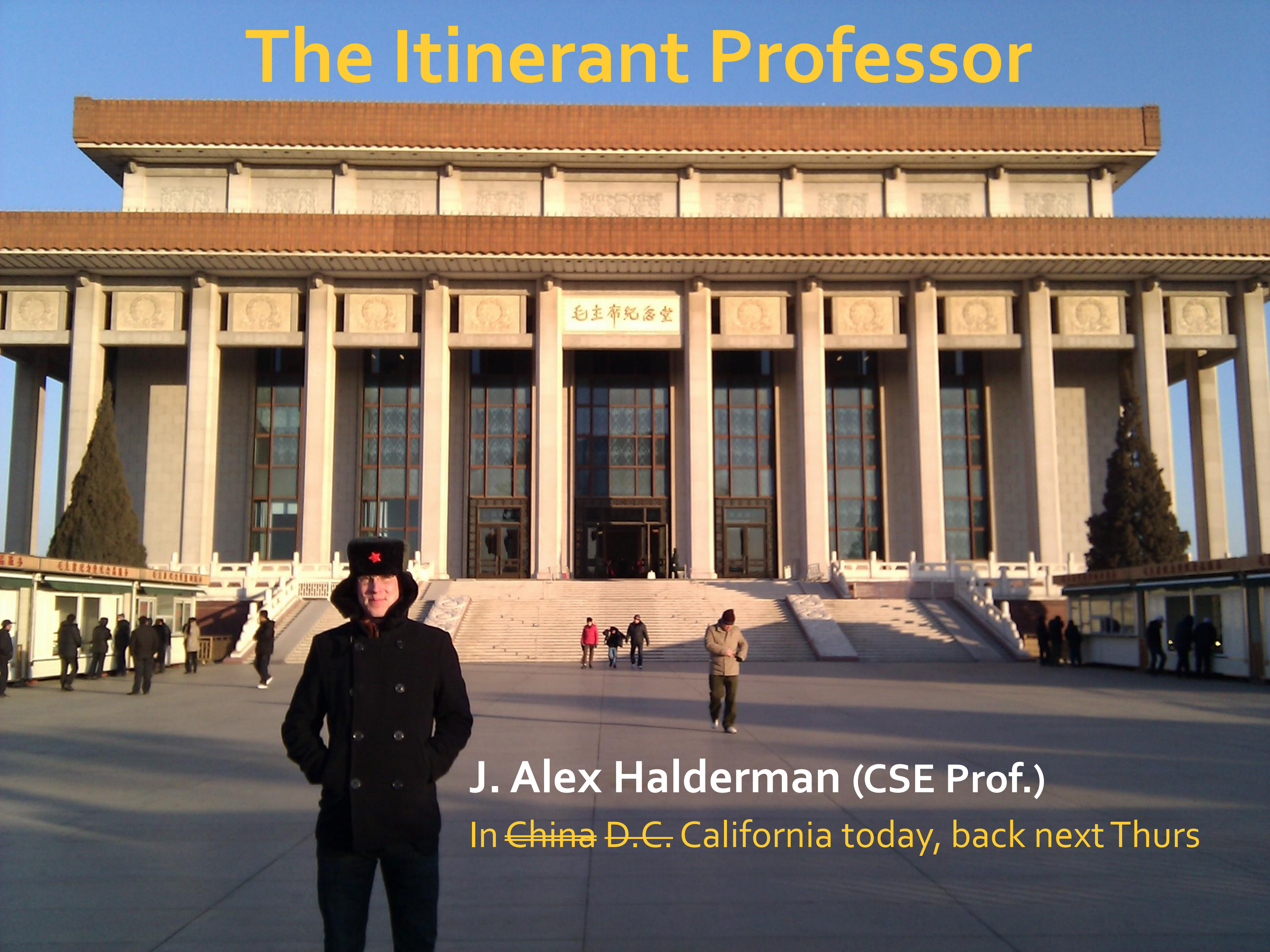
Essential Cryptography I

EECS 588: Computer and Network Security
January 10, 2013

The Itinerant Professor



J. Alex Halderman (CSE Prof.)

In ~~China~~ ~~D.C.~~ California today, back next Thurs

Goals for this Course

- Gain hands-on experience
  - Building secure systems
  - Evaluating system security
- Prepare for research
  - Computer security subfield
  - Security-related issues in other areas
- Generally, improve research and communication skills
- Learn to be a `1337 hax0r`, but an ethical one!

Building Blocks
The security mindset, thinking like an attacker, reasoning about risk, research ethics
Symmetric ciphers, hash functions, message authentication codes, pseudorandom generators
Key exchange, public-key cryptography, key management, the SSL protocol

Software Security
Exploitable bugs: buffer overflows and other common vulnerabilities – attacks and defenses
Malware: viruses, spyware, rootkits – operation and detection
Automated security testing and tools for writing secure code
Virtualization, sandboxing, and OS-level defenses

Web Security
The browser security model
Web site attacks and defenses: cross-site scripting, SQL injection, cross-site reference forgery
Internet crime: spam, phishing, botnets – technical and nontechnical responses

Network Security
Network protocols security: TCP and DNS – attacks and defenses
Policing packets: Firewalls, VPNs, intrusion detection
Denial of service attacks and defenses
Data privacy, anonymity, censorship, surveillance

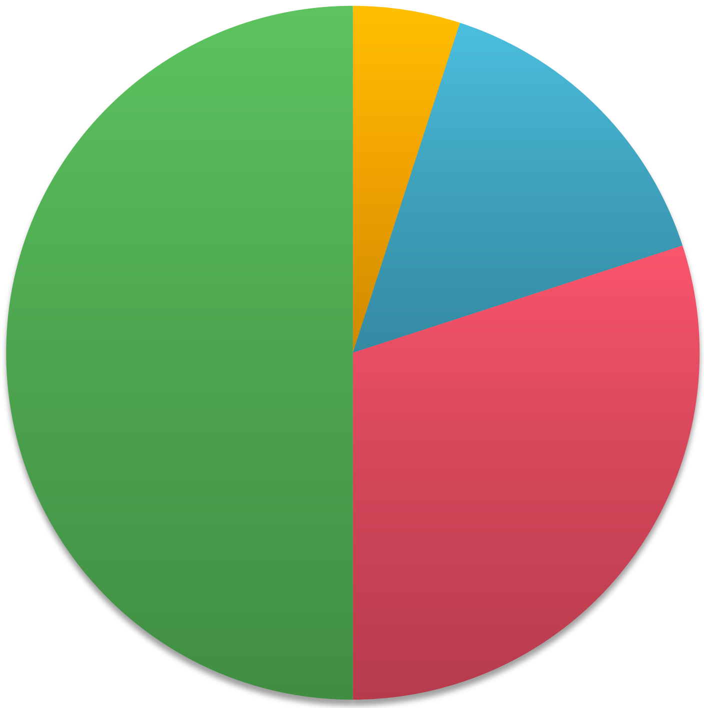
Advanced Topics
Hardware security – attacks and defenses
Trusted computing and digital rights management
Electronic voting – vulnerabilities, cryptographic voting protocols

Not a crypto course

Getting a Seat

- You probably will
- Alex intends to teach 588 again next winter

Grading



No exams, no problem sets!

