Essential Cryptography I

EECS 588: Computer and Network Security
January 13, 2009

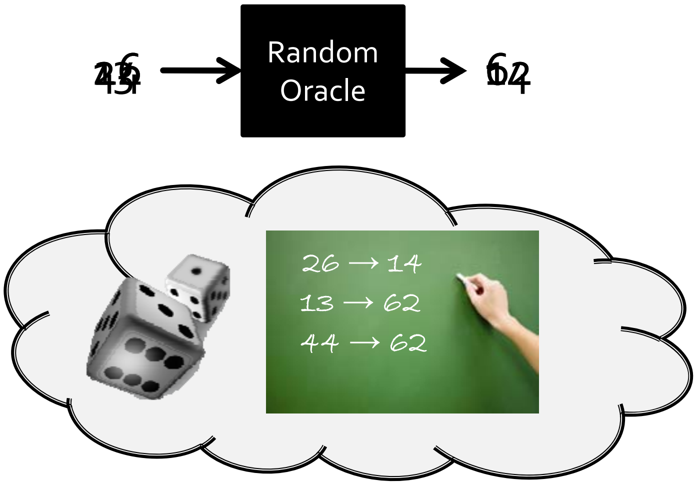
Today's Class

- The Cryptographer's View
- Hash Functions
- Message-Authentication Codes
- Block Ciphers

(BREAK)

- Generating Random Numbers
- Cipher Modes
- Padding
- Building a Secure Channel

The Cryptographer’s View



Practical Random Oracles?

Suppose domain is size 2^{256}…

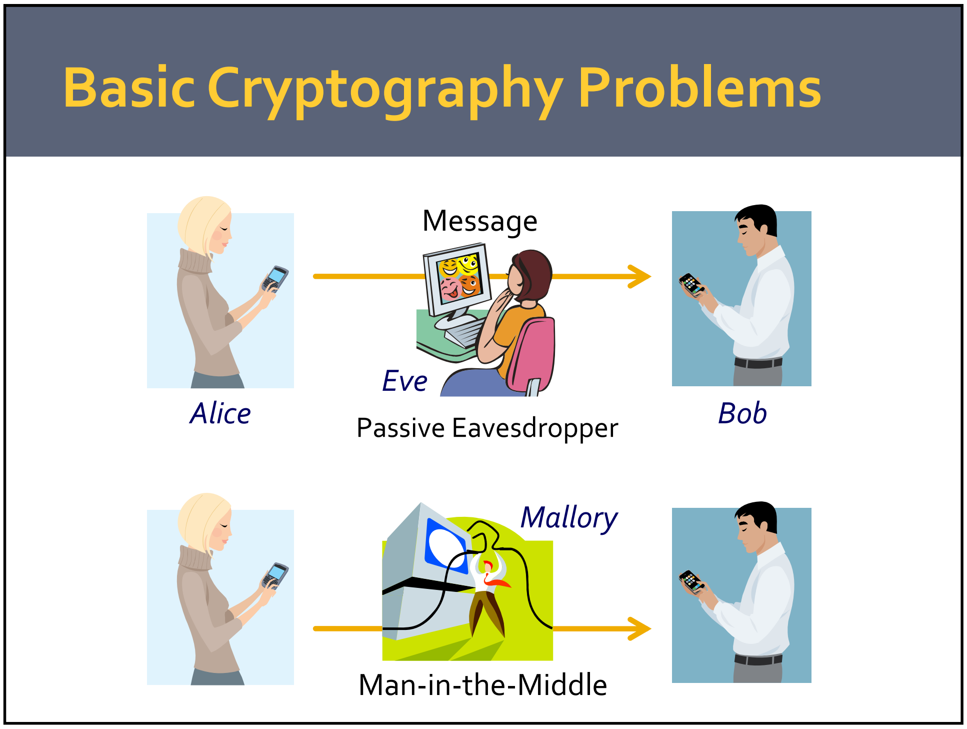
Pseudorandom Functions (PRFs)

