Lecture 20: Message Integrity (contd), Pseudorandomness

Professor Adam Bates
CS 461 / ECE 422
Fall 2019

Goals for Today

- Learning Objectives:
  - Briefly discuss midterm results
  - Conclude discussion of Message Integrity
- Announcements, etc:
  - Midterm grades released on Compass2G as of this morning.
  - MP3 Checkpoint #1: **Oct 14 at 6pm**

Reminder: Please put away devices at the start of class

Midterm Results

- Exams were graded on Wednesday evening by Instructional Team.
- Entered and analyzed by me over the weekend.
- Initially…
  - Median Score: 58%
  - Max Score: 88%

	Total
Mean (Raw)	60.82
Median (raw)	60
Min (Raw)	17
Max (Raw)	92
Mean (%)	58%
Median (%)	58%
Min (%)	16%
Max (%)	88%
Max Possible	104

Note: there was a typo on ^
Page 14; only 1 point (not 7) was possible on this page

Midterm Results

- Were the questions fair?

		Multiple Choice				Short Answer			MP1			MP2					
		Page 1	Page 2	Page 3	Page 4	Page 5	Page 6	Page 7	Page 8	Page 9	Page 10	Page 11	Page 12	Page 13	Page 14	Page 15	Page 16
Overal Performance	Q1	90%	69%	60%	73%	92%	98%	77%	77%	79%	84%	83%	68%	40%	60%	92%	70%
	Q2	86%	61%	50%	63%	85%	70%	68%	60%	71%	69%	69%	61%	29%	20%	93%	57%
	Q3	76%	56%	43%	61%	78%	60%	67%	48%	59%	54%	50%	63%	20%	27%	78%	30%
	Q4	76%	62%	30%	58%	76%	56%	63%	27%	43%	43%	39%	46%	13%	30%	62%	27%
	Q5	62%	48%	31%	54%	61%	42%	54%	14%	26%	5%	26%	45%	11%	11%	45%	7%

Midterm Results

- Were the questions fair?
  - For the most part, it looks like it — best overall performers did well on a page-by-page basis with steady decrease in points awarded for lesser performers

		Multiple Choice				Short Answer			MP1			MP2					
		Page 1	Page 2	Page 3	Page 4	Page 5	Page 6	Page 7	Page 8	Page 9	Page 10	Page 11	Page 12	Page 13	Page 14	Page 15	Page 16
Overal Performance	Q1	90%	69%	60%	73%	92%	98%	77%	77%	79%	84%	83%	68%	40%	60%	92%	70%
	Q2	86%	61%	50%	63%	85%	70%	68%	60%	71%	69%	69%	61%	29%	20%	93%	57%
	Q3	76%	56%	43%	61%	78%	60%	67%	48%	59%	54%	50%	63%	20%	27%	78%	30%
	Q4	76%	62%	30%	58%	76%	56%	63%	27%	43%	43%	39%	46%	13%	30%	62%	27%
	Q5	62%	48%	31%	54%	61%	42%	54%	14%	26%	5%	26%	45%	11%	11%	45%	7%

Midterm Results

- Were the questions fair?
  - For the most part, it looks like it — best overall performers did well on a page-by-page basis with steady decrease in points awarded for lesser performers

		Multiple Choice				Short Answer			MP1			MP2					
		Page 1	Page 2	Page 3	Page 4	Page 5	Page 6	Page 7	Page 8	Page 9	Page 10	Page 11	Page 12	Page 13	Page 14	Page 15	Page 16
Overal Performance	Q1	90%	69%	60%	73%	92%	98%	77%	77%	79%	84%	83%	68%	40%	60%	92%	70%
	Q2	86%	61%	50%	63%	85%	70%	68%	60%	71%	69%	69%	61%	29%	20%	93%	57%
	Q3	76%	56%	43%	61%	78%	60%	67%	48%	59%	54%	50%	63%	20%	27%	78%	30%
	Q4	76%	62%	30%	58%	76%	56%	63%	27%	43%	43%	39%	46%	13%	30%	62%	27%
	Q5	62%	48%	31%	54%	61%	42%	54%	14%	26%	5%	26%	45%	11%	11%	45%	7%

Top 30 performers averaged 60% credit on page 3….

Top 30 performers averaged 40% credit on page 14….

This looks worst than it is; Page 14 is only 1 point.

