CSE 484 / CSE M 584: Computer Security and Privacy

Cryptography:
Hash Functions, MACs (finish)
Asymmetric Cryptography (start)

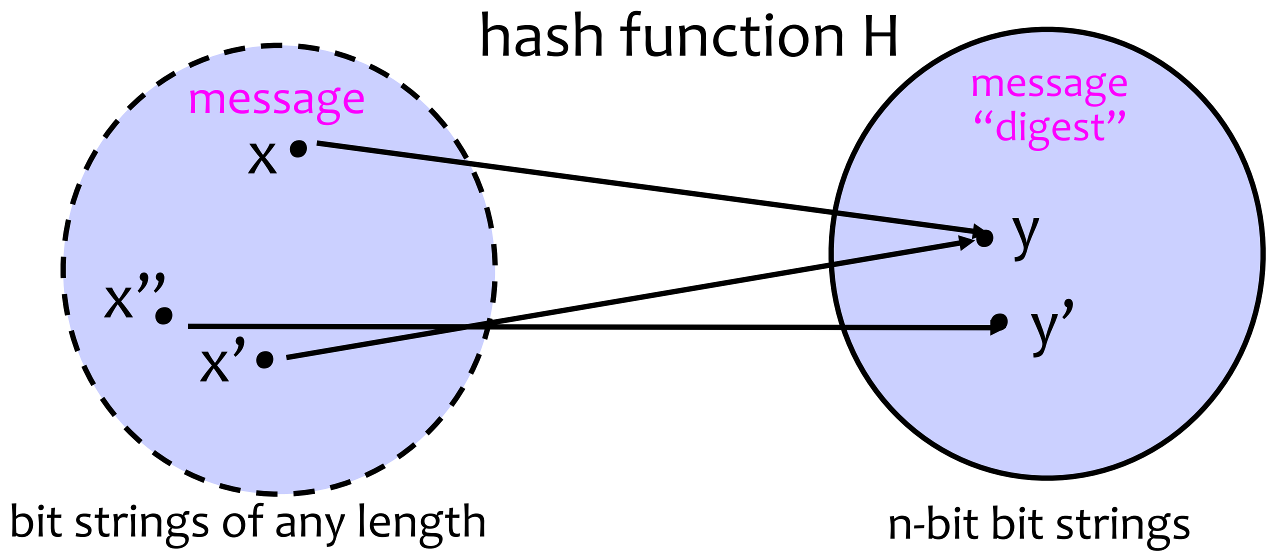
Fall 2016

Ada (Adam) Lerner
lerner@cs.washington.edu

Thanks to Franzi Roesner, Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Hash Functions

Hash Functions: Main Idea



- Hash function H is a lossy compression function
  - Collision: h(x)=h(x') for distinct inputs x, x'
- H(x) should look "random"
  - Every bit (almost) equally likely to be 0 or 1
- Cryptographic hash function needs a few properties…

Properties of a Cryptographic Hash Function

- One-wayness
  - Given h(x): hard to find x
- Collision resistance
  - Hard to find x ≠ x' s.t. h(x) == h(x')
- Weak collision resistance
  - Hard to find x ≠ x' s.t. h(x) == h(x') **for specific, random x**

Properties of a Cryptographic Hash Function

- One-wayness
  - Hard to find inputs that match outputs
- Collision resistance
  - Hard to find 2 inputs with the same hash
- Weak collision resistance
  - If I give you a random input, it's hard to find another input with the same hash.

Property 1: One-Way

- The hash should be hard to invert
  - "Preimage resistance"
  - Let $h(x') = y \in \{0,1\}^n$ for a random x'
  - Given y, it should be hard to find any x such that $h(x)=y$

Property 2: Collision Resistance

- Should be hard to find $x \neq x'$ such that $h(x)=h(x')$
- Birthday paradox means that brute-force collision search is only $O(2^{n/2})$, *not* $O(2^n)$
  – For SHA-1, this means $O(2^{80})$ vs. $O(2^{160})$

One-Way vs. Collision Resistance

- One-wayness does <u>not</u> imply collision resistance
  - Suppose g is one-way
  - Define h(x) as g(x') where x' is x with last bit removed
    - h is one-way (to invert h, must invert g)
    - Collisions for h are easy to find: for any x, h(x|0)=h(x|1)

One-Way vs. Collision Resistance

- One-wayness does not imply collision resistance
- Collision resistance does not imply one-wayness
  - Exercise for the reader (on HW#2)

Property 3: Weak Collision Resistance

- Given randomly chosen x, hard to find x' such that h(x)=h(x')
  - Attacker must find collision for a specific x.
    (By contrast, to break collision resistance it is enough to find any collision.)
  - Brute-force attack requires $O(2^n)$ time
- Weak collision resistance does not imply collision resistance.