(A function randomly chosen from a *family* of PRFs is computationally indistinguishable from a Random Oracle)

≈ Message Authentication Codes (MACs)

Pseudorandom Permutations

≈ Symmetric Ciphers



Basic Cryptography Problems

Message

Alice

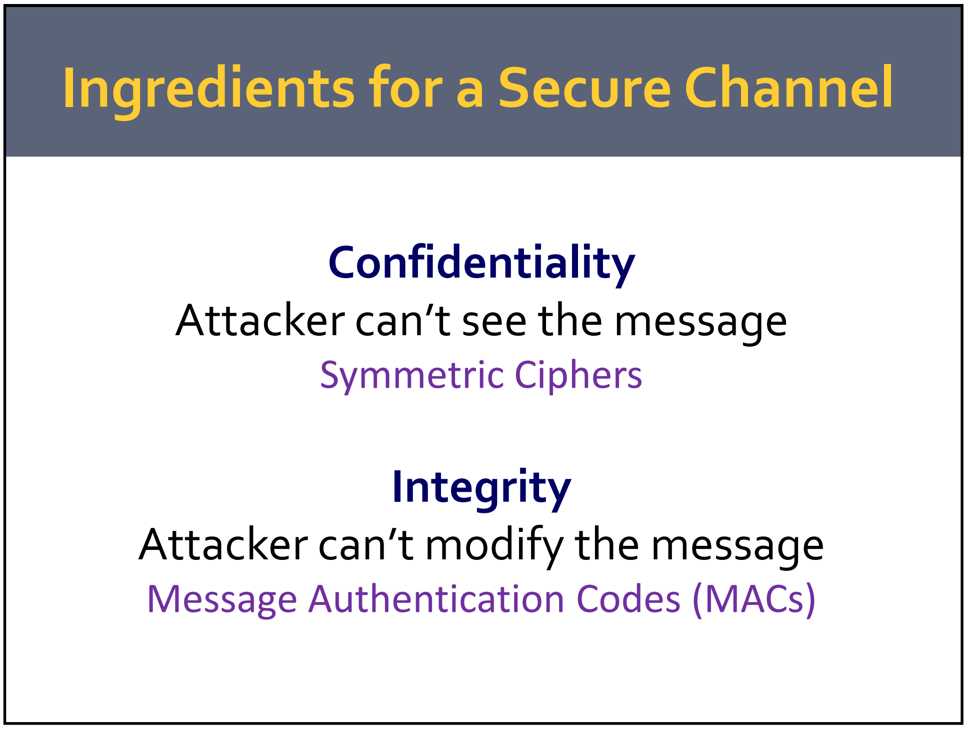
Eve

Passive Eavesdropper

Bob

Mallory

Man-in-the-Middle



Ingredients for a Secure Channel

Confidentiality

Attacker can't see the message

Symmetric Ciphers

Integrity

Attacker can't modify the message

Message Authentication Codes (MACs)

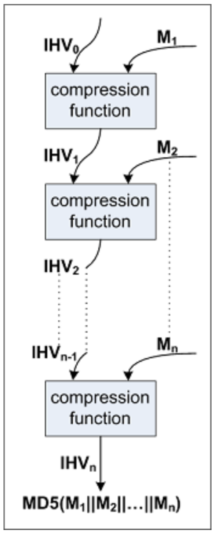
Hash Functions

- Ideal: Random mapping from *any input* to a *set of output*

- Requirements:
  - One-way
  - Collision-resistant
- Caution! Real hashes don't match our ideal

MD5 Hash Function

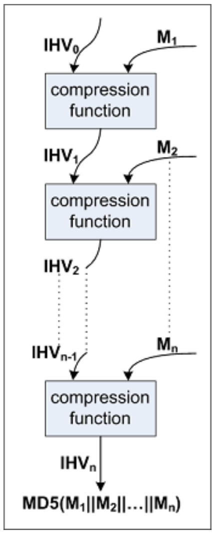
- Designed in 1992 by Ron Rivest
  - 128-bit output
  - 128-bit internal state
  - 128-bit block size
- Like most hash functions, uses block-chaining construction



