bit binary number} \\ S &= 821 \end{aligned}$$

- For encryption $\mathbf{a}$ is used as the public key.
- The person sending the message $\mathbf{x}$ performs $S = \mathbf{a} \cdot \mathbf{x}$ and sends S as the ciphertext. The receiving party must recover $\mathbf{x}$ from S and $\mathbf{a}$.

- Two requirements are as follows:
  1. That there be a unique inverse for each value of S. For example if $S = 3$ and $\mathbf{a} = (1, 3, 2, 5)$ then the problem would have two solutions, $\mathbf{x} = (1, 0, 1, 0)$ and $\mathbf{x} = (0, 1, 0, 0)$. The value of $\mathbf{a}$ must be chosen so that each combination of elements yields a unique value of S.
  2. That decryption is hard in general but easy if special knowledge is available. For large values of n the knapsack problem is hard in general. If however we impose the superincreasing condition then we have an easy solution.

- For example, consider the vector $\mathbf{a}' = (171, 197, 459, 1191, 2410)$ which satisfies the superincreasing condition.
- Suppose we have $S' = \mathbf{a}' \cdot \mathbf{x}' = 3798$.
- Because $3798 > 2410$, a_5 must be included ($x_5 = 1$) because without a_5 all the other elements cannot contribute enough to add up to 3798 (or 2410).
- Now consider $3798 - 2410 = 1388$. The number 1388 is bigger than 1191 so a_4 must be included ($x_4 = 1$).
- Continuing in this fashion we find that $x_3 = 0$, $x_2 = 1$ and $x_1 = 0$.

- This system would be completely insecure since anyone (including Oscar) can decrypt a message that is encrypted in this way.
- What Merkle and Hellman did was to tie an easy superincreasing knapsack problem to a hard general knapsack problem.
- The strategy therefore is to transform the list of sizes in such a way that it is no longer superincreasing.

- Bob (receiver) will be able to apply an inverse transformation to restore the superincreasing list of sizes.
- On the other hand Oscar, who does not know the transformation that was applied, is faced with what looks like a general, apparently difficult, instance of the subset sum problem when he tries to decrypt a ciphertext.
- One suitable type of transformation is a *modular transformation*

- Suppose we choose an easy knapsack vector $\mathbf{a}'$ with n elements. Also select two integers m and ω such that m is greater than the sum of the elements, and ω is relatively prime to m, that is:

$$m > \sum_{i=1}^{n} a'_i, \qquad gcd(\omega, m) = 1$$

- Now, we construct a hard knapsack vector, $\mathbf{a}$, by multiplying an easy vector $\mathbf{a}'$ by $\omega \pmod{m}$:

$$\mathbf{a} = \omega \mathbf{a}' \pmod{m}$$

- The vector $\mathbf{a}$ will in general not be superincreasing and therefore can be used to construct hard knapsack problems.
- However, knowledge of ω and m enables the conversion of this hard knapsack problem to an easy one.
- To see this, first observe that since ω and m are relatively prime, there exists a unique multiplicative inverse ω^{-1}, modulo m. Therefore:

$$\omega^{-1}\mathbf{a} = \mathbf{a}' \pmod{m}.$$

- We can now state the knapsack scheme. The ingredients are as follows:
  1. $\mathbf{a}'$, a superincreasing vector (private, chosen).
  2. m, an integer larger than the sum of all a'_j's (private, chosen).
  3. ω, an integer relatively prime to m (private, chosen).
  4. ω^{-1}, the inverse of ω, modulo m (private, calculated).
  5. $\mathbf{a}$, equal to $\omega\mathbf{a}' \pmod{m}$ (public, calculated).
- The private key consists of the triple $(\omega^{-1}, m, \mathbf{a}')$ and the public key consists of the value of $\mathbf{a}$.

- Suppose user A has published his public key $\mathbf{a}$ and that user B wishes to send a message $\mathbf{x}$ to A. B calculates the sum $S = \mathbf{a} \cdot \mathbf{x}$.
- The determination of $\mathbf{x}$ given S and $\mathbf{a}$ is difficult so this is a secure transmission.
- However, on receipt, user A is able to decrypt easily. Defining $S' = \omega^{-1} S \pmod{m}$ we have the following:

$$\begin{aligned} S &= \mathbf{a} \cdot \mathbf{x} = \omega \mathbf{a}' \cdot \mathbf{x} \\ S' &= \omega^{-1} S \pmod{m} \\ &= \omega^{-1} \omega \mathbf{a}' \cdot \mathbf{x} \pmod{m} \\ &= \mathbf{a}' \cdot \mathbf{x} \end{aligned}$$

Merkle-Hellman Knapsack Cryptosystem

- Let $\mathbf{a}' = (a'_1, \ldots, a'_n)$ be a superincreasing list of integers.
- Let m and ω be two positive integers such that $m > \Sigma_{i=1}^{n} a'_i$ and $gcd(\omega, m) = 1$.
- For $1 \leq i \leq n$, define $a_i = \omega a'_i \bmod m$, and denote $\mathbf{a} = (a_1, \ldots, a_n)$.
- Let $\mathcal{P} = \{0,1\}^n$, $\mathcal{C} = \{0, \ldots, n(m-1)\}$, and $\mathcal{K} = \{(\mathbf{a}', m, \omega, \mathbf{a})\}$. Here $\mathbf{a}$ is public and $\mathbf{a}', m, \omega$ are secret.

