

Chapter 5: Hash Functions++

“I'm sure [my memory] only works one way.” Alice remarked.

“I can't remember things before they happen.”

“It's a poor sort of memory that only works backwards,”
the Queen remarked.

“What sort of things do you remember best?” Alice ventured to ask.

“Oh, things that happened the week after next,”
the Queen replied in a careless tone.

Lewis Carroll, *Through the Looking Glass*

Confidentiality in the Real World

Symmetric Key vs Public Key

□ *Symmetric key + 's*

- *Speed*
- *No public key infrastructure (PKI) needed (but have to generate/distribute keys)*

□ *Public Key + 's*

- *Signatures* (non-repudiation)
- *No **shared** secret (but, do have to get private keys to the right user...)*

Notation Reminder

□ Public key notation

- Sign M with Alice's *private key*

$$[M]_{\text{Alice}}$$

- Encrypt M with Alice's *public key*

$$\{M\}_{\text{Alice}}$$

□ Symmetric key notation

- Encrypt P with *symmetric key* K

$$C = E(P, K)$$

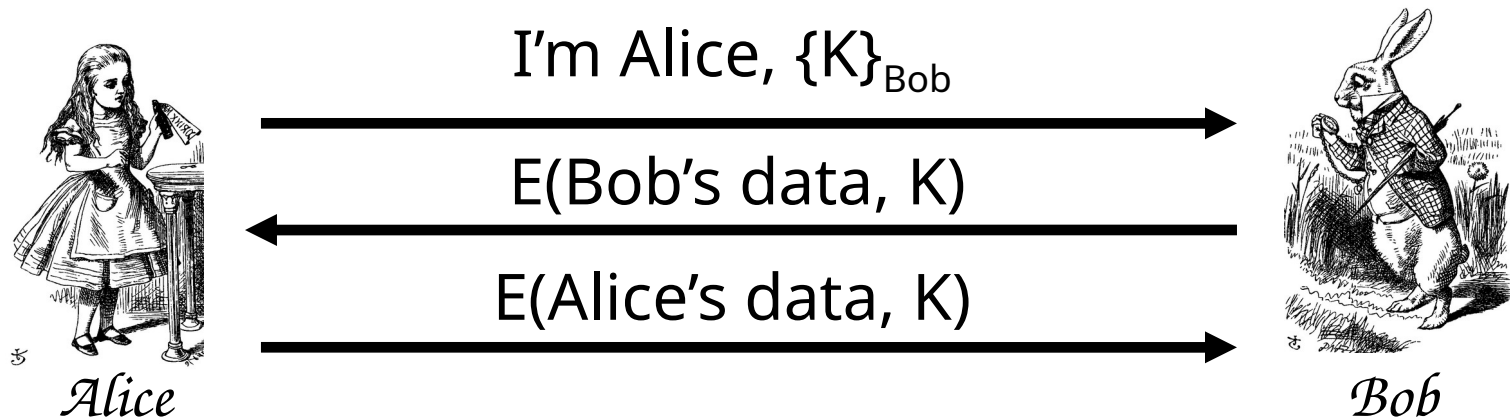
- Decrypt C with *symmetric key* K

$$P = D(C, K)$$

Real World Confidentiality

□ Hybrid cryptosystem

- Public key crypto to establish a key
- Symmetric key crypto to encrypt data...



□ Can Bob be sure he's talking to Alice?

Chapter 5: Hash Functions++

A boat, beneath a sunny sky
Lingering onward dreamily
In an evening of July
Children three that nestle near,
Eager eye and willing ear,

...

Lewis Carroll, *Through the Looking Glass*

Hash Function Motivation

- Suppose Alice signs M
 - Alice sends M and $S = [M]_{\text{Alice}}$ to Bob
 - Bob verifies that $M = \{S\}_{\text{Alice}}$
 - Can Alice just send S ?
- If M is big, $[M]_{\text{Alice}}$ costly to **compute & send**
- Suppose instead, Alice signs $h(M)$, where $h(M)$ is a much smaller “fingerprint” of M
 - Alice sends M and $S = [h(M)]_{\text{Alice}}$ to Bob
 - Bob verifies that $h(M) = \{S\}_{\text{Alice}}$

Hash Function Motivation

- ❑ *So, Alice signs $h(M)$*
 - *That is, Alice computes $S = [h(M)]_{\text{Alice}}$*
 - *Alice then sends (M, S) to Bob*
 - *Bob verifies that $h(M) = \{S\}_{\text{Alice}}$*
- ❑ *What properties must $h(M)$ satisfy?*
 - *Suppose Trudy finds M' so that $h(M) = h(M')$*
 - *Then Trudy can replace (M, S) with (M', S)*
- ❑ *Does Bob detect this tampering?*
 - *No, since $h(M') = h(M) = \{S\}_{\text{Alice}}$*

Crypto Hash Function

- *Crypto hash function $h(x)$ must provide*
 - *Compression* *output length is small*
 - *Efficiency* *$h(x)$ easy to compute for any x*
 - *One-way* *given a value y it is infeasible to find an x such that $h(x) = y$*
 - *Weak collision resistance* *given x and $h(x)$, infeasible to find y such that $h(y) = h(x)$*
 - *Strong collision resistance* *infeasible to find **any** x and y , with $x \neq y$ such that $h(x) = h(y)$*
- *Lots of collisions exist, but hard to find **any***

Hash functions

- A hash function maps a variable-length message into a fixed-length hash value, or message digest

- A *hash function* H accepts a variable-length block of data as input and produces a fixed-size hash value

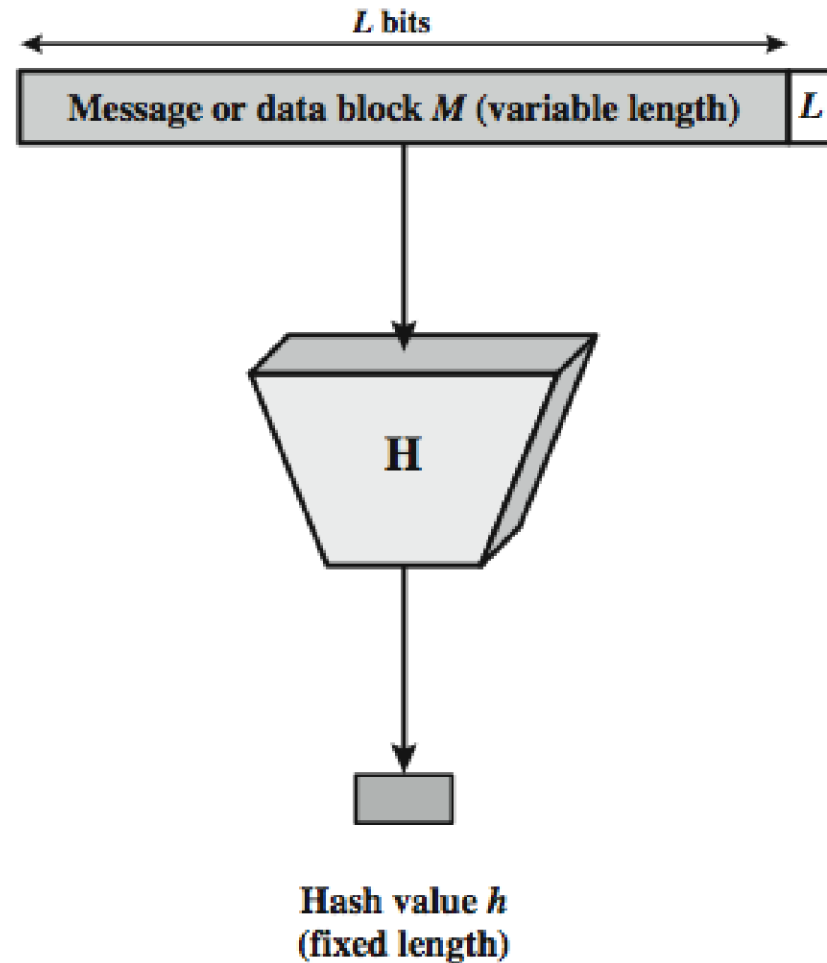
$$h = H(M)$$

- The principal object of a hash function is data integrity

Cryptographic Hash functions

- The kind of hash function needed for security applications is referred to as a cryptographic hash function.
- A cryptographic hash function is an algorithm for which it is computationally **infeasible**
- Because of these characteristics, hash functions are often used to determine whether or not data has changed

Cryptographic Hash functions



Pre-Birthday Problem

- *Suppose N people in a room*
- *How large must N be before the probability someone has same birthday as me is $1/2$?*
 - *Solve: $1/2 = 1 - (364/365)^N$ for N*
 - *We find $N = 253$*

Birthday Problem

- How many people must be in a room before probability is $1/2$ that any two (or more) have same birthday?
 - $1 \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \dots \cdot \frac{(365-N+1)}{365}$
 - Set equal to $1/2$ and solve: **$N = 23$**
- Surprising? A paradox?
- Maybe not: “Should be” about $\sqrt{365}$ since we compare all **pairs** X and Y
 - And there are 365 possible birthdays

Of Hashes and Birthdays

- If $h(x)$ is N bits, then 2^N different hash values are possible
- So, if you hash about $\text{sqrt}(2^N) = 2^{N/2}$ values then you expect to find a collision
- **Implication?** “Exhaustive search” attack...
 - Secure N -bit hash requires $2^{N/2}$ work to “break”
 - Recall that secure N -bit symmetric cipher has work factor of 2^N
- Hash output length vs cipher key length?