Class Participation (5%)

- 1-2 required papers for discussion in each sessions (other readings optional)

- Come prepared to contribute!
- Full points for speaking up and contributing substantial ideas
- Lose points for being silent, frequently missing class, browsing the web, etc.

Paper Responses (15%)

Brief written response to each paper (~400 words)

- In the first paragraph:
  - State the problem that the paper tries to solve; and
  - Summarize the main contributions.
- In one or more additional paragraphs:
  - Evaluate the paper's strengths and weaknesses;
  - Discuss something you would have done differently if you wrote the paper; and
  - Suggest at least two interesting open problems on related topics.
- List any areas you had trouble understanding.
  We'll try to explain them in class.

Attack Presentation (30%)

- *With a partner*, choose a specific attack from recent research and implement a demonstration
- Give a 15 minute presentation:

  (1) describe the attack

  (2) talk about how you implemented it, give a demo

  (3) discuss possible defenses
- Course schedule lists topics and dates
- **Each group email top 4 choices by Friday 1/18**

Research Project (50%)

In groups, investigate a new attack or defense

Should have potential to become a marketable product or conference paper

(*but not necessarily by the end of the term*)

Components: (see website for details)

- Project proposal (5%)
- Project checkpoint (5%)
- Conference-style presentation in class (15%)
- Final conference-style report (25%)

Communication

Course Web Site

https://www.eecs.umich.edu/courses/eecs588/

announcements, schedule, readings

Email Us

jhalderm@umich.edu

zakir@umich.edu

suggestions, questions, concerns

Law and Ethics

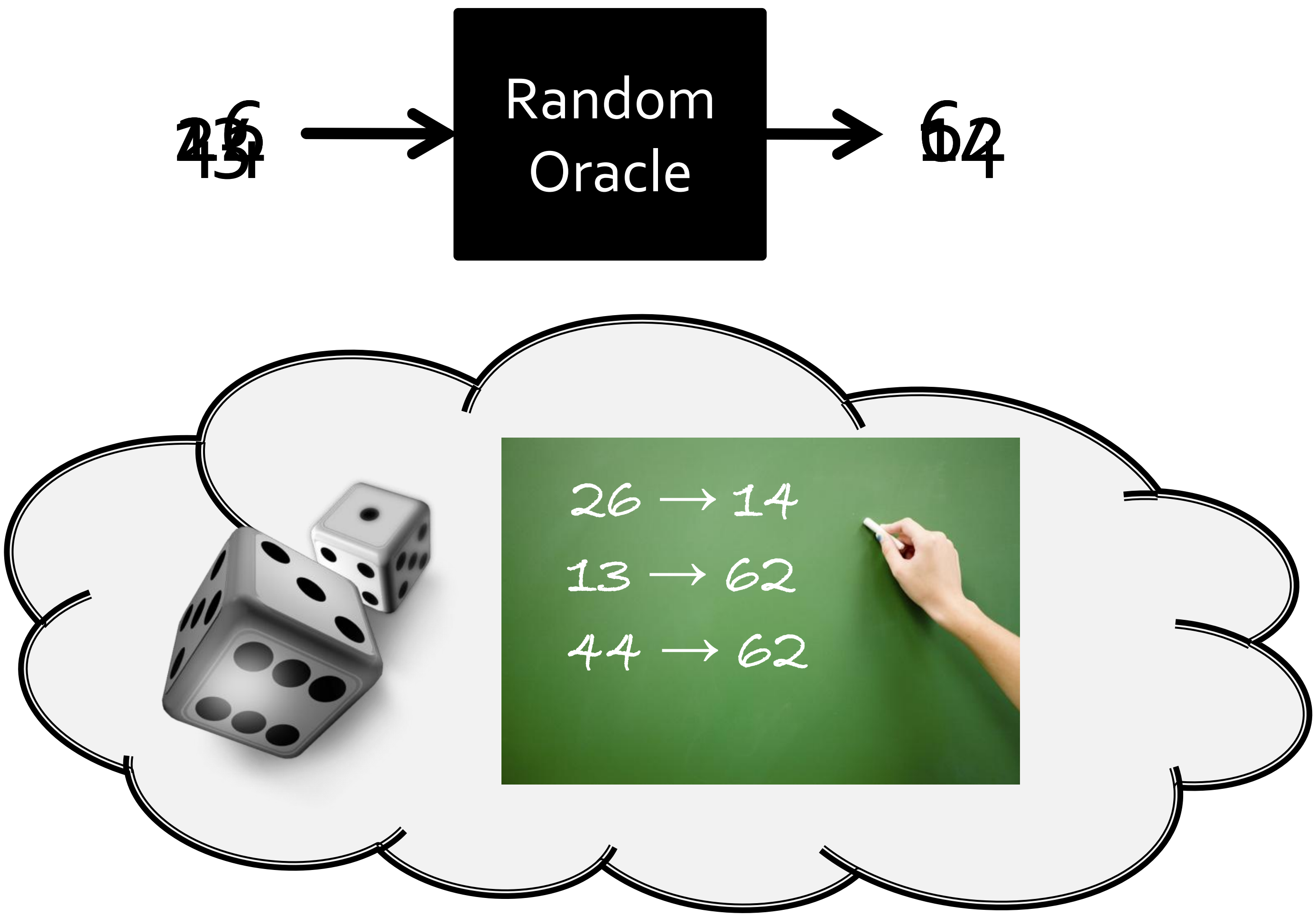
- **Don't be evil!**
  - Ethics requires you to refrain from doing harm
  - Always respect privacy and property rights
  - Otherwise <u>you will fail the course</u>
- Federal and state laws criminalize computer intrusion and wiretapping
  - e.g. Computer Fraud and Abuse Act (CFAA)
  - You can be sued or go to jail
- University policies prohibit tampering with campus systems
  - You can be disciplined, even expelled

Today’s Class

Essential Cryptography, Part 1

- The Cryptographer’s View
- Hash Functions
- Message-Authentication Codes
- Generating Random Numbers
- Block Ciphers

The Cryptographer's View



Practical Random Oracles?

