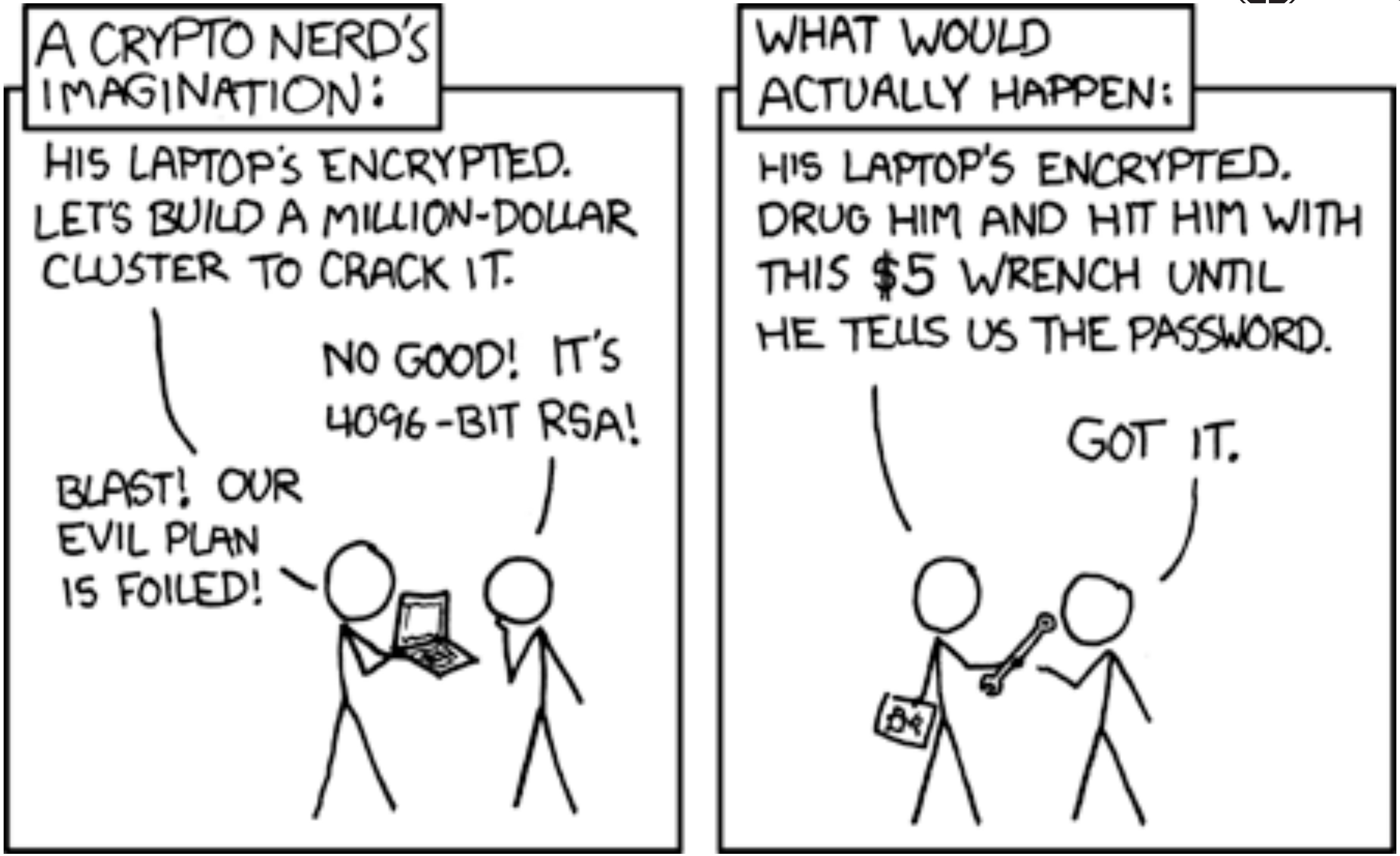
Modern Ciphers

Modern Ciphers

Oregon State University College of Engineering

- Modern ciphers are **product ciphers**
- Are **only computationally secure**
  - i.e., adversary with unlimited computing power can break them
- Current good security: force the adversary to do $\mathbf{2^{128}}$(~10^{38} or 100 trillion trillion trillion!!) computations
  - Current fastest supercomputer (442 PETAFLOPS or 10^{15} FLOPS/second with 7.3 million cores will take 3.17 quadrillion years!)

