Integrity

Most security failures in its area of interest are due to failures in implementation, not failure in algorithms or protocols
-NSA

Tamer ABUHMED
School of Computer and Information Engineering
INHA University

Outline

- Introduction
- Integrity functionality
- Digital Signature (Public key Cryptography)
- Hash Function
- Message Authentication Code (MAC) Symmetric Encryption

Introduction

Data integrity means maintaining and assuring the accuracy and consistency of data over its entire life-cycle

>> Data cannot be modified in an unauthorized or undetected manner

Integrity Functionality

- **Integrity** — detect unauthorized writing (i.e., modification of data)
- Example: Inter-bank fund transfers
  - Confidentiality may be nice, integrity is critical
- Encryption provides **confidentiality** (prevents unauthorized disclosure)
- Encryption alone does **not** provide integrity
  - One-time pad, ECB cut-and-paste, etc.

Integrity **not** the same as confidentiality

Message Integrity

Bob receives message from Alice, wants to ensure:

- message originally came from Alice (Authentication)
- message not changed since sent by Alice (Integrity)

Cryptographic Hash:

- takes input m, produces fixed length value, H(m)
  - e.g., as in Internet checksum
- computationally infeasible to find two different messages, x, y such that H(x) = H(y)
  - equivalently: given m = H(x), (x unknown), can not determine x.
  - note: Internet checksum *fails* this requirement!

Digital Signatures

Public Key Cryptography
Asymmetric cryptography

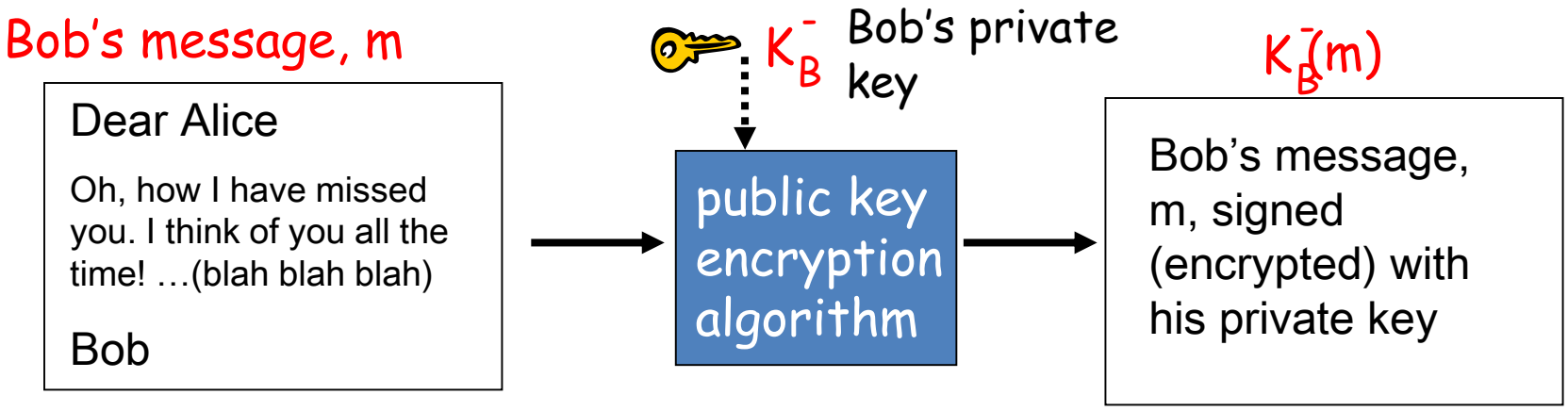
cryptographic technique analogous to hand-written signatures for preserving data integrity.

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- verifiable, nonforgeable: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

Digital Signatures

simple digital signature for message m:

- Bob "signs" m by encrypting with his private key K_B^-, creating "signed" message, $K_B^-(m)$



Digital Signatures (more)

- suppose Alice receives msg m, digital signature $K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.
- if $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.

