CS 361S

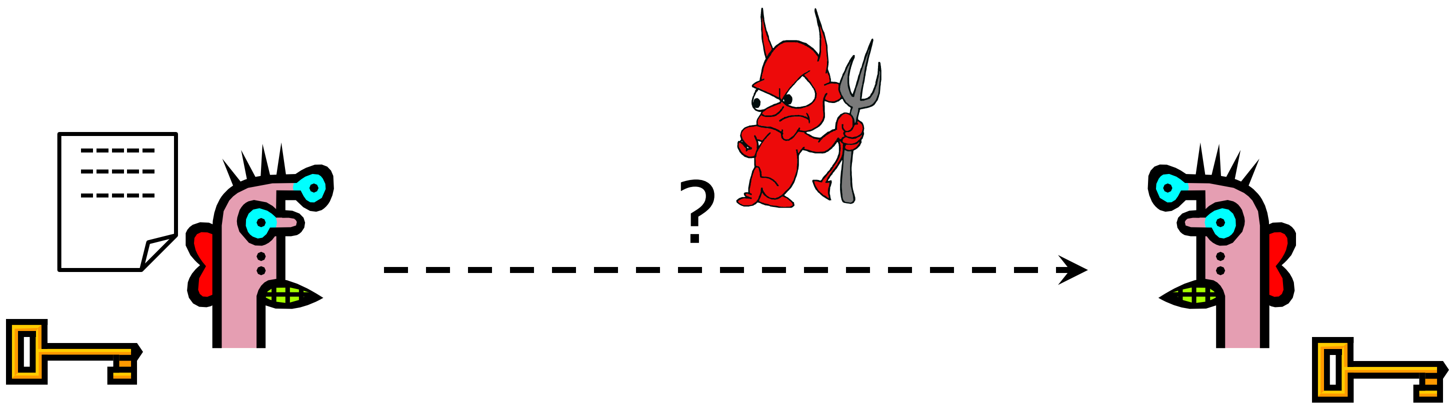
Overview of Symmetric Encryption

Vitaly Shmatikov

Reading Assignment

- Read Kaufman 2.1-4 and 4.2

Basic Problem



Given: both parties already know the same secret

Goal: send a message confidentially

How is this achieved in practice?

Any communication system that aims to guarantee confidentiality must solve this problem

Kerckhoffs's Principle

- An encryption scheme should be secure even if enemy knows everything about it except the key
  - Attacker knows all algorithms
  - Attacker does not know random numbers
- Do not rely on secrecy of the algorithms ("security by obscurity")

Easy lesson:
use a good random number generator!

Full name:

Jean-Guillaume-Hubert-Victor-François-Alexandre-Auguste Kerckhoffs von Nieuwenhof

Randomness Matters!

The New York Times

Business Day

Technology

WORLD | U.S. | N.Y. / REGION | BUSINESS | TECHNOLOGY | SCIENCE | HEALTH | SPORTS | OPINION | A

Flaw Found in an Online Encryption Method

By JOHN MARKOFF
Published: February 14, 2012

SAN FRANCISCO — A team of European and American mathematicians and cryptographers have discovered an unexpected weakness in the encryption system widely used worldwide for online shopping, banking, e-mail and other Internet services intended to remain private and secure.

The flaw — which involves a small but measurable number of cases — has to do with the way the system generates random numbers, which are used to

m
at
m

Internet users, there is nothing a
sites will need to make changes t
said

RECOMMEND
TWITTER
LINKEDIN
COMMENTS (127)
E-MAIL
PRINT
SINGLE PAGE
REPRINTS

Readers' Comments

Readers shared their thoughts on this article.

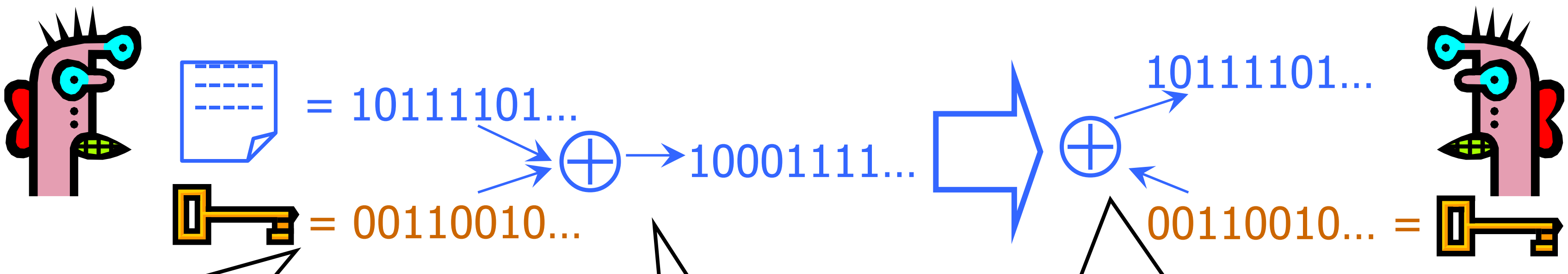
Read All Comments (127) »

