Integrity (Crypto Lecture #1)

UNIVERSITY OF
MICHIGAN

Administrivia

Office hours

SSD forms

Project one: released next week, watch Piazza for announcement

- Find your partner!

Homework 1 due on Friday, September 14, 6 PM

- Don’t be late!
- Submit w/ Canvas

Submission is through Gradescope, **must be a PDF**

Cryptography?

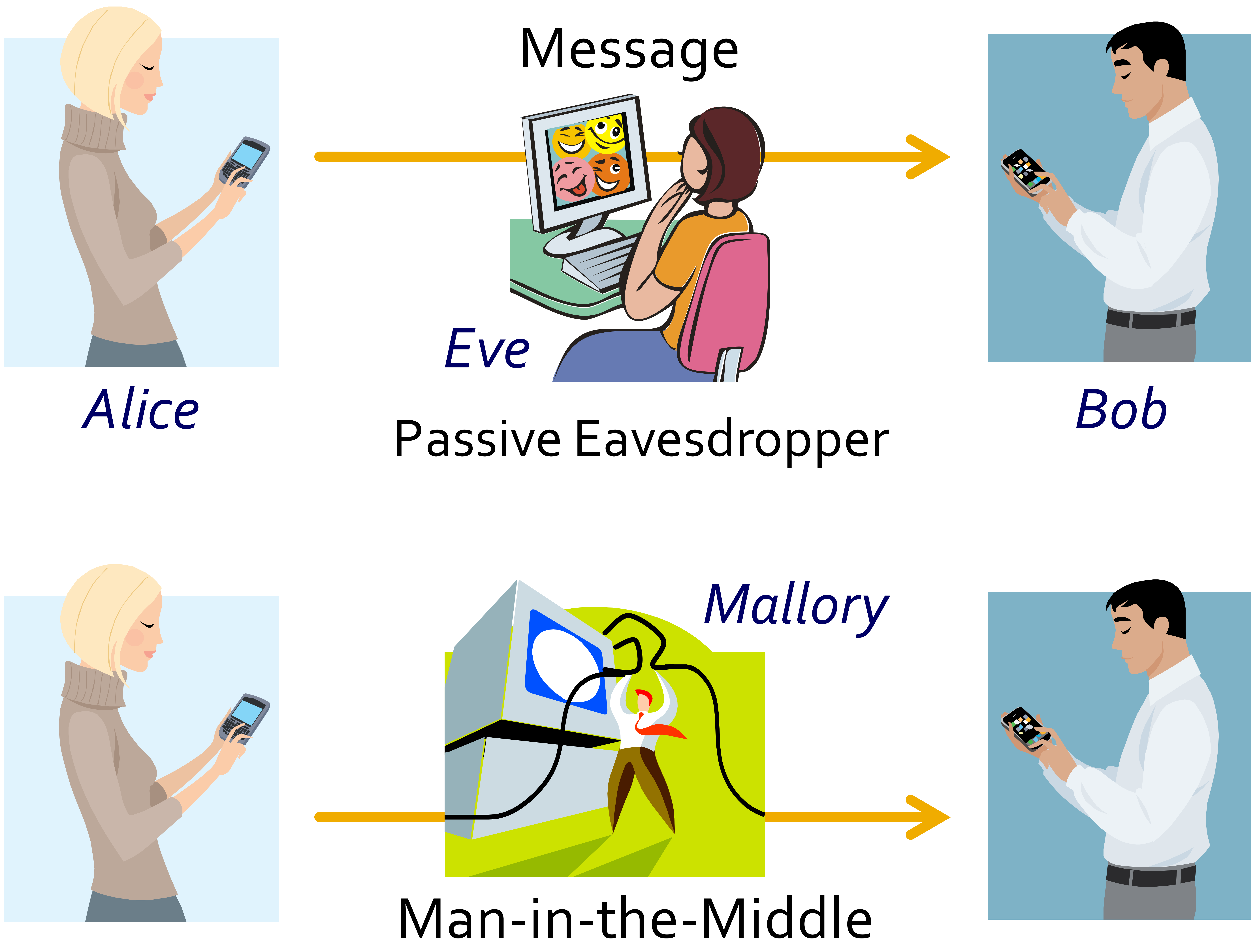
What is cryptography? $f: \{0,1\}^n \to \{0,1\}^m$

invertible

Why study cryptography?

What do we want to use cryptography to defend against?

What properties does cryptography provide / what properties do we want from the cryptography that we use?


