Cryptographic Primitives
A brief introduction

Ragesh Jaiswal

CSE, IIT Delhi

Cryptography: Introduction

- Throughout most of history:
  - Cryptography = **art** of secret writing
  - Secure communication

M

$M = D_K(C)$

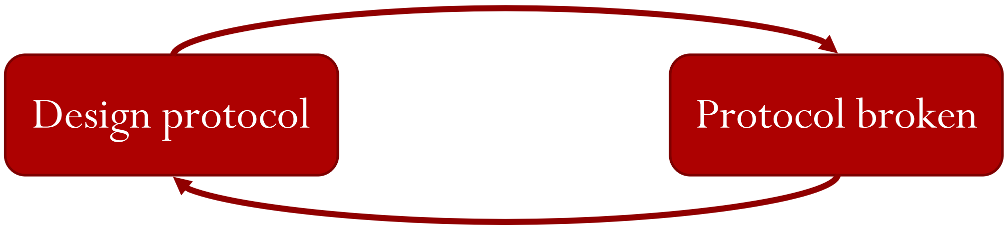
$C = E_K(M)$

$\boldsymbol{K}$ $\boldsymbol{K}$

Key exchange protocol

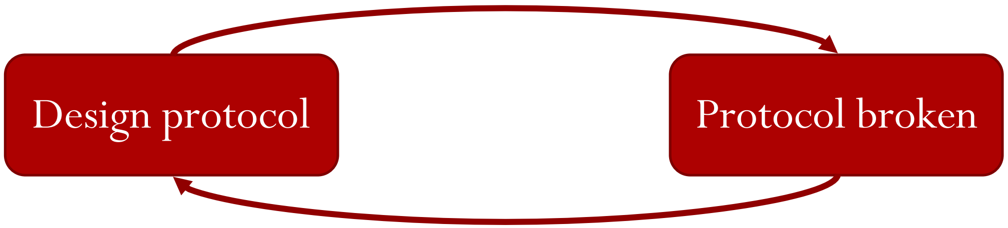
Cryptography: Introduction

- Early history (- early 70s):
  - Synonymous with secret communication.
  - Restricted to Military and Nobility.
  - More of *art* than rigorous science.



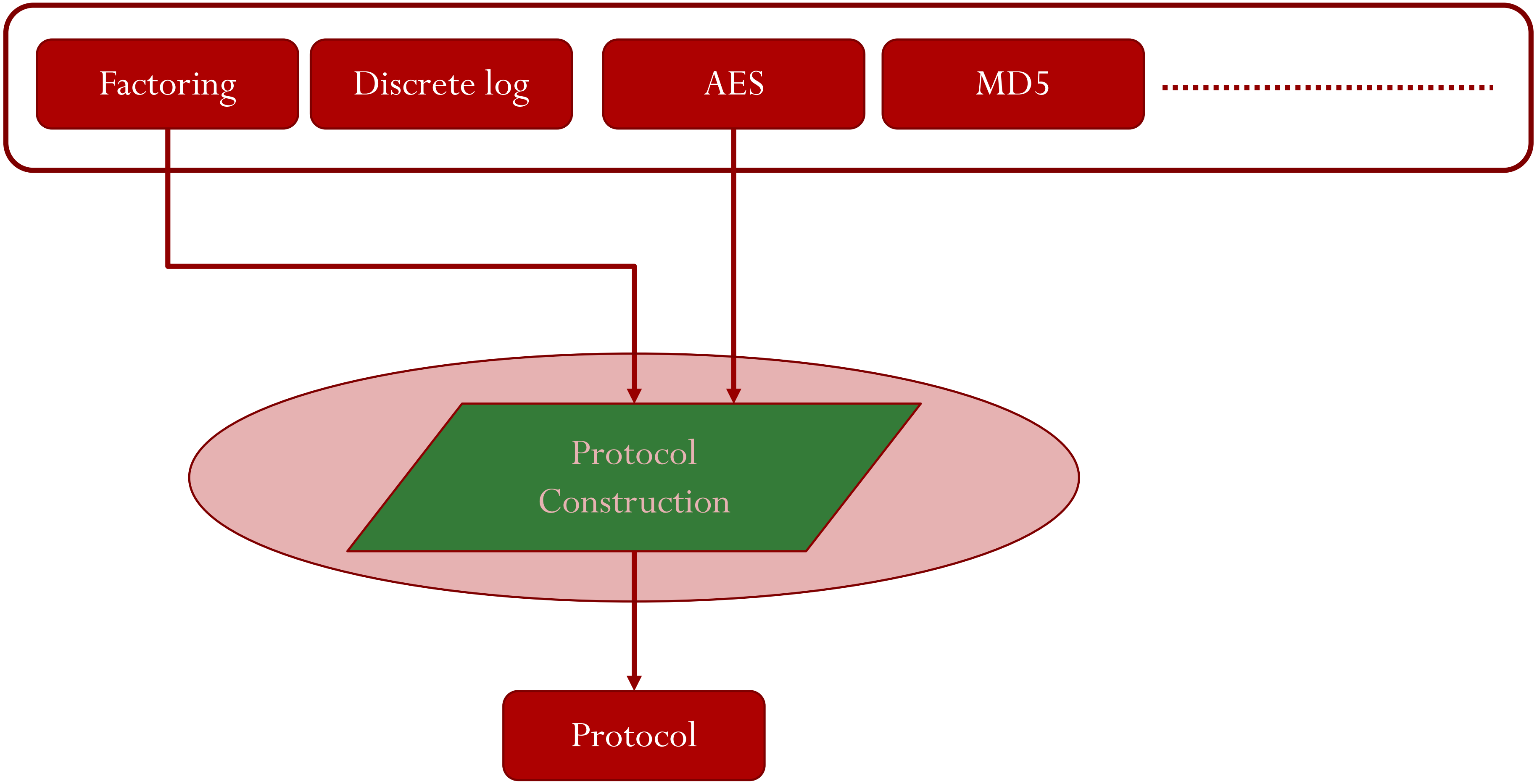
Cryptography: Introduction

- Early history (- early 70s):
  - Synonymous with secret communication.
  - Restricted to Military and Nobility.
  - More of *art* than rigorous science.

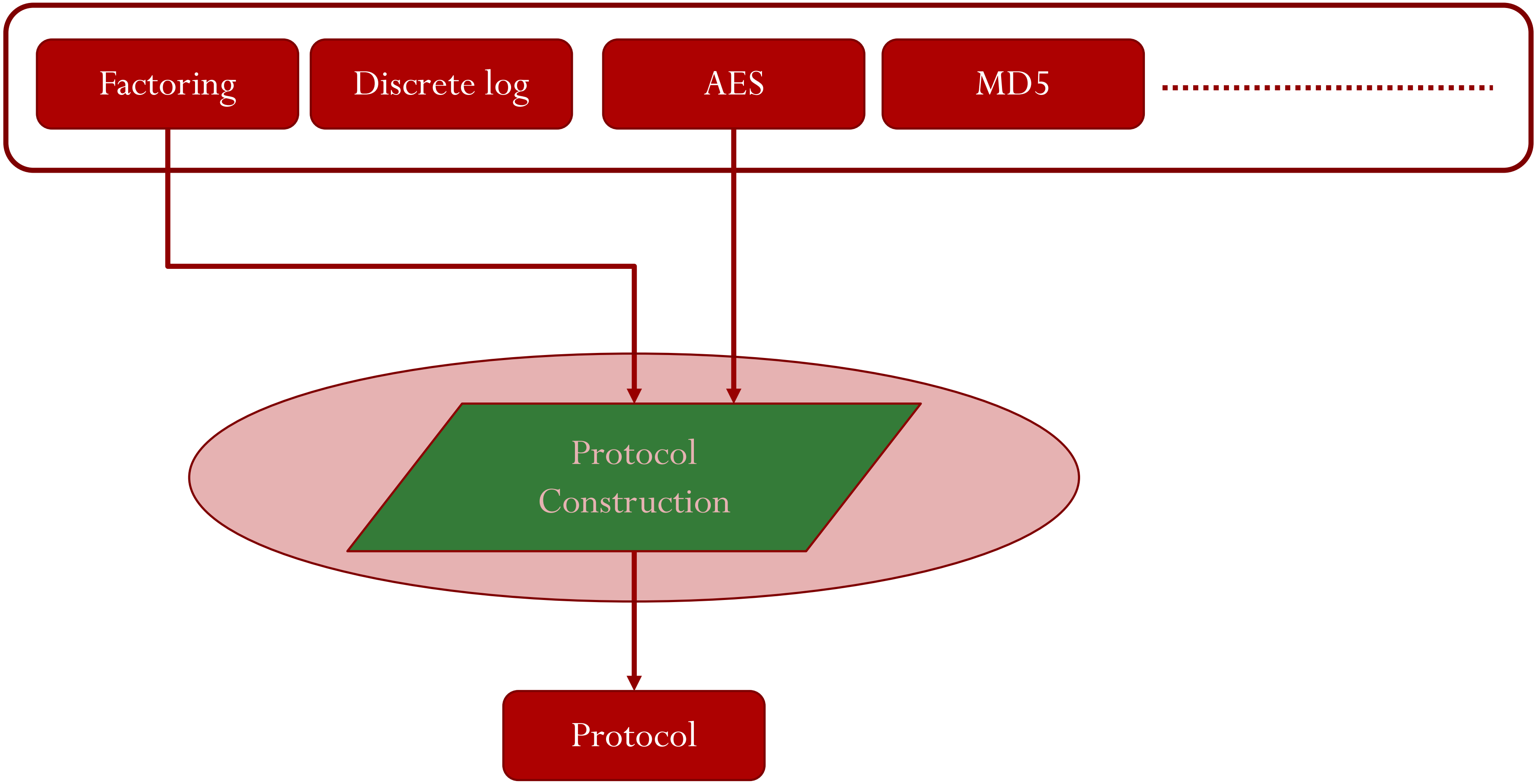


- Modern Cryptography:
  - Digital signatures, e-cash, secure computation, e-voting …
  - Touches most aspects of modern lifestyle.
  - Rigorous science:
    - *Reason about security of protocols.*

