Uses for Public Key Crypto

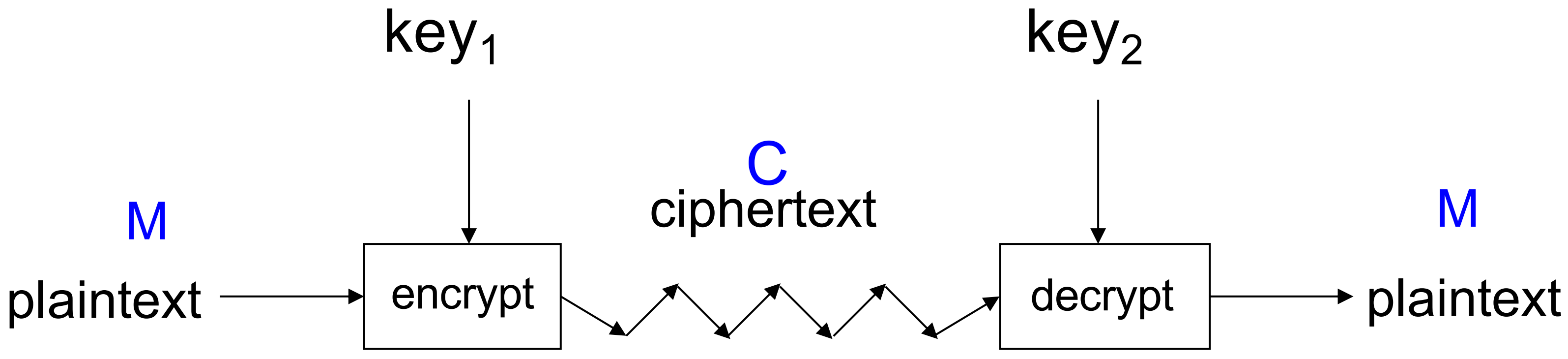
BINGHAMTON
UNIVERSITY
STATE UNIVERSITY OF NEW YORK

Public Key Crypto

- Knapsack public key cryptosystem
- RSA
- Diffie-Hellman
- El Gamal
- ECC

How to use public key crypto?

Public Key Crypto



- A key pair is used in public key cryptography
  - **Public Key**: K_{pub}
  - **Private key**: K_{pri}

Public Key Notation

- **Sign** message M with Alice's **private key:** $[\mathrm{M}]_{\mathrm{Alice}}$
- **Encrypt** message M with Alice's **public key:** $\{\mathrm{M}\}_{\mathrm{Alice}}$
- Then

$$\{[\mathrm{M}]_{\mathrm{Alice}}\}_{\mathrm{Alice}} = \mathrm{M}$$

$$[\{\mathrm{M}\}_{\mathrm{Alice}}]_{\mathrm{Alice}} = \mathrm{M}$$

- Notations:
  - Square brackets: [] → Private key
  - Curly brackets: {} → Public key

Uses for Public Key Crypto

- Alice has her private/public key pair
- Bob has his private/public key pair
- Confidentiality
- Integrity
- Non-repudiation

Alice's data

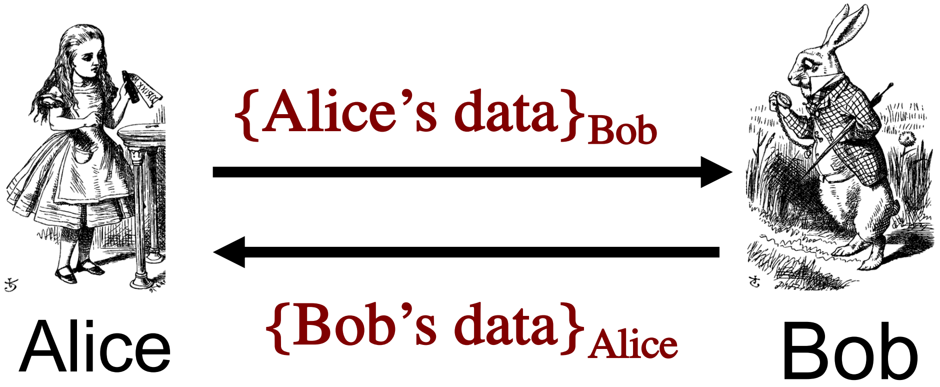
Alice

Bob's data

Bob

Confidentiality

- If Alice wants to send a secret message to Bob or Bob wants to send a secret message to Alice, what should they do?



$\{\text{Alice's data}\}_{\text{Bob}}$

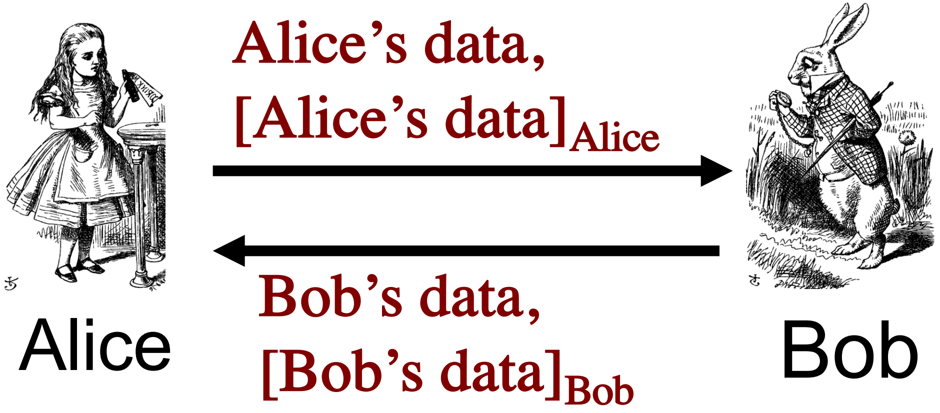
$\{\text{Bob's data}\}_{\text{Alice}}$

Alice

Bob

Integrity

- How should Alice send her data to Bob with integrity ensured? How should Bob send his data to Alice with integrity ensured?



Alice's data, $[\text{Alice's data}]_{\text{Alice}}$

Bob's data, $[\text{Bob's data}]_{\text{Bob}}$

Alice

Bob

Digital signature ensures integrity

Non-repudiation with symmetric keys?

- Alice orders 100 shares of stock from Bob
- Alice computes **MAC** using symmetric key
- Stock drops, Alice claims she did ***not*** order
- Alice and Bob went to the judge
- Can Bob prove that Alice placed the order?
- **No!** Since Bob also knows the symmetric key, he could have forged message
- **Problem:** Bob knows Alice placed the order, but he can't prove it

Non-repudiation with public key crypto

- Alice orders 100 shares of stock from Bob
- Alice **signs** order with her private key
- Stock drops, Alice claims she did not order
- Alice and Bob went to the judge
- Can Bob prove that Alice placed the order?
- **Yes!** Only someone with Alice's private key could have signed the order
- This assumes Alice's private key is not stolen

Digital signature ensures non-repudiation

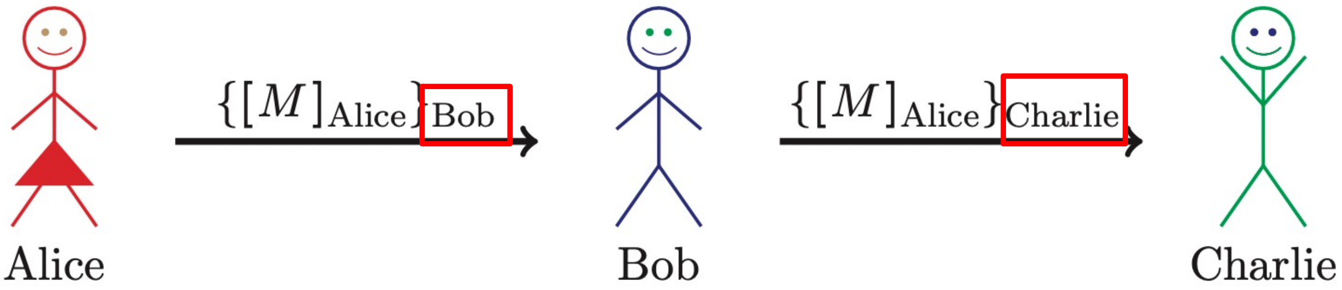
Sign and Encrypt
vs
Encrypt and Sign

Confidentiality and Non-repudiation?

