# Chapter 1: Classical Ciphers

## Caesar Cipher

Subjected to brute-force attack

## Monoalphabetic Ciphers

Cipher is produce by using any permutation of the 26 alphabetic characters. Subjected to frequency analysis attack.

## Playfair Cipher

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C | R | Y | P | T |
| O | A | B | D | E |
| F | G | H | I/J | K |
| K | N | B | Q | S |
| Y | V | W | X | Z |

Plaintext is encrypted two letters at a time, rules:

* Repeating letters pair are separated with a filler letter, such as y, so that “balloon” would be treated as “ba ly lo on”.
* If same row, replace with letter on the right
* If same column, replace with letter beneath
* Else, replace with letter in same row, but column of the paired letter

## Hill Cipher

Substitutes *m* successive plaintext to *m* ciphertext

## Polyalphabetic Ciphers (PolyC)

A set of related monoalphabetic substitution rules is used

A key determines which rule is chosen for a transformation

## PolyC: Vigenere Cipher

Let say m number of key:

## PolyC: Vernam Cipher

Keywords length equal to plaintext length. Works on binary data. Hard to generate key.

## One-Time Pad

Random key length equal to plaintext length. Key can only be used once. Unbreakable.

## Transposition Technique

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | 3 | 1 | 2 | 5 | 6 | 7 |
| A | T | T | A | C | K | P |
| O | S | T | P | O | N | E |

Ciphertext: TTAPTSAOCOKNPE

## Rotor Machines

Use polyalphabetic substitution with period of 26. Adding multiple cylinder equals more substitution.

# Chapter 2.1: Number Theory

## Divisibility

*n|m* -> m is divisible by n

## Greatest Common Divisor

If *c|a* and *c|b* and any divisor of a and b is a divisor of c

## Euclidean Algorithm

## Extended Euclidean Algorithm

Bezout’s Identity:

Example:

As a result:

## Modular Arithmetic

## Modular Arithmetic in Zn

for k

for k & x

### Commutative Laws:

### Associative Laws:

### Distributive Law:

### Identities:

### Additive Inverse (-w):

z, where

### Multiplicative inverse:

If and,

If a is relatively prime to n

## Groups, Rings, and Fields

A list of properties:

* (A1) Closure:
* (A2) Associative:
* (A3) Identity:
* (A4) Inverse:
* (A5) Commutative:
* (M1)
* (M2)
* (M3) and
* (M4)
* (M5) and
* (M6) If , then or
* (M7) and

Groups satisfies A1-A4

Abelian group satisfies A1-A5

Cyclic group, if every element of G is a power

Ring satisfies A1-M3

Commutative Ring satisfies A1-M4

Integral domain satisfies A1-M6

Fields satisfies A1-M7

# Chapter 2.2: Galois Field, GF(p)

is a commutative ring and is a finite field because M4-M6

### To find multiplicative inverse:

## Modular Polynomial Arithmetic

Consider polynomials where:

Such polynomial is a

irreducible polynomials:

# Chapter 2.3: Prime Number, Fermat’s Theorem, Euler’s Theorem

## Prime Number

Integer is prime number i.f.f. its only divisors are and

### Relatively Prime

are relatively prime if

### Prime Factorization

Any integer

Use it to compute gcd:

## Fermat’s Theorem

Fermat’s Theorem 1: If *p* is a prime number and *a* is a positive integer not divisible by *p*, then:

Fermat’s Theorem 2: If p is a prime number and a is a positive integer, then:

## Euler’s Theorem

Euler’s Phi function : number of positive integer less than *n* and relatively prime to *n*. If p is a prime number:

Let where *p* and *q* are distinct prime number, then:

Euler’s Product Formula:

Euler’s Theorem 1:

Euler’s Theorem 2:

# Chapter 2.4 Primality Test, Chinese Remainder Theorem, Discrete Logarithm

## Miller-Rabin Test

Property 1:

If :

i.f.f

Property 2: if and :

* Get *k* and *q* from
* OR

For property 2:

* If not property 2 hold, n is not prime
* If property 2, it is inconclusive, where

## Chinese Remainder Theorem

CRT Assertion 1:

What is *A*?