Non-crypto Hash (1)

- ❑ *Data $X = (X_1, X_2, X_3, \dots, X_n)$, each X_i is a byte*
- ❑ *Define $h(X) = (X_1 + X_2 + X_3 + \dots + X_n) \bmod 256$*
- ❑ *Is this a secure cryptographic hash?*
- ❑ *Example: $X = (10101010, 00001111)$*
- ❑ *Hash is $h(X) = 10111001$*
- ❑ *If $Y = (00001111, 10101010)$ then $h(X) = h(Y)$*
- ❑ *Easy to find collisions, so **not** secure...*

Non-crypto Hash (2)

□ Data $X = (X_0, X_1, X_2, \dots, X_{n-1})$

□ Suppose hash is defined as

$$h(X) = (nX_1 + (n-1)X_2 + (n-2)X_3 + \dots + 2X_{n-1} + X_n) \bmod 256$$

□ Is this a secure cryptographic hash?

□ Note that

$$h(10101010, 00001111) = h(00001111, 10101010)$$

□ But hash of $(000000001, 00001111)$ is same as hash of $(00000000, 00010001)$

□ Not “secure”, but this hash is used in the (non-crypto) application rsync

Non-crypto Hash (3)

- ❑ *Cyclic Redundancy Check (CRC)*
- ❑ *Essentially, CRC is the remainder in a long division calculation*
- ❑ *Good for detecting burst **errors***
 - *Such random errors unlikely to yield a collision*
- ❑ *But easy to **construct** collisions*
 - *In crypto, Trudy is the enemy, not “random”*
- ❑ *CRC has been mistakenly used where crypto integrity check is required (e.g., WEP)*

Popular Crypto Hashes

- ❑ **MD5** *invented by Rivest (of course...)*
 - 128 bit output
 - MD5 collisions easy to find, so it's broken
- ❑ **SHA family** *A U.S. government standard, inner workings similar to MD5*
 - SHA-1 160 bit output
- ❑ *Many other hashes, but MD5 and SHA-1 are the most widely used*
- ❑ *Hashes work by hashing message in blocks*

Crypto Hash Design

- ❑ *Desired property: **avalanche effect***
 - *Change to 1 bit of input should affect about half of output bits*
- ❑ *Crypto hash functions consist of some number of rounds*
- ❑ *Want security and speed*
 - *“Avalanche effect” after few rounds*
 - *But simple rounds*
- ❑ *Analogous to design of block ciphers*

Cryptographic hash functions

- *When security people talk about hash functions, they mean cryptographic (or secure) hash functions*
- *These should provide*
 - *Collision resistance*
 - *Difficult to find any $\mathcal{M}, \mathcal{M}' \neq \mathcal{M}$ s.t. $h(\mathcal{M}) = h(\mathcal{M}')$*
 - *Preimage resistance*
 - *Given $h(\mathcal{M})$, difficult to find \mathcal{M}' s.t. $h(\mathcal{M}') = h(\mathcal{M})$*
 - *Second preimage resistance*
 - *Given \mathcal{M} , difficult to find \mathcal{M}' s.t. $h(\mathcal{M}') = h(\mathcal{M}), \mathcal{M}' \neq \mathcal{M}$*
- *If a hash function h does not meet these*



FAIL!

But what does it all mean?

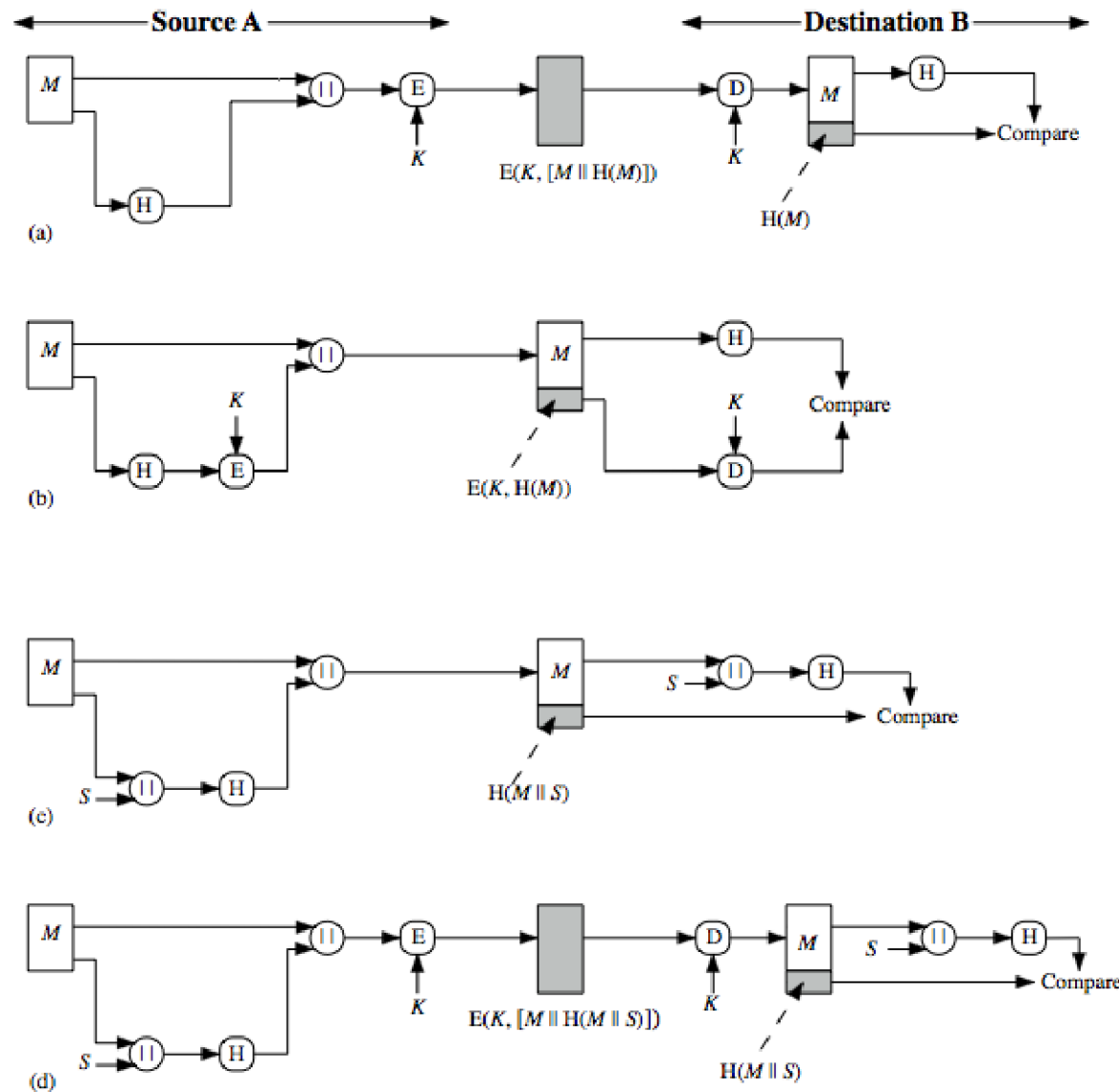
- *If h is secure*
 - *Easy to compute in one direction*
 - *Very difficult to compute in the other direction*
 - *Computationally infeasible*
 - *i.e. your grandchildren's grandchildren's grandchildren will be long gone before that computation finishes*
 - *Very difficult to find two messages that hash to the same value*

- *Can anyone name any?*

Message Authentication

- Message authentication is a mechanism or service used to verify the *integrity of a message*.
- Message authentication assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay).
- When a hash function is used to provide message authentication, the hash function value is often referred to as a message digest.

Hash Functions & Msg Authentication



Message Authentication – Picture a)

- The message plus concatenated hash code is encrypted using **symmetric encryption**.
- Because only A and B **share the secret key**, the message must have come from A and has not been altered.
- The hash code provides the structure or redundancy required to achieve authentication.
- Because encryption is applied to the entire message plus hash code, **confidentiality is also provided**

Message Authentication – Picture b)

- Only the hash code is encrypted, using symmetric encryption.
- This reduces the processing burden for those applications that do not require confidentiality