› Uptime | IT and business computing insights

Crypto shocker: four of every 1,000 public keys provide no security (updated)

By Dan Goodin | Published 7 days ago

One-Time Pad (Vernam Cipher)



Key is a random bit sequence as long as the plaintext

Encrypt by bitwise XOR of plaintext and key:
ciphertext = plaintext ⊕ key

Decrypt by bitwise XOR of ciphertext and key:
ciphertext ⊕ key =
(plaintext ⊕ key) ⊕ key =
plaintext ⊕ (key ⊕ key) =
plaintext

Cipher achieves perfect secrecy if and only if there are as many possible keys as possible plaintexts, and every key is equally likely (Claude Shannon, 1949)

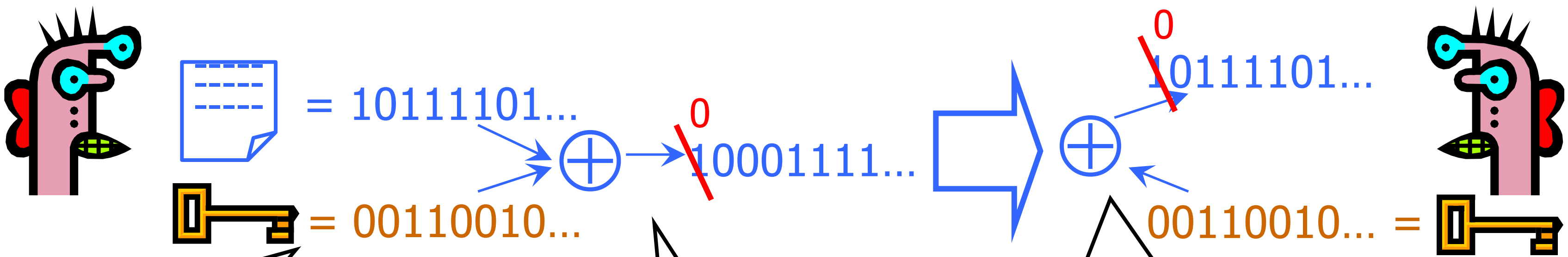
Advantages of One-Time Pad

- Easy to compute
  - Encryption and decryption are the same operation
  - Bitwise XOR is very cheap to compute
- As secure as theoretically possible
  - Given a ciphertext, all plaintexts are equally likely, regardless of attacker's computational resources
  - …<u>if and only if</u> the key sequence is truly random
    - True randomness is expensive to obtain in large quantities
  - …<u>if and only if</u> each key is as long as the plaintext
    - But how do the sender and the receiver communicate the key to each other? Where do they store the key?

Problems with One-Time Pad

- Key must be as long as the plaintext
  - Impractical in most realistic scenarios
  - Still used for diplomatic and intelligence traffic
- Does not guarantee integrity
  - One-time pad only guarantees confidentiality
  - Attacker cannot recover plaintext, but can easily change it to something else
- Insecure if keys are reused
  - Attacker can obtain XOR of plaintexts