CRT Assertion 2:

## Primitive Root

The smallest value of *m* is called the order of *a (mod n)*. If , then *x* is a primitive root of *n*.

## Discrete Logarithm

Primitive root for some prime *p*:

Properties:

# Chapter 3.1: Secret-key Cryptography

## Block Ciphers

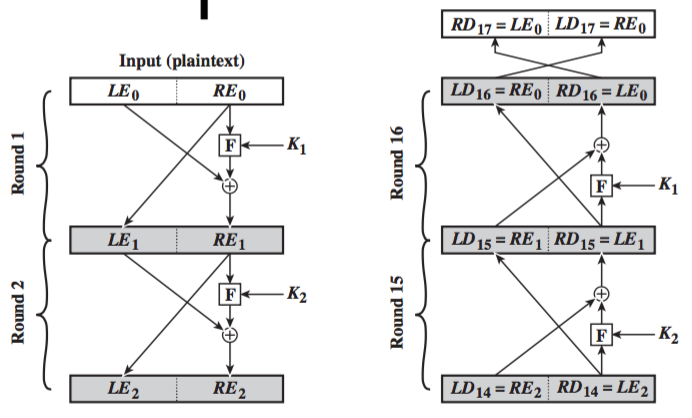
Block of plaintext is processed together, to create ciphertext of same size

Input space: , Output space:

All possible mappings:

All possible reversible mappings:

## Feistel Ciphers



Substitution: round function *F* & XOR

Permutation: swap two halves

Considerations:

* Large block size, better diffusion, slower computations
* Large key size, better confusion, resistance against brute-force attacks
* More rounds, greater confusion and diffusion, resistance to cryptanalysis

## Data Encryption Standard

It is based on Feistal Structure.

Avalanche effect: small changes in plaintext of key should product significant changes in the ciphertext.

### DES Steps

Splitting: The 64 bits input is split into two blocks of 32 bits.

Initial Permutation (IP): there is a permutation table that will guide they way we permute 32 bits input.

Expansion/permutation Table: 32 bits input is expanded into 48 bits by using an expansion table. The 48 bits input the XOR with 48 bits key.

Substitution / choice: The 48 bits input is split into 8 blocks, with 6 bits in each block. The first bit and last bit decide the row of S-box, the remaining four bits determine the column.

### DES Key generation

The 64 bits key is reduced to 56 bits by throwing away the last bits of 8 bits blocks.

The 56 bits key then go through Permuted Choice One and becomes 56 bits permutated key.

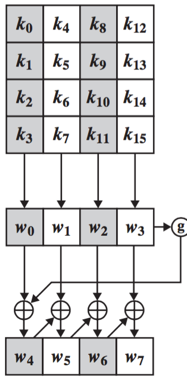
The 56 bits then go through a left circular shift. The number of left shifts for each round are different.

The 56 bits key then go through Permuted Choice Two. The permutation table has 8 positions that are not in the table. The result of this step is 48 bits key.

# Chapter 3.2: AES

Operations on 8-bit byte strings, in , with . Inputs: 4 x 4 column-major matrix of bytes. AES is not based on Feistel structure, has very strong avalanche effect than DES.

## Round Keys



Function g involves byte rotation, substitution, and XOR with round constant

## Substitute Bytes Transform

Left 4-bits of input for row, right 4-bits for column. Construction of s-box:

* Get multiplicative inverse bits of input
* Calculate the transformation value

Inverse s-box works similarly:

* Get multiplicative inverse bits of input
* Calculate the transformation value

## Shift Rows Transform

row get bytes left circular shift, so that every column has byte from every column of input. For inverse, just do similar right circular shift.