- For $k = (\mathbf{a}', m, \omega, \mathbf{a})$, we define the encryption function:

$$e_k(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i a_i.$$

- For a ciphertext $S \in \mathcal{C}$, i.e., $0 \leq S \leq n(m-1)$, define $S' = \omega^{-1} S \bmod m$ and solve the subset sum problem $(a'_1, \ldots, a'_n, S')$, obtaining $d_k(S) = (x_1, \ldots, x_n)$.

Example

- Suppose $\mathbf{a}' = (2, 5, 9, 21, 45, 103, 215, 450, 946)$ is the secret superincreasing list of sizes.
- Suppose $m = 2003$ and $\omega = 1289$. Then the public list of sizes is

$$\mathbf{a} = (575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570).$$

- Sender B wants to encrypt the plaintext $\mathbf{x} = (1, 0, 1, 1, 0, 0, 1, 1, 1)$, computes

$$S = 575 + 1586 + 1030 + 721 + 1183 + 1570 = 6665.$$

- When A receives the ciphertext S, he first computes

$$\begin{aligned} S' &= \omega^{-1} S \bmod m \\ &= 317 \times 6665 \bmod 2003 \\ &= 1643 \end{aligned}$$

- Then A solves the instance $I = (\mathbf{a}', S')$ of the subset sum (easy knapsack) problem using Algorithm 1.
- The plaintext $(1, 0, 1, 1, 0, 0, 1, 1, 1)$ is obtained.

- Thus we have converted the hard problem of finding $\mathbf{x}$ given S into the easy problem of finding $\mathbf{x}$ given S' and $\mathbf{a}'$.
- By the early 1980's, the Merkle-Hellman Knapsack cryptosystem had been broken by Adi Shamir (of RSA).
- All of the various knapsack systems have been shown to be insecure except the **Chor-Rivest** cryptosystem.

Some important public-key cryptosystems

Several public-key systems have been proposed, whose security rests on different computational problems and we study the computational security of public-key cryptosystem.

- RSA: The security is based on the difficulty of factoring large integers.
- Merkle-Hellman Knapsack: Security is based on the difficulty of the subset sum problem (which is NP-complete).

- McEliece: This is based on algebraic coding theory and the security is based on the problem of decoding a linear code (which is NP-complete).
- ElGamal: The security is based on the difficulty of discrete logarithm problem for finite field.
- Chor-Rivest: This also a "knapsack" type system.
- Elliptic Curve: Elliptic Curve Crytosystems (ECC) work in the domain of elliptic curves rather than finite fields. The ECC appears to remain secure for smaller keys than other public-key crytosystems.

RSA

- The RSA algorithm was developed by Ron **R**ivest, Adi **S**hamir and Len **A**dleman at MIT in 1978. Since this time it has recognised supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.
- The scheme makes use of an expression with exponential.
- For some plaintext $M < n$ and ciphertext $C < n$ we have:

$$\begin{aligned} C &= M^e \pmod{n} \\ M &= C^d \pmod{n} = (M^e)^d \pmod{n} \\ M &= M^{ed} \pmod{n} \end{aligned}$$

- Both sender and receiver know n. Sender knows the value of e and only receiver knows the value of d. To restate:

$$KU = \{e, n\} \tag{1}$$
$$KR = \{d\}$$

- For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:
  1. It is possible to find values of e, d and n such that $M^{ed} = M \pmod{n}$ for all $M < n$.
  2. It is relatively easy to calculate M^e and C^d for all values of $M < n$.
  3. It is infeasible to determine d given e and n.

- Focusing initially on the first question we need to find a relationship of the form: $M^{ed} = M \pmod{n}$.
- If we recall that Euler's theorem states that

$$a^{\phi(m)} \equiv 1 \pmod{m} \qquad \gcd(a, m) = 1 \tag{2}$$

- There is a corollary to this theorem that can be used to produce the required relationship. Given two prime numbers p and q and integers $n = pq$ and m, with $0 < m < n$, the following relationship holds:

$$m^{\phi(n)+1} \equiv m^{(p-1)(q-1)+1} \equiv m \pmod{n} \tag{3}$$

- If $\gcd(m, n) = 1$ then this holds by Euler's theorem.
- Suppose however $\gcd(m, n) \neq 1$. What does this mean?
- Well, because $n = pq$, the equality $\gcd(m, n) = 1$ is equivalent to the logical expression (m is not a multiple of p) **AND** (m is not a multiple of q).
- If m is a multiple of p then n and m share the prime factor p and are not relatively prime (the same can be said for q).
- Therefore, the expression $\gcd(m, n) \neq 1$ must be equivalent to the negation of the foregoing logical expression.