Suppose domain is size 2^{256}…

Pseudorandom Functions (PRFs)

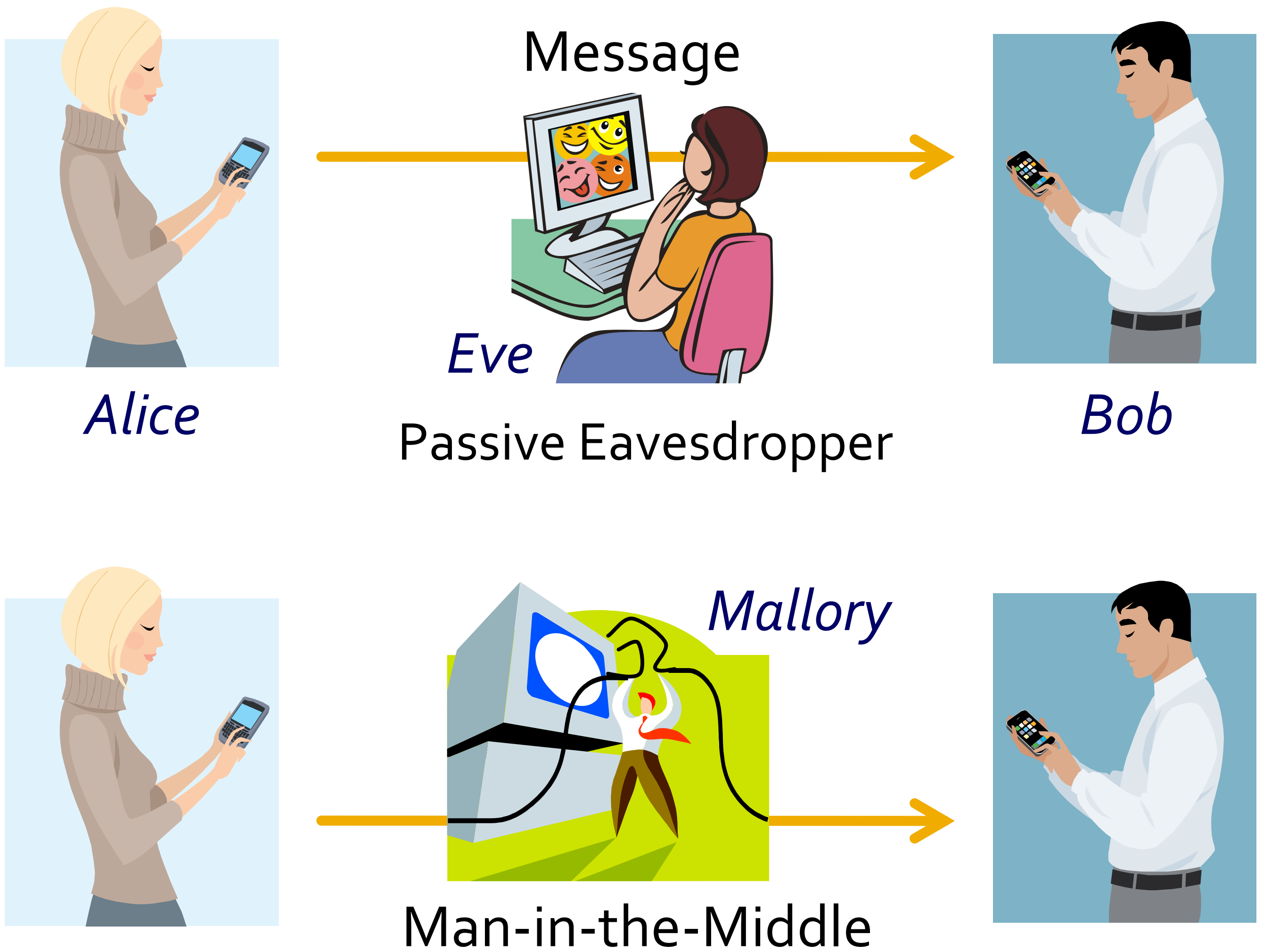
(A function randomly chosen from a *family* of PRFs is computationally indistinguishable from a Random Oracle)

≈ Message Authentication Codes (MACs)

Pseudorandom Permutations

≈ Symmetric Ciphers

Basic Cryptography Problems



Ingredients for a Secure Channel

Confidentiality

Attacker can't see the message

Symmetric Ciphers

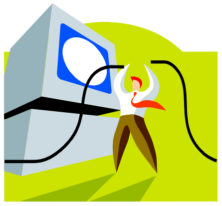
Eve

Integrity

Attacker can't modify the message

Message Authentication Codes (MACs)

Mallory



Hash Functions

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

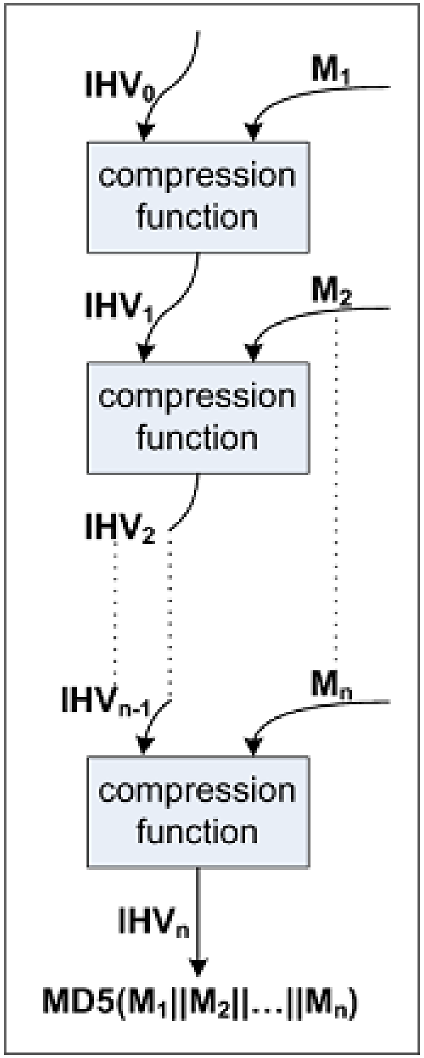
- Caution! Real hashes don't match our ideal

Hash Function Requirements

- First pre-image
  - Given h(x), find x
- Second pre-image
  - Given m_1, find m_2 s.t. $h(m_1) = h(m_2)$
- Collision
  - Given nothing, find *any* $m_1 \neq m_2$ s.t. $h(m_1) = h(m_2)$
  - Birthday Attack

