COURSE- FILE
UNIT-1
Basic Cryptographic Techniques, Computational Complexity, Finite Fields, Number Theory, DES and AES, Public Key Cryptosystems, Traffic Confidentiality ,Cryptanalysis, Intractable (Hard) Problems, Hash Functions, OSI Security Architecture Privacy of Data.

BASIC CRYTOGRAPHY TECHNIQUES
Introduction to Cryptography Basic Principles
Whenever we come across the term cryptography, the first thing and probably the only thing that comes to our mind is private communication through encryption. There is more to cryptography than just encryption. In this article, we will try to learn the basics of cryptography.
The Basic Principles

1. Encryption
In a simplest form, encryption is to convert the data in some unreadable form. This helps in protecting the privacy while sending the data from sender to receiver. On the receiver side, the data can be decrypted and can be brought back to its original form. The reverse of encryption is called as decryption. The concept of encryption and decryption requires some extra information for encrypting and decrypting the data. This information is known as key. There may be cases when same key can be used for both encryption and decryption while in certain cases, encryption and decryption may require different keys.

2. Authentication
This is another important principle of cryptography. In a layman's term, authentication ensures that the message was originated from the originator claimed in the message. Now, one may think how to make it possible? Suppose, Alice sends a message to Bob and now Bob wants proof that the message has been indeed sent by Alice. This can be made possible if Alice performs some action on message that Bob knows only Alice can do. Well, this forms the basic fundamental of Authentication.

3. Integrity
Now, one problem that a communication system can face is the loss of integrity of messages being sent from sender to receiver. This means that Cryptography should ensure that the messages that are received by the receiver are not altered anywhere on the communication path. This can be achieved by using the concept of cryptographic hash.

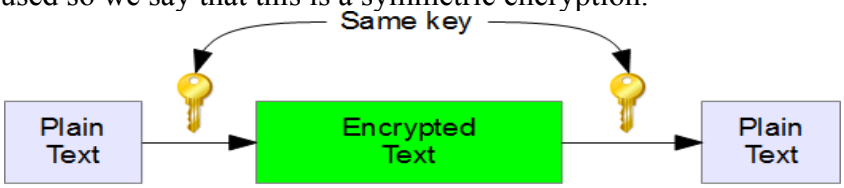
4. Non Repudiation
What happens if Alice sends a message to Bob but denies that she has actually sent the message? Cases like these may happen and cryptography should prevent the originator or sender to act this way. One popular way to achieve this is through the use of digital signatures.

Types of Cryptography
There are three types of cryptography techniques :
- Secret key Cryptography

- Public key cryptography
- Hash Functions

1. Secret Key Cryptography

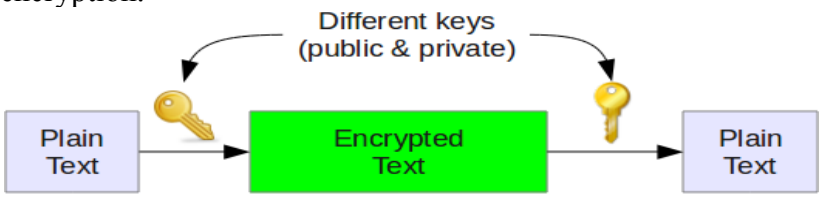
This type of cryptography technique uses just a single key. The sender applies a key to encrypt a message while the receiver applies the same key to decrypt the message. Since only single key is used so we say that this is a symmetric encryption.



The biggest problem with this technique is the distribution of key as this algorithm makes use of single key for encryption or decryption.

2. Public Key Cryptography

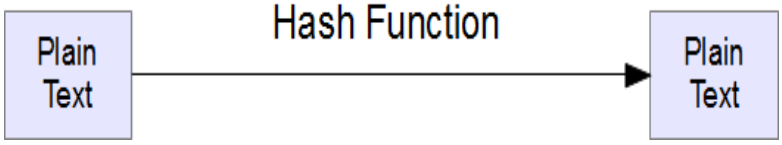
This type of cryptography technique involves two key crypto system in which a secure communication can take place between receiver and sender over insecure communication channel. Since a pair of keys is applied here so this technique is also known as asymmetric encryption.



In this method, each party has a private key and a public key. The private is secret and is not revealed while the public key is shared with all those whom you want to communicate with. If Alice wants to send a message to bob, then Alice will encrypt it with Bob's public key and Bob can decrypt the message with its private key.

This is what we use when we setup public key authentication in openssh to login from one server to another server in the backend without having to enter the password.

3. Hash Functions



This technique does not involve any key. Rather it uses a fixed length hash value that is computed on the basis of the plain text message. Hash functions are used to check the integrity of the message to ensure that the message has not be altered,compromised or affected by virus.

So we see that how different types of cryptography techniques (described above) are used to implement the basic principles that we discussed earlier. In the future article of this series, we'll cover more advanced topics on Cryptography.

Computation Complexity :

Computational complexity theory is a branch of the theory of computation in theoretical computer science that focuses on classifying computational problems according to their inherent difficulty, and relating those classes to each other. A computational problem is understood to be a task that is in principle amenable to being solved by a computer, which is equivalent to stating that the problem may be solved by mechanical application of mathematical steps, such as analgorithm.

A problem is regarded as inherently difficult if its solution requires significant resources, whatever the algorithm used. The theory formalizes this intuition, by introducing mathematical models of computation to study these problems and quantifying the amount of resources needed to solve them, such as time and storage. Other complexity measures are also used, such as the amount of communication (used in communication complexity), the number of gates in a circuit (used in circuit complexity) and the number of processors (used in parallel computing). One of the roles of computational complexity theory is to determine the practical limits on what computers can and cannot do.

Closely related fields in theoretical computer science are analysis of algorithms and computability theory. A key distinction between analysis of algorithms and computational complexity theory is that the former is devoted to analyzing the amount of resources needed by a particular algorithm to solve a problem, whereas the latter asks a more general question about all possible algorithms that could be used to solve the same problem. More precisely, it tries to classify problems that can or cannot be solved with appropriately restricted resources. In turn, imposing restrictions on the available resources is what distinguishes computational complexity from computability theory: the latter theory asks what kind of problems can, in principle, be solved algorithmically.

Finite field:

In mathematics, a finite field or Galois field (so-named in honor of Évariste Galois) is a field that contains a finite number of elements. As with any field, a finite field is a set on which the operations of multiplication, addition, subtraction and division are defined and satisfy certain basic rules. The most common examples of finite fields are given by the integers mod p when p is a prime number.