Does Collision Resistance imply Weak Collision Resistance?

- p: Hard to find $x \neq x'$ such that $h(x)=h(x')$
- q: Random x, hard to find x' s.t. $h(x)=h(x')$

Does Collision Resistance imply Weak Collision Resistance?

- p: Hard to find $x \neq x'$ such that $h(x)=h(x')$
- q: Random x, hard to find x' s.t. $h(x)=h(x')$

- Contrapositive: $p \rightarrow q$ same as $\neg q \rightarrow \neg p$
- If you can find a collision against a random x, can you find a collision in general?

Does Collision Resistance imply Weak Collision Resistance?

- p: Hard to find $x \neq x'$ such that $h(x)=h(x')$
- q: Random x, hard to find x' s.t. $h(x)=h(x')$

- If h is not weakly collision resistant, then there exists an algorithm which takes an input x and "quickly" finds x' which collides
- Call this adversarial algorithm A, so for random x, A(x) = x' (where x' collides with x)

Does Collision Resistance imply Weak Collision Resistance?

- p: Hard to find $x \neq x'$ such that $h(x)=h(x')$
- q: Random x, hard to find x' s.t. $h(x)=h(x')$

Hashing vs. Encryption

- Hashing is one-way. There is no "un-hashing"
  - A ciphertext can be decrypted with a decryption key… hashes have no equivalent of "decryption"
- Hash(x) looks "random" but can be compared for equality with Hash(x')
  - Hash the same input twice → same hash value
  - Encrypt the same input twice → different ciphertexts
- Crytographic hashes are also known as "cryptographic checksums" or "message digests"

Application: Password Hashing

- Instead of user password, store hash(password)
- When use submits password, hash it and compare to the stored hash

- User "alice" sets password "thisismypassword"
- Server stores hash("thisismypassword") = a0e863aba2d508b6e4744f07d7c260cd

- When alice logs in, server hashes the server she provides and compares it to the stored hash.

Application: Password Hashing

- Instead of user password, store hash(password)
- When use submits password, hash it and compare to the stored hash

- Let's say you break into a server and steal 100 million password hashes. What are the problems with this server's approach? (Q1)

Application: Password Hashing

- How to store password hashes: salt and hash.

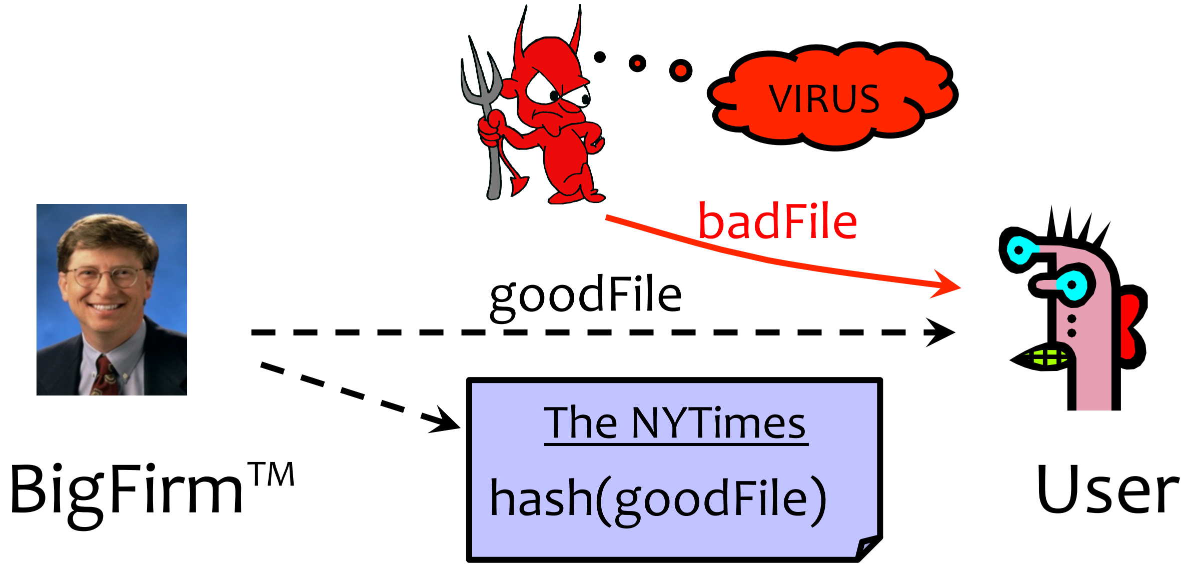
- Instead of storing hash(password), store “salt, hash(salt | password)”

