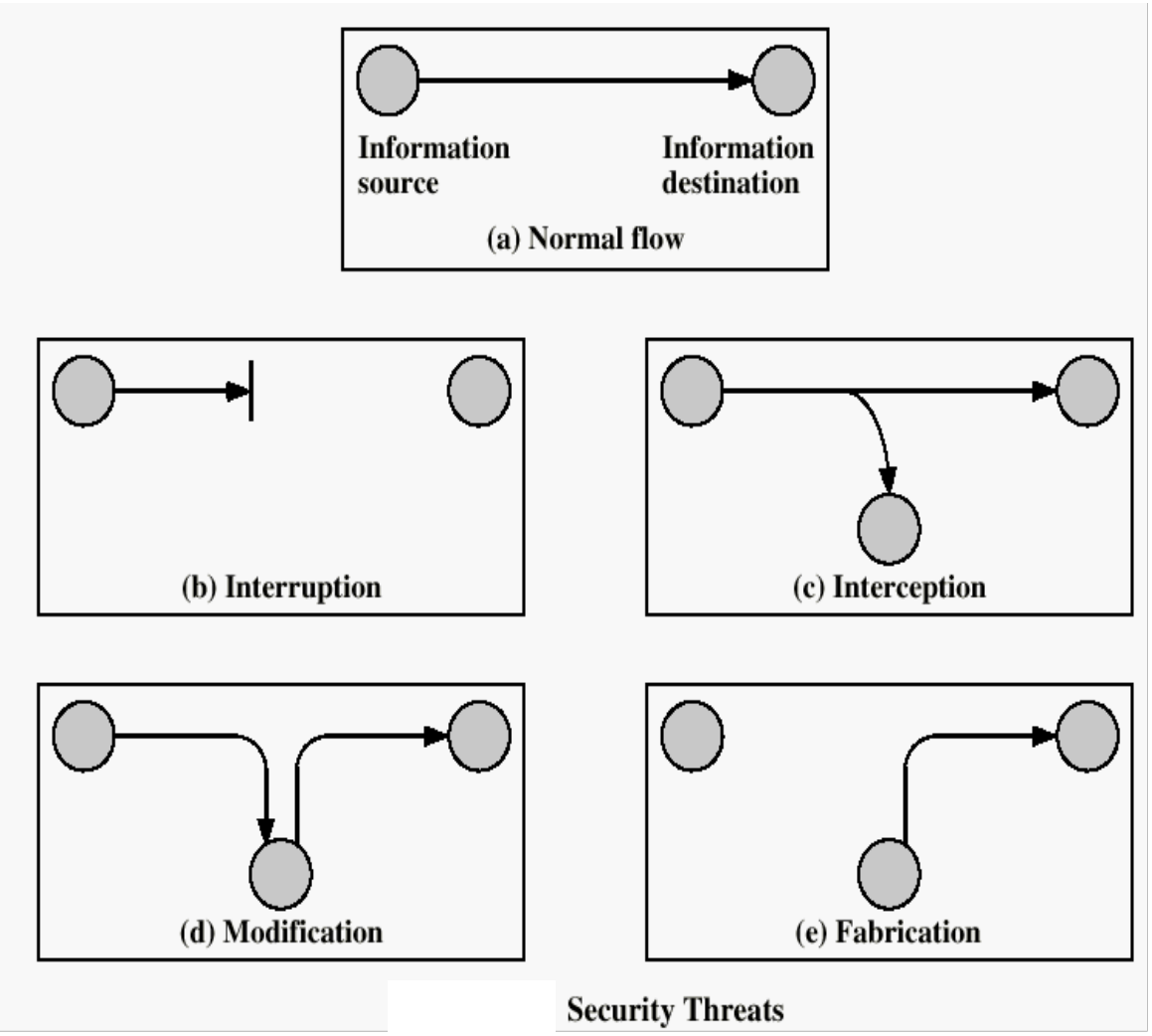
A Whirlwind Tour of Basic Cryptographic Tools and their Applications

What kind of Information Security Threats exist ?



Security Threats

Broad Classification:

Leakage

Tampering

Vandalism

Different Goals/ Services provided by Security

- **Confidentiality (for your eyes only): Against Eavesdropping, Sniffing, Tracing**
- **Integrity (has not been altered) Against Tampering**
- **Authentication (you are who you say you are) Against Impersonation, Masquerading, Spoofing**
- **Access control (only the intended can "use" the resources) Against unauthorized use/ abuse of resources**
- **Non-repudiation (the order is final) Against Denying One's Act, backing away from a deal**
- **Availability Against DoS Attacks**

Cryptography

More Terminology

- Encryption: converting plaintext to ciphertext
- Decryption: converting ciphertext to plaintext
- Cryptanalysis: to break the code by analyzing the the algorithm/system
- Brute-force attack: Enumerate over all the possible keys
- Types of attacks on Encrypted Messages by Cryptanalyst:
  - Ciphertext only: Given Ciphertext only to derive Plaintext/key
  - Known Plaintext: Given <Plaintext,Ciphertext> pair(s) to derive the key, e.g. “From:” at the beginning of each email
  - Chosen Plaintext: The attacker has the ability to inject chosen plaintext and observe the ciphertext outcome, e.g. send an email of chosen words to the victim while sniffing at the cipher

Increasing knowledge/level of control by the Cryptanalyst

A Classical Encryption Scheme: One-time Pad

```
XMCKLA EJKAWO FECIFE WSNZIP PXPKIY URMZHI JZTLBC YLGDYJ HTSVTV
RRYYEG EXNCGA GGQVRF FHZCIB EWLGGR BZXQDQ DGGIAK YHJYEQ TDLCQT
HZBSIZ IRZDYS RBYJFZ AIRCWI UCVXTW YKPQMK CKHVEX VXYVCS WOGAAZ
OUVVON GCNEVR LMBLYB SBDCDC PCGVJX QXAUIP PXZQIJ JIUWYH COVWMJ
UZOJHL DWHPER UBSRUJ HGAAPR CRWVHI FRNTQW AJVWRT ACAKRD OZKIIB
VIQGBK IJCWHF GTTSSE EXFIPJ KICASQ IOUQTP ZSGXGH YTYCTI BAZSTN
```

Encryption:

```
    H        E        L        L        O   plain
  7 (H)    4 (E)   11 (L)   11 (L)   14 (O) plain
+ 23 (X)  12 (M)    2 (C)   10 (K)   11 (L) key
= 30       16       13       21       25     plain + key
=  4 (E)  16 (Q)   13 (N)   21 (V)   25 (Z)(plain + key) mod 26
     E        Q        N        V        Z   → cipher
```

Decryption: **Plain = (cipher – key) mod 26**

- **Achieve "Perfect Secrecy": i.e. the Ciphertext alone does NOT tell you Any Information about the Plaintext**
  - **Can't be cracked even with Infinite Computational Power:**

    EQVNZ -> LATER with another key = TQURI

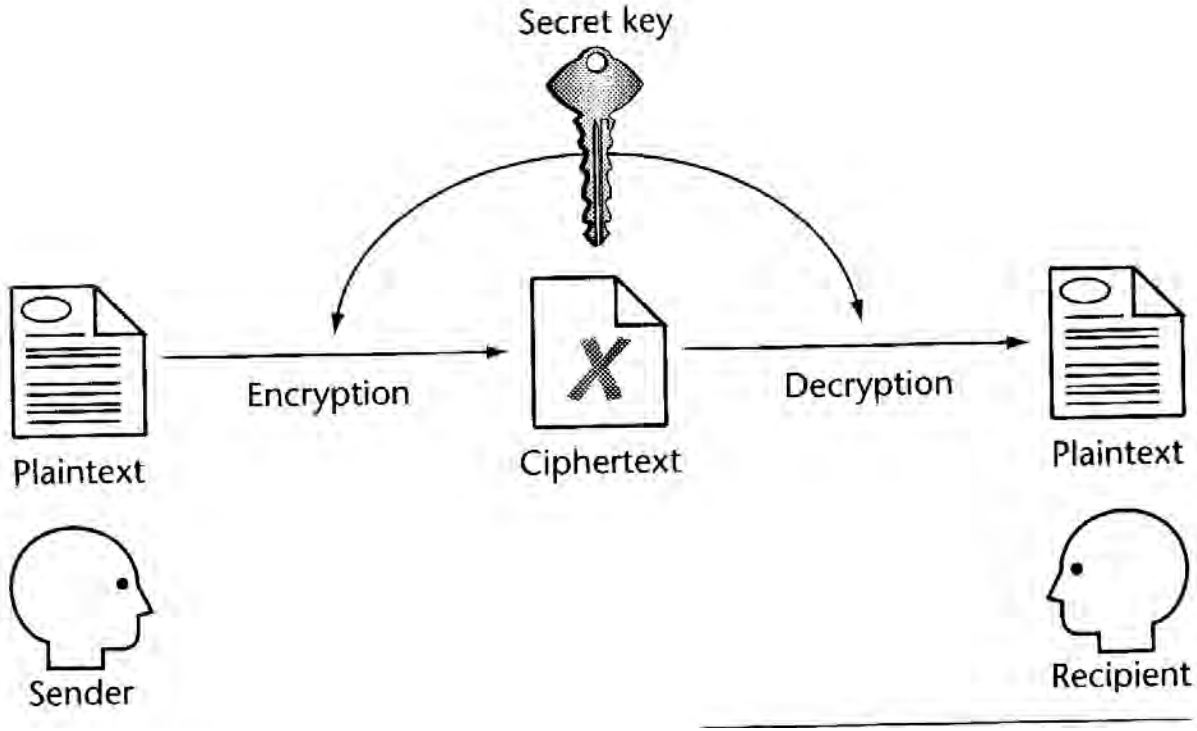
Classification of Cryptosystems and Terminology

- Secret-Key
- Conventional
- Classical
- Symmetric
- Symmetric-key

- Public-Key
- Asymmetric

Symmetric (aka Secret-Key) Cryptosystems

A Secret-key (aka Symmetric Key) Cryptosystem



- The Same key is used for encryption as well as decryption ; That's why it is also also "symmetric key" system
- The encryption/decryption algorithm is sometimes referred as the "cipher"

Block Cipher vs. Stream Cipher

- Process the message block by block of constant size, e.g. 64 bits
- Each block goes through multiple rounds of permutation and substitution
- Mixed operators, data or key dependent rotation/shifting =>permutation
- Key dependent substitution (S-boxes) =>substitution
- More complex key scheduling: part of the key is used to generate the "per-round key" for each round

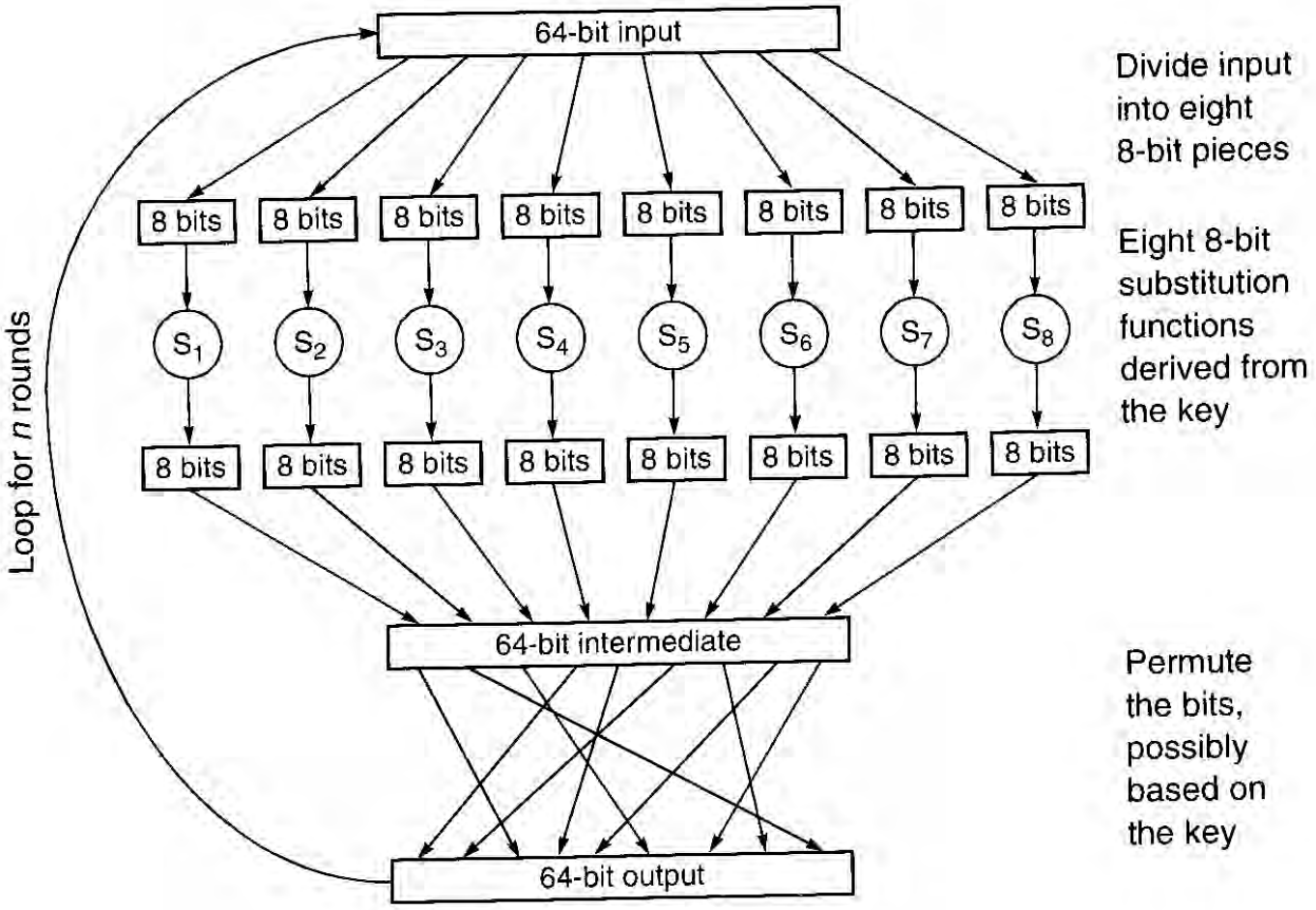
- Process the message bit by bit (as a stream) or byte-by-byte
- Typically have a (pseudo) random **stream key**
- combined ("⊕" XOR) with plaintext bit by bit
- randomness of **stream key** completely destroys any statistically properties in the message
  - $C_i = M_i \oplus S_i$

  Ci = i-th bit of ciphertext

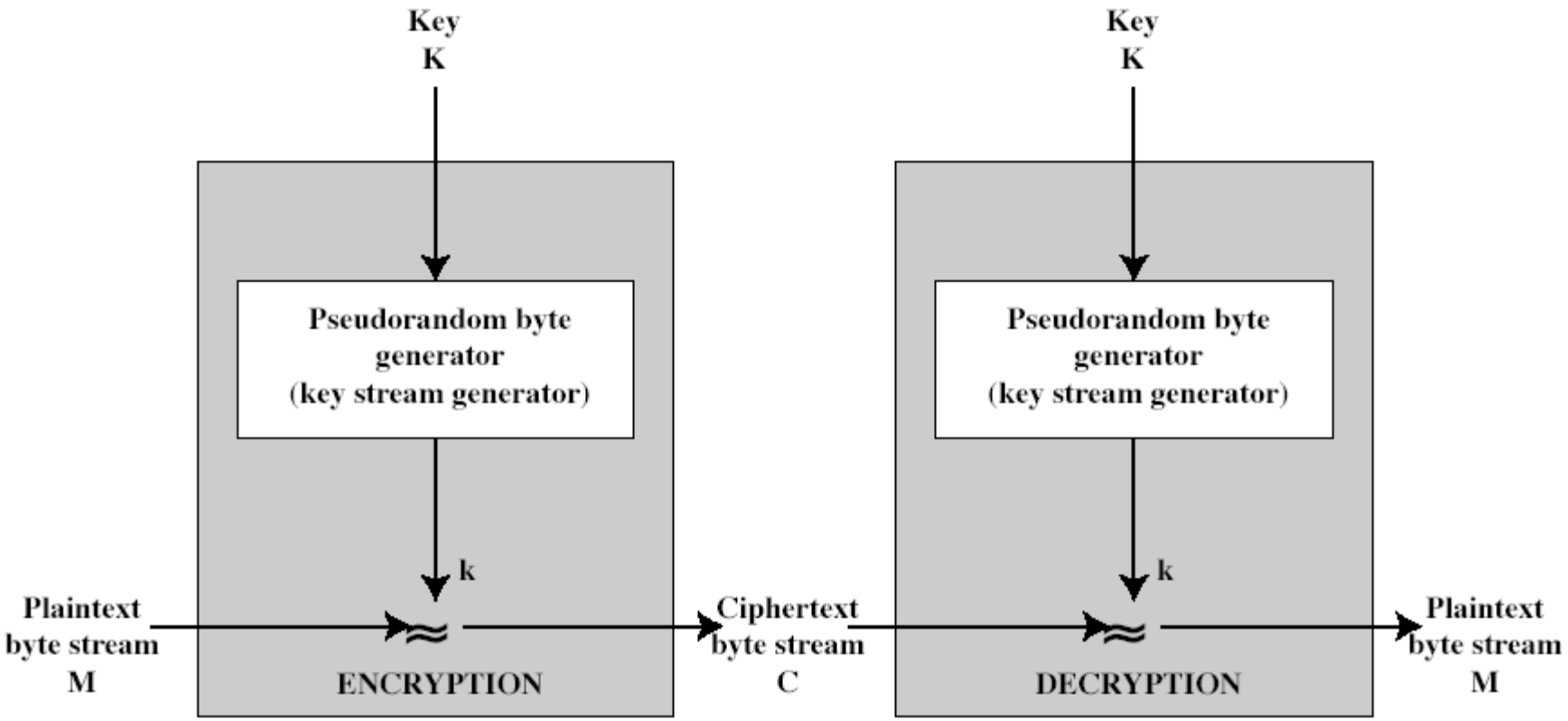
  Mi = i-th bit of plaintext

  Si = i-th bit of stream key
- what could be simpler, faster !!!!
- **but must never reuse stream key**
  - **otherwise can remove effect and recover messages**

General Example of a Block Cipher



Stream Cipher



The ⊕"XOR" function is its own Inverse

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

A	A	A⊕A
0	0	0
1	1	0

A	0	A⊕0
0	0	0
1	0	1

Given: Ci = Mi ⊕ Si ;

Ci ⊕ Si = (Mi ⊕ Si) ⊕ Si = Mi ⊕ (Si ⊕ Si) = Mi ⊕ **0** = Mi

because

A ⊕ **A = 0** for all A and

A ⊕ **0 = A** for all **A**

Minimum Key-length requirement

- Recommended by an NAS/NRC expert panel in 1996:

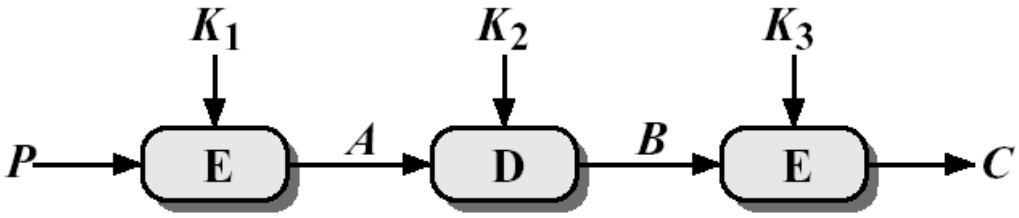
Minimum key length for symmetric-key ciphers					
Intruder	**Budget**	**Tools**	**Time**		**Secure key length**
			40 bits	**56 bits**	
Hacker	tiny	**PC**	**1 week**	infeasible	45
Small business	$400	FPGA	5 hrs	38 years	50
	$10,000	FPGA	12 min	18 months	55
Corporate department	**$300 K**	**FPGA**	**24 sec**	19 days	60
		ASIC	18 sec	**3 hrs**	
Big company	$10 M	FPGA	7 sec	13 hrs	70
		ASIC	5 ms	6 min	
Intelligence agency	$300 M	ASIC	0.2 ms	12 sec	75

- What should be the corresponding Secure key lengths in 2020 ?

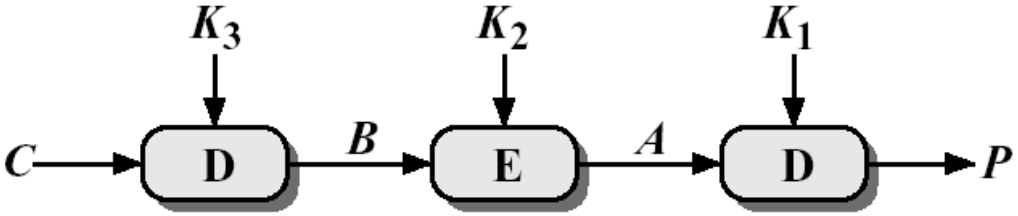
What should we do then ?