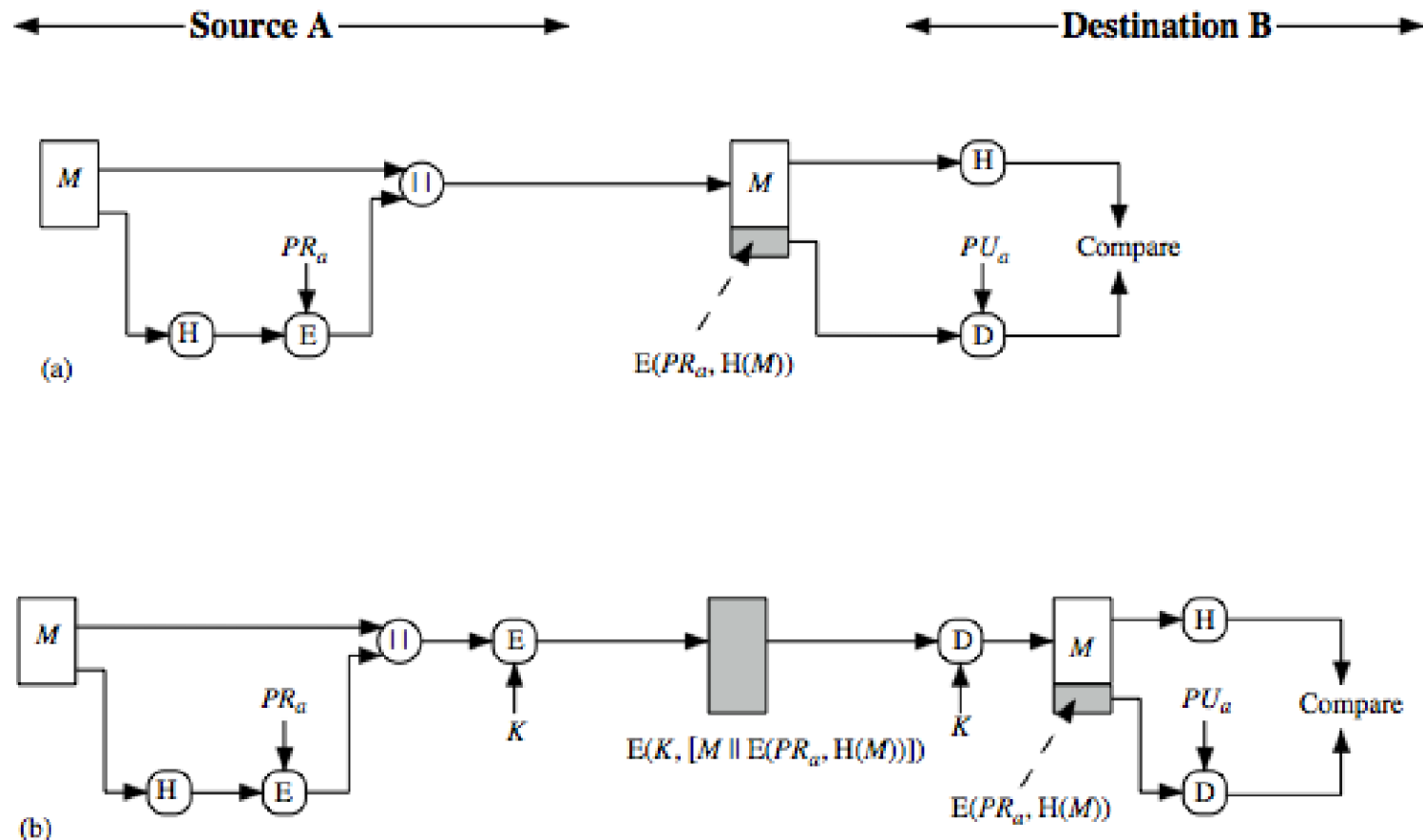
Message Authentication – Picture c)

- It is possible to use a hash function but no encryption for message authentication.
- The technique assumes that the two communicating parties share a common secret value S .
- A computes the hash value over the concatenation of M and S and appends the resulting hash value to.
- Because B possesses, it can recompute the hash value to verify.
- Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.

Message Authentication – Picture d)

- *Confidentiality can be added* to the approach of method (c) by encrypting the entire message plus the hash code

Hash Functions & Digital Signatures



Hash Functions & Dig. Signatures – a)

- The hash code is encrypted, using public-key encryption with the sender's private key.
- It also provides a digital signature, because only the sender could have produced the encrypted hash code.
- In fact, this is the essence of the digital signature technique.

Hash Functions & Dig. Signatures – b)

- If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key.

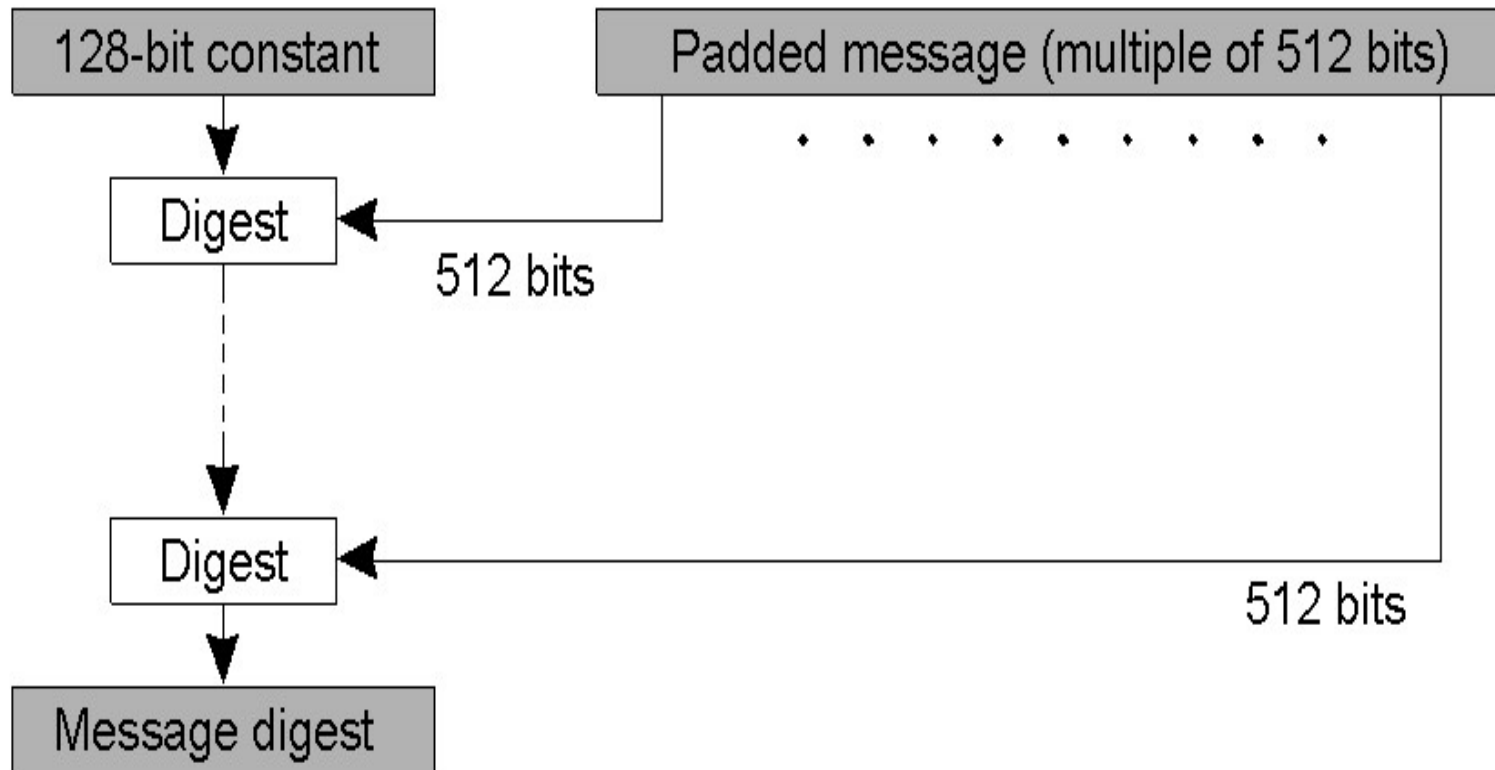
Other Hash Functions Uses

- Hash functions are commonly used to create a one-way password file.
 - Thus, the actual password is not retrievable by a hacker who gains access to the password file.
 - This approach to password protection is used by most operating systems.
- Hash functions can be used for intrusion detection and virus detection.
 - Store $H(F)$ for each file on a system and secure the hash values (e.g., on a CD-R that is kept secure).
 - One can later determine if a file has been modified by recomputing $H(F)$.
 - An intruder would need to change F without changing $H(F)$.
- Can be used to construct a pseudorandom function (PRF) or a pseudorandom number generator (PRNG).

MD5

- *MD5 algorithm was developed by Professor Ronald L. Rivest in 1991. According to RFC 1321, “MD5 message-digest algorithm takes as input a message of arbitrary length and produces as output a 128-bit “fingerprint” or “message digest” of the input ... The MD5 algorithm is intended for digital signature applications, where a large file must be “compressed” in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA.”*

MD5 Algorithm Structure



Implementation Steps

□ Step1 Append padding bits

The input message is "padded" (extended) so that its length (in bits) equals to $448 \bmod 512$. Padding is always performed, even if the length of the message is already $448 \bmod 512$.

Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to $448 \bmod 512$. At least one bit and at most 512 bits are appended.

Implementation Steps

□ Step2. Append length

A 64-bit representation of the length of the message is appended to the result of step1. If the length of the message is greater than 2^{64} , only the low-order 64 bits will be used.

The resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits. The input message will have a length that is an exact multiple of 16 (32-bit) words.

Implementation Steps

□ Step3. Initialize MD buffer

A four-word buffer (\mathcal{A} , \mathcal{B} , \mathcal{C} , \mathcal{D}) is used to compute the message digest. Each of \mathcal{A} , \mathcal{B} , \mathcal{C} , \mathcal{D} is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first):

word \mathcal{A} : 01 23 45 67

word \mathcal{B} : 89 ab cd ef

word \mathcal{C} : fe dc ba 98

word \mathcal{D} : 76 54 32 10

Implementation Steps

- *Step4. Process message in 16-word blocks*
Four functions will be defined such that each function takes an input of three 32-bit words and produces a 32-bit word output.

$$\mathcal{F}(X, Y, Z) = XY \text{ or } \text{not}(X) Z$$

$$\mathcal{G}(X, Y, Z) = XZ \text{ or } Y \text{ not}(Z)$$

$$\mathcal{H}(X, Y, Z) = X \text{ xor } Y \text{ xor } Z$$

$$\mathcal{I}(X, Y, Z) = Y \text{ xor } (X \text{ or } \text{not}(Z))$$

Implementation Steps

Round 1.

