SOS
Cryptography

Nikhil Injarapu

July 21, 2023

Mentor : Gowtam S

Tentative Deadlines

- Week 1:Introduction, perfect secrecy and stream ciphers
- Week 2 : Block ciphers and using Block ciphers
- Week 3 :Message Integrity and hash functions
- **Mid term report**
- Week 4:Message Authentication Codes and hash functions
- Week 5 & 6:Public key cryptography
- Week 7:Digital Signatures
- Week 8:Optional and end term report/video

Resources

- The Joy of Cryptography, Mike Rosulek
- Introduction to Modern Cryptograhy, Katz & Lindell.
- Coursera:Cryptography-1(Stanford University)

1 Introduction

1.1 Encryption-Scheme

1. An encryption scheme is defined by three algorithms Gen, Enc, and Dec, as well as a specification of a message space M with $|M| > 1$

2. The key-generation algorithm Gen is a probabilistic algorithm that outputs a key k chosen according to some distribution We denote by K the key space

3. The encryption algorithm Enc takes as input a key $k \in K$ and a message $m \in M$, and outputs a ciphertext c; we denote this by $E(k, m)$.

4. The decryption algorithm Dec takes as input a key $k \in K$ k and a ciphertext $c \in C$ and outputs a message $m \in M$

1.2 Correctness of Encryption

$$Dec(k, Enc(k, m)) = m$$

1.3 Adversarial indistinguishability

This definition is based on an experiment involving an adversary A and its inability to distinguish the encryption of one plaintext from the encryption of another, and we thus call it adversarial indistinguishability.

The experiment is as follows:

1. The adversary A outputs a pair of messages m0 , m1 ∈ M.

2. A random key k is generated by running Gen, and a random bit b ← 0, 1 is chosen. (These are chosen by some imaginary entity that is running the experiment with A.) Then, the ciphertext c ← E (m_b) is computed and given to A.

3. A outputs a bit b' .

4. The output of the experiment is defined to be 1 if b' = b, and 0 otherwise.If the output is 1 and in this case we say that A succeeded.

1.4 Perfect Secrecy

The following definitions are all equivalent

1. An encryption scheme (Gen, Enc, Dec) over a message space M is perfectly secret if for every probability distribution over M, every message m ∈ M, and every ciphertext c ∈ C for which $Pr[C = c] > 0$:

$$Pr[M = m|C = c] = Pr[M = m]$$

2. An encryption scheme (Gen, Enc, Dec) over a message space M is perfectly secret if and only if for every probability distribution over M, every message m ∈ M, and every ciphertext c ∈ C:

$$Pr[C = c|M = m] = Pr[C = c]$$

3. An encryption scheme (Gen, Enc, Dec) over a message space M is perfectly secret if and only if for every probability distribution over M, every m0 , m1 ∈ M, and every c ∈ C:

$$Pr[C = c|M = m0] = Pr[C = c|M = m1]$$

4. An encryption scheme (Gen, Enc, Dec) over a message space M is perfectly secret if for every adversary A it holds that:

$$Pr[b' == b] = \frac{1}{2}$$

Note: There is no limitation whatsoever on the computational power of A.

2 Stream ciphers

2.1 OTP:One-Time-Pad

Let $a \oplus b$ denote the bitwise exclusive-or (XOR) of two binary strings a and b ($i.e., if a = a1, ..., a_l and b = b1, ..., b_l, then a \oplus b = a1 \oplus b1, ..., a_l \oplus b_l$). The one-time pad encryption scheme is defined as follows:

1. We consider K,M,C (i.e key space,msg space,cipher text space) all are of length n∈N
2. Key generation algorithm chooses a key k∈ $\{0-1\}^n$ from a uniform distribution
3. Enc(k,m)=$k \oplus m$
4. Dec(k,c)=$k \oplus c$

Correctness:

$$\begin{aligned} Dec(k,c) &= k \oplus c \\ &= k \oplus (k \oplus m) \\ &= m \end{aligned}$$

Claim: The one-time pad is a perfectly-secret encryption scheme.
Proof:

$$\begin{aligned} Pr[C = c|M = m] &= Pr[M \oplus K = c|M = m] \\ &= Pr[m \oplus K = c] \\ &= Pr[K = m \oplus c] \\ &= 2^{-n} \end{aligned}$$

Limitations of OTP:

1. Length of key is large (i.e equal to $|M|$)
2. Key cannot be used twice, as $c1 \oplus c2 = (m1 \oplus k) \oplus (m2 \oplus k) = m1 \oplus m2$ can reveal greater information about the messages by frequency analysis.

Remark: Let (Gen, Enc, Dec) be a perfectly-secret encryption scheme over a message space M, and let K be the key space as determined by Gen. Then $|K| \geq |M|$.