- Increase the effective key-length of DES by doing multiple DES with different keys => also mean slow down the encryption for multiple times

=> Here comes the “Triple DES” or 3DES *where* $C = E_{K_3}[D_{K_2}[E_{K_1}[P]]]$



(a) Encryption

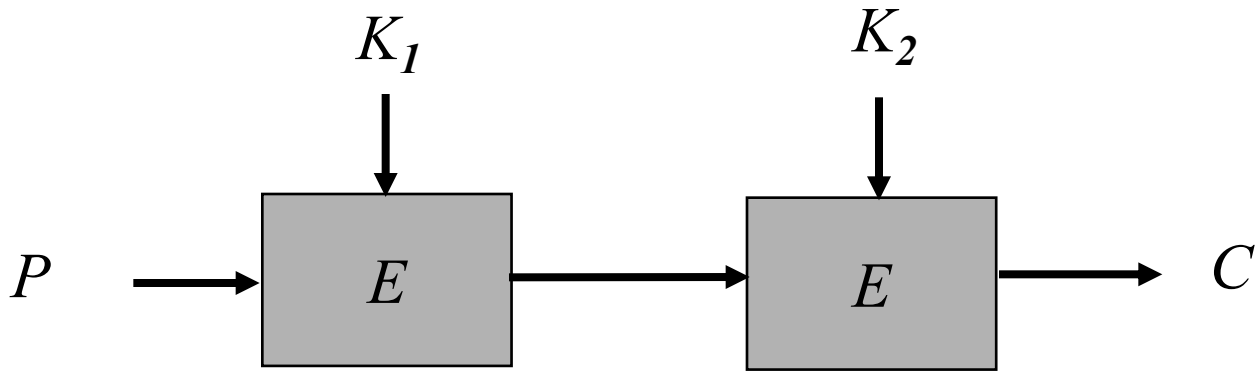


(b) Decryption

- The actual standard requires K3 to be equal to K1 => 112-bit key-length which is deemed to be sufficiently secure
- The use of E-E-E would still have worked. But using E-D-E instead of E-E-E enhance backward compatibility with DES by setting K1 equal to K2

Why not 2DES ?

- If a 112-bit key is already deemed to be secure enough, why not just doing DES encryption twice with 2 different keys, i.e. $C = E_{K_2}[E_{K_1}[P]]$?



- Because this is susceptible to a "Meet-in-the-Middle" attack which reduces the effective key-length to about 57-bit only.

Meet-in-the-Middle Attack on 2DES

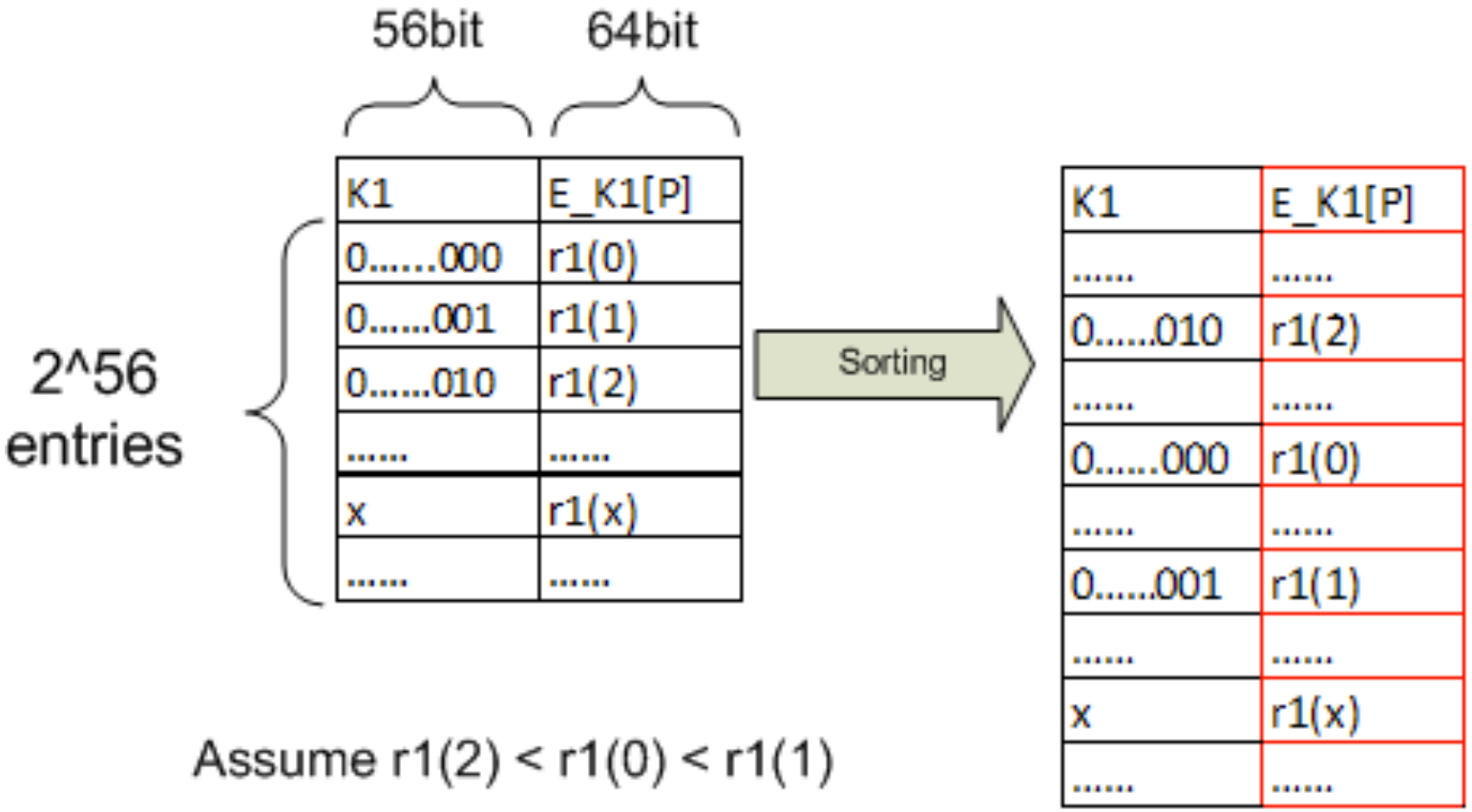
Assume the hacker has a few <plaintext,ciphertext> pairs, e.g. <*p1,c1*>, <*p2,c2*>, <*p3,c3*> where $ci = E_{K_2}[E_{K_1}[pi]]$

Key Observation: $D_{K_2}[c1] = E_{K_1}[p1]$ if K1 and K2 are the right keys.

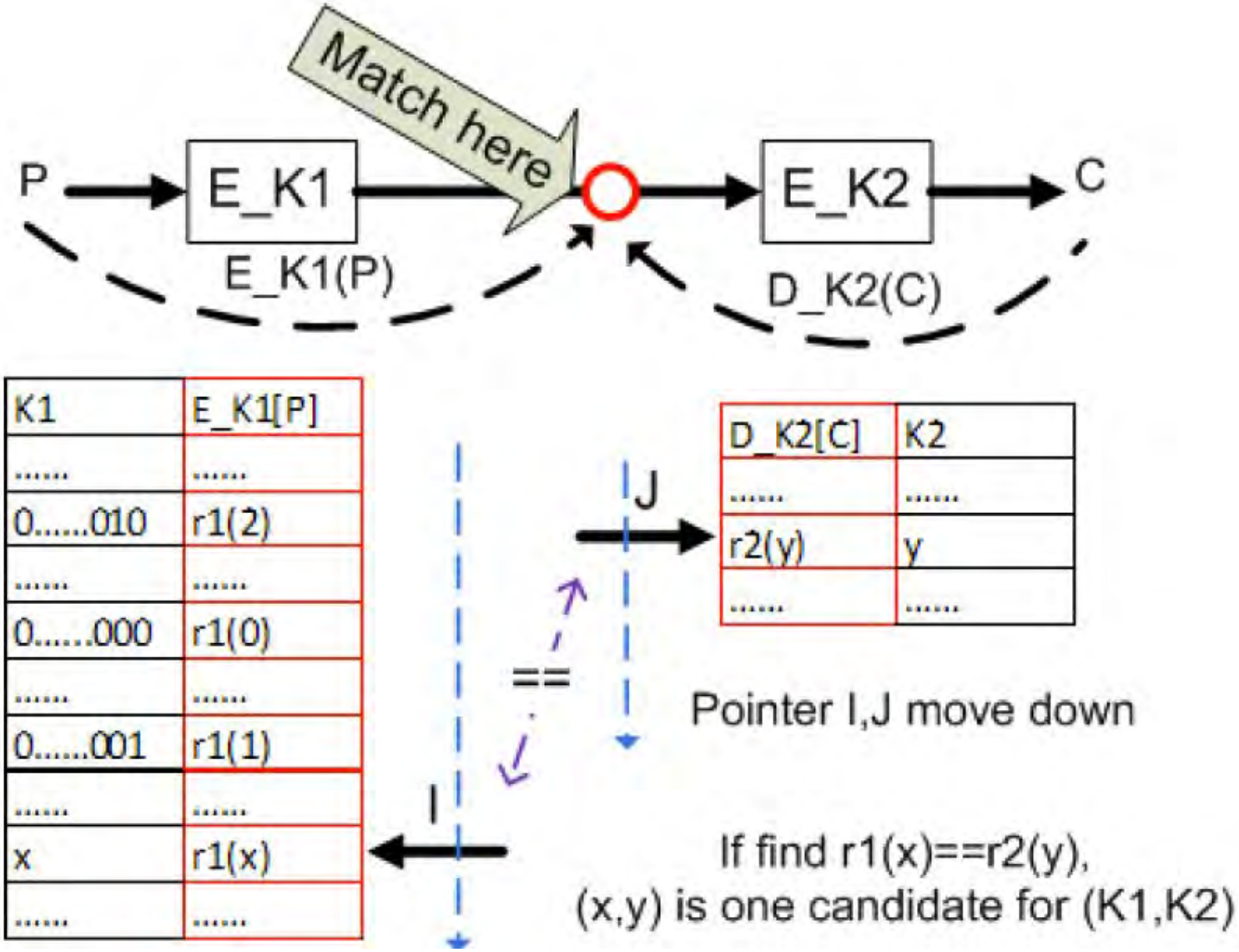
Attack Steps:

- First make Table A with 2^{56} entries, where each entry consists of a DES key K and the result *r* of applying that key to encrypt *p1.* Then sort the table in numerical order by *r*
- Make Table B with 2^{56} entries, where each entry consists of a DES key K and the result *r* of applying that key to decrypt *c1.* Also sort Table B in numerical order by *r*
- Search through the sorted tables to find matching entries $<K_A,r>$ from Table A and $<K_B,r>$ from Table B. Each match provides K_A as candidate K1 and K_B as a candidate K2 because $D_{K_B}[c1] = E_{K_A}[p1]$
- If multiple candidate pairs of K_A and K_B are found in Step 3, use each candidate key-pair to encrypt p2, p3, …etc to see if it results in c2, c3,…; The "real" key-pair will always work ; the other "coincident" key-pairs will almost surely fail on at least one of the other <pi,ci> pairs

Meet-in-the-Middle Attack on 2DES (cont'd)



Meet-in-the-Middle Attack on 2DES (cont'd)



NIST Contest for Advanced Encryption Standard (AES)

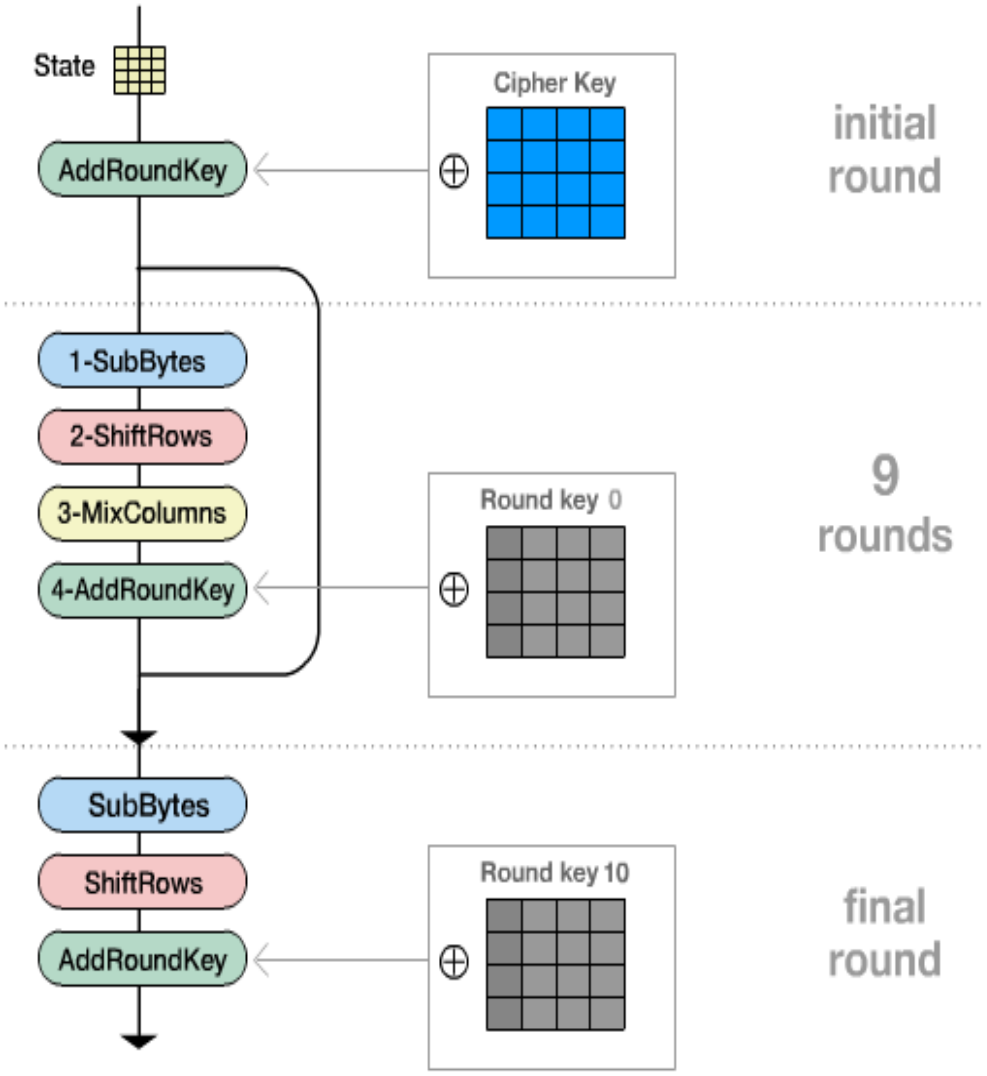
- NIST had an open call for proposals, actually a contest, in 1997
  - 21 submissions from all over the world ; 15 fulfilled all the requirement (8 from North America, 4 from Europe, 2 from Asia, 1 from Australia)
  - Narrow down to five final candidates in August 1999:
    - Rijndael (Belgium), Serpent (England, Israel, Norway), MARS (IBM) , Twofish (US), RC6 (US)
- After vigorous evaluation and testing, **Rijndael** [rain´ dow] was selected as the winner in 2000 and Standardized as AES effective May 2002
- By two Belgium cryptographers: Joan Daeman and Vincent Rijmen
- AES is expected to replace DES and 3DES as “The Standard” encryption work-horse world-wide

Overview of AES

4 transformations for Each Round:

- Substitute Bytes
- Shift Rows
- Mix Columns
- Add Round Key

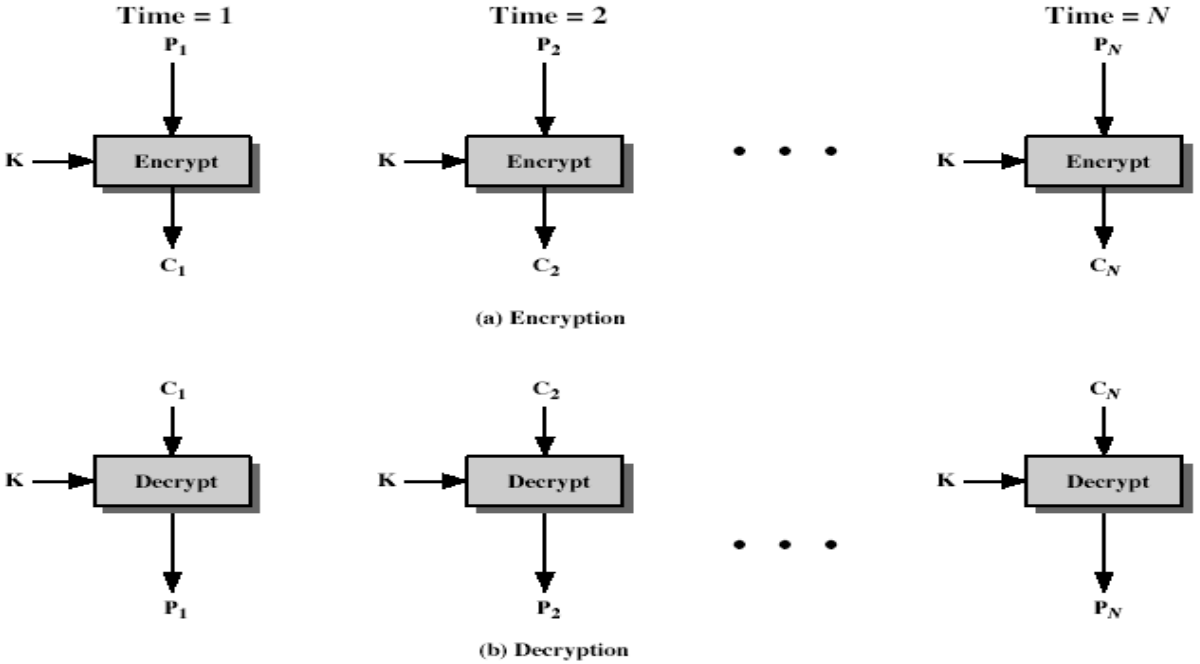
- Number of rounds depends on key length (10,12,14 rounds for 128,192,256-bit keys respectively



Modes of Operation for Block Cipher

- Break the piece of plaintext into **64-bit blocks ;** pad the last block to 64 bit
- Basic Mode of Operation: Use **Electronic Code Book** (ECB)
  - Each block of plaintext is encrypted *independently* using the same key
  - Repeated plaintext block will produce the same ciphertext block

    => can leak information
  - Blockwise swapping of ciphertext may still produce meaningful output upon decryption
  - Mainly used for sending a small number of blocks of information only



Cipher Block Chaining (CBC) Mode

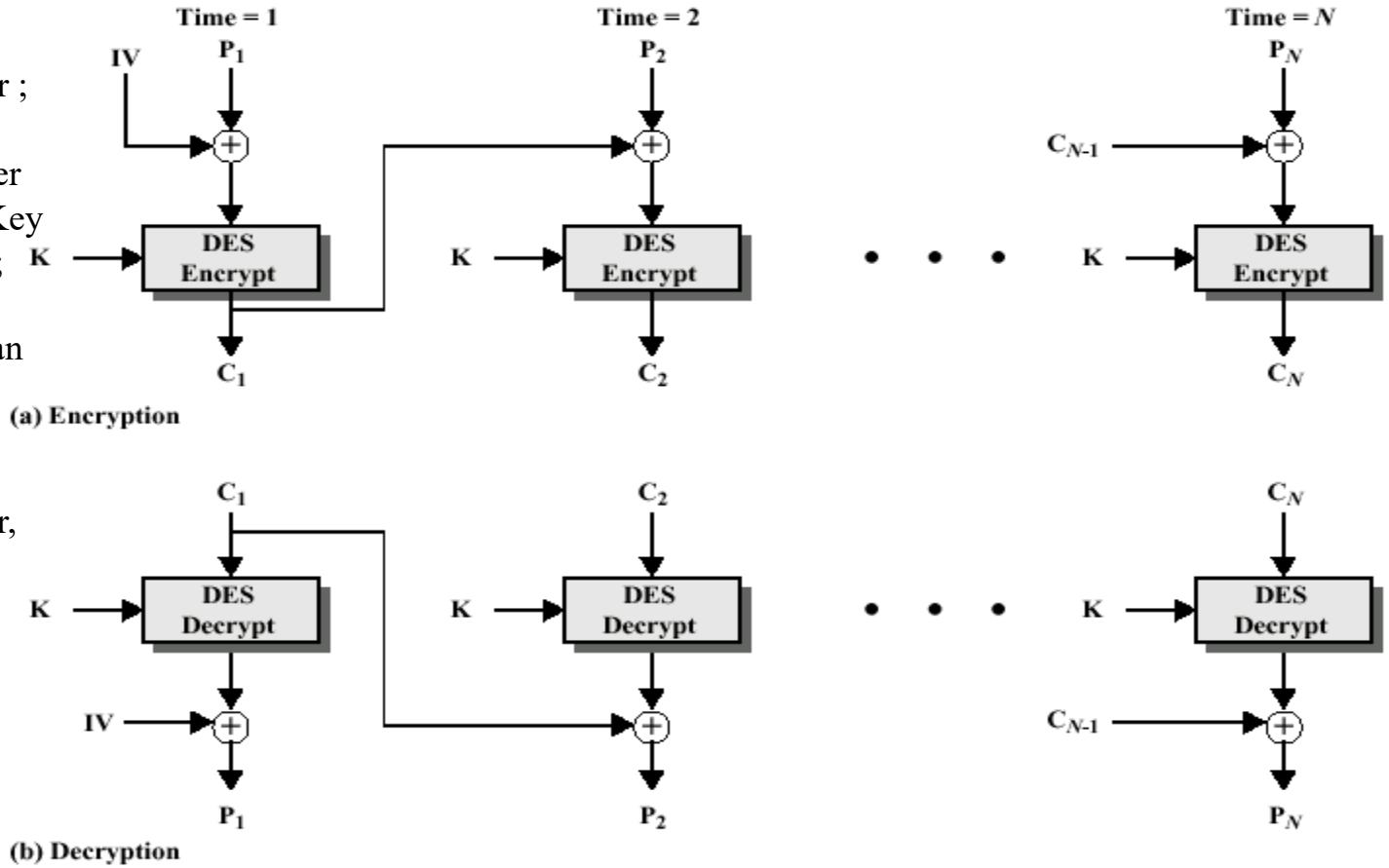
- Input to algorithm is the **XOR** of current plaintext block and preceding ciphertext block
- **Repeating patterns** are **not** exposed
- But what if the ciphertext is corrupted or lost during transmission ?