- Hence, $\gcd(m, n) \neq 1$ is equivalent to logical expression (m is a multiple of p) **OR** (m is a multiple of q).
- Looking at the case in which m is a multiple of p, so that the relationship $m = cp$ holds for some positive integer c.
- In this case we must have $\gcd(m, q) = 1$. Otherwise, we have m a multiple of p and m a multiple of q and yet $m < pq$.
- If $\gcd(m, q) = 1$ then Euler's theorem holds and

$$m^{\phi(q)} \equiv 1 \pmod{q}$$

- But then, by the rules of modular arithmetic,

$$[m^{\phi(q)}]^{\phi(p)} \equiv 1 \pmod{q}$$

$$m^{\phi(n)} \equiv 1 \pmod{q}$$

- Therefore, there is some integer k such that

$$m^{\phi(n)} = 1 + kq$$

- Multiplying each side by $m = cp$,

$$m^{\phi(n)+1} = m + kcpq = m + kcn$$

$$m^{\phi(n)+1} \equiv m \pmod{n}$$

- A similar line of reasoning is used for the case in which m is a multiple of q. Thus, equation 3 is proven.
- An alternative form of this corollary is directly relevant to RSA:

$$\begin{aligned} m^{k\phi(n)+1} &\equiv [(m^{\phi(n)})^k \times m)] \pmod{n} \\ &\equiv [(1)^k \times m)] \pmod{n} \text{ by Euler's theorem} \\ &\equiv m \pmod{n} \end{aligned} \tag{4}$$

- This can be achieved with $ed = k\phi(n) + 1 \equiv 1(\mathrm{mod}\phi(n))$
- i.e. $d \equiv e^{-1} \pmod{\phi(n)}$

- We can now state the RSA scheme. The ingredients are the following:

p, q, two primes	(private, chosen)
$n = pq$	(public, calculated)
e, with $\gcd(\phi(n), e) = 1$; $1 < e < \phi(n)$	(public, chosen)
$d \equiv e^{-1} \pmod{\phi(n)}$	(private, calculated)

- The private key consists of $\{d\}$ and public key is $\{e, n\}$.
- Suppose that user A has published his public key and that user B wishes to send the message M to A.
- B calculates $C = M^e \pmod{n}$ and transmits C.
- On receipt of the ciphertext C user A decrypts by calculating the following: $M = C^d \pmod{n}$. Figure 7 summarises the algorithm.

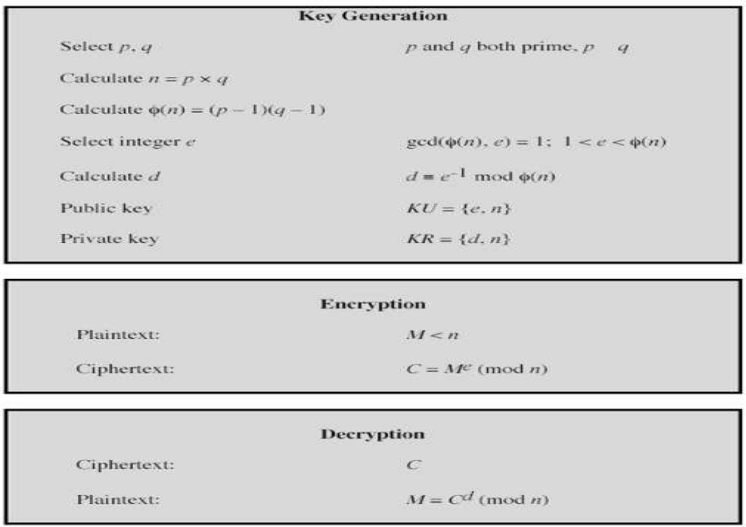

Figure 5: The RSA algorithm

Example

a) Select p=7 ,q=17

b) Calculate $n = pq = 7 \times 17 = 119$

c) Calculate $\phi(n) = (p-1)(q-1) = 96$.

d) Select e, relatively prime to and less than $\phi(n)$, say $e = 5$.

e) Determine d such that $de = 1 \pmod{96}$ and $d < 96$.

f) The correct value for d is 77 because
$77 \times 5 = 385 = 4 \times 96 + 1$ (can be calculated using the extended version of Euclid's algorithm).

g) The resulting public key is KU= $\{5, 119\}$ and private key is KR= $\{77\}$. Say the plaintext is $M = 19$. For encryption 19 is raised to the 5th power, yielding $2,476,099$. Upon division by 119, the remainder is 66, hence ciphertext sent is 66. For decryption it is determined using KR that $66^{77} \equiv 19 \pmod{119}$ so the recovered plaintext is 19.

Computational Aspects

- The complexity of the computation required boils down to two aspects:
  1. The actual encryption/decryption process.
  2. The key generation process.

