Public-Key Cryptography and Message Authentication

Hemant Ghayvat
Department of Computer Science and Media Technology
hemant.ghayvat@lnu.se

Modes of Operation

- Block ciphers encrypt fixed size blocks
  - eg. DES encrypts 64-bit blocks, with 56-bit key
- Need way to use in practise, given usually have arbitrary amount of information to encrypt
  - Partition message into separate block for ciphering
- A **mode of operation** describes the process of encrypting each of these blocks **under a single key**

Some modes may use randomized addition input value

Five Modes of Operation

- Electronic codebook mode (ECB)
- Cipher block chaining mode (CBC) – most popular
- Output feedback mode (OFB)
- Cipher feedback mode (CFB)
- Counter mode (CTR)

Electronic Code Book (ECB)

The plaintext is broken into blocks, P_1, P_2, P_3, ...

Each block is encrypted independently:

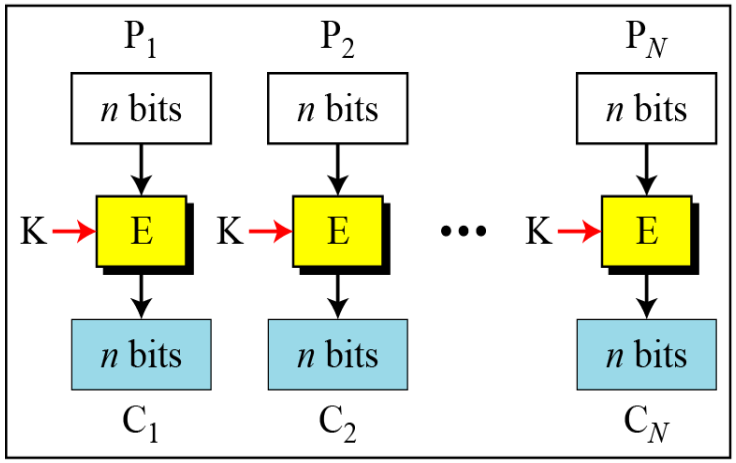
$$C_i = E_K(P_i)$$

For a given key, this mode behaves like we have a gigantic codebook, in which each plaintext block has an entry, hence the name Electronic Code Book

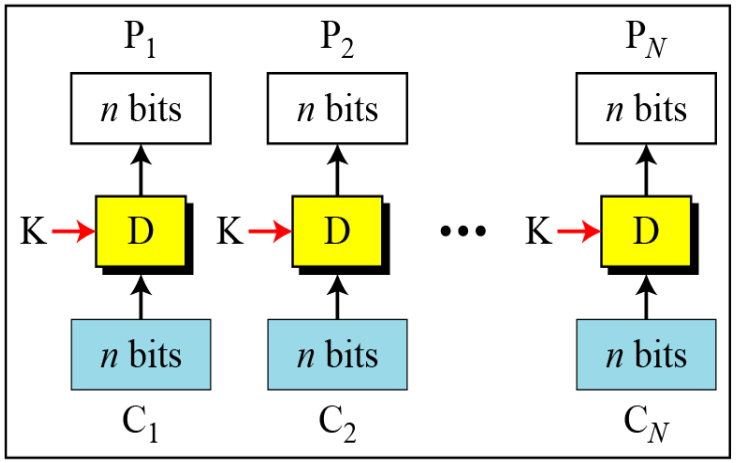
ECB Scheme

Encryption: $C_i = E_K (P_i)$ Decryption: $P_i = D_K (C_i)$

E: Encryption D: Decryption
P_i: Plaintext block i C_i: Ciphertext block i
K: Secret key



Encryption



Decryption

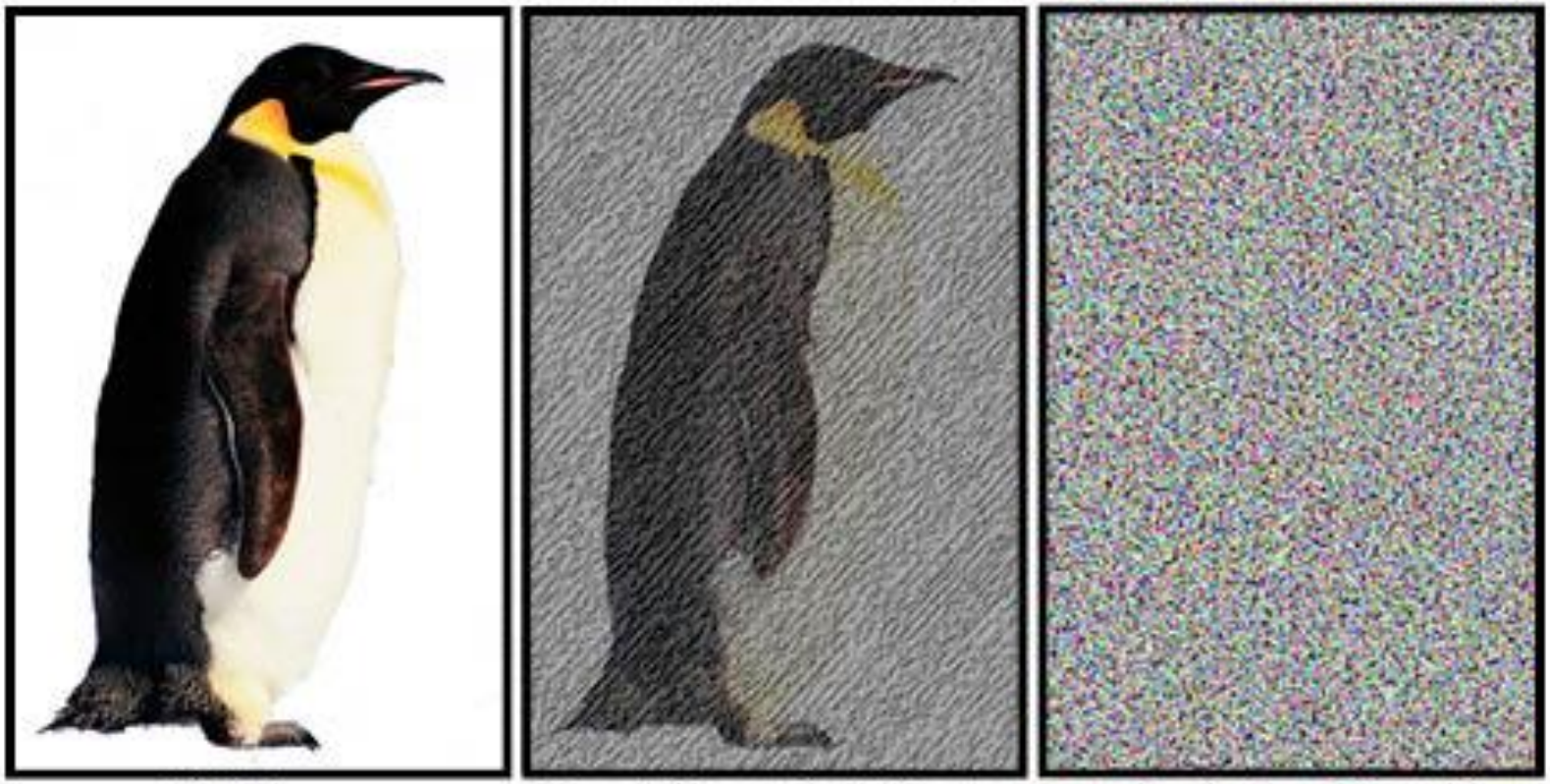
Remarks on ECB

- Strength: it's simple.

- Weakness:
  - Repetitive information contained in the plaintext may show in the ciphertext, if aligned with blocks.
  - If the same message (e.g., an SSN) is encrypted (with the same key) and sent twice, their ciphertexts are the same.

- Typical application: secure transmission of short pieces of information (e.g. a temporary encryption key)

Example of ECB



Original Image

Encrypted image using ECB mode

Modes other than ECB result in pseudorandomness

Cipher Block Chaining (CBC)

- Solve security deficiencies in ECB
  - Repeated same plaintext block result different ciphertext block

- Each previous cipher blocks is chained to be input with current plaintext block, hence name

- Use Initial Vector (IV) to start process

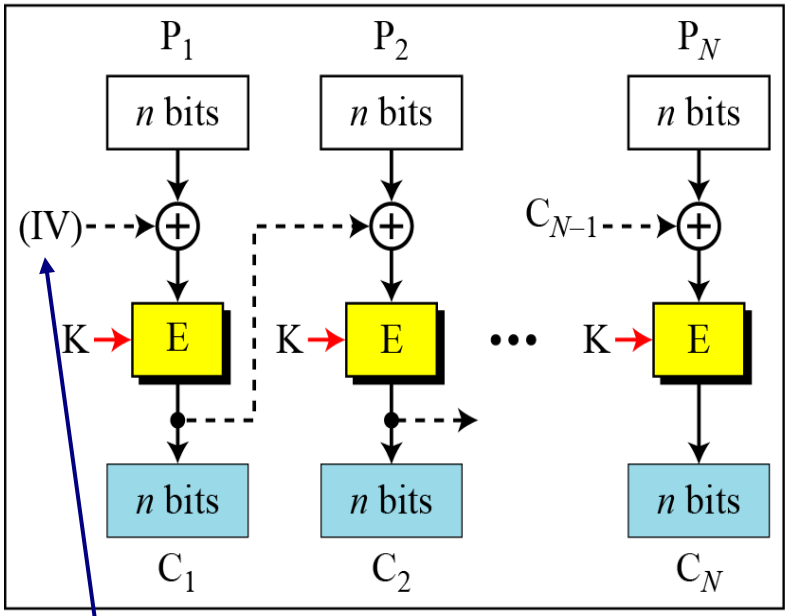
$$C_i = E_K (P_i \text{ XOR } C_{i-1})$$
$$C_0 = IV$$

Uses: bulk data encryption, authentication

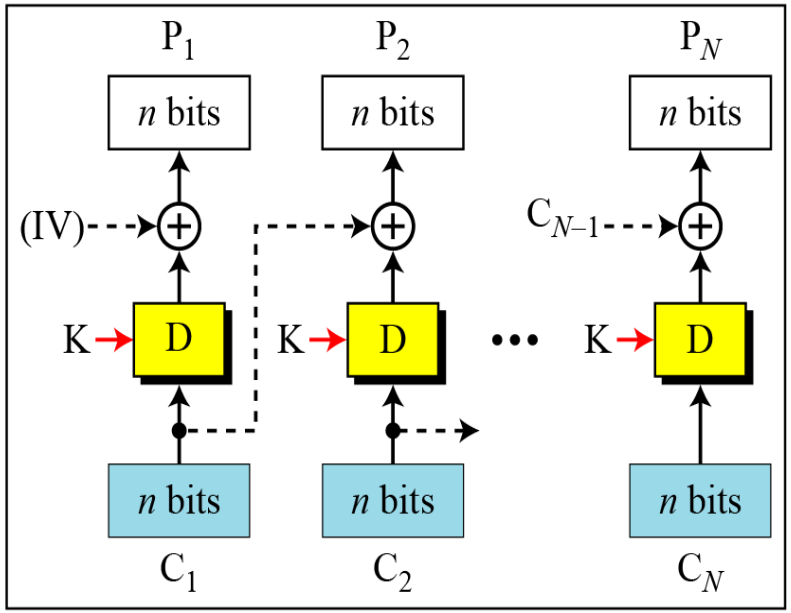
CBC scheme

E: Encryption
P_i: Plaintext block i
K: Secret key

D : Decryption
C_i : Ciphertext block i
IV: Initial vector (C_0)