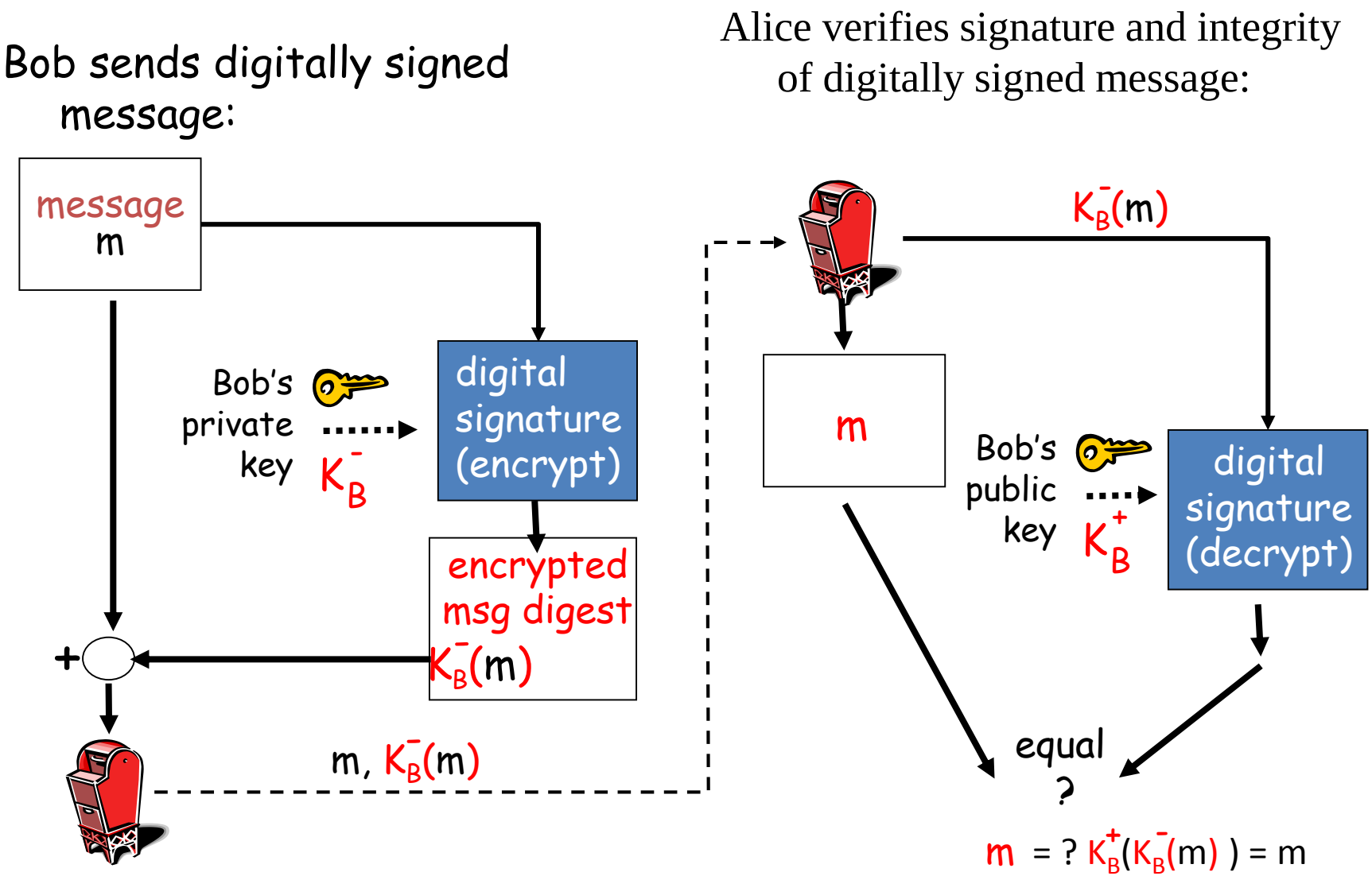
Alice thus verifies that:

- ✓ Bob signed m.
- ✓ No one else signed m.
- ✓ Bob signed m and not m'.

non-repudiation:

- ✓ Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m.