Encryption and Decryption

- Both involve raising a (large) integer to a (large) integer power modulo n.
- If the exponentiation was done over the integers and then reduced modulo n, the intermediate values would be gigantic.
- Fortunately we can make use of a property of modular arithmetic:

$$[(a \bmod n).(b \bmod n)] \bmod n = (a.b) \bmod n \qquad (5)$$

- Thus, intermediate results may be reduced modulo n. This makes the calculation more practical.
- Another consideration is the efficiency of exponentiation, since with RSA we are dealing with large exponents.
- To see how efficiency might be improved consider calculating x^{16}.
- A straightforward approach is to perform 15 multiplications,
  $x^{16} = x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x.$

- However we can receive the same result with just four multiplications if we repeatedly take the square of each partial result successively forming x^2, x^4, x^8 and x^{16}.
- Note that even utilising shortcuts etc. there is a requirement for arithmetic operations with arbitrarily large integers and most computers are restricted in this capability.

- More generally, suppose we wish to find the value a^m, with a, m positive integers. If we express m as a binary number $b_k b_{k-1} \ldots b_0$, then we have the following:

$$m = \sum_{i=0}^{k} b_i 2^i = \sum_{b_i \neq 0} 2^i$$

- Therefore,

$$a^m = a^{(\sum_{b_i \neq 0} 2^i)} = \prod_{b_i \neq 0} a^{(2^i)}$$

$$a^m \bmod n = \left[\prod_{b_i \neq 0} a^{(2^i)}\right] \bmod n = \left(\prod_{b_i \neq 0} [a^{(2^i)} \bmod n]\right) \bmod n$$

- Which can be done using a square an multiply algorithm.

- The square-and-multiply algorithm to compute $a^m \bmod n$:

1. $z = 1$
2. **for** $i = k$ **downto** 0 **do**
3. $\quad z = z^2 \bmod n$
4. $\quad$ **if** $b_i = 1$ **then** $z = z \times a \bmod n$

- Let $n = 11413$, $m = 3533$ and $a = 9726$. We now compute 9726^{3533} mod 11413 using squre-and-multiply algorithm.

i	b_i	z
11	1	$1^2 \times 9726 = 9726$
10	1	$9726^2 \times 9726 = 2659$
9	0	$2659^2 = 5634$
8	1	$5634^2 \times 9726 = 9167$
7	1	$9167^2 \times 9726 = 4958$
6	1	$4958^2 \times 9726 = 7783$
5	0	$7783^2 = 6298$
4	0	$6298^2 = 4629$
3	1	$4629^2 \times 9726 = 10185$
2	1	$10185^2 \times 9726 = 105$
1	0	$105^2 = 11025$
0	1	$11025^2 \times 9726 = 5761$

- 9726^{3533} mod $11413 = 5761$

Key Generation

- Before two parties can use a public key system, each must generate a pair of keys. This involves the following tasks:
  - Determining two prime numbers p, q.
  - Selecting either e or d and calculating the other.
- Firstly, considering selection of p and q.
- Because the value $n = pq$ will be known to any opponent, to prevent the discovery of p, q through exhaustive methods, these primes must be chosen from a sufficiently large set (must be large numbers).

- On the other hand the method used for finding large primes must be reasonably efficient.
- At present there are no useful techniques that yield arbitrarily large primes.
- The procedure is to pick at random an odd number of the desired magnitude and test that it is prime. If not, repeat until a prime is found.
- A variety of tests for primality have been developed, all of which are statistical in nature.

- The tests however may be run in such a way as to attain a probability, of as near 1 as is desired, that a particular number is prime.
- One of the more efficient algorithms is the **Miller-Rabin scheme**, which performs calculations on n, the candidate prime and a randomly chosen integer a. This procedure may be repeated as required.

Summary of Key Generation

- In summary the procedure for picking a prime is as follows:
  1. Pick an odd integer n at random (e.g., using a pseudorandom number generator).
  2. Pick an integer $a < n$ at random.
  3. Perform the probabilistic primality test, (such as Miller-Rabin). If n fails the test then go to step 1.
  4. If n passes a sufficient number of tests then accept it, otherwise go to step 2.

The Security of RSA

- RSA gets its security from the difficulty of factoring large numbers.
- The public and private keys are functions of a pair of large (100 to 200 digits) prime numbers.
- Recovering the plaintext from one key and the ciphertext is equivalent to factoring the product of two primes.

- Taking a first look at cryptographic considerations. Three possible approaches include:
  1. **Brute Force**
  2. **Mathematical attacks**
  3. **Timing attacks**

1. Brute Force

- Try all possible keys. Standard defense is a large key space.
- The larger e and d are the better, so we have the following:

	5 years ago	Today
Casual use	384 bits	768 bits
Commercial use	512 bits	1024
Military Spec.	1024 bits	4096 bits

- Where the military specification is only an estimate due to this information being classified. For comparison, 512 bits is about 150 decimal digits.