Midterm Results

- Since Page 3 and Page 13 were poor predictors of overall performance, I decided to drop those questions from the point total of the exam

		Multiple Choice				Short Answer			MP1			MP2					
		Page 1	Page 2	Page 3	Page 4	Page 5	Page 6	Page 7	Page 8	Page 9	Page 10	Page 11	Page 12	Page 13	Page 14	Page 15	Page 16
Overal Performance	Q1	90%	69%	60%	73%	92%	98%	77%	77%	79%	84%	83%	68%	40%	60%	92%	70%
	Q2	86%	61%	50%	63%	85%	70%	68%	60%	71%	69%	69%	61%	29%	20%	93%	57%
	Q3	76%	56%	43%	61%	78%	60%	67%	48%	59%	54%	50%	63%	20%	27%	78%	30%
	Q4	76%	62%	30%	58%	76%	56%	63%	27%	43%	43%	39%	46%	13%	30%	62%	27%
	Q5	62%	48%	31%	54%	61%	42%	54%	14%	26%	5%	26%	45%	11%	11%	45%	7%

Dropping this page,
i.e., 8 point curve

Also dropping this page,
i.e., another 6 point curve

Midterm Results

- After curve…
  - Median Score: 67%
  - Max Score: 102%
- **PICK-UP YOUR EXAMS IN DISCUSSION CLASS ON WEDNESDAY.**

	Total
Mean (Raw)	60.82
Median (raw)	60
Min (Raw)	17
Max (Raw)	92
Mean (%)	68%
Median (%)	67%
Min (%)	19%
Max (%)	102%
Max Possible	90

Midterm Results

- **<u>VERIFY OUR WORK</u>**
  - Our team graded 2,400 pages of exams in 4 hours… mistakes were caught, perhaps some weren't.
  - Check that point totals on cover page match bottom corner of each question page
  - We will not be releasing answer key; go to TA office hours if you have a question.
  - Email regrade requests to me (batesa@illinois.edu) by **<u>Wednesday, July 23rd.</u>**

	Total
Mean (Raw)	60.82
Median (raw)	60
Min (Raw)	17
Max (Raw)	92
Mean (%)	68%
Median (%)	67%
Min (%)	19%
Max (%)	102%
Max Possible	90