- Suppose that we want confidentiality and integrity/non-repudiation
- Can public key crypto achieve both?
- Alice sends message to Bob
  - **Sign and encrypt:** $\{[M]_{Alice}\}_{Bob}$
  - **Encrypt and sign:** $[\{M\}_{Bob}]_{Alice}$
- Can the order possibly matter?

Sign and Encrypt

- ❑ M = “I love you”

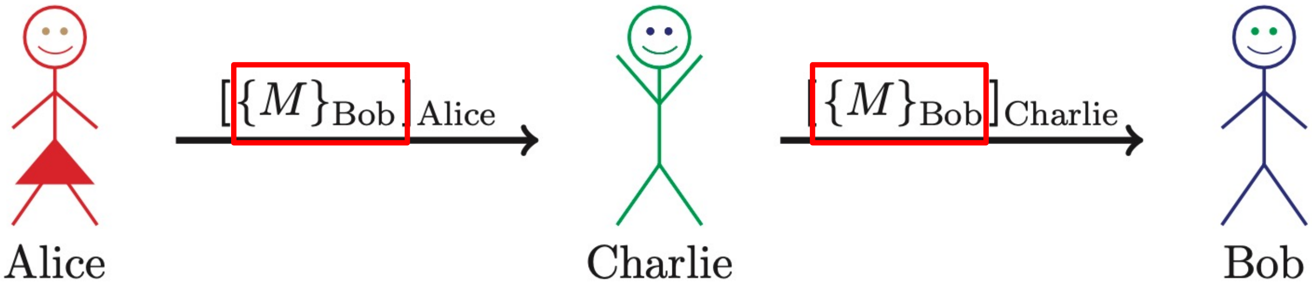

$\{[M]_{\text{Alice}}\}_{\boxed{\text{Bob}}}$
$\{[M]_{\text{Alice}}\}_{\boxed{\text{Charlie}}}$
Alice
Bob
Charlie

Alice signed the message.
But anyone can use Charlie’s public key

- ❑ **Q:** What’s the problem?
- ❑ A: Public key is public!

Encrypt and Sign

- M = “My theory, which is mine….”


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

Charlie signed the message.
But anyone can use Bob’s public key

- **Note** that Charlie cannot decrypt M
- **Q:** What is the problem?
- A: Public key is public!

Public Key Infrastructure(PKI)

Challenge of using public key crypto?

- Distribution of public keys!

Public Key Certificate

- **Certificate** contains name of user and user's public key (and possibly other info)
- It is ***signed*** by the issuer, a ***Certificate Authority*** (CA), such as VeriSign

  M = (Alice, Alice's public key), $S = [M]_{CA}$

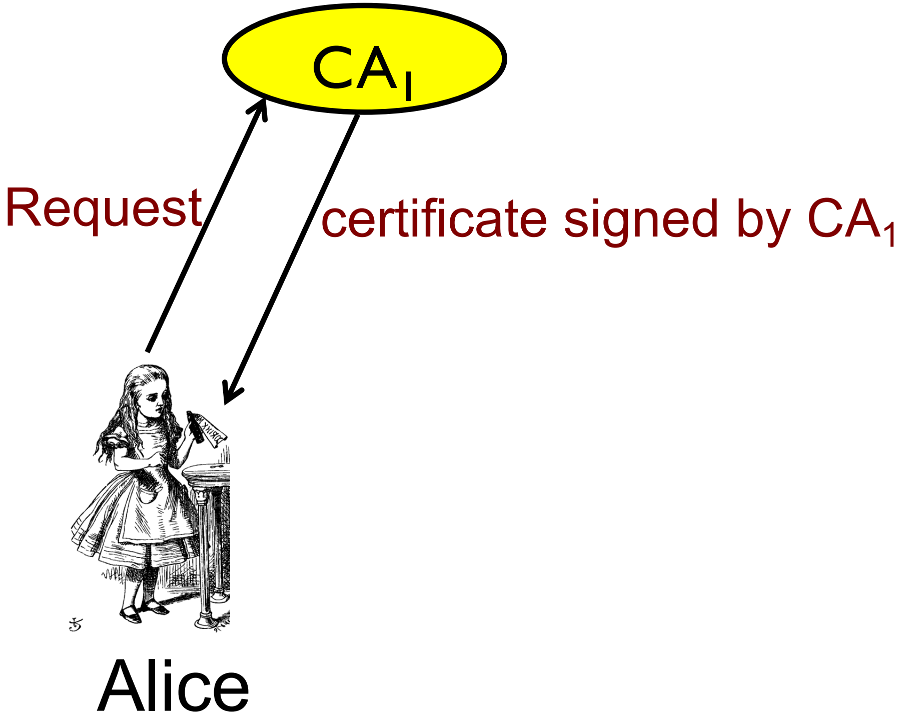
  Alice's Certificate $= (M, S)$
- Signature on certificate is verified using CA's public key:

  Verify that $M = \{S\}_{CA}$

Certificate Authority

- Certificate authority (CA) is a trusted 3rd party (TTP) — creates and signs certificates
- Verify signature to verify integrity & identity of **owner of corresponding private key**
  - Does **not** verify the identity of the **sender** of certificate — certificates are public information!
- Big problem if CA makes a mistake
  - CA once issued Microsoft cert. to someone else