$[abcd\ k\ s\ i]$ denote the operation $a = b + ((a + \mathcal{F}(b, c, d) + X[k] + T[i])) \lll s$.

Do the following 16 operations.

$[ABCD\ 0\ 7\ 1]\ [DABC\ 1\ 12\ 2]\ [CDAB\ 2\ 17\ 3]\ [BCDA\ 3\ 22\ 4]$
 $[ABCD\ 4\ 7\ 5]\ [DABC\ 5\ 12\ 6]\ [CDAB\ 6\ 17\ 7]\ [BCDA\ 7\ 22\ 8]$
 $[ABCD\ 8\ 7\ 9]\ [DABC\ 9\ 12\ 10]\ [CDAB\ 10\ 17\ 11]\ [BCDA\ 11\ 22\ 12]$
 $[ABCD\ 12\ 7\ 13]\ [DABC\ 13\ 12\ 14]\ [CDAB\ 14\ 17\ 15]\ [BCDA\ 15\ 22\ 16]$

Performance

	<i>Key size/hash size(bits)</i>	<i>Extrapolated Speed (Kbytes/sec.)</i>	<i>PRB Optimized (Kbytes/sec.)</i>
TEA	128	700	-
DES	56	350	7746
Triple-DES	112	120	2842
IDEA	128	700	4469
RSA	512	7	-
SHA	160	750	25162
MD5	128	1740	62425

Block Cipher as Hash Functions

- A number of proposals have been made for hash functions based on using a cipher block chaining technique, but without using the secret key.
- Divide a message M into fixed-size blocks M_1, M_2, \dots, M_N and use a symmetric encryption system such as DES to compute the has
 - H_0 = initial value
 - $H_i = E(M_i, H_{i-1})$
 - $G = H_N$
- use final block as the hash value

Secure Hash Functions (SHA)

- **SHA originally designed by NIST & NSA in 1993**
- **was revised in 1995 as SHA-1**
- **US standard for use with DSA signature scheme**
 - standard is FIPS 180-1 1995, also Internet RFC3174
 - Note that, the algorithm is SHA, the standard is SHS
- **based on design of MD4 with key differences**
- **produces 160-bit hash values**
- **recent 2005 results on security of SHA-1 have raised concerns on its use in future applications**

Revised Secure Hash Standard

- NIST issued revision FIPS 180-2 in 2002
- adds 3 additional versions of SHA
 - SHA-256, SHA-384, SHA-512
- designed for compatibility with increased security provided by the AES cipher
- structure & detail is similar to SHA-1
- hence analysis should be similar
- but security levels are rather higher

Secure Hash Algorithm (SHA)

- ❑ *NIST standards*
 - *Mandatory in US Government*
 - *Adopted globally*
- ❑ *SHA (SHA-0) is no good anymore*
- ❑ *SHA-1 has attacks and is not recommended*
- ❑ *SHA-2 looks good for now*
 - *What happens when there's an attack?*
 - *It takes years to create and analyze functions*

SHA-3

- ❑ *About halfway through the process of choosing the next SHA family of hash functions*
- ❑ *International competition*
 - *64 submissions*
 - *Round 1: 54*
 - *Round 2: 14*
 - *Round 3: ~5*
 - *And the winner is... ?*
- ❑ *Winner gets massive bragging rights*
- ❑ *A lot of new design techniques*
- ❑ *A lot of new attack techniques*

SHA Versions

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message digest size	160	224	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	512	1024	1024
Word size	32	32	32	64	64
Number of steps	80	64	64	80	80

Who can compute a hash?

- ❑ *A hash is a keyless algorithm*
- ❑ *Anyone can compute $h(x)$ if they know x*
- ❑ *Eve could replace \mathcal{M} with \mathcal{M}' and $h(\mathcal{M})$ with $h(\mathcal{M}')$*
 - *The hash matches what Bob computes, so he assumes that Alice sent him \mathcal{M}'*
- ❑ *How could we stop Eve from doing this?*

HMAC

- *Hash-based Message Authentication Code*
- *Keyed hash*
 - $y = \text{HMAC}(\mathcal{M}, k)$
- *Provides some level of authentication*
 - *If only Alice and Bob know the key and the HMAC is correct, it must have come from one of them*
- *Can make an HMAC algorithm from an unkeyed hash algorithm*

- *Why not just make a keyed hash algorithm?*
 - *Import/export restrictions*
 - *Keyless algorithms are not restricted*

Attacks on Hash Functions

- Brute-Force attacks
 - Preimage and second preimage attacks
 - Collision resistant attacks
- Cryptanalysis attacks

Brute-Force Attacks

- A brute-force attack does not depend on the specific algorithm but depends only on bit length.
- In the case of a hash function, a brute-force attack depends only on the bit length of the hash value.
- A cryptanalysis, in contrast, is an attack based on weaknesses in a particular cryptographic algorithm.

Preimage & Second Preimage Attacks

- For a preimage or second preimage attack, an adversary wishes to find a value such that $H(y)$ is equal to a given hash value.
- The brute-force method is to pick values of y at random and try each value until a collision occurs.
- For an m -bit hash value, the level of effort is proportional to 2^m
- Specifically, the adversary would have to try, on average, 2^{m-1} values of y to find one that generates a given hash value h .

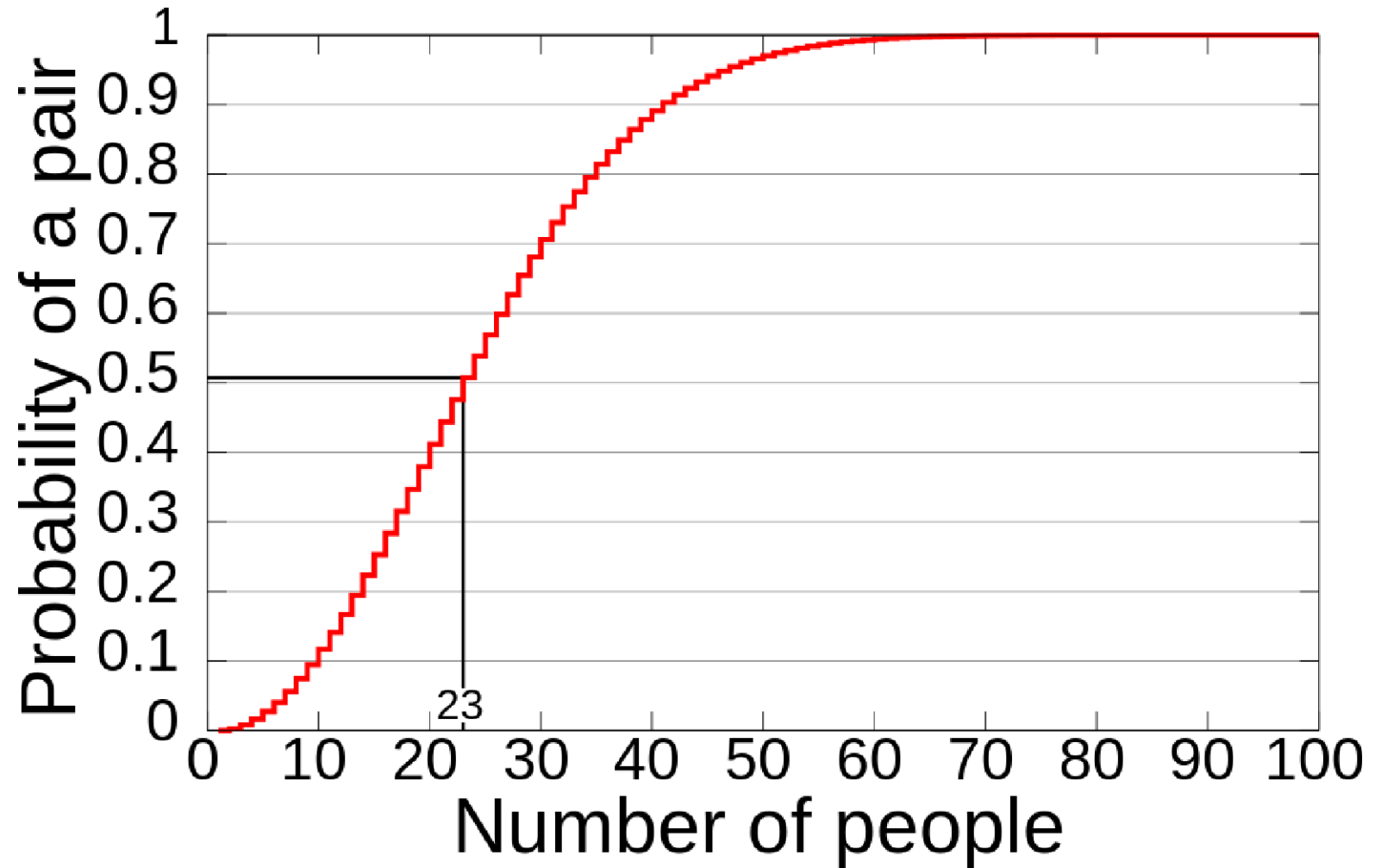
Collision Resistant Attacks

- For a collision resistant attack, an adversary wishes to find two messages or data blocks, x and y , that yield the same hash function: $H(x) = H(y)$.
- In essence, if we choose random variables from a uniform distribution in the range 0 through $N - 1$, then the probability that a repeated element is encountered exceeds 0.5 after $N^{1/2}$ choices have been made
- Thus, for an **m -bit** hash value, if we pick data blocks at random, we can expect to find two data blocks with the same hash value **within $2^{m/2}$ attempts**