2. Mathematical attacks

- Factor n into its 2 primes thus enabling calculation of $\phi(n)$ and the private key $e \equiv d^{-1} \pmod{\phi(n)}$. The best known algorithm used in factoring an integer n is time proportional to:

$$e^{\sqrt{ln(n).ln(ln(n))}} \tag{6}$$

- For a 200 digit number this would take about 1000 years on a large machine. However, there has been a lot of progress made in factorisation over the last number of years.

- Determining $\phi(n)$, given n or determining d given n and e.
- These are at least as time consuming as factoring n so the factorising performance of algorithms is used as the benchmark to evaluate the security of RSA.
- In addition to specifying n of order $150 - 200$ decimal digits, some other recommendations are that p and q should differ in length by only a few digits.
- Other constraints are also specified to ensure the difficulty of factorising is maintained.

3. Timing attacks

- These are an implementation attack that depends on the running time of an algorithm. We will look at them in more detail when we study attacks on cryptosystems.

Diffie Hellman Key Exchange

- The first published P-K algorithm appeared in the paper by Diffie and Hellman that defined public key cryptography however it is limited to the secure exchange of a secret key and not of a message.
- The security of the scheme depends on the difficulty of computing discrete logarithms.
- The Diffie-Hellman key exchange consists of two publicly known numbers: a prime number p and an integer α that is a primitive root of q.

- Suppose the users A and B wish to exchange a key.
- User A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{X_A} \bmod q$.
- Similarly user B independently selects a random integer $X_B < q$ and computes $Y_B = \alpha^{X_B} \bmod q$.
- Each side keeps the X values private and makes the Y value available publicly to the other side.
- User A computes the key as $K = (Y_B)^{X_A} \bmod q$ and user B computes the key as $K = (Y_A)^{X_B} \bmod q$.
- These two calculations produce identical results and the result is that the two sides have exchanged a secret key.

- This can be seen because:

$$\begin{aligned} K &= (Y_B)^{X_A} \bmod q \\ &= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\ &= (\alpha^{X_B})^{X_A} \bmod q \\ &= (\alpha^{X_A})^{X_B} \bmod q \\ &= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\ &= (Y_A)^{X_B} \bmod q \end{aligned}$$

- Furthermore because X_A and X_B are private, an opponent is forced to take a discrete logarithm to determine the key.

- For example, attacking the secret key of user B the opponent must compute:

$$X_B = ind_{\alpha,q}(Y_B)$$

- Where $ind_{\alpha,q}(Y_B)$ is the discrete logarithm, or index, of Y_B for the base α mod q.
- The scheme can be summarised as shown in figure 6

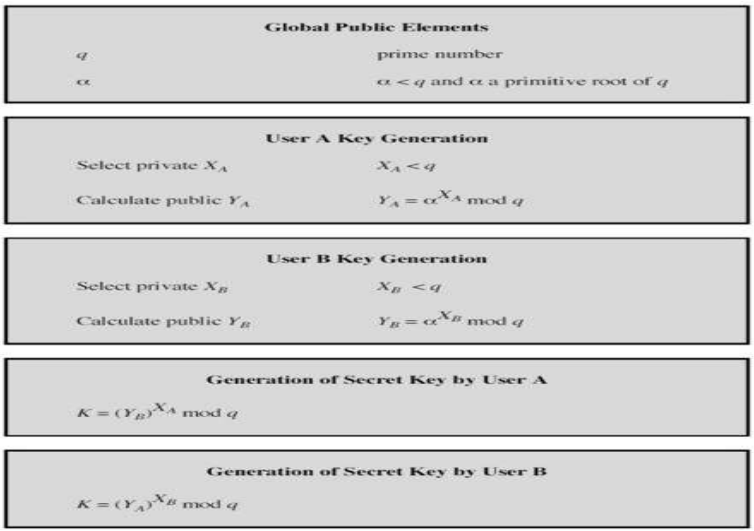

Figure 6: The Diffie Hellman Key Exchange Algorithm.

- For example lets say we have the values $q = 353$ and a primitive root $\alpha = 3$.
- We can see that $\alpha = 3$ is a primitive root of $q = 353$ due to the following reasoning.
- If α is a primitive root of a prime q then the set of numbers $\{\alpha, \alpha^2, \ldots, \alpha^{\phi(q)}\}$ are distinct modulo q and hence form the set $\{1, 2, \ldots (q-1)\}$ in some order.
- In this case $\alpha = 3$ and it can be seen to be a primitive root of $q = 353$ as
  $\{3 \bmod 353, 3^2 \bmod 353, \ldots, 3^{353} \bmod 353, \}$ which contains all the elements of $\{1, 2, \ldots 352\}$.

- Suppose A and B select the private keys $X_A = 97$ and $X_B = 233$ respectively.
- To calculate the secret key K user A calculates:

$$Y_A = \alpha^{X_A} \bmod q = 3^{97} \bmod 353 = 40$$

- Similarly user B calculates

$$Y_B = \alpha^{X_B} \bmod q = 3^{233} \bmod 353 = 248$$

- Then we have
  $K = 248^{97} \bmod 353 = 40^{233} \bmod 353 = 160.$

- We assume the attacker would have q, α, Y_A, Y_B which for this example might be enough using a brute force approach. However with large numbers this becomes impractical.