The number of elements of a finite field is called its *order*. A finite field of order q exists if and only if the order q is a prime power p^k (where p is a prime number and k is a positive integer). All fields of a given order are isomorphic. In a field of order p^k, adding p copies of any element always results in zero; that is, the characteristic of the field is p.

In a finite field of order q, the polynomial $X^q - X$ has all q elements of the finite field as roots. The non-zero elements of a finite field form a multiplicative group. This group is cyclic, so all non-zero elements can be expressed as powers of a single element called a primitive element of the field (in general there will be several primitive elements for a given field.)

A field has, by definition, a commutative multiplication operation. A more general algebraic structure that satisfies all the other axioms of a field but isn't required to have a commutative multiplication is called a division ring (or sometimes *skewfield*). A finite division ring is a finite field by Wedderburn's little theorem. This result shows that the finiteness condition in the definition of a finite field can have algebraic consequences.

Finite fields are fundamental in a number of areas of mathematics and computer science, including number theory, algebraic geometry, Galois theory, finite geometry, cryptography and coding theory.

Definitions, first examples, and basic properties[edit]

A finite field is a finite set on which the four operations multiplication, addition, subtraction and division (excluding division by zero) are defined, satisfying the rules of arithmetic known as the field axioms. The simplest examples of finite fields are the prime fields: for each prime number p, the field GF(p) (also denoted $\mathbb{Z}/p\mathbb{Z}$, $\mathbb{Z}_p$, or $\mathbb{F}_p$) of order (that is, size) p is easily constructed as the integers modulo p.

The elements of a prime field may be represented by integers in the range $0, \ldots, p-1$. The sum, the difference and the product are computed by taking the remainder by p of the integer result. The multiplicative inverse of an element may be computed by using the extended Euclidean algorithm (see Extended Euclidean algorithm § Modular integers).

Let F be a finite field. For any element x in F and any integer n, let us denote by $n \cdot x$ the sum of n copies of x. The least positive n such that $n \cdot 1 = 0$ must exist and is prime; it is called the *characteristic* of the field.

If the characteristic of F is p, the operation makes F a GF(p)-vector space. It follows that the number of elements of F is p^n.

For every prime number p and every positive integer n, there are finite fields of order p^n, and all fields of this order are isomorphic (see § Existence and uniqueness below). One may therefore identify all fields of order p^n, which are therefore unambiguously denoted $\mathbb{F}_{p^n}$, $\mathbb{F}_{p^n}$ or GF(p^n), where the letters GF stand for "Galois field".[1]

The identity

$$(x+y)^p = x^p + y^p$$

is true (for every x and y) in a field of characteristic p. (This follows from the fact that all, except the first and the last, binomial coefficients of the expansion of $(x+y)^p$ are multiples of p).

For every element x in the prime field GF(p), one has $x^p = x$ (This is an immediate consequence of Fermat's little theorem, and this may be easily proved as follows: the equality is trivially true for $x = 0$ and $x = 1$; one obtains the result for the other elements of GF(p) by applying the above identity to x and 1, where x successively takes the values 1, 2, ..., $p-1$ modulo p.) This implies the equality

for polynomials over GF(p). More generally, every element in GF(p^n) satisfies the polynomial equation $x^{pn} - x = 0$.

Any finite field extension of a finite field is separable and simple. That is, if E is a finite field and F is a subfield of E, then $E$is obtained from F by adjoining a single element whose minimal polynomial is separable. To use a jargon, finite fields areperfect.

Existence and uniqueness[edit]

Let $q = p^n$ be a prime power, and F be the splitting field of the polynomial

over the prime field GF(p). This means that F is a finite field of lowest order, in which P has q distinct roots (the roots are distinct, as the formal derivative of P is equal to −1). Above identity shows that the sum and the product of two roots of $P$are roots of P, as well as the multiplicative inverse of a root of P. In other word, the roots of P form a field of order q, which is equal to F by the minimality of the splitting field.

The uniqueness up to isomorphism of splitting fields implies thus that all fields of order q are isomorphic.

In summary, we have the following classification theorem first proved in 1893 by E. H. Moore:[2]

The order of a finite field is a prime power. For every prime power q there are fields of order q, and they are all isomorphic. In these fields, every element satisfies

and the polynomial $X^q - X$ factors as

It follows that GF(p^n) contains a subfield isomorphic to GF(p^m) if and only if m is a divisor of n; in that case, this subfield is unique. In fact, the polynomial $X^{pm} - X$ divides $X^{pn} - X$ if and only if m is a divisor of n.

Number theory:
Introduction to Number Theory

Modular Arithmetic

- modular arithmetic is 'clock arithmetic'
- a congruence a = b mod n says when divided by n that a and b have the same remainder
  - 100 = 34 mod 11
  - usually have 0<=b<=n-1
  - -12mod7 = -5mod7 = 2mod7 = 9mod7
  - b is called the residue of a mod n

- can do arithmetic with integers modulo n with all results between 0 and n

addition

a+b mod n

subtraction

a-b mod n = a+(-b) mod n

multiplication

a.b mod n

- derived from repeated addition
- can get a.b=0 where neither a,b=0
  - eg 2.5 mod 10

division

a/b mod n

- is multiplication by inverse of b: $a/b = a.b^{-1} \bmod n$
- if n is prime $b^{-1} \bmod n$ exists s.t $b.b^{-1} = 1 \bmod n$
  - eg $2.3=1 \bmod 5$ hence $4/2=4.3=2 \bmod 5$

- integers modulo n with addition and multiplication form a commutative ring with the laws of

  Associativity

  (a+b)+c = a+(b+c) mod n

  Commutativity

  a+b = b+a mod n

  Distributivity

  (a+b).c = (a.c)+(b.c) mod n

- also can chose whether to do an operation and then reduce modulo n, or reduce then do the operation, since reduction is a homomorphism from the ring of integers to the ring of integers modulo n
  - a+/-b mod n = [a mod n +/- b mod n] mod n
  - (the above laws also hold for multiplication)
- if n is constrained to be a prime number *p* then this forms a Galois Field modulo p denoted GF(p) and all the normal laws associated with integer arithmetic work

Exponentiation in GF(p)

- many encryption algorithms use exponentiation - raising a number a (base) to some power b (exponent) mod p
  - $b = a^e \bmod p$
- exponentiation is basically repeated multiplication, which take s O(n) multiples for a number n
- a better method is the square and multiply algorithm[1]

let base = a, result =1
for each bit ei (LSB to MSB) of exponent
  if ei=0 then

```
    square base mod p
  if ei=1 then
    multiply result by base mod p
    square base mod p (except for MSB)
required ae is result
```