Alice obtains a certificate from CA$_1$



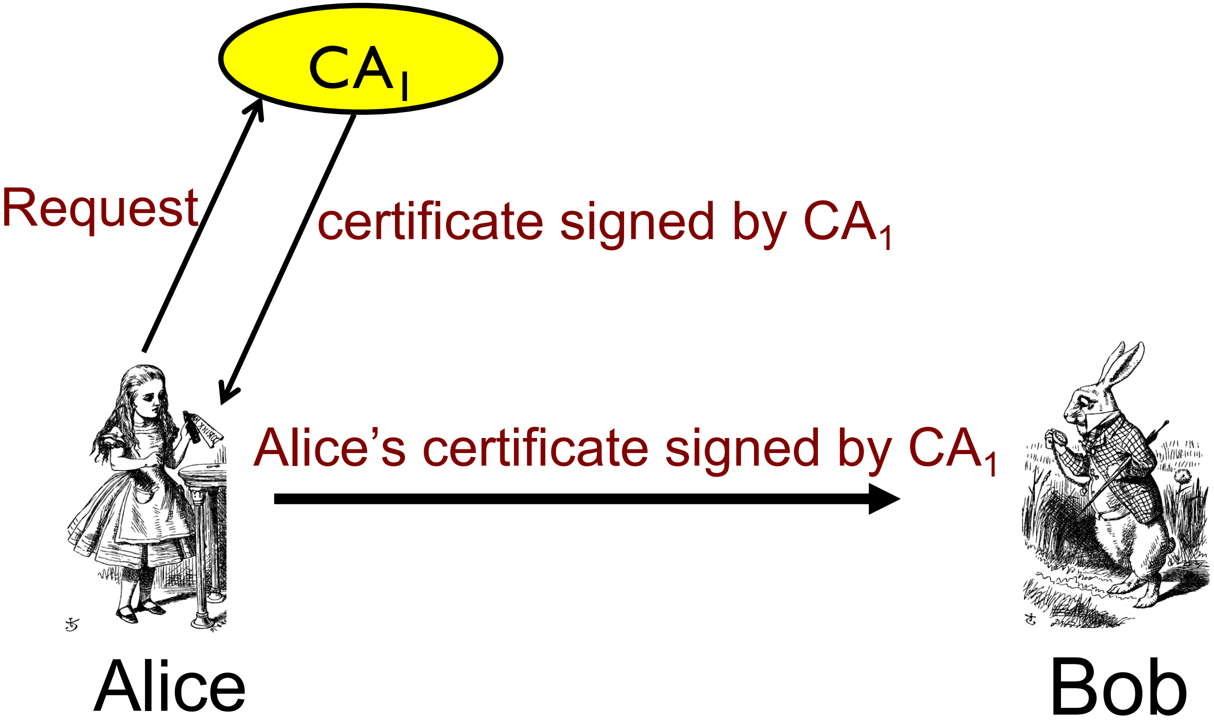
CA$_1$

Request

certificate signed by CA$_1$

Alice

Bob verifies Alice’s certificate with CA$_1$’s public key



CA$_1$

Request

certificate signed by CA$_1$

Alice’s certificate signed by CA$_1$

Alice

Bob

This requires Bob trusts CA$_1$’s public key

PKI

- Public Key Infrastructure (PKI): the stuff needed to securely use public key crypto
  - Key generation and management
  - Certificate authority (CA) or authorities
  - Certificate revocation lists (CRLs), etc.
    - The list of certificates that have been revoked
- No general standard for PKI
- We mention 3 generic "trust models"

PKI Trust Models

- **Monopoly model**
  - One universally trusted organization is the CA for the known universe
  - Big problems if CA is ever compromised
  - Who will act as CA???
    - System is useless if you don't trust the CA!

PKI Trust Models

- **Oligarchy**
  - Multiple trusted CAs
  - This is approach used in browsers today
  - Browser may have 80 or more certificates, just to verify certificates!
  - User can decide which CAs to trust

PKI Trust Models

- **Anarchy model**
  - Everyone is a CA…
  - Users must decide who to trust
- Why is it anarchy?
  - Suppose a certificate is signed by Frank and you don’t know Frank, but you do trust Bob and Bob says Alice is trustworthy and Alice vouches for Frank. Should you accept the certificate?
- **Many** other trust models and PKI issues

Symmetric Key vs Public Key

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

    $[M]_{Alice}$
  - Encrypt M with Alice’s **public key**

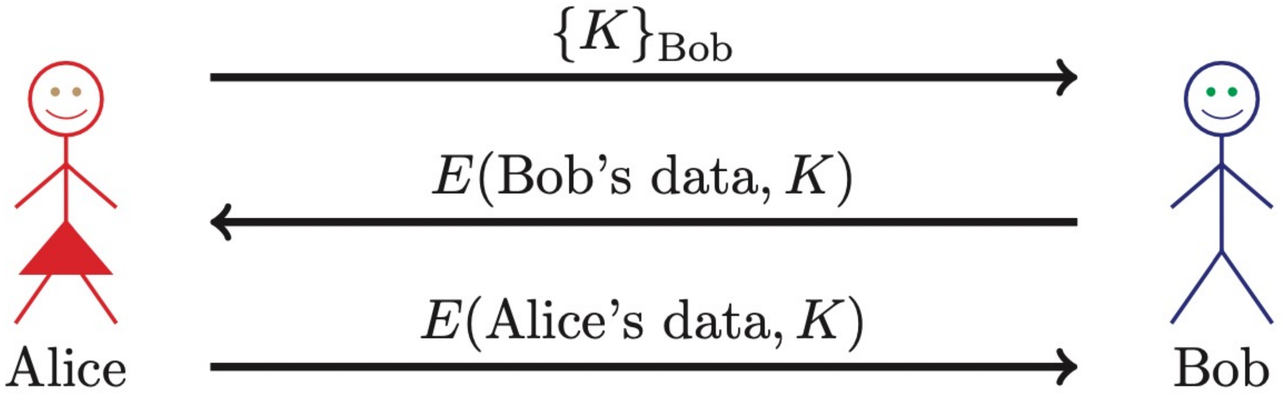
    $\{M\}_{Alice}$
- Symmetric key notation
  - Encrypt P with **symmetric key** K

    $C = E(P,K)$
  - 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…



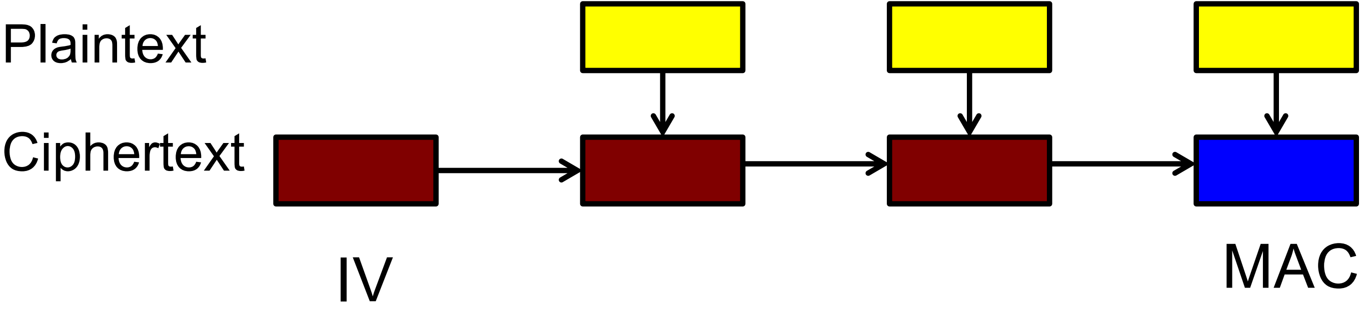
Crypto Hash

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 $y \neq x$ such that $h(y) = h(x)$
  - **Strong collision resistance** — infeasible to find ***any*** x and y, with $x \neq y$ such that $h(x) = h(y)$
- The output of h(x) is called the **hash value** (or **message digest**, **digital fingerprint**, **digest**, …)
- Lots of collisions exist
  - E.g., 150 bit input, 128 bit output; 2^{22} of these input values hash to each possible output

Data integrity with symmetric key crypto

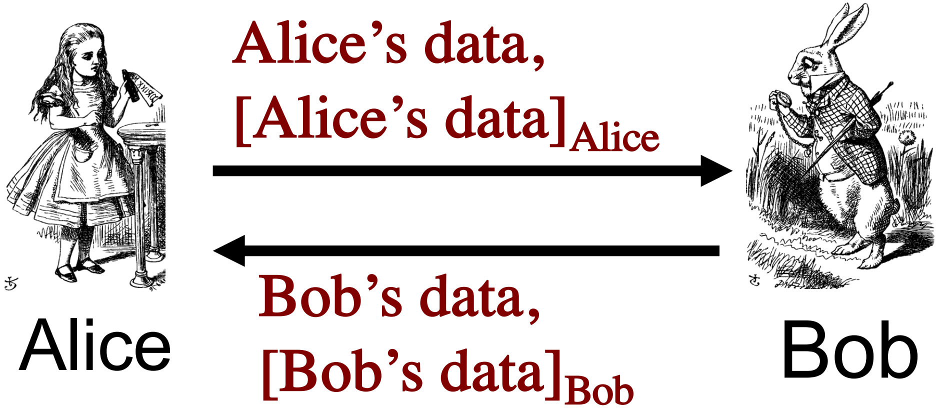
- Message Authentication Code (MAC)
  - Used for data **integrity**
  - Integrity **not** the same as confidentiality
- MAC is computed as **CBC residue**
  - That is, compute CBC encryption, saving only final ciphertext block, the MAC



A shared key is needed

Data integrity with public key crypto

- How should Alice send her data to Bob with integrity ensured? How should Bob send his data to Alice with integrity ensured?