Oregon State University
College of Engineering



Current Encryption Standard in USA

Oregon State University
College of Engineering

- Data Encryption Standard (DES) until 2001
  - Block-cipher
  - encrypts blocks of 64 bits using a 56 bit key
  - outputs 64 bits of ciphertext

- Advanced Encryption Standard since 2001
  - Block-cipher
  - encrypts blocks of 128 bits using keys of 128/196/256 bits
  - outputs 128 bits of ciphertext

Avalanche Effect

Oregon State University
College of Engineering

- Key desirable property of an encryption algorithm
- Where a change of **even one input or key bit** results in changing approx. **half (50%) of the output bits**
- If the change were small, this might provide a way to reduce the size of the key space to be searched
- DES and AES exhibit strong avalanche

DES Controversy

Oregon State University
College of Engineering

- DES considered weak even when it was a standard

- Diffie, Hellman said (in 1999) in a few years technology would allow DES to be broken in days
  - Design using 1999 technology published

- Design decisions not public
  - NSA controlled process
  - Some of the design decisions are unknown; suspected backdoors
  - Key size reduced from 112 bits in original Lucifer design to 56 bits

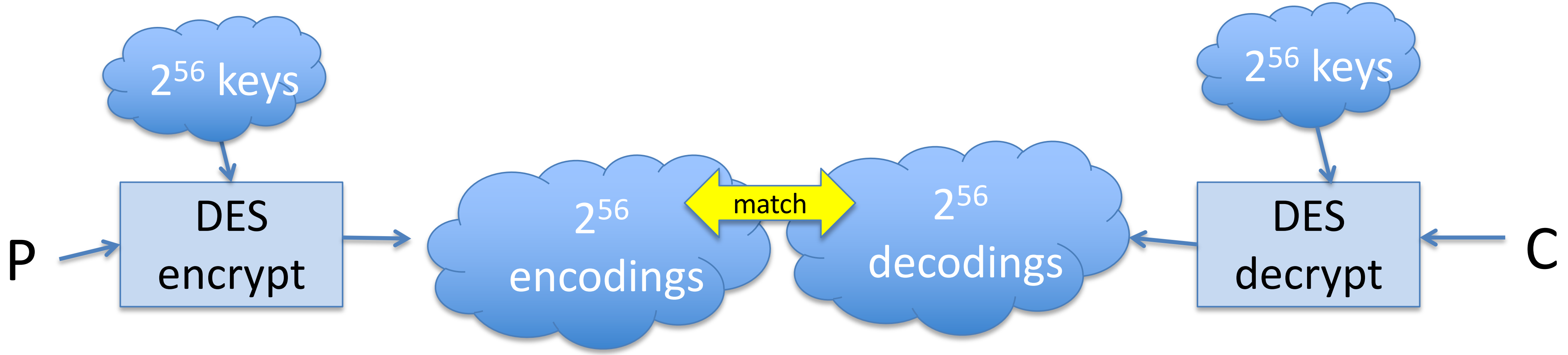
Brute Force Attack on DES

- What do you need?
- How many steps should it take?
- How can you do better?

Double DES

Oregon State University
College of Engineering

- Double encryption not generally used
  - $C = E_{k2}(E_{k1}(P))$
    - Encode twice, using 2 different keys
  - Susceptible to "Meet in the Middle (MTM) attack
    - Suppose you have plaintext P and corresponding ciphertext C



Modifies brute force to require only 2^{n+1} steps instead of 2^{2n}

Meet in the Middle Attack

Cyphertext C created by using encryption function E twice, first with key K_1, then with key K_2:

$$C = E_{K_2}(E_{K_1}(P))$$

Meet in the Middle Attack

Oregon State University
College of Engineering

Cyphertext C created by using encryption function E twice, first with key K_1, then with key K_2:

$$C = E_{K_2}(E_{K_1}(P))$$

Notice that

$$\begin{aligned} C &= E_{K_2}(E_{K_1}(P)) \quad \text{implies} \\ D_{K_2}(C) &= D_{K_2}(E_{K_2}(E_{K_1}(P))) \quad \text{implies} \\ D_{K_2}(C) &= E_{K_1}(P) \end{aligned}$$

Meet in the Middle Attack

This implies that we can

- compute all possible encodings of P (through all possible keys) $S_P = \{(k, E_k(P))\}$

Meet in the Middle Attack

Oregon State University
College of Engineering

This implies that we can

- compute all possible encodings of P (through all possible keys) $\mathcal{S}_P = \{(k, E_k(P))\}$
- compute all possible decodings of C (through all possible keys) $\mathcal{D}_C = \{(k, D_k(C))\}$
  - for each $(k_2, D_{k_2}(C))$, look for $(k_1, E_{k_1}(P) \in \mathcal{S}_P$ such that $E_{k_1}(P) = D_{k_2}(D)$.