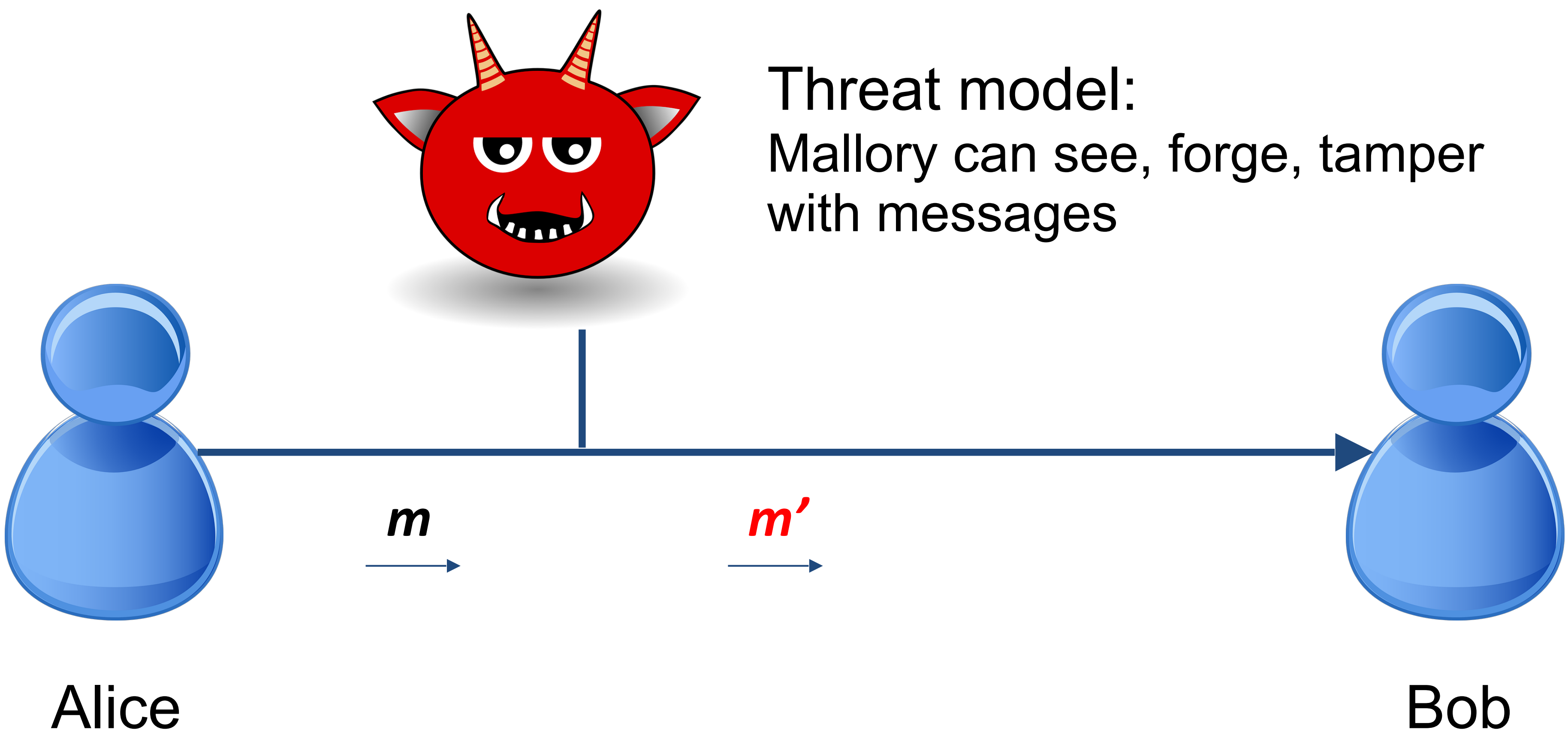
Goal: Secure File Transfer

Alice wants to send file ***m*** to Bob (let's say, a 4 Gigabyte movie)

Mallory wants to trick Bob into accepting a file Alice didn't send

Threat model:

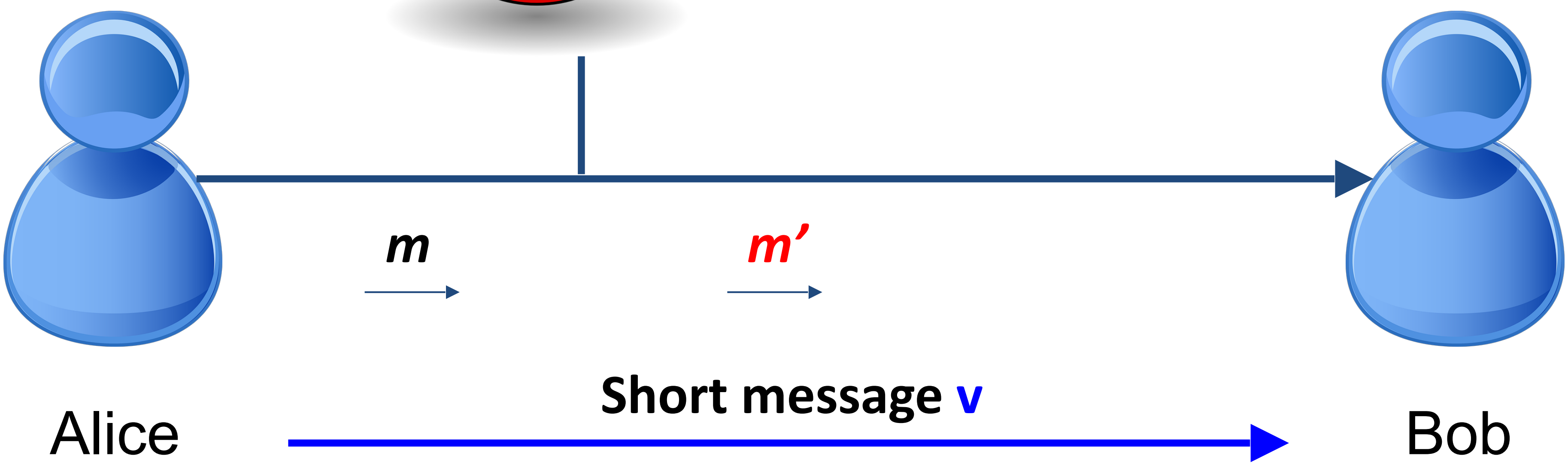
Mallory can see, forge, tamper with messages