Alice’s data, [Alice’s data]$_{Alice}$

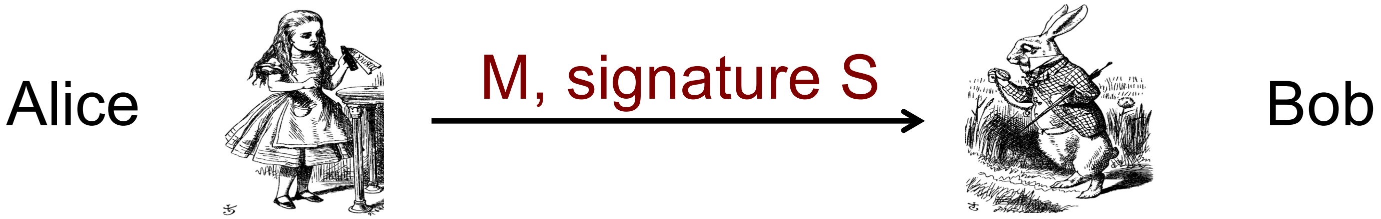
Bob’s data, [Bob’s data]$_{Bob}$

Alice

Bob

Digital signature ensures integrity

Hash Function Motivation



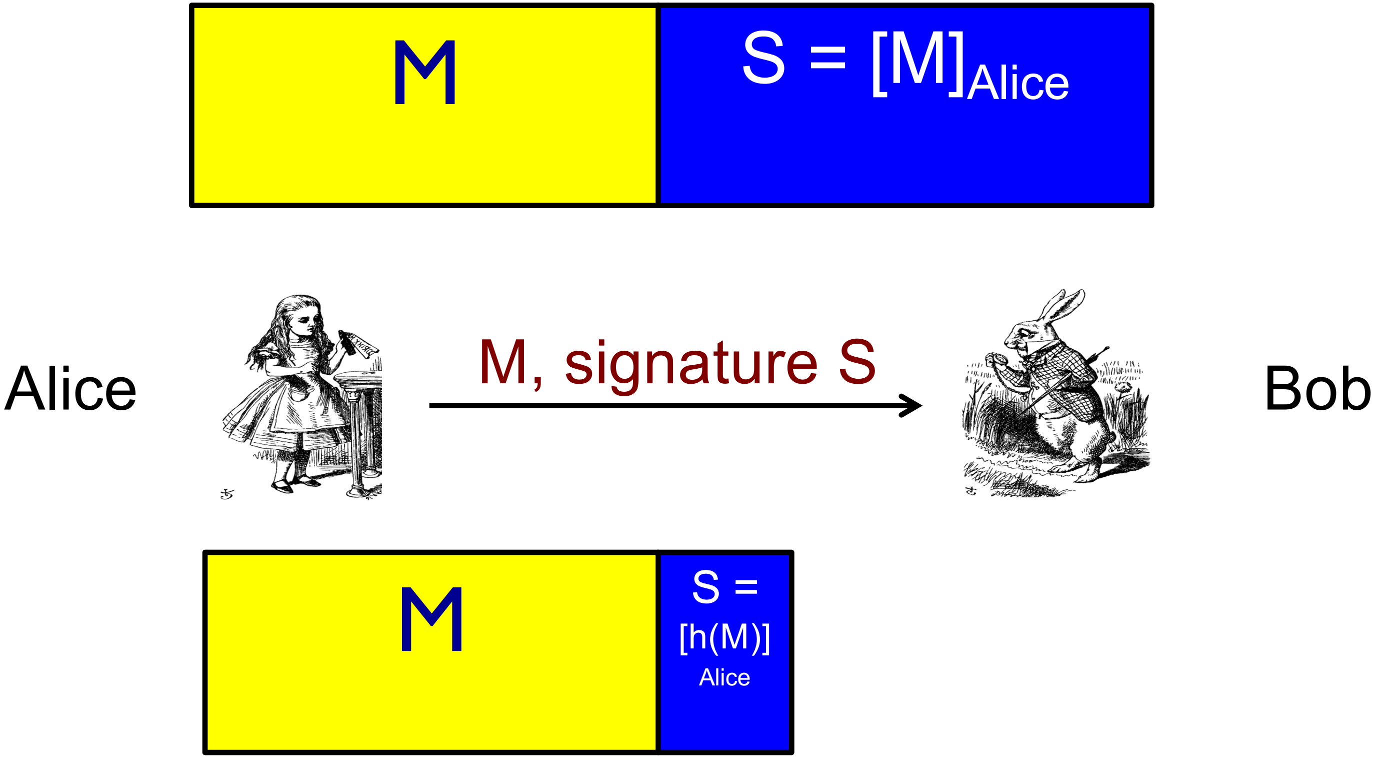
Alice

M, signature S

Bob

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

Comparison



M

$S = [M]_{Alice}$

Alice

M, signature S

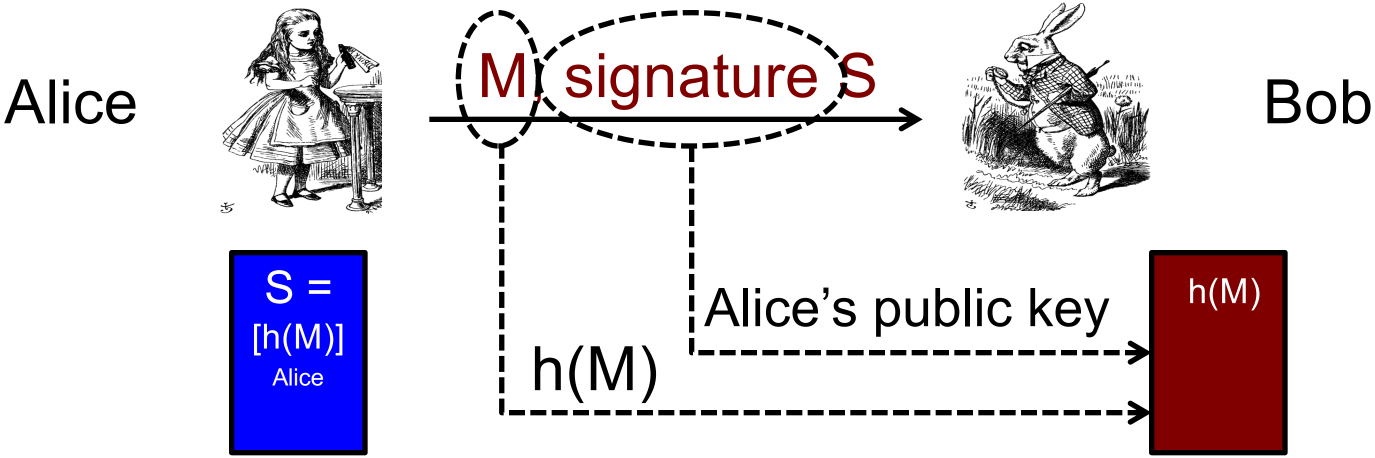
Bob

M

$S = [h(M)]_{Alice}$

Verification by Bob: hash function public

1. Computes h(M)
2. Use Alice's public key to decrypt S