Digital signature scenario



Hash Function

Hash Function Motivation

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

Hash Function Motivation

- So, Alice signs h(M)
  - That is, Alice computes $S = [h(M)]_{Alice}$
  - Alice then sends (M, S) to Bob
  - Bob verifies that $h(M) = \{S\}_{Alice}$
- 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 $h(M') = h(M) = \{S\}_{Alice}$

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 ≠ x such that h(y) = h(x)
  - **Strong collision resistance** — infeasible to find ***any*** x and y, with x ≠ y such that h(x) = h(y)
- Lots of collisions exist, but hard to find ***any***

Pre-Birthday Problem

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

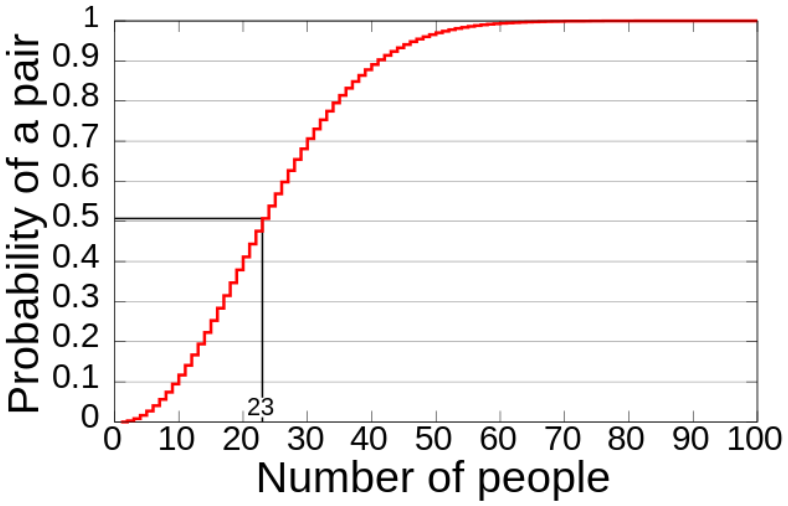
Birthday Problem

- How many people must be in a room before probability is ≥ 1/2 that any two (or more) have same birthday?
  - 1 – 365/365 · 364/365 · · ·(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, 2^N different hash values are possible
- So, if you hash about $2^{N/2}$ random values then you expect to find a collision
  - Since $\text{sqrt}(2^N) = 2^{N/2}$
- **Implication:** secure N bit symmetric key requires 2^{N-1} work to "break" while secure N bit hash requires $2^{N/2}$ work to "break"
  - Exhaustive search attacks, that is

Non-crypto Hash (1)

- Data $X = (X_0, X_1, X_2, \ldots, X_{n-1})$, each X_i is a byte
- Define $h(X) = X_0 + X_1 + X_2 + \ldots + X_{n-1}$
- 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 $X = (X_0, X_1, X_2, \ldots, X_{n-1})$
- Suppose hash is defined as

  $h(X) = nX_0 + (n-1)X_1 + (n-2)X_2 + \ldots + 1 \cdot X_{n-1}$
- Is this a secure cryptographic hash?
- Note that

  $h(10101010, 00001111) \neq h(00001111, 10101010)$
- But hash of (00000001, 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**
  - Random errors unlikely to yield a collision
- But easy to construct collisions
- CRC has been mistakenly used where crypto integrity check is required (e.g., WEP)

Internet checksum: poor crypto hash function

checksum has some properties of hash function:

- ✓ produces fixed length digest (16-bit sum) of message
- ✓ is many-to-one

But given message with given hash value, it is easy to find another message with same hash value:

message	ASCII format
I O U 1	49 4F 55 31
0 0 . 9	30 30 2E 39
9 B O B	39 42 4F 42
	B2 C1 D2 AC

message	ASCII format
I O U 9	49 4F 55 39
0 0 . 1	30 30 2E 31
9 B O B	39 42 4F 42
	B2 C1 D2 AC

different messages but identical checksums!

Popular Crypto Hashes

- **MD5** — invented by Rivest
  - 128 bit output
  - Note: MD5 collisions easy to find
- **SHA-1** — A U.S. government standard, inner workings similar to MD5
  - 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 & Integrity

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

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)

The Right 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
- ipad = 0x36 repeated B times
- opad = 0x5C repeated B times
- Then