SHA Hash Functions

- Very in software compared to MD5
- SHA-1 – standardized by NIST in 1995
  - 160-bit output and internal state
  - 512-bit block size
- SHA-256 – extension published in 2001
  - 256-bit output and internal state
  - 512-bit block size
- SHA-512 – extension published in 2001
  - 512-bit output and internal state
  - 1024-bit block size

Tricky! Length Extension Attacks



The *i*-th than internal state (IHV) is equivalent to the hash of the first *i* blocks.

Given hash of secret *x*, trivial to find hash of *x* || *m* for many values of *m* (slight issues of blocking and padding).

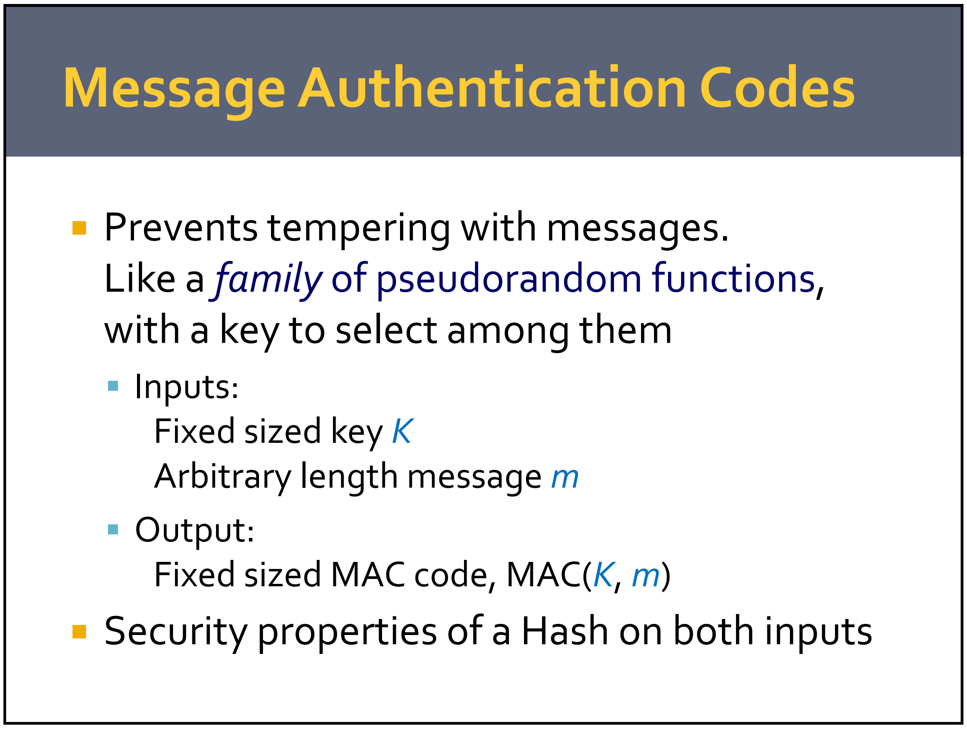
MD5 and SHA family all vulnerable!

MD5 is Unsafe – Never use it!