Cryptography: Provable security



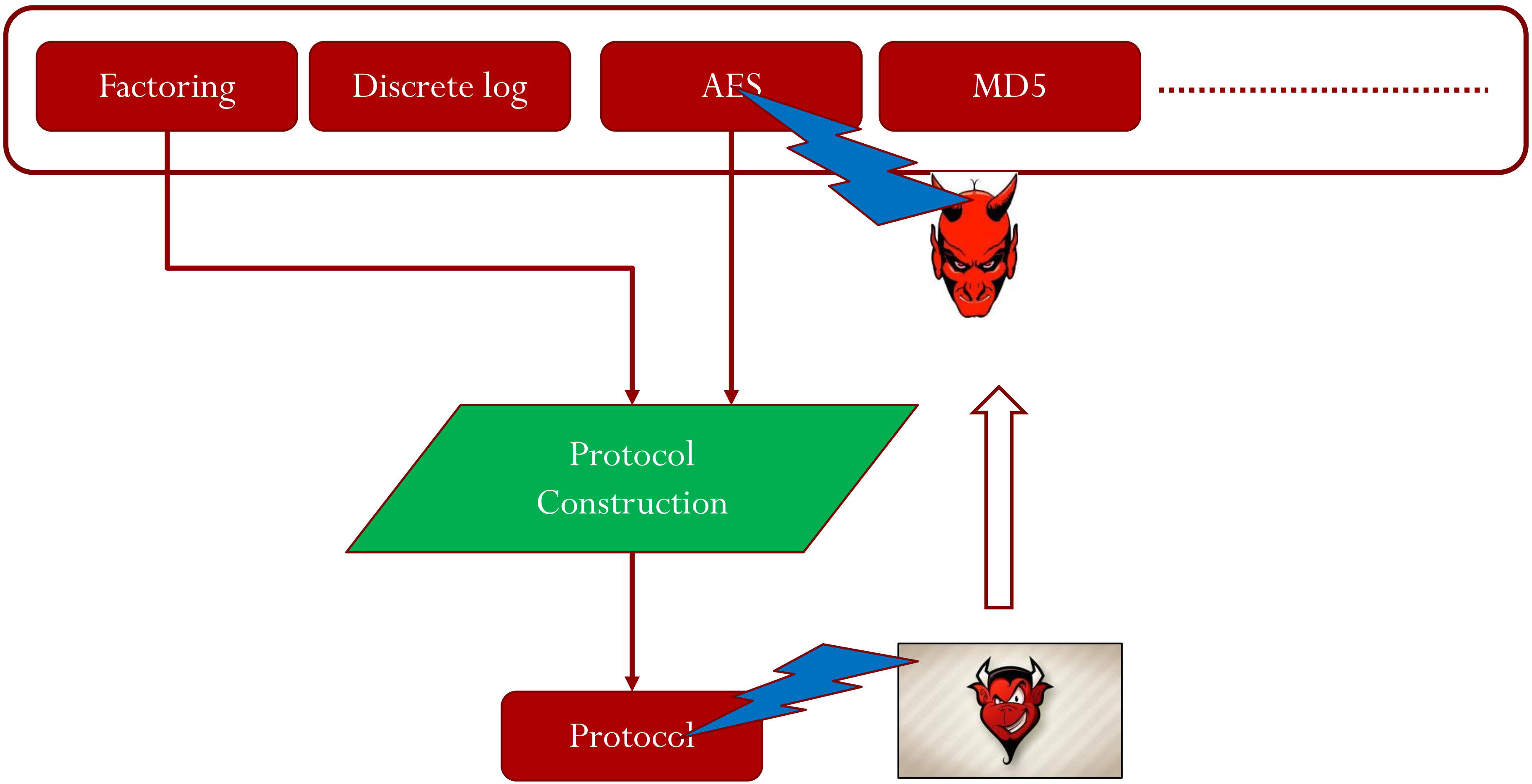
Cryptography: Provable security



We would like to argue:

- If the basic primitive/problem is secure/hard, then the constructed protocol is "secure"

Cryptography: Provable security



- :If there is an adversary that successfully attacks the protocol, then there is another adversary that successfully attacks/solves at least one of the basic primitives/problems.

Secure Communication

Secure communication

- Secure communication: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.

- Simple idea (Ceaser Cipher): Substitute each letter with the letter that is the αth letter after the letter in the sequence AB…Z
- Example ($\alpha = 2$): SEND TROOPS →

Secure communication

- <u>Secure communication</u>: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.

- <u>Simple idea (Ceaser Cipher)</u>: Substitute each letter with the letter that is the αth letter after the letter in the sequence AB…Z
- <u>Example ($\alpha = 2$)</u>: SEND TROOPS → UGPF VTQQRU

Secure communication

- Secure communication: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.

- Simple idea (Ceaser Cipher): Substitute each letter with the letter that is the αth letter after the letter in the sequence AB…Z
- Security was based on the fact that the encryption algorithm was a secret (Security through obscurity)

Secure communication

- <u>Secure communication</u>: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.

- <u>Simple idea (Ceaser Cipher)</u>: Substitute each letter with the letter that is the αth letter after the letter in the sequence AB…Z
- ~~Security was based on the fact that the enc~~ ~~was a secret (Security through obscurity)~~

- Should be avoided at all cost!
- Algorithm should be public and security should come from secret keys.

Secure communication

- <u>Secure communication</u>: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.

- <u>Simple idea (Ceaser Cipher)</u>: Substitute each letter with the letter that is the αth letter after the letter in the sequence AB…Z
- Suppose we make the algorithm public and use the secret key as α. Can you break this protocol?

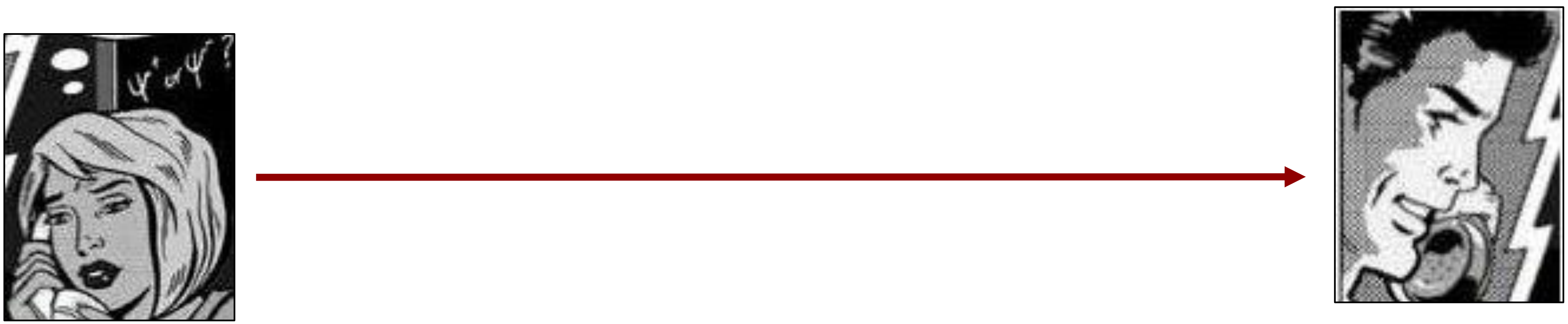
Secure communication

- <u>Secure communication</u>: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.

- <u>Simple idea (Substitution Cipher)</u>: Let π be a permutation of the English letters. Substitute each letter α with the letter $\pi(\alpha)$. π acts as the secret key.
- <u>Example</u>: Let $\pi(A) = U, \pi(B) = T, \pi(C) = P, \ldots$then encryption of CAB is PUT.

Secure communication

- <u>Secure communication</u>: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.



- <u>Simple idea (Substitution Cipher)</u>: Let π be a permutation of the English letters. Substitute each letter α with the letter $\pi(\alpha)$. π acts as the secret key.
- <u>Question</u>: How much space you need to use to store the secret key?

Secure communication

- <u>Secure communication</u>: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.

- <u>Simple idea (Substitution Cipher)</u>: Let π be a permutation of the English letters. Substitute each letter α with the letter $\pi(\alpha)$. π acts as the secret key.
- Consider a brute-force attack where you try to guess the secret key. Is such an attack feasible?

Secure communication

- Secure communication: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.

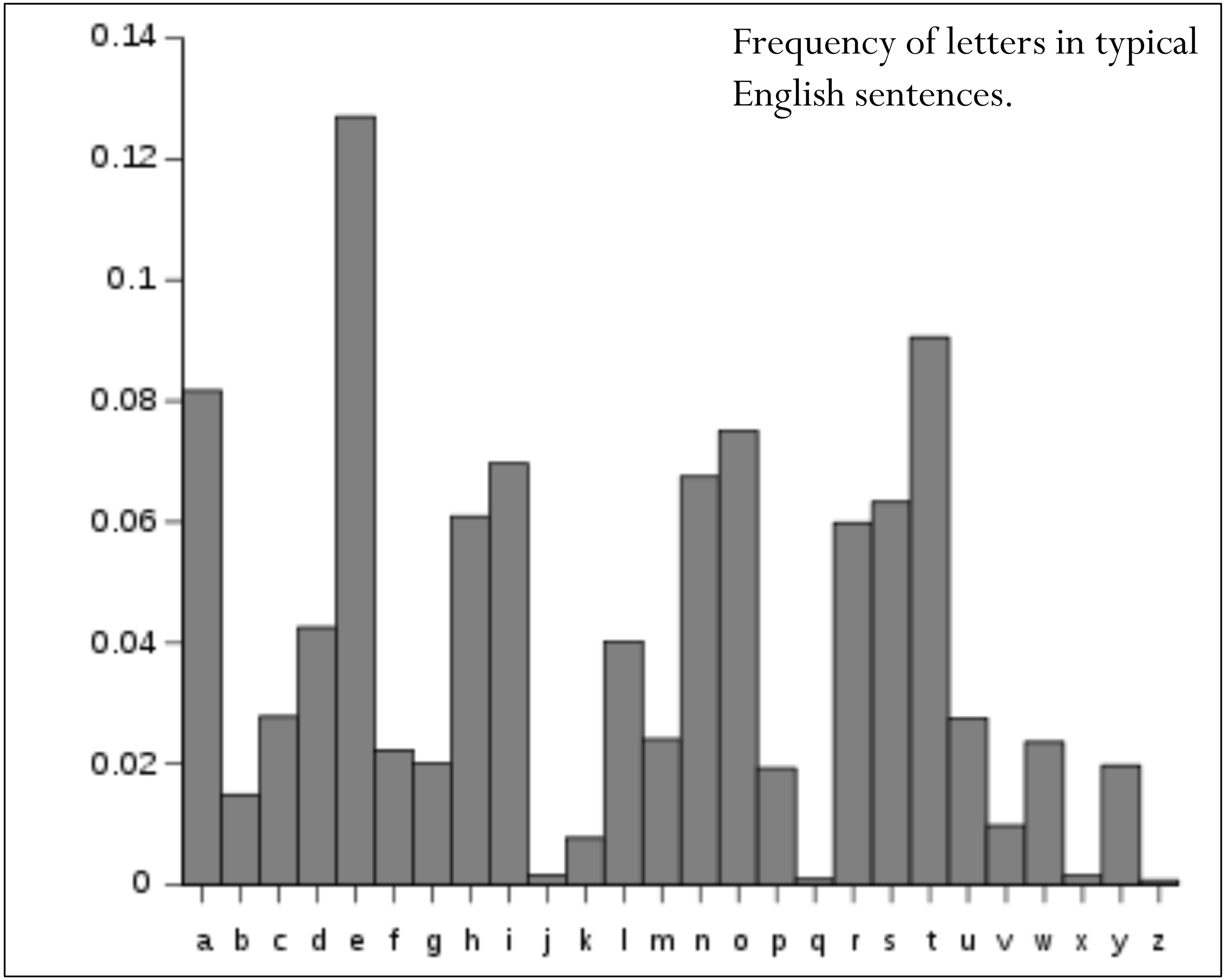
- Simple idea (Substitution Cipher): Let π be a permutation of the English letters. Substitute each letter α with the letter $\pi(\alpha)$.
- Can you break this scheme?