Birthday Attacks

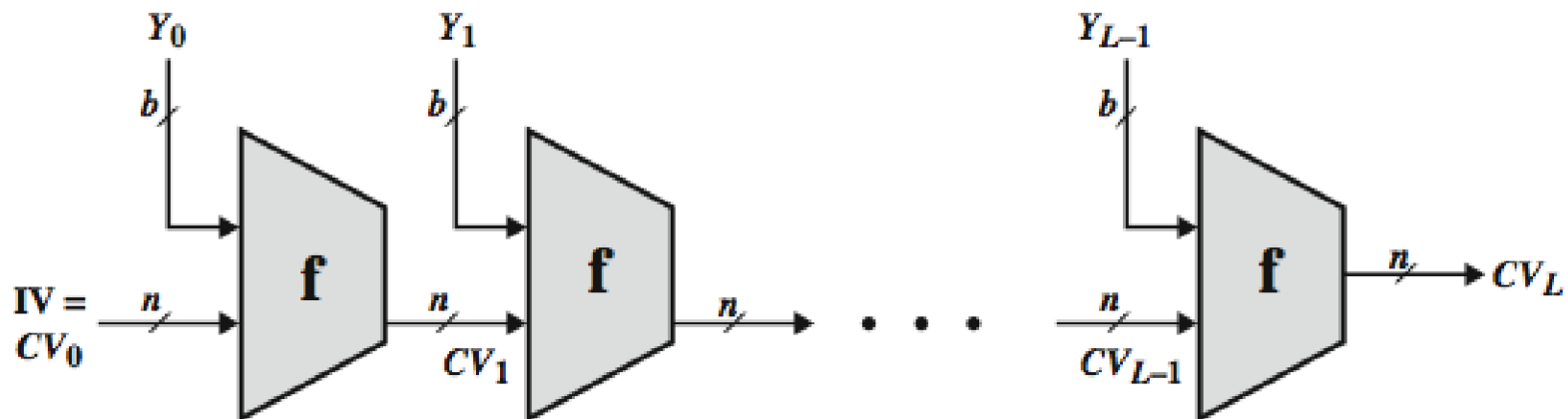
- might think a 64-bit hash is secure
- but by Birthday Paradox is not
- birthday attack works thus:
 - given user prepared to sign a valid message x
 - opponent generates $2^{m/2}$ variations x' of x , all with essentially the same meaning, and saves them
 - opponent generates $2^{m/2}$ variations y' of a desired fraudulent message y
 - two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
 - have user sign the valid message, then substitute the forgery which will have a valid signature
- conclusion is that need to use larger MAC/hash

Birthday Attacks



Cryptanalysis Attacks

- As with encryption algorithms, cryptanalytic attacks on hash functions seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.
- The hash algorithm involves repeated use of a compression function, f , that takes two inputs (an n -bit input from the previous step, called the *chaining variable*, and a b -bit block) and produces an n -bit output



References

1. Cryptography and Network Security, Principles and Practice, William Stallings, Prentice Hall, Sixth Edition, 2013
2. Computer Networking: A Top-Down Approach 6th Edition, Jim Kurose, Keith Ross, Pearson, 2013

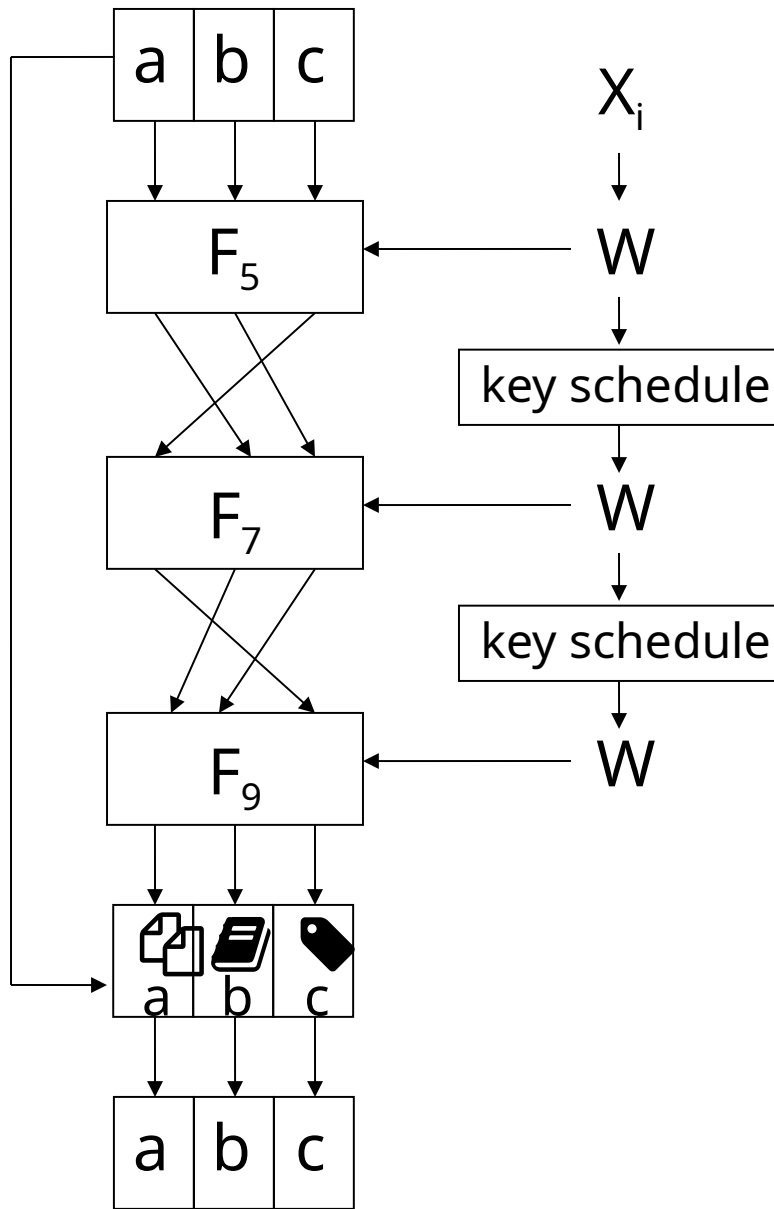


Tiger Hash

- ❑ *“Fast and strong”*
- ❑ *Designed by Ross Anderson and Eli Biham
leading cryptographers*
- ❑ *Design criteria*
 - *Secure*
 - *Optimized for **64-bit** processors*
 - *Easy replacement for MD5 or SHA-1*

Tiger Hash

- ❑ Like MD5/SHA-1, input divided into 512 bit blocks (padded)
- ❑ Unlike MD5/SHA-1, output is **192 bits** (three 64-bit words)
 - Truncate output if replacing MD5 or SHA-1
- ❑ Intermediate rounds are all 192 bits
- ❑ 4 S-boxes, each maps 8 bits to 64 bits
- ❑ A “key schedule” is used

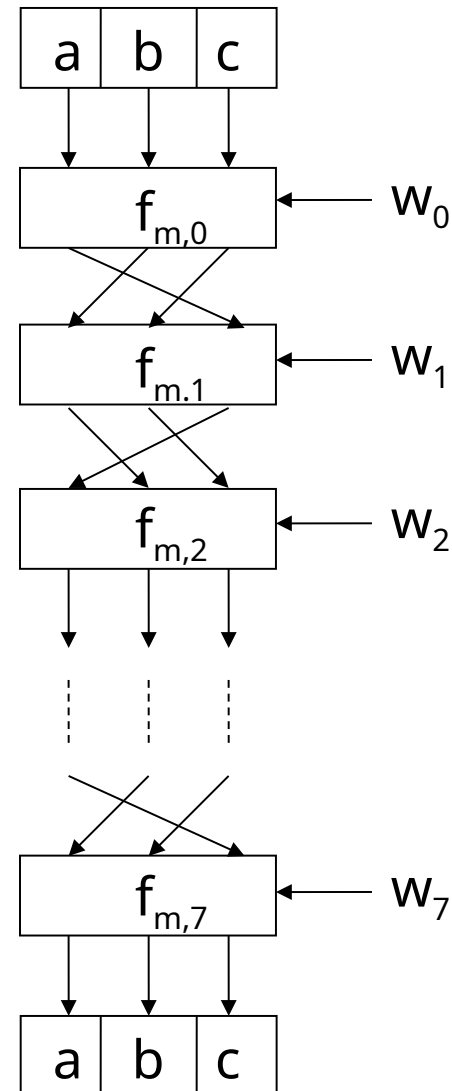


Tiger Outer Round

- ❑ *Input is X*
 - $X = (X_0, X_1, \dots, X_{n-1})$
 - X is padded
 - Each X_i is 512 bits
- ❑ *There are n iterations of diagram at left*
 - One for each input block
- ❑ *Initial (a,b,c) constants*
- ❑ *Final (a,b,c) is hash*
- ❑ *Looks like block cipher!*