Encryption



Decryption

Encryption:
$C_0 = IV$
$C_i = E_K (P_i \oplus C_{i-1})$

Decryption:
$C_0 = IV$
$P_i = D_K (C_i) \oplus C_{i-1}$

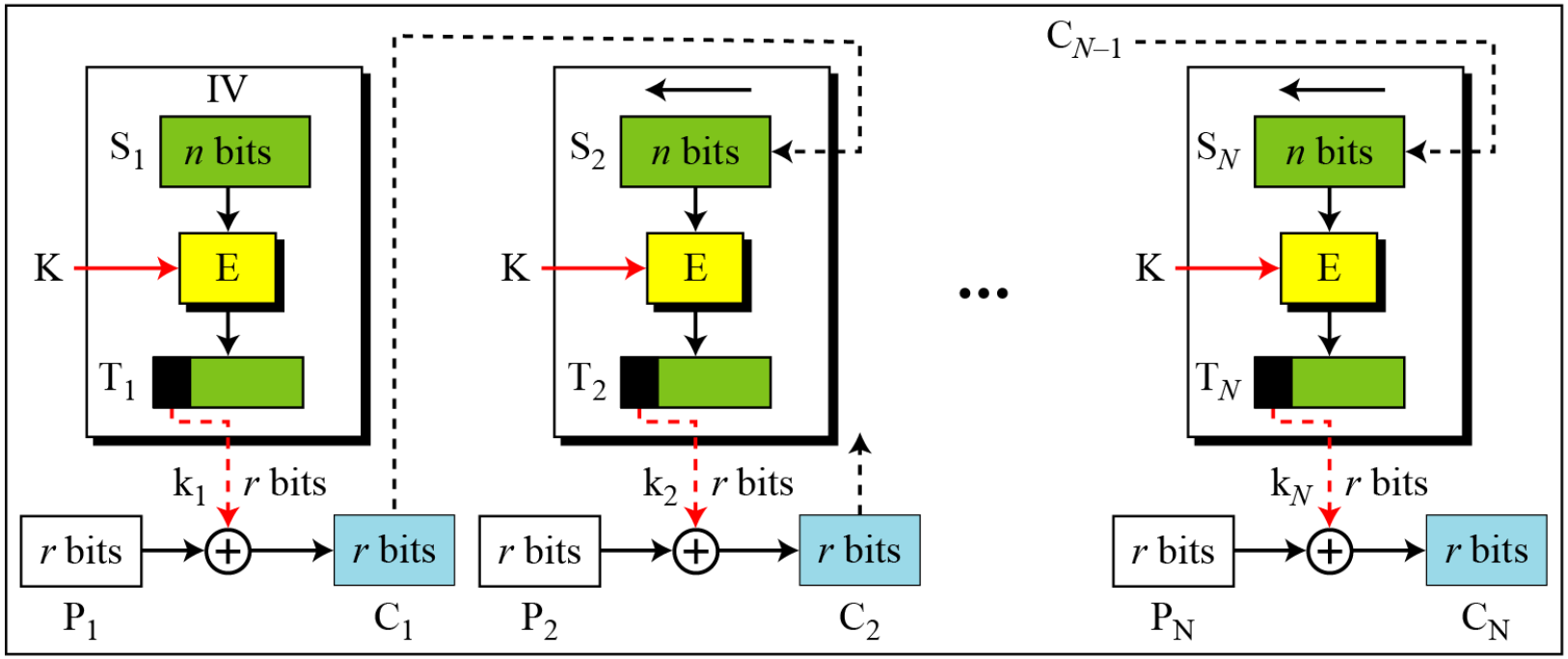
Cipher feedback mode (CFB) Scheme

Encryption: $C_i = P_i \oplus \text{SelectLeft}_r \{E_K [\text{ShiftLeft}_r (S_{i-1}) \mid C_{i-1})]\}$
Decryption: $P_i = C_i \oplus \text{SelectLeft}_r \{E_K [\text{ShiftLeft}_r (S_{i-1}) \mid C_{i-1})]\}$

E : Encryption
P_i: Plaintext block *i*
K: Secret key

D : Decryption
C_i: Ciphertext block *i*
IV: Initial vector (S_1)

S_i: Shift register
T_i: Temporary register



Encryption

Public Key Encryption or Asymmetric


WHY DO WE NEED IT???
memegenerator.net

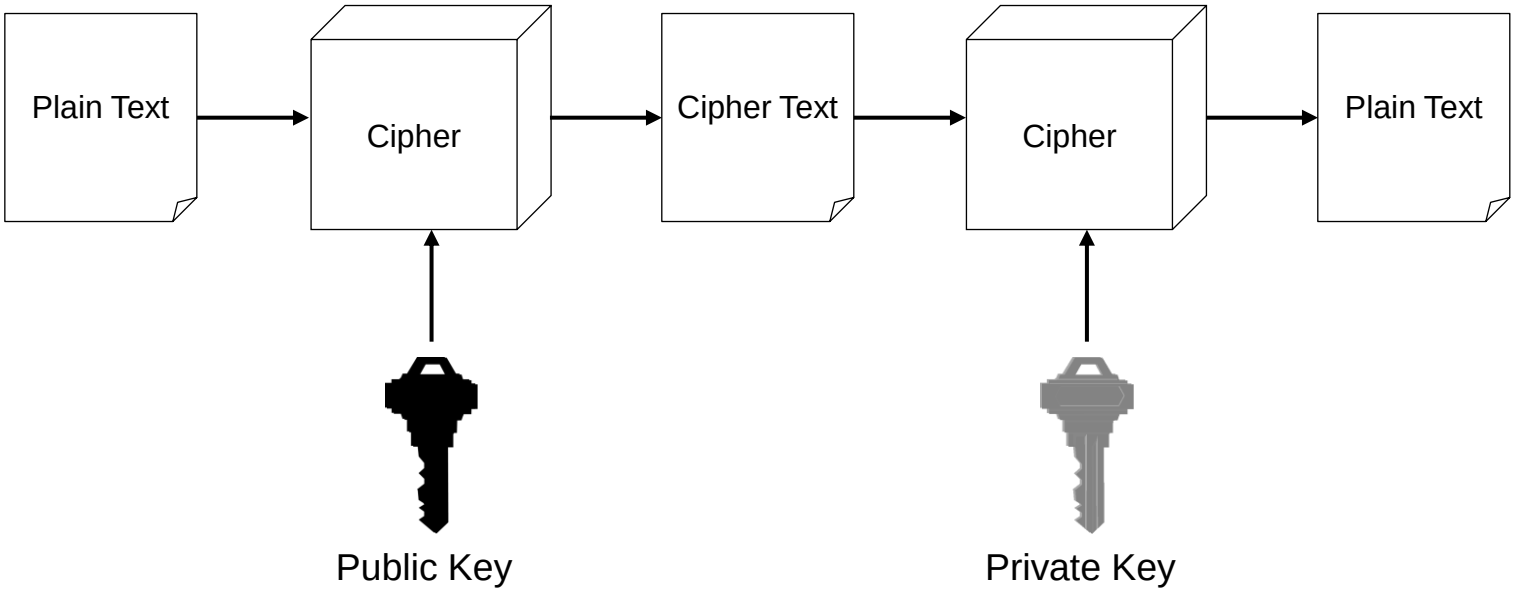
Asymmetric Encryption

Uses a pair of keys for encryption

- Public key for encryption
- Private key for decryption

Messages encoded using public key can only be decoded by the private key

- Secret transmission of key for decryption is not required
- Every entity can generate a key pair and release its public key



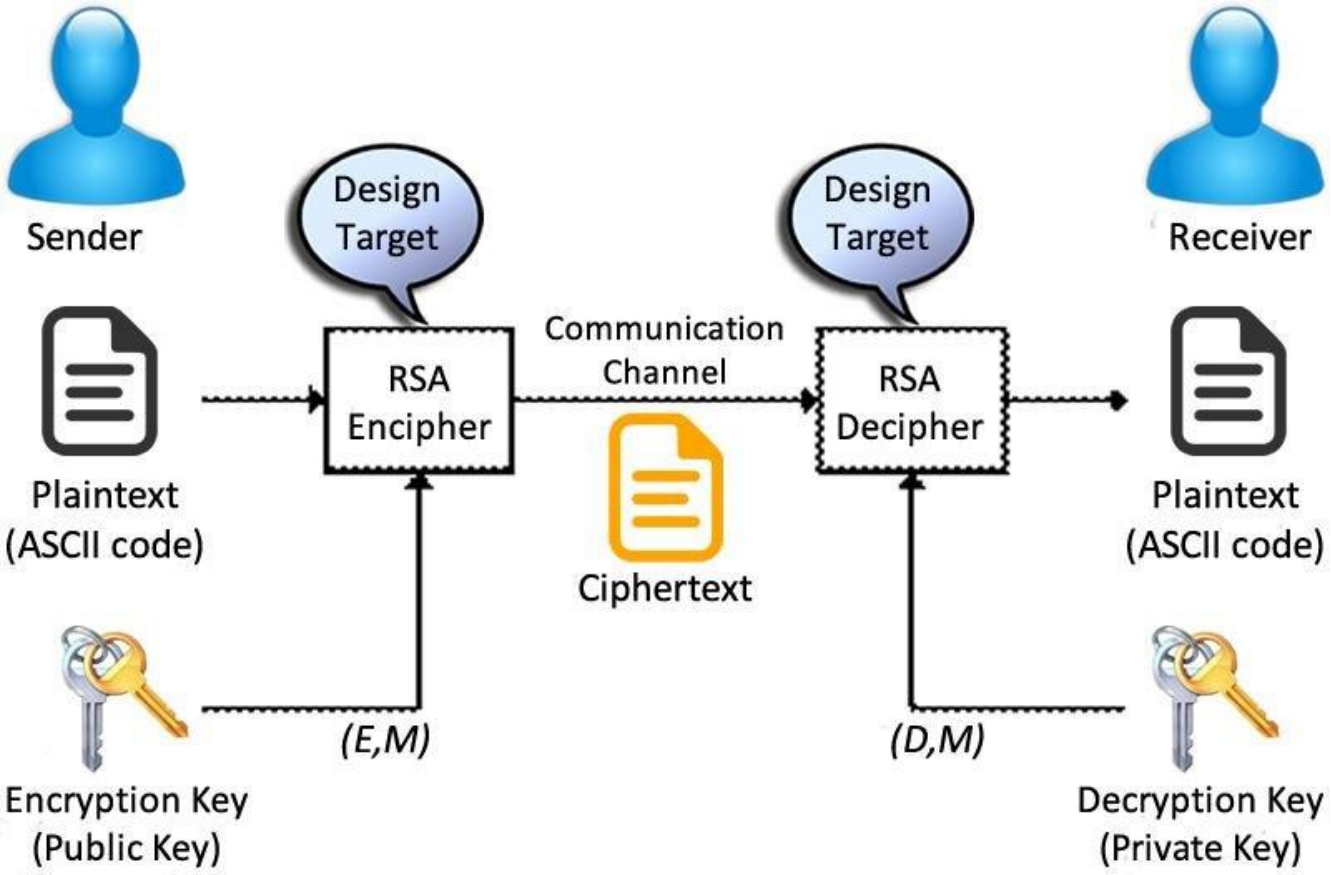
Asymmetric Encryption

Two most popular algorithms are RSA and Diffie-Hellman

- RSA
  - Developed by Ron **Rivest**, Adi **Shamir**, Len **Adelman**
  - Both public and private key are interchangeable
  - Variable Key Size (512, 1024, or 2048 bits)
  - Most popular public key algorithm

- Diffie-Hellman
  - Exchange a secret key securely
  - Compute discrete logarithms

RSA


Sender
Design Target
Design Target
Receiver
Communication Channel
RSA Encipher
RSA Decipher
Plaintext (ASCII code)
Ciphertext
Plaintext (ASCII code)
(E,M)
(D,M)
Encryption Key (Public Key)
Decryption Key (Private Key)

RSA Key Generation

- Two large distinct prime numbers ***p* and *q*, chosen at random**

- Block *B* of text has been encoded by some function g into an integer *T* such that *T* is an integer and $0 < T < n$

  Calculate *n=pq*

- **Compute** the Euler phi function. $\varphi(n) = (p-1)(q-1)$ because *n* is the product of 2 primes.
- **Choose an integer *e*** such that $1 < e < \varphi(n)$ and **$\gcd(e,\varphi(n)) = 1$** (*e* and $\varphi(n)$ are co-prime)

- $\gcd(e,\varphi(n)) = 1$ means that 1 is a linear combination of *e* and $\varphi(n)$:

Use Euclidean algorithm to find unique value for d and such that $1 < d < \varphi(n)$

$$d * e \mod \varphi(n) = 1$$

Or , $\mathbf{d = e^{-1} \pmod{\varphi(n)}}$

- Public key is pair of values (*e*,*n*).
- Private key is pair values (*d*,*n*)

Example 1

Problem:

prime numbers p=3, q=11, e =7, plaintext T= 31

Find the key pair?

Calculate

$n = pq$ = 3x11 = 33

$\varphi(n) = (p-1)(q-1) = 2\text{x}10 = 20$

Example 1 cont'd

So $n = 33$, $\varphi(n) = 20$, $e = 7$

and d<20
with 7 and 20 the (gcd(20, 7)).

$d * e \mod \varphi(n) = 1$

7*d mod 20 = 1
so d = 3

Encryption and Decryption

Plaintext : $T < n$

Cipher Text: $C = T^{e} \pmod{n}$

Cipher Text : C

Plaintext : $T = C^{d} \pmod{n}$

Example 1 Cont'd

Encrypt:

Plaintext : Let $T = 31 < n$

Cipher Text: $C = T^e \pmod{n} = 31^7 \pmod{33}$

$= 27512614111 \bmod 33$

$= 4$

Example 1 Cont'd

Decrypt:

Cipher Text : $C = 4$

Plaintext *:* $T = C^d \pmod{n} = 4^3 \pmod{33}$

$=64 \bmod 33$

$=31$

So the

Public key (encryption key)= (7, 33)

Private key (decryption key) = (3, 33)

The Problem of Key Exchange

One of the main problems of symmetric key encryption!