Goal: Secure File Transfer

Alice wants to send file ***m*** to Bob (let’s say, a 4 Gigabyte movie)
Mallory wants to trick Bob into accepting a file Alice didn’t send

Threat model:
Mallory can see, forge, tamper with messages



Setup assumption: *Securely transfer a short message!*

Collision-Resistant Hash Function

Hash Function $\boldsymbol{h}$: $\{0,1\}^* \rightarrow \{0,1\}^{256}$ (or other fixed number)

1. Alice computes **v** := ***h***(**m**)
2. Alice transfers **v** over secure channel, **m** over insecure channel

3. Bob verifies that **v** = ***h***(**m'**), accepts file iff this is true

v

Function ***h*** ? We're sunk if Mallory can compute **m' ≠ m** where ***h***(**m**) = ***h***(**m'**)! A *collision*!

Contrast with: "checksums" e.g. CRC32.... defend against random errors, not a deliberate attacker!

Collision-Resistant Hash Function

Good hash functions should make it difficult to find …

First pre-image:

given h(m), find m

Which of these properties implies which others?

Second pre-image:

given m_1, find m_2 s.t. $h(m_1) = h(m_2)$

Collision:

find *any* m_1 != m_2 s.t. $h(m_1) = h(m_2)$

SHA256

`$ sha256sum file.dat` The **SHA256** compression function, **h**

Cryptographic hash

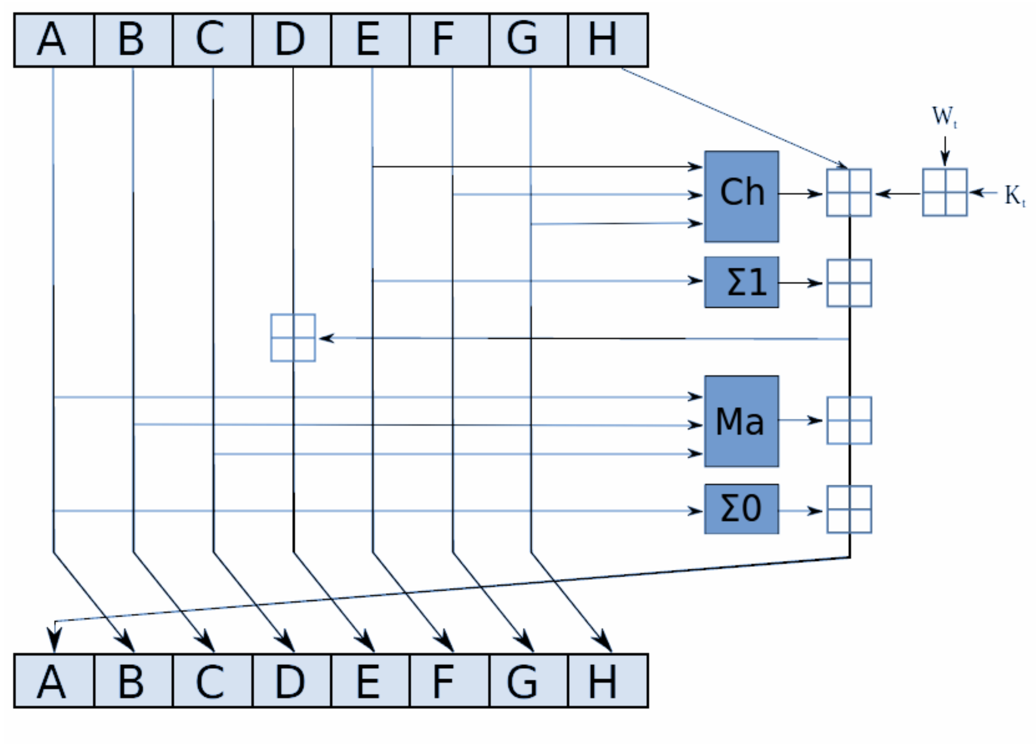
Input: arbitrary length data
(<u>No key</u>)
Output: 256 bits

Built with compression function, ***h***

(256 bits, 512 bits) in → 256 bits out

Designed to be really hairy (64 rounds of this)!

Provides <u>Confusion</u> and <u>Diffusion</u>



One iteration in a SHA-2 family compression function. The blue components perform the following operations:

$$\mathrm{Ch}(E,F,G) = (E \land F) \oplus (\neg E \land G)$$
$$\mathrm{Ma}(A,B,C) = (A \land B) \oplus (A \land C) \oplus (B \land C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

How do you find a collision?