## Mix Columns Transform

In , multiplication by 02 can be done by a-bit left shift followed by [if leftmost bit of original value is 1, XOR with 0001 1011]. For inverse:

## Add Round Key Transform

The 128 bits of state are bitwise XOR with the 128 bits of round key

# Chapter 4: Public Key Cryptography

It solves two problems with symmetric key cryptography: key distribution and digital signatures.

## The RSA Algorithm

Let , where *p* and *q* are prime, , e and d are relatively prime with , then:

Public key,

Private key,

Encryption:

Decryption:

## RSA Computation Optimizations

Choices of because these numbers only have two 1 bits in binary, so exponent can be calculated as:

However, small number of is vulnerable, and can be solve using CRT:

Small number of *d* is vulnerable to brute-force attack and other cryptanalysis, but we can speed up computation using CRT:

The exponent computation can be speed up using Fermat’s Theorem:

## Elliptic Curve over

We use simplified equation, :

To form abelian group over addition:

If , , , , and , then:

Multiplication in is defined as repeated addition

## Elliptic Curve Cryptography

Based on discrete logarithm: given , it is hard to find *k* given *Q* and *P*

Pick a base point *G*. The smallest positive integer *n* that satisfied is called the order of *G*. The order must be large.

Bob private key,

Bob public key,

Alice generate a random *k*, then send:

Bob decrypts the message using :

Attacker need to solve discrete logarithm. He knows *kG* and *G*, but not *k*:

## Elgamal Cryptographic System

Based on discrete logarithm. Choose a prime number *p*, primitive root .

Alice private key,

Alice public key,

Bob want to sent plaintext . Select random integer .

Alice decrypts the message using :

Attacker need to solve discrete logarithm:

Alternatively:

# Chapter 5: Message Authentication Codes (MAC)

MAC provided authentication, message integrity, and confidentiality. MAC function receives input message *M* and secret key *K*.

Message authentication:

Two authentication and confidentiality:

## Naïve HMAC

Merkle-Damgard hash function construct:

Attack on secret prefix MAC:

where and . Bob accepts the message.

## Cipher Block Chaining MAC (CMAC)

DAA Algorithm, we take as MAC:

Attack on DAA:

Since , Bob accepts.

To fix DAA algorithm:

To generate

# Chapter 6: Digital Signature

MAC protects two parties who exchange message from any third party, but not against each other. Difference between MAC and digital signature is non-repudiation, the sender cannot deny the creation of the message. Properties of digital signatures:

* Verify the author and date and time of signature
* Authenticate the contents
* Verifiable by third parties to resolve disputes

Basic digital signature protocol:

## RSA Signature Algorithm

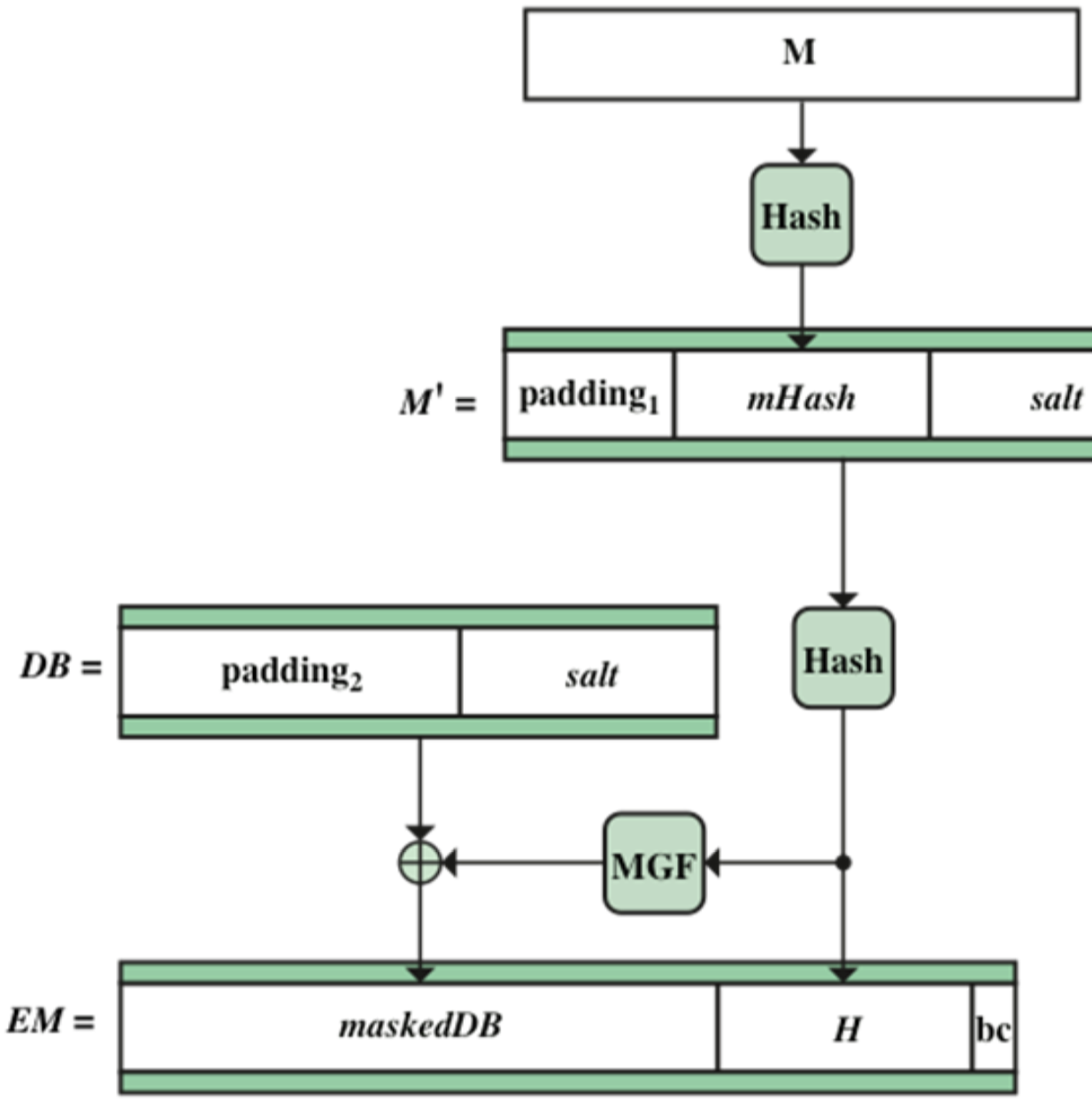
Textbook RSA Digital Signature:

Existential forgery attack: *A* will accept

## RSA-PSS Algorithm

To reduce effect of existential forgery, messages need to have certain formats.