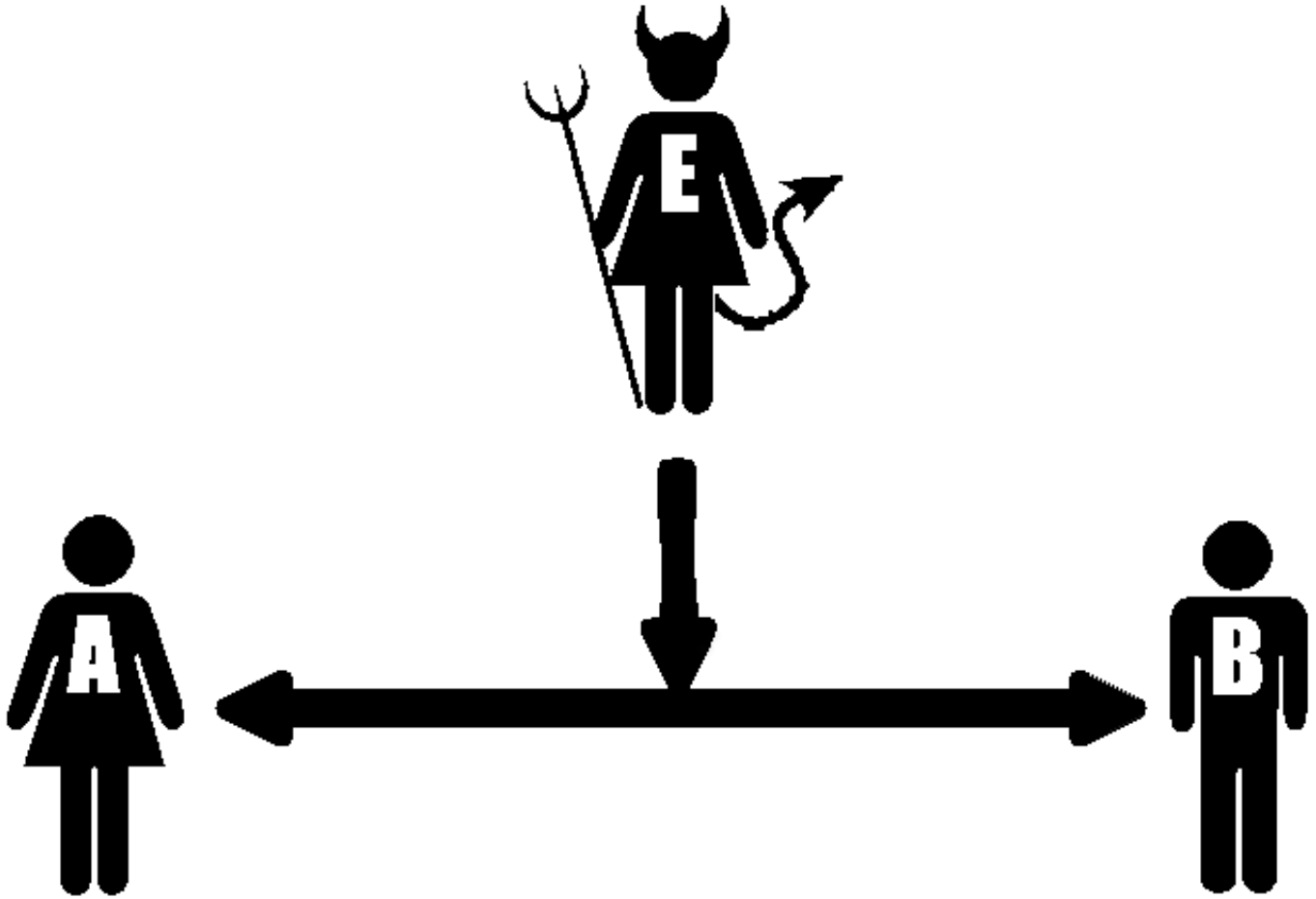
Diffie-Hellman Key Exchange

The protocol offers a solution

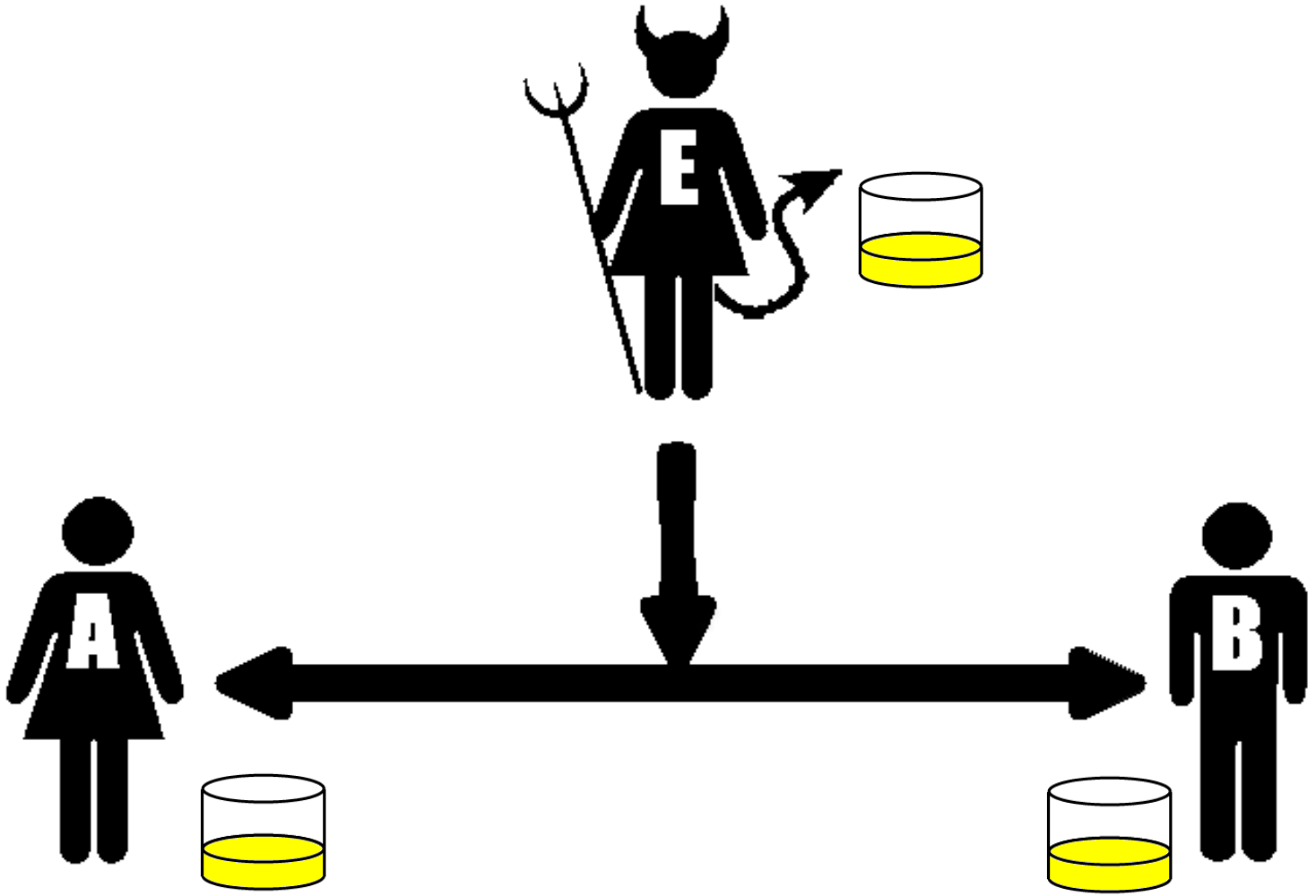
Through a complex mathematics

In practice the shared encryption key relies on such complex concepts as Modular Exponentiation, Primitive Roots and Discrete Logarithm Problems.

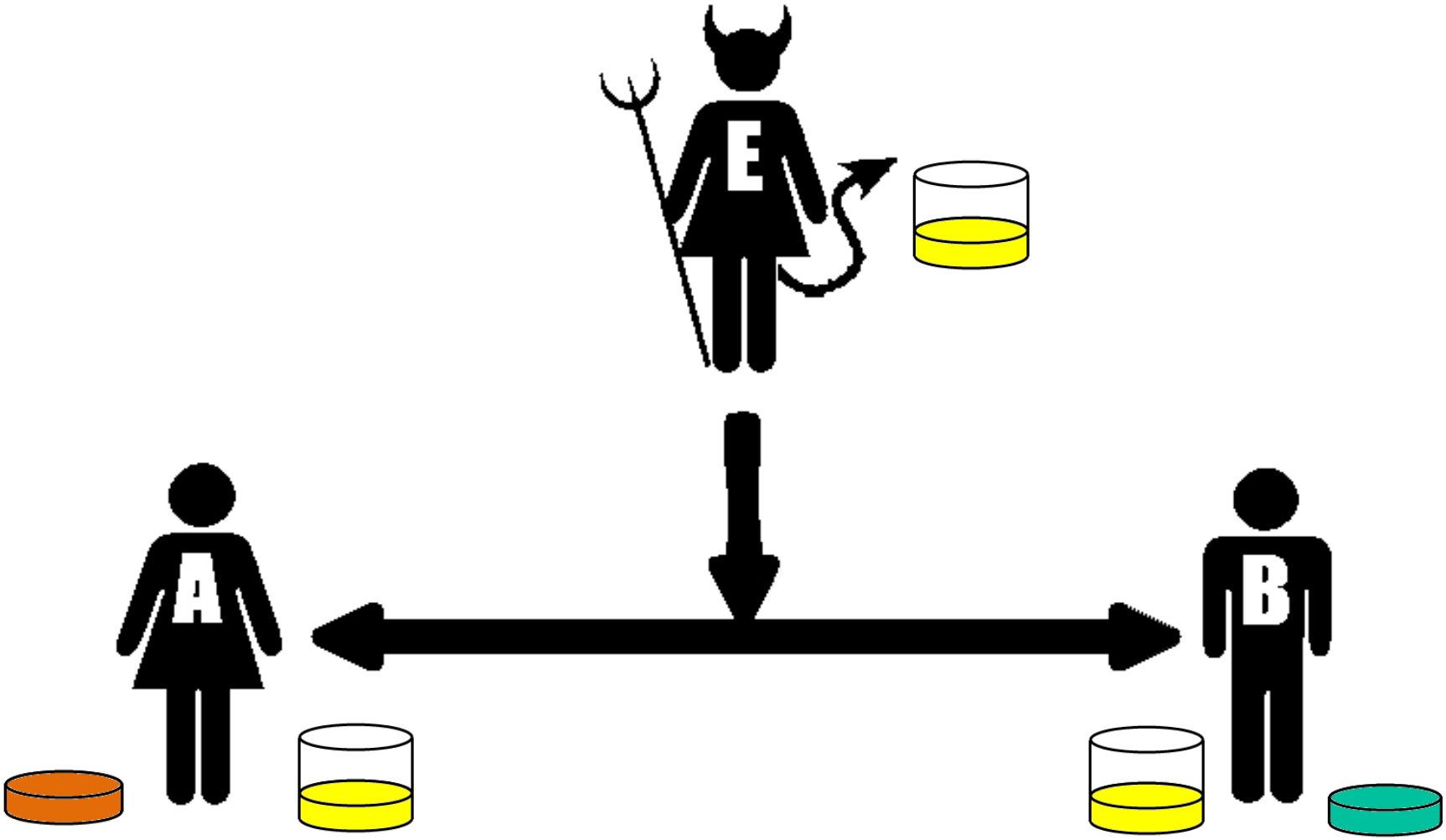
Alice & Bob with Eve listening wish to make a secret shared color