Secure communication

- <u>Secure communication</u>: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.

- <u>Simple idea (Substitution Cipher)</u>: Let π be a permutation of the English letters. Substitute each letter α with the letter $\pi(\alpha)$.
- <u>Attack idea</u>: E's occur more frequently than X's

Secure communication



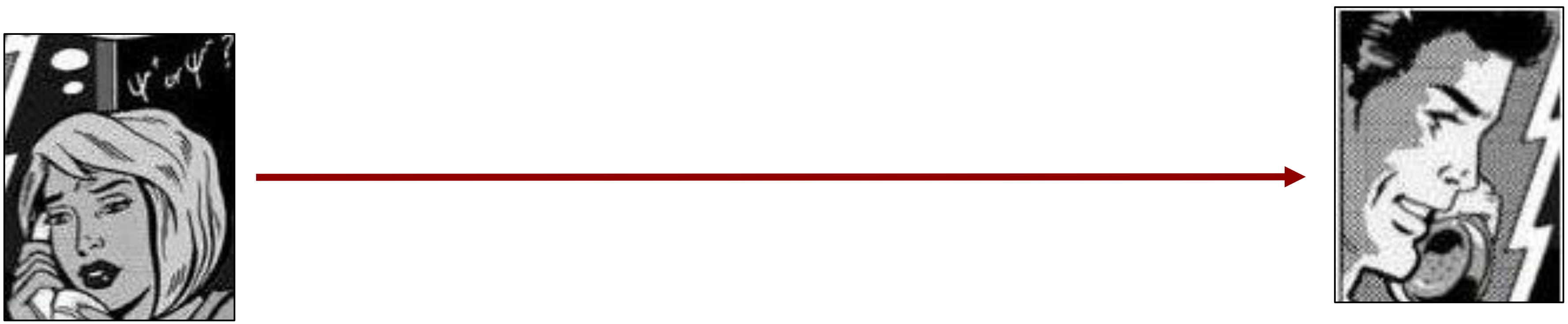
Secure communication

- Secure communication: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.

- Simple idea (One Time Pad(OTP)):Let the message M be an n binary string. Let K be an n bit binary string that is used as a secret key. Add M and K modulo 2 to get the ciphertext.
- Example: $M = 1101, K = 0101$, then $C = M + K \ (mod\ 2) = M \oplus K = 1000$

Secure communication

- <u>Secure communication</u>: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.



- <u>Simple idea (One Time Pad(OTP))</u>:Let the message M be an n binary string. Let K be an n bit binary string that is used as a secret key. Add M and K modulo 2 to get the Ciphertext.
- Can you break this scheme?

Secure communication

- Secure communication: Alice wants to talk to Bob without Eve (who has access to the channel) knowing the communication.
- Perfect Secrecy (Information Theoretic Security):
  - Let the message space be $\{0,1\}^n$.
  - For any two message M_0, M_1, and Ciphertext C
    $$\Pr[E_K(M_0) = C] = \Pr[E_K(M_1) = C]$$
    where the probability is over uniformly random K in the Keyspace.
  - Given the ciphertext, all messages are equally likely to be the secret message

Secure communication

- Perfect Secrecy (Information Theoretic Security):
  - Let the message space be $\{0,1\}^n$.
  - For any two message M_0, M_1, and Ciphertext C
    $$\Pr[E_K(M_0) = C] = \Pr[E_K(M_1) = C]$$
    where the probability is over uniformly random K in the Keyspace.
- One Time Pad (OTP):
  - The Keyspace is $\{0, 1\}^n$.
  - $E_K(M) = K \oplus M$
  - $D_K(C) = K \oplus C$
  - For any messages M_0, M_1 and ciphertext C:
    $$\Pr[E_K(M_0) = C] = \Pr[E_K(M_1) = C] = ??$$

Secure communication

- Perfect Secrecy (Information Theoretic Security):
  - Let the message space be $\{0,1\}^n$.
  - For any two message M_0, M_1, and Ciphertext C

$$\Pr[E_K(M_0) = C] = \Pr[E_K(M_1) = C]$$

    where the probability is over uniformly random K in the Keyspace.
- One Time Pad (OTP):
  - The Keyspace is $\{0, 1\}^n$.
  - $E_K(M) = K \oplus M$
  - $D_K(C) = K \oplus C$
  - For any messages M_0, M_1 and ciphertext C:

$$\Pr[E_K(M_0) = C] = \Pr[E_K(M_1) = C] = 1/2^n$$

Secure communication

- Perfect Secrecy (Information Theoretic Security):
  - Let the message space be $\{0,1\}^n$.
  - For any two message M_0, M_1, and Ciphertext C
    $$\Pr[E_K(M_0) = C] = \Pr[E_K(M_1) = C]$$
    where the probability is over uniformly random K in the Keyspace.
- One Time Pad (OTP):
  - The Keyspace is $\{0,1\}^n$.
  - $E_K(M) = K \oplus M$
  - $D_K(C) = K \oplus C$
  - For any messages M_0, M_1 and ciphertext C:
    $$\Pr[E_K(M_0) = C] = \Pr[E_K(M_1) = C] = 1/2^n$$
  - <u>Disadvantage</u>: Key is as long as the message.

Secure communication

- Perfect Secrecy (Information Theoretic Security):
  - Let the message space be $\{0,1\}^n$.
  - For any two message M_0, M_1, and Ciphertext C
    $$\Pr[E_K(M_0) = C] = \Pr[E_K(M_1) = C]$$
    where the probability is over uniformly random K in the Keyspace.
- One Time Pad (OTP):
  - The Keyspace is $\{0, 1\}^n$.
  - $E_K(M) = K \oplus M$
  - $D_K(C) = K \oplus C$
  - For any messages M_0, M_1 and ciphertext C:
    $$\Pr[E_K(M_0) = C] = \Pr[E_K(M_1) = C] = 1/2^n$$
  - <u>Disadvantage</u>: Key is as long as the message.
- <u>Fact</u>: If $|M| > |K|$, then no scheme is perfectly secure.

Secure communication

- Perfect Secrecy (Information Theoretic Security):
  - Let the message space be $\{0,1\}^n$.
  - For any two message M_0, M_1, and Ciphertext C
    $$\Pr[E_K(M_0) = C] = \Pr[E_K(M_1) = C]$$
    where the probability is over uniformly random K in the Keyspace.
- Fact: If $|M| > |K|$, then no scheme is perfectly secure.
- How do we get around this problem?

Secure communication

- Perfect Secrecy (Information Theoretic Security):
  - Let the message space be $\{0,1\}^n$.
  - For any two message M_0, M_1, and Ciphertext C
    $$\Pr[E_K(M_0) = C] = \Pr[E_K(M_1) = C]$$
    where the probability is over uniformly random K in the Keyspace.
- Fact: If $|M| > |K|$, then no scheme is perfectly secure.
- How do we get around this problem?
  - Relax our notion of security: Instead of saying "it is impossible to break the scheme", we would like to say "it is *computationally infeasible* to break the scheme".

Pseudorandom generator

- Suppose there was a *generator* that *stretches* random bits.

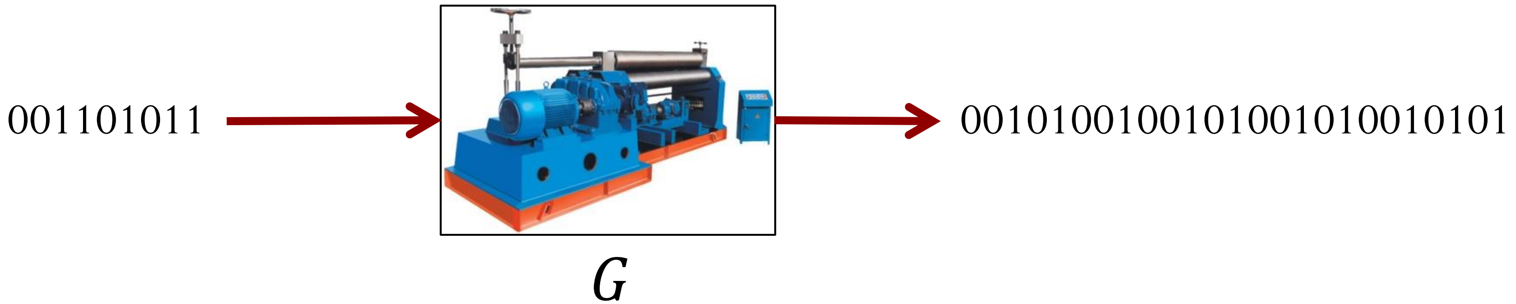
001101011 →  → 0010100100101001010010101011

G

- Idea:
  - Choose a short key K randomly.
  - Obtain $K' = G(K)$.
  - Use K' as key for the one time pad.
- Issue: ?

Pseudorandom generator

- Suppose there was a *generator* that *stretches* random bits.



- Idea:
  - Choose a short key K randomly.
  - Obtain $K' = G(K)$.
  - Use K' as key for the one time pad.
- Issue:
  - Such a generator is not possible!
  - Any such generator produces a longer string but the string is not *random*.