- First flaws in 1996; by 2007, researchers demonstrated a collision
- Chaining allows chosen prefix attack
- Dec. 2008: others used this to fake SSL certificates (cluster of 200 PS3s)

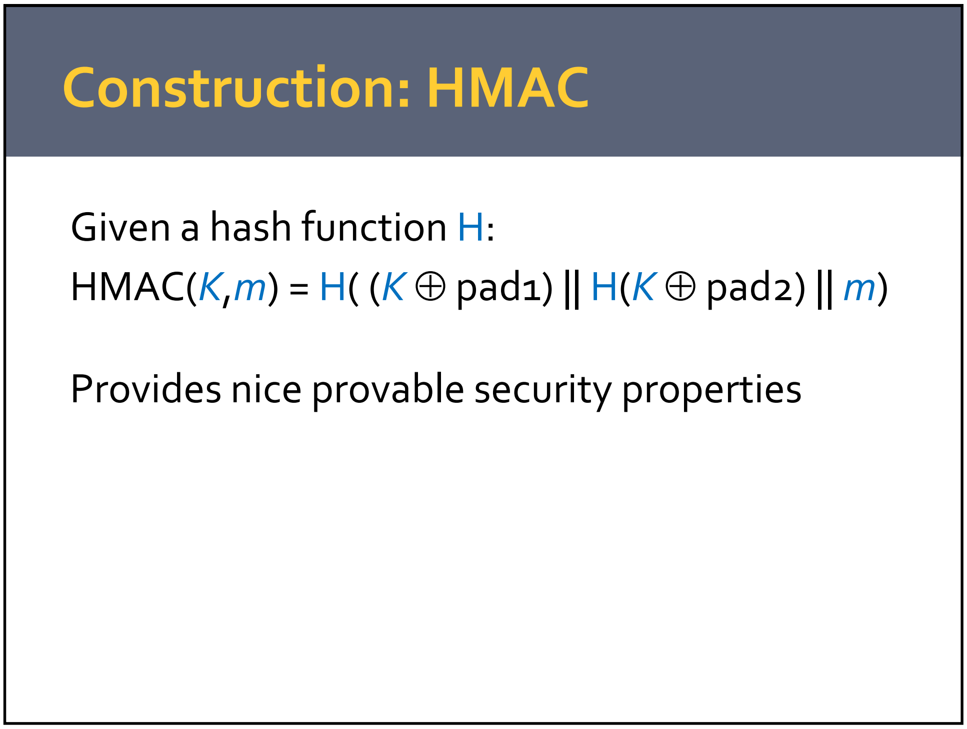
Is SHA-1 Safe?

- Significant cryptanalysis since 2005
- Improved attacks show complexity of finding a collision < 2^{63} (should be 2^{80} – why?)
- Attacks only bet better…
- Don't use SHA-1. Use SHA-256 until we have something better.



Message Authentication Codes

- Prevents tempering with messages.
  Like a *family* of pseudorandom functions,
  with a key to select among them
  - Inputs:
    Fixed sized key K
    Arbitrary length message m
  - Output:
    Fixed sized MAC code, MAC(K, m)
- Security properties of a Hash on both inputs



Construction: HMAC

Given a hash function H:

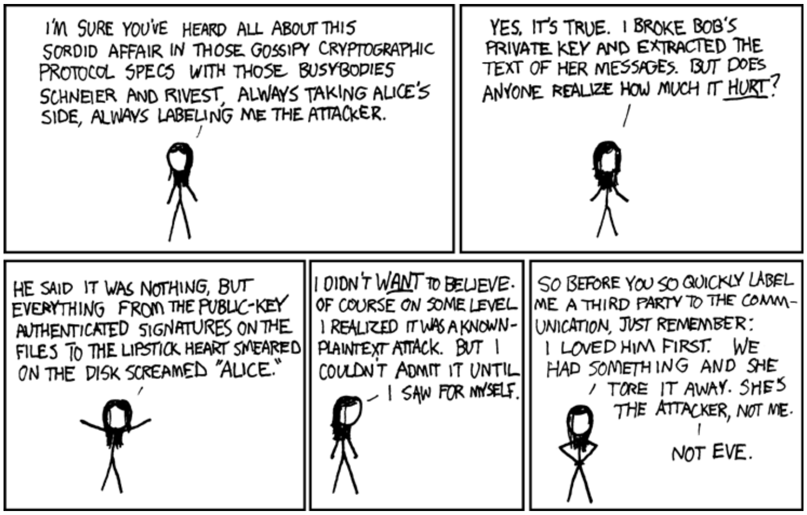
$\mathrm{HMAC}(K,m) = \mathrm{H}(\,(K \oplus \mathrm{pad1}) \,||\, \mathrm{H}(K \oplus \mathrm{pad2}) \,||\, m)$

Provides nice provable security properties

What Should You Use?