Tiger Inner Rounds

- Each F_m consists of precisely **8 rounds**
- 512 bit input W to F_m
 - $W = (w_0, w_1, \dots, w_7)$
 - W is one of the input blocks X_i
- All lines are 64 bits
- The $f_{m,i}$ depend on the S -boxes (next slide)



Tiger Hash: One Round

- Each $f_{m,i}$ is a function of a, b, c, w_i and m
 - Input values of a, b, c from previous round
 - And w_i is 64-bit block of 512 bit W
 - Subscript m is multiplier
 - And $c = (c_0, c_1, \dots, c_7)$
- Output of $f_{m,i}$ is
 - $c = c \oplus w_i$
 - $a = a \oplus (S_0[c_0] \oplus S_1[c_2] \oplus S_2[c_4] \oplus S_3[c_6])$
 - $b = b + (S_3[c_1] \oplus S_2[c_3] \oplus S_1[c_5] \oplus S_0[c_7])$
 - $b = b \lll m$
- Each S_i is *S-box*: 8 bits mapped to 64 bits

Tiger Hash Key Schedule

- *Input is X*
 - $X = (x_0, x_1, \dots, x_7)$
- *Small change in X will produce large change in key schedule output*

$$x_0 = x_0 \oplus (x_7 \oplus 0xA5A5A5A5A5A5A5A5)$$

$$x_1 = x_1 \oplus x_0$$

$$x_2 = x_2 \oplus x_1$$

$$x_3 = x_3 \oplus (x_2 \oplus ((\sim x_1) \ll 19))$$

$$x_4 = x_4 \oplus x_3$$

$$x_5 = x_5 + x_4$$

$$x_6 = x_6 \oplus (x_5 \oplus ((\sim x_4) \gg 23))$$

$$x_7 = x_7 \oplus x_6$$

$$x_0 = x_0 + x_7$$

$$x_1 = x_1 \oplus (x_0 \oplus ((\sim x_7) \ll 19))$$

$$x_2 = x_2 \oplus x_1$$

$$x_3 = x_3 + x_2$$

$$x_4 = x_4 \oplus (x_3 \oplus ((\sim x_2) \gg 23))$$

$$x_5 = x_5 \oplus x_4$$

$$x_6 = x_6 + x_5$$

$$x_7 = x_7 \oplus (x_6 \oplus 0x0123456789ABCDEF)$$

Tiger Hash Summary (1)

- ❑ *Hash and intermediate values are 192 bits*
- ❑ *24 (inner) rounds*
 - ***S-boxes:** Claimed that each input bit affects a, b and c after 3 rounds*
 - ***Key schedule:** Small change in message affects many bits of intermediate hash values*
 - ***Multiply:** Designed to ensure that input to S-box in one round mixed into many S-boxes in next*
- ❑ *S-boxes, key schedule and multiply together designed to ensure strong **avalanche** effect*

Tiger Hash Summary (2)

- *Uses lots of ideas from block ciphers*
 - *S-boxes*
 - *Multiple rounds*
 - *Mixed mode arithmetic*
- *At a higher level, Tiger employs*
 - *Confusion*
 - *Diffusion*

HMAC

- ❑ Can compute a MAC of the message M with key K using a “hashed MAC” or **HMAC**
- ❑ HMAC is a *keyed* hash
 - Why would we need a key?
- ❑ How to compute HMAC?
- ❑ Two obvious choices: $h(K, M)$ and $h(M, K)$
- ❑ Which is better?

HMAC

- ❑ *Should we compute HMAC as $h(K, M)$?*
- ❑ *Hashes computed in blocks*
 - $h(B_1, B_2) = F(F(A, B_1), B_2)$ for some F and constant A
 - Then $h(B_1, B_2) = F(h(B_1), B_2)$
- ❑ *Let $M' = (M, X)$*
 - Then $h(K, M') = F(h(K, M), X)$
 - Attacker can compute HMAC of M' without K
- ❑ *Is $h(M, K)$ better?*
 - Yes, but... if $h(M') = h(M)$ then we might have
 $h(M, K) = F(h(M), K) = F(h(M'), K) = h(M', K)$

Correct Way to HMAC

- ❑ *Described in RFC 2104*
- ❑ *Let B be the block length of hash, in bytes*
 - $B = 64$ for MD5 and SHA-1 and Tiger
- ❑ $\text{ipad} = 0x36$ repeated B times
- ❑ $\text{opad} = 0x5C$ repeated B times
- ❑ *Then*
$$\text{HMAC}(M, K) = h(K \parallel \text{opad}, h(K \parallel \text{ipad}, M))$$

Hash Uses

- ❑ *Authentication (HMAC)*
- ❑ *Message integrity (HMAC)*
- ❑ *Message fingerprint*
- ❑ *Data corruption detection*
- ❑ *Digital signature efficiency*
- ❑ *Anything you can do with symmetric crypto*
- ❑ *Also, many, many clever/surprising uses...*

Online Bids

- ❑ *Suppose Alice, Bob and Charlie are bidders*
- ❑ *Alice plans to bid A, Bob B and Charlie C*
- ❑ *They don't trust that bids will stay secret*
- ❑ *A possible solution?*
 - *Alice, Bob, Charlie submit **hashes** $h(A)$, $h(B)$, $h(C)$*
 - *All hashes received and posted online*
 - *Then bids A, B, and C submitted and revealed*
- ❑ *Hashes don't reveal bids (one way)*
- ❑ *Can't change bid after hash sent (collision)*
- ❑ *But there is a serious flaw here...*

Hashing for Spam Reduction

- ❑ *Spam reduction*
- ❑ *Before accept email, want proof that sender had to “work” to create email*
 - *Here, “work” == CPU cycles*
- ❑ *Goal is to limit the amount of email that can be sent*
 - *This approach will not eliminate spam*
 - *Instead, make spam more costly to send*

Spam Reduction

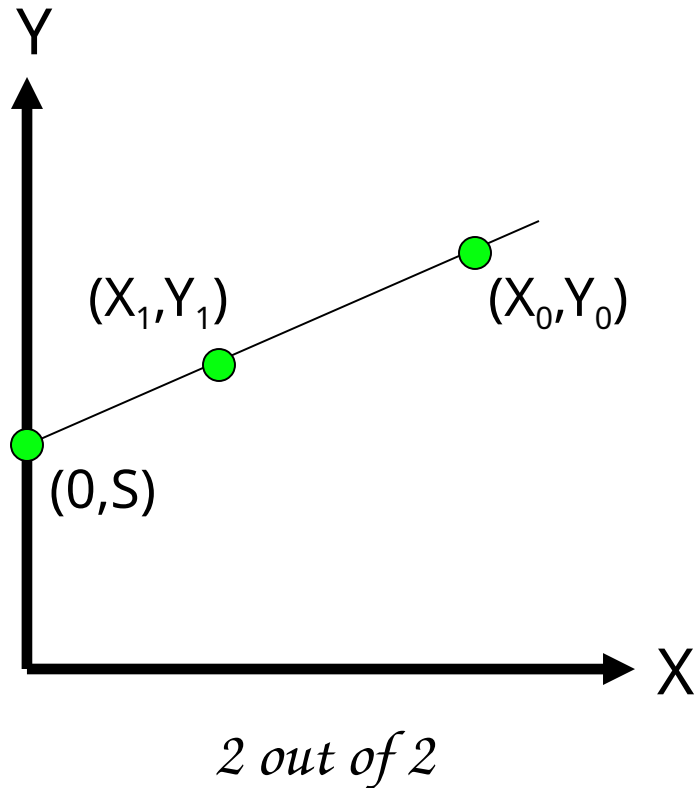
- Let M = complete email message
 R = value to be determined
 T = current time
- Sender must determine R so that
 $h(M, R, T) = (00\dots 0, X)$, that is,
 initial N bits of hash value are *all zero*
- Sender then sends (M, R, T)
- Recipient accepts email, provided that...
 $h(M, R, T)$ begins with N zeros

Spam Reduction

- ❑ *Sender: $h(M,R,T)$ begins with N zeros*
- ❑ *Recipient: verify that $h(M,R,T)$ begins with N zeros*
- ❑ *Work for sender: on average 2^N hashes*
- ❑ *Work for recipient: always 1 hash*
- ❑ *Sender's work increases exponentially in N*
- ❑ *Small work for recipient, regardless of N*
- ❑ *Choose N so that...*
 - *Work acceptable for normal amounts of email*
 - *Work is too high for spammers*