Proof:
Lets assume the contradiction that our $|K| < |M|$
We check the following definition for checking perfect secrecy
Let m ∈ M and k_1 ∈ K such that E(k_1,m)=c_1 and D(k_1,c_1)=m

$$Pr[M = m|C = c] = Pr[M = m]$$

$Pr[M = m] = \frac{1}{|M|}$
Let $M_c = \{m|D(k, c_1) = m, k \in K\}$
$|M_c| \leq |K| < |M|$
Clearly m ∈ M_c, so $Pr[M = m|C = c] = \frac{1}{|M_c|}$
Hence $Pr[M = m|C = c] \neq Pr[M = m]$

2.2 Shannons theorm:

Let (Gen, Enc, Dec) be an en- cryption scheme over a message space M for which $|M| = |K| = |C|$. This scheme is perfectly secret if and only if:

1. Every key k ∈ K is chosen with equal probability $1/|K|$ by algorithm Gen.

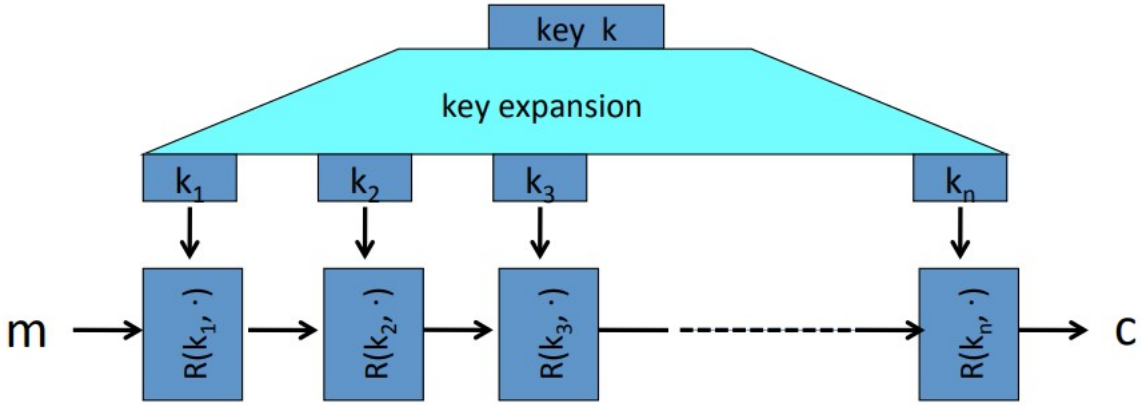
2. For every m ∈ M and every c ∈ C, there exists a single key k ∈ K such that E(m) outputs c.

Proof can be derived in a similar manner as the above remark.

3 Block ciphers

3.1 General construction of a block cipher

The general construction of a block cipher is done using the below mechanism



Here R(k,.) is called as Round function which generally uses a function or permutation called pseudo random function(PRF) or pseudo random permutation(PRP).

3.2 Pseudorandom function

Let

$$F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$$

be and efficient , length preserving, keyed function. We say F is a pseudorandom function if for all probabilistic polynomial-time distinguishers D, there exists a negligible function negl such that:

$$|Pr[D^{F_k(.)}(1^n) = 1] - Pr[D^{f_n(.)}(1^n) = 1]| \leq negl(n),$$

where k $\leftarrow 0, 1$ is choosen uniformly at random and f_n chosen uniformly at random from the set of functions mapping n-bit strings to n-bit strings
Notice that D interacts freely with its oracle. Thus it can ask queries adaptively, choosing the next input based on the previous outputs recieved. However, since D runs in polynomial time, it can only ask a polynomial number of queries.

3.3 Pseurandom permutation

Let

$$F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$$

be and efficient , length preserving, keyed function. We say F is a pseudo-random permutation if for all probabilistic polynomial-time distinguishers D, there exists a negligible function negl such that:

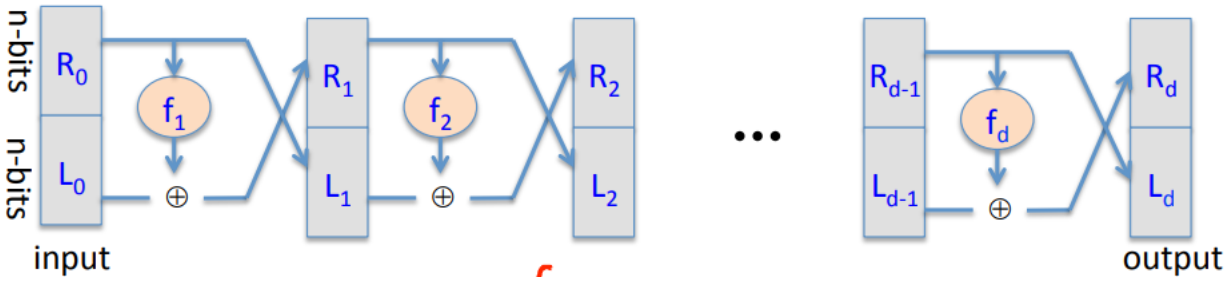
$$|Pr[D^{F_k(.),F_k^{-1}(.)}(1^n) = 1] - Pr[D^{f_n(.),f_n^{-1}(.)}(1^n) = 1]| \leq negl(n),$$