- Salt is a random value **per password**

Application: Password Hashing

- **Username**: `ahaha`
- **Hash function**: `SHA512`
- **Salt**: `FlItSjGy`
- **Hashed password**: `54IaMBy6ThxAbvnUztWzrl4FjtEwn1sX81/8Ll7PtMpPAiy57QM4q.oyUD2cHFL4nwhguDk7eP7c3t0ArKep.`

Application: Software Integrity



Goal: Software manufacturer wants to ensure file is received by users without modification.

Idea: given goodFile and hash(goodFile), very hard to find badFile such that hash(goodFile)=hash(badFile)

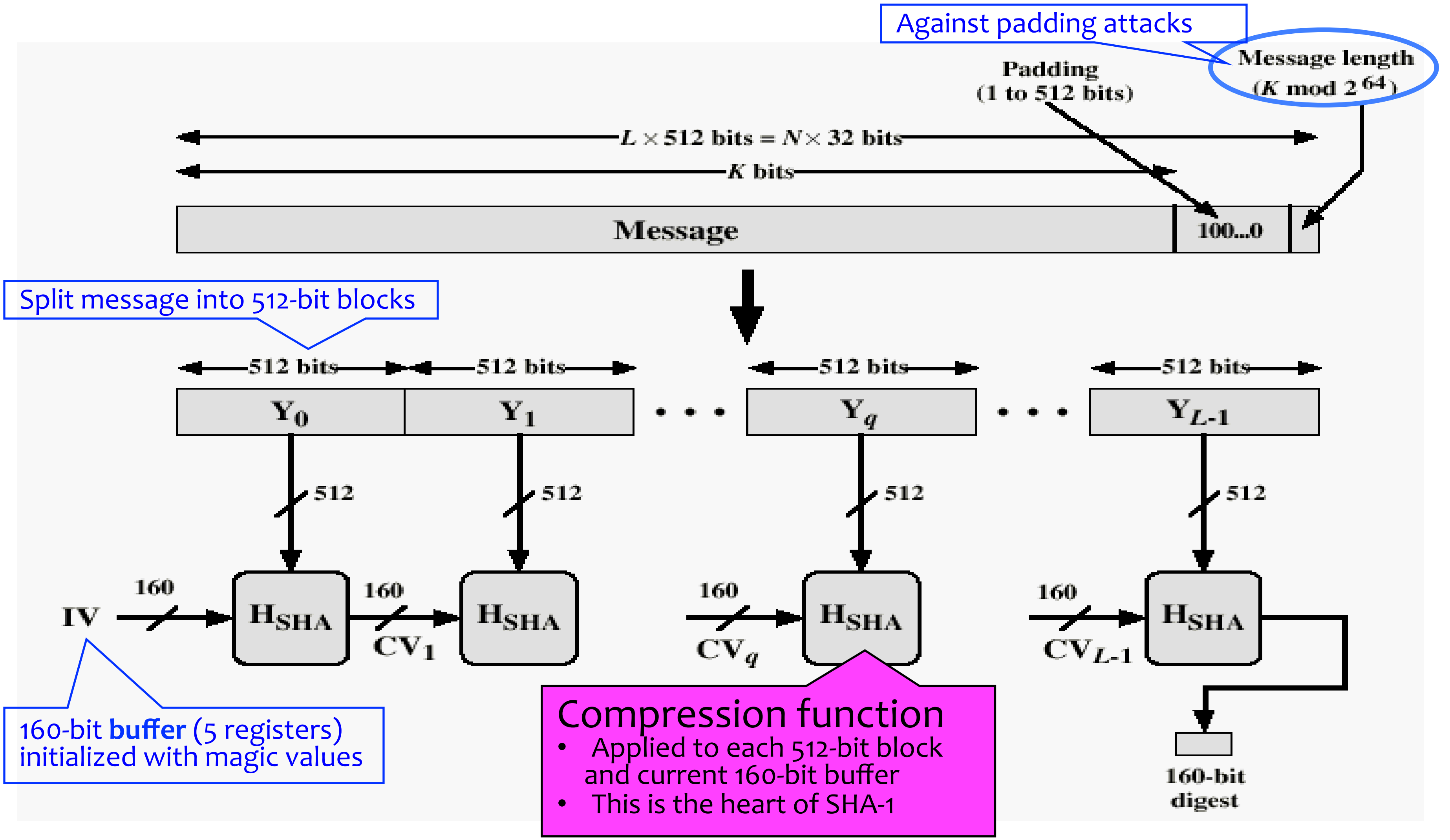
Which Property Do We Need?