Meet in the Middle Attack

This implies that we can

- compute all possible encodings of P (through all possible keys) $\mathcal{S}_P = \{(k, E_k(P))\}$
- compute all possible decodings of C (through all possible keys) $\mathcal{D}_C = \{(k, D_k(C))\}$
  - for each $(k_2, D_{k_2}(C))$, look for $(k_1, E_{k_1}(P) \in \mathcal{S}_P$ such that $E_{k_1}(P) = D_{k_2}(D)$.
- Match identifies keys used k_1, k_2, can be used to decode other messages thought to use those keys

Meet in the Middle Attack

This implies that we can

- compute all possible encodings of P (through all possible keys) $\mathcal{S}_P = \{(k, E_k(P))\}$
- compute all possible decodings of C (through all possible keys) $\mathcal{D}_C = \{(k, D_k(C))\}$
  - for each $(k_2, D_{k_2}(C))$, look for $(k_1, E_{k_1}(P) \in \mathcal{S}_P$ such that $E_{k_1}(P) = D_{k_2}(D)$.
- Match identifies keys used k_1, k_2, can be used to decode other messages thought to use those keys
- for a given search, matching can be done in $O(\log 2^{len(K)})$ time

Meet in the Middle Attack

This implies that we can

- compute all possible encodings of P (through all possible keys) $\mathcal{S}_P = \{(k, E_k(P))\}$
- compute all possible decodings of C (through all possible keys) $\mathcal{D}_C = \{(k, D_k(C))\}$
  - for each $(k_2, D_{k_2}(C))$, look for $(k_1, E_{k_1}(P) \in \mathcal{S}_P$ such that $E_{k_1}(P) = D_{k_2}(D)$.
- Match identifies keys used k_1, k_2, can be used to decode other messages thought to use those keys
- for a given search, matching can be done in $O(\log 2^{len(K)})$ time

Overall complexity is just $2 \times \log 2^{len(K)}$ larger than brute force on one encoding.

Triple DES (or TDEA)

Oregon State University
College of Engineering

- Encrypt-Decrypt-Encrypt Mode (2 or 3 keys: k, k', k''))
  - $c = \text{DES}_k(\text{DES}_{k'}^{-1}(\text{DES}_{k''}(m)))$

- Decrypt-Encrypt-Decrypt
  - $m = \text{DES}_{k''}^{-1}\ (\text{DES}_{k'}(\text{DES}^{-1}{}_k(c)))$

- Protects against MTM : middle operation inverse of 1st and 3rd

Triple DES – Keying Options

Oregon State University
College of Engineering

- All three keys independent (Keying Option 1)
  - 168 key bits
  - Only 112 bit security because of MITM attacks
  - Strongest among the three keying options

- k and k' are independent, $k = k''$ (Keying Option 2 or 2TDEA)
  - 112 key bits
  - Considered to have <= 80 bit strength
  - Not recommended for encryption use after December 31st 2015

- All three keys are identical
  - backwards compatible with DES

- 3DES not yet practical to break but AES much faster

- Encrypt-Encrypt-Encrypt Mode (3 keys: k, k', k'')
  - $c = \mathrm{DES}_k(\mathrm{DES}_{k'}(\mathrm{DES}_{k''}(m)))$ (only 112 bits of safety due to MTM)

Current Status of DES

- A design for computer system and an associated software that could break any DES-enciphered message in a few days was published in 1998
- Several challenges to break DES messages solved using distributed computing
- National Institute of Standards and Technology (NIST) selected Rijndael as Advanced Encryption Standard (AES), successor to DES
  - Designed to withstand attacks that were successful on DES
  - It can use keys of varying length (128, 192, or 256)

AES Background

- Clear a replacement for DES was needed
  - Can use Triple-DES –but slow with small blocks
- US NIST issued call for ciphers in 1997
  - 15 candidates accepted in Jun 98
  - 5 were short-listed in Aug-99
- Rijndael (designed by Rijmen-Daemenin of Belgium) was selected as AES in Oct-2000
  - issued as NIST FIPS PUB 197 standard in Nov. 2001

AES Requirements

Oregon State University
College of Engineering

- Private key symmetric block cipher
  - 128-bit data, 128/192/256-bit keys
- Stronger & faster than Triple-DES
- Active life of 20-30 years (+ archival use)
- Provide full specification & design details
- Both C & Java implementations
- NIST has released all submissions & unclassified analyses

Summary

Oregon State University
College of Engineering

- Block vs. Stream Ciphers
- DES
  - Symmetric key block cipher
  - Yesteryear’s workhorse algorithm
  - Product cipher
- AES
  - Symmetric key block cipher
  - Today's workhorse algorithm
  - Product cipher
- 3DES in use but slower than AES