If Mallory finds an m' such that H(m') = v, they can swap m without m' without being detected… how to find?

- **Pigeonhole principle:** collisions must exist
  Input space $\{0,1\}^*$ larger than output $\{0,1\}^{256}$
- **Birthday attack:** build a table with 2^{128} entries
  With ~50% probability, have a collision
- **Cycle finding:** "Tortoise and hare" algorithm
  $h(x)$, $h(h(x))$, $h(h(h(x)$, .., $h^i(x)$

These are **generic**; actual attacks rely on **structure** of the particular function

Birthday Problem

n	***Probability of two people sharing birthday***
1	0.0%
5	2.7%
10	11.7%
20	41.1%
23	50.7%
30	70.6%
40	89.1%
50	97.0%
60	99.4%
70	99.9%
75	99.97%
100	99.99997%

Likelihood of Collision?

Most cryptographic primitives come with a **security parameter**… not the key, but an estimate of the difficulty of successful attack.

- Notation: Usually k, or λ
- Often *corresponds* to a key size

Cryptography protocols run in **polynomial** time

i.e., as a function of λ, O(poly(λ))

In a good cryptosystem, we should be able to show that the chance of failure is **negligible**, or **vanishingly** small, as a function of λ, i.e., O(negl(λ))

Security Parameterization

How large of a hash digest size should we choose?

1. Estimate an attacker's budget

E.g., the entire NSA

2. Consider the best known attacks

Reduction from protocol to well-studied problem

3. Add a safety margin

If all goes well, adding 1 bit increases search space by 2x

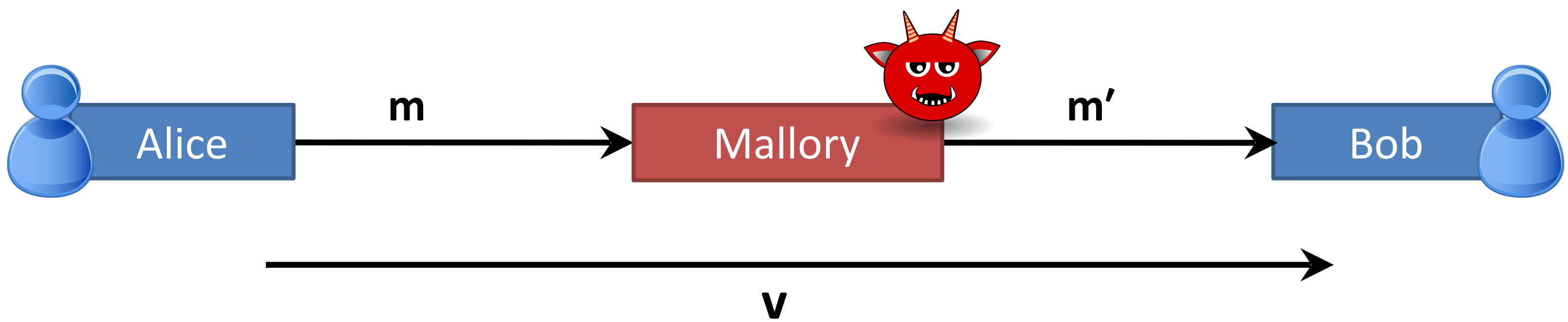
Security Parameterization

SHA512 provides a 512 bit output size. The security parameter k=512 implies…

- k/2 bit collision resistance
- k bit preimage resistance
- k-N bit second presage resistance
  - N = log(target message length)

What's missing?

- In practice, we don't use a magically secure channel to send v over; it will have to traverse the same channel as m