- What should you use when you need a hash function?
  - Conservative answer: Use HMAC-SHA256
  - Avoids length extension attacks

10 Minute Break


I'M SURE YOU'VE HEARD ALL ABOUT THIS SORDID AFFAIR IN THOSE GOSSIPY CRYPTOGRAPHIC PROTOCOL SPECS WITH THOSE BUSYBODIES SCHNEIER AND RIVEST, ALWAYS TAKING ALICE'S SIDE, ALWAYS LABELING ME THE ATTACKER.
YES, IT'S TRUE. I BROKE BOB'S PRIVATE KEY AND EXTRACTED THE TEXT OF HER MESSAGES. BUT DOES ANYONE REALIZE HOW MUCH IT HURT?
HE SAID IT WAS NOTHING, BUT EVERYTHING FROM THE PUBLIC-KEY AUTHENTICATED SIGNATURES ON THE FILES TO THE LIPSTICK HEART SMEARED ON THE DISK SCREAMED "ALICE."
I DIDN'T WANT TO BELIEVE. OF COURSE ON SOME LEVEL I REALIZED IT WAS A KNOWN-PLAINTEXT ATTACK. BUT I COULDN'T ADMIT IT UNTIL I SAW FOR MYSELF.
SO BEFORE YOU SO QUICKLY LABEL ME A THIRD PARTY TO THE COMMUNICATION, JUST REMEMBER: I LOVED HIM FIRST. WE HAD SOMETHING AND SHE TORE IT AWAY. SHE'S THE ATTACKER, NOT ME.
NOT EVE.

One-Time Pads

Provably secure encryption…

… that often fails in practice.

Block Ciphers

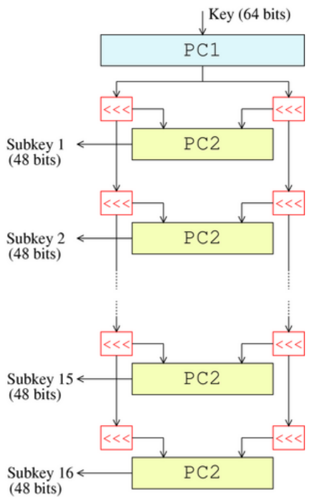
- Ideal block cipher:
  Like a *family* of pseudorandom *permutations* with a key to select among them
- Unlike hashes and MACs, ciphers are invertible – encryption and decryption functions

DES—Data Encryption Standard

- US Government standard (1976)
- Designed by IBM
  Tweaked by NSA

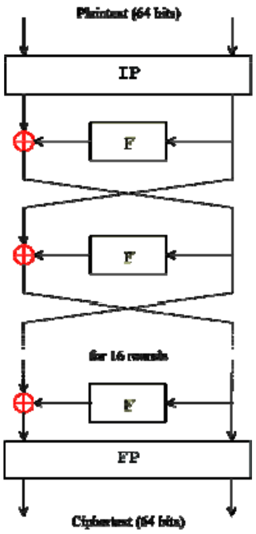
- 56-bit *key*
- 64-bit *blocks*
- 16 *rounds*

- Key schedule function generates 16 round keys:



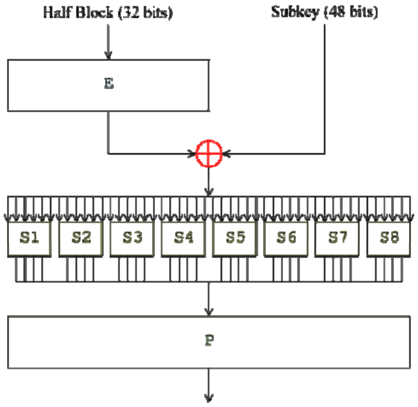
DES Encryption

- Feistel network
  - common block cipher construction
  - makes encryption and decryption symmetric—just reverse order of round keys
  - Each round uses the same Feistel function *F*
    (by itself a weak block cipher)



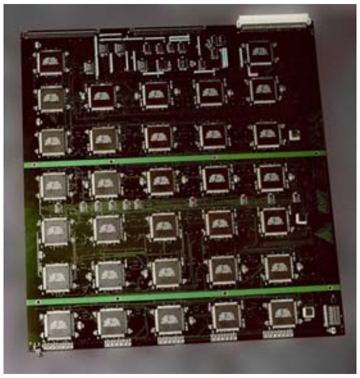
DES Feistel Function

- In each round:
  - Expansion Permutation *E* 32 → 48 bytes
  - S-boxes ("substitution") replace 6-bit values
  - Fixed Permutation P rearrange the 32 bits



DES is Unsafe – Don't Use It!

- Design has known weaknesses
- 56-bit key ***way*** too short
- EFF's "Deep Crack" machine can brute force in 56 hours using FPGAs ($250k in 1998, far cheaper today)



- 3-DES?