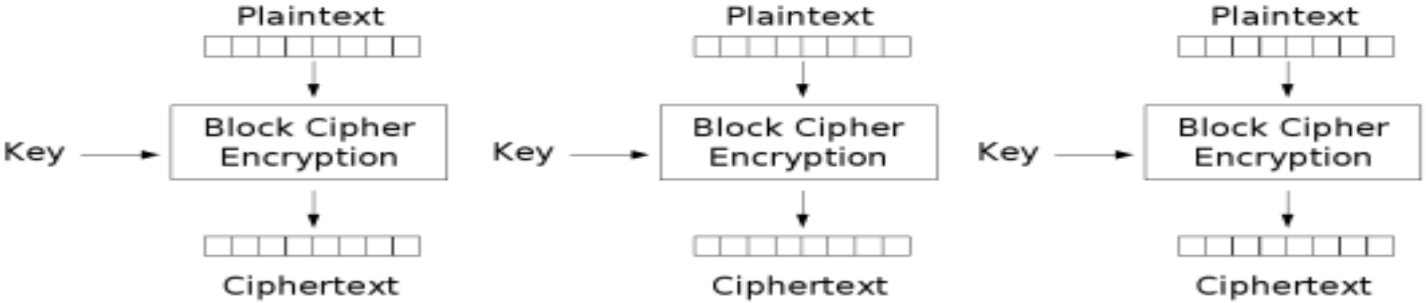
Encryption Modes

What is an Encryption Mode?

A way for block ciphers to encrypt messages larger than their block size

Electronic Code Book (ECB)

- Divide message into blocks
- Each block enciphered independently



Electronic Codebook (ECB) mode encryption

Stream vs. Block Ciphers

Oregon State University
College of Engineering

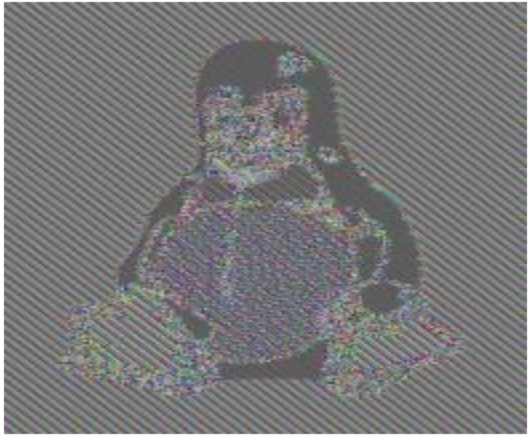
- E encipherment function
  - $E_k(b)$ encipherment of message b with key k
  - In what follows, $m = b_1 b_2 \ldots$, each b_i of fixed length
- Block cipher
  - $E_k(m) = E_k(b_1)E_k(b_2) \ldots$
- Stream cipher
  - $k = k_1 k_2 \ldots$
  - $E_k(m) = E_{k1}(b_1)E_{k2}(b_2) \ldots$
  - If $k_1 k_2 \ldots$ repeats itself, cipher is *periodic* and the length of its period is one cycle of $k_1 k_2 \ldots$

ECB Problem

- Problem: identical plaintext blocks produce identical ciphertext blocks
  - Example: two database records
    - `MEMBER: HOLLY INCOME $100,000`
    - `MEMBER: HEIDI INCOME $100,000`
  - Encipherment
    - `ABCQZRME GHQMRSIB CTXUVYSS RMGRPFQN`
    - `ABCQZRME ORMPABRZ CTXUVYSS RMGRPFQN`
- Leaks patterns (information)!

ECB Problem in Pictures

Original image (bitmap)