Pseudorandom generator

- Suppose there was a *generator* that *stretches* random bits.

001101011 →  → 001010010010100101001010111

G

- Idea:
  - Choose a short key K randomly.
  - Obtain $K' = G(K)$.
  - Use K' as key for the one time pad.
- Issue:
  - Such a generator is not possible!
  - Any such generator produces a longer string but the string is not *random*.
- What if we can argue that the output of the generator is *computationally indistinguishable* from truly random string.

Stream Ciphers

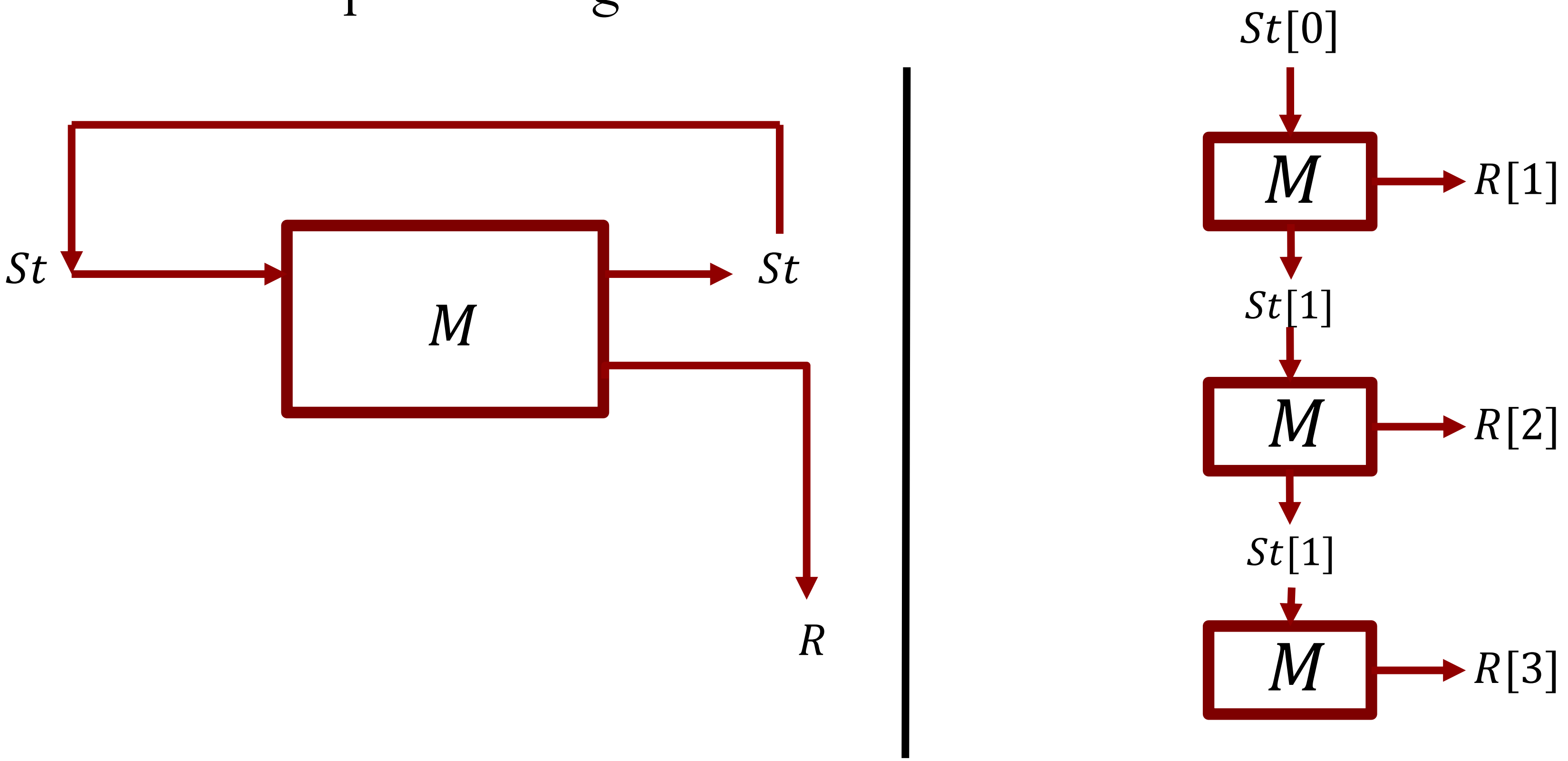
Pseudorandom generators

Stream Ciphers: Pseudorandom generators

- A pseudorandom generator (PRG) is a function:

$$G: \{0,1\}^s \rightarrow \{0,1\}^n, n \gg s$$

  such that $G(x)$ "appears" to be a random n bit string.
- The input to the generator is called the *seed*.



Stream Ciphers: Pseudorandom generators

- A pseudorandom generator (PRG) is a function:

$$G: \{0,1\}^s \rightarrow \{0,1\}^n, n \gg s$$

  such that $G(x)$ "appears" to be a random n bit string.
- Let us see if we can rule out some popular random generators based on this intuitive understanding of PRG:
  - Linear Congruential Generator (LCG): parameters m, a, c:
    - $R_n = (a \cdot R_{n-1} + c)(mod\ m)$, the seed is R_0 and the output is $R_1 R_2 R_3 \ldots$
    - This has some nice statistical properties but it is "predictable".
    - Never use such "predictable" random number generators for Cryptography.

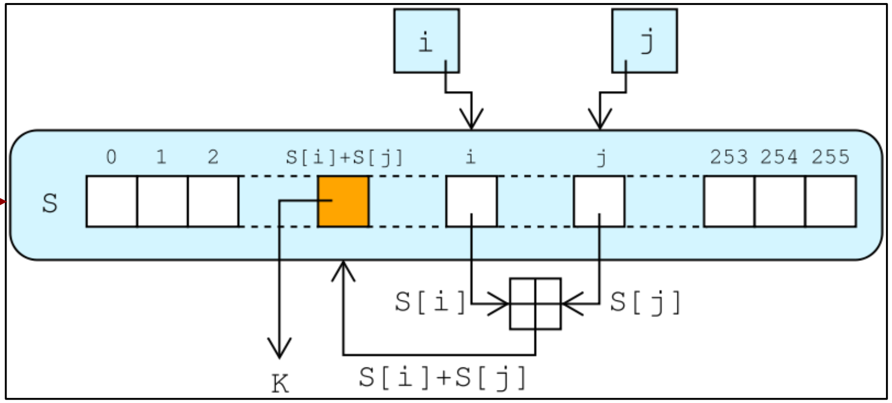
Stream Ciphers: Pseudorandom generators

- Let us see if we can rule out some popular random generators based on this intuitive understanding of PRG:
  - Linear Congruential Generator(LCG):
  - RC4: Used in SSL and WEP

K

(128 bits)

Seed used as secret key



Stream Ciphers

- How do we use a stream cipher?

$$M$$
$$\oplus$$
$$PRG(K)$$

$$M \oplus PRG(K)$$

- What is the issue with this idea?
  - What if there are more than one message that you want to encrypt?

Stream Ciphers

- How do we use a stream cipher?

$$M$$

$$\oplus$$

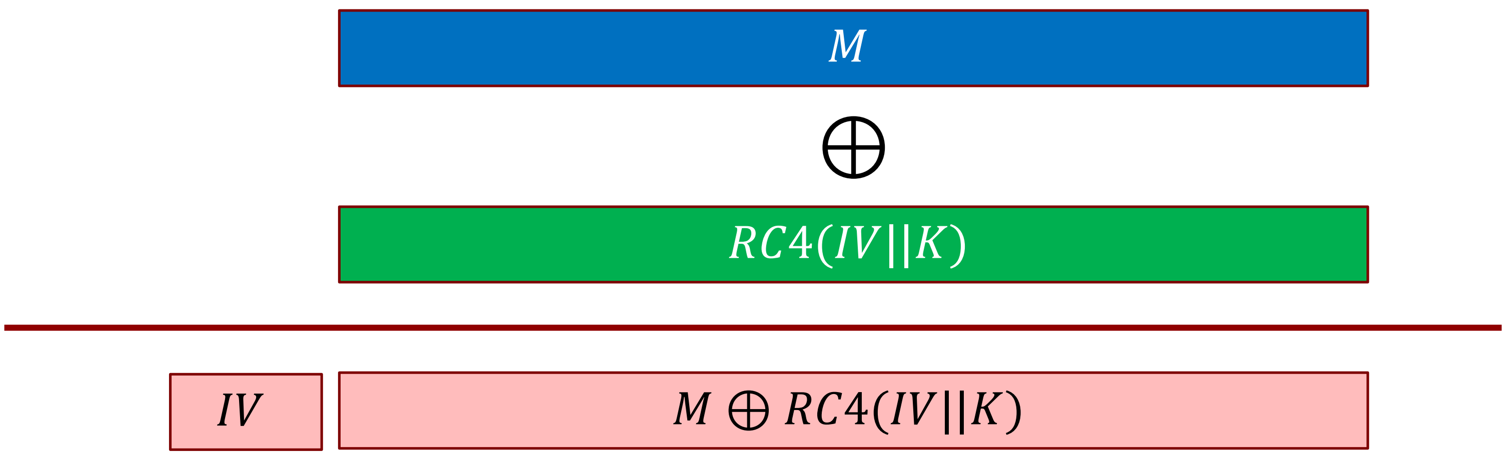
$$PRG(K)$$

$$M \oplus PRG(K)$$

- What is the issue with this idea?
  - What if there are more than one message that you want to encrypt?
  - ***Key reusability should always be avoided when using stream ciphers.***

Stream Ciphers

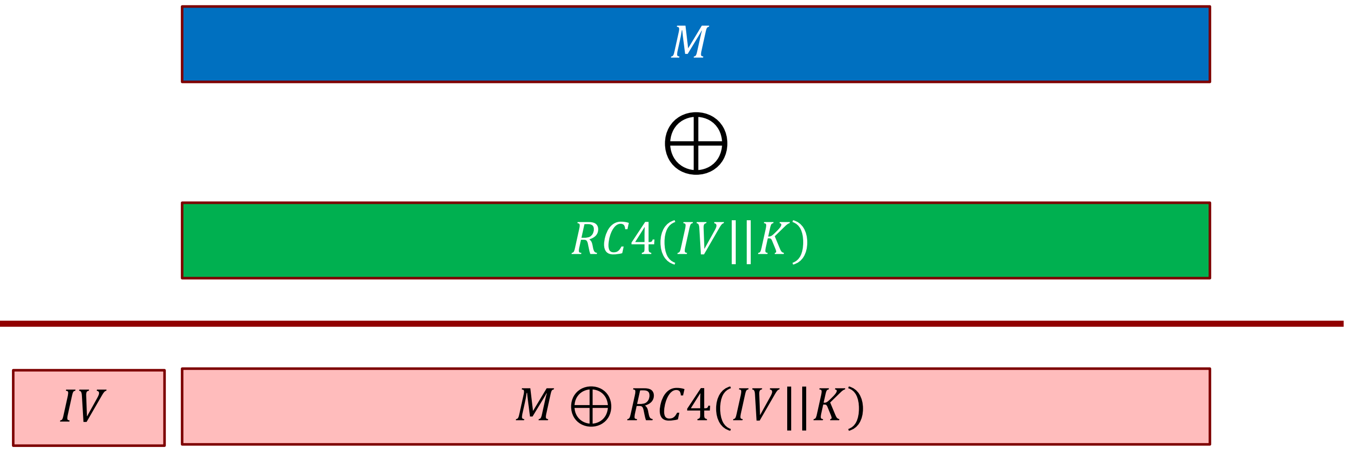
- How do we use a stream cipher?
  - Another idea: This is actually used in 128 bit WEP where $|IV| = 24$ and $|K| = 104$.