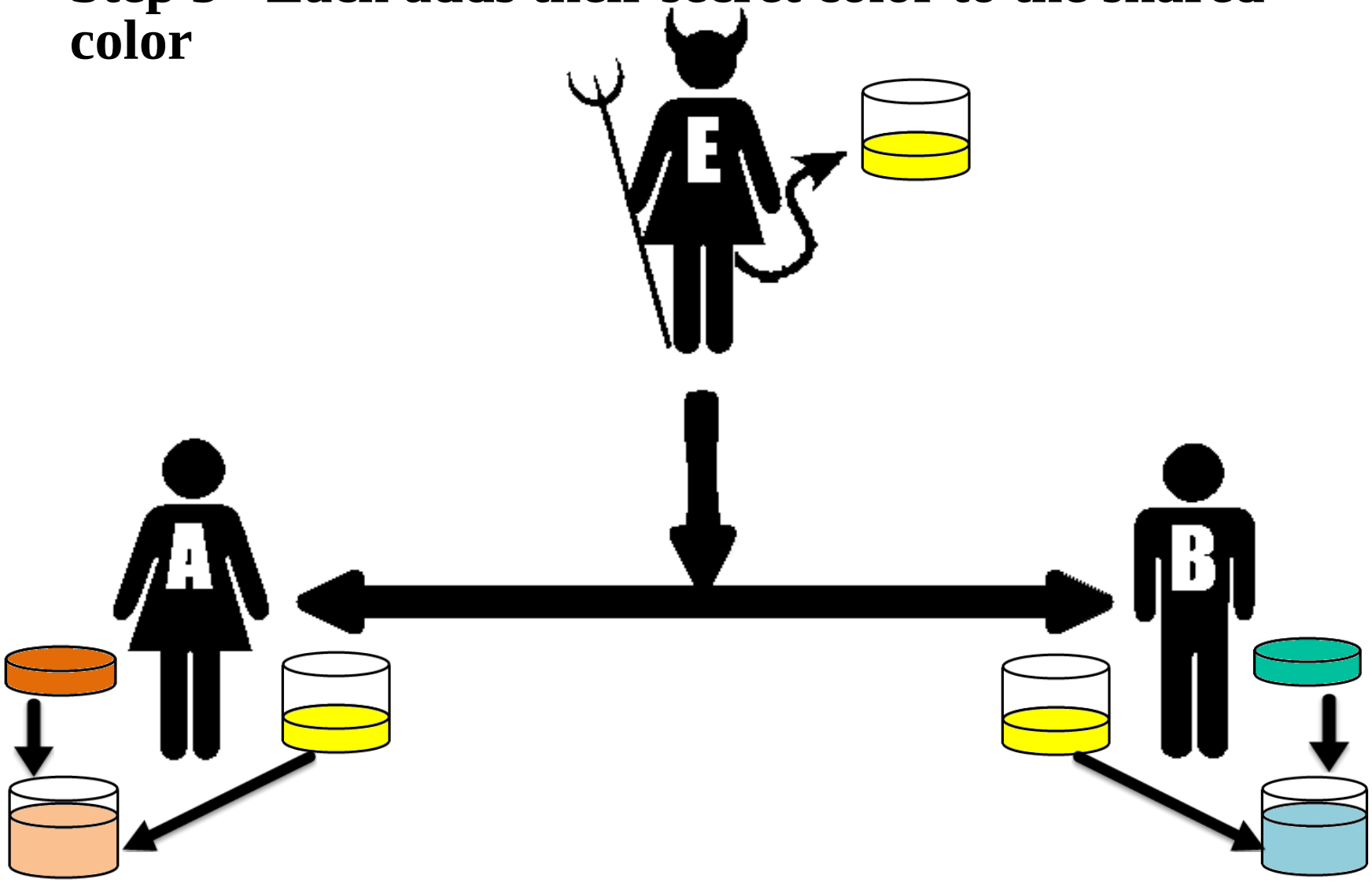
Step 1 - Both publicly agree to a shared color



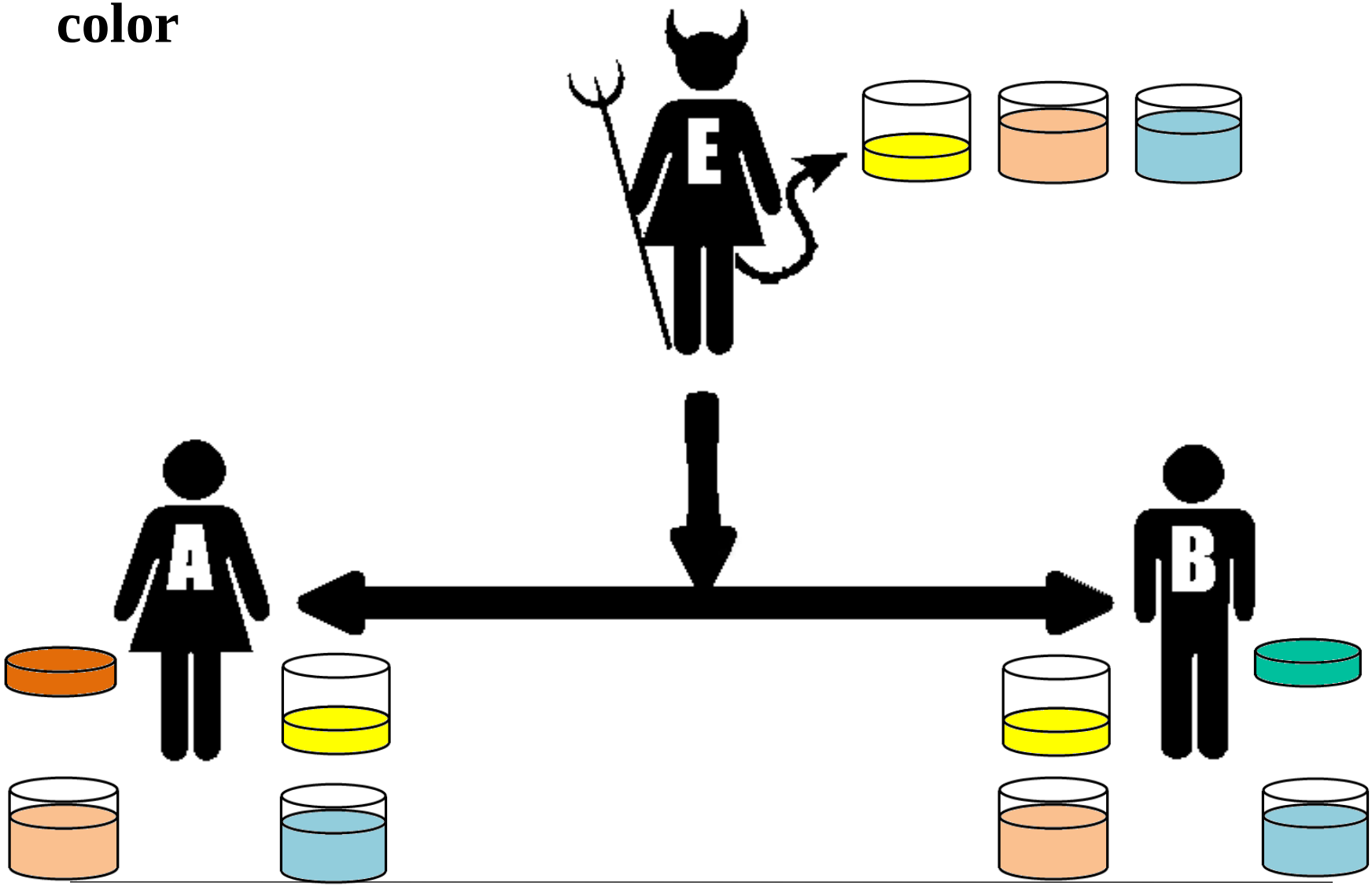
Step 2 - Each picks a secret color



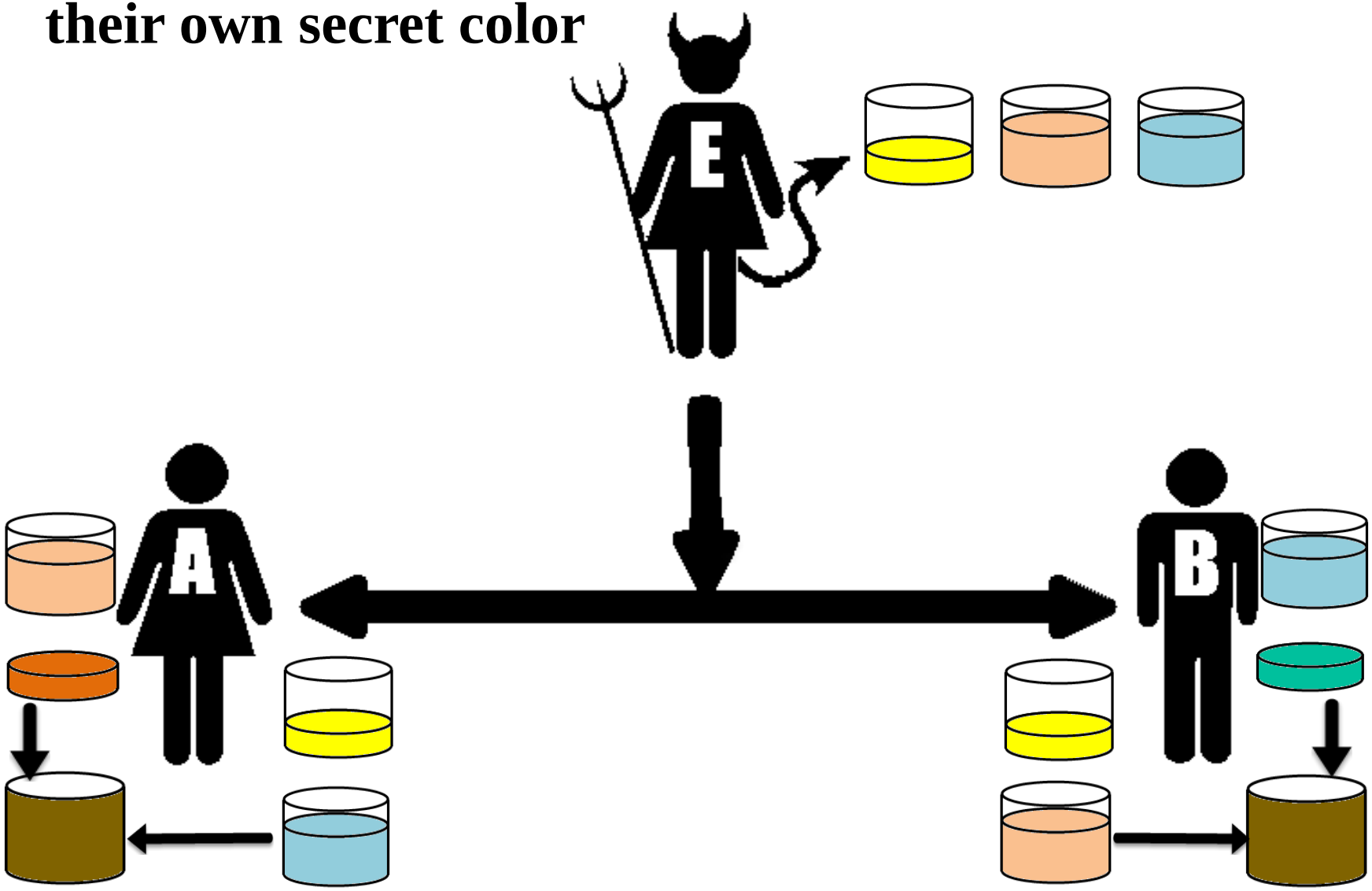
Step 3 - Each adds their secret color to the shared color



Step 4 - Each sends the other their new mixed color



Each combines the shared color from the other with their own secret color



⇒ Consider a prime number 'q'

⇒ Set α such that it must be the primitive root of q and $\alpha < q$

⇒ α is primitive root of q means:

$\alpha^1 \bmod q$

$\alpha^2 \bmod q$

$\alpha^3 \bmod q$

$\vdots$

$\alpha^{q-1} \bmod q$

→ gives result $\{1, 2, 3, \ldots q-1\}$

such as

$3^1 \bmod 7 = 3$

$3^2 \bmod 7 = 2$

$3^3 \bmod 7 = 6$

$3^4 \bmod 7 = 4$

$3^5 \bmod 7 = 5$

$3^6 \bmod 7 = 1$

⇒ Values should not be repeated & we should have all the values in O/p set from 1 to 9-1

such as

$2^1 \bmod 7 = 2$

$2^2 \bmod 7 = 4$

$2^3 \bmod 7 = 1$

$2^4 \bmod 7 = 2$

$2^5 \bmod 7 =$

$2^6 \bmod 7 =$

So 2 can't be primitive root of 7

α and q $\Rightarrow$ these are global (public) elements (known to everyone in the network)

X $\Rightarrow$ Private key of the user (assume)

Y $\Rightarrow$ Public key of the user (assume)

(1) Now Assume X_A (Private key of A user) and $X_A < q$

So find Y_A (Public key of A) $= \alpha^{X_A} \bmod q$

(2) Similarly X_B (Private key of B user) and $X_B < q$

So find Y_B (Public key of B user) $= \alpha^{X_B} \bmod q$

(3) For secure exchang $K_A = K_B$

$K_A = (Y_B)^{X_A} \bmod q$, $K_B = (Y_A)^{X_B} \bmod q$

Question:- Diffi Hellman key exchange for the prime number 7 and primitive root 5 (4)
Private key for user 'A' is 3 and 'B' is 4.
Calculate their respective public keys and check 'did they securely send the secret key'?

Ans:- $q=7$; $\alpha=5$; $X_A=3$; $X_B=4$; $Y_A=?$, $Y_B=?$

K_A equal to K_B ?