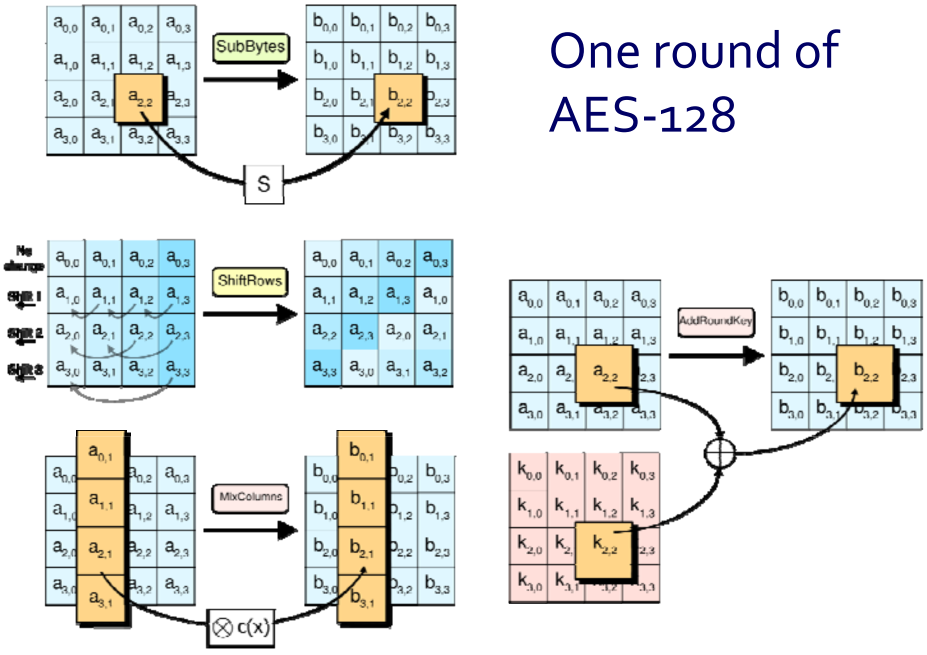
AES—Advanced Encryption Standard

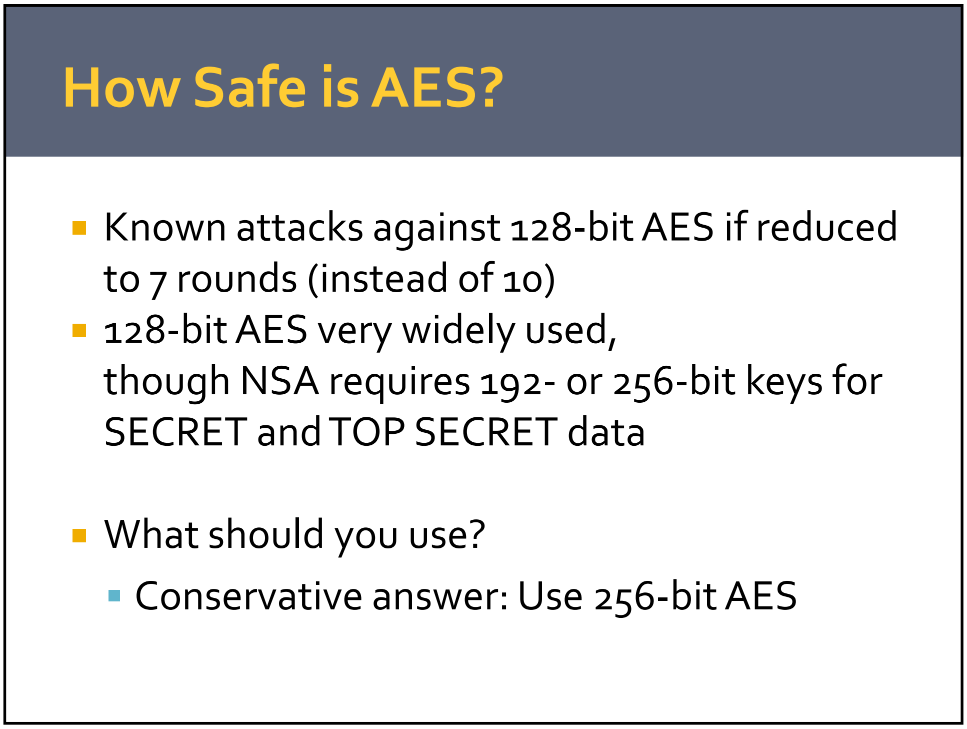
- Standardized by NIST in 2001 following open design competition (a.k.a. Rijndael)