- What is the issue with the above protocol?
  - The IV gets repeated after 2^{24} frames.
  - In some 802.11 cards, the IV is set to 0 after every power cycle.

Stream Ciphers

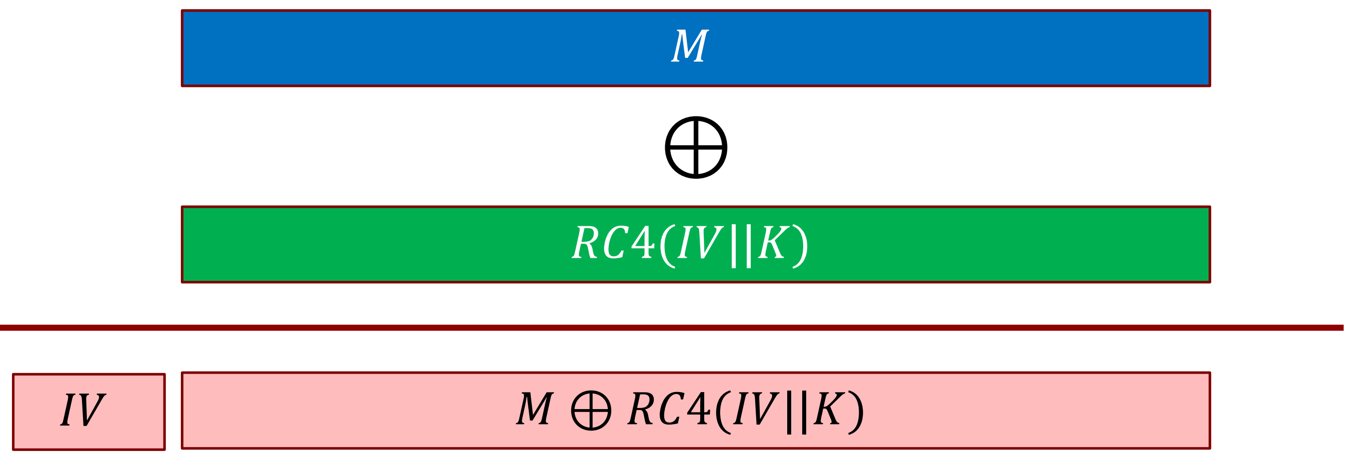
- How do we use a stream cipher?
  - Another idea: This is actually used in 128 bit WEP where $|IV| = 24$ and $|K| = 104$.



- What is the issue with the above protocol?
  - The IV gets repeated after 2^{24} frames.
  - In some 802.11 cards, the IV is set to 0 after every power cycle.
  - Related key attack: IV is incremented by 1 for each frame. So, the key though different, are very similar and one may use the correlation property to attack.

Stream Ciphers

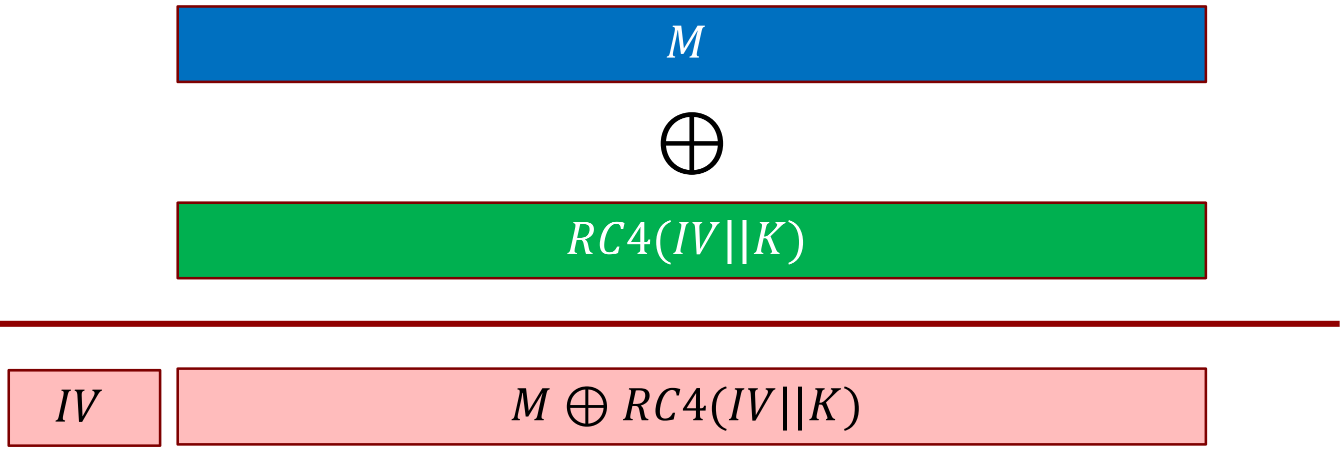
- How do we use a stream cipher?
  - Another idea: This is actually used in 128 bit WEP where $|IV| = 24$ and $|K| = 104$.



128 bit WEP is insecure. DO NOT USE!
There are attacks that will figure out your secret key in less than a minute. Check out *aircrack-ptw*.

Stream Ciphers

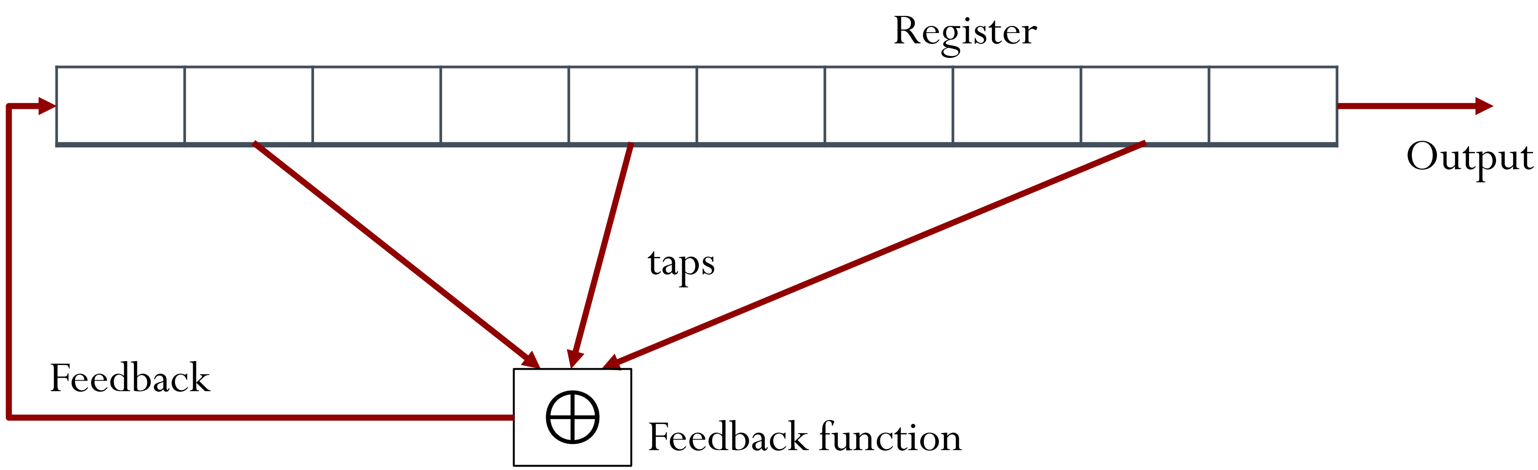
- How do we use a stream cipher?
  - Another idea: This is actually used in 128 bit WEP where $|IV| = 24$ and $|K| = 104$.



- So what is the fix? How do we use PRGs like RC4?
  - Throw away initial few bytes of RC4 output.
  - Use unrelated keys.

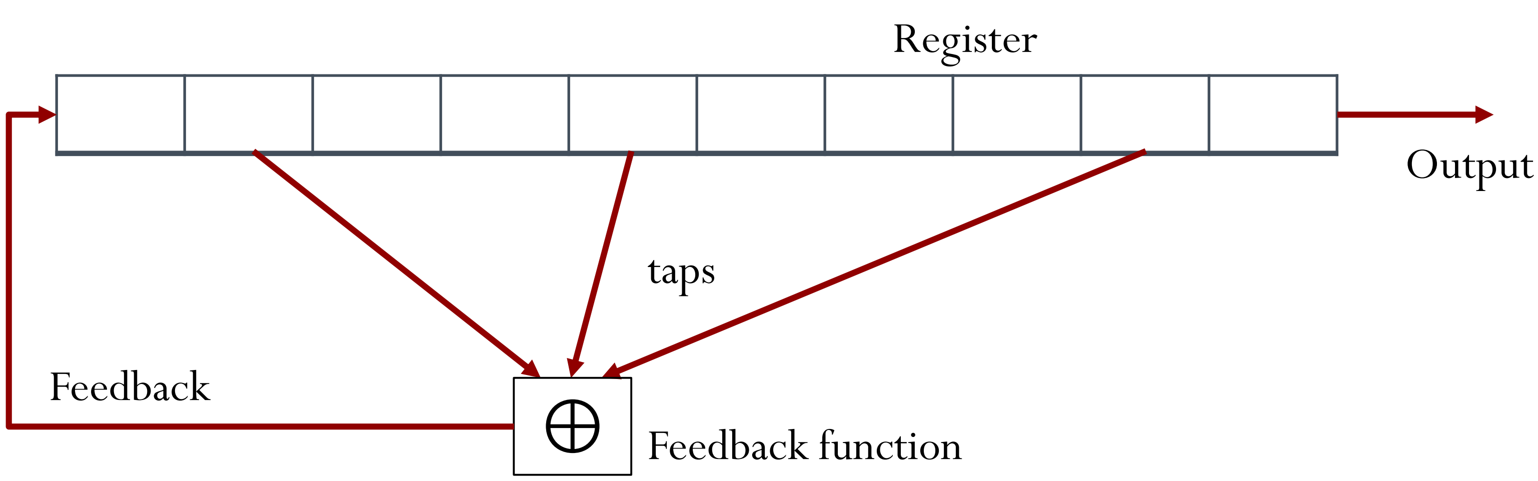
Stream Ciphers: Pseudorandom generators

- Linear Feedback Shift Registers (LFSR):
  - Fast hardware implementation.
  - Examples: DVD encryption (CSS), GSM encryption (A5/1,2).
  - Is this generator predictable?