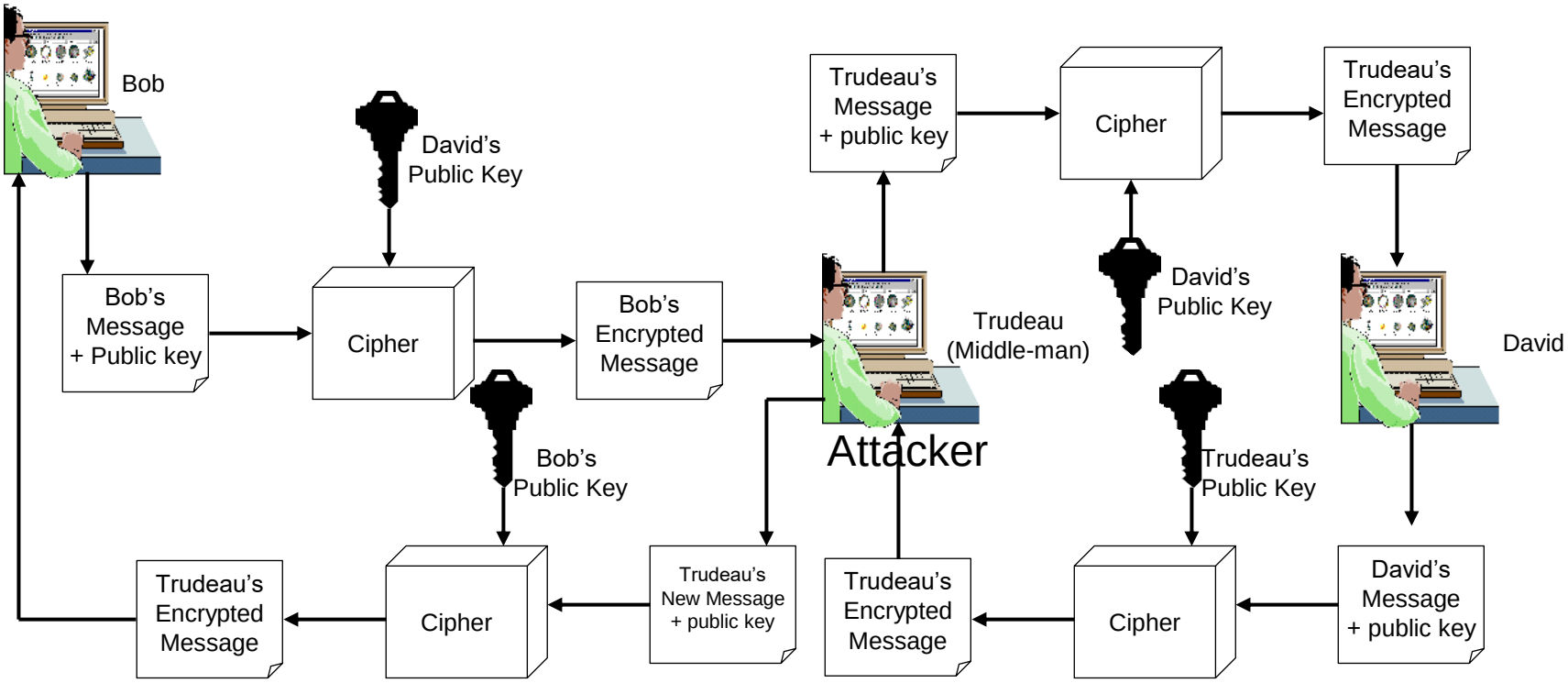
$Y_A \Rightarrow 5^3 \bmod 7 \Rightarrow 125 \bmod 7 \Rightarrow 6$, $Y_A = 6$

$Y_B = 5^4 \bmod 7 \Rightarrow 2$ $Y_B = 2$

$K_A = (2)^3 \bmod 7 \Rightarrow 1$

$K_B = (6)^4 \bmod 7 \Rightarrow 1$

Asymmetric Encryption Man-in-the-middle Attack



Man-in-the-Middle (MITM) Attack Concept

E{a,b,c} = Alice's, Bob's, and Charlie's public keys, respectively

Alice wants to send secure messages to Bob.

Charlie intercepts Alice's messages.

Charlie talks to Alice and pretends to be Bob.

Charlie talks to Bob and pretends to be Alice.

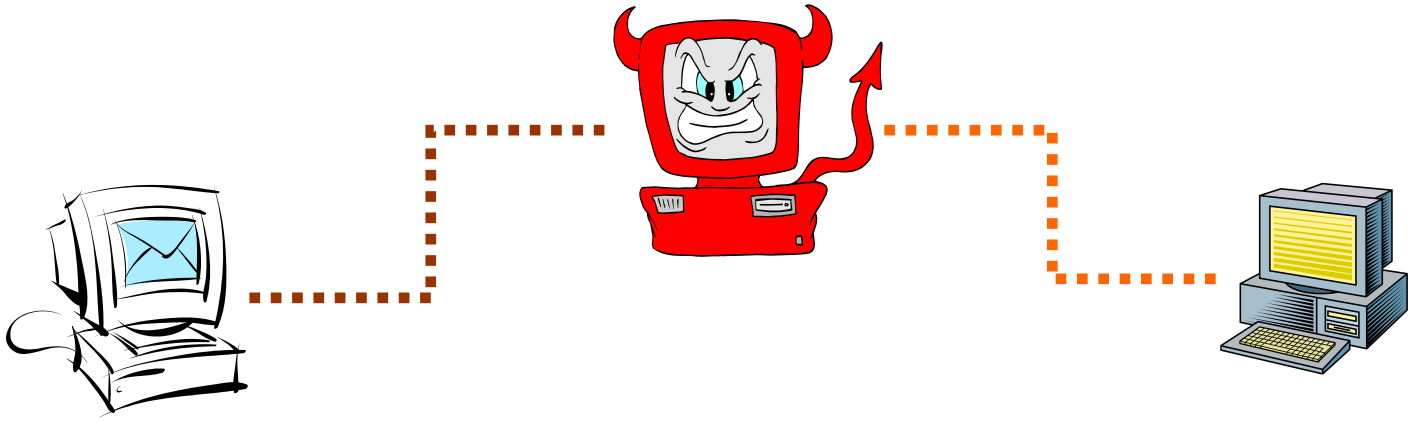
MITM Attack Concept

Alice uses the *public key* she thinks she received from Bob (Charlie's)

Bob uses the key he thinks is Alice's (also Charlie's)

As a result, Charlie not only gains *access* to secure information but also can *modify* it (e.g. *transfer money to a different account* etc.)

Data Integrity and Source Authentication



- Encryption does not protect data from modification by another party.
  - Why?
- Need a way to ensure that data arrives at destination in its original form as sent by the sender and it is coming from an authenticated source.

Authentication Basics

Authentication is the process of validating the identity of a user or the integrity of a piece of data.

technologies that provide authentication

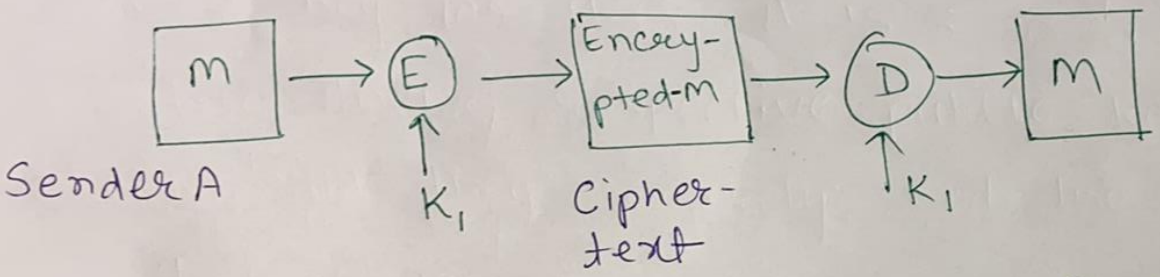
- Message Encryption / Public Key Infrastructure
- Message Authentication Codes and HASH (Message Digest)
- Digital Signatures

There are two types of user authentication:

- Identity presented by a remote or application participating in a session
- Sender's identity is presented along with a message.

① Message encryption: Ciphertext is an authenticator. ①

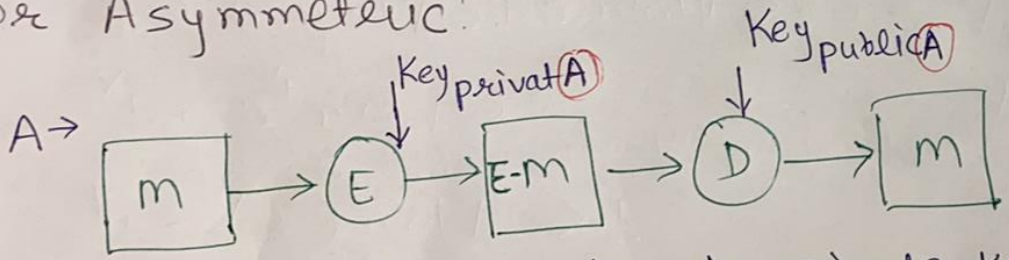
ⓐ For Symmetric:



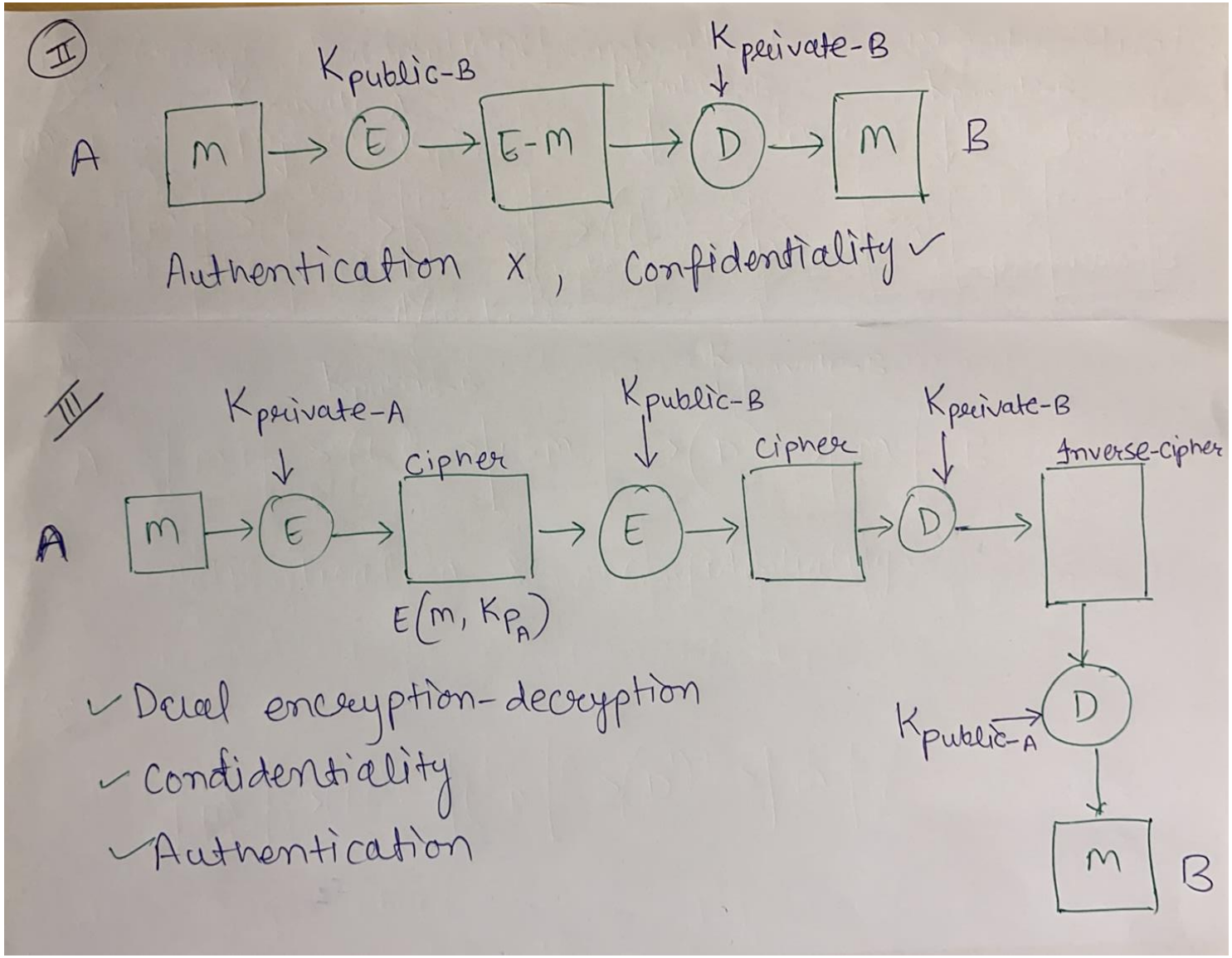
Key K_1 shared betⁿ A and B is same. That's what we did till now.