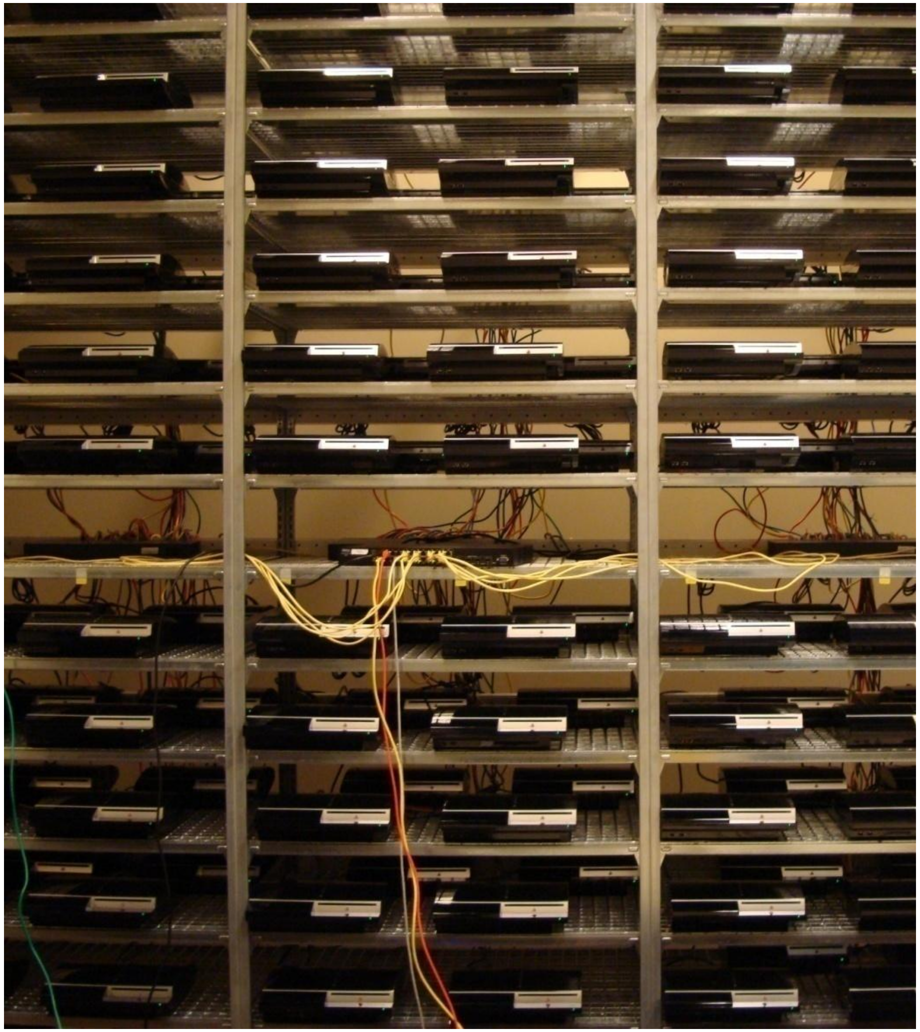
MD5 Hash Function

- 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



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)



MD5 Collision

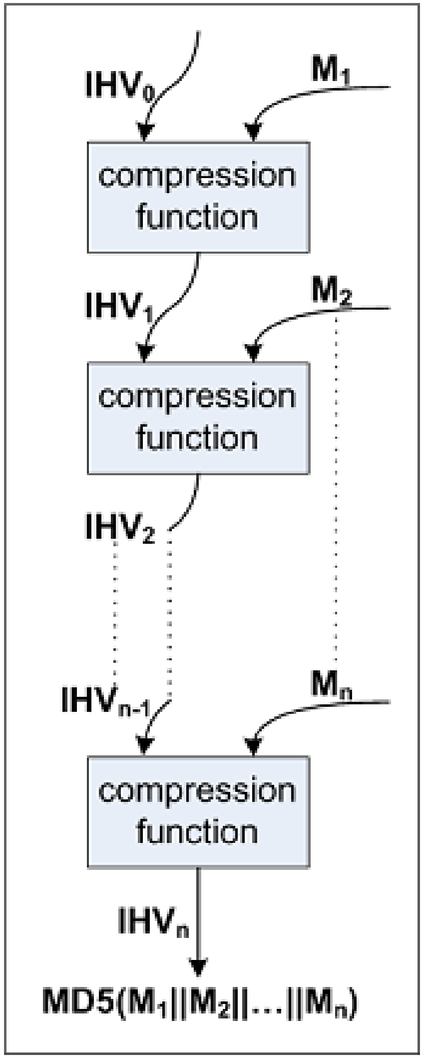
```
d131dd02c5e6eec4693d9a0698aff95c 2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e488832571415a 085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6 dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1e c69821bcb6a8839396f9652b6ff72a70

d131dd02c5e6eec4693d9a0698aff95c 2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a 085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6 dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080280d1e c69821bcb6a8839396f965ab6ff72a70
```

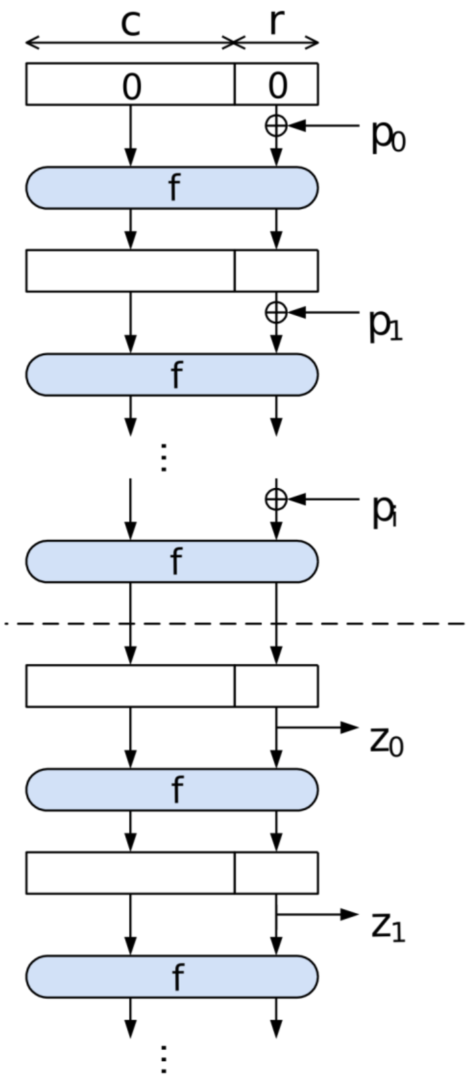
Both of these blocks hash to 79054025255fb1a26e4bc422aef54eb4