IV stands for Initialization Vector ;

Sender and Receiver Need to share the Key and IV in advance ;

An encrypted IV can be Sent in advance

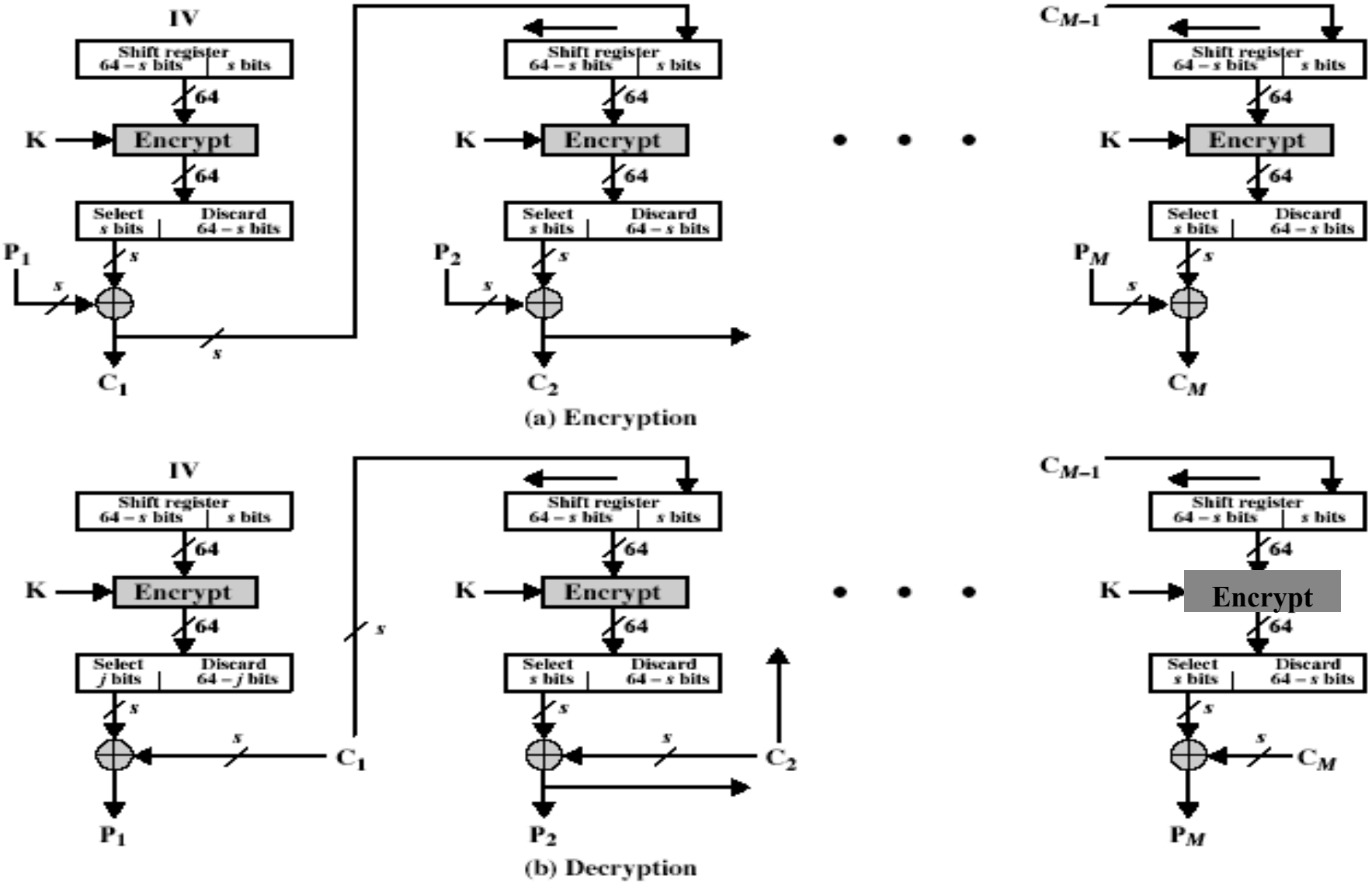
If IV is sent in clear, Attacker can manipulate P1 and compensate by modifying the IV



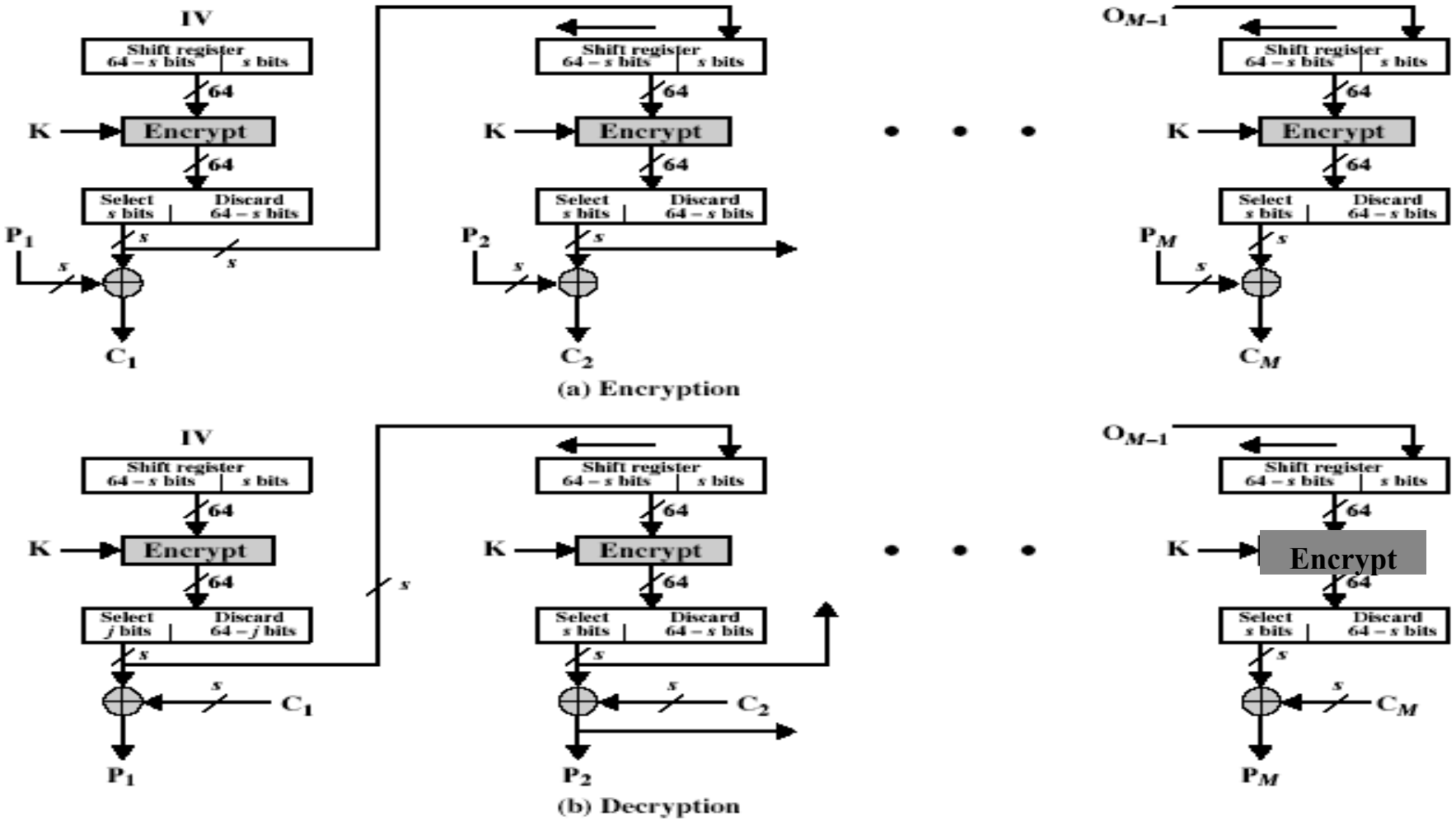
Insecurity of MAC-then-Encrypt mode of Cipher Block Chaining (CBC)

- Which one is better: MAC-then-Encrypt or Encrypt-then-MAC ?
- TLS Ver1.1 chose MAC-then-Encrypt (MtE), namely
  1. Authenticate (protect the integrity of) the plaintext (using HMAC) ;
  2. Add the HMAC code at the end of the message ;
  3. Pad the message length to the required block-length ;
  4. Encrypt the resultant block using a block cipher, e.g. AES ;
- **Unfortunately, such implementations of MAC-then-Encrypt mode of CBC modes, in TLS-1.1, 1.2, are subject to various variants of Padding-Oracles+Timing attacks,**
  - **e.g. Vaudenay Padding Oracle (2002), Lucky 13 (2013), BEAST (2011), etc,**
    https://blog.cloudflare.com/padding-oracles-and-the-decline-of-cbc-mode-ciphersuites/
  - **CBC MtE modes have been depreciated (i.e. removed, disallowed) by new TLS standards (v1.3)**
  - https://www.cloudflare.com/learning-resources/tls-1-3/

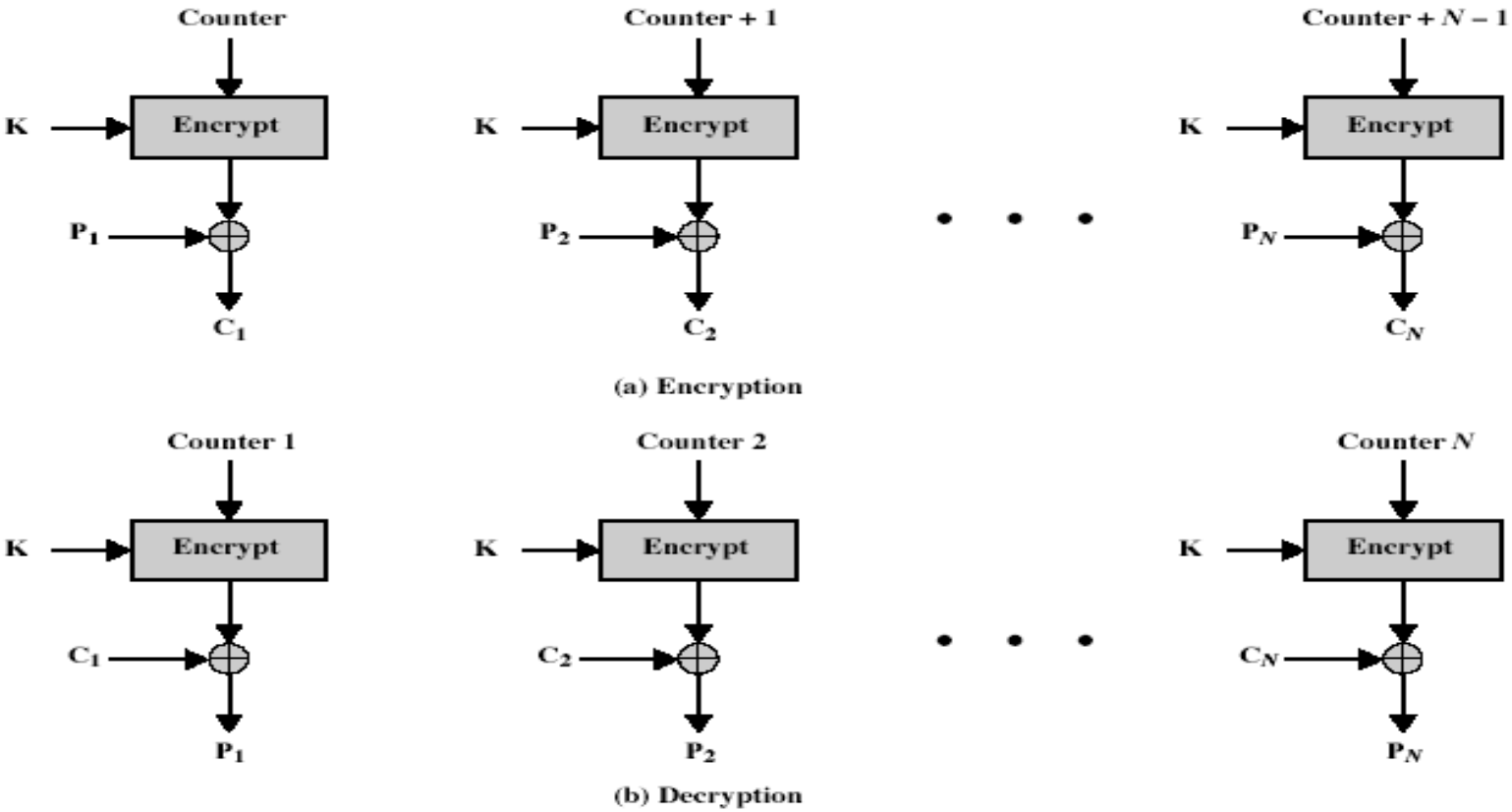
Cipher FeedBack (CFB) Mode



Output FeedBack (OFB)



Counter (CTR)


Counter
K
Encrypt
P_1
C_1
Counter + 1
K
Encrypt
P_2
C_2
Counter + N – 1
K
Encrypt
P_N
C_N
(a) Encryption
Counter 1
K
Encrypt
C_1
P_1
Counter 2
K
Encrypt
C_2
P_2
Counter N
K
Encrypt
C_N
P_N
(b) Decryption

Birthday Attacks on CBC, CFB, OFB, CTR

- For a lot of modes of operations for Block Cipher, we must ensure not to reuse key/ counter values for different plaintexts, otherwise information can be leaked ;

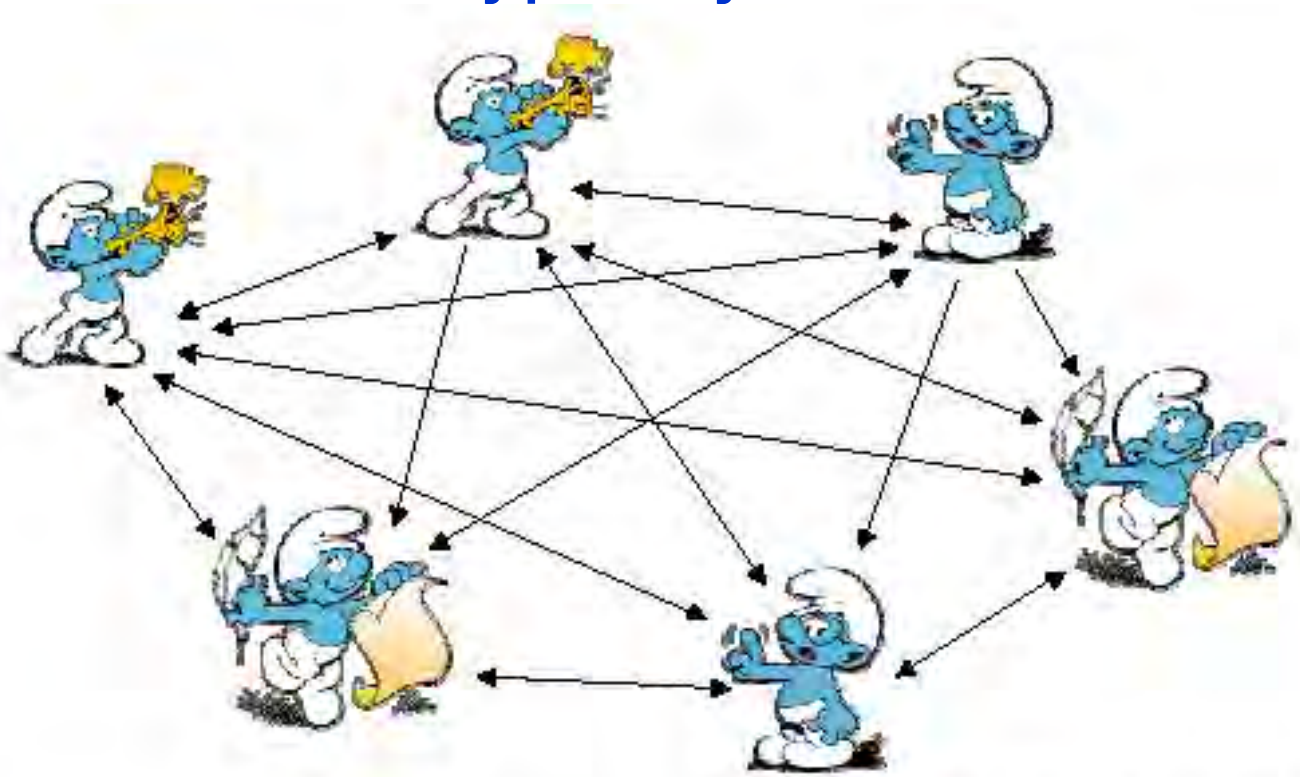
=> CBC, CFB, OFB and CTR modes of operation for Block ciphers are subject to so-called Birthday Attacks:

  - To stay safe, re-keying is required before encrypting $2^{(\text{block-size} / 2)}$ of input blocks, e.g. for DES, or 3DES, it means rekeying more frequent than 2^{32} input blocks

  Refer to the following for further details:

  - Sweet32 paper in CCS 2016 https://sweet32.info/SWEET32_CCS16.pdf
  - D. McGrew 2012 paper https://eprint.iacr.org/2012/623

Key Distribution Problem for Secret Key Crypto-systems



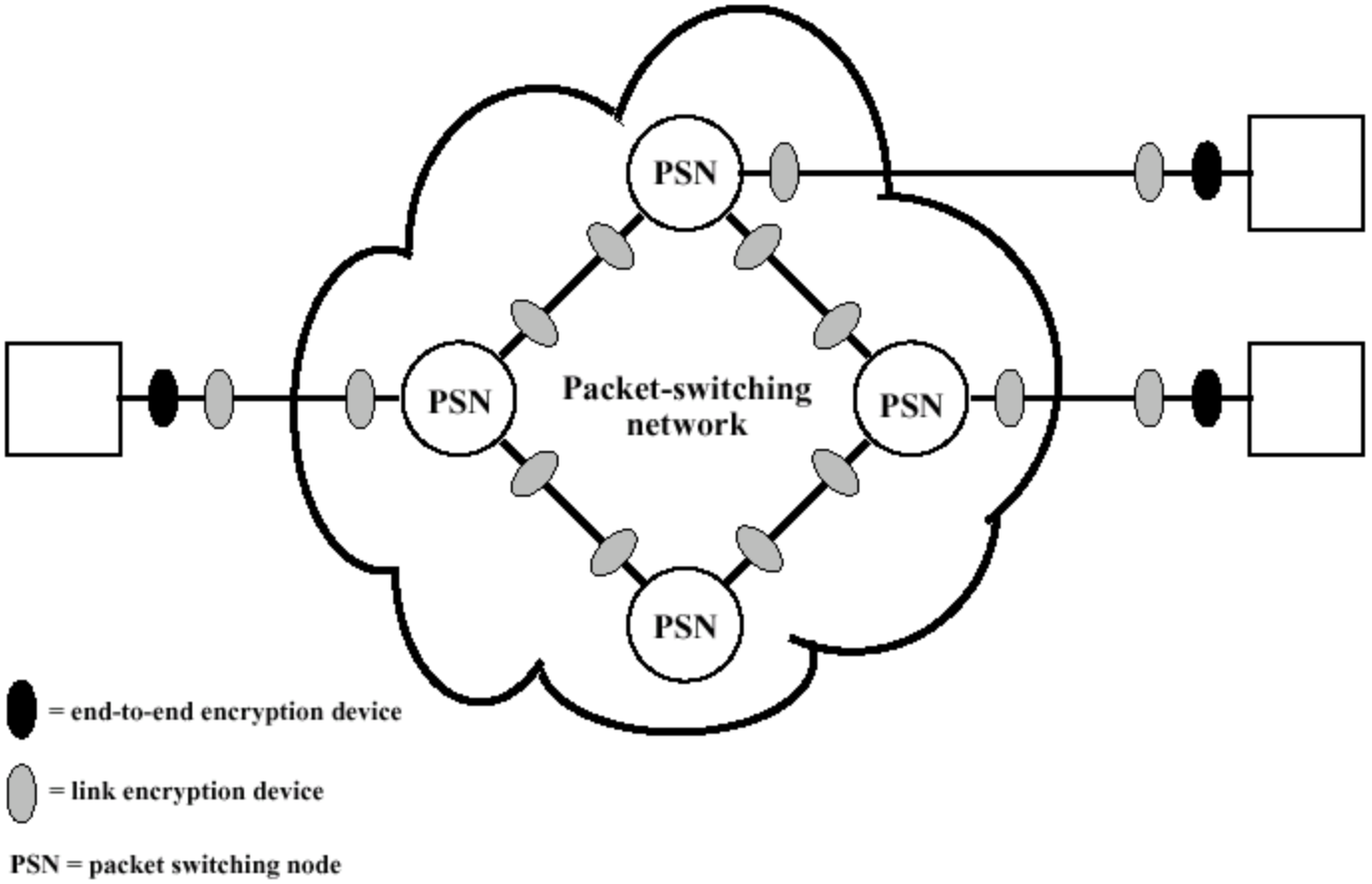
N - Users ⇨ $\frac{N \cdot (N-1)}{2}$ Keys