- only takes $O(\log_2 n)$ multiples for a number n

see Sebbery p9 Fig2.1 + example

Discrete Logarithms in GF(p)

- the inverse problem to exponentiation is that of finding the discrete logarithm of a number modulo p
  - find x where $a^x = b \bmod p$

Seberry examples p10

- whilst exponentiation is relatively easy, finding discrete logarithms is generally a hard problem, with no easy way
- in this problem, we can show that if p is prime, then there always exists an a such that there is always a discrete logarithm for any b!=0
  - successive powers of a "generate" the group mod p
- such an a is called a primitive root and these are also relatively hard to find

Greatest Common Divisor

- the greatest common divisor (a,b) of a and b is the largest number that divides evenly into both a and b
- Euclid's Algorithm is used to find the Greatest Common Divisor (GCD) of two numbers a and n, a<n
  - use fact if a and b have divisor d so does a-b, a-2b

GCD (a,n) is given by:

```
  let g0=n
    g1=a
    gi+1 = gi-1 mod gi
  when gi=0 then (a,n) = gi-1
```

eg find (56,98)

```
  g0=98
  g1=56
  g2 = 98 mod 56 = 42
  g3 = 56 mod 42 = 14
  g4 = 42 mod 14 = 0
```

hence (56,98)=14

see Seberry Fig 2.2 p11 GCD alg [2]

Inverses and Euclid's Extended GCD Routine

- unlike normal integer arithmetic, sometimes a number in modular arithmetic has a unique inverse
  - a^{-1} is inverse of a mod n if $a.a^{-1} = 1 \bmod n$
  - where a,x in {0,n-1}
  - eg $3.7 = 1 \bmod 10$
- if (a,n)=1 then the inverse always exists
- can extend Euclid's Algorithm to find Inverse by keeping track of $g_i = u_i.n + v_i.a$
- Extended Euclid's (or Binary GCD) Algorithm to find Inverse of a number a mod n (where (a,n)=1) is:

Inverse(a,n) is given by:
g0=n u0=1 v0=0
g1=a u1=0 v1=1
let
y = gi-1 div gi
gi+1 = gi-1 - y.gi = gi-1 mod gi
ui+1 = ui-1 - y.ui
vi+1 = vi-1 - y.vi
when gi=0 then Inverse(a,n) = vi-1

Example

eg: want to find Inverse(3,460):

i	y	g	u	v
0	-	460	1	0
1	-	3	0	1
2	153	1	1	-153
3	3	0	-3	460

hence Inverse(3,460) = -153 = 307 mod 460

see Seberry Fig 2.3 p14 Inverse alg

Euler Totient Function [[phi]](n)

- if consider arithmetic modulo n, then a reduced set of residues is a subset of the complete set of residues modulo n which are relatively prime to n
  - eg for n=10,
  - the complete set of residues is {0,1,2,3,4,5,6,7,8,9}
  - the reduced set of residues is {1,3,7,9}
- the number of elements in the reduced set of residues is called the Euler Totient function [[phi]](n)
- there is no single formula for [[phi]](n) but for various cases count how many elements are excluded:

p (p prime)	[[phi]](p) =p-1
pr (p prime)	[[phi]](p) =pr-1(p-1)
p.q (p,q prime)	[[phi]](p.q) =(p-1)(q-1)

see Seberry Table 2.1

- several important results based on [[phi]](n) are:
- Theorem (Euler's Generalization)
  - let gcd(a,n)=1 then
  - $a^{[[phi]](n)} \bmod n = 1$

- Fermat's Theorem
  - let p be a prime and gcd(a,p)=1 then
  - $a^{p-1} \bmod p = 1$

- Algorithms to find Inverses $a^{-1} \bmod n$
  1. search 1,...,n-1 until an a^{-1} is found with $a.a^{-1} \bmod n$
  2. if [[phi]](n) is known, then from Euler's Generalization
     - $a^{-1} = a^{[[phi]](n)-1} \bmod n$
  3. otherwise use Extended Euclid's algorithm for inverse

Computing with Polynomials in GF(qn)

- have seen arithmetic modulo a prime number GF(p)
- also can do arithmetic modulo q over polynomials of degree n, which also form a Galois Field $GF(q^n)$
- its elements are polynomials of degree (n-1) or lower
  - $a(x)=a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+...+a_1x+a_0$
- have residues for polynomials just as for integers
  - $p(x)=q(x)d(x)+r(x)$
  - and this is unique if deg[r(x)]<deg[d(x)]
- if r(x)=0, then d(x) divides p(x), or is a factor of p(x)
- addition in $GF(q^n)$ just involves summing equivalent terms in the polynomial modulo q (XOR if q=2)
  - $a(x)+b(x)=(a_{n-1}+b_{n-1})x^{n-1}+...+(a_1+b_1)x+(a_0+b_0)$

Multiplication with Polynomials in GF(qn)

- multiplication in $GF(q^n)$ involves
  - multiplying the two polynomials together (cf longhand multiplication; here use shifts & XORs if q=2)
  - then finding the residue modulo a given irreducible polynomial of degree n
- an irreducible polynomial d(x) is a 'prime' polynomial, it has no polynomial divisors other than itself and 1
- modulo reduction of p(x) consists of finding some r(x) st: $p(x)=q(x)d(x)+r(x)$
  - nb. in $GF(2^n)$ with $d(x)=x^3+x+1$ can do simply by replacing x^3 with x+1

- eg in $GF(2^3)$ there are 8 elements:
  - 0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1
- with irreducible polynomial $d(x)=x^3+x+1$* arithmetic in this field can be summarised as:

Seberry Table 2.3 p20

- can adapt GCD, Inverse, and CRT algorithms for $GF(q^n)$
  - [[phi]](p(x)) = 2^n-1 since every poly except 0 is relatively prime to p(x)
- arithmetic in $GF(q^n)$ can be much faster than integer arithmetic, especially if the irreducible polynomial is carefully chosen
  - eg a fast implementation of $GF(2^{127})$ exists
- has both advantages and disadvantages for cryptography, calculations are faster, as are methods for breaking
-

DES and AES

Data Encryption Standard (DES):

a) Data Encryption Standard (DES) DES (Data Encryption Standard) algorithm purpose is to provide a standard method for protecting sensitive commercial and unclassified data. In this same key used for encryption and decryption process. DES algorithm consists of the following steps

i. Encryption 1. DES accepts an input of 64-bit long plaintext and 56-bitkey (8 bits of parity) and produce output of 64 bit block. 2. The plaintext block has to shift the bits around. 3. The 8 parity bits are removed from the key by subjecting the key to its Key Permutation. 4. The plaintext and key will processed by following i. The key is split into two 28 halves

ii. Each half of the key is shifted (rotated) by one or two bits, depending on the round.

iii. The halves are recombined and subject to a compression permutation to reduce the key from 56 bits to 48 bits. This compressed keys used to encrypt this round's plaintext block.

iv. The rotated key halves from step 2 are used in next round.