Stream Ciphers: Pseudorandom generators

- Linear Feedback Shift Registers (LFSR):
  - Fast hardware implementation.
  - Examples: DVD encryption (CSS), GSM encryption (A5/1,2).
  - Is this generator predictable?
    - Yes.
    - One solution that is used in practice is to use a combination of multiple LFSRs.



Block Ciphers

Block Ciphers

- Block ciphers work on “blocks” of message bits rather than a “stream” of message bits.
- Main Idea:
  - Suppose we encrypt in blocks of size n.
  - Let $E: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a function.
  - For a message block M of n bits, and key K, the ciphertext is given by $C = E(K, M)$.

Block Ciphers

- Block ciphers work on "blocks" of message bits rather than a "stream" of message bits.
- Main Idea:
  - Suppose we encrypt in blocks of size n.
  - Let $E: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a function.
  - For a message block M of n bits, and key K, the ciphertext is given by $C = E(K, M)$.
- What are properties that E should satisfy?

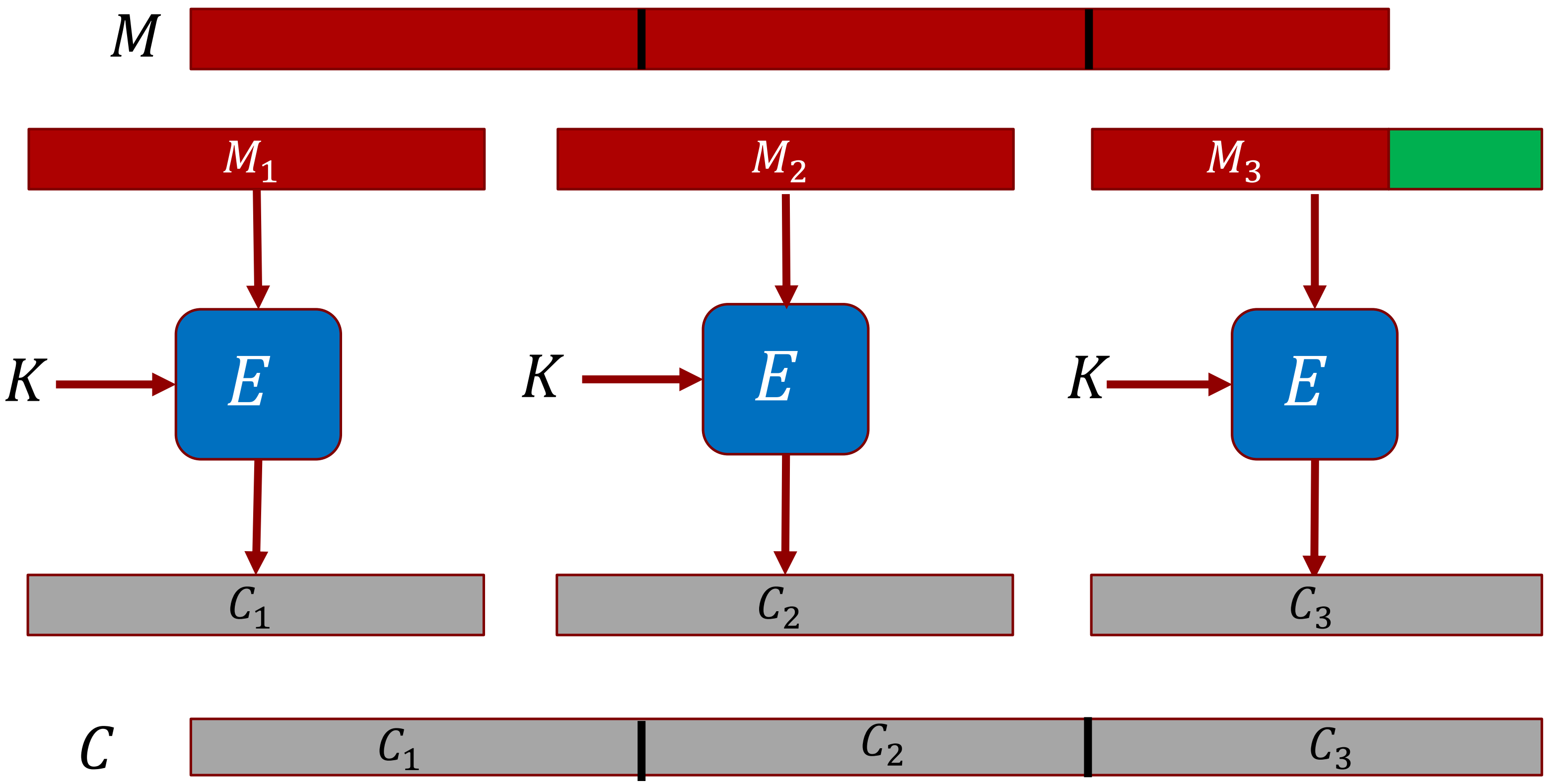
Block Ciphers

- Block ciphers work on "blocks" of message bits rather than a "stream" of message bits.
- Main Idea:
  - Suppose we encrypt in blocks of size n.
  - Let $E: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a function.
  - For a message block M of n bits, and key K, the ciphertext is given by $C = E(K, M)$.
- What are properties that E should satisfy?
  - For all $K \in \{0,1\}^k$, the function $\mathrm{E_K}: \{0,1\}^{\mathrm{n}} \rightarrow \{0,1\}^n$ defined as $E_K(M) = E(K, M)$ is a one-one function. In other words, E_K is a permutation.

Block Ciphers

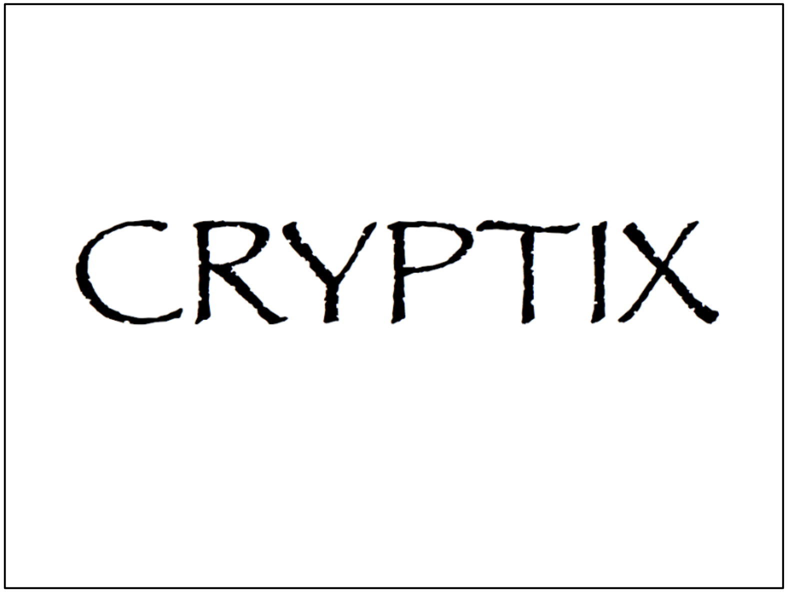
- Block ciphers work on "blocks" of message bits rather than a "stream" of message bits.
- Main Idea:
  - Suppose we encrypt in blocks of size n.
  - Let $E: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a function.
  - For a message block M of n bits, and key K, the ciphertext is given by $C = E(K, M)$.
- What are properties that E should satisfy?
  - For all $K \in \{0,1\}^k$, the function $\mathrm{E_K}: \{0,1\}^{\mathrm{n}} \rightarrow \{0,1\}^n$ defined as $E_K(M) = E(K, M)$ is a one-one function. In other words, E_K is a permutation.
  - Both E_K (encryption function) and E_K^{-1} (decryption function) are efficient.
  - E should be computationally indistinguishable from a random permutation.