Users	Keys
100	5,000
1000	500,000

Key Distribution

- **Both parties** must have the **secret key**
- Key is **changed frequently**
- Requires either manual **delivery** of keys, or a third-party encrypted channel
- Most effective method is a **Key Distribution Center** (e.g. Kerberos)
- More later in the course…

Location of Encryption Devices



Message Authentication Code
Hash Function and Message Digest

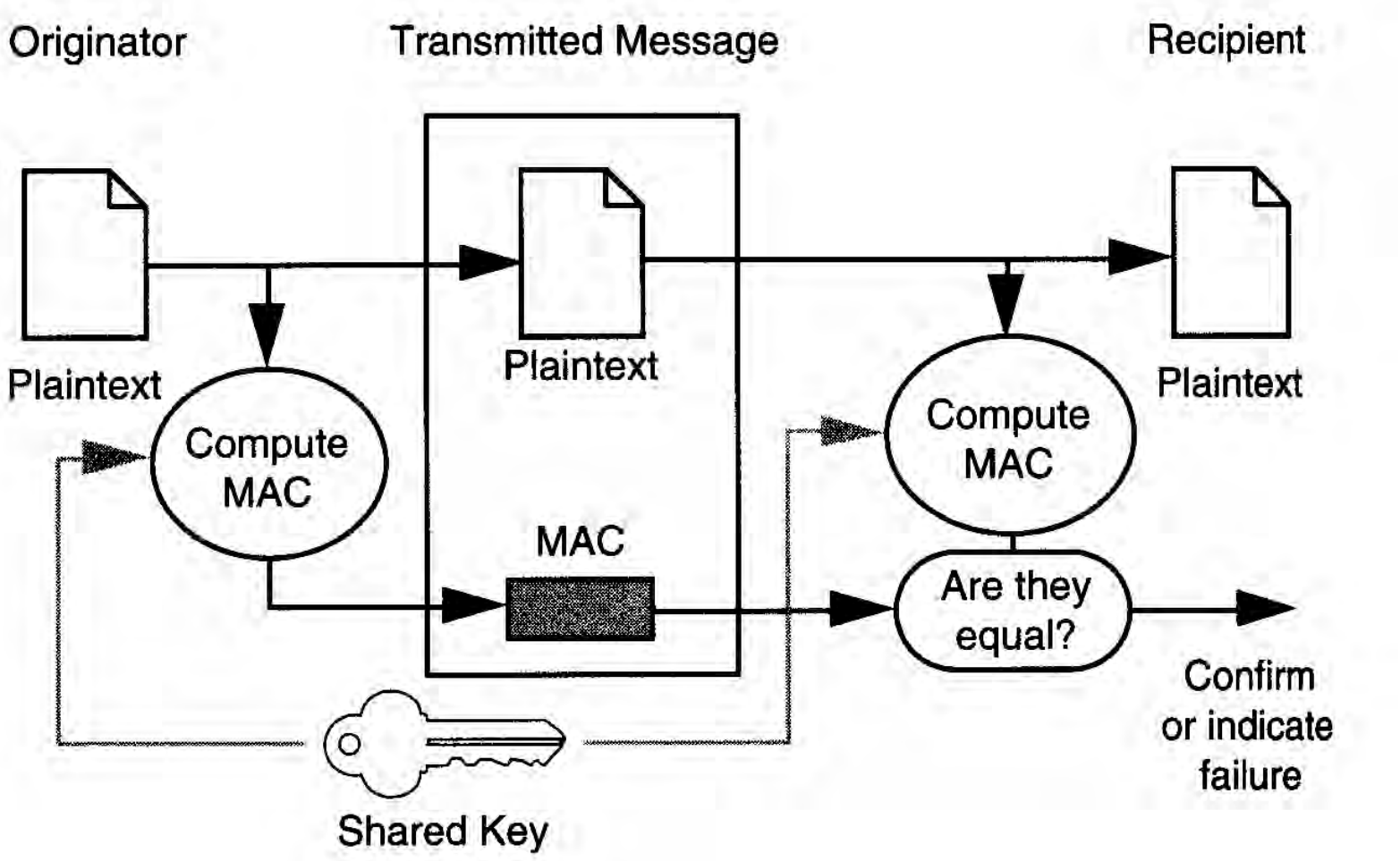
What is Message Authentication ?

- Procedure that allows communicating parties to verify that received messages are authentic, namely
  - source is authentic – not from masquerading
  - contents unaltered – message has not been modified
  - timely sequencing – the message is not a replay of a previously sent one

Ways to provide Message Authentication

- Message Authentication via Conventional Encryption
  - Only the sender and receiver should share a key ;
  - Include a time-stamp or “nonce” to prevent replay attack
  - Implicitly assume the receiver can recognize if the output from the decryption unit is garbage or not ;
    - easy if they know the message has some specific format, e.g. English
    - May be difficult if the original plaintext are random binary data =>need to impose some structure, e.g. Checksum
- Message Authentication without Message Encryption (thus no message confidentiality)
  - An authentication tag (aka Message Authentication Code or MAC) is generated and appended to each message where
    - the MAC is computed as a publicly known function F, of the message M and a shared secret key K:
      - MAC = F(K, M)
  - A *one-way* Hash function can be used as F

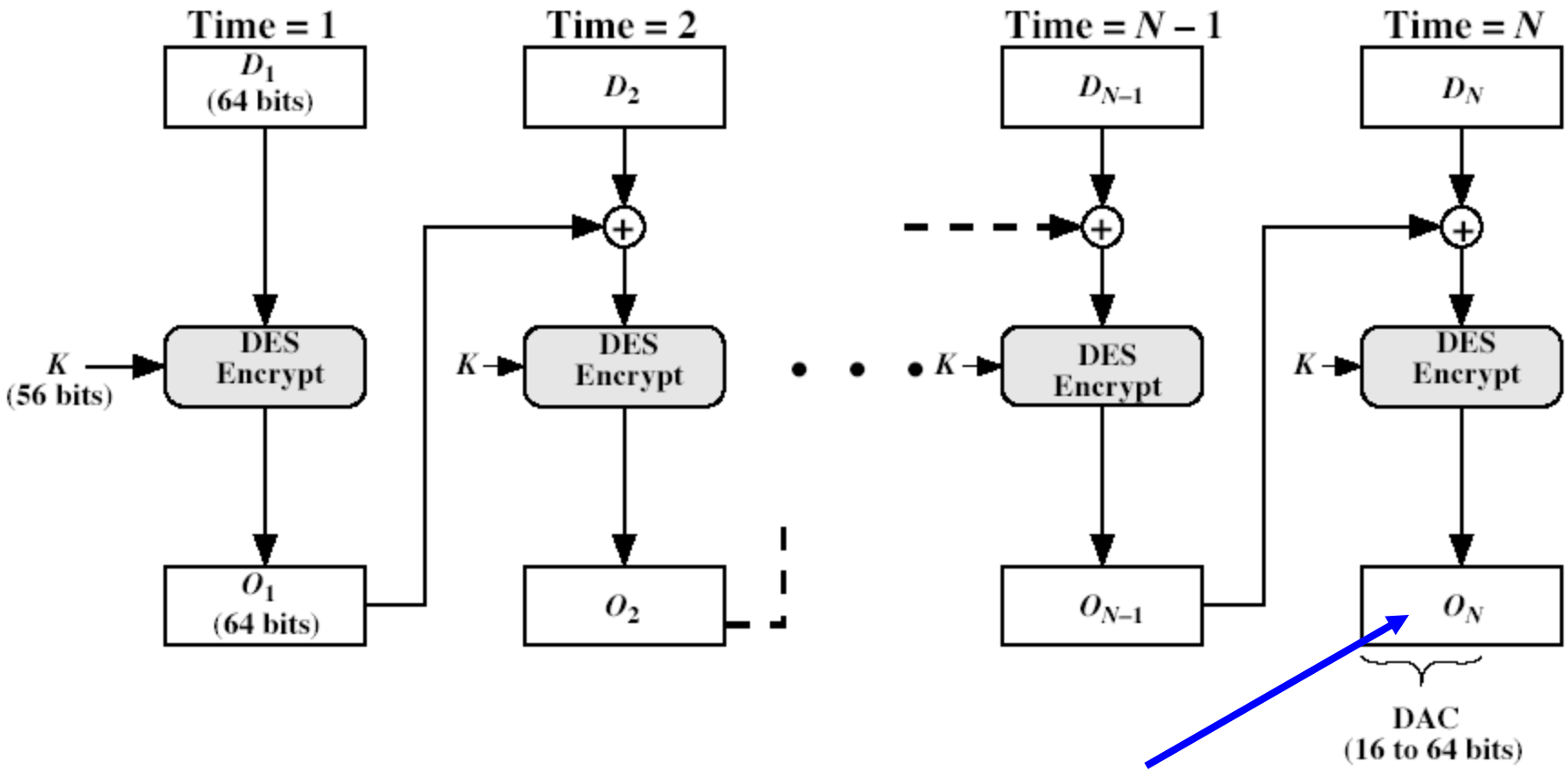
Ensuring Message Authenticity using a MAC



Message Authentication Code

- Receiver assured that message is not altered – no modification
- Receiver assured that the message is from the alleged sender – no masquerading
- Include a sequence number, assured proper sequence – no replay

CBC-residue as MAC



CBC-residue, aka DAC = Data Authentication Code

One-Way Hash Function

- Hash function accepts a *variable* size message M as input and produces a *fixed-size* message digest $H(M)$ as output
- Message digest is sent with the message for authentication
- Produces a fingerprint of the message
- No secret key is involved

One-way Hash Function Requirements

weak

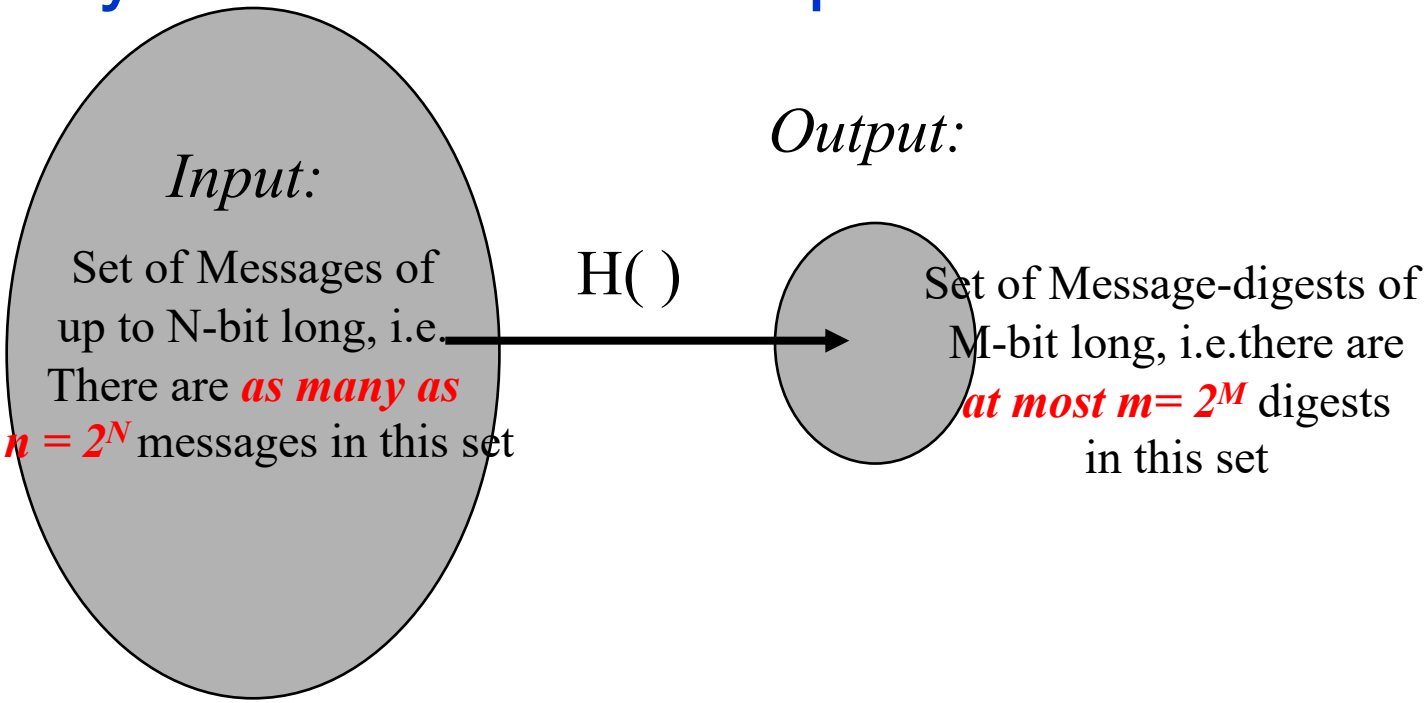
1. H can be applied to a block of data of any size
2. H produces a fixed length output
3. $H(x)$ is relatively easy to compute
4. For any given code h, it is computationally infeasible to find x such that $H(x) = h$ (i.e. safe against the so-called 1st preimage attack)
5. For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$ (i.e. safe against the so-called 2nd preimage attack)
6. It is computationally infeasible to find any pair (x,y) such that $H(x) = H(y)$

one way

strong collision resistance
birthday attack

weak collision resistance

How likely to have Hash output collisions ?



- Since $N >> M$, (and therefore) $n >> m$, collisions are inevitable no matter how secure the one-way function H( ) is.

The Birthday Paradox

- In a room with ***n*** people, what is the probability that we will find at least 2 people who have the same birthday (there are $m = 365$ possible choices of birthday)?

An *approximate* analysis:

- Assuming birthdays are uniformly distributed over the entire year. For any given pair of people, the possibly of them having the same birthday is $1/m = 1/365$;
- There are ${}_nC_2 = n(n-1)/2$ ways to select a pair out of n people
- Let $P_{collision}$ be the Probability of at least one collision,
  - $P_{collision}$ approx. $= n(n-1)/2 * 1/m = n(n-1)/2m$;
- $P_{collision} > ½$ when $n >= 20$
- In general, $P_{collision} > ½$ when n becomes $>= \sqrt{m}$
- The approximation is not good when n approaches m

- Where is the approximation ?

How difficult to find a Hash collision ?

How secure is a one-way hash with 64-bit output, e.g. CBC-DES ?

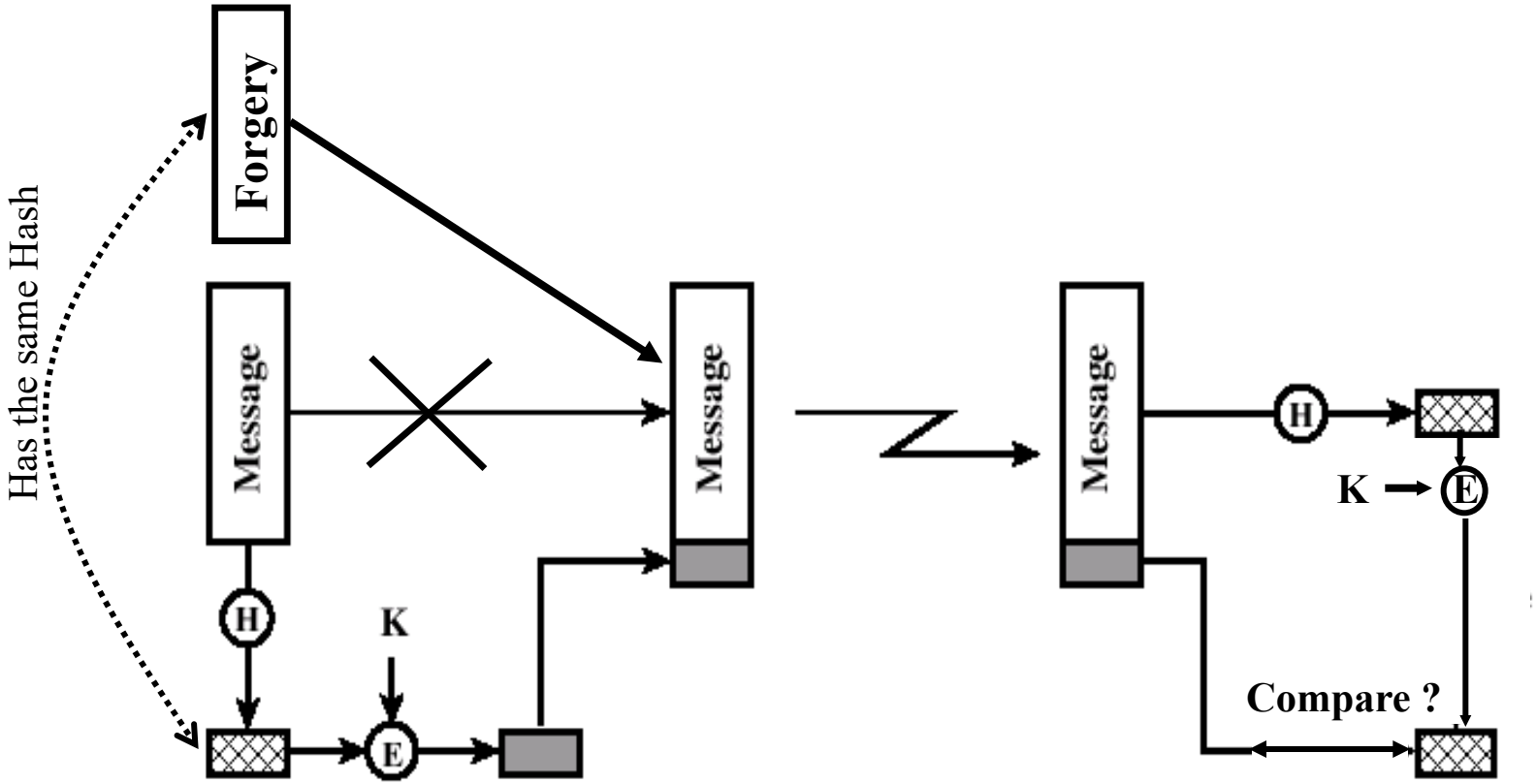
- Based on the property of a good hash function, the hash output of any input string should be uniformly distributed over the hash output space of size $m=2^{64}$
  - This is analogous to the fact that the birthday of any given person is uniformly distributed over any days within a year (i.e. output space of size m = 365)

- Thus, according to the Birthday Paradox, if no. of all possible outcomes = m, we only need to try about $n = \sqrt{m}$ inputs to the hash function to have a good chance to find a collision, e.g.

  For, a hash function with 64-bit output, $m=2^{64}$

=> it only takes about $\sqrt{m} = 2^{32}$ tries to find a pair of inputs which will produce the same hash output, i.e. a collision

Birthday Attack on Message Digest



Using CBC-residue as Message Authentication Code

Birthday Attacks

- **Birthday attack can proceed as follows:**
  - opponent generates 2^{32} variations of a valid message, all with essentially the same meaning ; this is “doable” given current technology.
  - opponent also generates 2^{32} variations of a desired fraudulent message
  - two sets of messages are compared to find a pair with same hash output (by argument similar to the Birthday paradox, this probability > 0.5)
  - have user (the victim) sign the valid message, but sent the forgery message which will have a valid message digest

- Conclusion is that we need to use longer MACs

- BTW, how can we generate 2^{32} variations of a letter carrying the same meaning ?

  Just 2 choices of wording at 32 different places.

How to generate large no. of messages of each type to get the necessary message digest collision to pull off a B-day attack ?

Type 1 message

I am writing {this memo | } to {demand | request | inform you} that {Fred | Mr. Fred Jones} {must | } be {fired | terminated} {at once | immediately}. As the {July 11 | 11 July} {memo | memorandum} {from | issued by} {personnel | human resources} states, to meet {our | the corporate} {quarterly | third quarter} budget {targets | goals}, {we must eliminate all discretionary spending | all discretionary spending must be eliminated}.

{Despite | Ignoring} that {memo | memorandum | order}, Fred {ordered | purchased} {Post-Its | nonessential supplies} in a flagrant disregard for the company's {budgetary crisis | current financial difficulties}.

Type 2 message

I am writing {this letter | this memo | this memorandum | } to {officially | } commend Fred {Jones | } for his {courage and independent thinking | independent thinking and courage}. {He | Fred} {clearly | } understands {the need | how} to get {the | his} job {done | accomplished} {at all costs | by whatever means necessary}, and {knows | can see} when to ignore bureaucratic {nonsense | impediments}. I {am hereby recommending | hereby recommend} {him | Fred} for {promotion | immediate advancement} and {further | } recommend a {hefty | large} {salary | compensation} increase.