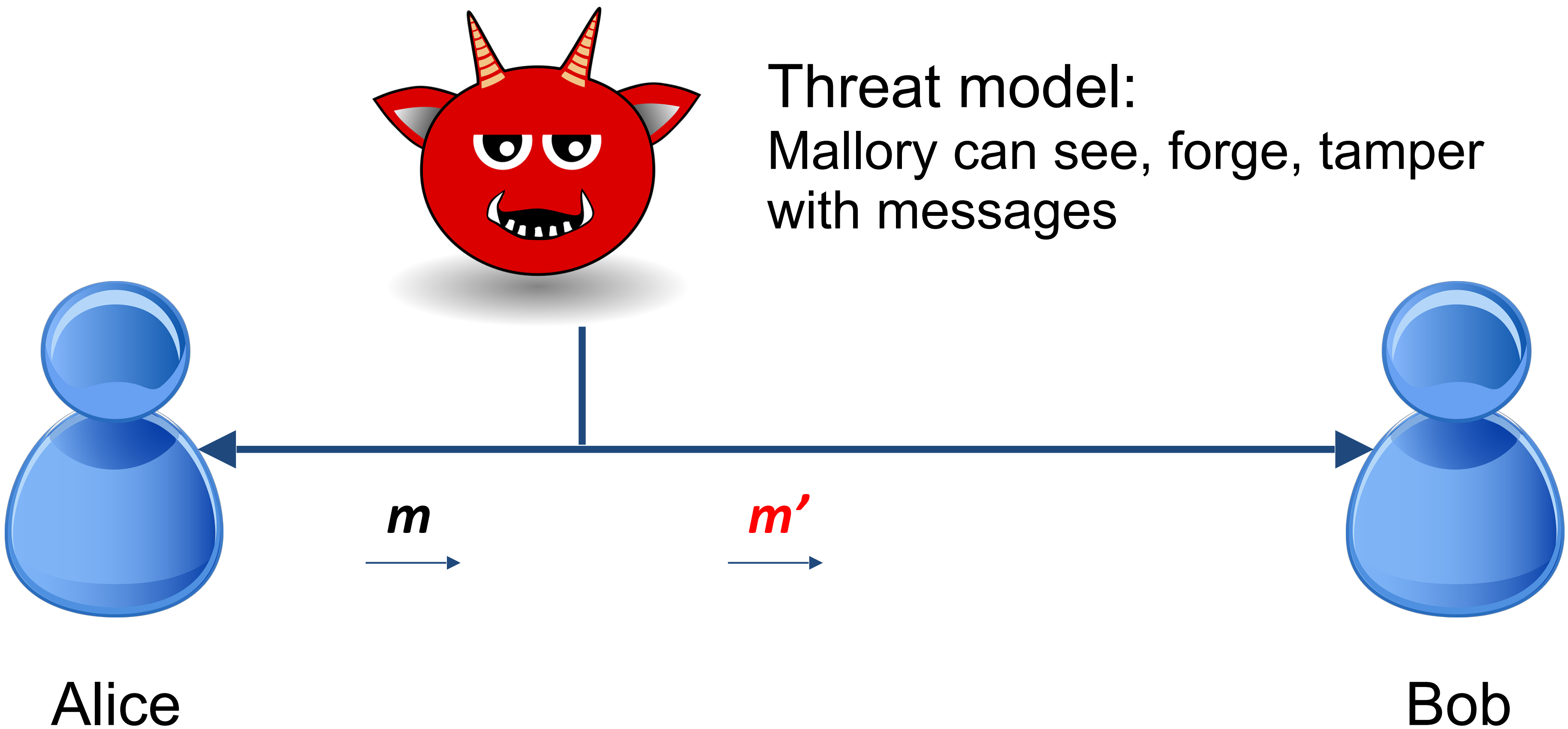
- If Mallory intercepts m, v; they can calculate a new v' to pass along with the tampered message m'