ElGamal Cryptosystem

- The ElGamal Cryptosystem is based on Discreate Logarithm problem
- The ElGamal Cryptosystem is non-deterministic, since the ciphertext depends on both the plaintext x and on the random value k chosen by encryptor. So there will be many ciphertexts that are encryptions of the same plaintext.

The discrete logarithm problem in Z_p

- **Problem Instance:** $I = (p, \alpha, \beta)$, where p is prime, $\alpha \in Z_p$ is a primitive element, and $\beta \in Z_p^*$.
- **Objective:** Find the unique integer a, $0 \leq a \leq p-2$, such that

$$\alpha^a \equiv \beta \pmod{p}.$$

We will denote this integer a by $\log_\alpha \beta$.

ElGamal Public-key Cryptosystem in Z_p^*

- Let p be a prime such that the discrete log problem in Z_p is intractable, and let $\alpha \in Z_p^*$ be a primitive element.
- Let $\mathcal{P} = Z_p^*$, $\mathcal{C} = Z_p^* \times Z_p^*$, and define

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

- The values p, α and β are public, and a is secret.

- $K = (p, \alpha, a, \beta)$, for a (secret) random number $k \in Z_{p-1}$, define

$$e_K(x, k) = (y_1, y_2),$$

where

$$y_1 = \alpha^k \bmod p$$

and

$$y_2 = x\beta^k \bmod p.$$

- For $y_1, y_2 \in Z_p^*$, define

$$d_K(y_1, y_2) = y_2(y_1^a)^{-1} \bmod p.$$

- The plaintext x is "masked" by multiplying it by β^k, yielding y_2. The value α^k is also transmitted as part of the ciphertext. The decryptor, who knows the secret exponent a, can compute β^k from α^k. Then he can "remove the mask" by dividing y_2 by β^k to obtain the plaintext x.

Example

- Suppose $p = 2579, \alpha = 2, a = 765$, and hence

$$\beta = 2^{765} \bmod 2579 = 949.$$

- Now, suppose that Alice wishes to send the message $x = 1299$ to Bob. Say $k = 853$ is the random integer she chooses. Then she compute

$$y_1 = 2^{853} \bmod 2579 = 435$$

and

$$y_2 = 1299 \times 949^{853} \bmod 2579 = 2396.$$

- When Bob receives the ciphertext $y = (435, 2396)$, he compute

$$x = 2396 \times (435^{765})^{-1} \bmod 2579 = 1299,$$

which was the plaintext that Alice encrypted.

Algorithm for the Discrete Log Problem

- Given $\beta \in Z_p^*$, find the unique exponent $a, 0 \leq a \leq p-1$, such that $\alpha^a \equiv \beta \pmod{p}$.
- Clearly, the discrete logarithm (DL) problem can be solved by exhaustive search in $O(p)$ time.
- Other algorithms to the solve the DL problem.
  - *–Shanks' algorithm*
  - *–Pohlig-Hellman algorithm*
  - *–Index Calculus method*

The discrete logarithm problem in $(G, \circ)$

- **Problem Instance:** $I = (G, \alpha, \beta)$, where G is a finite group with group operation $\circ$, $\alpha \in G$ and $\beta \in H$, where $H = \{\alpha^i : i \geq 0\}$ is the subgroup generated by α.
- **Objective:** Find the unique integer a such that $0 \leq a \leq |H| - 1$ and $\alpha^a = \beta$, where the notation α^a means

$$\alpha \circ \alpha \circ \ldots \circ \alpha \qquad (a \text{ times})$$

Generalized ElGamal Public-key Cryptosystem

- Let G be a finite group with group operation $\circ$, and let $\alpha \in G$ be an element such that the discrete log problem in H is intractable, where $H = \{\alpha^i : i \geq 0\}$ is the subgroup generated by α.
- Let $\mathcal{P} = G$, $\mathcal{C} = G \times G$, and define

$$\mathcal{K} = \{(G, \alpha, a, \beta) : \beta = \alpha^a\}.$$

- The values α and β are public, and a is secret.

- $K = (G, \alpha, a, \beta)$, for a (secret) random number $k \in Z_{|H|}$, define
$$e_K(x, k) = (y_1, y_2),$$
where
$$y_1 = \alpha^k$$
and
$$y_2 = x \circ \beta^k.$$
- For a ciphertext $y = (y_1, y_2)$, define
$$d_K(y_1, y_2) = y_2 \circ (y_1^a)^{-1}.$$

Elliptic Curves over the Reals

- **Definition:** Let $a, b \in R$ be constants such that $4a^3 + 27b^2 \neq 0$. A non-singular elliptic curve is the set E of solutions $(x, y) \in R \times R$ to the equation

$$y^2 = x^3 + ax + b,$$

together with a special point $\mathcal{O}$ called the point at infinity.