where k $\leftarrow 0,1$ is chosen uniformly at random and f_n chosen uniformly at random from the set of functions mapping n-bit strings to n-bit strings

A pseudorandom permutation can be used in place of a pseudorandom function in any cryptographic construction. This is due to fact that to any polynomial time observer , a pseudorandom permuation cannot be distinguised from a pseudorandom function.

3.4 Feistel Network

Given functions $f_1, f_2, ..., f_d : \{0,1\}^n \to \{0,1\}^n$
Goal : build invertible function $F : \{0,1\}^{2n} \to \{0,1\}^{2n}$



$$R_i = f_i(R_{i-1}) \oplus L_{i-1}$$

$$L_i = R_{i-1}$$

Claim: for all $f_1, ..., f_d : \{0,1\}^n \to \{0,1\}^n$ Feistel network $F : \{0,1\}^{2n} \to \{0,1\}^{2n}$ is invertible.

Proof: In the function that is formed,

$$R_{i-1} = L_i$$

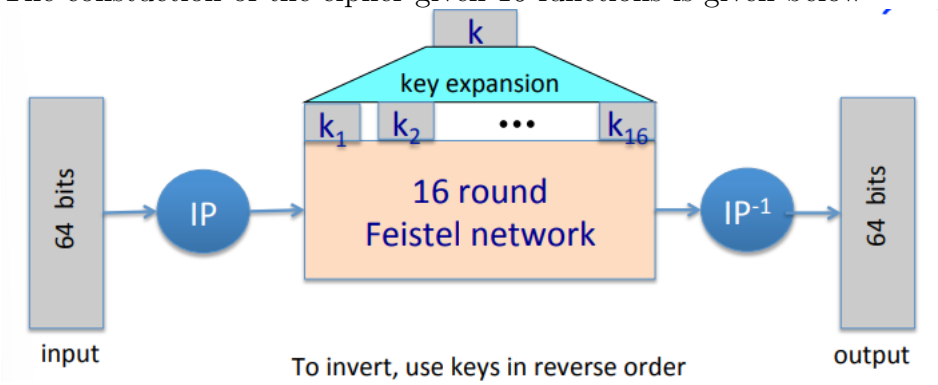
$$L_{i-1} = R_i \oplus f_i(L_i)$$

- since from the above two equations we can see that given R_i and L_i we can constuct R_{i-1} and L_{i-1} for all i
- There fore the above consturction is invertible
- Inversion is basically the same cirtuit with $f_1, ..., f_d$ applied in reverse order.
- This is the gerernal method for building invertible functions(block ciphers) from arbitary functions.
- Let f: $K \times \{0,1\}^n \to \{0,1\}^n$ is a secure PRF, then 3-round feistel F: $K^3 \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ is secure PRP.

3.5 Some examples of Block ciphers

3.5.1 DES

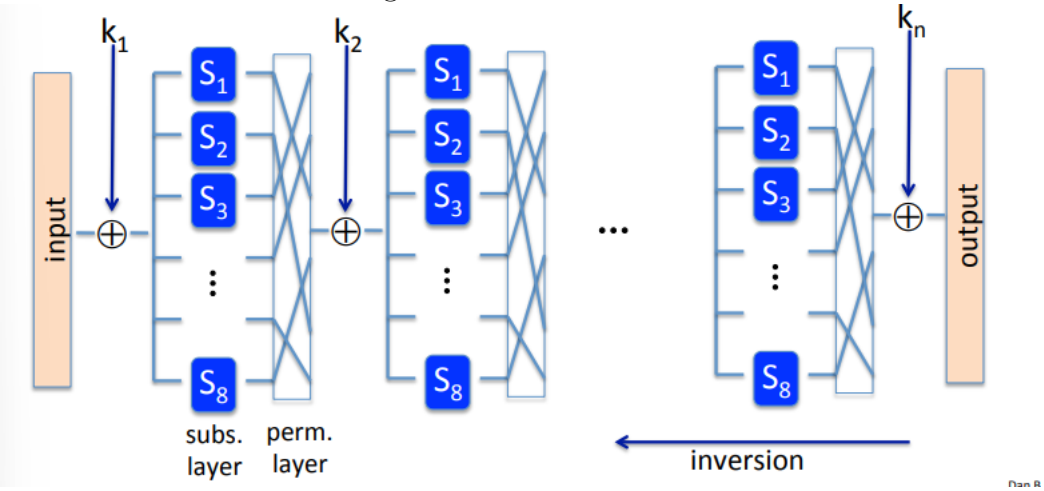
- DES is a 16 round Feistel network where it converts a 64 bit message into 64 bit cipher text.
- $f_1, ...f_{16} : \{0,1\}^{32} \to \{0,32\}^{32}, f_i(x) = F(k_i, x)$
- The constuction of the cipher given 16 functions is given below