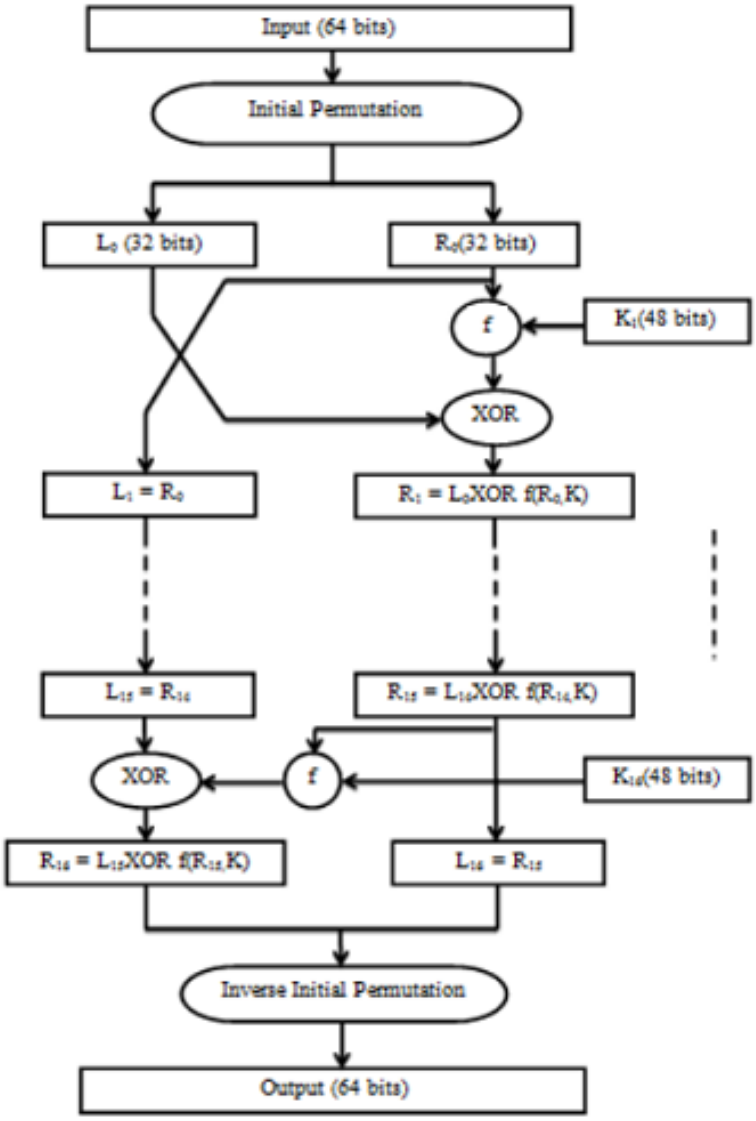
v. The data block is split into two 32-bit halves.

vi. One half is subject to an expansion permutation to increase its size to 48 bits.

vii. Output of step 6 is exclusive-OR'ed with the 48- itcompressed key from step 3.

viii. Output of step 7 is fed into an S-box, which substitutes key bits and reduces the 48-bit block back down to 32-bits.

ix. Output of step 8 is subject to a P-box to permute the bits. M



The output from the P-box is exclusive-OR'ed with other half of the data block. k. The two data halves are swapped and become the next round's input. © 2013 Global Journals Inc. (US) Global Journal of Computer Science and Technology Volume XIII Issue XV Version I 16 ( ) Year 013 2 E Figure 1 : Diagram of DES Algorithm\

b) Advanced Encryption Standard (AES) Advanced Encryption Standard (AES) algorithm not only for security but also for great speed. Both hardware and software implementation are faster

still. New encryption standard recommended by NIST to replace DES. Encrypts data blocks of 128 bits in 10, 12 and 14 round depending on key size as shown in Figure - 2. It can be implemented on various platforms specially in small devices. It is carefully tested for many security applications.

i. Algorithm Steps : These steps used to encrypt 128-bit block 1. The set of round keys from the cipher key. 2. Initialize state array and add the initial round key to the starting state array. 3. Perform round = 1 to 9 : Execute Usual Round. 4. Execute Final Round. 5. Corresponding cipher text chunk output of Final Round Step A Study of Encryption Algorithms AES, DES and RSA for Security

ii. Usual Round : Execute the following operations which are described above. 1. Sub Bytes 2. Shift Rows 3. Mix Columns 4. Add Round Key , using K(round)

iii. Final Round: Execute the following operations which are described above. 1. Sub Bytes 2. Shift Rows 3. Add Round Key, using K(10)

iv. Encryption : Each round consists of the following four steps:

i Sub Bytes : The first transformation, Sub Bytes, is used at the encryption site. To substitute a byte, we interpret the byte as two hexadecimal digits.

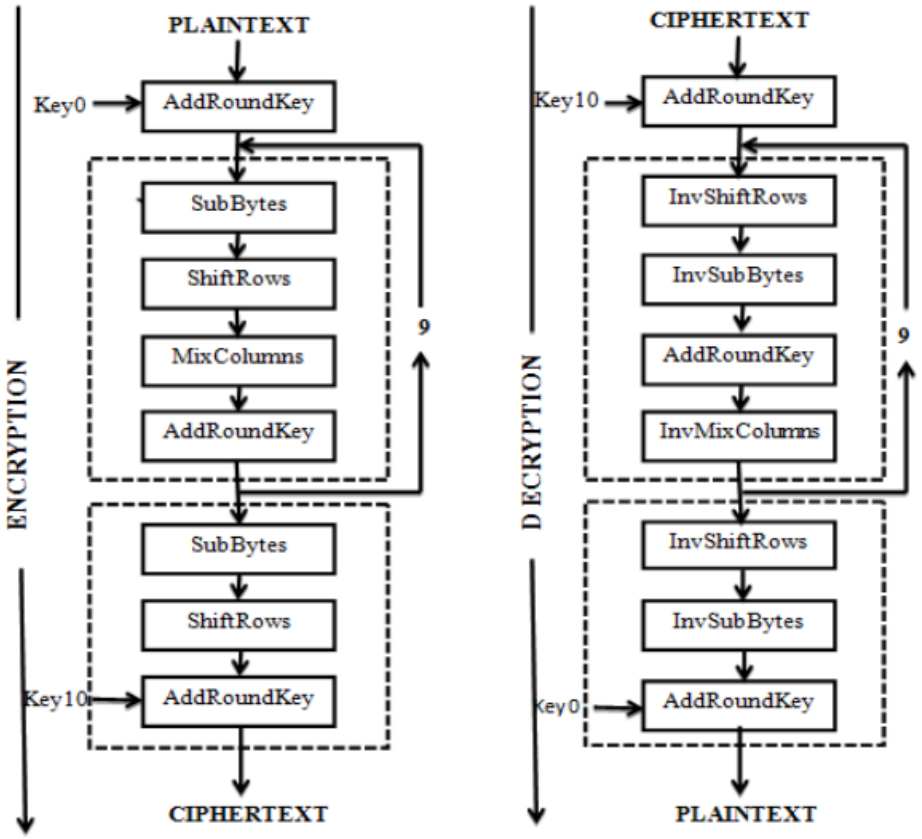
ii Shift Rows : In the encryption, the transformation is called Shift Rows.