Message Integrity

Alice wants to send message ***m*** to Bob

Mallory wants to trick Bob into accepting a message Alice didn't send

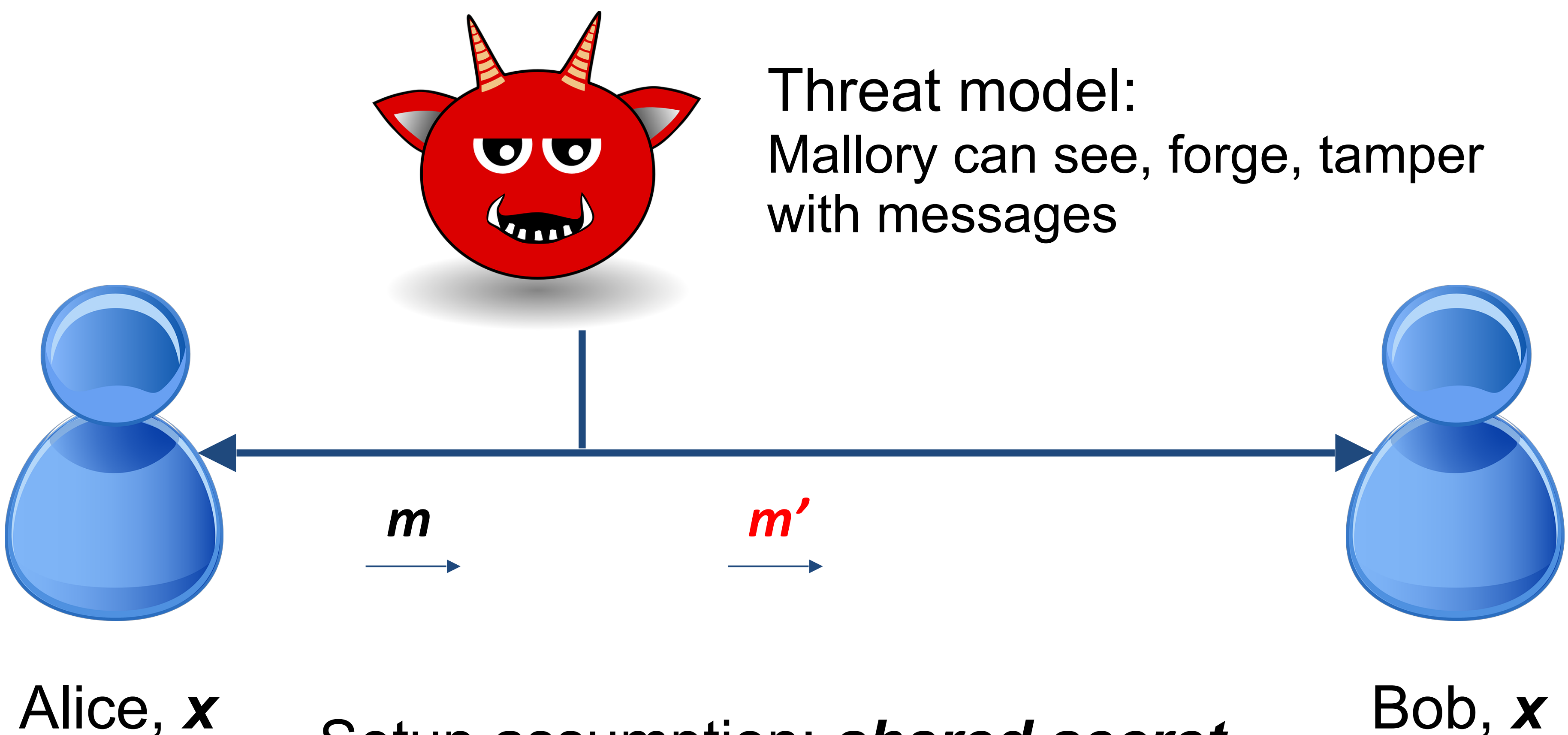
Threat model:
Mallory can see, forge, tamper with messages



Message Integrity

Alice wants to send message ***m*** to Bob

Mallory wants to trick Bob into accepting a message Alice didn't send



Setup assumption: ***shared secret***

Hash-Based MAC (HMAC)

HMAC-SHA256 see RFC 2104

HMAC$_k$(**m**) =

$$\text{SHA256}\left(k \oplus c_1 \;\|\; \text{SHA256}\left(k \oplus c_2 \;\|\; m\right)\right)$$

XOR

`0x3636…`

Concatenation

`0x5c5c…`

What are those two constant values for?

Why not just use SHA256?

Is this a PRF?

$$f_k(\mathbf{m}) = \text{SHA256}(\ \mathbf{k}\ ||\ \mathbf{m1}\ )$$

- On its own, SHA256 is *not a strong PRF*
- ***Length extension attacks***: If an attacker knows the value of m1 and the length of k, they can produce a valid hash for some message m2 that is an extension of m1.
  - Problem for SHA2, addressed in SHA3 family.
- Can be used, e.g., to overwrite values for protocols that accept the last specified value for a field.

```
Original Data: count=10&lat=37.351&user_id=1&long=-119.827&waffle=eggo
Desired New Data:
count=10&lat=37.351&user_id=1&long=-119.827&waffle=eggo&waffle=liege
```

MAC Crypto Game

Game against Mallory

1. Give Mallory MAC(k, m_i) for all m_i in M

In other words, Mallory has an ***oracle***

Mallory can choose next m_i after seeing answer

2. Mallory tries to discover MAC(k, m') for a new m' not in M

We can show the **MAC game** ***reduces*** to the **PRF game**.
Mallory wins MAC game → she wins PRF game.