3. If h(M) = $\{S\}_{Alice}$, true



Alice

M, signature S

Bob

S = $[h(M)]_{Alice}$

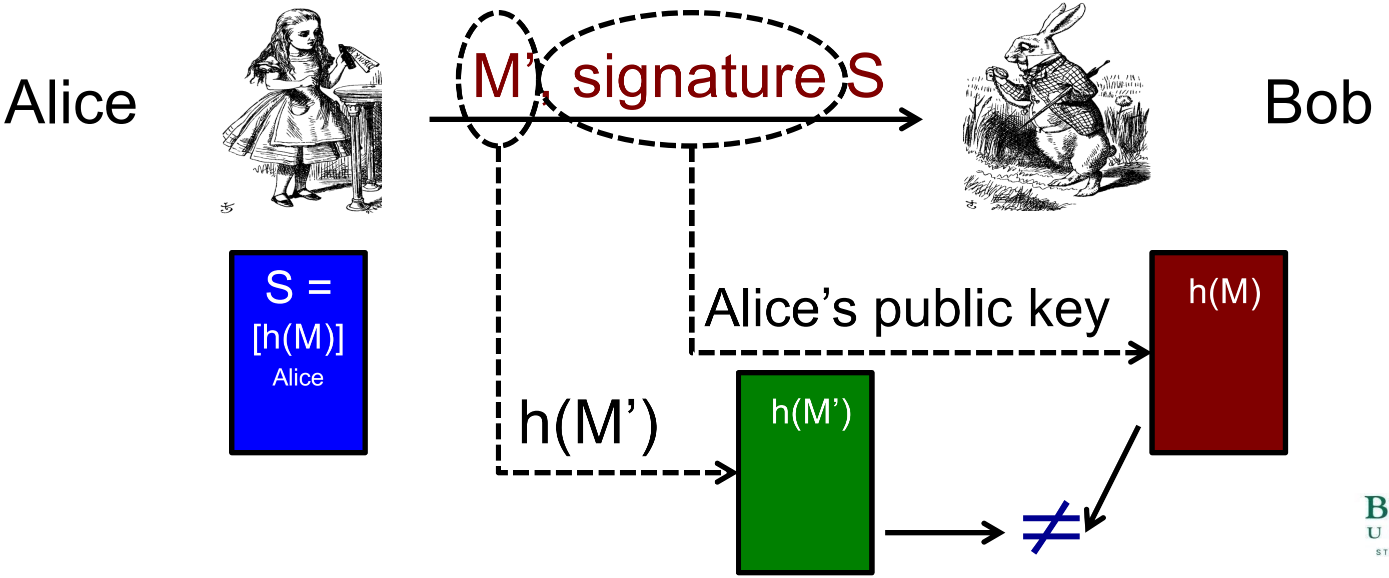
Alice's public key

h(M)

h(M)

Verification by Bob: hash function public

1. Computes h(M’)
2. Use Alice’s public key to decrypt S
3. If $h(M') \neq \{S\}_{Alice}$, false



Alice

M’, signature S

Bob

$S = [h(M)]_{Alice}$

Alice’s public key

h(M)

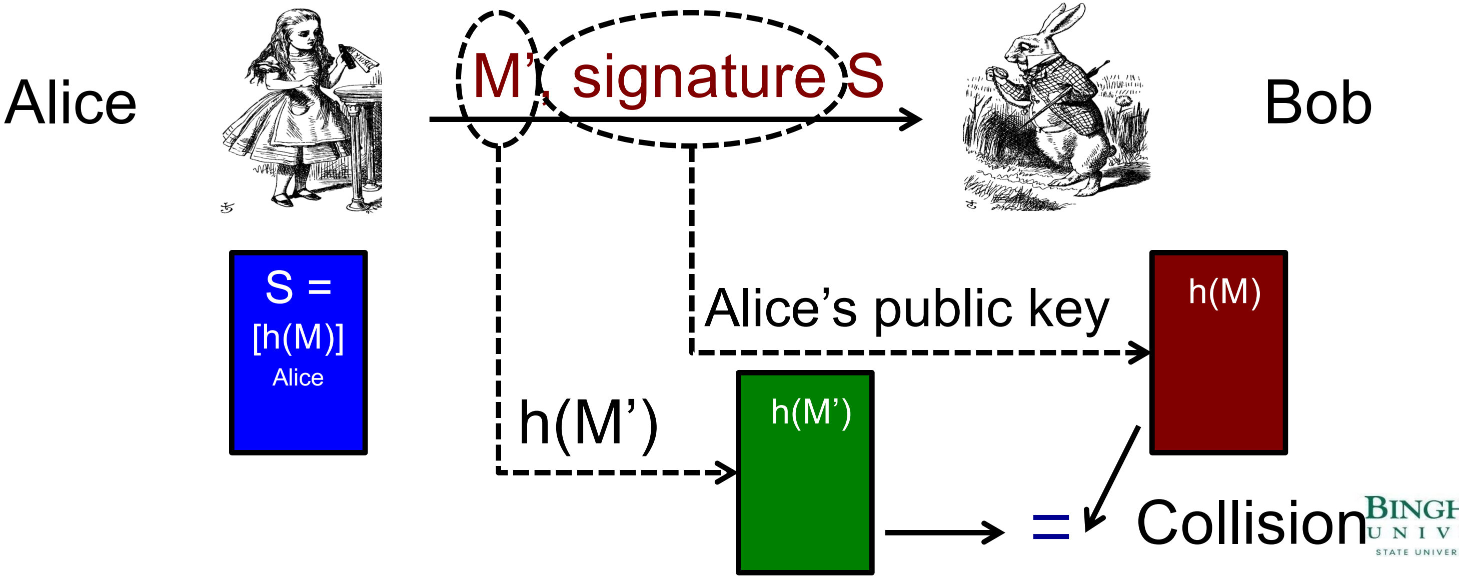
h(M’)

h(M’)

≠

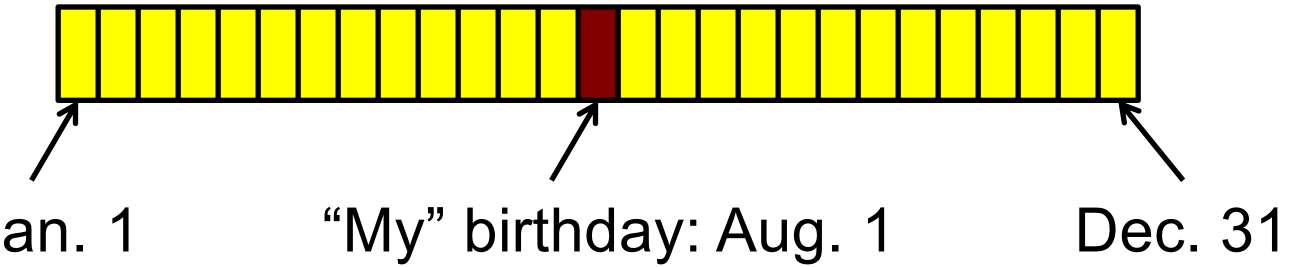
Trudy’s goal is to make h(M’) = h(M)