Encrypted using ECB

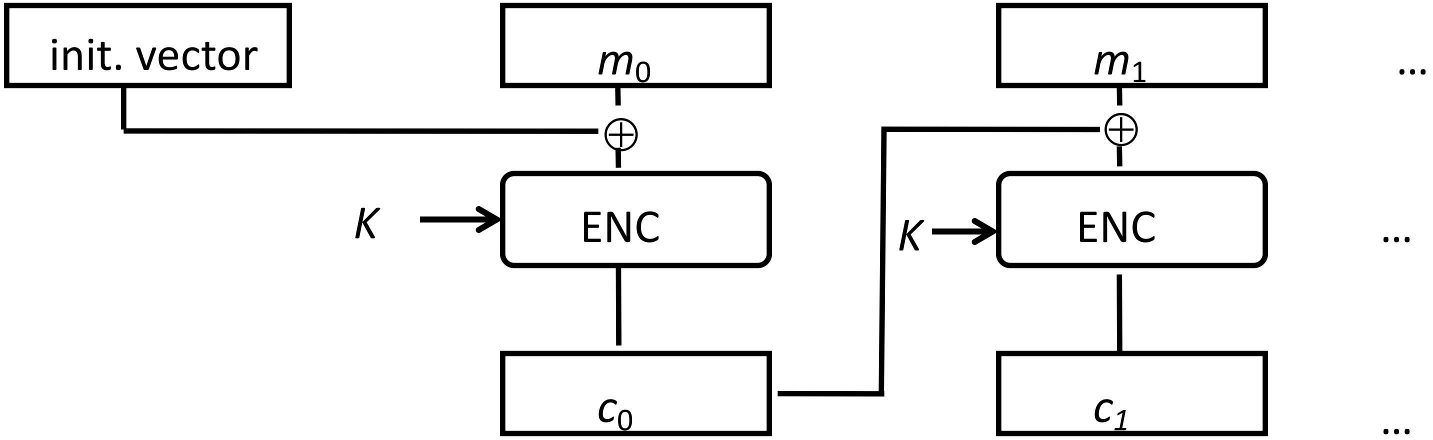
Solutions

Oregon State University
College of Engineering

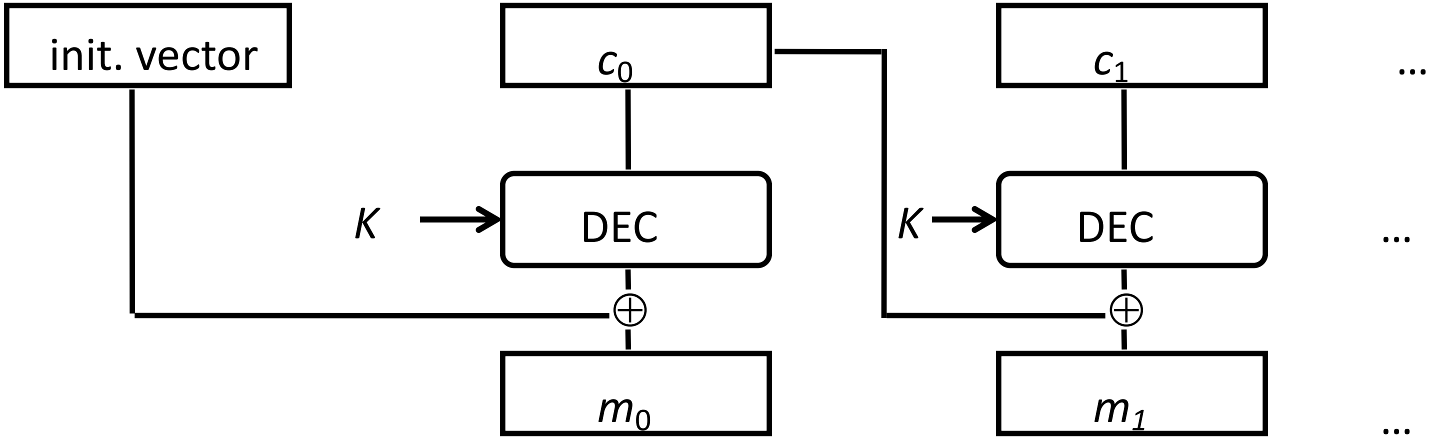
- Insert information about block's position into the plaintext block, then encipher
  - Variety of ways one might encode "position"
    - $c_i = E_k(m_i \oplus i)$ for i > 0, or
    - $c_i = E_k(m_i \oplus f(i))$ for i > 0
    - $c_0 = E_k(m_0 \oplus I)$ for i = 0, where I is some random number
  - Trick is to use something the receiver knows and so can apply XOR in reverse when decoding.
- *Cipher block chaining (CBC)*:
  - Exclusive-or current plaintext block with previous ciphertext block:
    - $c_i = E_k(m_i \oplus c_{i-1})$ for i > 0
    - $c_0 = E_k(m_0 \oplus I)$
- where I is the initialization vector

CBC Mode Encryption

Oregon State University
College of Engineering



CBC Mode Decryption

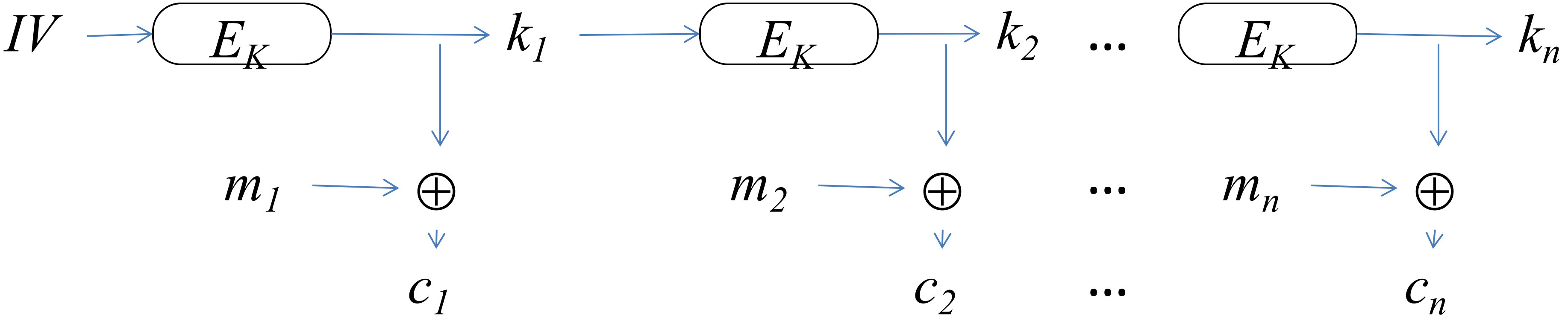


Self-Healing Property

Oregon State University
College of Engineering

- If one block of ciphertext is altered, the error propagates for at most two blocks
- Initial message
  - 3231343336353837 3231343336353837 3231343336353837 3231343336353837
- Received as (underlined 4c should be 4b)
  - ef7c<u>4c</u>b2b4ce6f3b f6266e3a97af0e2c 746ab9a6308f4256 33e60b451b09603d
- Which decrypts to
  - <u>efca61e19f4836f1</u> 3231<u>33</u>3336353837 3231343336353837 3231343336353837
  - Incorrect bytes underlined
  - Plaintext "heals" after 2 blocks