This is a **Security Proof**

What is a security proof?

- A ***reduction*** from an ***attack on your protocol*** to an attack on a ***widely studied, hard problem***
- Excludes large classes of attacks, guides **composition**
    - Proofs are in **models**. So, attack outside the model!
- It does **NOT** ***prove*** that your protocol is ***secure***
- Also, we don't know if there are any hard problems!
- The field of **Modern Cryptography** is based on proofs
- Many widely used primitives (SHA-256, AES, DSA) have no security proof. We rely on them because they're widely studied

… so where do keys come from?

- To set up a MAC, we need a random key k… how do we generate?
- **True Randomness**:
  - Output of a physical process that is inherently random
  - Scarce, and hard to get
- **Pseudorandom Function (PRF)**:
  - Sampled from a family of functions using a key
- **Pseudorandom Generator (PRG)**:
  - Takes small seed that is really random
  - Generates a stream (arbitrarily long sequence) of numbers that are "as good as random"

Pseudorandom Generator

Definition: **PRG** is secure if it's indistinguishable from a random stream of bits

Similar game to PRF definition:

1. We flip a coin secretly to get a bit **b**
2. If **b**=0, let ***s*** be a truly random stream
   If **b**=1, let ***s*** be $\boldsymbol{g}_{\mathbf{k}}$ for random secret **k**
3. Mallory can see as much of the output of ***s*** as he/she wants
1. Mallory guesses **b**,
   wins if guesses correctly

g is a secure PRG if no winning strategy for Mallory*

Pseudorandom Generator

Here's a *simple PRG that works:*

For some random k and PRF *f*,
output: $f_k(0) \;||\; f_k(1) \;||\; f_k(2) \;||\; \ldots$

Theorem: If ***f*** is a secure PRF, and ***g*** is built from ***f*** by this construction, then ***g*** is a secure PRG.

Proof: Assume ***f*** is a secure PRF, we need to show that ***g*** is a secure PRG.

Proof by contradiction:

1. Assume ***g*** is *not* secure; so Mallory can win the PRG game
2. This gives Mallory a winning strategy for the PRF game:
   a. query the PRF with inputs 0, 1, 2, …
   b. apply the PRG-distinguishing algorithm
3. Therefore, Mallory can win PRF game; this is a contradiction
4. Therefore, g is secure

How do we get true randomness?

Want "indistinguishable from random"
which means: adversary can't guess it

Gather lots of details about the computer that the adversary will have trouble guessing [Examples?]

- Problem: Adversary can predict some of this
- Problem: How do you know when you have enough randomness?

Modern OSes typically collect randomness, give you API calls to get it

- e.g., Linux:

`/dev/random` a device that gives random bits, blocks until available

`/dev/urandom` gives output of a PRG, nonblocking, seeded from `/dev/random` *eventually*

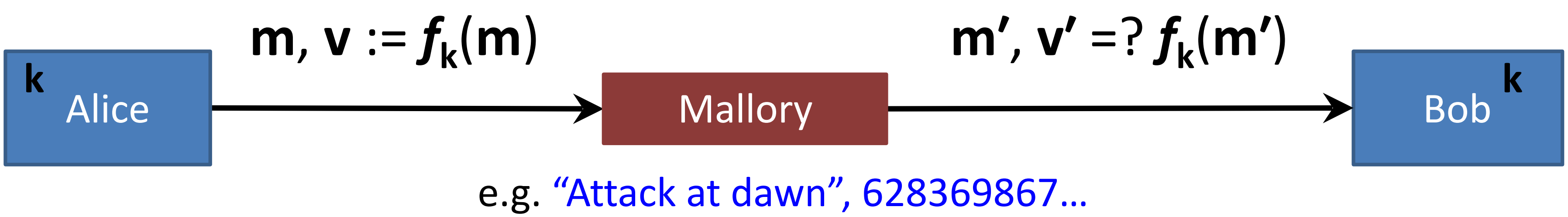
Review: Message Integrity

Integrity of message sent over an untrusted channel

Alice must append bits to message that only Alice (or Bob) can make

Idealized solution: Random function

Practical solution:



(Hash-based) MAC

f_k is (we hope!) indistinguishable in practice from a random function, unless you know **k**