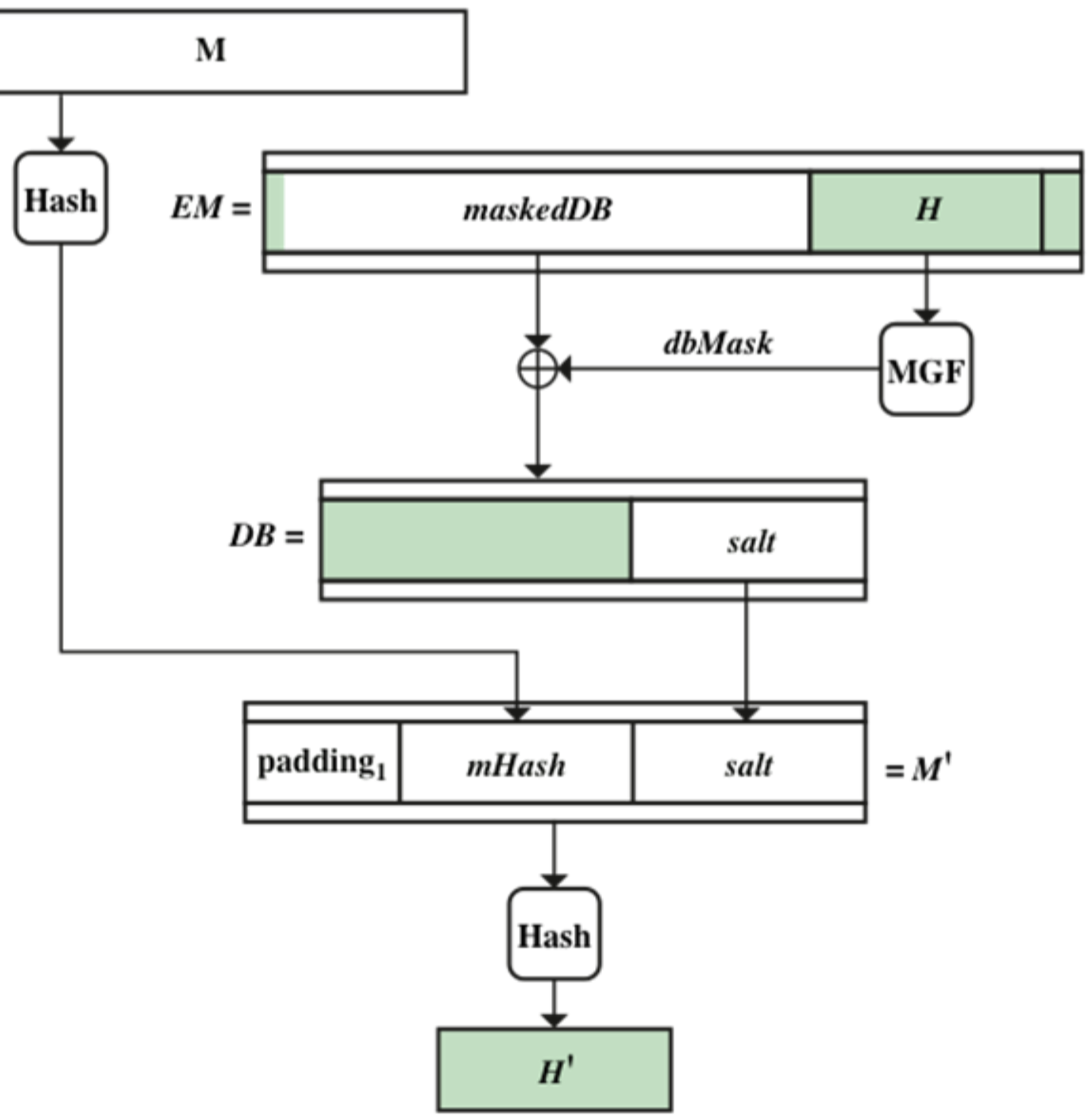
### RSA-PSS Encoding:



* EM: encoded message
* padding1 and padding2 are strings of zeros
* MGF: mask generation function
* bc: a fixed hexadecimal value

The final signature of M is:

### RSA-PSS EM Verification:



## ElGamal Digital Signature

Key generation:

* Choose large prime *p*
* Choose primitive root
* Choose random *d*,
* Compute
* Public key:
* Private key:

Signature generation:

* Signature for *x* with
* Choose random *k*, ,
* Compute
* Compute
* The signature is

Signature verification:

* Compute
* If then is valid for x

Existential forgery, the attacker intercepts the message and modify it to:

To prevent existential forgery, we hash the message *x*:

And signature verification step:

## Digital Signature Algorithm (DSA)

Yield shorter signatures than ElGamal

Key generation:

* Choose
* Choose
* Choose
* Choose random
* Compute
* Public key:
* Private key:

Signature Generation:

* Choose *k*,
* Compute
* Compute
* The signature is

Signature verification:

* Compute
* Compute
* Compute
* Compute
* If then valid for *x*

# Chapter 7: Key Establishment

Key establishment deals with establishing a shared secret between two or more parties. It can be classified into:

* Key transport protocol: a technique where one party securely transfers a secret value to others.
* Key agreement protocol: two or more parties derive the shared secret where all parties contribute to the secret.