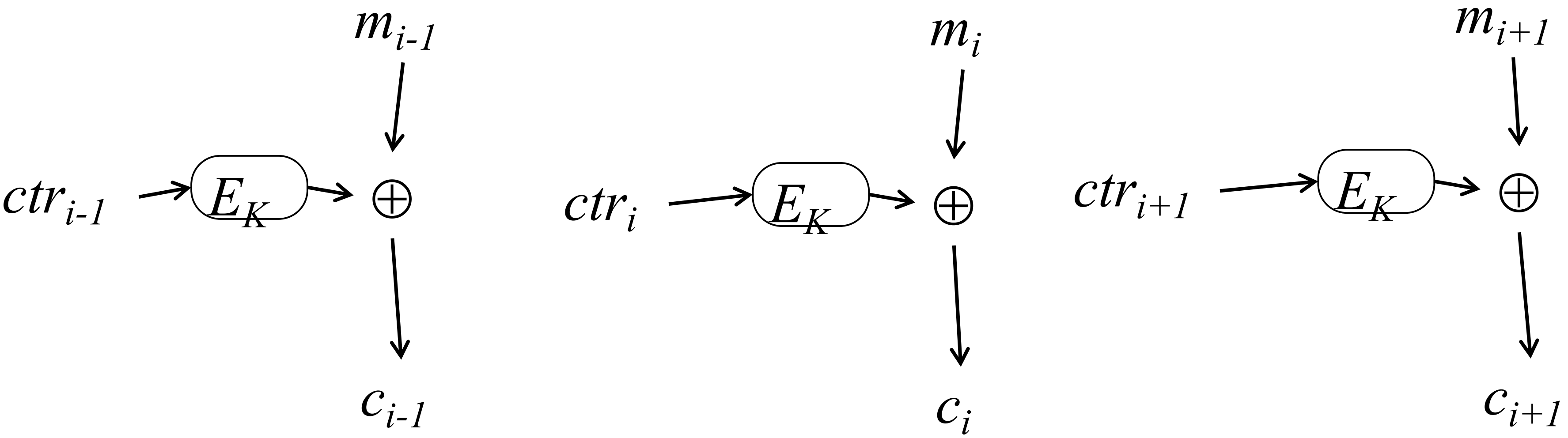
Output Feedback Mode (OFB)



Successively encode the key (IV)

Counter Mode

Successively encode a counter value



Comments on OFB/Counter

Oregon State University
College of Engineering

- Additional standard modes for DES/AES
- Losing Synchronicity is fatal
  - All later decryptions will be garbled
- OFB needs an initialization vector
- Counter mode lets you generate a bit in the middle of the stream. Lets you operate on blocks in parallel.

Many Other Modes Exist

Oregon State University
College of Engineering

- Propagating CBC (PCBC)
- Cipher Feedback (CFB)
- XCBC, CCM, EAX, GCM etc.

Summary

- Electronic Code Book (ECB) not desirable

Some currently used modes

- Cipher-Block Chaining (CBC)
- Counter Mode (CTR)
- Output Feedback Mode (OFB)

Cryptographic Hash Functions

Motivation/Context

- How can I be sure that a file stored/communicated has NOT been modified

Hash or Checksums

- Mathematical function to generate a set of k bits from a set of n bits (where $k \leq n$).
  - k is smaller then n except in unusual circumstances

- Example: ASCII parity bit
  - ASCII has 7 bits; 8th bit is “parity”
  - Even parity: even number of 1 bits
  - Odd parity: odd number of 1 bits

Example Use

- Bob receives “10111101” as bits.
  - Sender is using even parity; 6 1 bits, so character was received correctly
    - Note: could be garbled, but 2 bits would need to have been changed to preserve parity
  - Sender is using odd parity; even number of 1 bits, so character was not received correctly

Another Example

Oregon State University
College of Engineering

- 8-bit Cyclic Redundancy Check (CRC)
  - XOR (exclusive-OR) all bytes in the file/message
  - Good for detecting accidental errors
  - But easy for malicious user to "fix up" to match altered message

- For example, change the 4th bit in one of the bytes
  - Fix up by flipping the 4th bit in the CRC