Use a Secure Hash Function to commit an Answer/Choice without disclosing it (yet)

Challenge:

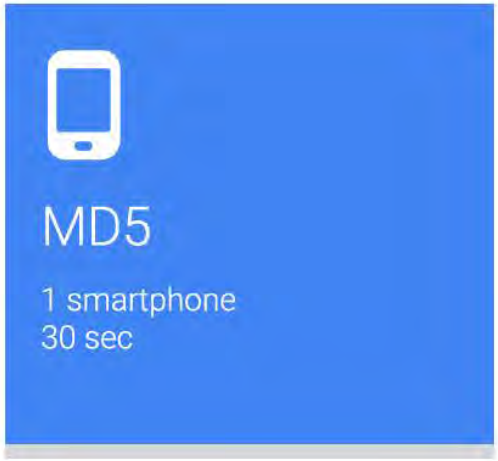
Alice: How many COVID-19 cases will there be tomorrow ?

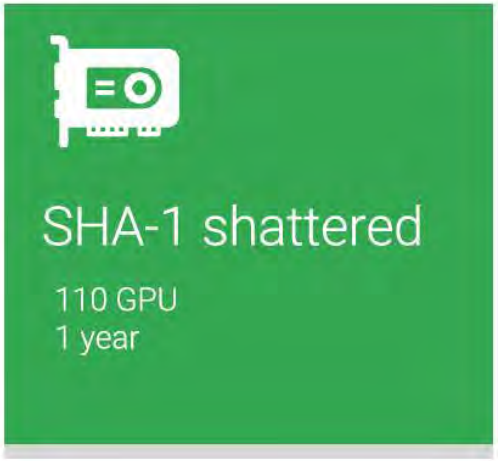
Bob: I actually know the Secret Answer to your question, but I am not allowed to tell you right now **!**

Alice: Can you prove that you really know the Secret Answer in advance ?

Bob: Here is Hash output of the Secret Answer
= Hash(My Answer, followed by Some Random Bits) ;
Hold onto this Hash output and wait till tomorrow. You will agree I indeed know this Secret Answer in advance.

Computational Cost Comparison





The SHA-2 Family

- SHA-2 is a set of cryptographic hash functions:
  - SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224,SHA-512/256
- Designed by NSA and published by NIST in 2001 as a U.S. FIPS (Federal Information Processing Standard).
- SHA-2 bears some similarities of SHA-1, but contains some key changes
  - Attacks on SHA-1 have not been successfully extended to SHA-2.

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Word size (bits)	Rounds	Operations	Collisions found	Example Performance (MiB/s)[13]
MD5 (as reference)		128	128	512	$2^{64} - 1$	32	64	+,and,or,xor,rot	Yes	255
SHA-0		160	160	512	$2^{64} - 1$	32	80	+,and,or,xor,rot	Yes	-
SHA-1		160	160	512	$2^{64} - 1$	32	80	+,and,or,xor,rot	Yes	153
SHA-2	*SHA-224* *SHA-256*	224 256	256	512	$2^{64} - 1$	32	64	+,and,or,xor,shr,rot	None	111
	SHA-384 *SHA-512* *SHA-512/224* *SHA-512/256*	384 512 224 256	512	1024	$2^{128} - 1$	64	80	+,and,or,xor,shr,rot	None	99

The NIST SHA-3 Competition (2006-2012)

http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

- On Dec. 9, 2010, the Final FIVE candidates for the Round 3 of the competition were announced:
  - http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/documents/Email_Announcing_Finalists.pdf
  - http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/submissions_rnd3.html
- The Winning algorithm: Keccak, (pronounced “catch-ack”) was announced on Oct 2, 2012, to be called SHA-3 in Standards ;
  - Designed by a team of researchers from Belgium and Italy
  - http://keccak.noekeon.org
  - NSA believes both SHA-2 and SHA-3 are secure and can be used in practice.
    - Since SHA-2 and SHA-3 differ substantially in their designs and theory, this diversity can provide system designers a fallback solution in case one of them is broken in the future.
- SHA-3 approved as a new hashing standard by NIST of U.S..
  - Published as FP202 on Aug. 5, 2015.

Complementing SHA2

- SHA3 is expected to deployed into a world full of SHA2 implementations
- SHA2 still looks strong
- NIST expect the standards to coexist.
- SHA3 should *complement* SHA2.
  - Good in different environments
  - Susceptible to different analytical insights

Keccak (SHA3) is fundamentally different from SHA2. Its performance properties and implementation tradeoffs have little in common with SHA2.

Comparison of SHA functions

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Rounds	Operations	Security (bits)	Example Performance[29] (MiB/s)
MD5 (as reference)		128	128 (4×32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or	<64 (collisions found)	335
SHA-0		160	160 (5×32)	512	$2^{64} - 1$	80	And, Xor, Rot, Add (mod 2^{32}), Or	<80 (collisions found)	-
SHA-1		160	160 (5×32)	512	$2^{64} - 1$	80		<80 (theoretical attack[30] in 2^{61})	192
SHA-2	*SHA-224*	224	256 (8×32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr	112	139
	SHA-256	256						128	
	SHA-384	384	512 (8×64)	1024	$2^{128} - 1$	80	And, Xor, Rot, Add (mod 2^{64}), Or, Shr	192	154
	SHA-512	512						256	
	SHA-512/224	224						112	
	SHA-512/256	256						128	
SHA-3	*SHA3-224*	224	1600 ($5 \times 5 \times 64$)	1152	∞	24	And, Xor, Rot, Not	112	-
	SHA3-256	256		1088				128	
	SHA3-384	384		832				192	
	SHA3-512	512		576				256	
	SHAKE128	*d* (arbitrary)		1344				min (*d*/2, 128)	-
	SHAKE256	*d* (arbitrary)		1088				min (*d*/2, 256)	

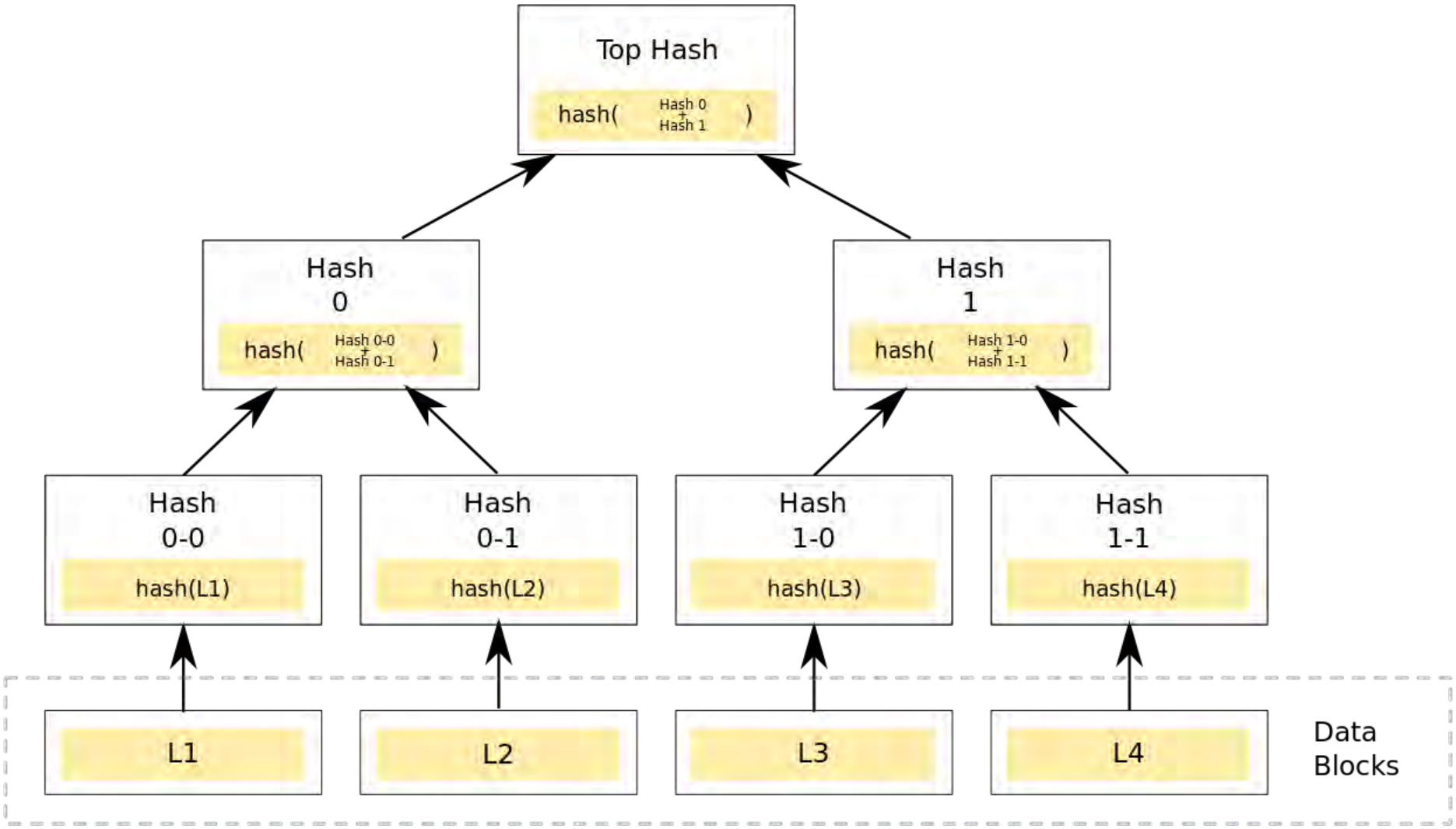
The Future of Hash Security is Diversity

	Security Claim	Fixed prefix	Chosen attack
~~SHA-1~~	MD		
SHA-256	MD	2^{128}	
SHA-3	Sponge	2^{128}	2^{128}
BLAKE	HAIFA	2^{128}	2^{256}

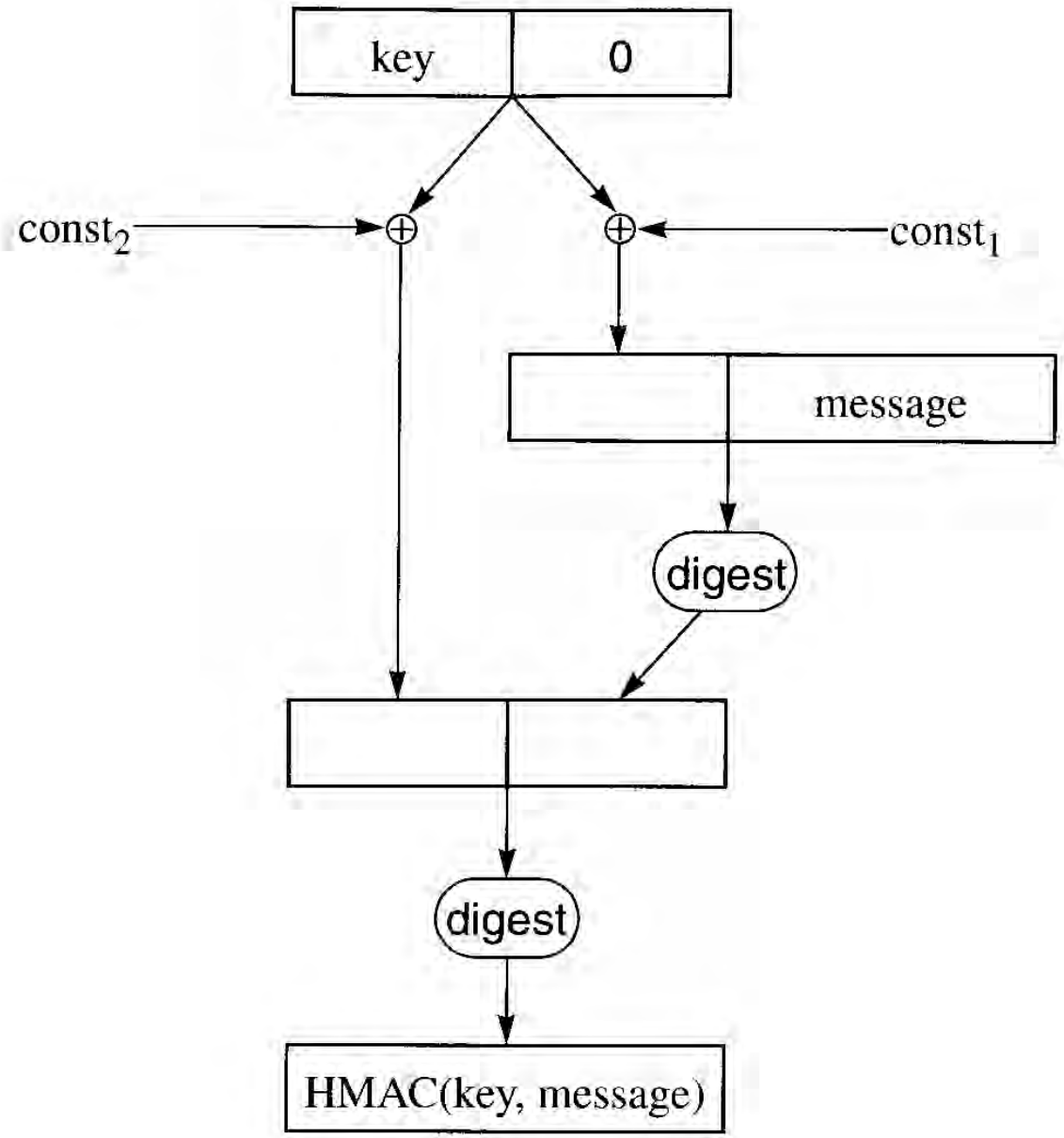
Research at Google

https://shattered.io

A Hash Tree (Merkle Tree)



HMAC



By XORing key with const1 and const2, we have pseudo-randomly generated two new keys from the original key

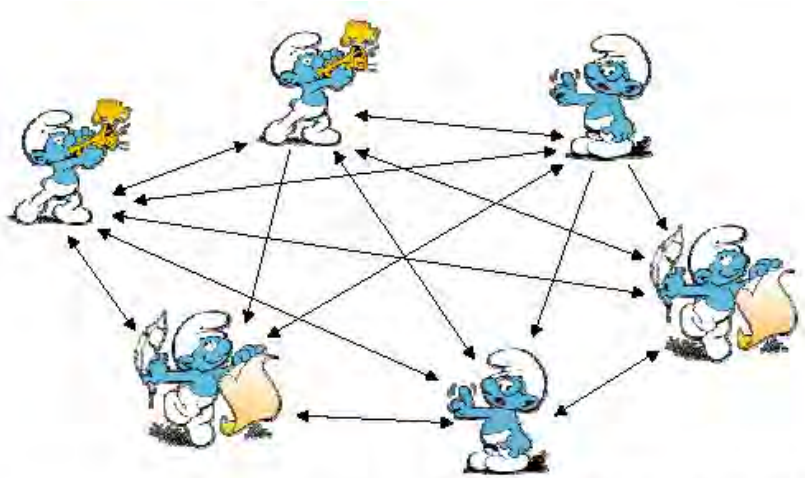
HMAC

- Effort to develop a MAC derived from a cryptographic hash codes such as SHA-256
- Executes faster in software
- No export restrictions
- Relies on a secret key
- RFC 2104 list design objectives and
- Provable security properties
- Used in IPsec, TLS
- Can use different digest functions as a component, e.g.
  - HMAC-SHA256 , HMAC-SHA3 ;
- Informational RFC6151 (circa 2011) concluded that: Although the security of MD5 hash function itself is severely compromised, the currently known “attacks on HMAC-MD5 do not seem to indicate a practical vulnerability when used as a message authentication code,” but “for new protocol design, a ciphersuite with HMAC-MD5 should NOT be included.”

Public Key Cryptography

Motivation for Public Key Cryptography

- In symmetric key or secret key cryptosystems, the communication parties must have some pre-share secret, i.e. the master key
- Distribution of such keys in a secure and scalable manner is a major problem
  - The introduction of a trust third party, namely the Key Distribution Center (KDC) solves the problem partially by reducing the number of master keys from $O(N^2)$ to $O(N)$ but still inconvenient and KDC can become the single point of failure and/or performance bottleneck (more details later)
  - Symmetric key system also cannot provide *non-repudiation*

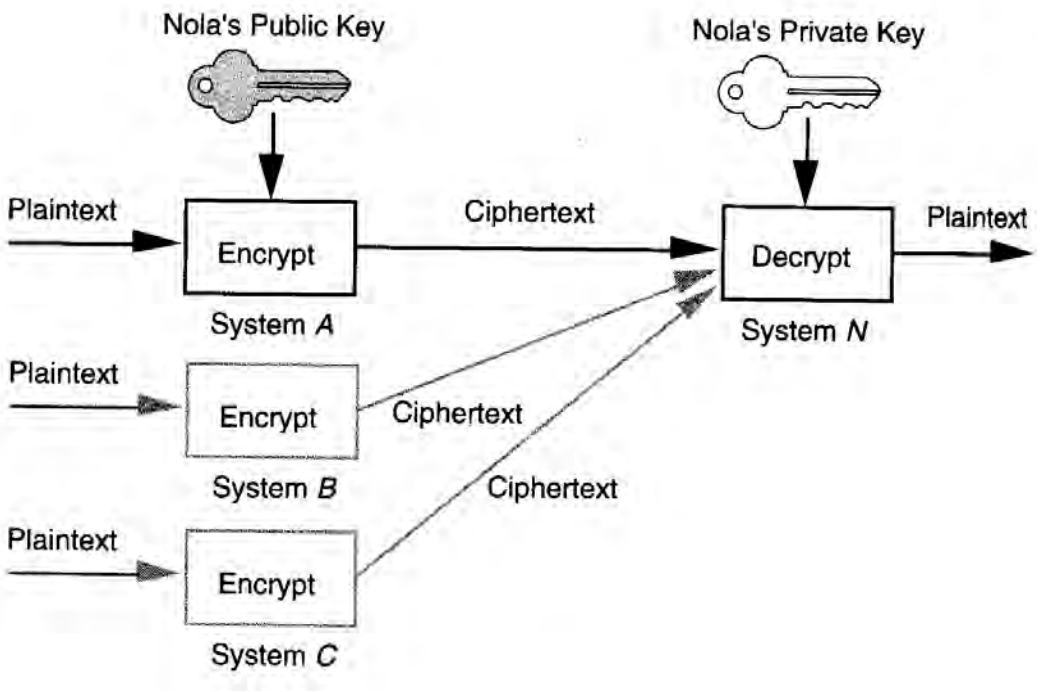


N - Users ⇨ $\frac{N \cdot (N-1)}{2}$ Keys

Users	Keys
100	5,000
1000	500,000

The Concept of Public Key Cryptography

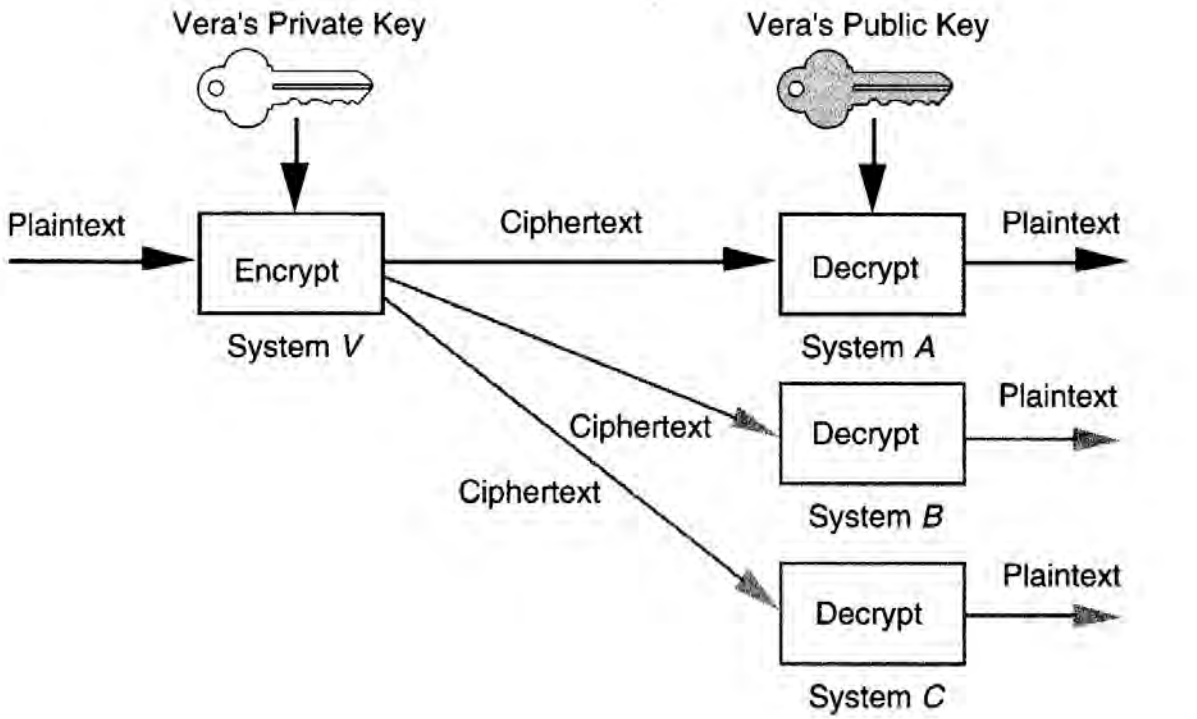
- Every participant has a pair of keys: the Public Key and Private Key
- The Public key is published or sent to everyone else in the community openly
- The Private key is kept secret by its owner
- Plaintext encrypted by A's public key can only be decrypted by A's private key
- Some Ciphertext can be decrypted by A's public key if and only if it has been encrypted by A's private key