- Auction bidding
  - Alice wants to bid B, sends H(B), later reveals B
  - One-wayness: rival bidders should not recover B (this may mean that she needs to hash some randomness with B too)
  - Collision resistance: Alice should not be able to change her mind to bid B' such that H(B)=H(B')

Common Hash Functions

- MD5
  - 128-bit output
  - Designed by Ron Rivest, used very widely
  - Collision-resistance broken (summer of 2004)
- RIPEMD-160
  - 160-bit variant of MD5
- SHA-1 (Secure Hash Algorithm)
  - 160-bit output
  - US government (NIST) standard as of 1993-95
  - Also recently broken! (Theoretically -- not practical.)
- SHA-256, SHA-512, SHA-224, SHA-384
- SHA-3: standard released by NIST in August 2015

Basic Structure of SHA-1 [FYI only]

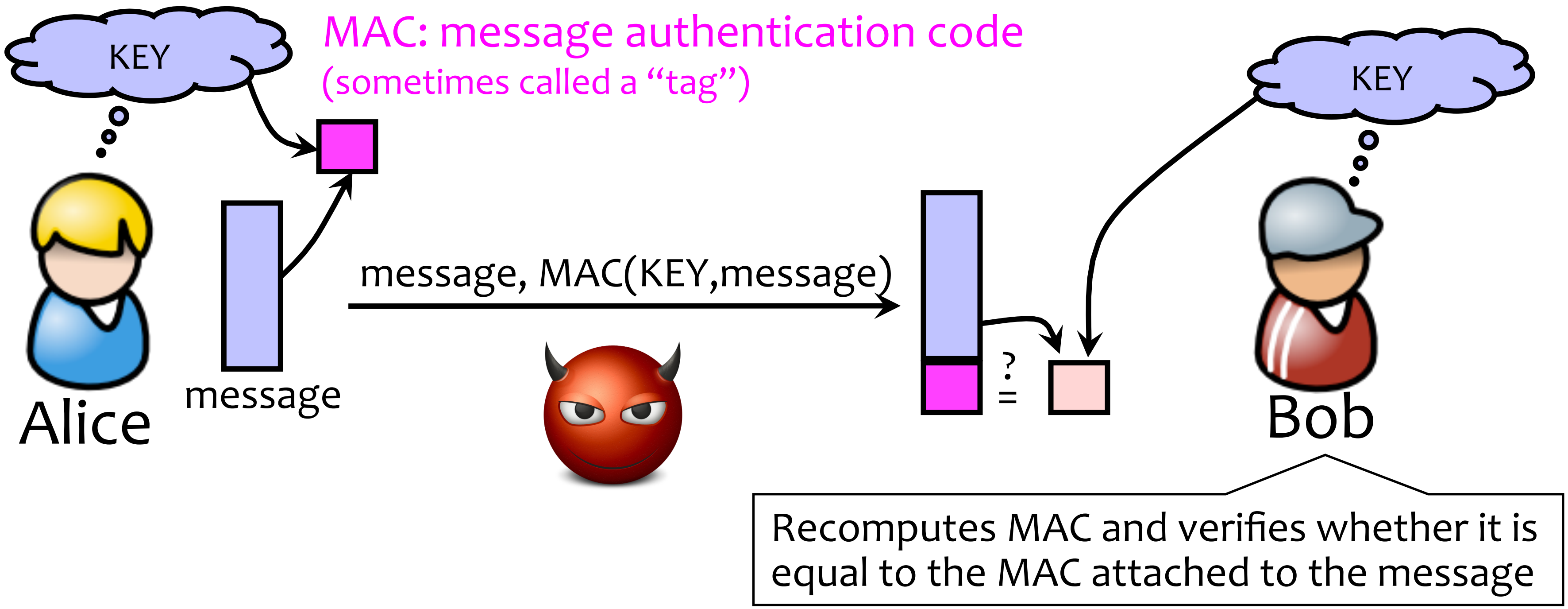


How Strong is SHA-1?

- Every bit of output depends on every bit of input
  - Very important property for collision-resistance
- Brute-force inversion requires 2^{160} ops, birthday attack on collision resistance requires 2^{80} ops
- Some weaknesses, e.g., collisions can be found in 2^{63} ops (2005)

Recall: Achieving Integrity

Message authentication schemes: A tool for protecting integrity.