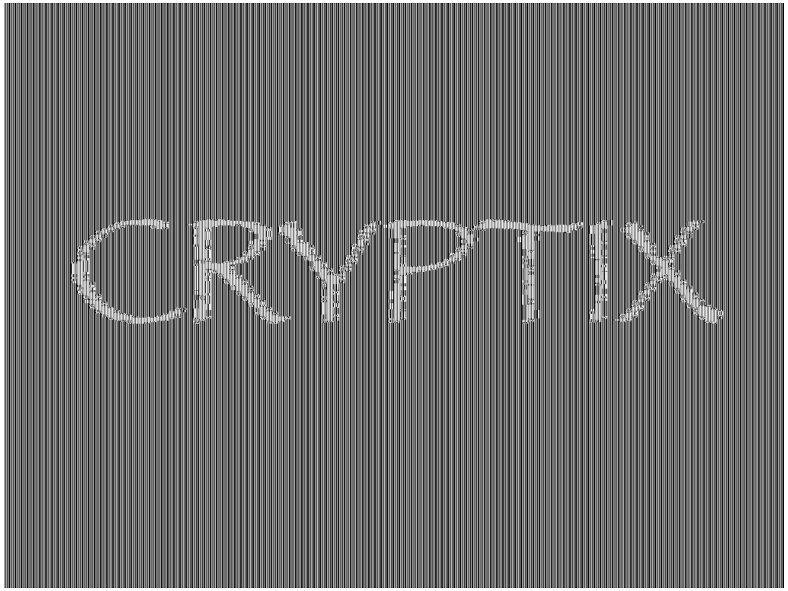
ECB Mode: Electronic Codebook Mode



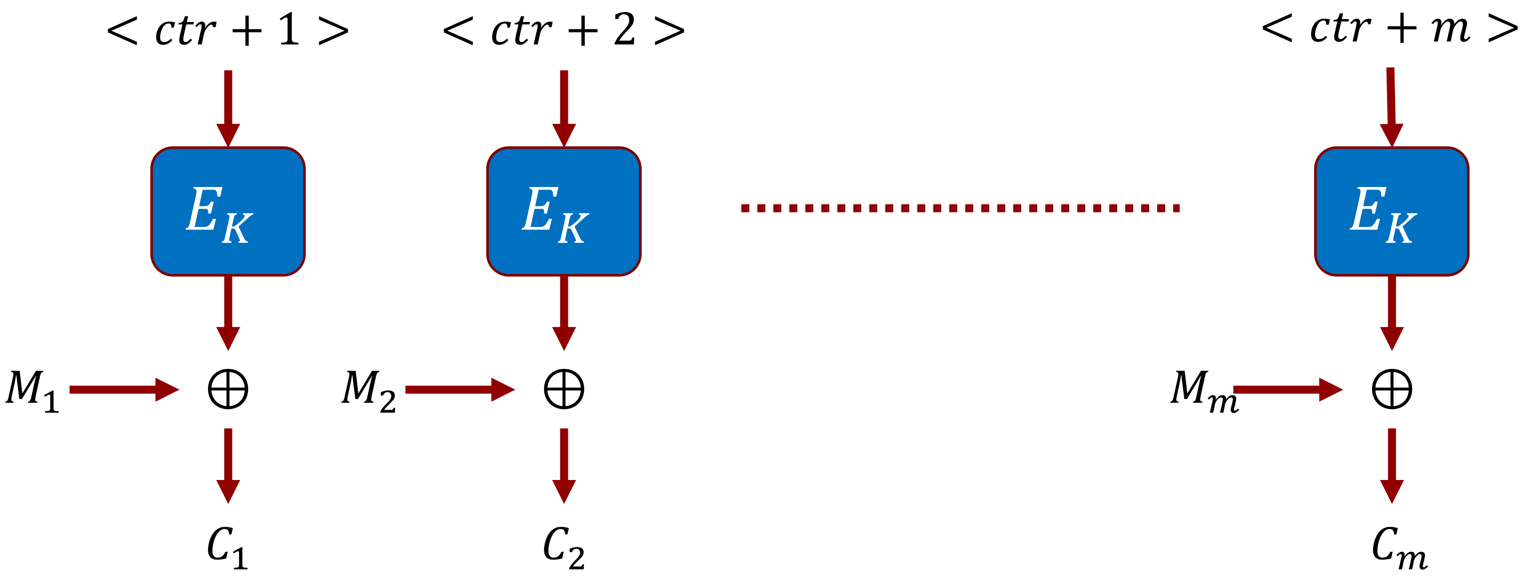
- Is the encryption scheme using the ECM mode secure?

ECB Mode: Electronic Codebook Mode



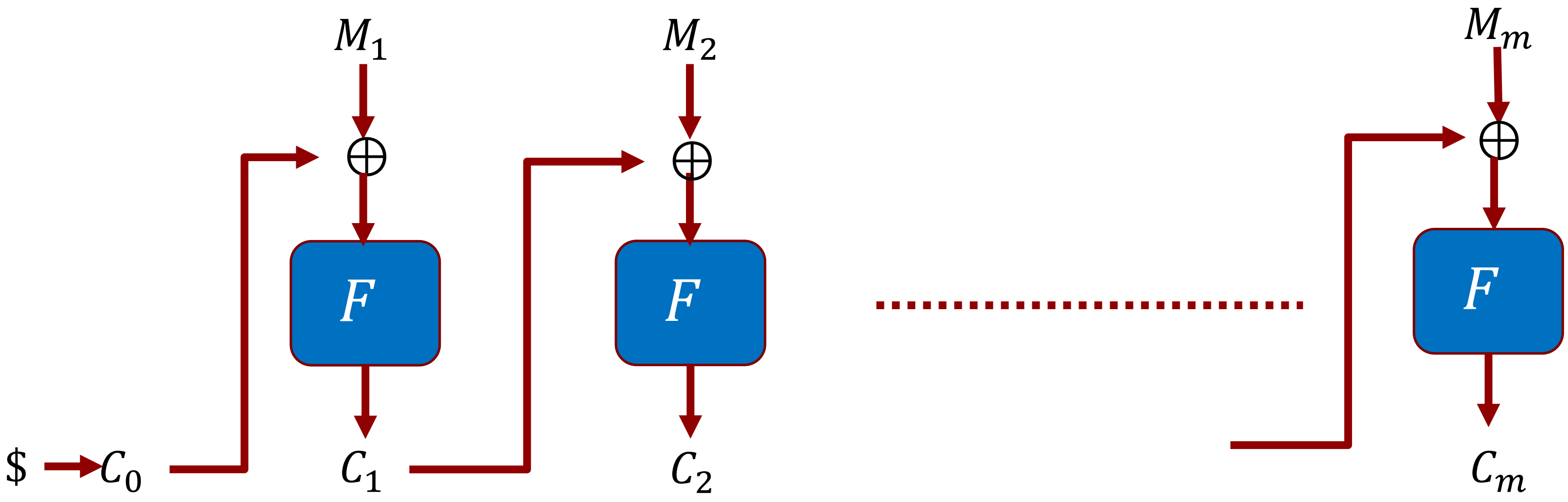


CTRC Mode



- The encryption algorithm maintains a counter ctr that is initialized to 0.
- For a m block message $M_1, \ldots, M_m$ the ciphertext $C_0, C_1, \ldots C_m$ is sent where $C_0 = ctr$.

CBC$ Mode



- C_0 is chosen randomly from $\{0,1\}^n$.
- The ciphertext corresponding to $M_1, \dots, M_m$ is $C_0, C_1, \dots, C_m$.
- E_K needs to be a block cipher (i.e., it should be invertible).

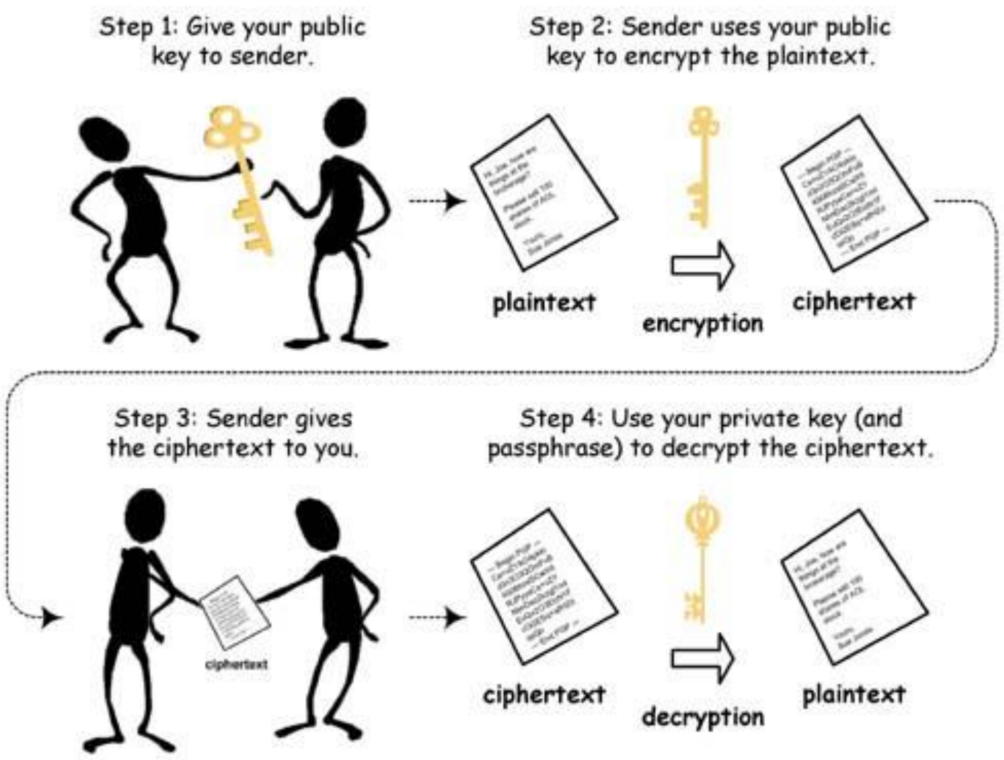
Key Distribution/Exchange

K

K

- How do Alice and Bob share a secret key in the first place?

Public key cryptography



- Generate a **pair** of related keys. One is called public key and other the secret key.
- <u>Examples</u>: RSA, El-gamal (using number theory you learnt in Discrete Math).