SHA Hash Functions

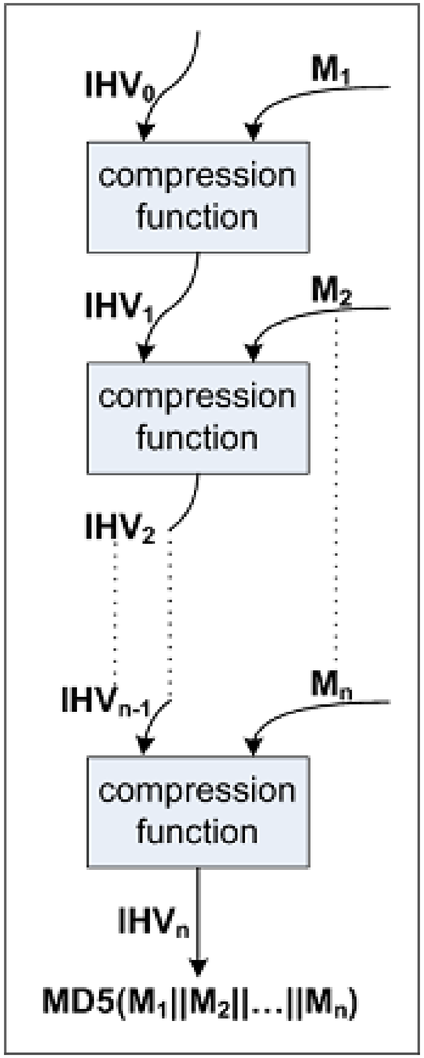
- SHA-1 – standardized by NIST in 1995
  - 160-bit output and internal state
  - 512-bit block size
- SHA-2 – extension published in 2001
  - 256 (or 512)-bit output and internal state
  - 512 (or 1024)-bit block size
- SHA-3 – chosen by NIST in 2012
  - 256 (512)-bit output
  - Different "sponge" construction

Block chaining vs. Sponge-construction





Tricky! Length Extension Attacks



Given hash of secret *x*, trivial to find hash of *x* || *p* || *m* for padding *p* and arbitrary *m*

MD5 and SHA family all vulnerable!

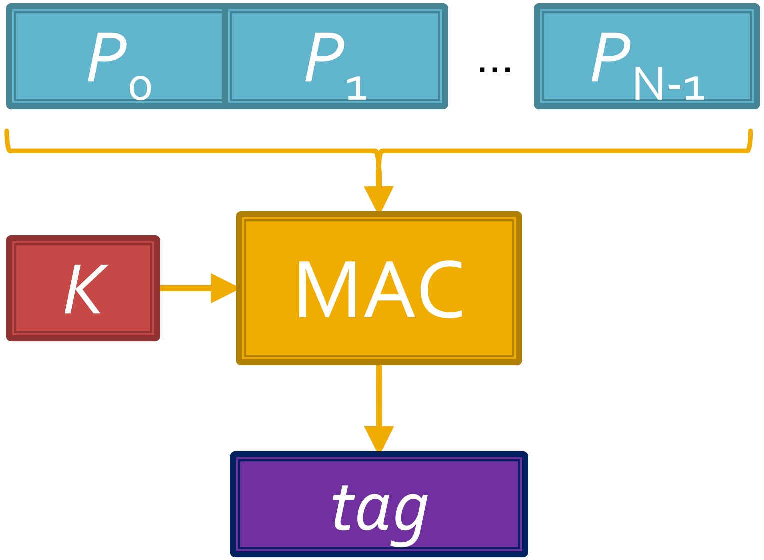
Is SHA-1 Safe?

- Significant cryptanalysis since 2005
- Improved attacks show complexity of finding a collision < 2^{51}(ideally security would be 2^{80} – why?)
- Attacks only get better …

- **Use SHA-256**

Message Authentication Codes

- **Prevents tampering with messages.**
  Like a *family* of pseudorandom functions, with a key to select among them

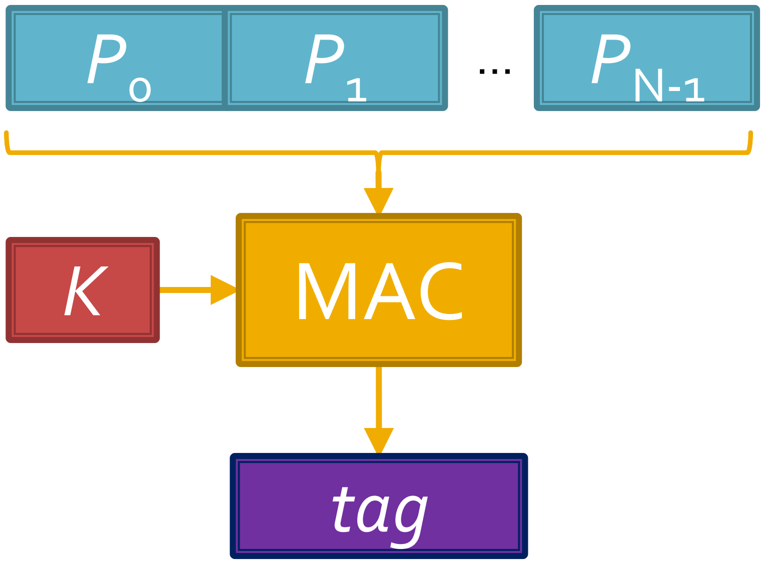


MAC Security Properties

- Attacker given a MAC oracle: (unknown K)

  MAC(K,   )

- Must discover a new MAC output:
  - MAC(K,   );



Construction: HMAC

Given a hash function H:

HMAC(K,m) = H(($K \oplus$ pad1) || H($K \oplus$ pad2 || m))
for constants pad1 and pad2

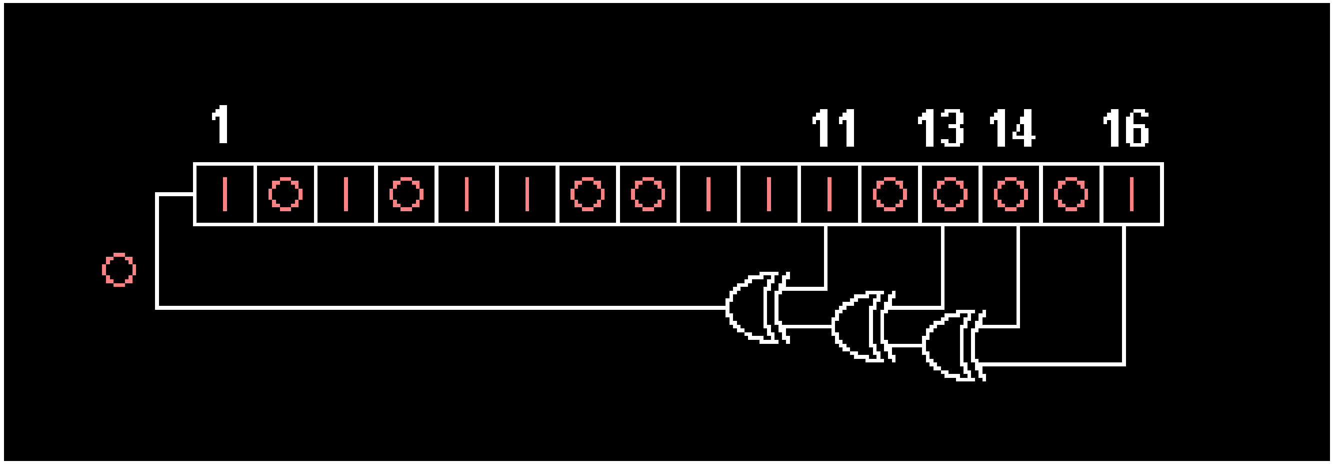
Provides nice provable security properties

What Should You Use?