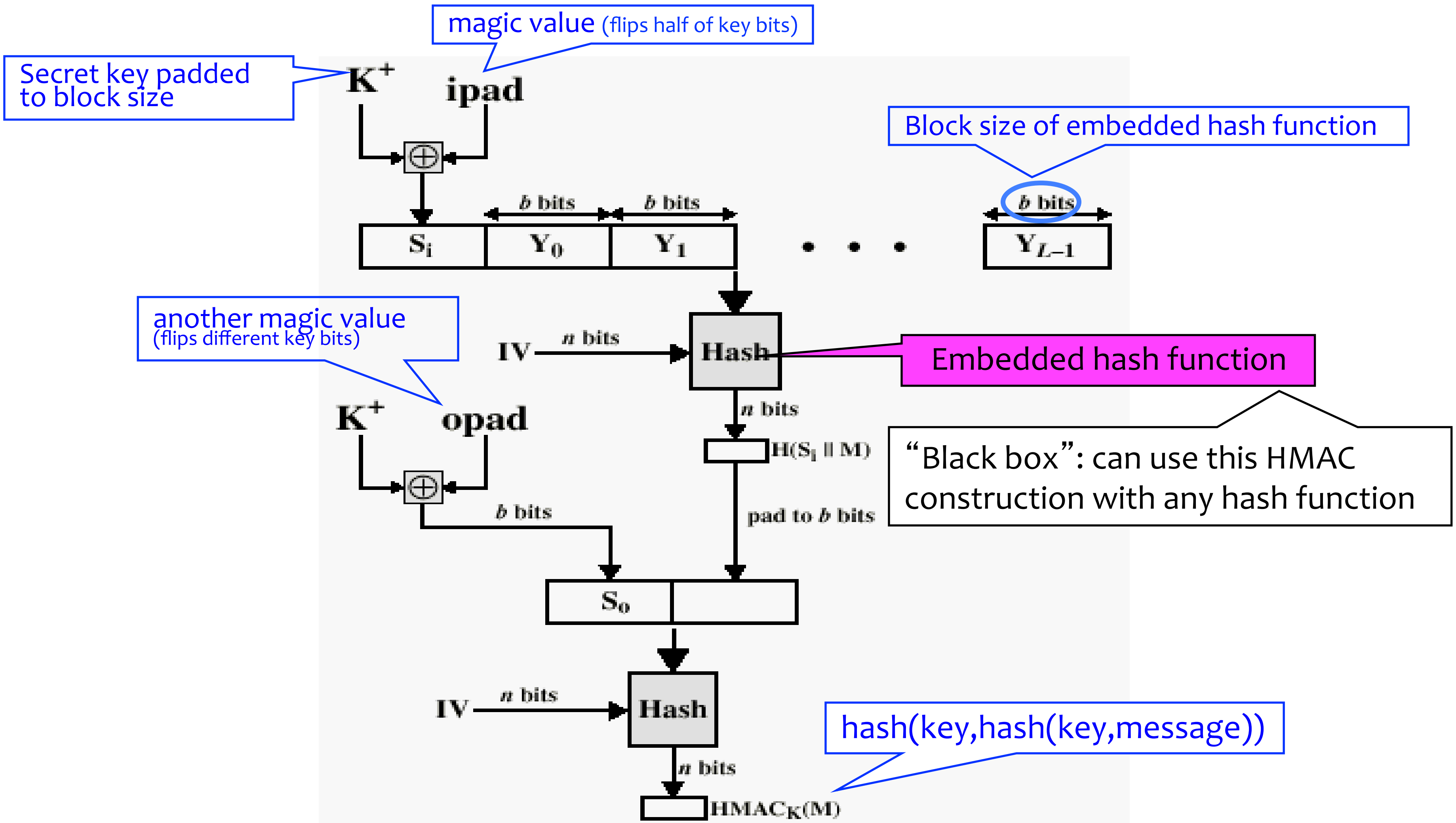


Integrity and authentication: only someone who knows KEY can compute correct MAC for a given message.

HMAC

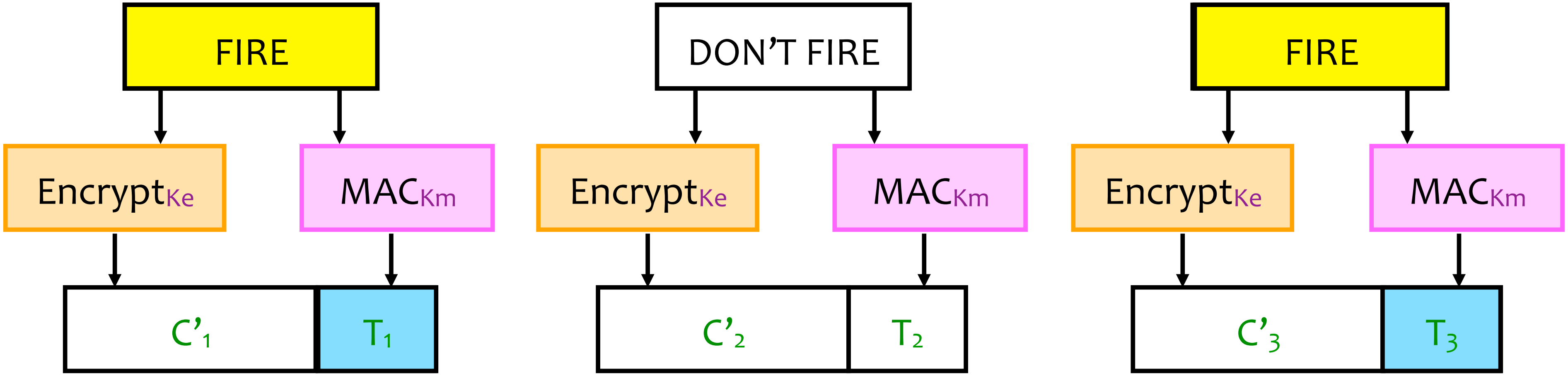
- Construct MAC from a cryptographic hash function
  - Invented by Bellare, Canetti, and Krawczyk (1996)
  - Used in SSL/TLS, mandatory for IPsec
- Why not encryption?
  - Hashing is faster than block ciphers in software
  - Can easily replace one hash function with another