iii Mix Columns : The Mix Columns transformation operates at the column level; it transforms each column of the state to a new column.

iv Add Round Key : Add Round Key proceeds one column at a time. Add Round Key adds a round key word with each state column matrix; the operation in Add Round Key is matrix addition. The last step consists of XO Ring the output of the previous three steps with four words from the key schedule. And the last round for encryption does not involve the “Mix columns” step. [8]

v. Decryption: Decryption involves reversing all the steps taken in encryption using inverse functions like a) Inverse shift rows, b) Inverse substitute bytes, c) Add round key, and d) Inverse mix columns. The third step consists of XO Ring the output of the previous two steps with four words from the key schedule. And the last round for decryption does not involve the “Inversemix columns” .

Figure 2 : AES Encryption and Decryption



	DES	AES
Developed	1977	2000
Key Length	56 bits	128, 192, or 256 bits
Cipher Type	Symmetric block cipher	Symmetric block cipher
Block Size	64 bits	128 bits
Security	Proven inadequate	Considered secure

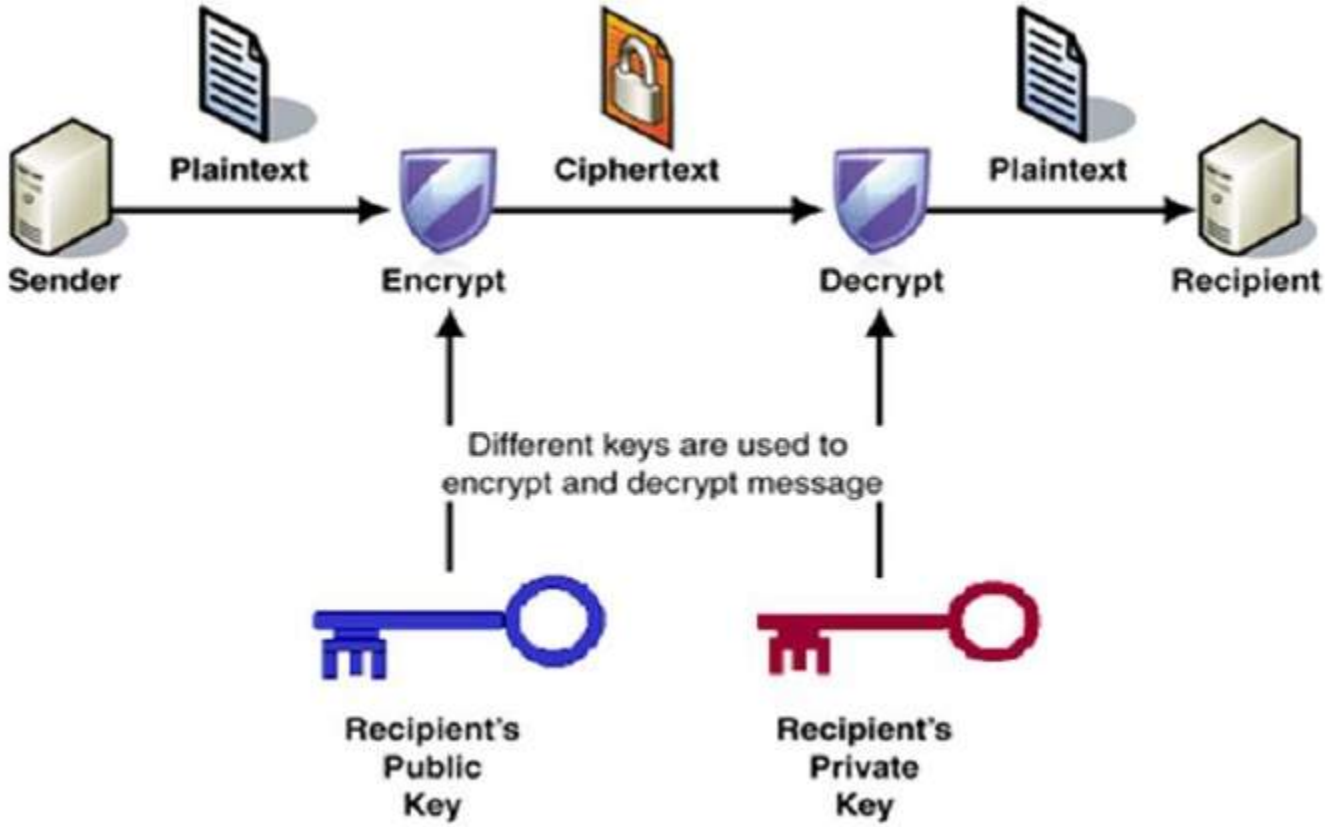
Public Key Cryptosystems

Public Key Cryptography

Unlike symmetric key cryptography, we do not find historical use of public-key cryptography. It is a relatively new concept.

Symmetric cryptography was well suited for organizations such as governments, military, and big financial corporations were involved in the classified communication.

With the spread of more unsecure computer networks in last few decades, a genuine need was felt to use cryptography at larger scale. The symmetric key was found to be non-practical due to challenges it faced for key management. This gave rise to the public key cryptosystems.
The process of encryption and decryption is depicted in the following illustration −



The most important properties of public key encryption scheme are −

- Different keys are used for encryption and decryption. This is a property which set this scheme different than symmetric encryption scheme.
- Each receiver possesses a unique decryption key, generally referred to as his private key.
- Receiver needs to publish an encryption key, referred to as his public key.
- Some assurance of the authenticity of a public key is needed in this scheme to avoid spoofing by adversary as the receiver. Generally, this type of cryptosystem involves trusted third party which certifies that a particular public key belongs to a specific person or entity only.
- Encryption algorithm is complex enough to prohibit attacker from deducing the plaintext from the ciphertext and the encryption (public) key.
- Though private and public keys are related mathematically, it is not be feasible to calculate the private key from the public key. In fact, intelligent part of any public-key cryptosystem is in designing a relationship between two keys.