- The function F(k,) takes different k extracted from the K given as the input and the 32 bit message.
- Inside this function there is a function called S-box is used which takes 6 bit input and outputs 4 bit input, implemented as a lookup table.

3.5.2 AES

- Unlike DES this does not use feistel network for encryption. Instead it uses subs-perm network
- The construction of AES is given below



- Each step in the network is functions which consists of three operations named Byte Sub, ShiftRow and MixColumn.
- The input for this function will be given as a two dimensional matrix of size 4 x 4 where each entry of the matrix is of size 8 bits i.e 1 byte.
- **Byte sub** is a process of substituting each byte in the matrix using a standard substitution table.
- **shitfRows** is shifting the entries of row by a fixed procedure.
- **Mixcolums** is mixing different colums using a fixed algorithm.

3.5.3 3DES

- Since inf DES the key size is only of 64 bits, as we will see after an attack called exhaustive attack is very much posssible and this encryption is badly broken.
- Due to this we no longer use DES but instead of DES a new encryption called 3DES is used
- Let E: K X M $\rightarrow$ M be a DES
- Define 3E: $K^3 \times M \rightarrow M$ as

$$3E((k_1, k_2, k_3), m) = E(k_1, D(k_2, E(k_3, m)))$$

- There fore for 3DES key size is 168 bits and is not possible for an exhaustive attack.
- But this method is 3 times slower than DES.

3.6 Attacks on Block ciphers

3.6.1 Exhaustive attack

- Evaluate the decryption value for every possible key that is allowed for that particular method.
- In case of DES the lenght of key is 64 bits. This means DES can be broken in time 2^64 which is achievable at present quite fast.
- But for AES key is of 128 bits which is quite large and cannot be broken using exhaustive search

3.6.2 Attacks on implementation

- Side channel attacks- measure time to do enc/dec, measure power for enc/dec.
- Fault attacks - coumputing errors in the last round expose the secret key k.

3.6.3 Linear attack

- Let c=DES(k,m) suppose for random k,m:

$$Pr[m[i_1] \oplus ... \oplus m[i_r] \bigoplus c[j_j] \oplus .. \oplus c[j_v] = k[l_1] \oplus ... \oplus k[l_u]] = \frac{1}{2} + \epsilon$$

  for some epsilon. For DES , this exists with $\epsilon = \frac{1}{21}$

- Given $\frac{1}{\epsilon^2}$ random (m,c=DES(k,m)) pairs then

$$k[l_1, ...l_u] = MAJ[m[i_1, ..., i_r] \bigoplus c[j_j, ..., j_v]]$$

  with prob $\geq 97.7\%$

- That is with $\frac{1}{\epsilon^2}$ inp/out pairs one can find $k[l_1, ..l_u]$ in time $\frac{1}{\epsilon^2}$

- For DES with 2^{42} inp/out pairs one can find roughly 14 bits in this way in 2^{42} time

- And using these known bits the remaining bits can be found in a total time of 2^{43}

3.7 Modes of Operation

3.7.1 Electronic Code Book(ECB)

- This is the most naive mode of operation possible.

- Given a plaintext message $m = m_1, m_2, ...m_l$ the cipher text is obtained by "encrypting" each block seperately.