To send Nola a secret message, any sender first finds Nola's Public Key, e.g. from a public directory, and uses it for encrypting the message. Only the person who has Nola's private key (presumably Nola's herself) can decrypt the message successfully

•Note: No need for secure distribution of pre-shared secret key anymore

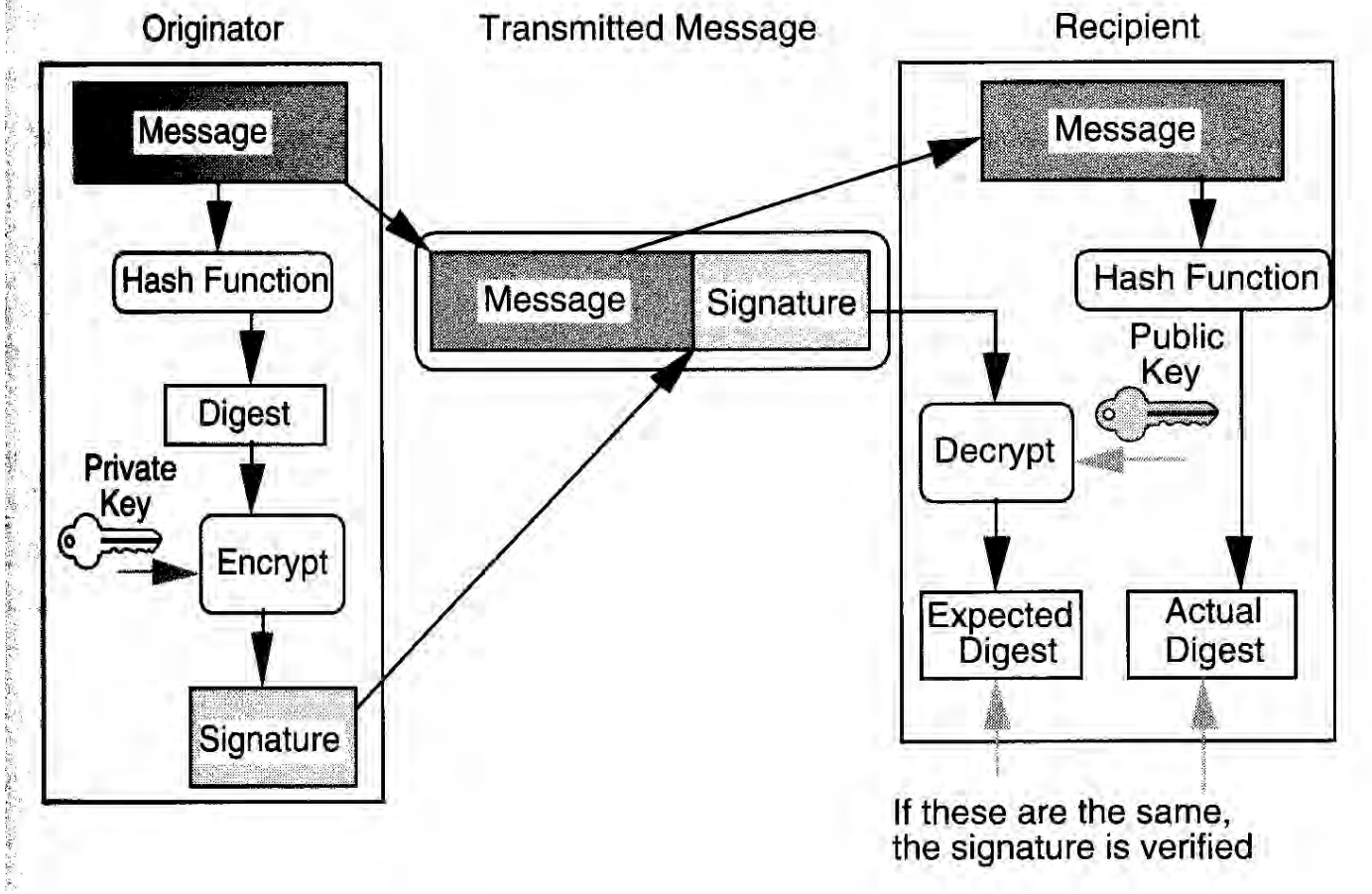
The Concept of Public Key Cryptography (cont'd)



Hmm…if I can decrypt successfully an incoming message with Vera's public key, the message must have been encrypted with Vera's private key. Since Vera is required (e.g. by law) to keep her private key secret to herself, no one but Vera could have encrypted (and sent) the message

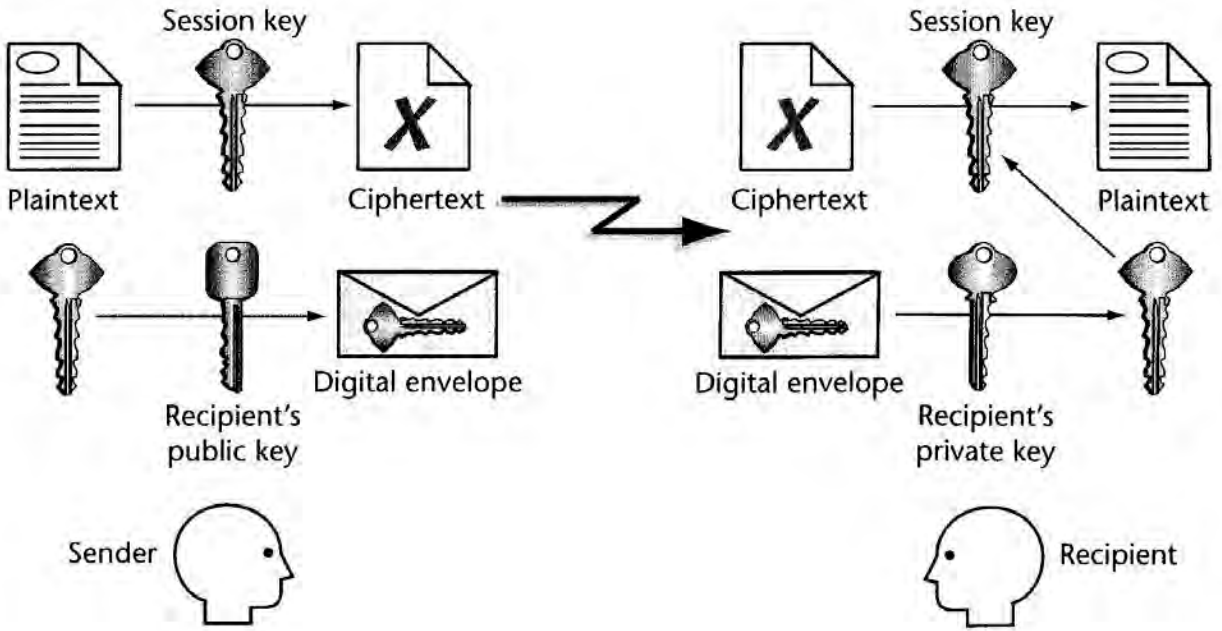
=> This provides the notion of digital signature and thus non-repudiation service

Digital Signature (cont'd)



- Instead of signing the entire message, one can sign the digest of the message to improve performance because public key algorithms are much slower than secret key ones. One should avoid using public key algorithms to encrypt large amount of data (long messages)

Use Public-key encryption to “seal” a digital envelope



- The sender picks a “secret” Session Key to encrypt the long message using a secret key algorithm, e.g. AES.
- By encrypting the session key with the Recipient’s public key, the session key can be delivered securely to the recipient without any pre-shared secret between the 2 parties
- Conversely, we can consider this as doing a secure session-key exchange using public key encryption

RSA Algorithm

- Ron Rivest, Adi Shamir, Len Adleman – found the functions and published the results in 1978:
  - *D[E[m]] = m = E[D[m]]*
- Most widely accepted and implemented approach to public key encryption
- Block cipher where m = plaintext ; and c =ciphertext are integers, between $0 <= m , c <= n-1$ for some n
- Following form:

This is the E[ ]

This is the D[ ]

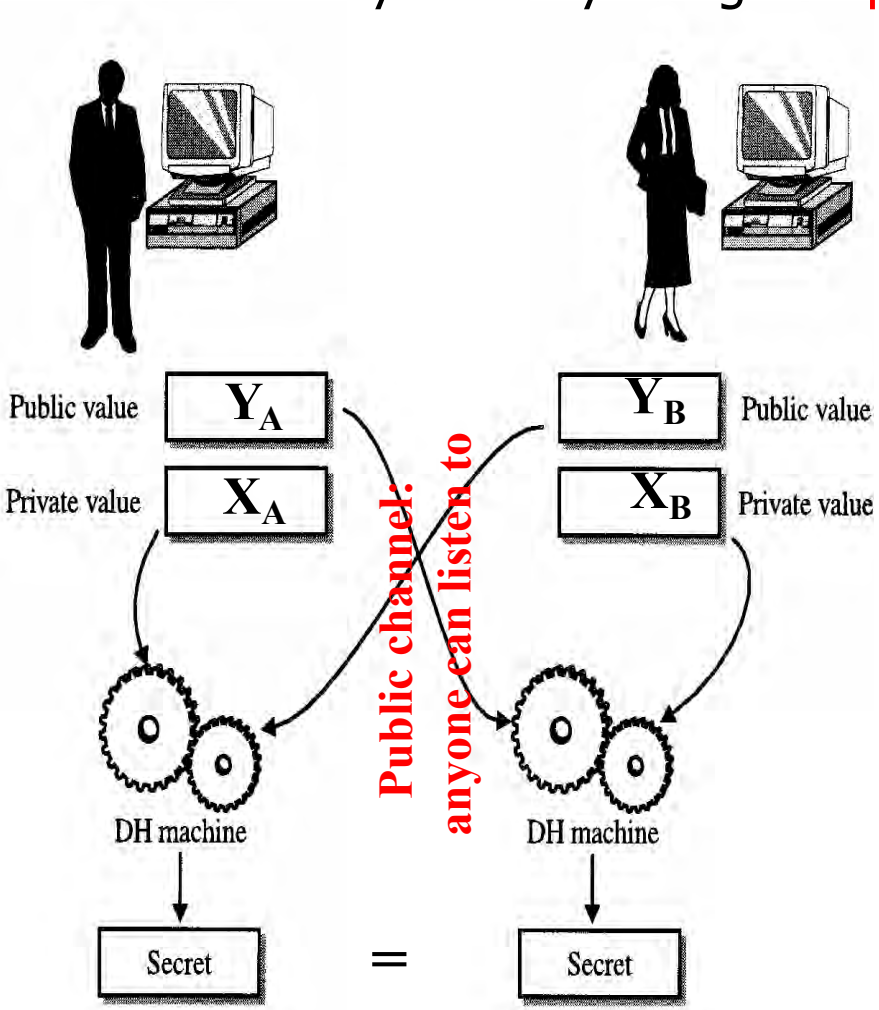
$$c = m^e \bmod n$$

$$m = c^d \bmod n$$

Public key is *(n,e)*. *Private* key is *(n,d)*.

Diffie-Hellman Key Exchange

- Diffie-Hellman key-exchange enables two users to establish a shared secret key securely using an open/ public communications channel.



Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation	
Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation	
Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

- $(Y_B)^{X_A} \bmod q = a^{X_B X_A} \bmod q = Secret = a^{X_A X_B} \bmod q = (Y_A)^{X_B} \bmod q$

How secure is Diffie-Hellman Key Exchange ?

- It relies on the fact that "Discrete Logarithm" is a computationally difficult problem, i.e.:

Knowing that $Y_A = a^{X_A} \bmod q$ and the values of a, q and Y_A

It is still computationally difficult to find X_A

- But still subject to Man-in-the-Middle Attack !! Because Alice does not know for sure if it's actually Bob who is sending her the Y_B
  - Remedy: Published those public numbers, i.e. *a*, q and Y_A , Y_B in a "Trusted, publicly accessible directory for each person"
    - This also allows Alice to send Bob an encrypted message even when he is currently offline.
  - But how can you be sure that you are looking at the directory hosted by the "true trusted directory server" ?

Man-in-the-middle (MITM) Attack

- DH protocol:

  1. Alice -> Bob: $\alpha^x \pmod q$
  2. Bob -> Alice: $\alpha^y \pmod q$

- Attack scenario

- Vulnerability: lack of what?

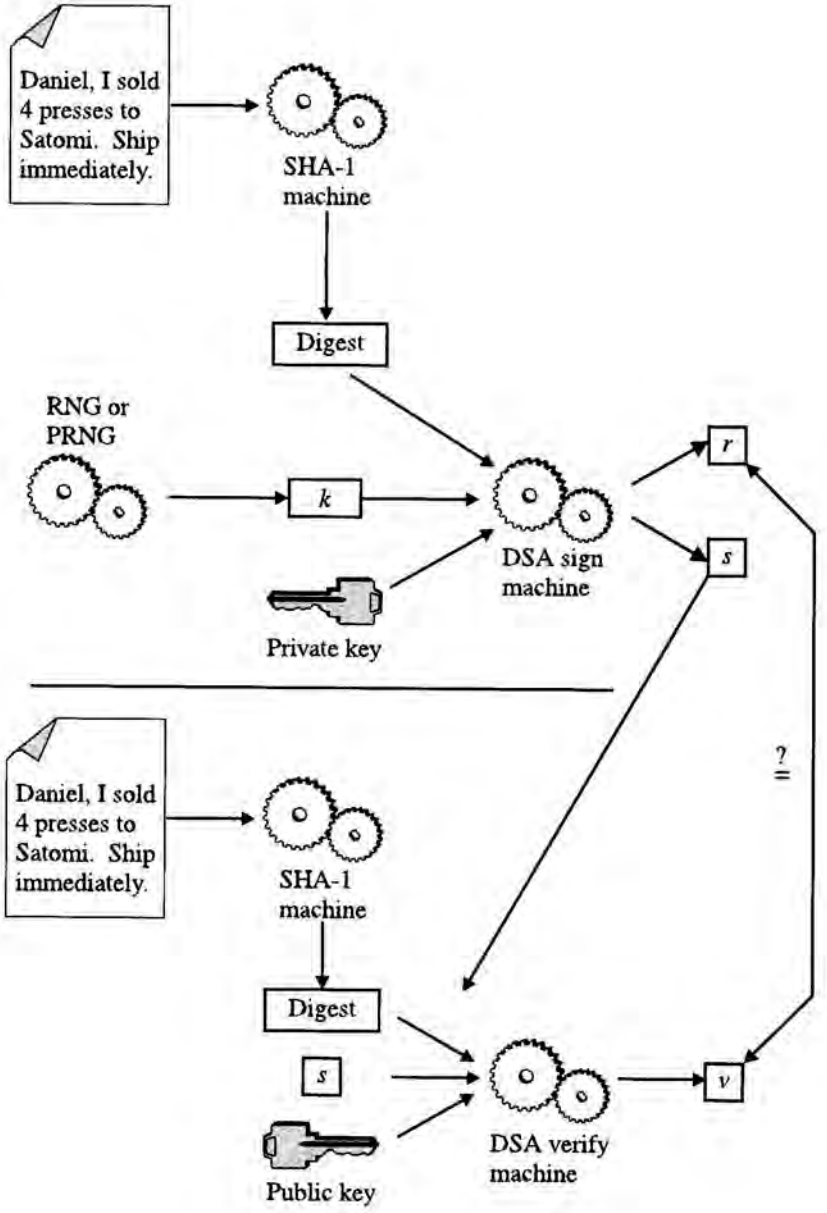
Other Public Key Algorithms

- 1978: Merkle/Hellman (Knapsack), subsequently found to be insecure
- 1985: El Gamal (Discrete logarithm Problem)
- 1985: Miller/Koblitz (Elliptic curves)
- 1991: Digital Signature Standard (DSS) (Discrete logarithm Problem)

And many others, too

Digital Signature Standard (DSS)

- In 1991, NIST in US standardized Digital Signature Standard (DSS). SHA-1 is used to first compute the message digest which is then signed by the Digital Signature Algorithm (DSA).
- DSA is based on a variant of El Gamal digital signature, thus also inherits it's "size-doubling" property => SHA-1 digest is 160-bit long, the DSA signature is 320 bits long: signature = (r,s).
- Since DSA does not support encryption by design, it avoids US technology-export concerns.



Elliptic Curve Cryptosystems (ECC)

- Independent proposed by Koblitz (U. of Washington) and Miller (IBM) in 1985
- Depends on the difficulty of the elliptic curve logarithm problem
  - fastest method is "Pollard rho method"
  - Best attacks for discrete logarithm problem do NOT apply to elliptic curve logarithm problem
- The first true alternative for RSA
- ECC is beginning to challenge RSA in practical deployment in selected areas: embedded, wireless/mobile systems
- It is a family of cryptosystems instead of a single one:
  - ECC replaces modulo exponentiation by elliptic curve multiplication (and modulo multiplication replaced by ECC addition)
  - Apply directly to Diffie-Hellman, El Gamal and DSA to yield ECC Diffie-Hellman (ECDH), ECC-ElGamal and ECC-DSA algorithms to support key exchange, encryption and digital signature respectively
- Certicom (http://www.certicom.com, a canadian-based company, is one of the leading companies for ECC commercialization

ECC Vs. RSA

ECC

- Shorter keys (equivalent key sizes: ~150bits Vs. 1024bits of RSA) and thus, shorter signature as well.
- Fast and compact implementations, especially in hardware

=> Advantageous in environments with limited bandwidth and storage, e.g. wireless applications, smartcards, embedded systems

- Shorter history of cryptanalysis (since early 90's)
- Complex mathematical description
- No patents for the cryptosystems themselves, but many on the implementation optimization
- Shorter signature generation time
- Shorter key generation time
- Larger no. of operations for attacks against the algorithm

RSA

- Proven technology,
- Widely deployed and used in a large set of general applications
- Efficient software implementation
- Longer history of cryptanalysis (since late 70's)
- Patent expired in 2000
- Shorter signature verification time
- Larger Memory requirements for attacks against the algorithm

ECC Vs. RSA (cont'd): Equivalent Key-size to support same level of Security

Elliptic Curve PKC	
Key Size	**MIPS-Years to Crack**
150	3.8×10^{10}
205	7.1×10^{18}
234	1.6×10^{28}

RSA PKC	
Key Size	**MIPS-Years to Crack**
512	3×10^{4}
768	2×10^{8}
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

Example:

Equivalent key-sizes given current acceptable security level of 4.12×10^{12} MIPS-year:

RSA : ECC : Symmetric cipher, (e.g. AES) = 1024:163:79

[Ref: 1GHz Pentium PC ~= 250 MIPS]

Relative Performance: ECC Vs. RSA (cont'd)

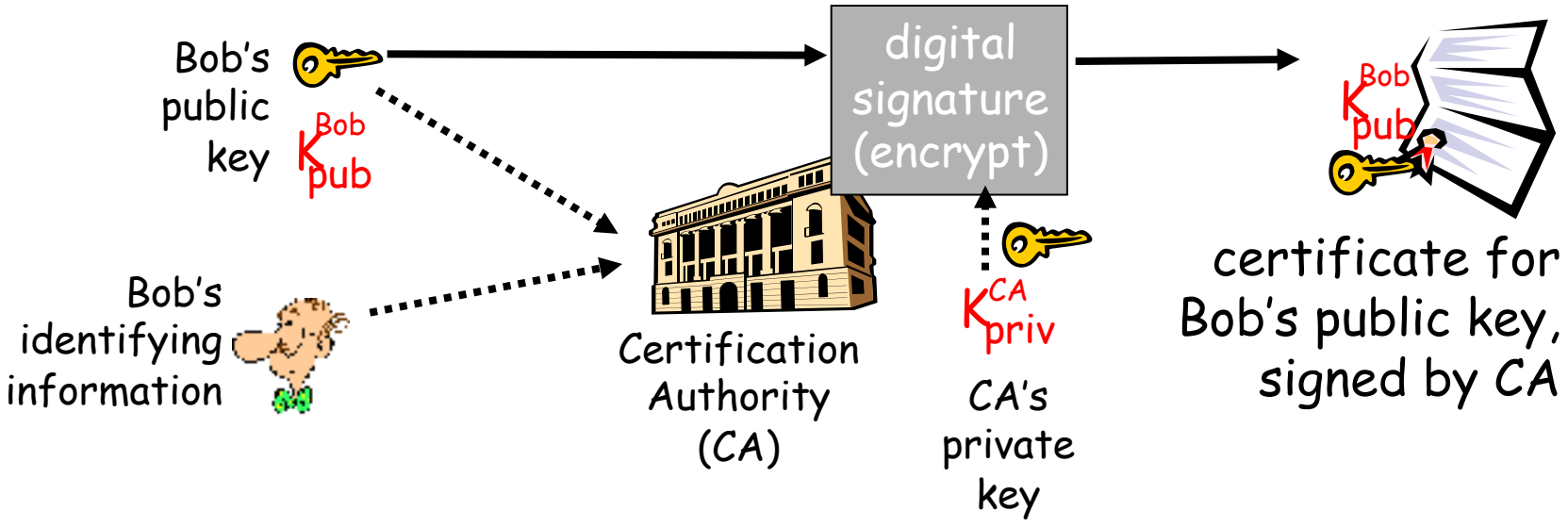
Estimated Relative Time units of Encryption/Decryption and/or Key-exchange (source: RSA)				
	RSA	DH	ECC	ECC with acceleration
Initiate contact (Public Key)	1	32	18	N/A
Receive message (Private Key)	13	16	6	2