- It can be shown that the condition $4a^3 + 27b^2 \neq 0$ is necessary and sufficient to ensure that the equation $x^3 + ax + b = 0$ has three distinct roots (which may be real or complex numbers).
- If $4a^3 + 27b^2 = 0$ then the corresponding elliptic curve is called a *singular elliptic curve*.
- Suppose E is a non-singular elliptic curve. We will define "+" operation over E which makes E into an abelian group.
- *Identity element:* The point at infinity, $\mathcal{O}$ is the identity element, so $P + \mathcal{O} = \mathcal{O} + \mathcal{P} = \mathcal{P}$ for all $P \in E$

Addition operation

- Suppose $P, Q \in E$, where $P = (x_1, y_1)$ and $Q = (x_2, y_2)$. We consider three cases:
  1. $x_1 \neq x_2$
  2. $x_1 = x_2$ and $y_1 = -y_2$
  3. $x_1 = x_2$ and $y_1 = y_2$

Case 1: $x_1 \neq x_2$

- We define L to be line through P and Q. L intersects E in the two points P and Q, and it is easy to see that L will intersect E in one futher point, which we call R'. If we reflect R in the x-axis, then we get a point which we name R. We define $P + Q = R$.
- $R = (x_3, y_3)$, where $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$, and $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$

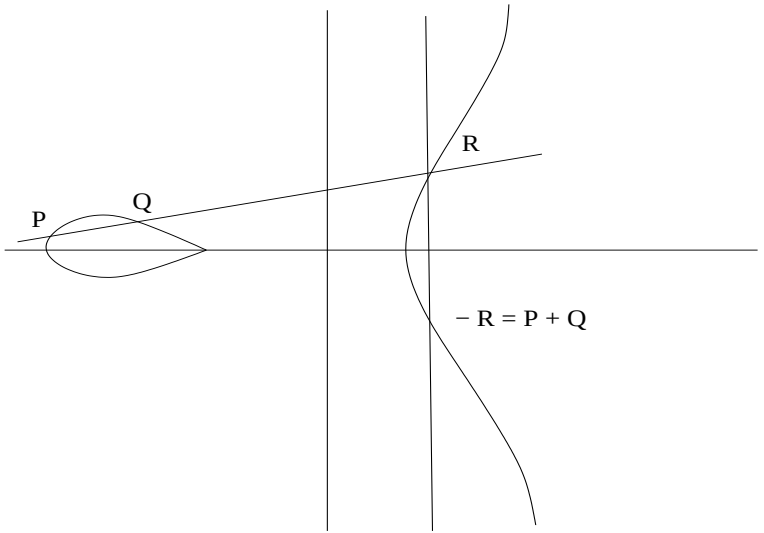

Figure 7: Chord and Tangent law

Case 2: $x_1 = x_2$ and $y_1 = -y_2$

- We define $(x, y) + (x, -y) = \mathcal{O}$ for all $(x, y) \in E$.
- $Q = -P$, then $P + Q = \mathcal{O}$, *i.e.* $\mathcal{O}$ is the third point of intersection of any vertical line through P (or Q) with the curve E. Any vertical line through P (or Q) meets the curve E at infinity. This is why $\mathcal{O}$ is called point at infinity. $\mathcal{O}$ serves as the identity of the abelian group E.
- Therefore $P = (x, y)$ and $-P = (x, -y)$ are inverses with respect to the elliptic curve addition operation.

Case 3: $x_1 = x_2$ and $y_1 = y_2$

- Here we are adding a point $P = (x_1, y_1)$ to itself. We can assume that $y_1 \neq 0$, for then we would be in case 2. Case 3 is handled much like case 1, expect that we define L to be tangent to E at the point P.
- If $P = (x_1, y_1) \in E$ then $P + P = (x_3, y_3)$, where $x_3 = \lambda^2 - 2x_1$, $y_3 = \lambda(x_1 - x_3) - y_1$, and $\lambda = \frac{3x_1^2 + a}{2y_1}$

- If $P = (x_1, y_1) \in E$, $Q = (x_2, y_2) \in E$, $P \neq -Q$, then $P + Q = (x_3, y_3)$, where $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$, and

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \text{ if } P \neq Q;$$
$$\lambda = \frac{3x_1^2 + a}{2y_1} \text{ if } P = Q.$$

(E, +) is an abelian group

- At this point the following properties of the addition operation, as defined above, should be clear:
  1. addition is closed on the set E
  2. addition is commutative
  3. $\mathcal{O}$ is an identity with respect to addition, and
  4. every point on E has an inverse with respect to addition
- The proof of assiociativity is quite messy by algebric method. But this proof can be made simpler by using some results from geometry.

Elliptic Curves Modulo a Prime

- **Definition:** Let $p > 3$ be prime. The elliptic curve $y^2 = x^3 + ax + b$ over Z_p is the set of solutions $(x, y) \in Z_p \times Z_p$ to the congruence

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

where $a, b \in Z_p$ are constants such that $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$, together with a special point $\mathcal{O}$ called the point at infinity.