There are three types of Public Key Encryption schemes. We discuss them in following sections –

RSA Cryptosystem

This cryptosystem is one the initial system. It remains most employed cryptosystem even today. The system was invented by three scholars Ron Rivest, Adi Shamir, and Len Adleman and hence, it is termed as RSA cryptosystem.

We will see two aspects of the RSA cryptosystem, firstly generation of key pair and secondly encryption-decryption algorithms.

Generation of RSA Key Pair

Each person or a party who desires to participate in communication using encryption needs to generate a pair of keys, namely public key and private key. The process followed in the generation of keys is described below –

- Generate the RSA modulus (n)
  - Select two large primes, p and q.
  - Calculate n=p*q. For strong unbreakable encryption, let n be a large number, typically a minimum of 512 bits.
- Find Derived Number (e)
  - Number e must be greater than 1 and less than $(p - 1)(q - 1)$.
  - There must be no common factor for e and $(p - 1)(q - 1)$ except for 1. In other words two numbers e and $(p – 1)(q – 1)$ are coprime.
- Form the public key
  - The pair of numbers (n, e) form the RSA public key and is made public.
  - Interestingly, though n is part of the public key, difficulty in factorizing a large prime number ensures that attacker cannot find in finite time the two primes (p & q) used to obtain n. This is strength of RSA.
- Generate the private key
  - Private Key d is calculated from p, q, and e. For given n and e, there is unique number d.
  - Number d is the inverse of e modulo $(p - 1)(q – 1)$. This means that d is the number less than $(p - 1)(q - 1)$ such that when multiplied by e, it is equal to 1 modulo $(p - 1)(q - 1)$.
  - This relationship is written mathematically as follows −

$$ed = 1 \bmod (p - 1)(q - 1)$$

The Extended Euclidean Algorithm takes p, q, and e as input and gives d as output.

Example

An example of generating RSA Key pair is given below. (For ease of understanding, the primes p & q taken here are small values. Practically, these values are very high).

- Let two primes be p = 7 and q = 13. Thus, modulus $n = pq = 7 \times 13 = 91$.
- Select e = 5, which is a valid choice since there is no number that is common factor of 5 and $(p - 1)(q - 1) = 6 \times 12 = 72$, except for 1.
- The pair of numbers (n, e) = (91, 5) forms the public key and can be made available to anyone whom we wish to be able to send us encrypted messages.
- Input p = 7, q = 13, and e = 5 to the Extended Euclidean Algorithm. The output will be d = 29.
- Check that the d calculated is correct by computing −

$$de = 29 \times 5 = 145 = 1 \bmod 72$$

- Hence, public key is (91, 5) and private keys is (91, 29).

Encryption and Decryption

Once the key pair has been generated, the process of encryption and decryption are relatively straightforward and computationally easy.

Interestingly, RSA does not directly operate on strings of bits as in case of symmetric key encryption. It operates on numbers modulo n. Hence, it is necessary to represent the plaintext as a series of numbers less than n.

RSA Encryption

- Suppose the sender wish to send some text message to someone whose public key is (n, e).
- The sender then represents the plaintext as a series of numbers less than n.
- To encrypt the first plaintext P, which is a number modulo n. The encryption process is simple mathematical step as −

$$C = P^e \bmod n$$

- In other words, the ciphertext C is equal to the plaintext P multiplied by itself e times and then reduced modulo n. This means that C is also a number less than n.
- Returning to our Key Generation example with plaintext P = 10, we get ciphertext C −

$$C = 10^5 \bmod 91$$

RSA Decryption

- The decryption process for RSA is also very straightforward. Suppose that the receiver of public-key pair (n, e) has received a ciphertext C.
- Receiver raises C to the power of his private key d. The result modulo n will be the plaintext P.

$$\text{Plaintext} = C^d \bmod n$$

- Returning again to our numerical example, the ciphertext C = 82 would get decrypted to number 10 using private key 29 −

$$\text{Plaintext} = 82^{29} \bmod 91 = 10$$

RSA Analysis

The security of RSA depends on the strengths of two separate functions. The RSA cryptosystem is most popular public-key cryptosystem strength of which is based on the practical difficulty of factoring the very large numbers.

- Encryption Function − It is considered as a one-way function of converting plaintext into ciphertext and it can be reversed only with the knowledge of private key d.
- Key Generation − The difficulty of determining a private key from an RSA public key is equivalent to factoring the modulus n. An attacker thus cannot use knowledge of an RSA public key to determine an RSA private key unless he can factor n. It is also a one way function, going from p & q values to modulus n is easy but reverse is not possible.

If either of these two functions are proved non one-way, then RSA will be broken. In fact, if a technique for factoring efficiently is developed then RSA will no longer be safe.

The strength of RSA encryption drastically goes down against attacks if the number p and q are not large primes and/ or chosen public key e is a small number.

Traffic Confidentiality

Traffic Confidentiality:

Users are concerned about security from traffic analysis. Knowledge about the number and length of messages between nodes may enable an opponent to determine who is talking to whom.

This can have obvious implications in a military conflict. Even in commercial applications, traffic analysis may yield information that the traffic generators would like to conceal. [MUFT89] lists the following types of information that can be derived from a traffic analysis attack:

Identities of partners

How frequently the partners are communicating

Message pattern, message length, or quantity of messages that suggest important information is being exchanged

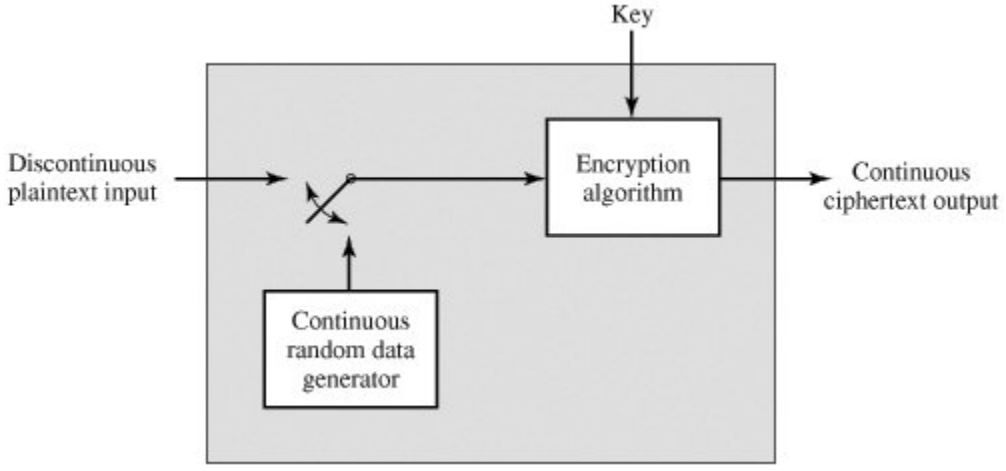
The events that correlate with special conversations between particular partners