We use shared secret to derive session key. Session key can limit the available cipher texts for cryptanalysis attack, avoid long term storage of large number of session keys, create independence across communication session.

Perfect Forward Secrecy: a protocol satisfies this property if compromise of long-term keys does not compromise past session keys.

## Key Distribution

In P2P network with *n* nodes, to set up a secure channel between every pair of users, we need symmetric keys. By using Key Distribution Center, we only need to store *n* keys for *n* users.

## Authenticated Key Exchange Protocol 2 (AKEP2), without KDC

Objective: A and B derive a session key W. They authenticate to each other at the end of protocol session.

## Needham-Schroeder Shared Key Protocol (NSSK), with KDC

## Otway-Rees Protocol, with KDC

## One-pass Protocol, without signature

Suitable for one-way communication

## Needham Schroeder Public-key Protocol, without signature

## One-pass Protocol, with signature

Example 1:

Example 2:

Example 3:

## Key Transport using Public-Key Authority

Single point of failure: At each run of the protocol, participant needs to obtain the public key from the public key authority.

## Key Transport using Certificate Authority

Obtaining certificate in secure channel:

Exchanging certificates:

# Chapter 8: Strong Authentication

In strong authentication, both parties take both the roles of claimant and verifier.

Basic idea of challenge-response authentication: claimant proves its identity to verifier by demonstrating knowledge of secrets, without revealing the secrets to the verifier.

Time-variant challenges are typically numbers that are unique across instances of protocols. It can be random numbers, sequence numbers, timestamps.

## Kerberos

### A simple flawed authentication protocol

Replay attack at step 3:

### Ticket-granting ticket

Once per logon session:

Once per type of service:

Once per service session:

Where:

Timestamp and lifetime is used to check the expiry of a ticket to prevent indefinite replay attack.

Replay attack 1:

Replay attack 2:

### Authenticator and Kerberos V4

Obtaining ticket-granting ticket:

Use TGT to obtain service granting ticket:

Client / server authentication exchange:

## X.509 Authentication Protocol

: fresh session keys

The use of certificates ensures the authenticity of the public keys of A and B

## Secure Socket Layer (SSL)

It is a security service that runs on top of TCP and used in HTTPS and SSH. The SSL Record Protocol provides two services for SSL connections:

* Confidentiality: encryption of SSL payloads, using a shared secret key defined by the Handshake Protocol.
* Message integrity: message authentication, using a shared MAC key, also defined by Handshake Protocol.

Transport Layer Security (TLS): It is a standard version of SSL

HTTPS refers to the combination of HTTP and SSL to implement secure communication between Web browser and Web server.

# Missing List

Chapter 1: Type of attack based on information known to attackers

Chapter 2.2: Polynomial Modular Arithmetic Generator

Chapter 2.4: Proof for Discrete Logarithm

Chapter 3.1: DES Example

Chapter 3.1: Cryptanalysis of DES

Chapter 3.1: Design criteria of DES components

Chapter 3.2: Overall AES Structure

Chapter 3.2: Rationale for each steps

Chapter 3.2: AES Example

Chapter 3.2: Simplified AES

Chapter 4: Elgamal Cryptographic System

Chapter 5.2: Naïve hash, secret suffix

Chapter 5.2: Hash-based MAC function (HMAC)

Chapter 6: ElGamal, reuse of key, attack

Chapter 6: DSA Computational Aspects

Chapter 7: Motivation of session key, motivation examples, assumptions on attacker, possible strategies of attacks, key compromise, known key attack, public key infrastructure and certificate authority, X.509 certificate, CA chaining, certification revocation

Chapter 8: Details of SSL, TLS and HTTPS, attack on SSL