1. Computes h(M’)
2. Use Alice’s public key to decrypt S
3. If h(M’) ≠ $\{S\}_{Alice}$, false



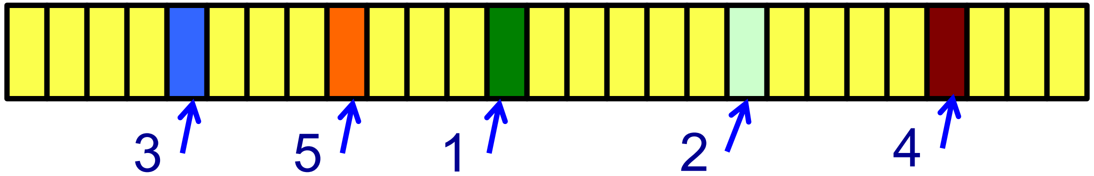
Pre-Birthday Problem

- Suppose N people in a room
- How large must N be before the probability someone has same birthday as mine is $\geq 1/2$?
  - We compute none of the people has the same birthday as you, then subtract that from 1
  - 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?
  - Compute none of any two has the same birthday then subtract from 1
  - $1 - 365/365 \cdot 364/365 \cdot \cdot \cdot (365{-}N{+}1)/365$
  - Set equal to 1/2 and solve: **N = 23**
- Intuition: "**Should** be" about `sqrt(365)` since we compare all **pairs** of x and y
  - `N (N-1) / 2` close to N^2
  - And there are 365 possible birthdays



Of Hashes and Birthdays

- If $h(x)$ output 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 $\mathrm{sqrt}(2^N) = 2^{N/2}$
- **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

Example of Non-crypto Hash

- 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…

Popular Crypto Hashes

- **MD5** — invented by Rivest
  - 128 bit output
  - Note: MD5 collisions easy to find
- **SHA-1** — 160 bit output; designed by NSA
  - Feb 2017: A collision attack found against SHA-1; proof-of-concept shown by publishing two dissimilar PDF files with the same SHA-1 hash
- **SHA-2(SHA-224, SHA-256, …)**: consists of six hash functions with hash values that are 224, 256, 384 or 512 bits; designed by NSA
- **RIPEMD** – Performance close to SHA-1, but less popular
  - RIPEMD-128, RIPEMD-160, RIPEMD-256, RIPEMD-320
- Hashes work by hashing message in blocks

Crypto Hash Design

- Desired property: **avalanche effect**
  - Change to 1 bit of input should affect a large portion of the 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

Trudy’s goal against crypto hash

Weak collision — **given** $\mathbf{x}$ and $\mathbf{h(x)}$, find $\mathbf{y} \neq \mathbf{x}$ such that $\mathbf{h(y) = h(x)}$

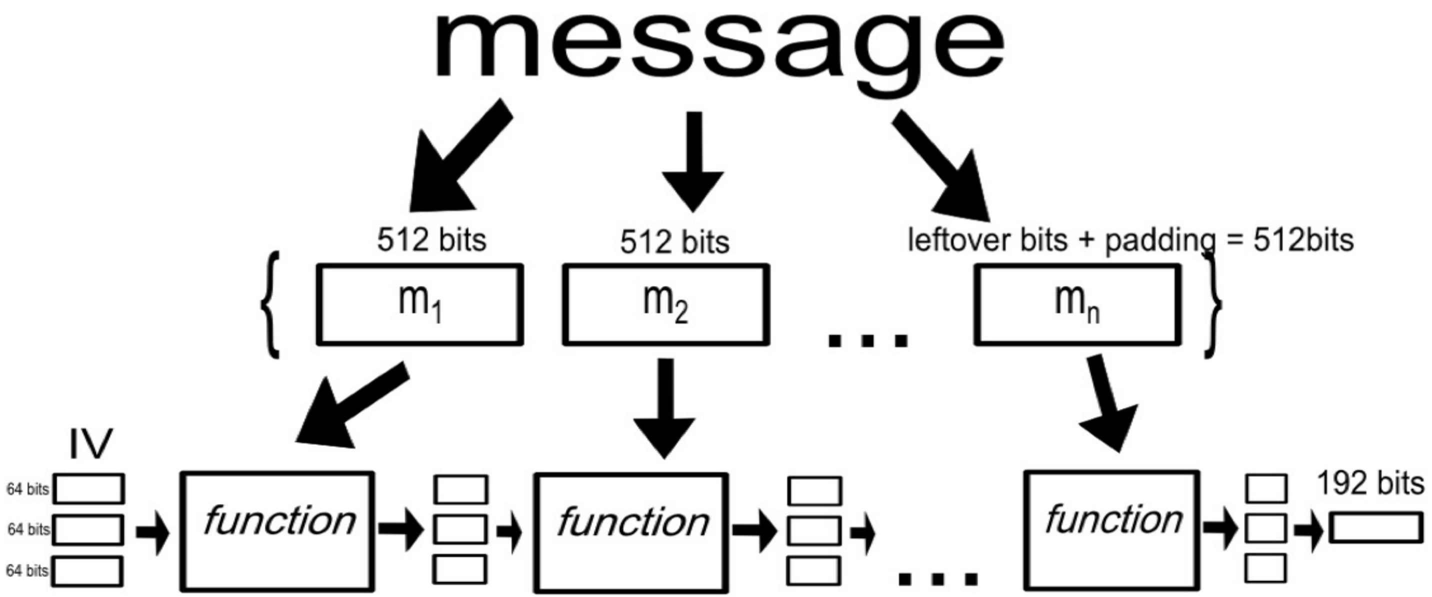
Strong collision — find ***any*** $\mathbf{x}$ and $\mathbf{y}$, with $\mathbf{x} \neq \mathbf{y}$ such that $\mathbf{h(x) = h(y)}$

Tiger Hash

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
  - Why quote here? Because no secret key involved.
  - Applied to input block

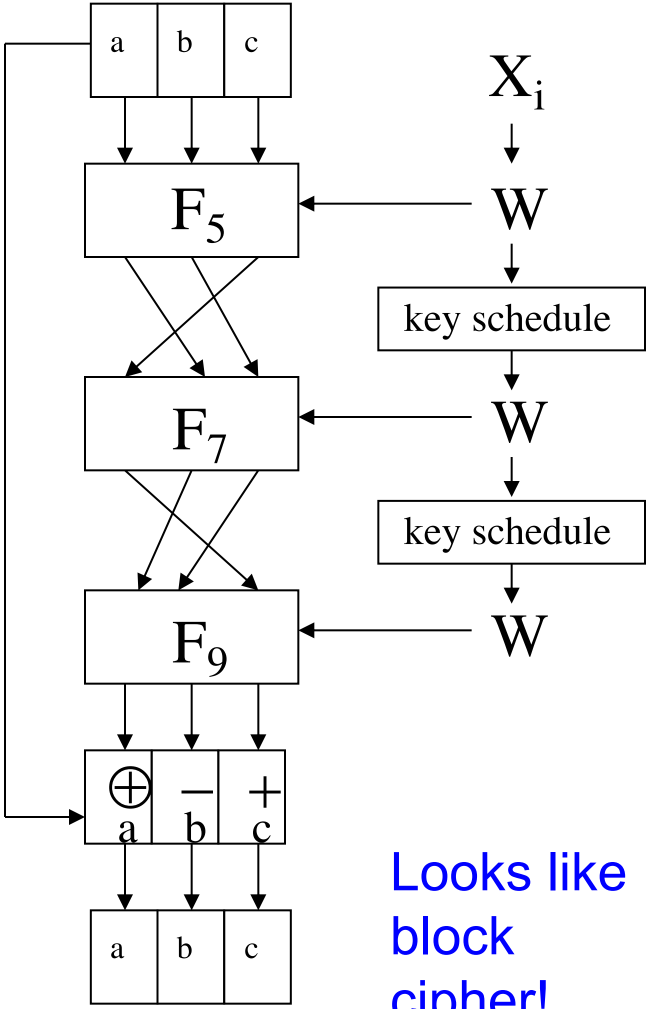
Tiger hash overview



Similar to MAC based on CBC residual. Key difference?