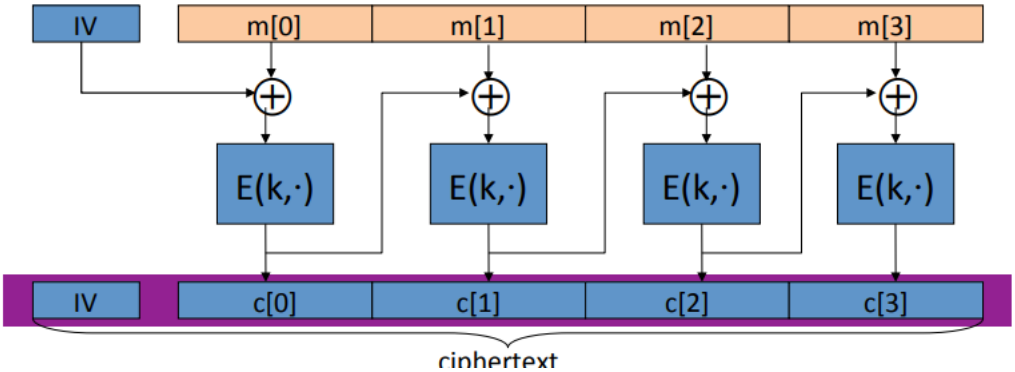
- That is, c=$\langle F_k(m_1), F_k(m_2), ...F_k(m_l)\rangle$. Decryption is done in the obvious way, using the fact that F_k^{-1} is efficiently computable.

- The encryption process here is deterministic and therefore this mode of operation cannot possibly be CPA-secure.

- Even worse, ECB encryption does not have indistinguishable encryptions in the presence of an eavesdropper, even if only used once.

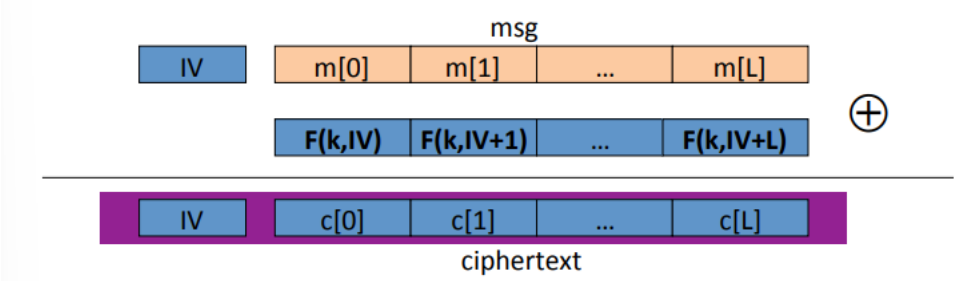
3.7.2 Cipher Block chaining(CBC) mode

- In this random initial(IV) of length n is first chosen.
- Then, the first ciphertext block is generated by applying the pseudo-random permutation to $IV \oplus m_1$
- The remainder of the ciphertext is obtained by XORing the ith cipher-text block with $i + 1^{th}$ plaintext.
- Here is the construction of CBC



3.7.3 Counter (CTR) mode

- Let $F : K \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a secure PRF.
- $E(k, m)$: choose a random IV $\in \{0,1\}^n$ and do:



- Both CTR and CBC are CPS-secure and CTR is parallelizable unlike CBC.
- If F is a pseudorandom function, then randomized counter mode has indistinguishable encryptions under a chosen-plain attack.

4 Message integrity

4.1 What is message integrity

- One of the most basic goals of cryptography is to enable parites to communicate over an open communication in a secure way.
- One immediate question that rises, however, is what do we mean by "secure communication"
- However not all security concerns are related to the ability of an adversary to learn something about messages being sent.
- An attacker can tamper a message without knowing what's about the message which is a major security concern.
- In this chapter we will show how to cryptographically prevent any tampering of messages that are sent over an open communication line

4.2 Message Authentication Codes

- The aim of message authentication code is to prevent an adversary from modifying a message sent by one party to another, without the parties detecting that a modification has been made.
- As in the case of encryption, such a task is only possible if the communicating parties have some secret that the adversary does not know (otherwise nothing can prevent an adversary from impersonating the party sending the message).
- The setting that we consider here therefore assumes that the parties share the same secret key.
- Loosely speaking, a MAC is an algorithm that is applied to a message. The output of the algorithm is a MAC tag that is sent along with the message.
- Security is formulated by requiring that no adversary can generate a valid MAC tag on any message that was not sent by the legitimate communicating parties.

4.3 Syntax of message authentication code

A message authentication code or MAC is a tuple of probabilistic polynomial-time algorithms(Gen,Mac,Vrfy) fulfilling the following

1. Upon the input 1^n the algorithm Gen outputs a uniformly distributed key k of length n, $k \leftarrow Gen(1^n)$.

2. The algorithm Mac receives for input some k $\in \{0,1\}^n$ and m $\in \{0,1\}^*$, and outputs some t$\in \{0,1\}^*$. The value of t is called the MAC tag.

3. The algorithm vrfy receives for input some k $\in \{0,1\}^n$, m $\in \{0,1\}^*$ and t $\in \{0,1\}^*$ and outputs a bit b $\in \{0,1\}$.