Another concern related to traffic is the use of traffic patterns to create a covert channel. A covert channel is a means of communication in a fashion unintended by the designers of the communications facility. Typically, the channel is used to transfer information in a way that violates a security policy. For example, an employee may wish to communicate information to an outsider in a way that is not detected by management and that requires simple eavesdropping on the part of the outsider. The two participants could set up a code in which an apparently legitimate message of a less than a certain length represents binary zero, whereas a longer message represents a binary one. Other such schemes are possible.

Link Encryption Approach

With the use of link encryption, network-layer headers (e.g., frame or cell header) are encrypted, reducing the opportunity for traffic analysis. However, it is still possible in those circumstances for an attacker to assess the amount of traffic on a network and to observe the amount of traffic entering and leaving each end system. An effective countermeasure to this attack is traffic padding, illustrated in Figure 7.6.

Figure 7.6. Traffic-Padding Encryption Device



Traffic padding produces ciphertext output continuously, even in the absence of plaintext. A continuous random data stream is generated. When plaintext is available, it is encrypted and transmitted. When input plaintext is not present, random data are encrypted and transmitted. This makes it impossible for an attacker to distinguish between true data flow and padding and therefore impossible to deduce the amount of traffic.

End-to-End Encryption Approach

Traffic padding is essentially a link encryption function. If only end-to-end encryption is

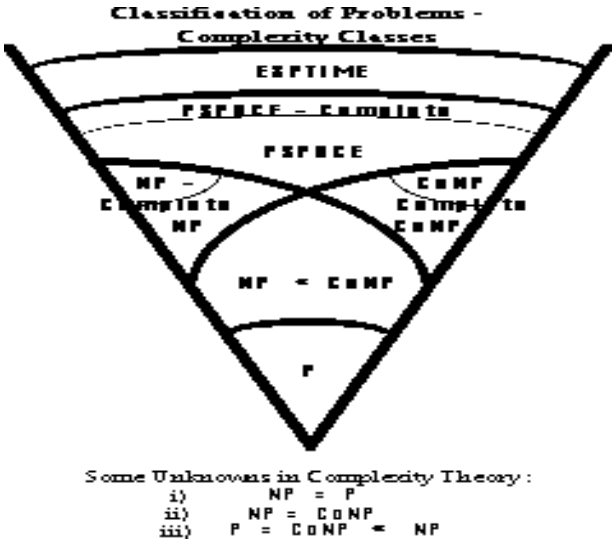
employed, then the measures available to the defender are more limited. For example, if encryption is implemented at the application layer, then an opponent can determine which transport entities are engaged in dialogue. If encryption techniques are housed at the transport layer, then network-layer addresses and traffic patterns remain accessible.
One technique that might prove useful is to pad out data units to a uniform length at either the transport or application level. In addition, null messages can be inserted randomly into the stream. These tactics deny an opponent knowledge about the amount of data exchanged between end users and obscure the underlying traffic pattern.

Cryptanalysis
Cryptanalysis is the decryption and analysis of codes, ciphers or encrypted text. Cryptanalysis uses mathematical formulas to search for algorithm vulnerabilities and break into cryptography or information security systems.
Cryptanalysis attack types include:

- Known-Plaintext Analysis (KPA): Attacker decrypt ciphertexts with known partial plaintext.
- Chosen-Plaintext Analysis (CPA): Attacker uses ciphertext that matches arbitrarily selected plaintext via the same algorithm technique.
- Ciphertext-Only Analysis (COA): Attacker uses known ciphertext collections.
- Man-in-the-Middle (MITM) Attack: Attack occurs when two parties use message or key sharing for communication via a channel that appears secure but is actually compromised. Attacker employs this attack for the interception of messages that pass through the communications channel. Hash functions prevent MITM attacks.
- Adaptive Chosen-Plaintext Attack (ACPA): Similar to a CPA, this attack uses chosen plaintext and ciphertext based on data learned from past encryptions.

Complexity Theory - A Potted Intro

- Complexity theory - study of how hard a problem is to solve in general
- allows classification of types of problems
- some problems intrinsically harder than others, eg
    - multiplying numbers O(n^(2))
    - multiplying matrices O(n^(2)(2n-1))
    - solving crossword O(26^(n))
    - recognizing primes O(n^(log log n))
- deal with worst case complexity
    - may on average be easier



Complexity Theory - Some Terminology

- an instance of a problem is a particular case of a general problem
- the input length of a problem is the number n of symbols used to characterize a particular instance of it
- the order of a function f(n) - is some O(g(n)) of some function g(n) s.t.
  - f(n)<=c.|g(n)|, for all n>=0, for some c
- a polynomial time algorithm (P) is one which solves any instance of a particular problem in a length of time O(p(n)), where p is some polynomial on input length
- an exponential time algorithm (E) - is one whose solution time is not so bounded
- a non-deterministic polynomial time algorithm (NP) - is one for which any guess at the solution of an instance of the problem may be checked for validity in polynomial time
- NP-complete problems - are a subclass of NP problems for which it is known that if any such problem has a polynomial time solution, then all NPproblems have polynomial solutions. These are thus the hardest NP problems
- Co-NP problems - are the complements of NP problems, to prove a guess at a solution of a Co-NP problem may well require an exhaustive search of the solution space.

HASH FUNCTIONS:

A cryptographic hash function is a mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size (a hash function) which is designed to also be one-way function, that is, a function which is infeasible to invert. The only way to recreate the input data from an ideal cryptographic hash function's output is to try a large number of possible inputs to see if they produce a match.Bruce Schneier has called one-way hash functions "the workhorses of Modern cryptography".[1] The input data is often called the *message*, and the output (the *hash value*or *hash*) is often called the *message digest* or simply the*digest*.