  - There used to be US export restrictions on encryption

Structure of HMAC [FYI only]



Authenticated Encryption

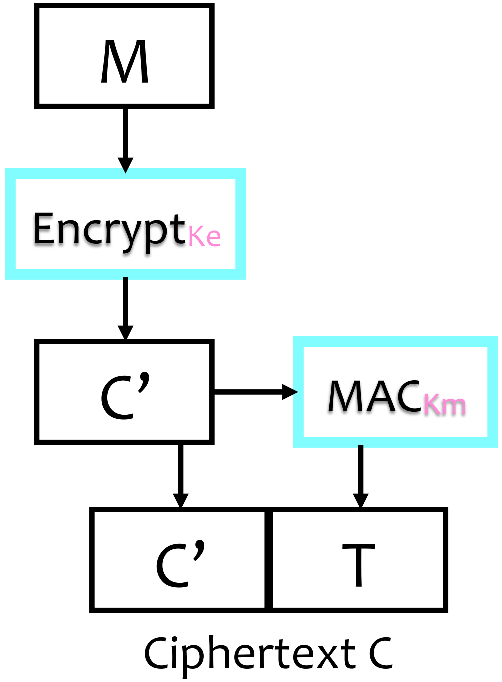
- What if we want both privacy and integrity?
- Natural approach: combine encryption scheme and a MAC.
- But be careful!
  - Obvious approach: Encrypt-and-MAC
  - Problem: MAC is deterministic! same plaintext → same MAC



Authenticated Encryption

- Instead:
  Encrypt *then* MAC.