Message
Alice
Eve
Passive Eavesdropper
Bob
Mallory
Man-in-the-Middle

Properties of a Secure Channel

Confidentiality defeat Eve

Integrity defeat Mallory

Authentication how does Bob know the message from Alice

Message Integrity

An attacker cannot modify messages without being detected.

Goal: Message Integrity

Alice wants to send message ***m*** to Bob, *but…*

- We don’t trust the messenger
- I.e., we don’t trust the network

Want to be sure **what Bob receives** is actually **what Alice sent**.

Goal: Message Integrity

Threat Model

- Mallory can see, modify, and forge messages
- Mallory wants to trick Bob into accepting a message Alice didn't send

man-in-the-middle

One Approach…

1. Alice computes ***v := f(m)***

2. E.g. ***m*** = "Attack at dawn", ***f(m)*** = 628369867…

3. Bob verifies that **v' = f(m')**, and accepts the message if and only if this is true.

Function *f* ?

Bob accepts the message iff $\mathbf{v'} = \boldsymbol{f}(\mathbf{m'})$.

We want ***f*** to be easily computable by Alice and Bob, but **not** computable by Mallory.

(As we shall see) we lose if Mallory can learn $\boldsymbol{f(x)}$ for any $\boldsymbol{x \neq m}$!

Candidate: Random Function

Input: Any size up to huge maximum

Output: Fixed size (e.g. 256 bits)

Defined by a giant lookup table that's filled in by flipping coins.

0	→	0011111001010001…
1	→	1110011010010100…
2	→	0101010001010000…

Random Function

Looks pretty **secure**

Mallory can’t do better than random guessing

Looks pretty **impractical**

0	→	0011111001010001…
1	→	1110011010010100…
2	→	0101010001010000…

Candidate: Pseudorandom Function (PRF)

Want a function that's practical but "looks random"…

Building a PRF

Start with a big family of functions $\boldsymbol{f_0()}$, $\boldsymbol{f_1()}$, $\boldsymbol{f_2()}$, … all known to Mallory

f_i:$\{0,1\}^n \rightarrow \{0,1\}^n$

Let $\boldsymbol{g()}$ be $\boldsymbol{f_k()}$, where $\boldsymbol{k}$ is a secret value (or “key”) known only to Alice and Bob

$\boldsymbol{k}$ is n bits, chosen randomly

Kerchoff’s Principle: *A cryptosystem should be secure even if everything about the system, <u>except the key</u>, is public knowledge*

Formal Definition of a Secure PRF

Mallory can always win slowly!

1. We flip a coin secretly to get bit ***b***

2. If ***b = 0***, let g be a random function
   If ***b = 1***, let ***g =*** $\boldsymbol{f_k}$, where ***k*** is a randomly chosen secret.

3. Mallory chooses ***x***; we announce ***g(x)***. Repeat as often as M likes.

4. Mallory guesses ***b*** *quickly*

Security Definition: *We say* ***g()*** *is a secure PRF if Mallory can't select* **b** *any better than random guessing*

PRF for Alice and Bob

1. Let ***g()*** = ***f_k()*** be a secure PRF, where ***k*** is a random key known only to Alice and Bob.

2. Alice computes **v** := ***f_k***(**m**).

3. Bob verifies **v′** = ***f_k***(**m′**), accepts if and only if this is true.

Annoying Question!

Q: Do PRF's actually exist?

A: We don't know.

Best We Can Do: *Well studied functions where we haven't spotted a problem yet (e.g. HMAC-SHA256)*

Jargon

Message Authentication Code (MAC): *Effectively the same thing as a PRF*

Currently popular "PRF" (we hope!): *HMAC-SHA256*

$$\mathrm{SHA256}\left(k \oplus c_1 \,\|\, \mathrm{SHA256}\left(k \oplus c_2 \,\|\, m\right)\right)$$

SHA-256 Function:
N bits in,
256 bits out

XOR

Concatenation

Standard
Constant

Hash Functions, e.g., SHA-256

Input: Arbitrary Length Data

Output: Fixed-size digest (n bits)

No key, fixed function

Examples: ~~MD5~~, ~~SHA-1~~, SHA-256, SHA-512, SHA-3

Good Hash Functions

Collision Resistance: Hard to find pair of input ***x***, ***x'*** such that ***H(x) = H(x')***

Preimage Resistance: Given y, hard to find **x'** such that ***H(x') = y***

Second Preimage Resistance: Given **x**, hard to find an ***x'*** such that ***H(x) = H(x')***

Other Hash Functions

MD5

- Once ubiquitous
- Broken in 2004
- Now it’s easy to find collisions (pairs of messages with the same MD5 hash)
- Exploited to attack real systems
- Project 1

(More) Other Hash Functions

SHA-1

- Fairly widely used
- Started to be unsupported in HTTPS in Jan 2016
- Turns out getting the Internet to change is hard
- Don't use in new applications!
- Might be project 1 in the future!