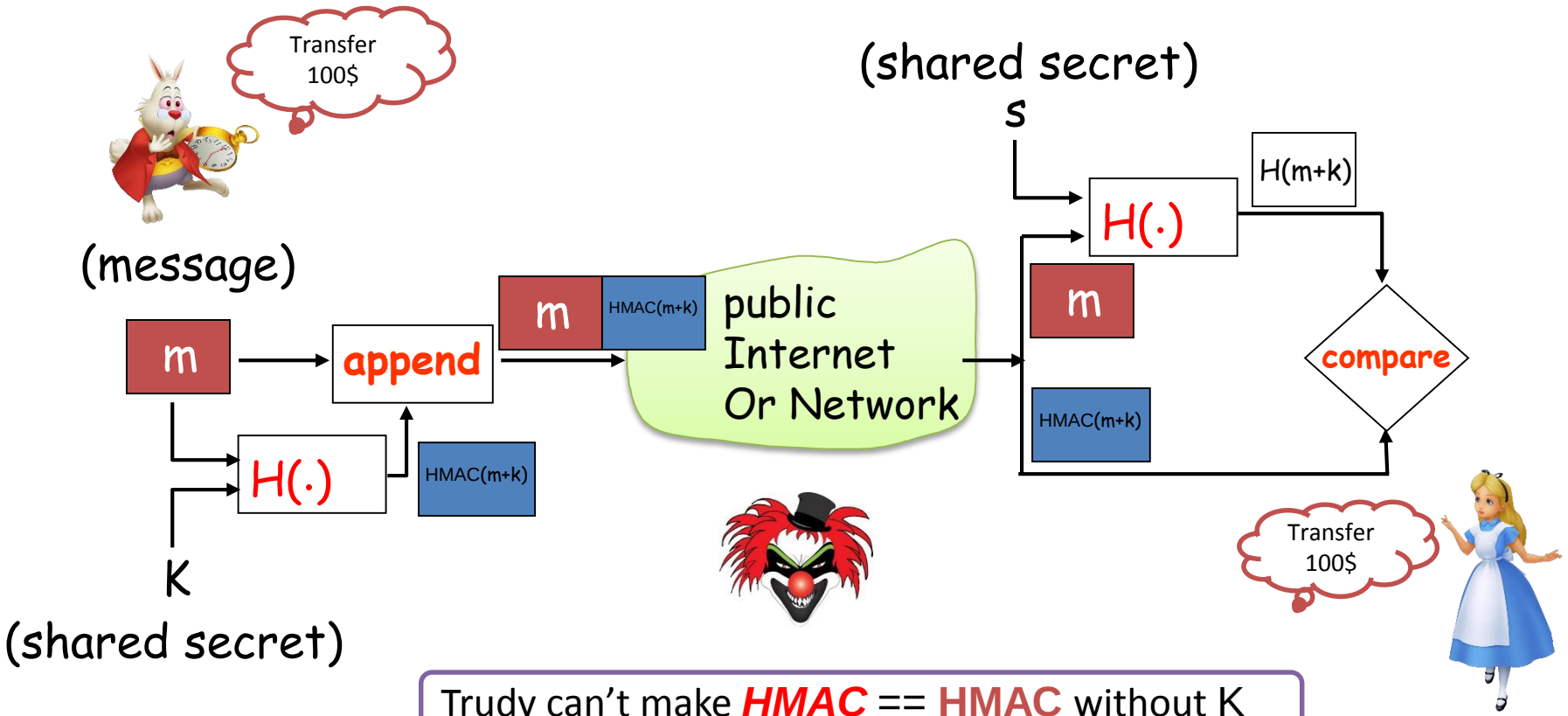
  HMAC(M,K) = h(K ⊕ opad, h(K ⊕ ipad, M))

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…

Message Authentication Code

Symmetric cryptography



Trudy can't make ***HMAC*** == **HMAC** without K

MACs in practice

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit MAC in 4-step process.
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x
    - recent (2005) attacks on MD5
- SHA-1 is also used
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit MAC

Confidentiality and Integrity

- Encrypt with one key, MAC with another key
- Why not use the same key?
  - Send last encrypted block (MAC) twice?
  - This cannot add any security!
- Using different keys to encrypt and compute MAC works, even if keys are related
  - But, twice as much work as encryption alone
  - Can do a little better — about 1.5 “encryptions”
- Confidentiality and integrity with same work as one encryption is a research topic

Uses for Symmetric Crypto

- Confidentiality
  - Transmitting data over insecure channel
  - Secure storage on insecure media
- Integrity (MAC)
- Authentication protocols (later…)