- (Not as good:
  MAC-then-Encrypt)



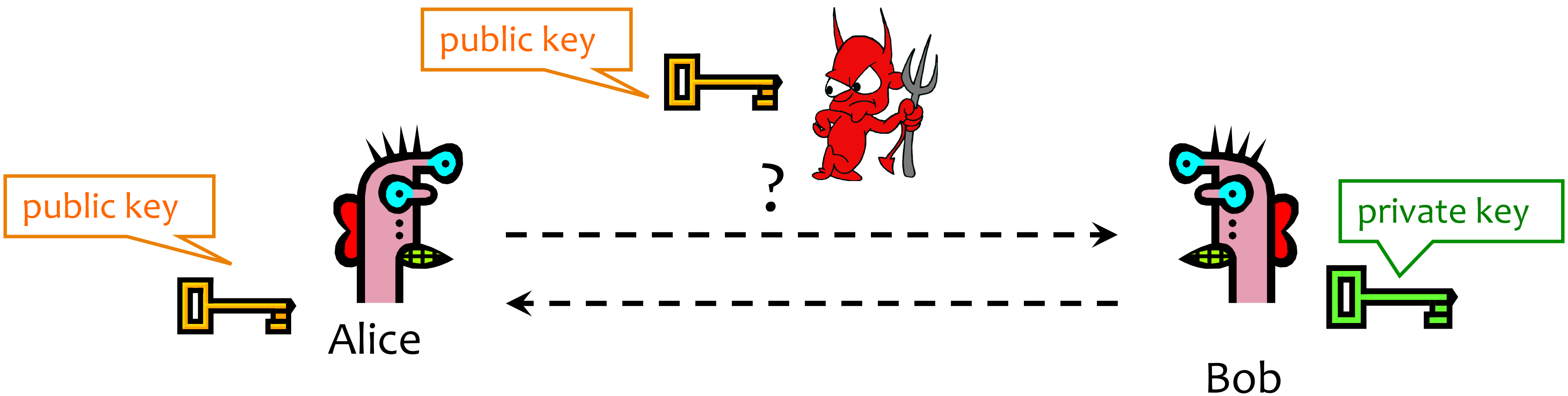
Encrypt-then-MAC

Asymmetric (Public Key) Cryptography

Reminder: Symmetric Cryptography

- **1 secret key (or 2 or …)**, shared between sender/receiver
- Repeat fast and simple operations lots of times (rounds) to mix up key and ciphertext
- **Why do we think it is secure?** (simplistic)
  - Lots of heuristic arguments
    - If we do lots and lots and lots of mixing, no simple formula (and reversible) describing the whole process (cryptographic weakness).
    - Mix in ways we think it's hard to short-circuit all the rounds. Especially non-linear mixing, e.g., S-boxes.
  - Some math gives us confidence in these assumptions

Public Key Crypto: Basic Problem



Given: Everybody knows Bob's public key
Only Bob knows the corresponding private key

Goals: 1. Alice wants to send a secret message to Bob
2. Bob wants to authenticate himself

Public Key Cryptography

- Everyone has **1 private key and 1 public key**
  - Or 2 private and 2 public, when considering both encryption and authentication
- Mathematical relationship between private and public keys
- **Why do we think it is secure?** (simplistic)
  - Relies entirely on **problems we believe are "hard"**

What can Public Key Crypto Do?