4. For every n, every k$\in \{0,1\}^n$ and every m $\in \{0,1\}^*$ it holds that Vrfy(m,Mac(m)) = 1

we remark that as for encryption, the second requirement can be relaxed so that

$$Pr[Vrfy_k(m, Mac_k(m)) = 1] > 1 - negl(n)$$

where negl is a neliglible function and the probabilities are taken over the choice of k and any internal coin tosses of Mac and vrfy

4.4 Secure MACs

4.4.1 The message authentication experiment Mac-forg$e_{A,II}(n)$

1. A random key k $\leftarrow \{0,1\}^n$ is chosen

2. The adversary A is given oracle access to Mac$_k(.)$ and outputs a pair (m,t). Formally ,(m,t) $\leftarrow A^{Mac_k(.)}(1^n)$. Let Q denote the queries asked by A during the execution.

3. The output of the experiment is defined to be 1 if and only if m $\notin$ Q and Vrfy_k(m,t) =1.

The definition states that no efficient adversary should succeed in the above experiment with non-negligible probability.

4.4.2 Definition of secure MAC

A message authentication code II = (Gen,Mac,Vrfy) is existentially unforgeble under an adaptive chosen-message attack, or just secure, if for all probabilistic polynomial-time adversaries A, there exits a negligible function negl such that:

$$Pr[Mac - forge_{A,II}(n) = 1 \leq negl(n)$$

We remark that a message authentication code can always be broken with negligible probability(that is , there is no hope of ensuring that an adversary's success in the experiment is zero).

4.5 Construction of Fixed-length MAC

Let F : $\{0,1\}^* \rightarrow \{0,1\}^*$ be a function such that for every k, $F_k(.)$ maps n-bit strings to n-bit strings. Define a fixed-length MAC as follows:

- Gen(1^n): upon input 1^n, choose k $\leftarrow \{0,1\}^n$
- Mac$_k(m)$: upon input key k $\in \{0,1\}^n$ and message m $\in \{0,1\}^n$, compute t= F_k(m).
- Vrfy(m,t): upon input key $\in \{0,1\}^n$, message m $\in \{0,1\}^n$ and tag t $\in \{0,1\}^n$, output 1 if and only if t=$F_k(m)$. (If the lenghts are incorrect , then output 0).

4.6 Construction of variable length MACs

Let F: $\{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ be a function such that for every k $\in \{0,1\}^n$, $F_k(.)$ maps n-bit strings to n-bit strings. Define a variable length MAC as follows:

- **Gen(1^n) :** upon input 1^n, choose k $\leftarrow \{0,1\}^n$
- **Mac**$_k(m)$: upon input a key k $\in \{0,1\}^n$ and a message m $\in \{0,1\}^*$ of length at most $2^{\frac{n}{4}-1}$, first parse m into d blocks $m_1, ..m_d$ each of length n/4. Next choose a randome indentifier r $\leftarrow \{0,1\}^{n/4}$. Then for i=1,...d, compute $t_i = F_k(r\|d\|i\|m_i)$, where i and d are uniquely encoded into strings of length n/4 and "$\|$" denotes concactenation.. Finally, output the tag t=$(r, t_1, ...t_d)$.

- **Vrf$_{y_k}$(m,t)**: Upon input key k, message m and tag t, run the MAC tage generation algorithm Mac except that instead of choosing random identifier, take the r taht appears in t. Output 1 if and if only if the tag tht is received by runnin Mac with this r is identical to t.