- 128-, 192-, or 256-bit key
- 128-bit blocks
- 10, 12, or 14 rounds

- Not a Feistel-network construction

One round of AES-128

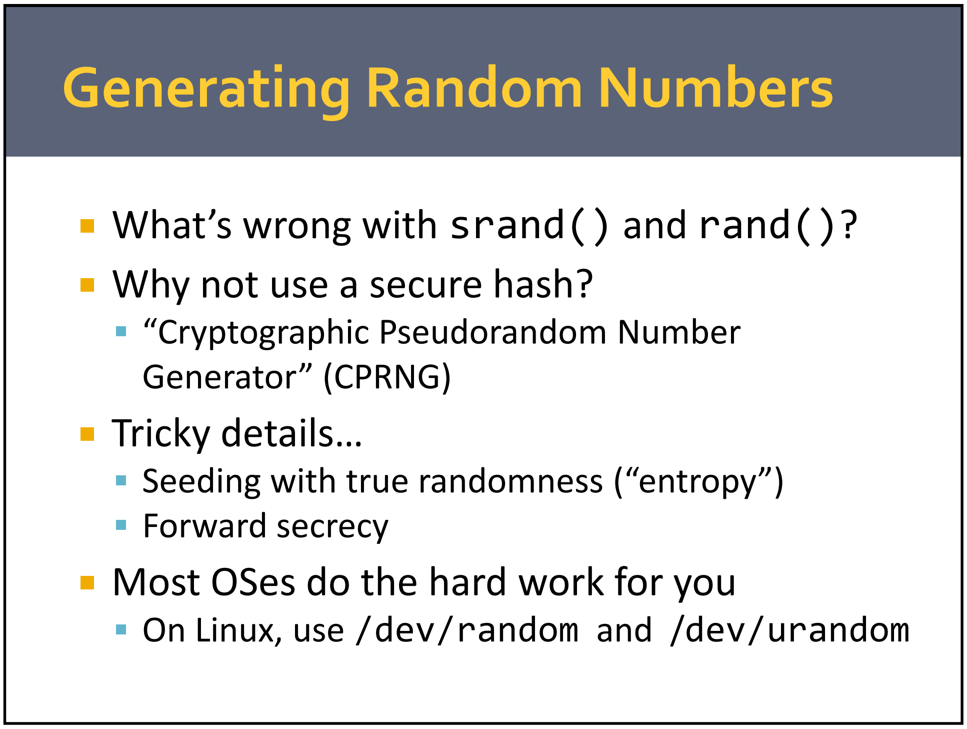




How Safe is AES?

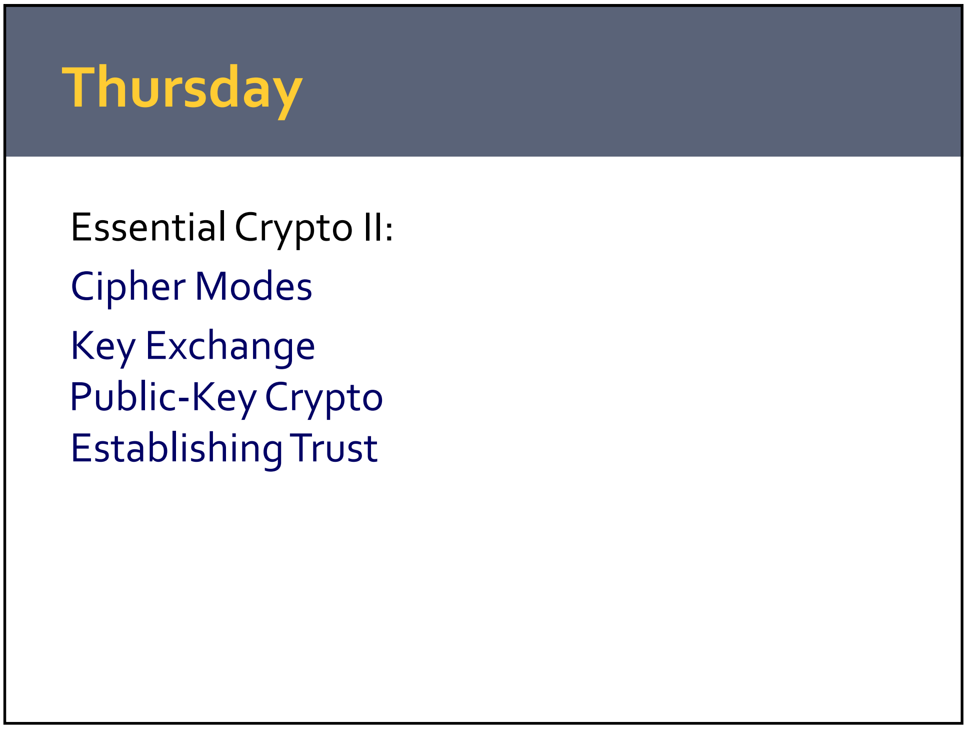
- Known attacks against 128-bit AES if reduced to 7 rounds (instead of 10)
- 128-bit AES very widely used,
  though NSA requires 192- or 256-bit keys for SECRET and TOP SECRET data

- What should you use?
  - Conservative answer: Use 256-bit AES



Generating Random Numbers

- What’s wrong with `srand()` and `rand()`?
- Why not use a secure hash?
  - “Cryptographic Pseudorandom Number Generator” (CPRNG)
- Tricky details…
  - Seeding with true randomness (“entropy”)
  - Forward secrecy
- Most OSes do the hard work for you
  - On Linux, use `/dev/random` and `/dev/urandom`



Thursday

Essential Crypto II:

Cipher Modes

Key Exchange

Public-Key Crypto

Establishing Trust