- Encryption for confidentiality
  - Anyone can encrypt a message only you can read
- Digital signatures for authentication
  - Can "sign" a message with your private key

Session Establishment

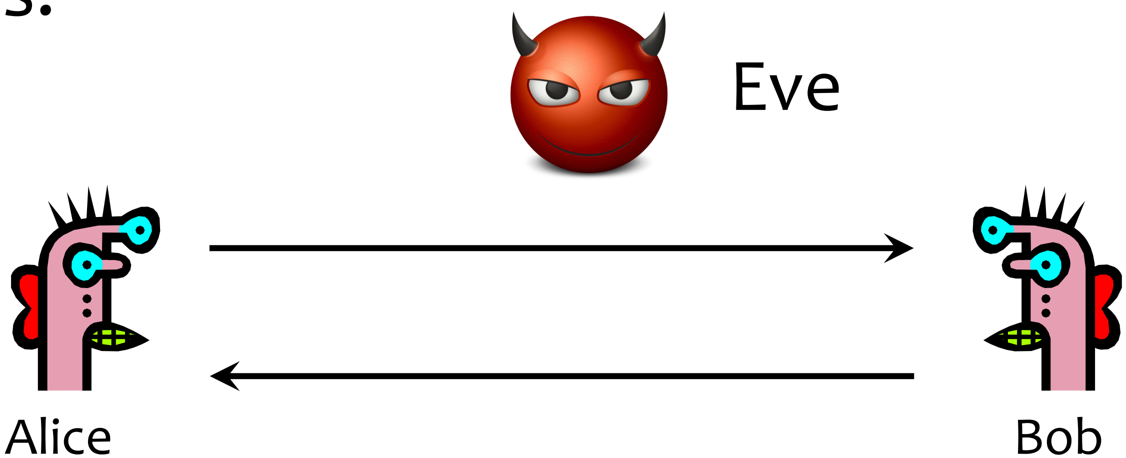
- Session key establishment
  - Exchange messages to create a secret session key
  - Then switch to symmetric cryptography (why?)

Refresher: Modular Arithmetic

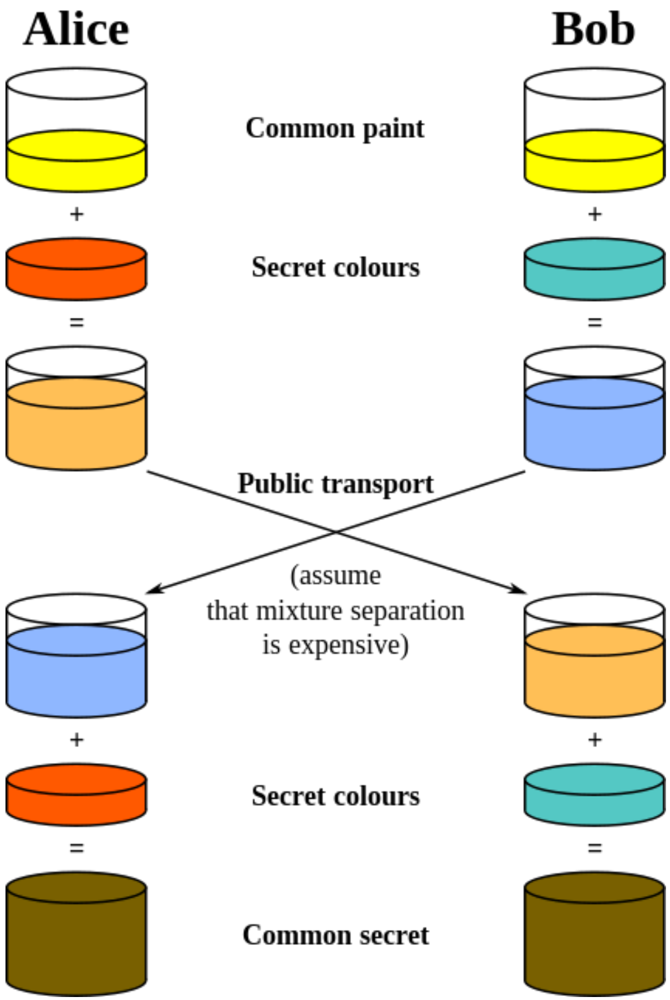
(see worksheet Qs 3-5)

Diffie-Hellman Protocol (1976)

- Alice and Bob never met and share no secrets
- They talk publically, with everything they say overheard by Eve. By the end of the conversation, they share a secret nobody else knows.



Diffie-Hellman: Conceptually



Common paint: p and g

Secret colors: x and y

Send over public transport:
g^x mod p
g^y mod p

Common secret: g^{xy} mod p

[from Wikipedia]