- **Use HMAC-SHA256**
  - Use a constant key to get a Length-extension resistant hash function

Generating Random Numbers

- What's wrong with srand() and rand()?



Generating Random Numbers

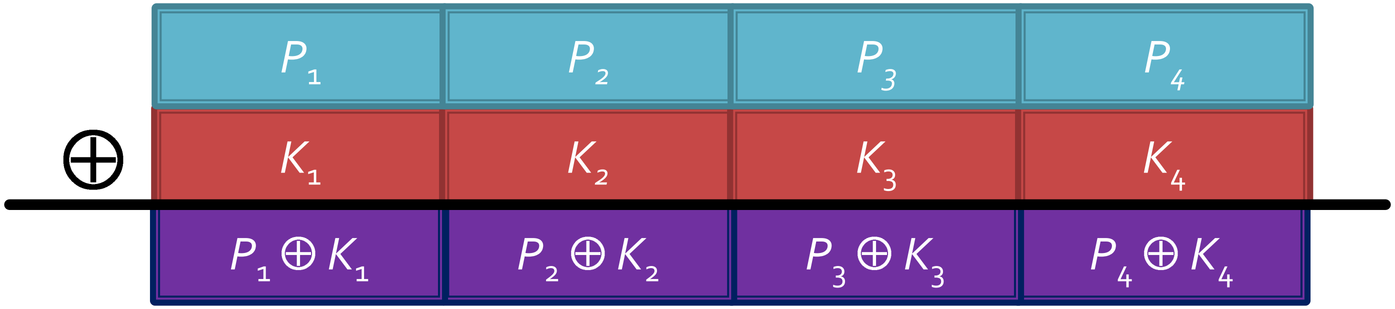
- What's wrong with srand() and rand()?
- Why not use a secure hash?
  - "Cryptographic Pseudorandom Number Generator" (CPRNG)
- Tricky details…
  - Seeding with true randomness ("entropy")
  - Forward secrecy
- Most OSes do the hard work for you*
  - On Linux, use `/dev/random` and `/dev/urandom`

One-Time Pads

Provably secure encryption…

… that often fails in practice.

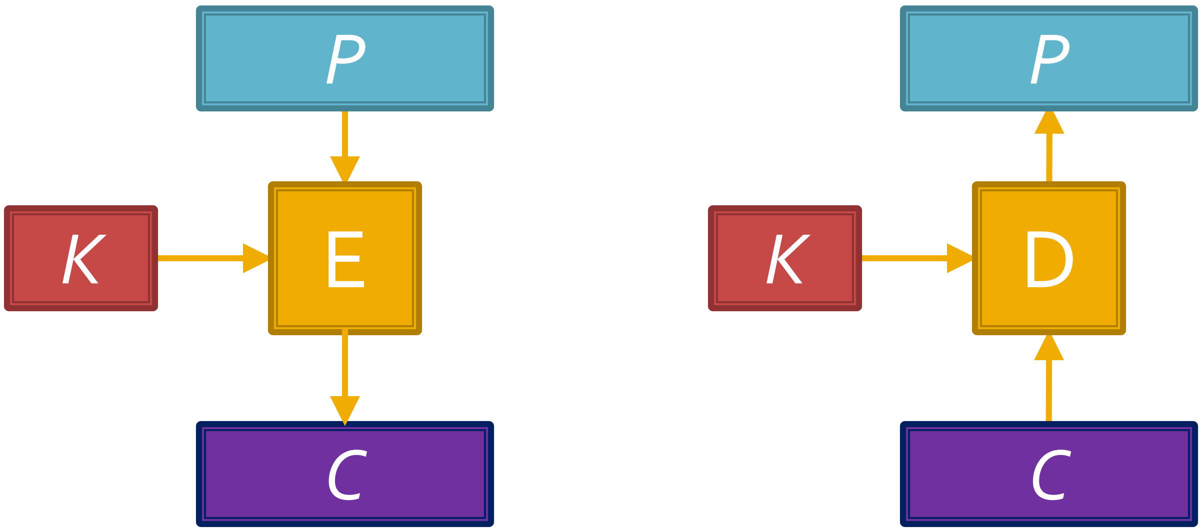
One-Time Pads



$P_i \oplus K_i$	P_i	K_i
0	0	0
0	1	1
1	0	1
1	1	0

Block Ciphers

- Ideal block cipher:
  Like a *family* of pseudorandom *permutations* with a key to select among them