The ideal cryptographic hash function has four main properties:

- it is quick to compute the hash value for any given message
- it is infeasible to generate a message from its hash value except by trying all possible messages
- a small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value
- it is infeasible to find two different messages with the same hash value

Cryptographic hash functions have many information-security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. They can also be used as ordinary hash functions, to index data in hash tables, for fingerprinting, to detect duplicate data or uniquely identify files, and as checksums to detect accidental data corruption. Indeed, in information-security contexts, cryptographic hash values are sometimes called (*digital*) *fingerprints*, *checksums*, or just *hash values*, even though all these terms stand for more general functions with rather different properties and purposes.

Properties

Most cryptographic hash functions are designed to take a string of any length as input and produce a fixed-length hash value.

A cryptographic hash function must be able to withstand all known types of cryptanalytic attack. At a minimum, it must have the following properties:

- *Pre-image resistance*

  Given a hash value h it should be difficult to find any message m such that $h = hash(m)$. This concept is related to that of one-way function. Functions that lack this property are vulnerable to preimage attacks.

- *Second pre-image resistance*

  Given an input m_1 it should be difficult to find different input m_2 such that $hash(m_1) = hash(m_2)$. Functions that lack this property are vulnerable to second-preimage attacks.

- *Collision resistance*

  It should be difficult to find two different messages m_1 and m_2 such that $hash(m_1) = hash(m_2)$. Such a pair is called a cryptographic hash collision. This property is sometimes referred to as *strong collision resistance.* It requires a hash value at least twice as long as that required for preimage-resistance; otherwise collisions may be found by a birthday attack.

These properties imply that a malicious adversary cannot replace or modify the input data without changing its digest. Thus, if two strings have the same digest, one can be very confident that they are identical.

A function meeting these criteria may still have undesirable properties. Currently popular cryptographic hash functions are vulnerable to *length-extension* attacks: given *hash*(*m*) and *len*(*m*) but not *m*, by choosing a suitable *m*' an attacker can calculate *hash*(*m* || *m*') where || denotes concatenation.[2] This property can be used to break naive authentication schemes based on hash functions. The HMAC construction works around these problems.

Ideally, one may wish for even stronger conditions. It should be impossible for an adversary to find two messages with substantially similar digests; or to infer any useful information about the data, given only its digest. Therefore, a cryptographic hash function should behave as much as possible like a random function while still being deterministic and efficiently computable.

Checksum algorithms, such as CRC32 and other cyclic redundancy checks, are designed to meet much weaker requirements, and are generally unsuitable as cryptographic hash functions. For example, a CRC was used for message integrity in the WEP encryption standard, but an attack was readily discovered which exploited the linearity of the checksum.

To get a sense of how system security is established about, we must know the generally accepted architecture of cyber security setups. The Open System Interconnect(OSI) security architecture was designated by the the ITU-T (International Telecommunication Union - Telecommunication). The ITU-T decided that their standard "X.800" would be the ISO security architecture.

This standardized architecture defines security requirements and specifies means by which these requirements might be satisfied.

OSI Security Architecture Privacy of Data.

The OSI architecture focuses on

i. Security attacks
ii. Security services
iii. Security mechanisms

• Security attack: Any actions that compromises the security of information owned by an organization (or a person)

It includes:

1) Active attack

2) Passive attack

• Security mechanism: a mechanism that is designed to detect, prevent, or recover from a security attack

• Security service: a service that enhances the security of the data processing systems and the information transfers of an organization. The services make use of one or more security mechanisms to provide the service

UNIT-2

Linear Cryptanalysis, Differential Cryptanalysis, DES, Triple DES, Message Authentication and Digital Signatures, Attacks on Protocols, Elliptic Curve Architecture and Cryptography, Public Key Cryptography and RSA, , Evaluation criteria for AES, Key Management, Authentication requirements Digital forensics including digital evidence handling: Media forensics, Cyber forensics, Software forensics, Mobile forensics.

Linear Cryptanalysis:

linear cryptanalysis is a general form of cryptanalysis based on finding affine approximations to the action of a cipher. Attacks have been developed for block ciphers and stream ciphers. Linear cryptanalysis is one of the two most widely used attacks on block ciphers; the other being differential cryptanalysis.

The attack on DES is not generally practical, requiring 2^{47} known plaintexts.A variety of refinements to the attack have been suggested, including using multiple linear approximations or incorporating non-linear expressions, leading to a generalized partitioning cryptanalysis. Evidence of security against linear cryptanalysis is usually expected of new cipher designs

Overview

There are two parts to linear cryptanalysis. The first is to construct linear equations relating plaintext, ciphertext and key bits that have a high bias; that is, whose probabilities of holding (over the space of all possible values of their variables) are as close as possible to 0 or 1. The second is to use these linear equations in conjunction with known plaintext-ciphertext pairs to derive key bits.

Constructing linear equations

For the purposes of linear cryptanalysis, a linear equation expresses the equality of two expressions which consist of binary variables combined with the exclusive-or (XOR) operation. For example, the following equation, from a hypothetical cipher, states the XOR sum of the first and third plaintext bits (as in a block cipher's block) and the first ciphertext bit is equal to the second bit of the key:

In an ideal cipher, any linear equation relating plaintext, ciphertext and key bits would hold with probability 1/2. Since the equations dealt with in linear cryptanalysis will vary in probability, they are more accurately referred to as linear*approximations*.

The procedure for constructing approximations is different for each cipher. In the most basic type of block cipher, asubstitution-permutation network, analysis is concentrated primarily on the S-boxes, the only nonlinear part of the cipher (i.e. the operation of an S-box cannot be encoded in a linear equation). For small enough S-boxes, it is possible to enumerate every possible linear equation relating the S-box's input and output bits, calculate their biases and choose the best ones. Linear approximations for S-boxes then must be combined with the cipher's other actions, such as permutation and key mixing, to arrive at linear approximations for the entire cipher. The piling-up lemma is a useful tool for this combination step. There are also techniques for iteratively improving linear approximations (Matsui 1994).

Deriving key bits

Having obtained a linear approximation of the form:

we can then apply a straightforward algorithm (Matsui's Algorithm 2), using known plaintext-ciphertext pairs, to guess at the values of the key bits involved in the approximation.

For each set of values of the key bits on the right-hand side (referred to as a *partial key*), count how many times the approximation holds true over all the known plaintext-ciphertext pairs; call this count *T*. The partial key whose *T* has the greatest absolute difference from half the number of plaintext-ciphertext pairs is designated as the most likely set of values for those key bits. This is because it is assumed that the correct partial key will cause the approximation to hold with a high bias. The magnitude of the bias is significant here, as opposed to the magnitude of the probability itself.

This procedure can be repeated with other linear approximations, obtaining guesses at values of key bits, until the number of unknown key bits is low enough that they can be attacked with brute force.