The Register®
Biting the hand that feeds IT

DATA CENTER SOFTWARE **SECURITY** TRANSFORMATION DEVOPS BUSINESS PERSONAL TECH

Security 113

'First ever' SHA-1 hash collision calculated. All it took were five clever brains... and 6,610 years of processor time

Tired old algo underpinning online security must die now

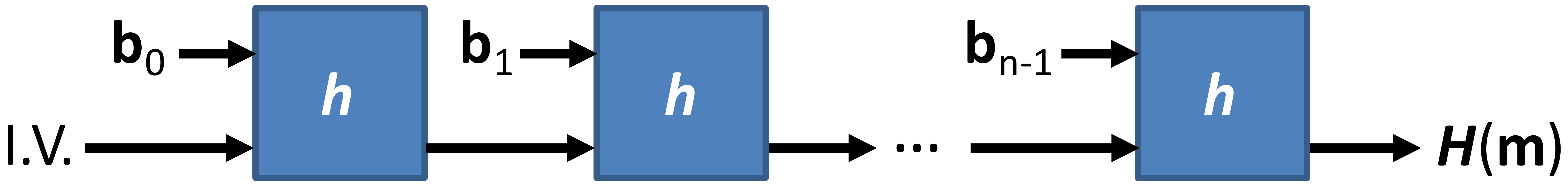
By John Leyden, Thomas Claburn and Chris Williams 23 Feb 2017 at 18:33

Constructing SHA-256

<u>Input</u>: Arbitrary Length Data
<u>Output</u>: 256-bit digest

Built with "compression function" ***h*** using Merkle–Damgård construction

- (256 bits, 512 bits) -> 256 bits out
  Designed to be "really hairy", not going into details



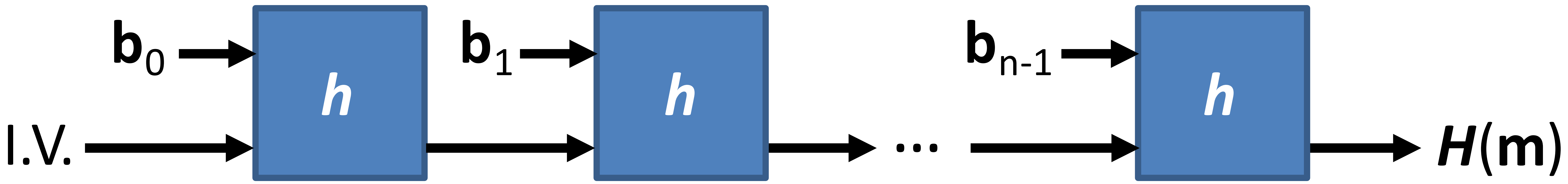
SHA-256 Algorithm

1. Pad input $\boldsymbol{m}$ to multiple of 512 bits (using a fixed algorithm) *[why?]*

2. Split into 512-bit blocks $\mathbf{b}_0$, $\mathbf{b}_1$, … $\mathbf{b}_{n-1}$

3. $\mathbf{y}_0$ =constant initialization vector., $\mathbf{y}_1 = \boldsymbol{h}(\mathbf{y}_0,\mathbf{b}_0)$, …, $\mathbf{y}_i = \boldsymbol{h}(\mathbf{y}_{i-1},\mathbf{b}_{i-1})$

4. Return $\boldsymbol{y}_n$



MAC / PRF / Hash / Oh My!

Message Authentication Code (MAC): *Effectively the same thing as a PRF*

Cryptographic Hash Function: *Not a strong PRF, but can be used to construct a MAC/PRF (e.g. HMAC). Example: SHA-256.*

Stand alone, a hash function is (usually) vulnerable to <u>length extension attacks</u>

- Given **z = H(m)** for some unknown m, calculate **H(m || padding || v)** for attacker selected **v**. You'll do this in Project 1.

So Far

The Security Mindset

What is Cryptography?

Message Integrity

Next Week…

Confidentiality