4.7 Some examples of MAC

4.7.1 CBC-MAC

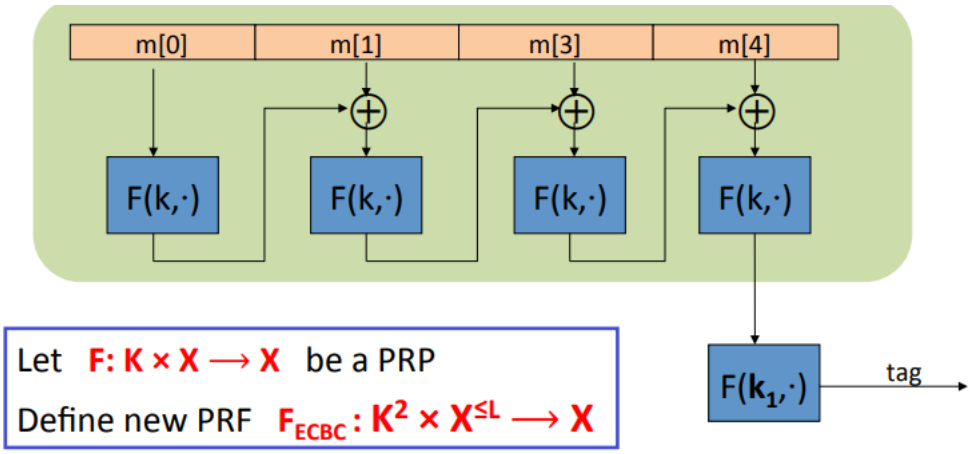
Let

$$F : K \times X \to X$$

be a PRP Define new PRF

$$F_{ECBC} : K^2 \times X^{\leq L} \to X$$

construction of the CBC-MAC function is as follows



4.7.2 NMAC

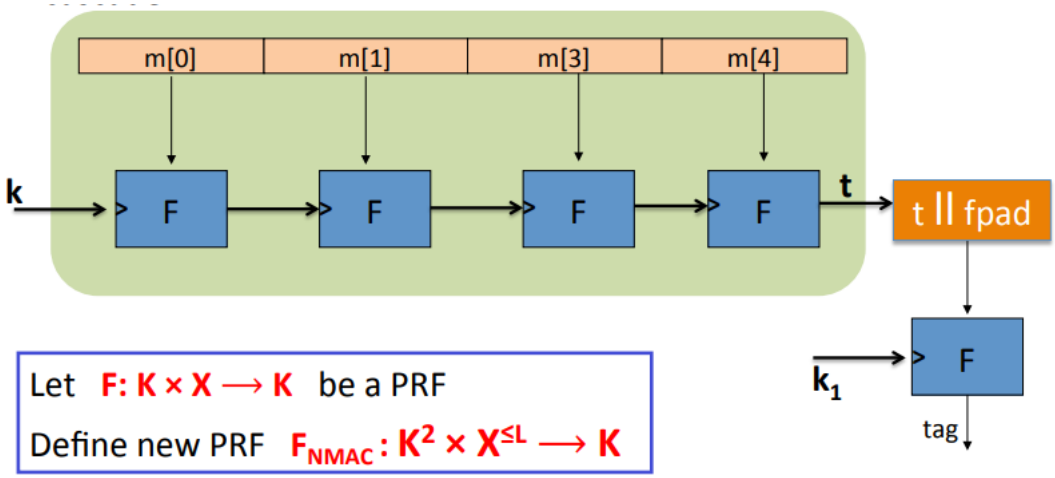
Let

$$F : K \times X \to X$$

be a PRF Define new PRF

$$F_{NMAC} : K^2 \times X^{\leq L} \to X$$

construction of the NMAC function is as follows



4.7.3 PMAC

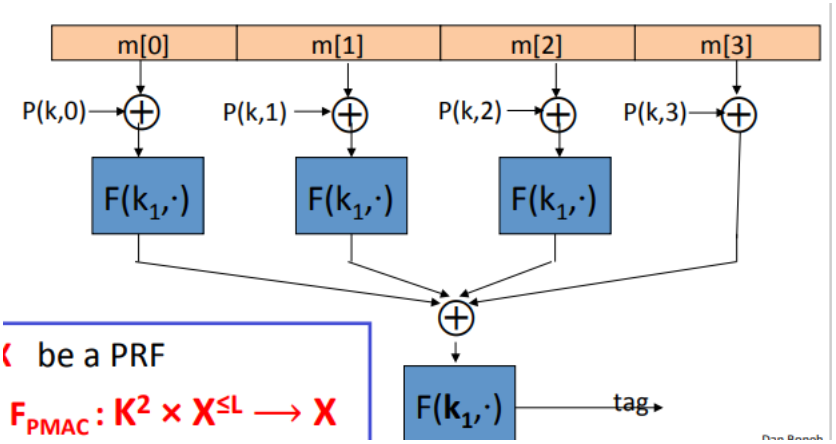
Let

$$F : K \times X \to X$$

be a PRP Define new PRF

$$F_{PMAC} : K^2 \times X^{\leq L} \to X$$

construction of the PMAC-MAC function is as follows



Unlike remaining this is a parallel MAC. Therefore can be calculated fast.

5 Collision resistant Hash functions

5.1 Definition of collision resistance

- A collision in a function H is a pair of distinct inputs x and x' such that H(x)=H(x'). In this case we also say that x and x' collide under H.
- As we have mentioned , a function H is collision-resistant if it is infeasible for any probabilistic polynomial time algorithm to find a collision in H.
- Typically we will be interested in functions H that have an infinite domain i.e they accept all strings of all input lengths and finite range

5.2 Syntax of hash function

A hash function is a pair of probabilistic polynomial time algorithms (Gen,H) fulfilling the following

- Gen is a probabilistic algorithm which takes as input a security parameter 1^n and outputs a key s.
- There exists a polynomial l such that H is polynomial-time algorithm that takes as input a key s and any string x $\in \{0,1\}^*$, and outputs a string $H^s(x) \in \{0,1\}^{l(n)}$

If for every n and s, H^s is defined only over inputs of length l'(n) and l'(n) ¿ l(n), then we say that (Gen,H) is a fixed length hash function with length parameter l'

5.3 Collision-finding experiment Hash-coll$_{A,II}(n)$:

- A key s is chosen: $S \leftarrow Gen(1^n)$
- The adversary A is given s and outputs a pair x and x'. Formally, (x,x') $\leftarrow A(s)$.
- The output of the experiment is 1 if and only if x $\neq$ x' and $H^s(x) = H^s(x')$. In such a case we say that A has found a collision.