Estimated Relative Time units of Digital signing and verification (source: RSA)				
	RSA	DSA	ECC	ECC with acceleration
Sign (Private Key)	13	17	7	2
Verify (Public Key)	1	33	19	N/A

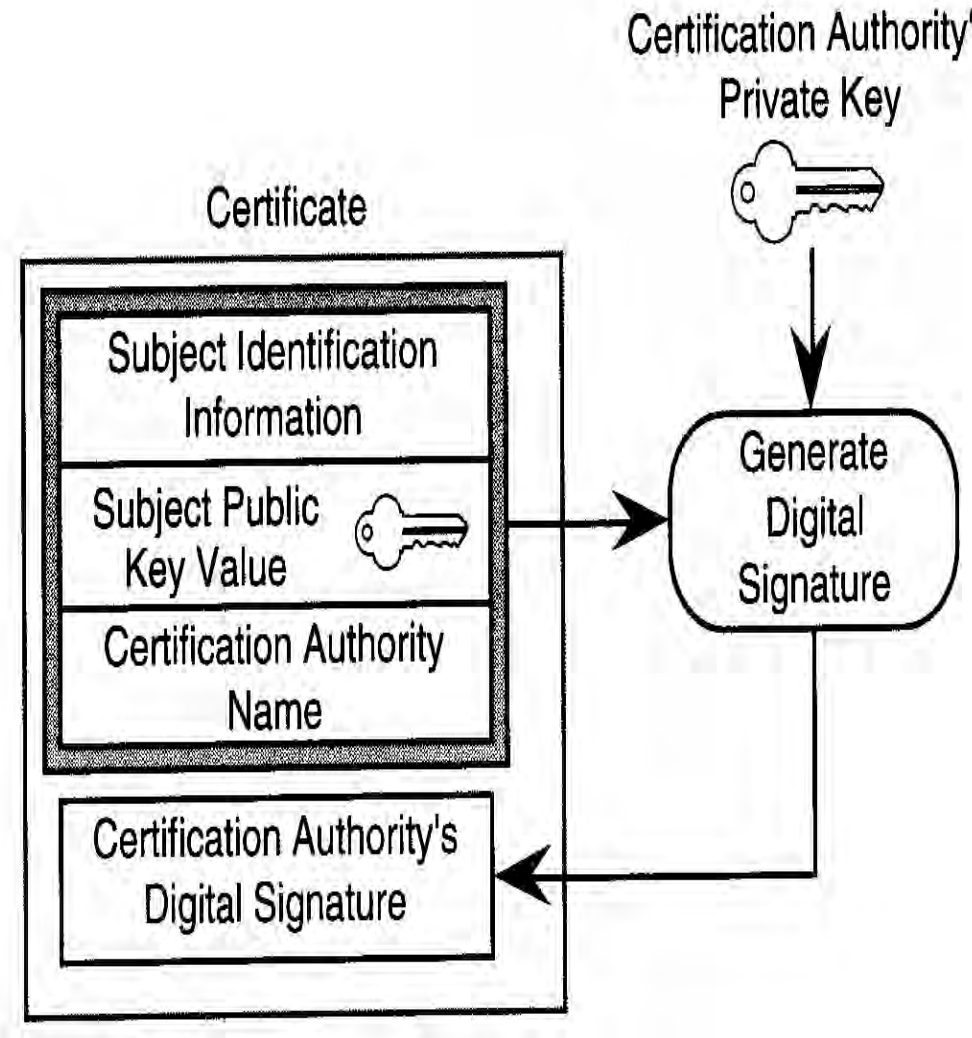
Digital Certificate and Public Key Infrastructure (PKI)

Certification Authorities

- Certification authority (CA): binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E's public key digitally signed by CA – CA says "this is E's public key"

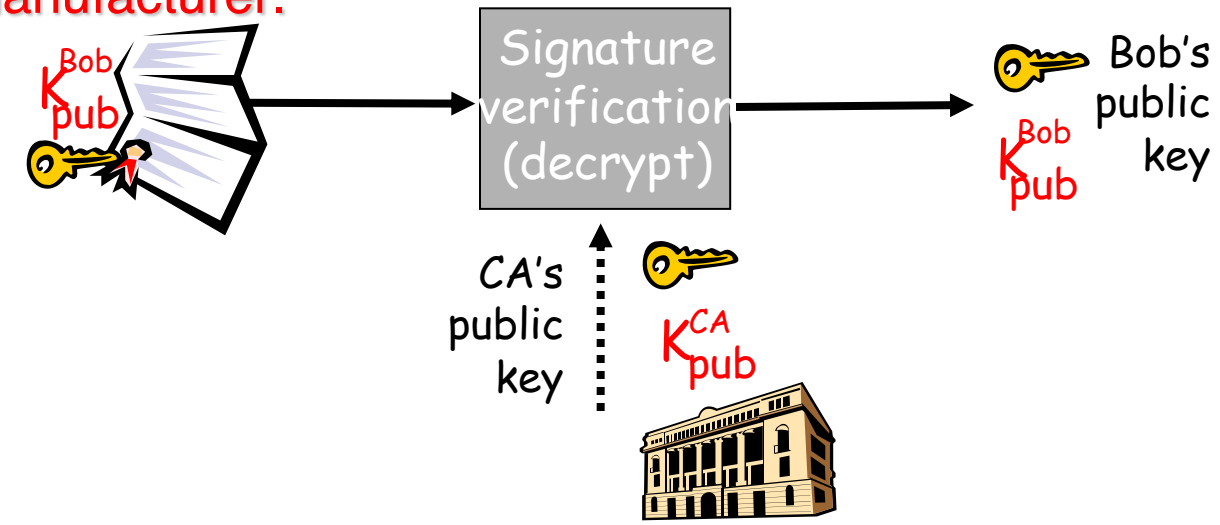


A Conceptual Digital Certificate

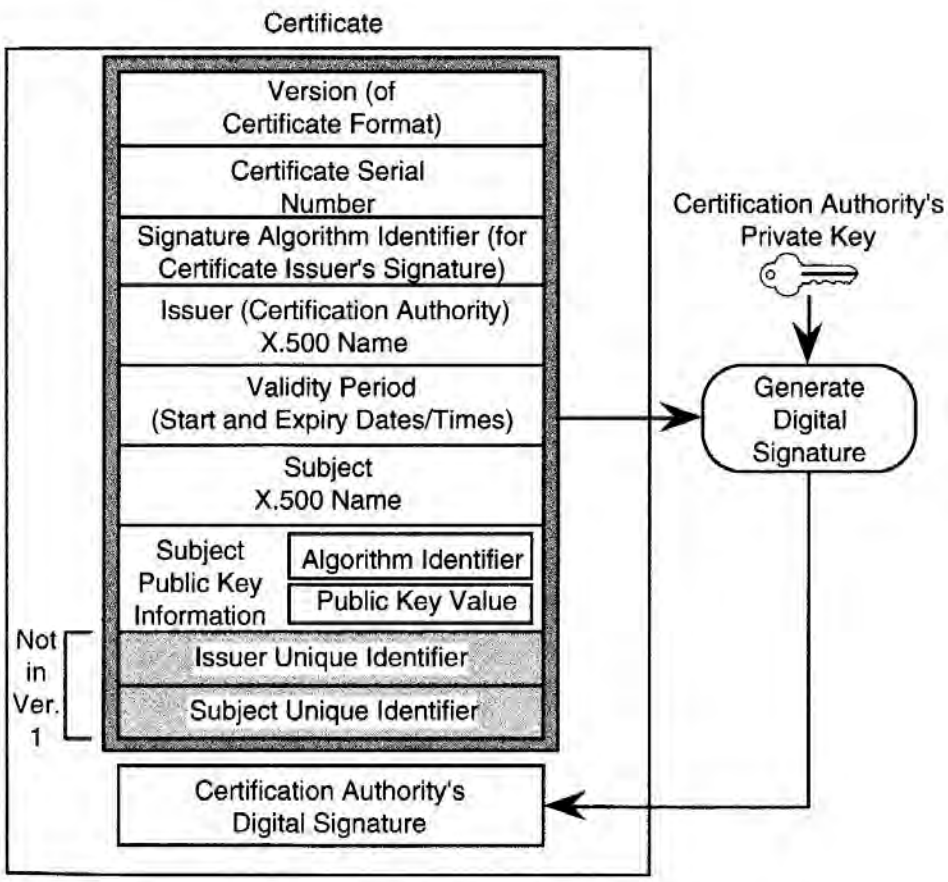


Certification Authorities

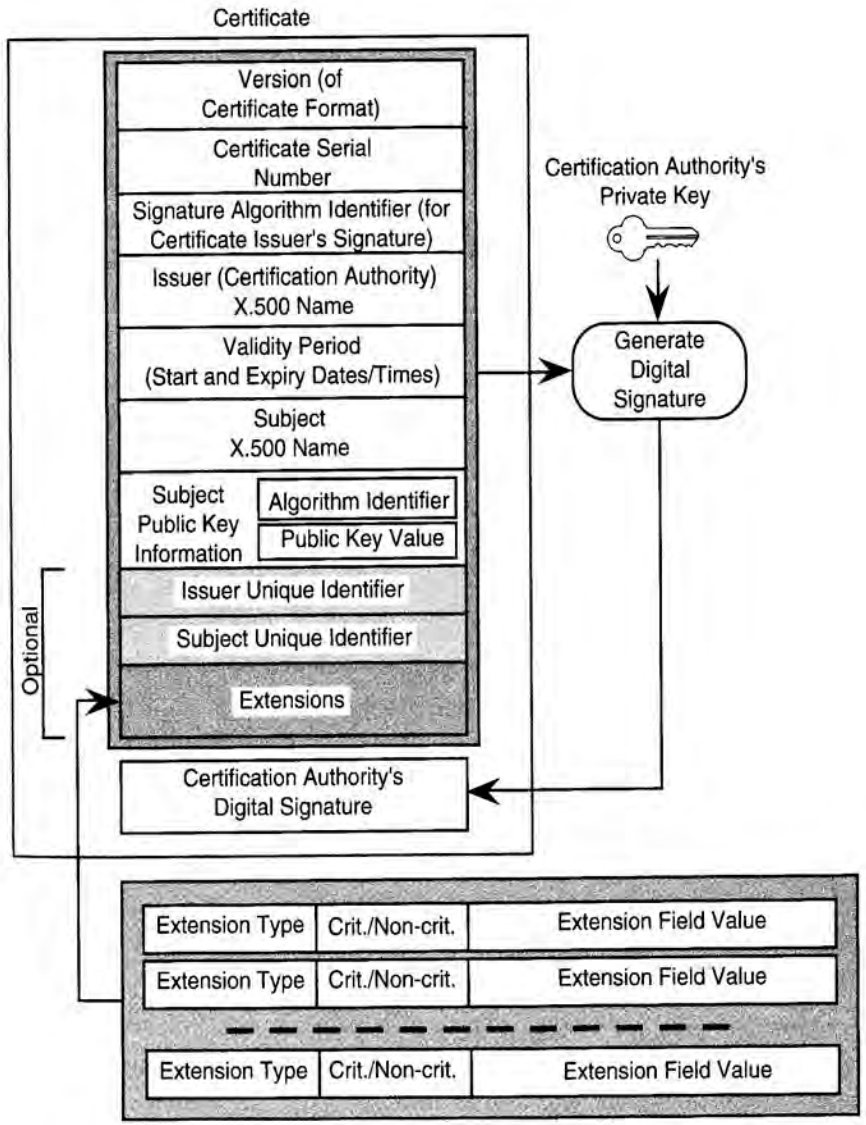
- When Alice wants Bob's public key:
  - gets Bob's certificate (from Bob or elsewhere).
  - apply CA's public key to verify Bob's certificate to confirm Bob's public key
  - Alice only needs to know the CA's public key in advance, e.g. preinstalled by computer/operating system manufacturer.



Versions 1 to 3 of an X.509 Digital Certificate

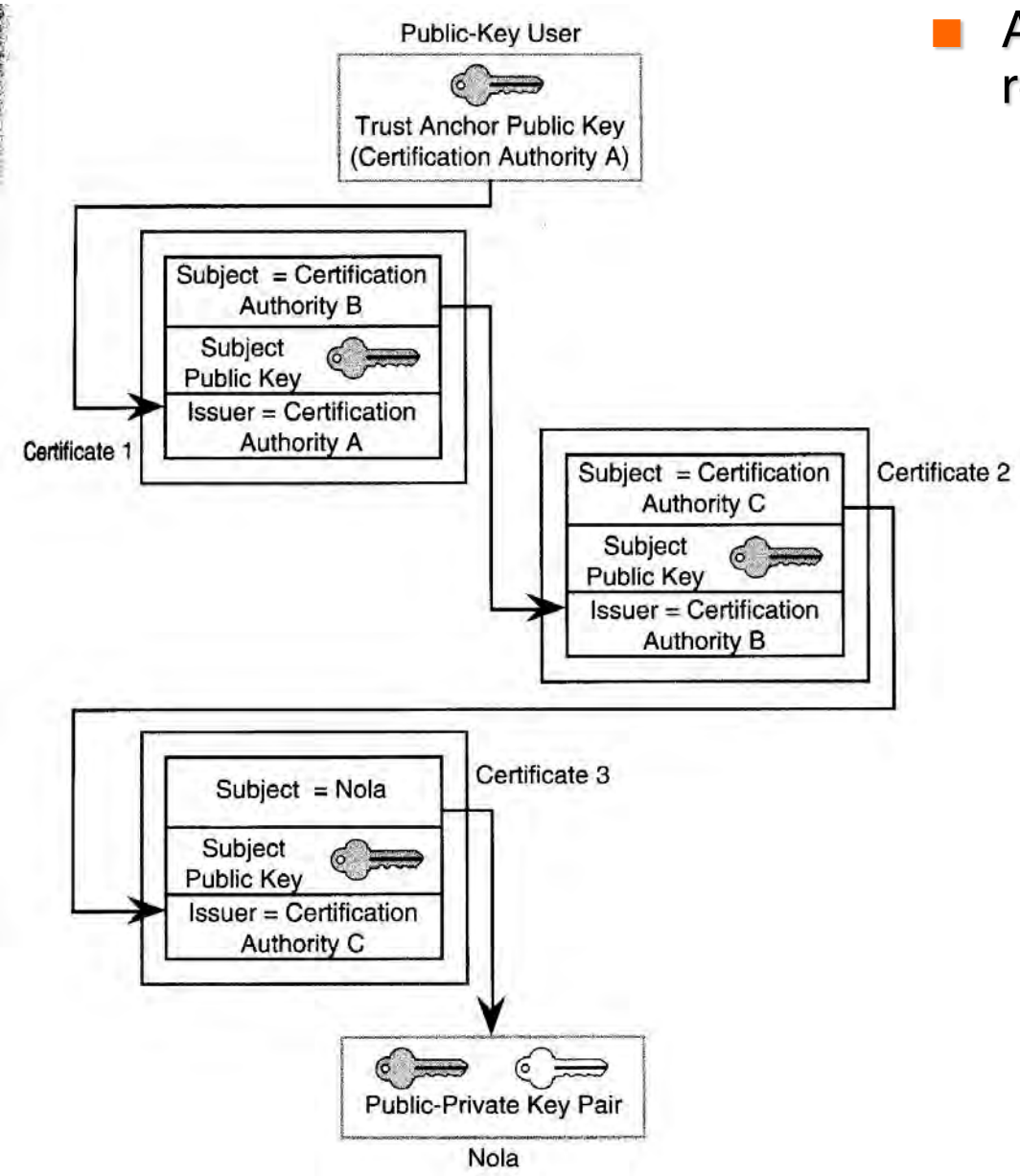


Version 1 and 2



Version 3

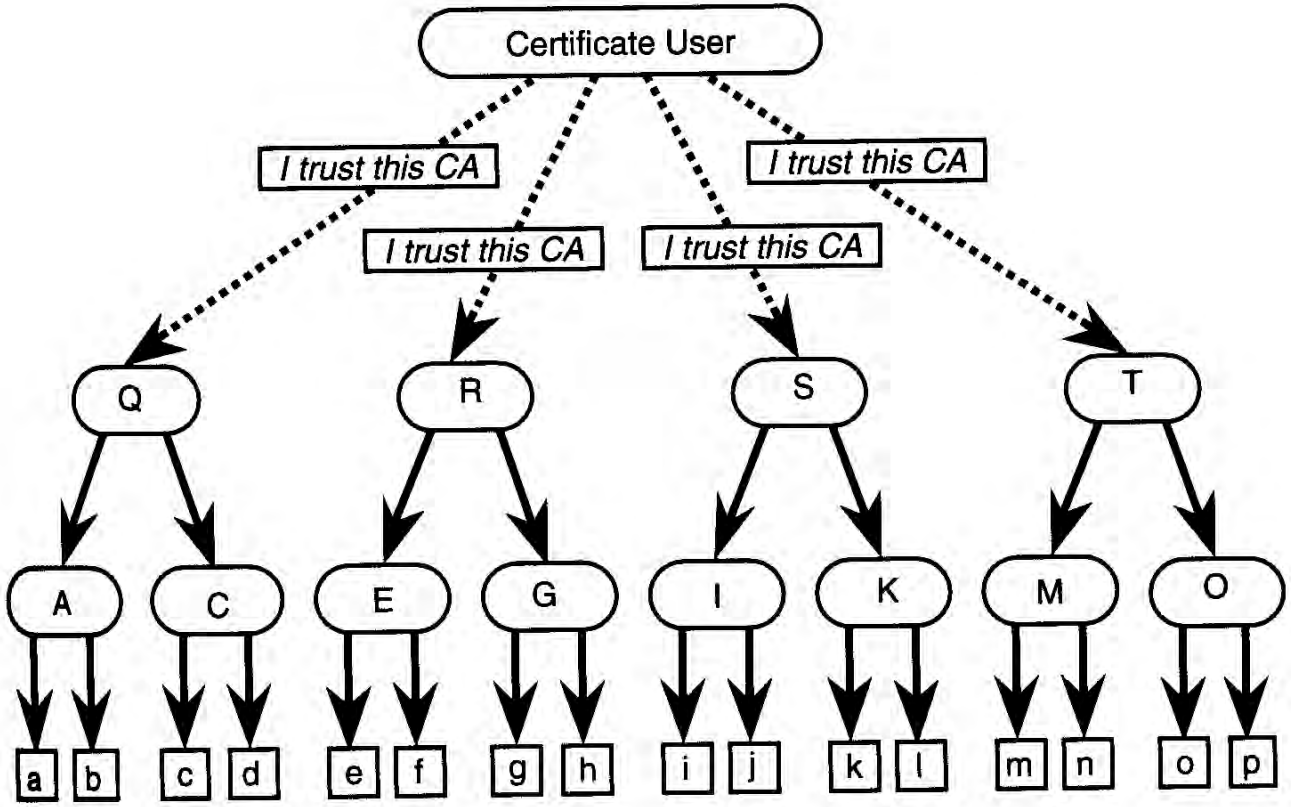
Certification Paths – chain of trust



- Apply the certificate paradigm recursively
  - At the beginning, a public-key user acquires, with high assurance, the public-keys of one or more CAs called the "Trust anchors" or "Root Certification Authorities",
  - e.g. Public keys of those trust anchors may be preconfigured in your browser.
  - The public-key user can accept any public key of a key-pair holder provided that a trusted certification path exists from a trust anchor of the public-key user to that key-pair holder possibly via other intermediate certification authorities

PKI Trust Hierarchy

- In practice, it is unrealistic to have an “universial” Root CA world wide, e.g. due to monopolistic, political concerns. So the trust hierarchy typically begins at a non-root level of the tree with multiple “root” trust anchors



Authentication

On the Internet, nobody knows you're a dog
- by Peter Steiner, New York, July 5, 1993

By 2006:
"On the Internet, EVERYBODY knows you're a dog, drinking Starbuck."