No key used here.

Tiger Outer Round

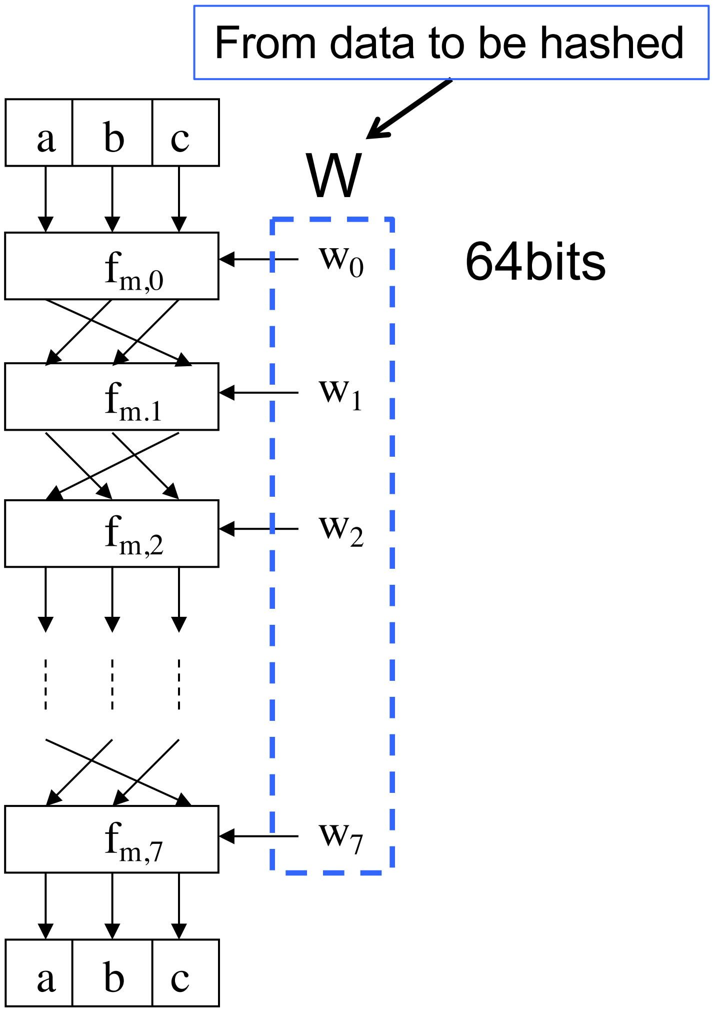


Looks like block cipher!

- Input is X
  - $X = (X_0, X_1, \ldots, X_{n-1})$
  - X is padded
  - Each X_i is 512 bits
- There are n iterations of diagram at left
  - One outer round for each input block
- Initial (a, b, c) constants (64bit)
  - `a = 0x0123456789ABCDEF`
  - `b = 0xFEDCBA9876543210`
  - `c = 0xF096A5B4C3B2E187`
- Final (a, b, c) is hash

Tiger Inner Rounds

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



8 * 64 bits = 512 bits

Tiger Hash: One Inner 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 (64bits)
  - And w_i is 64-bit block of 512 bit W
  - Subscript m is multiplier
  - And $c = (c_0, c_1, \ldots, c_7)$, so each c_i is a single byte
- **Output of $f_{m,i}$ is**
  - $c = c \oplus w_i$
  - $a = a - (S_0[c_0] \oplus S_1[c_2] \oplus S_2[c_4] \oplus S_3[c_6])$
  - $b = b + (S_3[c_1] \oplus S_2[c_3] \oplus S_1[c_5] \oplus S_0[c_7])$
  - $b = b * m$
- Each S_i is **S-box**: 8 bits mapped to 64 bits

Tiger Hash "Key Schedule"

- **Input** is X (512 bits)
  - $X = (x_0, x_1, \ldots, x_7)$
  - Each x_i has 64 bits
- **Goal**: small change in X will produce large change in key schedule output

$$
\begin{aligned}
x_0 &= x_0 - (x_7 \oplus \mathrm{0xA5A5A5A5A5A5A5A5}) \\
x_1 &= x_1 \oplus x_0 \\
x_2 &= x_2 + x_1 \\
x_3 &= x_3 - (x_2 \oplus ((\sim x_1) << 19)) \\
x_4 &= x_4 \oplus x_3 \\
x_5 &= x_5 + x_4 \\
x_6 &= x_6 - (x_5 \oplus ((\sim x_4) >> 23)) \\
x_7 &= x_7 \oplus x_6 \\
x_0 &= x_0 + x_7 \\
x_1 &= x_1 - (x_0 \oplus ((\sim x_7) << 19)) \\
x_2 &= x_2 \oplus x_1 \\
x_3 &= x_3 + x_2 \\
x_4 &= x_4 - (x_3 \oplus ((\sim x_2) >> 23)) \\
x_5 &= x_5 \oplus x_4 \\
x_6 &= x_6 + x_5 \\
x_7 &= x_7 - (x_6 \oplus \mathrm{0x0123456789ABCDEF})
\end{aligned}
$$

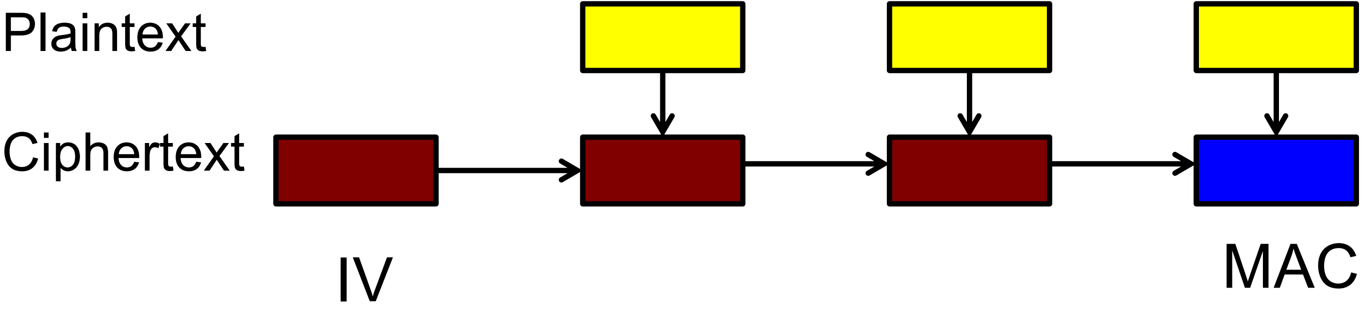
Tiger Hash Summary

- Hash and intermediate values are 192 bits
- 24 (inner) rounds: 3 outer rounds, each having 8 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