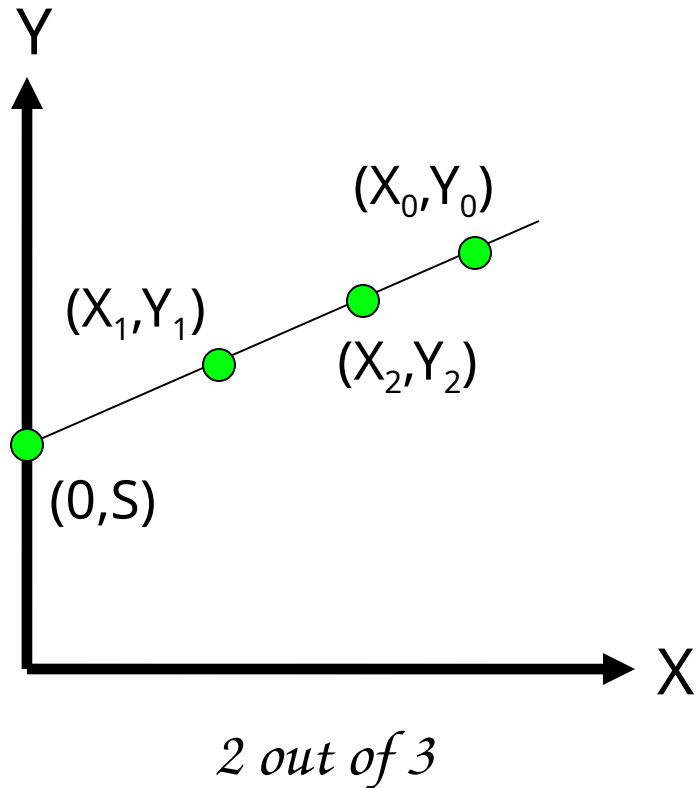
Secret Sharing

Shamir's Secret Sharing



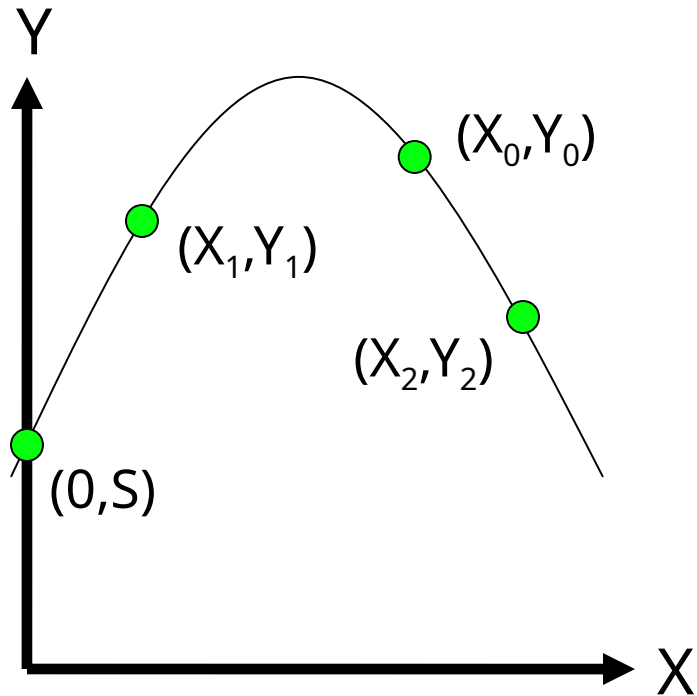
- Two points determine a line
- Give (X_0, Y_0) to Alice
- Give (X_1, Y_1) to Bob
- Then Alice and Bob must cooperate to find secret S
- Also works in discrete case
- Easy to make "m out of n" scheme for any $m \leq n$

Shamir's Secret Sharing



- Give (X_0, Y_0) to Alice
- Give (X_1, Y_1) to Bob
- Give (X_2, Y_2) to Charlie
- Then any **two** can cooperate to find secret S
- No **one** can determine S
- A "2 out of 3" scheme

Shamir's Secret Sharing



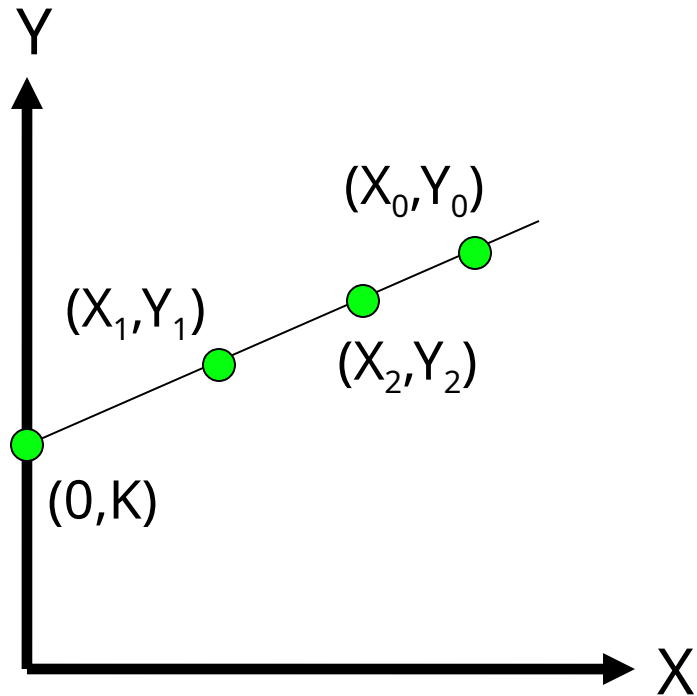
3 out of 3

- Give (X_0, Y_0) to Alice
- Give (X_1, Y_1) to Bob
- Give (X_2, Y_2) to Charlie
- 3 pts determine parabola
- Alice, Bob, *and* Charlie must cooperate to find S
- A “3 out of 3” scheme
- What about “3 out of 4”?

Secret Sharing Use?

- ❑ *Key escrow* suppose it's required that your key be stored somewhere
- ❑ Key can be “recovered” with court order
- ❑ But you don't trust FBI to store your keys
- ❑ We can use secret sharing
 - Say, three different government agencies
 - Two must cooperate to recover the key

Secret Sharing Example



















- *Your symmetric key is K*
- *Point (X_0, Y_0) to FBI*
- *Point (X_1, Y_1) to DoJ*
- *Point (X_2, Y_2) to DoC*
- *To recover your key K , two of the three agencies must cooperate*
- *No one agency can get K*

Visual Cryptography

- ❑ *Another form of secret sharing...*
- ❑ *Alice and Bob “share” an image*
- ❑ *Both must cooperate to reveal the image*
- ❑ *Nobody can learn anything about image from Alice’s share or Bob’s share*
 - *That is, both shares are required*
- ❑ *Is this possible?*

















Visual Cryptography

- How to “share” a pixel?
- Suppose image is black and white
- Then each pixel is either black or white
- We split pixels as shown

	Pixel	Share 1	Share 2	Overlay
a.				
b.				
c.				
d.				

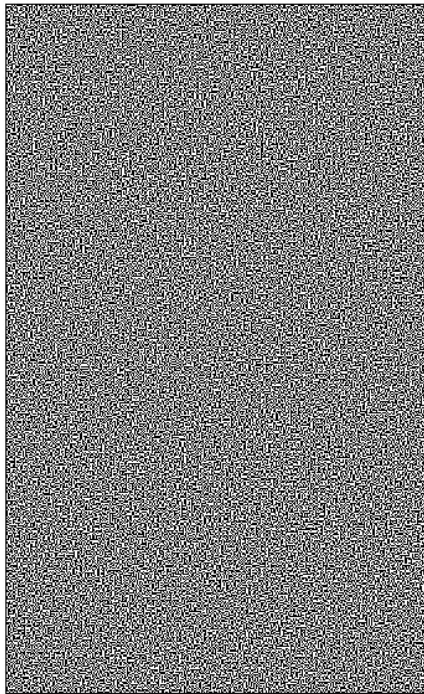
Sharing Black & White Image

- If pixel is white, randomly choose a or b for Alice's/Bob's shares
- If pixel is black, randomly choose C or d
- *No information* in one "share"

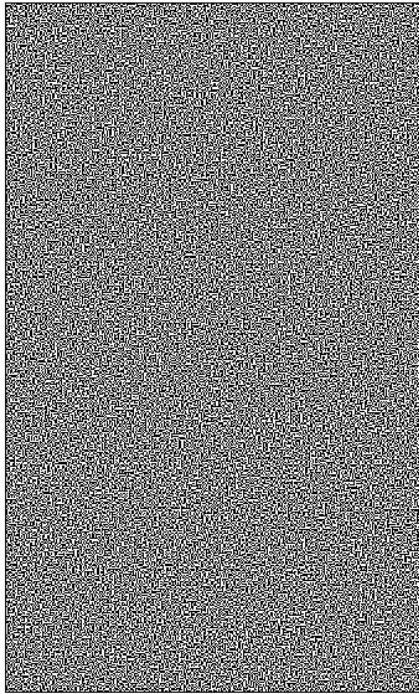
	Pixel	Share 1	Share 2	Overlay
a.				
b.				
c.				
d.				