ⓑ for Asymmetric:

ⓘ



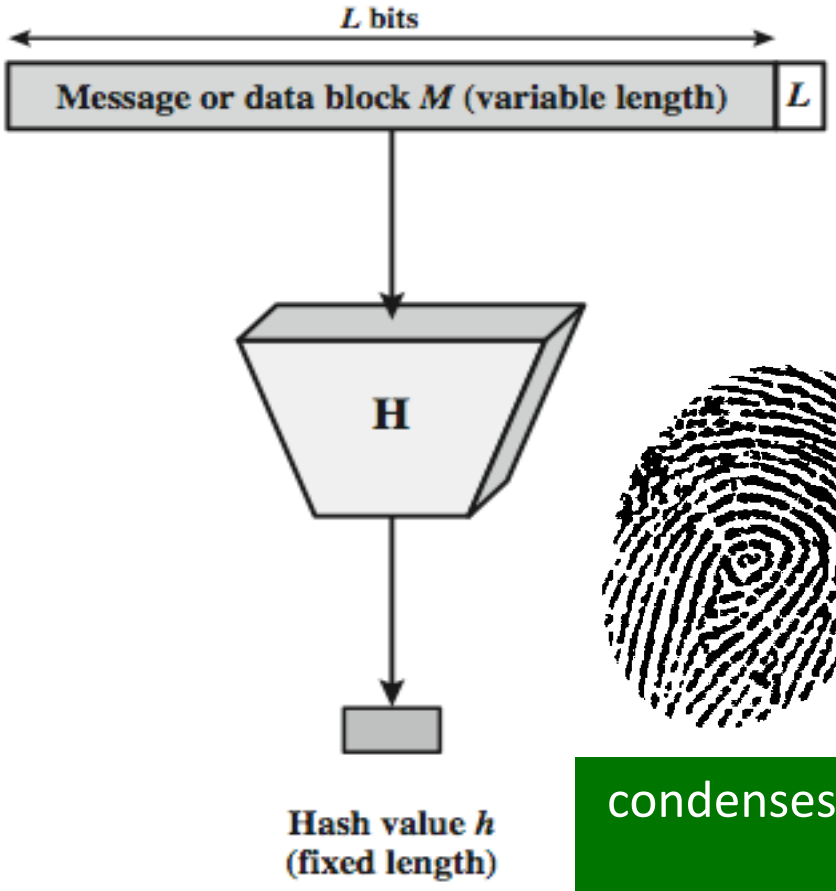
A user encryptes m with A's private key. B so now it reaches to B, B can identify (authenticate) the sender. but as B is using Public Key of A i.e no confidentiality


II
A
M
E
E-M
D
M
B
Authentication X, Confidentiality ✓
III
A
M
E
Cipher
E(M, KpA)
E
Cipher
D
Inverse-cipher
D
M
B
✓ Dual encryption-decryption
✓ Confidentiality
✓ Authentication

Hash Functions

- condenses arbitrary message to fixed size
- h = H(M)
- usually assume hash function is public
- hash used to detect changes to message

Hash Function



The hash value represents concisely the longer message

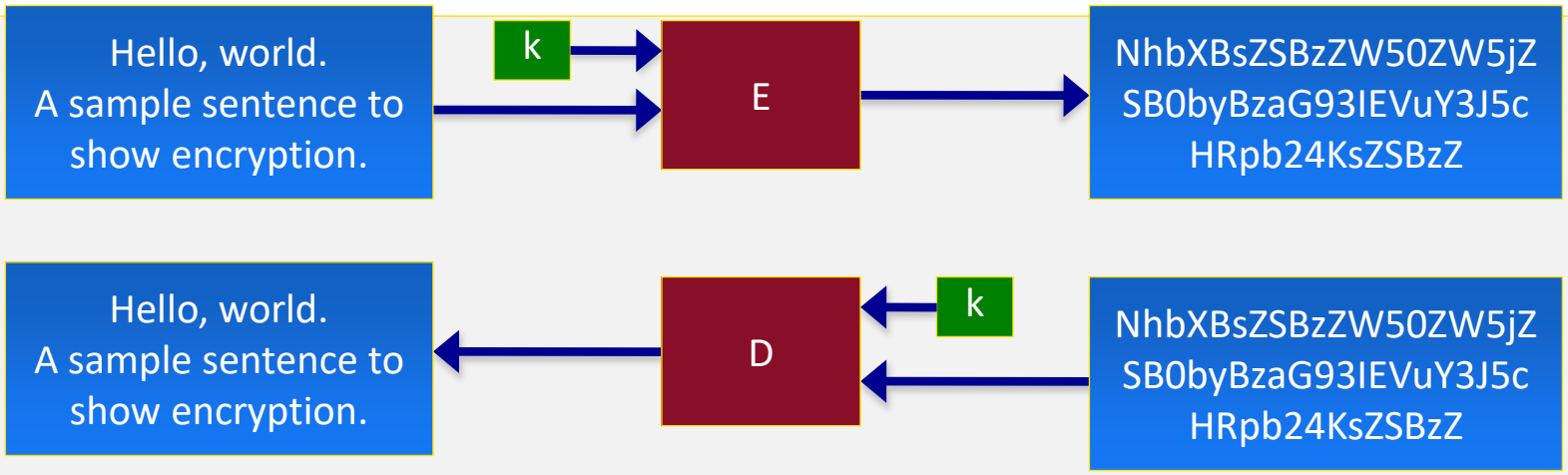
- may called the *message digest*

A message digest is as a ``digital fingerprint'' of the original document

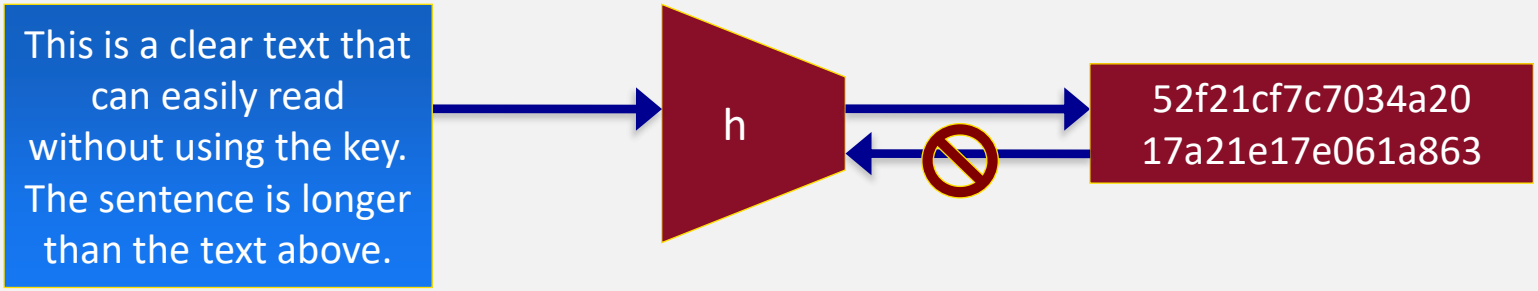
condenses arbitrary message to fixed size

$h = H(M)$

Hashing V.S. Encryption



- Encryption is two way, and requires a key to encrypt/decrypt



– Hashing is one-way. There is no 'de-hashing'

Motivation for Hash Algorithms

Intuition

- Limitation on non-cryptographic checksum
- Very possible to construct a message that matches the checksum

Goal

- Design a code where the original message can not be inferred based on its checksum
- such that an accidental or intentional change to the message will change the hash value

Hash Function Properties

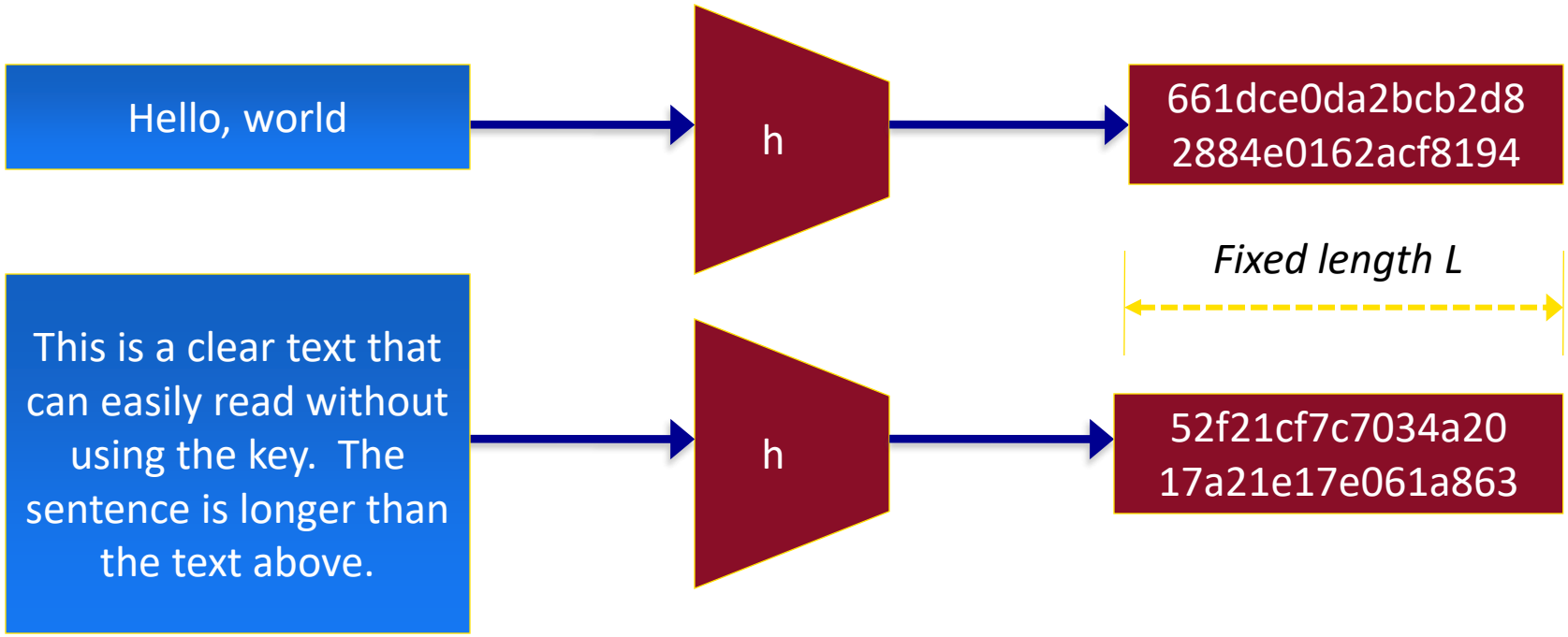
Arbitrary-length message to fixed-length digest

Preimage resistant (**One-way property**)

Second preimage resistant (**Weak collision resistant**)

Collision resistant (**Strong collision resistance**)