- Easy to find a M′ that has the same CRC

Secure Hash functions

Oregon State University
College of Engineering

- Crytpo Hash or Checksum
  - Unencrypted one–way hash functions
  - Easy to compute hash
  - Hard to find message with a particular hash value
  - Use to verify integrity of publicly available information
    - E.g., packets posted on mirror sites
- Message Authentication Code (MAC)
  - Hash to pass along with message
  - Such a hash must be accessed with key
    - Otherwise, attacker could change MAC in transit

Cryptographic Hash or Checksum

Oregon State University
College of Engineering

- h: $A \rightarrow B$:
  - Efficient: For any $x \in A$, $h(x)$ is easy to compute
  - Pre-image Resistance: For any $y \in B$, it is computationally infeasible to find $x \in A$ such that $h(x) = y$
    - E.g., computing x^3 given x vs. computing cube root of x^3 by hand
  - Weak-Collision Resistance: Given any $x \in A$, it is computationally infeasible to find $x' \in A$ such that $h(x) = h(x')$ and $x \neq x'$
  - Strong Collision Resistance: It is computationally infeasible to find two inputs $x, x' \in A$ such that $x \neq x'$ and $h(x) = h(x')$

Collisions

- If $x \neq x'$ and $h(x) = h(x')$, x and x' are a *collision*
  - There are many many … collisions in a hash function
  - Pigeonhole principle: if there are n containers for n+1 objects, then at least one container will have 2 objects in it.
  - Application: if there are 32 files and 8 possible cryptographic checksum values, at least one value corresponds to at least 4 files
  - How many files until you are guaranteed a collision?

Birthday Paradox

- What is the probability that someone in the room with n people has the same birthday as me?
  - Assuming uniform birthday distribution over 365 days
  - $P(n) = 1 - (364/365)^n$
  - $P(n) > 0.5$ for $n = 253$
  - $P(n) = 0.63$ for $n = 365$

- What is the probability that two people in a room with n people have the same birthday?
  - $P(n) = 1 – (365!/(365^n * (365-n)!)$
  - $P(n) > 0.5$ for $n = 23$
  - $P(n) = 0.999$ for $n = 70$; $P(n) = 0.9999997$ for $n = 100$

Birthday Paradox

- In general, probability of a collision reaches 50% for M units when
  - $N = sqrt(M)$
- If hash has m bits, this means M = 2^m possible hash values
  - $n = 2^{m/2}$ for 50% probability collision
  - n is # of tries

Another View of Collisions

Oregon State University College of Engineering

- **Birthday attack** works thus:
  - opponent generates $2^{m/2}$ variations of a valid message all with essentially the same meaning
  - opponent also generates $2^{m/2}$ variations of a desired fraudulent message
  - two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
  - have user sign the valid message, then substitute it with the forgery which will have a valid signature

- **Need to use larger Hashes**

MD5 and SHA

Oregon State University College of Engineering

- Keyless crypto hashes

- Both are round based bit operations
  - Similar in spirit to AES and DES
  - Looking for avalanche effect to make output appear random

- MD5 is 128 bits and SHA-1 is 160 bits
  - **MD5 is no longer recommended for cryptographic use. Shown not to be collision resistant in 2005**
  - **SHA-1 has also been broken**

More on SHA

- Standard put forth by NIST
- Comes in different output sizes
  - SHA-1 outputs 160 bits
  - The other SHA-X flavors output X bits - e.g., 256, 512

SHA-1 Broken

- Researchers had a break-through recently (~2005)
  - Results show that you can find collisions in 2^69 attempts which would be less than 2^80 (brute force)
  - ~2000 time faster!
- Google and CWI team showed a real collision in 2017
- SHA-1 is no longer recommended for applications requiring collision resistance
- NIST published standards promoting use of larger SHA's
  - SHA-256 and SHA-512

Key Points

- Data integrity is important
  - Sometimes more important than confidentiality
- Cryptographic hashes help us detect unauthorized changes
- Birthday paradox -> use hash functions with larger outputs
- SHA-256 and SHA-512 are current recommended standards

Message Authentication Codes (MACs)