The definition of collision resistance for hash functions states that no efficient adversary can find a collision except with negligible probability.

5.4 Collision resistant function

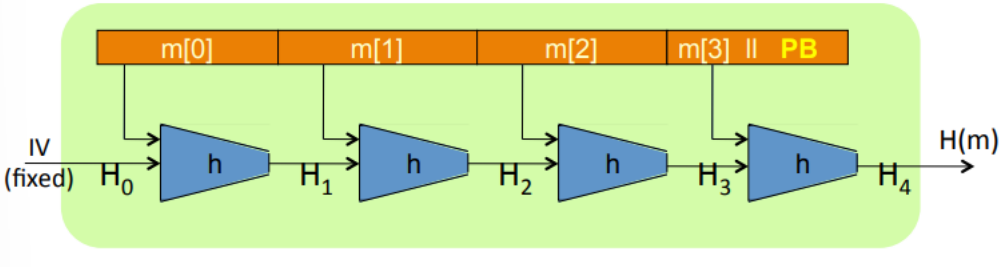
A hash function II=(Gen,H) is collision resistant if for all probabilistic polynomial-time adversaries A there exists a negligible function negl such that

$$Pr[Hash - coll_{A,II}(n) = 1] \leq negl(n)$$

For simplicity, we refer to H, H_s and (Gen,H) all using same term "collision-resistant hash function"

5.5 Merkle-Damgard iterated construction

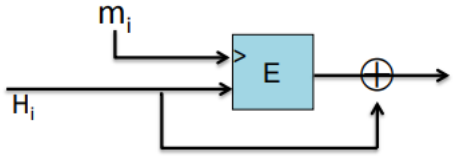
- Given $h : \times X \rightarrow T$(compression function)
- we obtain $H : X^{\leq L} \rightarrow T$ H_i - changing variables
- The construction of this method is as follows



- In this construction if h is collision resistant then so is H.
- Now our goal is to construct this h

5.6 Davies-Meyer compression function

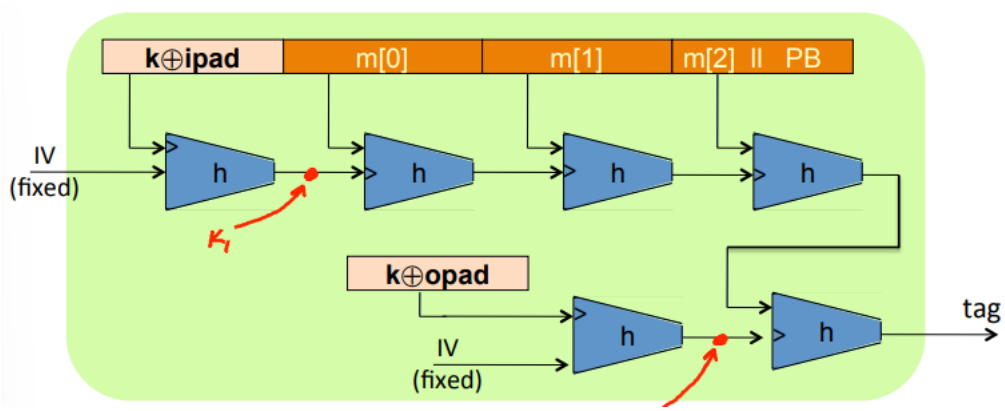
- Let E: $k \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a block cipher
- The Davies-Meyer compression function : h(H,m) = E(m,H $\oplus$ H
- This construction of Davies-Meyer is given below



- There are many other variants of this construction such as h(H,m)=E(m,H)$\oplus H \oplus m$ and h(H,m) =E $(H \oplus m, m) \oplus m$ ..etc

5.7 HMAC

- This is most widely used MAC on the internet
- H:hash function example SHA-256 which ouputs 256 bits.
- HMAC : S(k,m)=H$(k \oplus opad || H(k \oplus ipad || m))$
- Given below the construction diagram of HMAC



- HMAC is an industry standard and is highly efficient and easy to implement and is supported by proof of security.