HMAC – Hashed MAC

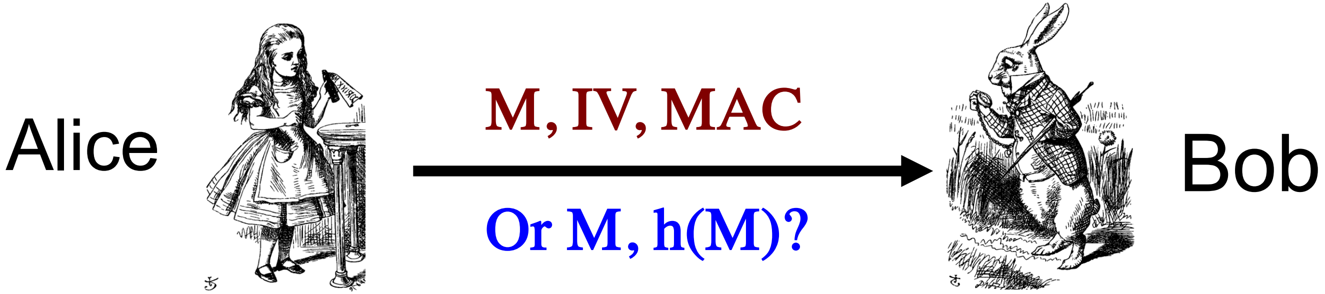
Refresher: MAC

- Message Authentication Code (MAC)
  - Used for data **integrity**
  - Integrity **not** the same as confidentiality
- MAC is computed as **CBC residue**
  - That is, compute CBC encryption, saving only final ciphertext block, the MAC



A shared key is needed

HMAC – Hashed MAC



Alice

M, IV, MAC

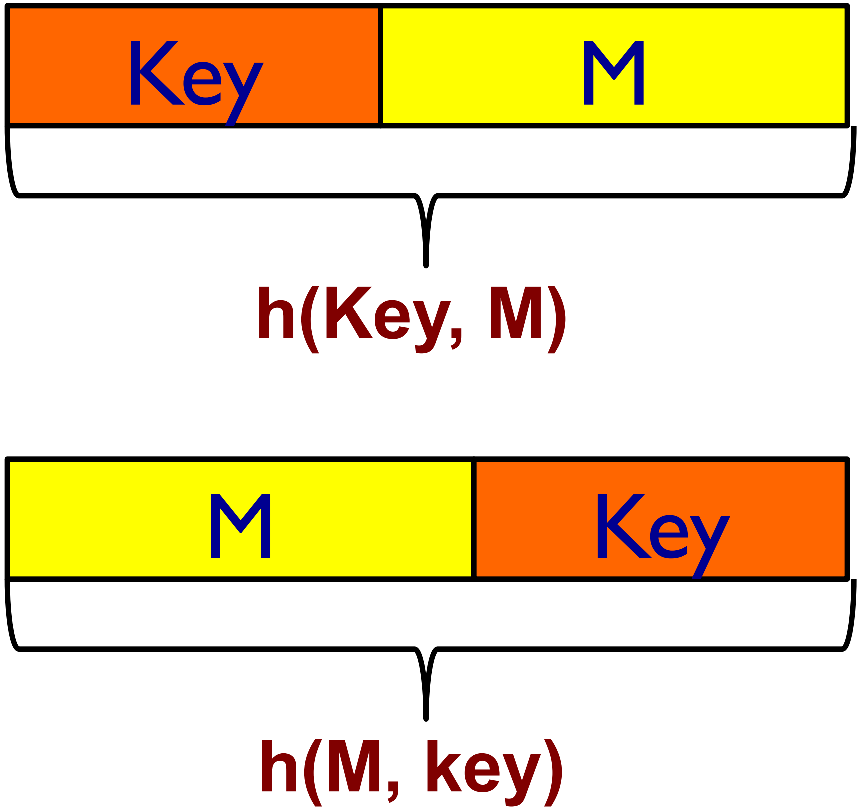
Or M, h(M)?

Bob

- Can we replace IV, MAC with a hashed value (e.g. MD5)?
  - **No, Trudy can replace M with M’, h(M) with h(M’)**
  - **We need a secret shared by Alice and Bob!**
- Maybe send E(h(M), K)?
- Instead of encrypting the hash, one can compute a MAC of the message M **with key K** using a “hashed MAC” or **HMAC**
- HMAC is a **keyed hash**
  - Mix key into M
- How to compute HMAC?

HMAC

- Two choices: **h(K, M)** and **h(M, K).** Which is better?



Prepend key to the message

Key | M

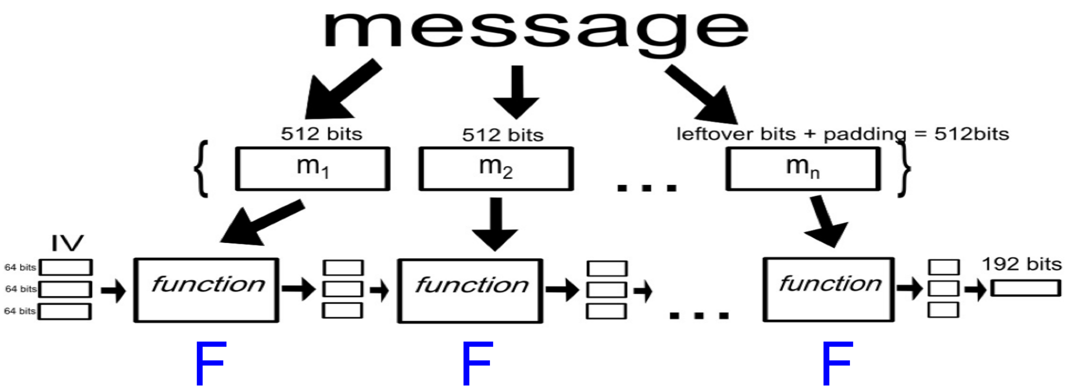
h(Key, M)

Append key to the message

M | Key

h(M, key)

HMAC as prepend h(K, M) with Tiger



- Hashes computed in blocks(512 bits)
  - m1, m2 are 512 bits input, IV(a,b,c), F the 24 rounds function
  - $h(m_1, m_2) = F(F(IV, m_1), m_2)$
  - **For including another block m3:**
  - $h(m_1, m_{2,} m_3) = F(h(m_1, m_2), m_3)$
- Assume that Trudy sees M and h(K, M). Let M’ = (M, X).
  - Then h(K, M’) = h(K, M, X) = F(h(K, M), X)
  - **h(K,M) is known, use it as input for F**
  - **Trudy can create M’ and compute HMAC of M’ without K**

BINGHAMTON
UNIVERSITY
STATE UNIVERSITY OF NEW YORK

Is $h(M, K)$ better for HMAC?

- Yes, it does prevent previous attack
  - But…

- If $h(M') = h(M)$ (known collision) then we might have

$$\begin{aligned} h(M, K) &= F(h(M), K) \\ &= F(h(M'), K) \\ &= h(M', K) \end{aligned}$$

The Right Way to HMAC

- More sophisticated way to mix the key into the data
  - Prepend, append are too simple
- Let B be the block length of hash, in bytes
  - $\mathrm{B} = 64$ (512bits) for MD5 and SHA-1 and Tiger
- $\mathbf{ipad} = 0\mathrm{x}36$ repeated B times, form a 512bits output
- $\mathbf{opad} = 0\mathrm{x}5\mathrm{C}$ repeated B times, form a 512bits output
- If K is less than 64 bytes, then append 0's behind it
- Then

$$\mathrm{HMAC}(\mathrm{M}, \mathrm{K}) = \mathrm{h}(\mathrm{K} \oplus \mathrm{opad}, \mathrm{h}(\mathrm{K} \oplus \mathrm{ipad}, \mathrm{M}))$$

Hash Usage – 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
- But there is a flaw here
  - Forward search attack