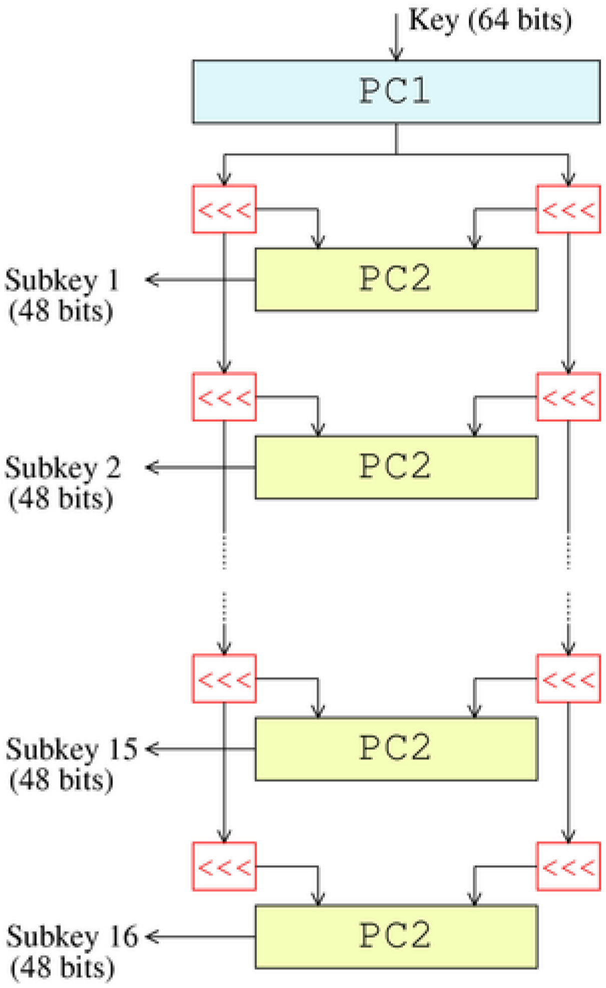


DES—Data Encryption Standard

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

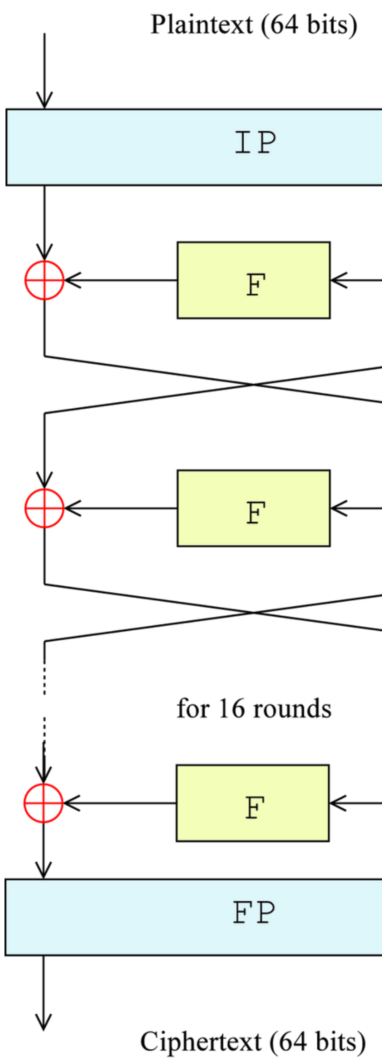
- 56-bit *key*
- 64-bit *blocks*
- 16 *rounds*

- Key schedule function generates 16 round keys:



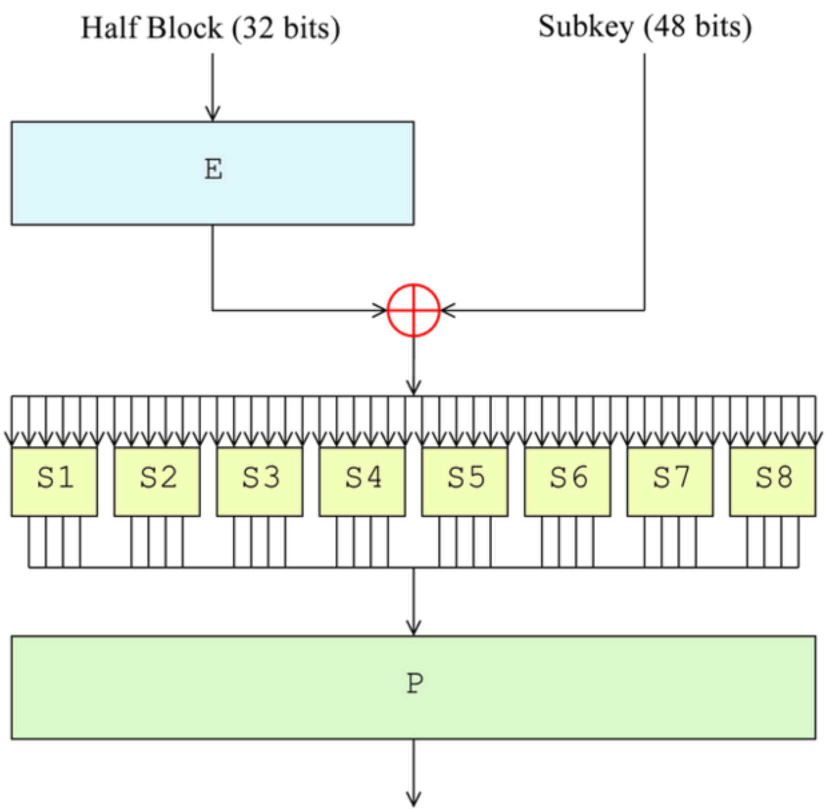
DES Encryption

- 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)



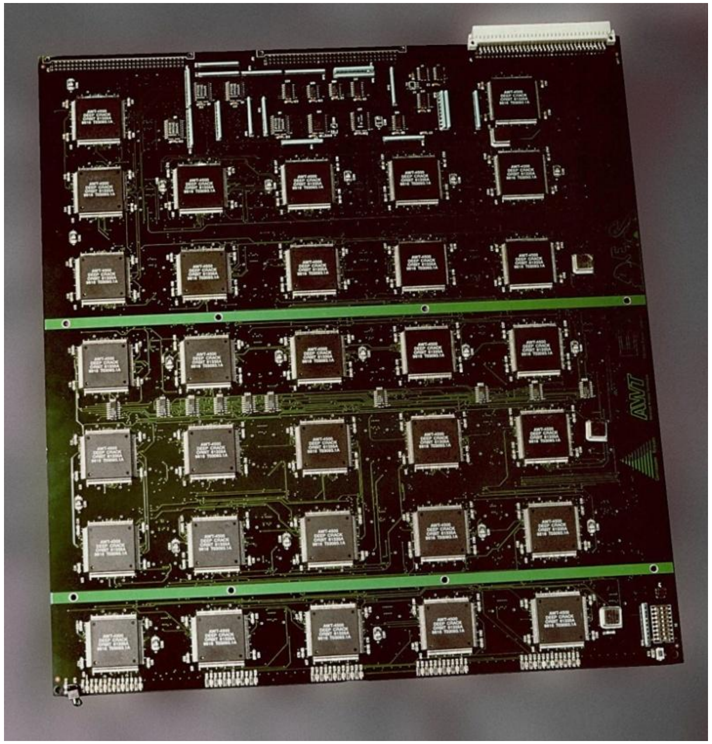
DES Feistel Function

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



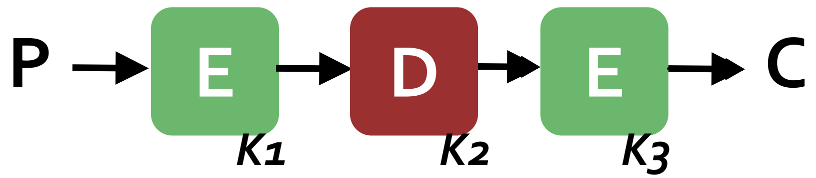
DES is Unsafe – Don't Use It!

- 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)



3DES

- $E_{K_1, K_2, K_3}(P) = E_{K_3}(D_{K_2}(E_{K_1}(P)))$

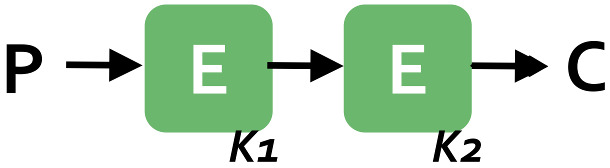


- Key options:
  - Option 1: independent keys (56*3 = 168 bit key)
  - Option 2: $K_1 = K_3$ (56*2 = 112 bit key)
  - Option 3: $K_1 = K_2 = K_3$ (Backward-compatible DES)

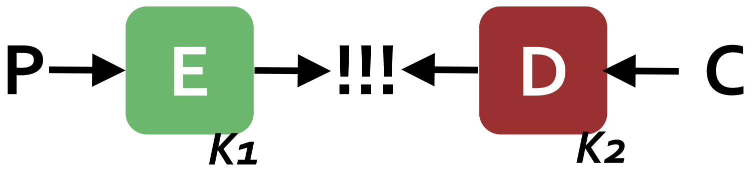
- What happened to 2DES?