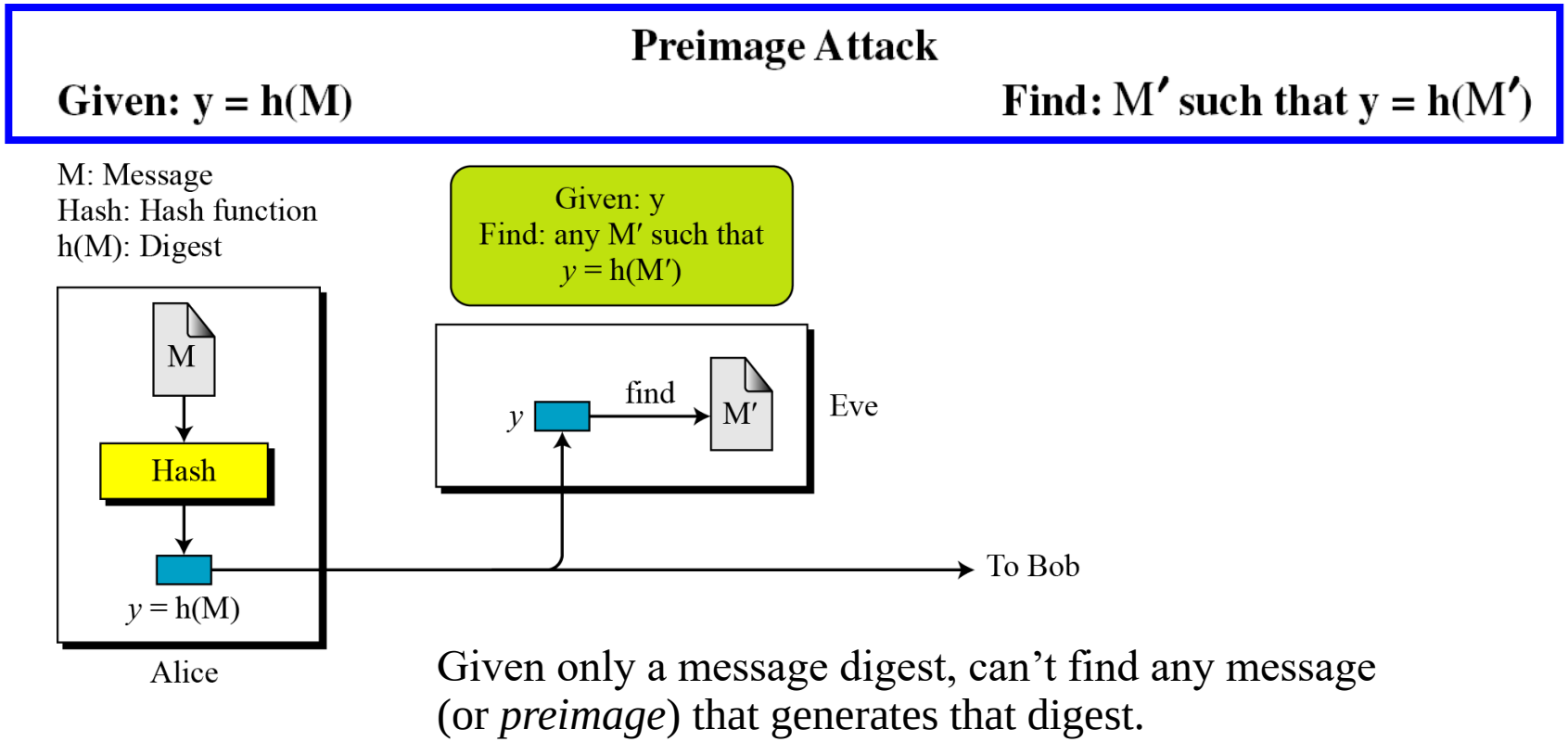
Properties : Fixed length



Arbitrary-length message to fixed-length digest

Preimage resistant

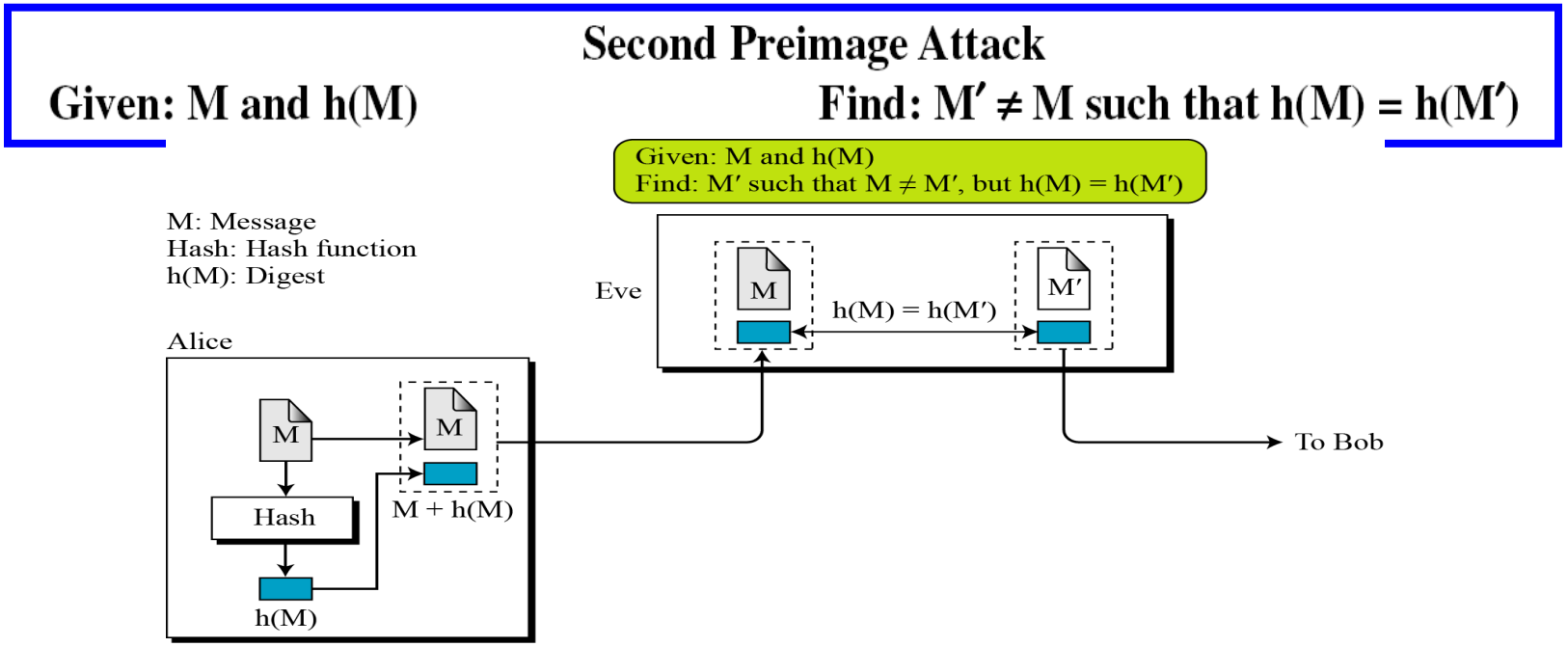
This measures how difficult to devise a message which hashes to the known digest
Roughly speaking, the hash function must be one-way.



Given only a message digest, can't find any message (or *preimage*) that generates that digest.

Second preimage resistant

This measures how difficult to devise a message which hashes to the known digest and its message



Given one message, can't find another message that has the same message digest.

An attack that finds a second message with the same message digest is a *second pre-image* attack.

- It would be easy to forge new digital signatures from old signatures if the hash function used weren't second preimage resistant

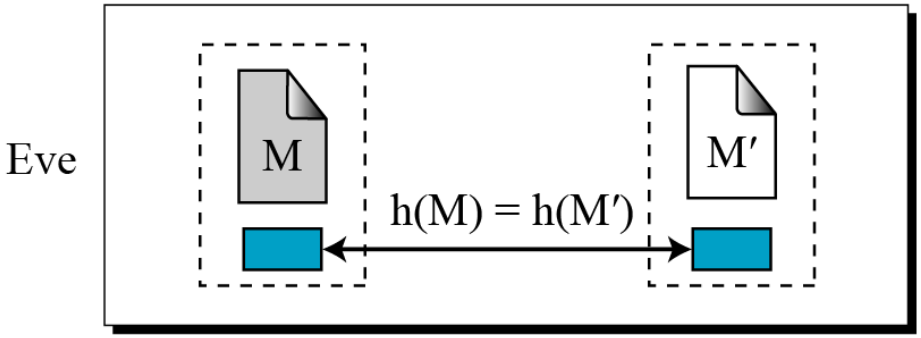
Collision Resistant

Collision Attack

Given: none **Find: $M' \neq M$ such that $h(M) = h(M')$**

M: Message
Hash: Hash function
h(M): Digest

Find: M and M′ such that $M \neq M'$, but $h(M) = h(M')$



- Can't find any two different messages with the same message digest
  - Collision resistance implies second preimage resistance
  - Collisions, if we could find them, would give signatories a way to repudiate their signatures

Hash Function are applied in

- Message Integrity Check (MIC)
  - send hash of message (digest)
  - MIC always encrypted, message optionally

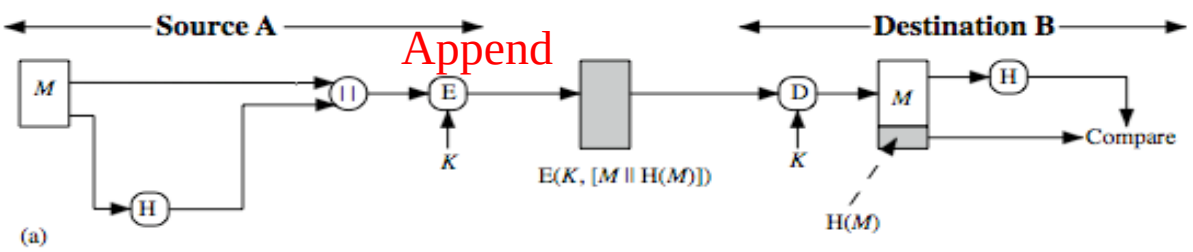
- Message Authentication Code (MAC)
  - send keyed hash of message
  - MAC, message optionally encrypted

- Digital Signature (non-repudiation)
  - Encrypt hash with private (signing) key
  - Verify with public (verification) key

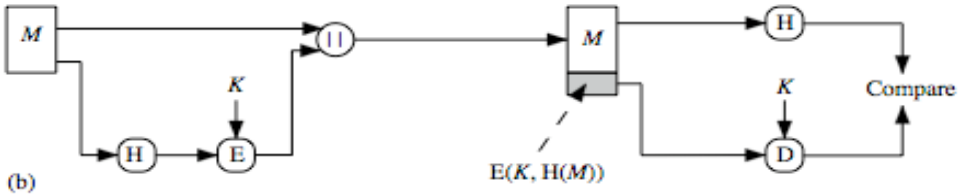
Hash Functions & Message Authentication

Symmetric Key
Unkeyed Hash

a) Message encrypted



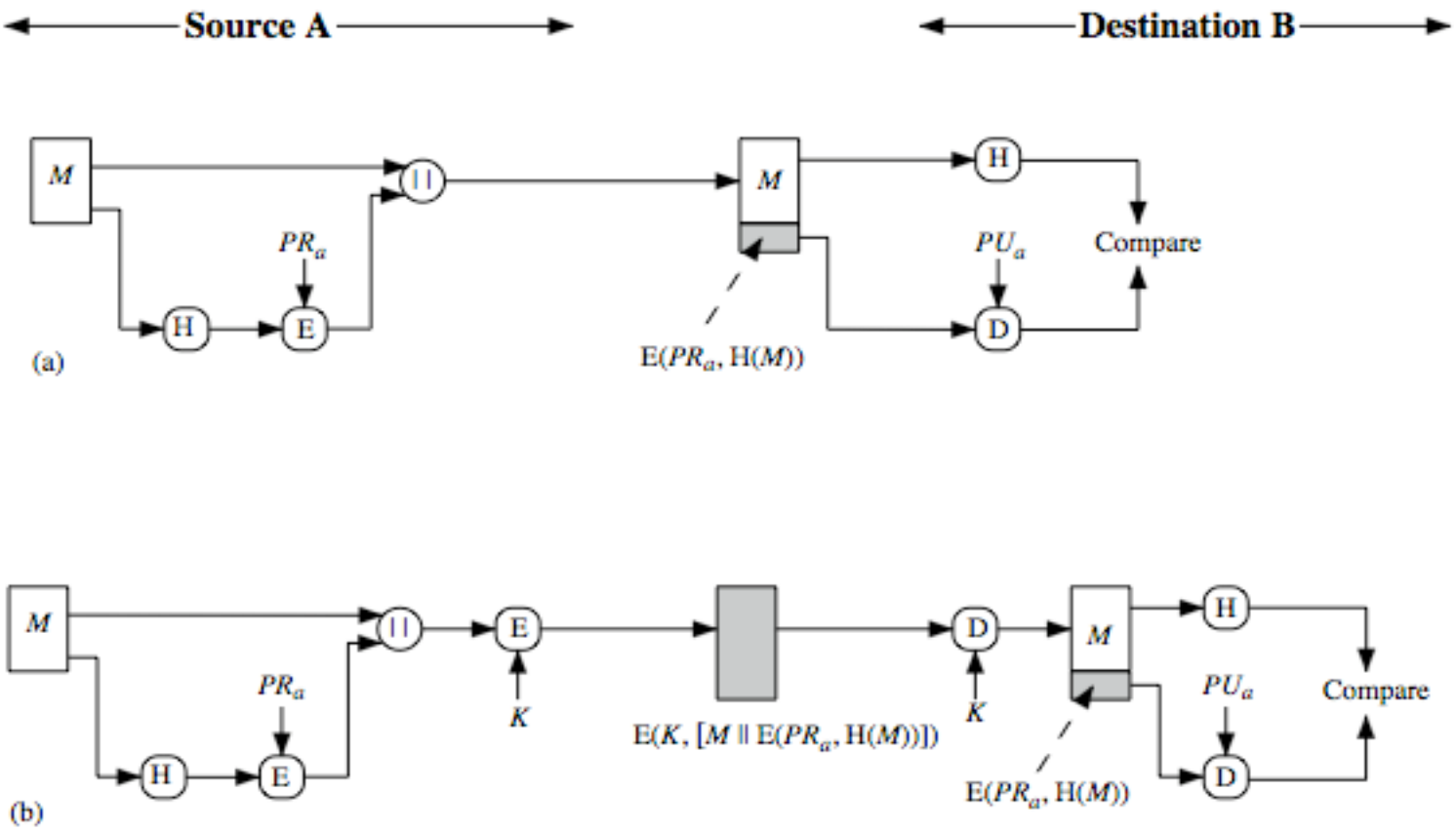
b) Message unencrypted



a) If both hash codes equal in the end, then authentication is performed, as only A and B share the secret key and msg must have come from A i.e has not been altered. Confidentiality is also there as msg was encrypted before sending

b) Only hash code is encrypted not the whole msg (it takes less time) and B can identify the sender. So only authentication, but no confidentiality as any one can read our Msg.