Visual Crypto Example

□ *Alice's share*



□ *Bob's share*



□ *Overlaid shares*



Visual Crypto

- ❑ *How does visual “crypto” compare to regular crypto?*
- ❑ *In visual crypto, no key...*
 - *Or, maybe both images are the key?*
- ❑ *With encryption, exhaustive search*
 - *Except for the one-time pad*
- ❑ *Exhaustive search on visual crypto?*
 - *No exhaustive search is possible!*

Visual Crypto

- ❑ *Visual crypto no exhaustive search...*
- ❑ *How does visual crypto compare to crypto?*
 - *Visual crypto is “information theoretically” secure also true of secret sharing schemes*
 - *With regular encryption, goal is to make cryptanalysis computationally infeasible*
- ❑ *Visual crypto an example of **secret sharing***
 - *Not really a form of crypto, in the usual sense*

Random Numbers in Cryptography

Random Numbers

- ❑ *Random numbers used to generate **keys***
 - *Symmetric keys*
 - *RSA: Prime numbers*
 - *Diffie Hellman: secret values*
- ❑ *Random numbers used for nonces*
 - *Sometimes a sequence is OK*
 - *But sometimes nonces must be random*
- ❑ *Random numbers also used in simulations, statistics, etc.*
 - *In such apps, need “statistically” random numbers*

Random Numbers

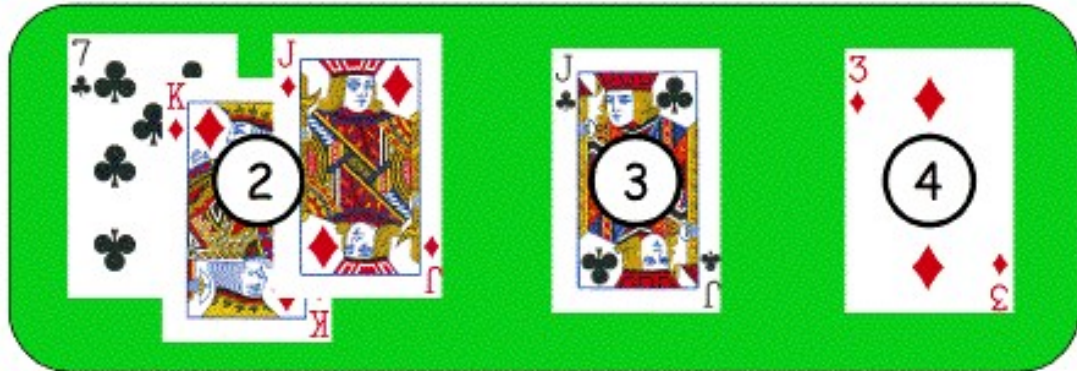
- ❑ *Cryptographic random numbers must be statistically random and **unpredictable***
- ❑ *Suppose server generates symmetric keys*
 - *Alice: K_A*
 - *Bob: K_B*
 - *Charlie: K_C*
 - *Dave: K_D*
- ❑ *Alice, Bob, and Charlie don't like Dave...*
- ❑ *Alice, Bob, and Charlie, working together, must not be able to determine K_D*

Non-random Random Numbers

- ❑ *Online version of Texas Hold 'em Poker*
 - *ASF Software, Inc.*



Player's hand



Community cards in center of the table

- ❑ *Random numbers used to shuffle the deck*
- ❑ *Program did not produce a random shuffle*
- ❑ *A serious problem, or not?*

Card Shuffle

- ❑ *There are $52! > 2^{225}$ possible shuffles*
- ❑ *The poker program used “random” 32-bit integer to determine the shuffle*
 - *So, only 2^{32} distinct shuffles could occur*
- ❑ *Code used Pascal pseudo-random number generator (PRNG): Randomize()*
- ❑ *Seed value for PRNG was function of number of milliseconds since midnight*
- ❑ *Less than 2^{27} milliseconds in a day*
 - *So, less than 2^{27} possible shuffles*

Card Shuffle

- ❑ *Seed based on milliseconds since midnight*
- ❑ *PRNG re-seeded with each shuffle*
- ❑ *By synchronizing clock with server, number of shuffles that need to be tested ~~is~~ 2^{18}*
- ❑ *Could then test all 2^{18} in real time*
 - *Test each possible shuffle against “up” cards*
- ❑ *Attacker knows **every card** after the first of five rounds of betting!*

Poker Example

- ❑ *Poker program is an extreme example*
 - *But common PRNGs are predictable*
 - *Only a question of how many outputs must be observed before determining the sequence*
- ❑ *Crypto random sequences not predictable*
 - *For example, keystream from RC4 cipher*
 - *But “seed” (or key) selection is still an issue!*
- ❑ *How to generate initial **random** values?*
 - *Keys (and, in some cases, seed values)*

What is Random?

- ❑ *True “random” is hard to define*
- ❑ ***Entropy** is a measure of randomness*
- ❑ *Good sources of “true” randomness*
 - *Radioactive decay but, radioactive computers are not too popular*
 - *Hardware devices many good ones on the market*
 - *Lava lamp relies on chaotic behavior*

Randomness

- ❑ *Sources of randomness via software*
 - *Software is supposed to be deterministic*
 - *So, must rely on external “random” events*
 - *Mouse movements, keyboard dynamics, network activity, etc., etc.*
- ❑ *Can get **quality** random bits by such methods*
- ❑ *But **quantity** of bits is very limited*
- ❑ *Bottom line: “The use of pseudo-random processes to generate secret quantities can result in pseudo-security”*

Information Hiding

Information Hiding

□ *Digital Watermarks*

- *Example: Add “invisible” info to data*
- *Defense against music/software piracy*

□ *Steganography*

- *“Secret” communication channel*
- *Similar to a **covert channel** (more later)*
- *Example: Hide data in an image file*

Watermark

- ❑ *Add a “mark” to data*
- ❑ *Visibility (or not) of watermarks*
 - *Invisible Watermark is not obvious*
 - *Visible Such as **TOP SECRET***
- ❑ *Strength (or not) of watermarks*
 - *Robust Readable even if attacked*
 - *Fragile Damaged if attacked*

Watermark Examples

- ❑ Add *robust invisible* mark to digital music
 - If pirated music appears on Internet, can trace it back to original source of the leak
- ❑ Add *fragile invisible* mark to audio file
 - If watermark is unreadable, recipient knows that audio has been tampered with (integrity)
- ❑ Combinations of several types are sometimes used
 - E.g., visible plus robust invisible watermarks

Watermark Example (1)

- *Non-digital watermark: U.S. currency*



- *Image embedded in paper on rhs*
 - *Hold bill to light to see embedded info*

Watermark Example (2)

- ❑ Add *invisible* watermark to photo
- ❑ Claim is that 1 inch² contains enough info to reconstruct entire photo
- ❑ If photo is damaged, watermark can be used to reconstruct it!

Steganography

- ❑ *According to Herodotus (Greece 440 BC)*
 - *Shaved slave's head*
 - *Wrote message on head*
 - *Let hair grow back*
 - *Send slave to deliver message*
 - *Shave slave's head to expose a message warning of Persian invasion*
- ❑ *Historically, steganography used by military more often than cryptography*

Images and Steganography

- ❑ Images use 24 bits for color: **R****G****B**
 - 8 bits for *red*, 8 for *green*, 8 for *blue*
- ❑ For example
 - **0x7E 0x52 0x90** is *this color*
 - **0xFE 0x52 0x90** is *this color*
- ❑ While
 - **0xAB 0x33 0xF0** is *this color*
 - **0xAB 0x33 0xF1** is *this color*
- ❑ Low-order bits don't matter...

Images and Stego

- *Given an uncompressed image file...*
 - *For example, BMP format*
- *...we can insert information into low-order RGB bits*
- *Since low-order RGB bits don't matter, changes will be “invisible” to human eye*
 - *But, computer program can “see” the bits*

Stego Example 1



- ❑ *Left side: plain Alice image*
- ❑ *Right side: Alice with entire Alice in Wonderland (pdf) “hidden” in the image*

Non-Stego Example

❑ *Walrus.html in web browser*

"The time has come," the Walrus said,
"To talk of many things:
Of shoes and ships and sealing wax
Of cabbages and kings
And why the sea is boiling hot
And whether pigs have wings."

❑ *"View source" reveals:*

```
<font color=#000000>"The time has come," the Walrus said,</font><br>  
<font color=#000000>"To talk of many things: </font><br>  
<font color=#000000>Of shoes and ships and sealing wax </font><br>  
<font color=#000000>Of cabbages and kings </font><br>  
<font color=#000000>And why the sea is boiling hot </font><br>  
<font color=#000000>And whether pigs have wings." </font><br>
```


Stego Example 2

❑ *stegoWalrus.html in web browser*

"The time has come," the Walrus said,
"To talk of many things:
Of shoes and ships and sealing wax
Of cabbages and kings
And why the sea is boiling hot
And whether pigs have wings."

❑ *"View source" reveals:*

```
<font color=#000101>"The time has come," the Walrus  
said,</font><br>  
<font color=#000100>"To talk of many things: </font><br>  
<font color=#010000>Of shoes and ships and sealing wax </font><br>  
<font color=#010000>Of cabbages and kings </font><br>  
<font color=#000000>And why the sea is boiling hot </font><br>  
<font color=#010001>And whether pigs have wings." </font><br>
```

❑ *"Hidden" message: 011 010 100 100 000 101*

Steganography

- ❑ *Some formats (e.g., image files) are more difficult than html for **humans** to read*
 - *But easy for computer programs to read...*
- ❑ *Easy to hide info in **unimportant bits***
- ❑ *Easy to damage info in unimportant bits*
- ❑ *To be **robust**, must use **important bits***
 - *But stored info must not damage data*
 - *Collusion attacks are also a concern*
- ❑ *Robust steganography is tricky!*

Information Hiding: The Bottom Line

- ❑ *Not-so-easy to hide digital information*
 - “Obvious” approach is **not** robust
 - **Stirmark**: tool to make most watermarks in images unreadable without damaging the image
 - Stego/watermarking are active research topics
- ❑ *If information hiding is suspected*
 - Attacker may be able to make information/watermark unreadable
 - Attacker may be able to read the information, given the original document (image, audio, etc.)