2DES: Meet-in-the-middle attack

- "2DES": $E_{K_1, K_2}(P) = E_{K_2}(E_{K_1}(P))$



- Given P and $C = E_{K_2}(E_{K_1}(P))$, find both keys
  - For all K, generate $E_K(P)$ and $D_K(C)$
  - Find a match where $D_{K_2}(C) == E_{K_1}(P)$



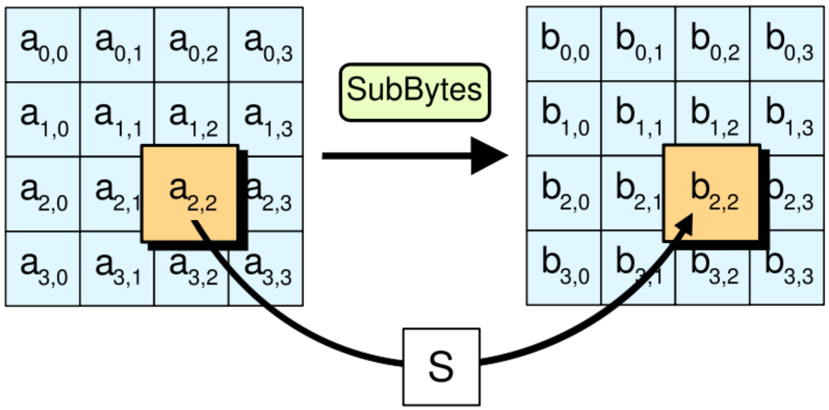
AES—Advanced Encryption Standard

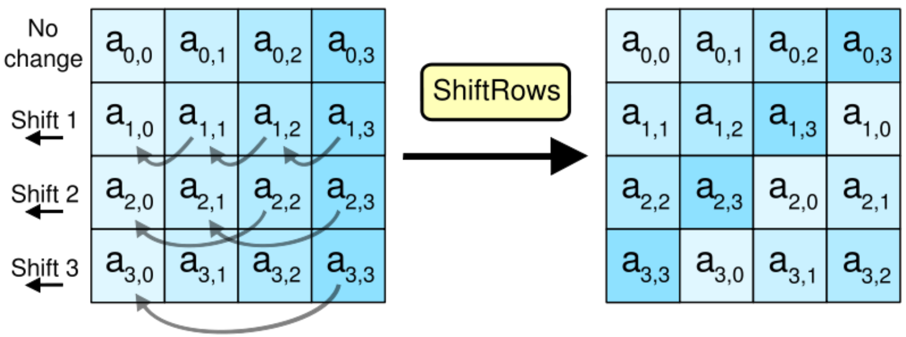
- Standardized by NIST in 2001 following open design competition (a.k.a. Rijndael)

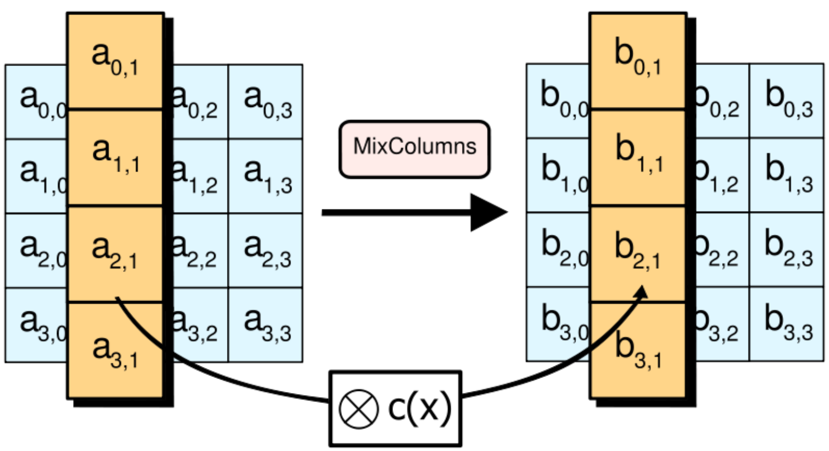
- 128-, 192-, or 256-bit key
- 128-bit blocks
- 10, 12, or 14 rounds

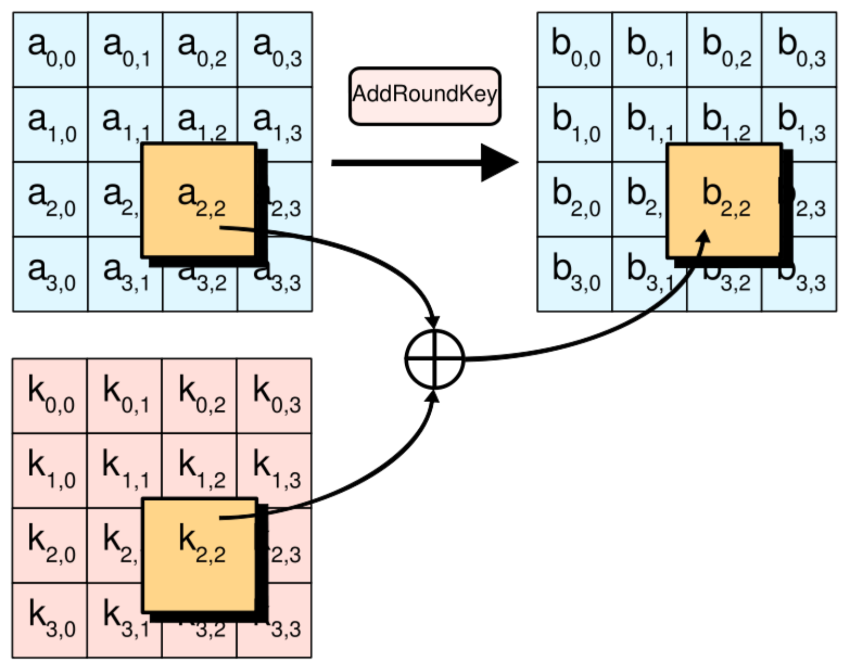
- Not a Feistel-network construction

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

Reading for Tuesday

- Crypto notes (on course website)
- No written response required

Tuesday's Class

Essential Crypto II:

Cipher Modes
Secure Channels
Key Exchange
Public-Key Crypto
Establishing Trust