Hash Functions & Digital Signatures - PKPK



Message Authentication Code

Uses a shared secret key to generate (known as a cryptographic checksum or MAC) that is appended to the message

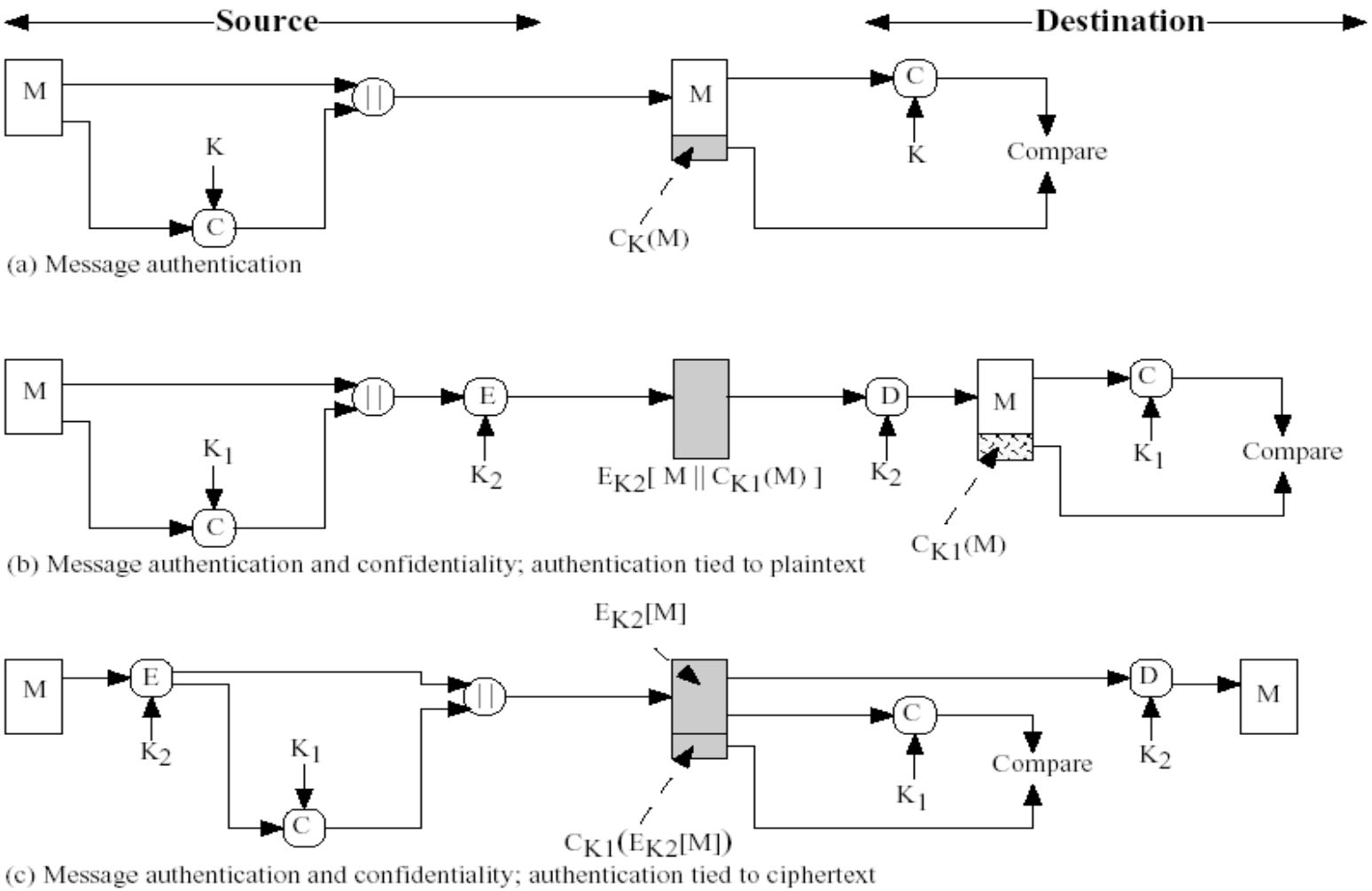
MAC = $C_K(M)$

It generate the small fixed size block of data from arbitrary length of input msg

Assurances:

- Message has not been altered
- Message is from alleged sender
- Message sequence is unaltered (requires internal sequencing)

Basic Uses of MAC



Basic Uses of MAC

(a) $A \rightarrow B: M \| C_K(M)$

- Provides authentication
  - —Only A and B share K

(b) $A \rightarrow B: E_{K_2}\left[M \| C_{K_1}(M)\right]$

- Provides authentication
  - —Only A and B share K_1
- Provides confidentiality
  - —Only A and B share K_2

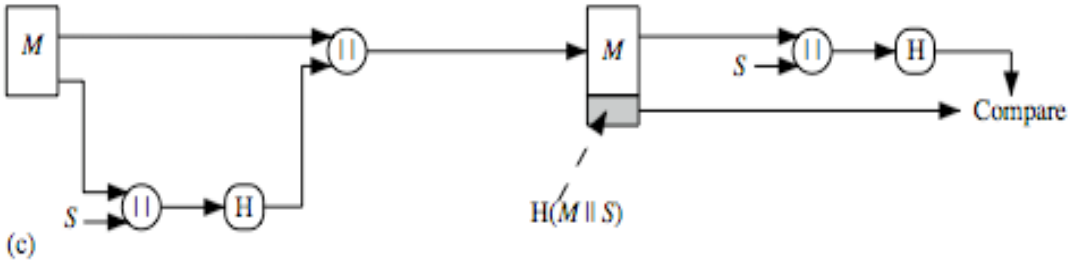
(c) $A \rightarrow B: E_{K_2}[M] \| C_{K_1}\left(E_{K_2}[M]\right)$

- Provides authentication
  - —Using K_1
- Provides confidentiality
  - —Using K_2

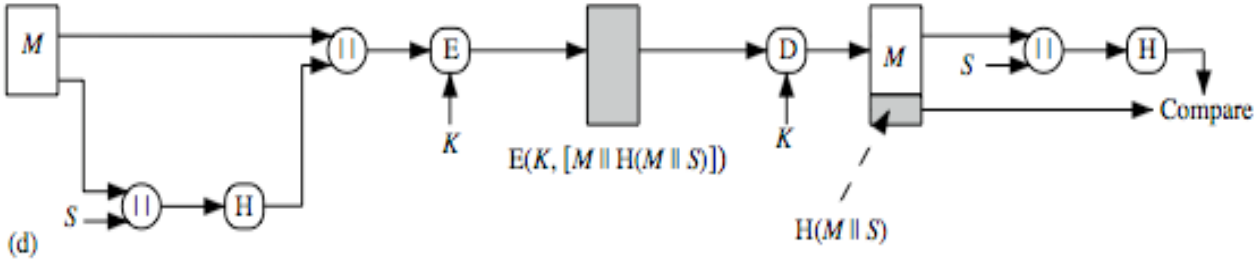
Keyed Hash Functions based on MAC

Symmetric Key
Keyed Hash

c) Message unencrypted



d) Message encrypted



Why/where Use MACs?

Digital Signatures

have looked at message authentication

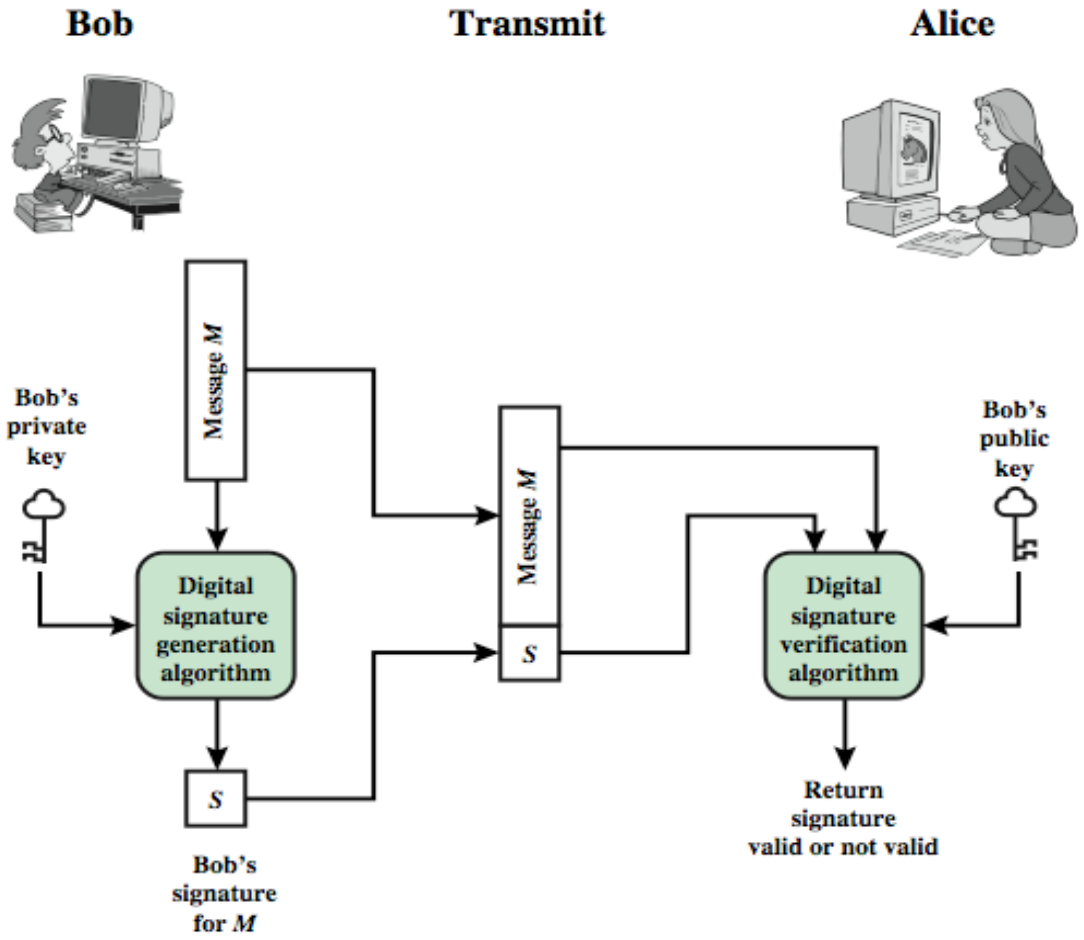
- but does not address issues of lack of trust

digital signatures provide the ability to:

- verify author, date & time of signature
- authenticate message contents
- be verified by third parties to resolve disputes

hence include authentication function with additional capabilities

Digital Signature Model



Definition

Birthday attacks are a class of brute-force techniques that target the cryptographic hash functions. The goal is to take a cryptographic hash function and find two different inputs that produce the same output.

Birthday Attacks

The probability that all 23 people have different birthdays is

$$1\times(1-\frac{1}{365})(1-\frac{2}{365})\ldots(1-\frac{22}{365})=0.493$$

Therefore, the probability of at least two having the same birthday is 1- 0.493=0.507

More generally, suppose we have N objects, where N is large. There are r people, and each chooses an object. Then

$$P(\text{there is a match}) \approx 1-e^{-r^2/2N}$$

Birthday Attacks

(Example) We have 40 license plates, each ending in a 3-digit number. What is the probability that two of the license plates end in the same 3 digits?

(Solution) N=1000, r=40

1. Approximation: $1-(1-\frac{1}{1000})(1-\frac{2}{1000})...(1-\frac{39}{1000})=0.546$

2. The exact answer:

$$1-e^{-40^2/2\cdot 1000}=0.551$$

Attack Prevention

The important property is the length in bits of the message digest produced by the hash function.

If the number of *m* bit hash , the cardinality *n* of the hash function is

$$n = 2^m$$

The 0.5 probability of collision for m bit hash, expected number of operation *k* before finding a collision is very close to

$$k \approx \sqrt{n} = 2^{m/2}$$

m should be large enough so that it's not feasible to compute hash values!!!

References

1. http://www.sfu.ca/~ljilja/ENSC427/News/Kurose_Ross/Chapter_8_V7.0_Accessible.pdf
2. Stallings, W., Cryptography and Network Security: Principles and Practice 0133354695, 9780133354690.
3. **A.K. Dewdney, The New Turning Omnibus, pp. 250-257, Henry Holt and Company, 2001.**

Lnu.se