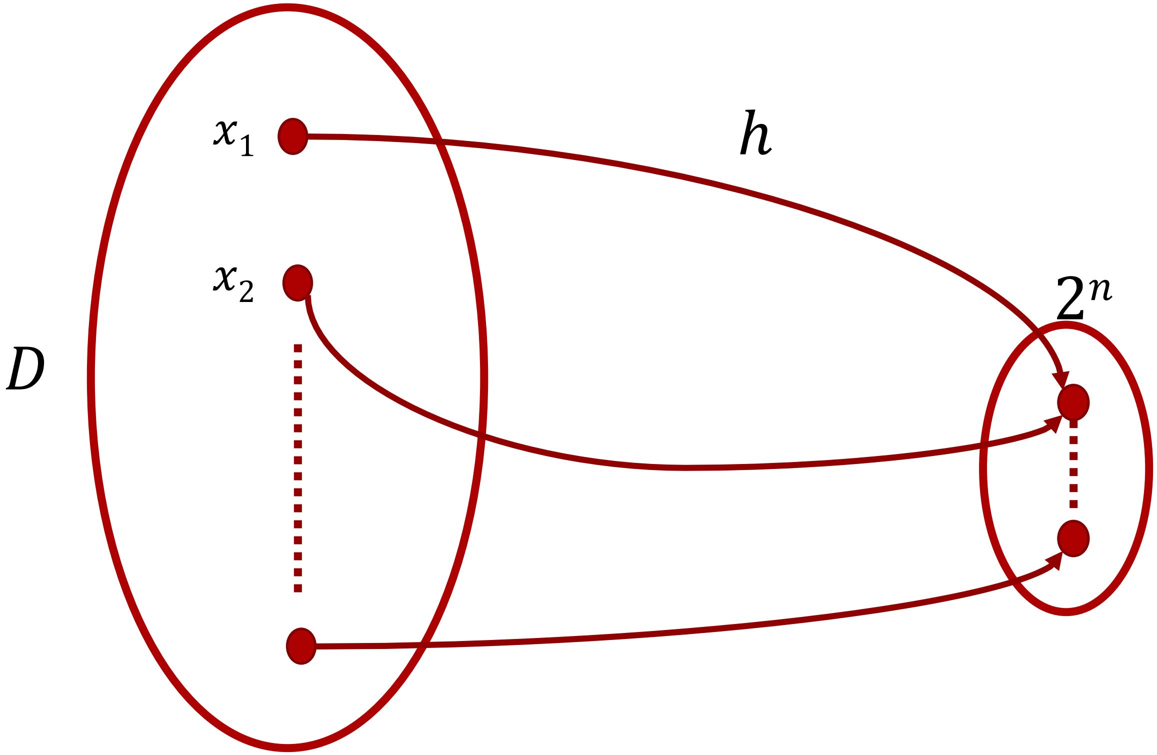
Hash Functions

Hash Functions: Introduction

- A hash function is a map $h: D \rightarrow \{0,1\}^n$ that is compressing, i.e., $|D| > 2^n$.
- Usually $|D| \gg 2^n$ and n is small.
  - Example:
    - $D = \{0,1\}^{\leq 2^{64}}$ i.e., all binary strings of length at most 2^{64}.
    - $n = 128, 160, 256$ etc.
- Examples of Cryptographic Hash Functions:

h	n
MD4	128
MD5	128
SHA1	160
SHA-256	256
SHA-512	512
WHIRLPOOL	512

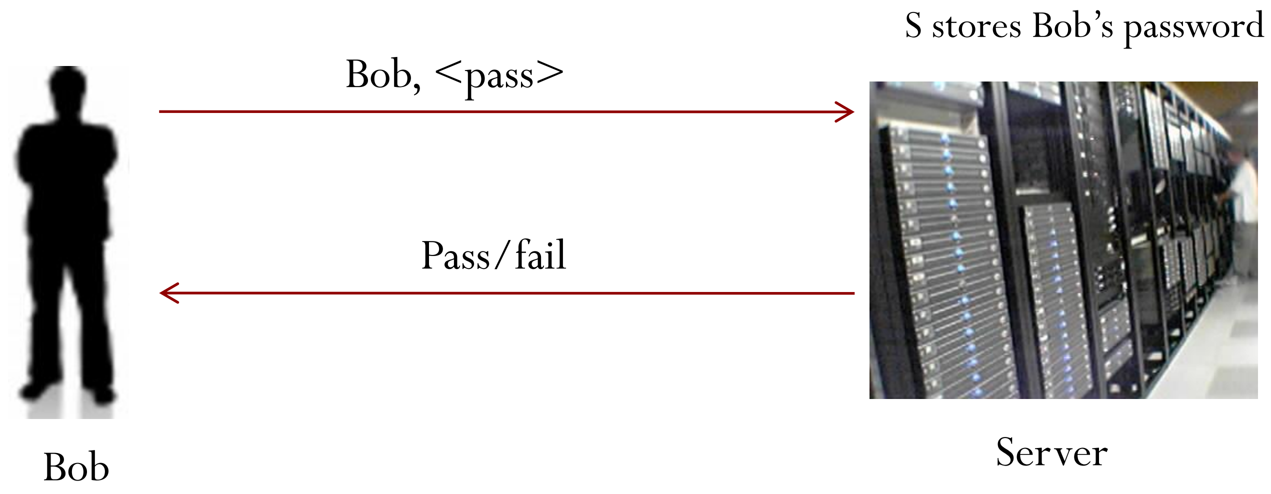
Hash Functions: Collision



Pigeonhole Principle: $h(x_1) = h(x_2), x_1 \neq x_2$

Hash Functions: Applications

1. Password Authentication:



- <u>Problem</u>: If Eve hacks into the server or if the communication channel is not secure, then Eve knows the password of Bob.

Hash Functions: Applications

1. Password Authentication:

S stores *h*(Bob's password)

Bob, h(<pass>)

Pass/fail

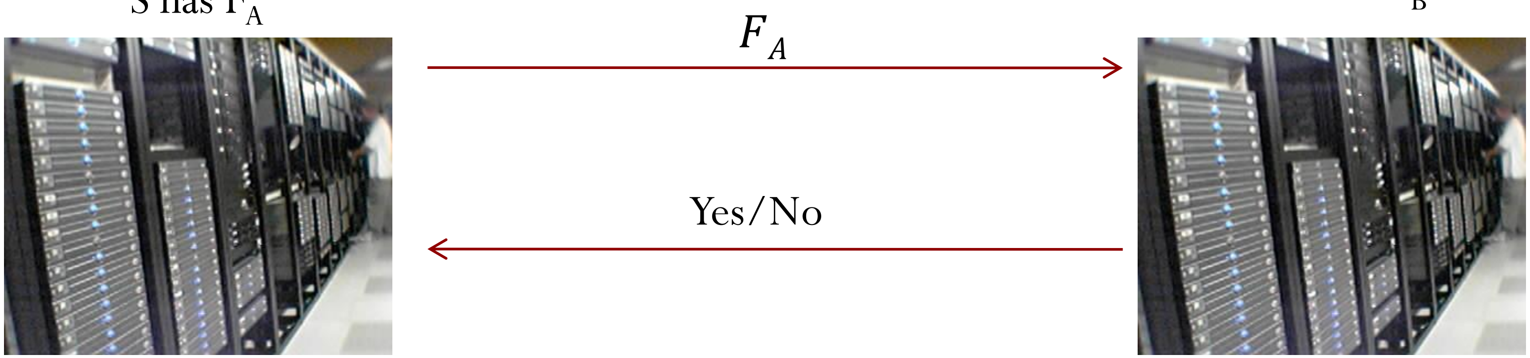
Bob

Server

- Eve can only get access to *h*(<pass>).

Hash Functions: Applications

2. Comparing files by hashing:



• <u>Problem</u>: Files are usually very large and we would like to save communication costs/delays.

Hash Functions: Applications

2. Comparing files by hashing:

S has F_A

Server A

$h(FA)$

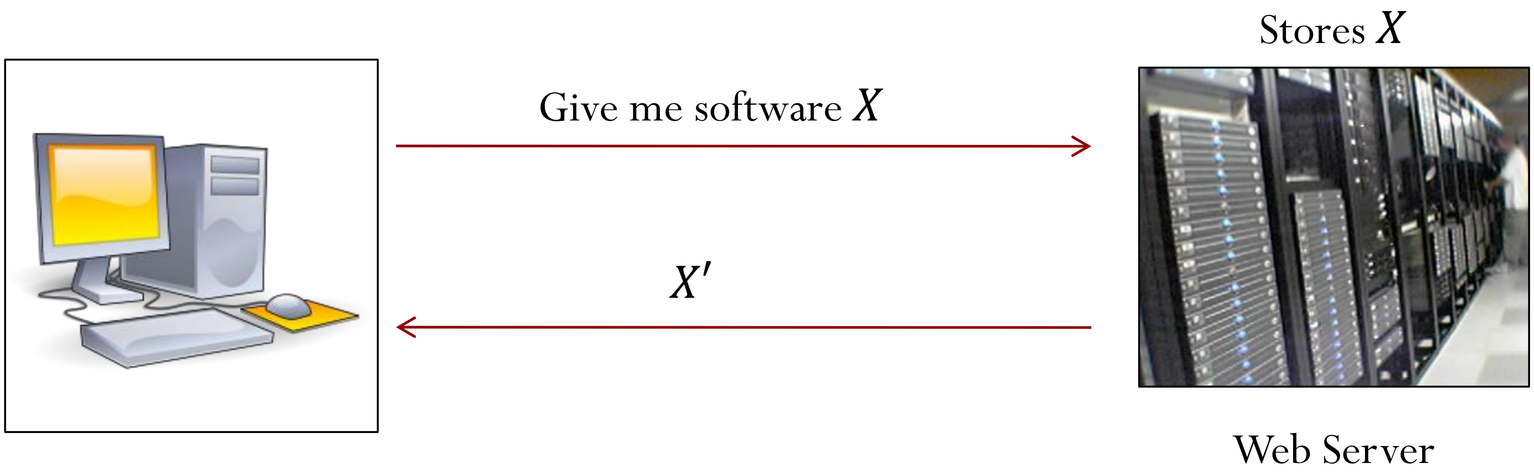
$h(F_A) =?\, h(F_B)$

S has F_B

Server B

Hash Functions: Applications

3. Downloading new software



- <u>Problem</u>: X' could be a virus-infected version of X.

Hash Functions: Applications

3. Downloading new software

Stores X,
Also stores $h(X)$ in read-only mode

Give me software X

$X', h(X)$

Web Server

Collision Resistance

- Password Authentication: If Eve is able to find a string S (perhaps different from $< pass >$) such that
$$h(S) = h(< pass >)$$
then the scheme breaks.
- Comparing files: If there is a different file F_S such that
$$h(FS) = h(FB)$$
the servers may agree incorrectly.
- Downloading software: If Eve can find $X' \neq X$ such that $h(X) = h(X')$, then software might cause problems.
- Collision Resistance: It is computationally infeasible to find a pair (x_1, x_2) such that $x_1 \neq x_2$ and
$$h(x_1) = h(x_2)$$
- If a hash function h is collision resistant, then the above two problems are avoided.

Collision Resistance: Discussion

- Are there functions that are collision resistant?
  - Fortunately, there are functions for which no one has been able to find a collision!
  - Example: $SHA - 1\colon \{0,1\}^D \rightarrow \{0,1\}^{160}$
- Is the world drastically going to change if someone finds one or few collision for SHA-1?
  - Not really. Suppose the collision has some very specific structure, then we may avoid such structures in the strings on which the hash function is applied.
  - On the other hand, if no one finds a collision then that is a very strong notion of security and we may sleep peacefully without worrying about maintaining complicated structures in the strings.
  - We are once again going for a very strong definition of security for our new primitive similar to Block Ciphers and Symmetric Encryption.

End