- If $P = (x_1, y_1) \in E$, $Q = (x_2, y_2) \in E$, $P \neq -Q$, then $P + Q = (x_3, y_3)$, where $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$, and

$$\begin{aligned} \lambda &= (y_2 - y_1)(x_2 - x_1)^{-1} && \text{if } P \neq Q; \\ \lambda &= (3x_1^2 + a)(2y_1)^{-1} && \text{if } P = Q. \end{aligned}$$

- To determine points on E we look at each possible $x \in Z_p$ and compute $x^3 + ax + b \pmod{p}$ which is y^2 and then need to check whether this is a quadratic residue module p.

Quadratic Residue Modulo p

- **Definition:** Let p be an odd prime and x is an integer, $1 \leq x \leq p-1$. x is defined to a quadratic residue modulo p if the congruence $y^2 \equiv x \pmod{p}$ has a solution $y \in Z_p$.
- Example: The quadratic residues modulo 11 are $1, 3, 4, 5$ and 9. Note that $(\pm 1)^2 = 1, (\pm 5)^2 = 3, (\pm 2)^2 = 4, (\pm 4)^2 = 5$ and $(\pm 3)^2 = 9$ (where all arithmetic is in Z_{11}).

- **Problem:** An add prime p, and an integer x such that $1 \leq x \leq p-1$. Is x is a quadratic residue modulo p?
- **Eular's Criterion:** x is a quadratic residue modulo p *if and only if*

$$x^{\frac{(p-1)}{2}} \equiv 1 \pmod{p}.$$

- Suppose z is a quadratic residue and $p \equiv 3 \pmod{4}$. Then, the two square roots of z modulo p are $\pm z^{(p+1)/4} \bmod p$.

Example

- Let E be the elliptic curve $y^2 = x^3 + x + 6$ over Z_{11}.
- For each possible $x \in Z_{11}$, compute $x^3 + x + 6 \bmod 11$.
- For a given x, we can test to see if $z = x^3 + x + 6 \bmod 11$ is a quadratic residue by applying Eular's criterion.
- We have that the square roots of a quadratic residue z are

$$\pm z^{(11+1)/4} \bmod 11 = \pm z^3 \bmod 11.$$

- Points on the elliptic curve $y^2 = x^3 + x + 6$ over Z_{11}:

x	$x^3 + x + 6 \bmod 11$	quadratic residue?	y
0	6	no	
1	8	no	
2	5	yes	4, 7
3	3	yes	5, 6
4	8	no	
5	4	yes	2, 9
6	8	no	
7	4	yes	2, 9
8	9	yes	3, 8
9	7	no	
10	4	yes	2, 9

- E has 13 points on it including $\mathcal{O}$
- We take a point $\alpha = (2, 7)$ and compute the "power" of α (which we will write as multiples of α, since the group operation is additive).
- To compute $2\alpha = (2, 7) + (2, 7)$, we first compute

$$\lambda = (3 \times 2^2 + 1)(2 \times 7)^{-1} \bmod 11 = 2 \times 3^{-1} \bmod 11$$

$$= 2 \times 4 \bmod 11 = 8.$$

- Then we have $x_3 = 8^2 - 2 - 2 \bmod 11 = 5$ and $y_3 = 8(2 - 5) - 7 \bmod 11 = 2$, so $2\alpha = (5, 2)$.

- The next multiple would be $3\alpha = 2\alpha + \alpha = (5,2) + (2,7)$.

$\alpha = (2,7)$	$2\alpha = (5,2)$	$3\alpha = (8,3)$
$4\alpha = (10,2)$	$5\alpha = (3,6)$	$6\alpha = (7,9)$
$7\alpha = (7,2)$	$8\alpha = (3,5)$	$9\alpha = (10,9)$
$10\alpha = (8,8)$	$11\alpha = (5,9)$	$12\alpha = (2,4)$

- $\alpha = (2,7)$ is a primitive element.
- We now look at an example of ElGamal encryption and decryption using elliptic curve of this example.

- $\alpha = (2, 7)$ and Bob's (the receiver) private key is 7, so

$$\beta = 7\alpha = (7, 2).$$

Thus the encryption operation is

$$e_K(x, k) = (k(2, 7), x + k(7, 2)),$$

where $x \in E$ and $0 \leq k \leq 12$, and the decryption operation is

$$d_K(y_1, y_2) = y_2 - 7y_1.$$

- Suppose Alice (the sender) wishes to encrypt the plaintext $x = (10, 9)$ (which is a point on E). If she chooses the random value $k = 3$, then she will compute

$$y_1 = 3(2, 7) = (8, 3)$$

and

$$y_2 = (10, 9) + 3(7, 2) = (10, 9) + (3, 5) = (10, 2).$$

- Hence $y = ((8,3),(10,2))$. Now if Bob receives the ciphertext y, he decrypts it as follows:

$$x = (10,2) - 7(8,3) = (10,2) - (3,5)$$

$$= (10,2) + (3,6) = (10,9).$$

- Hence, the decryption yields the correct plaintext.