Authentication of People: To prove you are who you say you are

- By means of:
  - What you know ?

  e.g. password, your own HKID#
  - What you have ?

  e.g. A door-key, Secure token
  - Where you are ?

  e.g. caller-id, network (IP) address
  - Who you are ?

  e.g. fingerprint, other biometric: iris, retina patterns, voice
- Can combine more than one of the above, e.g. Automatic Teller Machine (ATM) use a 2-factor authentication scheme:
  - your ATM card + PIN

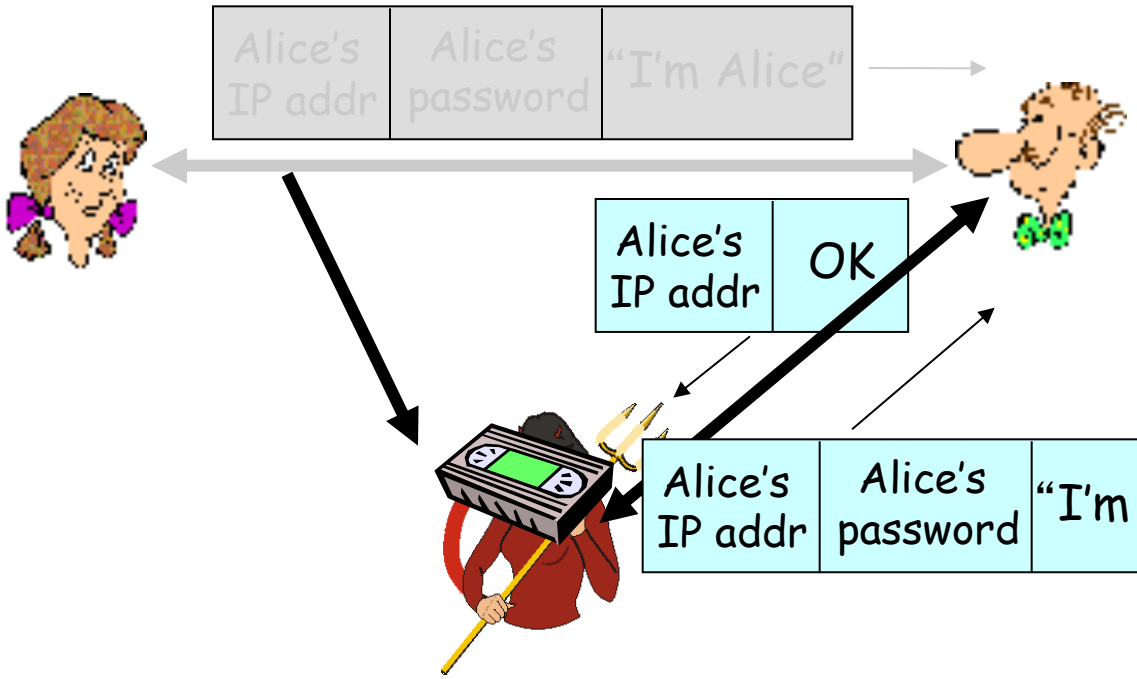
Design of Security Handshake for Authentication and Common Pitfalls

Threats of Concern

- Offline password cracking attacks
- Replay
- Security of Password Database at server Vs. sending password in clear across network
- Subsequent compromised of password to endanger previously encrypted (and recorded by the attacker) traffic
- Man in the Middle attack

Authentication: another try

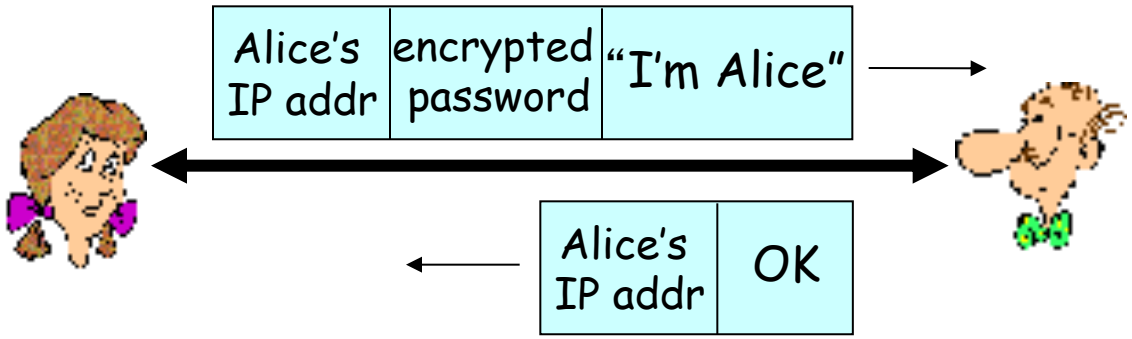
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



playback attack: Trudy records Alice's packet and later plays it back to Bob

Authentication: yet another try

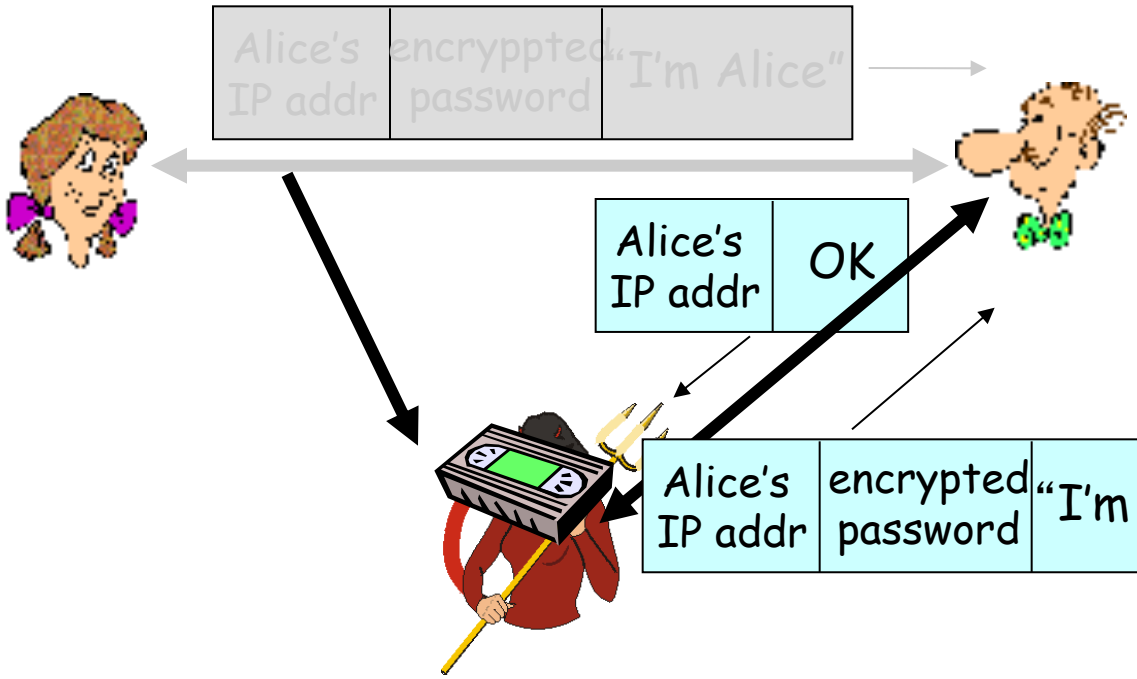
Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Failure scenario??

Authentication: another try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



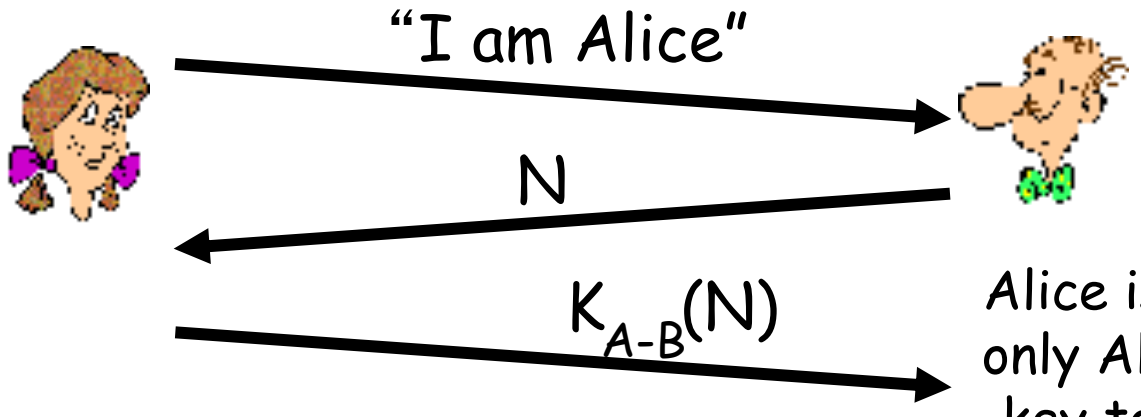
record and playback still works!

Authentication: yet another try

Goal: avoid playback attack

Nonce: number (N) used only *once –in-a-lifetime*

ap4.0: to prove Alice "live", Bob sends Alice nonce, N. Alice must return N, encrypted with shared secret key



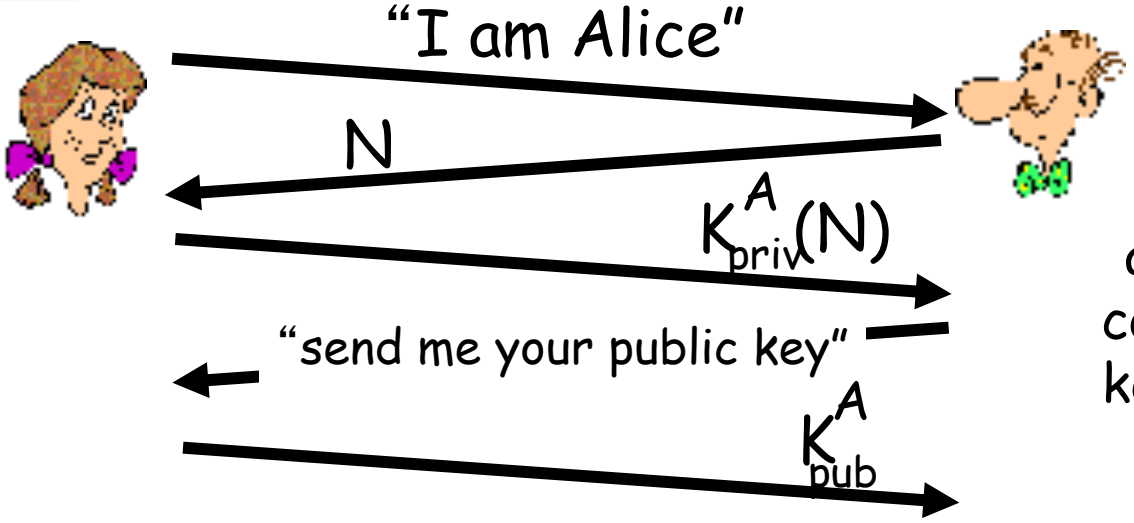
Drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key

- can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography

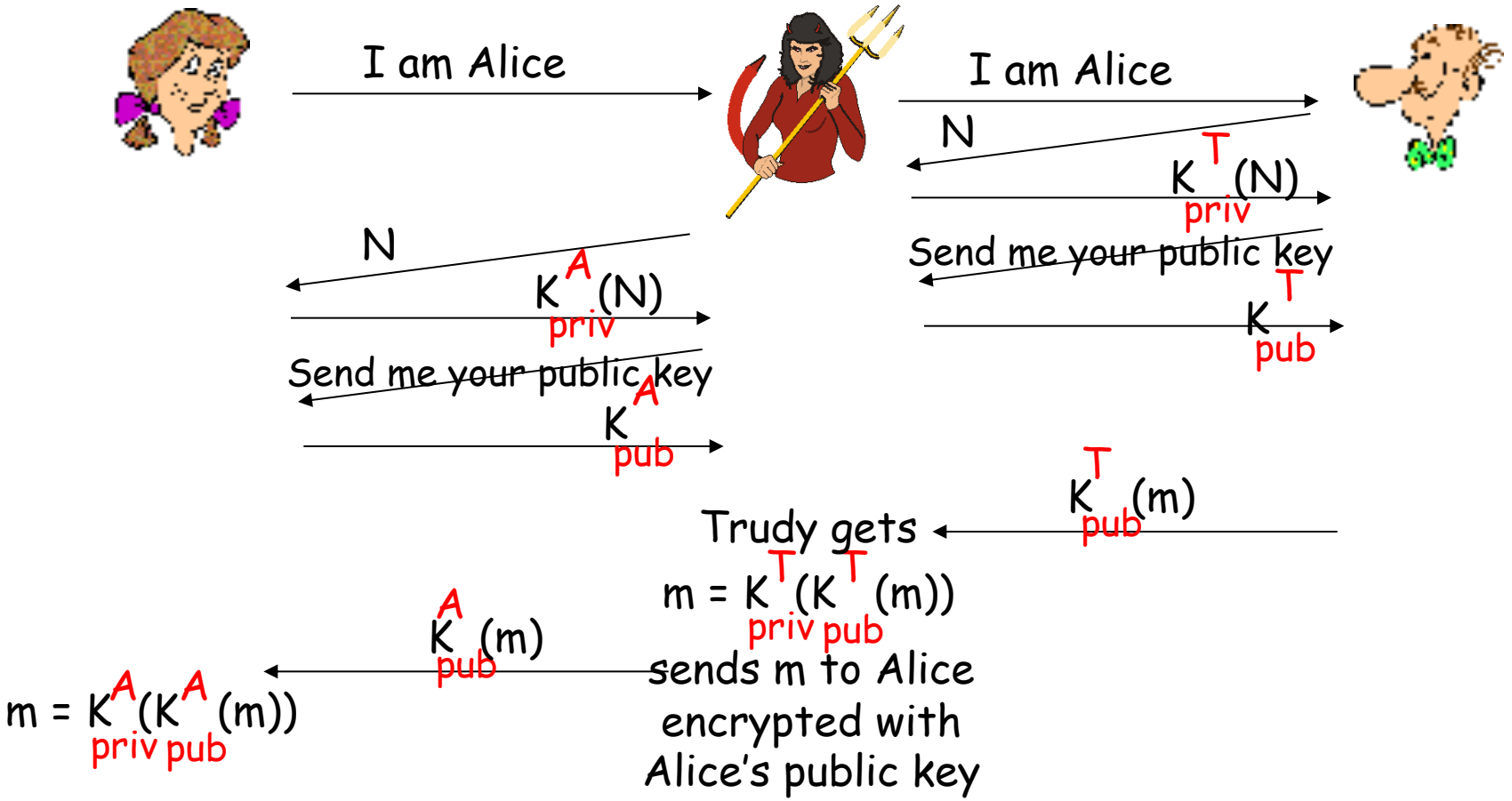


Bob computes $K^A_{pub}(K^A_{priv}(N)) = N$ and knows only Alice could have the private key, that encrypted N such that $K^A_{pub}(K^A_{priv}(N)) = N$

Also, what if "N" is some message (digest) that Alice does not want to sign
=> Each person uses multiple pairs of public/private keys ; one pair for encryption/decryption ; the other pair for authentication/signing

ap5.0: security hole

Man in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



Key Management

Key Management Problem: Getting Help from Trusted Intermediaries

Symmetric (Secret) key problem:

- How do two entities establish shared secret key over network?

Solution:

- trusted key distribution center (KDC) acting as intermediary between entities

Public key problem:

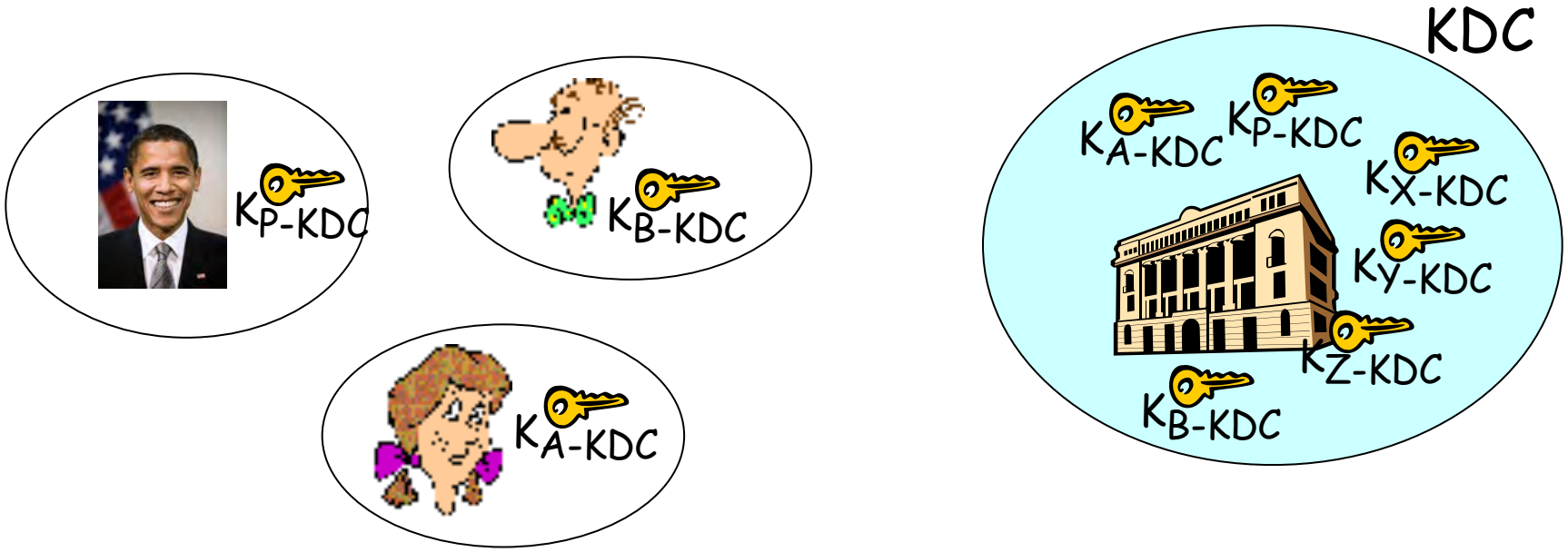
- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

- Digital Certificate Issued by trusted certification authority (CA)

Key Distribution Center (KDC)

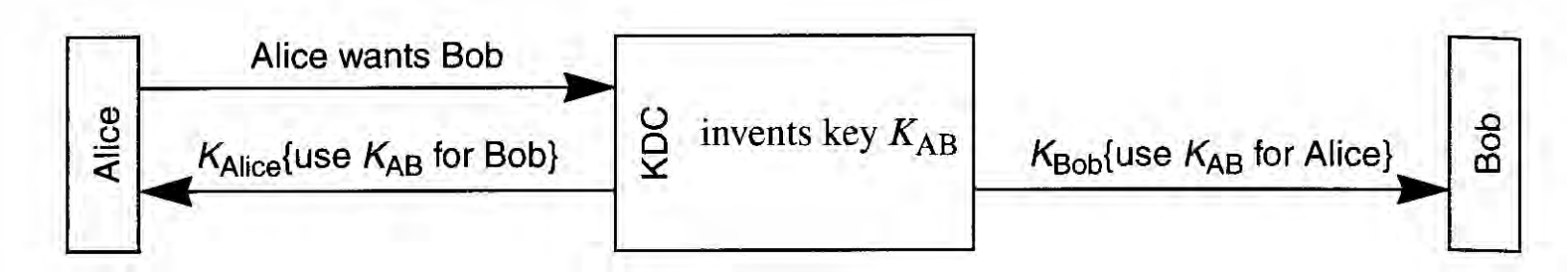
- Alice, Bob need a shared symmetric key.
- KDC: server shares different secret key with *each* registered user (many users)
- Alice, Bob know own symmetric keys, $K_{A\text{-}KDC}$ $K_{B\text{-}KDC}$, for communicating with KDC.



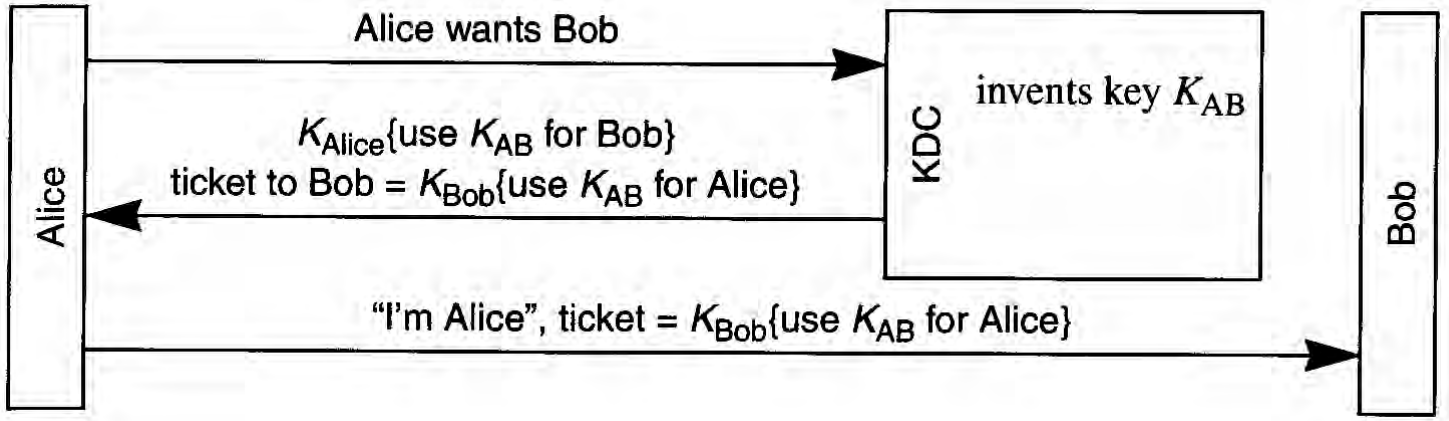
Key Distribution Center (KDC)

- Responsible for distributing keys to pairs of users (hosts, processes, applications)
- Each user must share a unique key, the *master* key, with the KDC
  - Use the master key to communicate with KDC to get a temporary *session* key for establishing a secure "session" with another user
  - Master keys are distributed in some *non-cryptographic* ways

KDC Operating Principles



a) Overall concept



b) First Refinement

Inter-working between multiple KDC domains