Message Authentication Codes

Oregon State University
College of Engineering

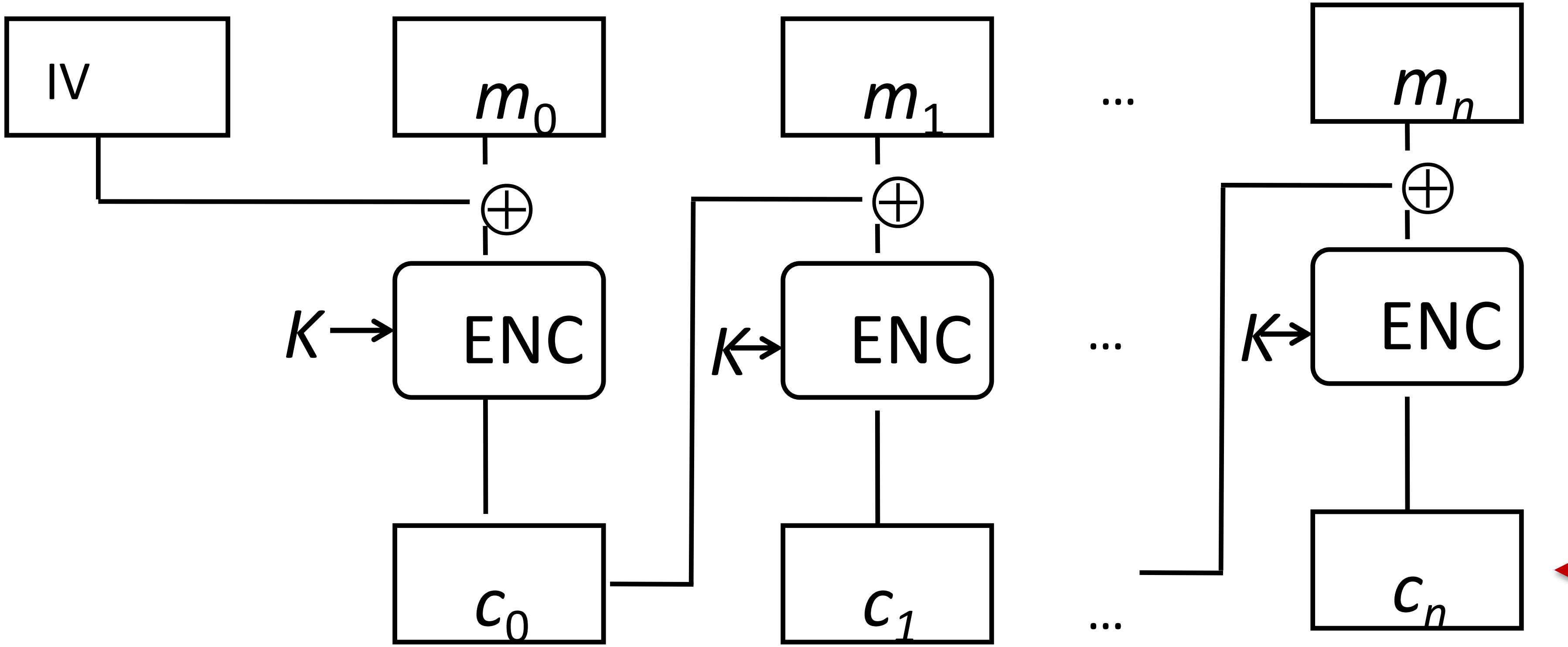
- MAC is a crypto hash that is a proof of a message's integrity
  - Important that adversary cannot fixup MAC if he changes message
- MAC's rely on keys to ensure integrity
  - Either Crypto Hash is encrypted
  - Or Crypto Hash must be augmented to take a key

Use Symmetric Ciphers for Keyed Hash

- Can use DES or AES in CBC mode
  - Last block is the hash

- DES with 64 bit block size is too small to be effective MAC
  - Why?

CBC-MAC



CBC-MAC = C_n

Add Key to Keyless Hash

Oregon State University
College of Engineering

- Sender and receiver agree on key K and crypto-hash h()
- $MAC_K(m) = h(K \,||\, m \,||\, K)$
- Send $MAC_K(m)$ and m
- Receiver can verify $MAC_K(m)$
- Interceptor does not know K, so he cannot re-compute $MAC_K(m)$

HMAC

- Make keyed cryptographic checksums from keyless cryptographic checksums

- *h*() is keyless cryptographic checksum function that takes data in blocks of *b* bytes and outputs blocks of *l* bytes. *K′* is cryptographic key of length *b* bytes

- *ipad* is 00110110 repeated *b* times

- *opad* is 01011100 repeated *b* times

- HMAC-$h(k', m) = h(k' \oplus opad \,||\, h(k' \oplus ipad \,||\, m))$

  $\oplus$ exclusive or, || concatenation

HMAC-SHA512

Oregon State University
College of Engineering

- Apply HMAC to SHA512 to make a keyed MAC

- HMAC-SHA512(k′, m) =

$$\text{SHA512}(k' \oplus [01011100]^8 \;||\; \text{SHA512}(k' \oplus [00110110]^8 \;||\; \text{m}))$$

HMAC and Strong Collisions

Oregon State University
College of Engineering

- Birthday attacks don't make sense in HMAC scenario – why?
  - Attacker would need to know K to generate candidate message/hash pairs
  - Is HMAC-SHA1 still a reasonable option?

Key Points

Oregon State University
College of Engineering

- MACs are cryptographic hashes with keys
- MACs are used when the hash has to be transmitted along with the message/data
- Both encryption ciphers and cryptographic hashes can be used to construct MACs
- Birthday attacks don't apply to HMAC