No Integrity

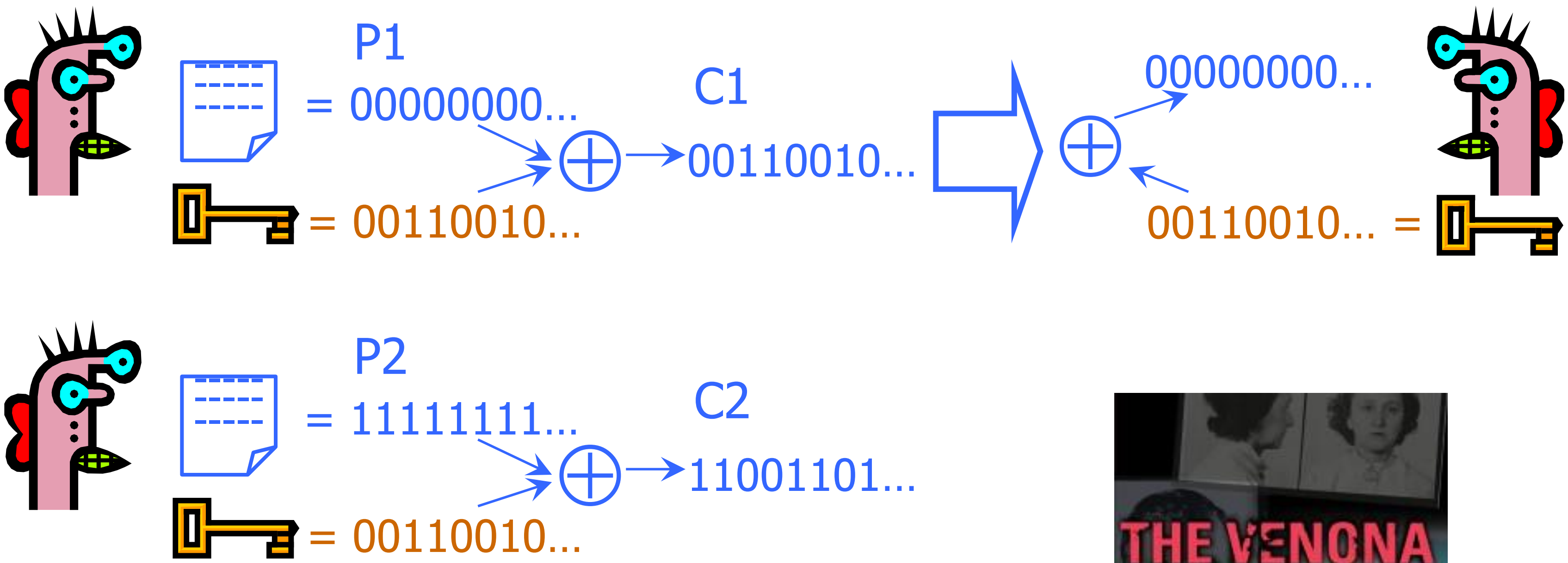


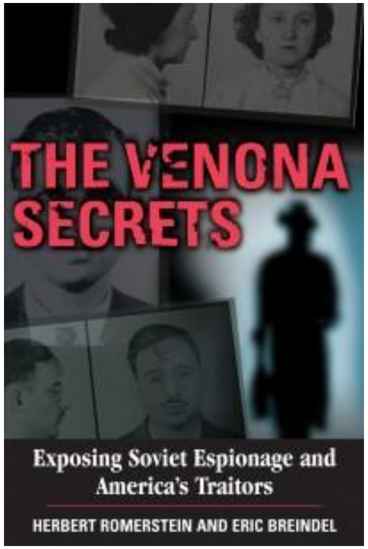
Key is a random bit sequence as long as the plaintext

Encrypt by bitwise XOR of plaintext and key:
ciphertext = plaintext ⊕ key

Decrypt by bitwise XOR of ciphertext and key:
ciphertext ⊕ key =
(plaintext ⊕ key) ⊕ key =
plaintext ⊕ (key ⊕ key) =
plaintext

Dangers of Reuse





Learn relationship between plaintexts

$C1 \oplus C2 = (P1 \oplus K) \oplus (P2 \oplus K) =$

$(P1 \oplus P2) \oplus (K \oplus K) = P1 \oplus P2$

Reducing Key Size

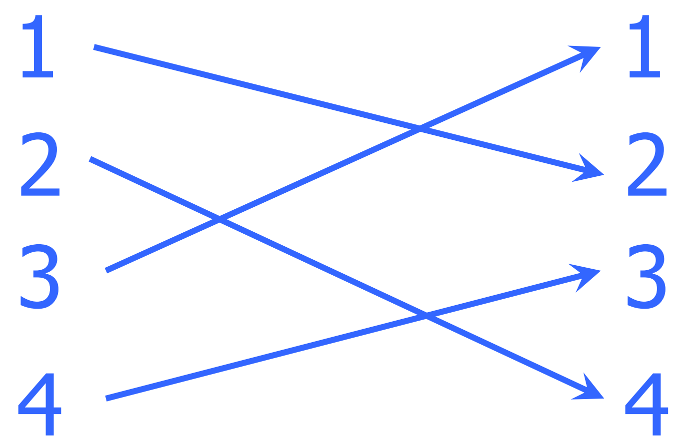
- What to do when it is infeasible to pre-share huge random keys?
- Use special cryptographic primitives:

  block ciphers, stream ciphers
  - Single key can be re-used (with some restrictions)
  - Not as theoretically secure as one-time pad

Block Ciphers

- Operates on a single chunk ("block") of plaintext
  - For example, 64 bits for DES, 128 bits for AES
  - Same key is reused for each block (can use short keys)
- Result should look like a random permutation
- Not impossible to break, just very expensive
  - If there is no more efficient algorithm (unproven assumption!), can only break the cipher by brute-force, try-every-possible-key search
  - Time and cost of breaking the cipher exceed the value and/or useful lifetime of protected information

Permutation



CODE becomes DCEO

- For N-bit input, N! possible permutations
- Idea: split plaintext into blocks, for each block use secret key to pick a permutation, rinse and repeat
  - Without the key, permutation should "look random"

A Bit of Block Cipher History

- Playfair and variants (from 1854 until WWII)
- Feistel structure

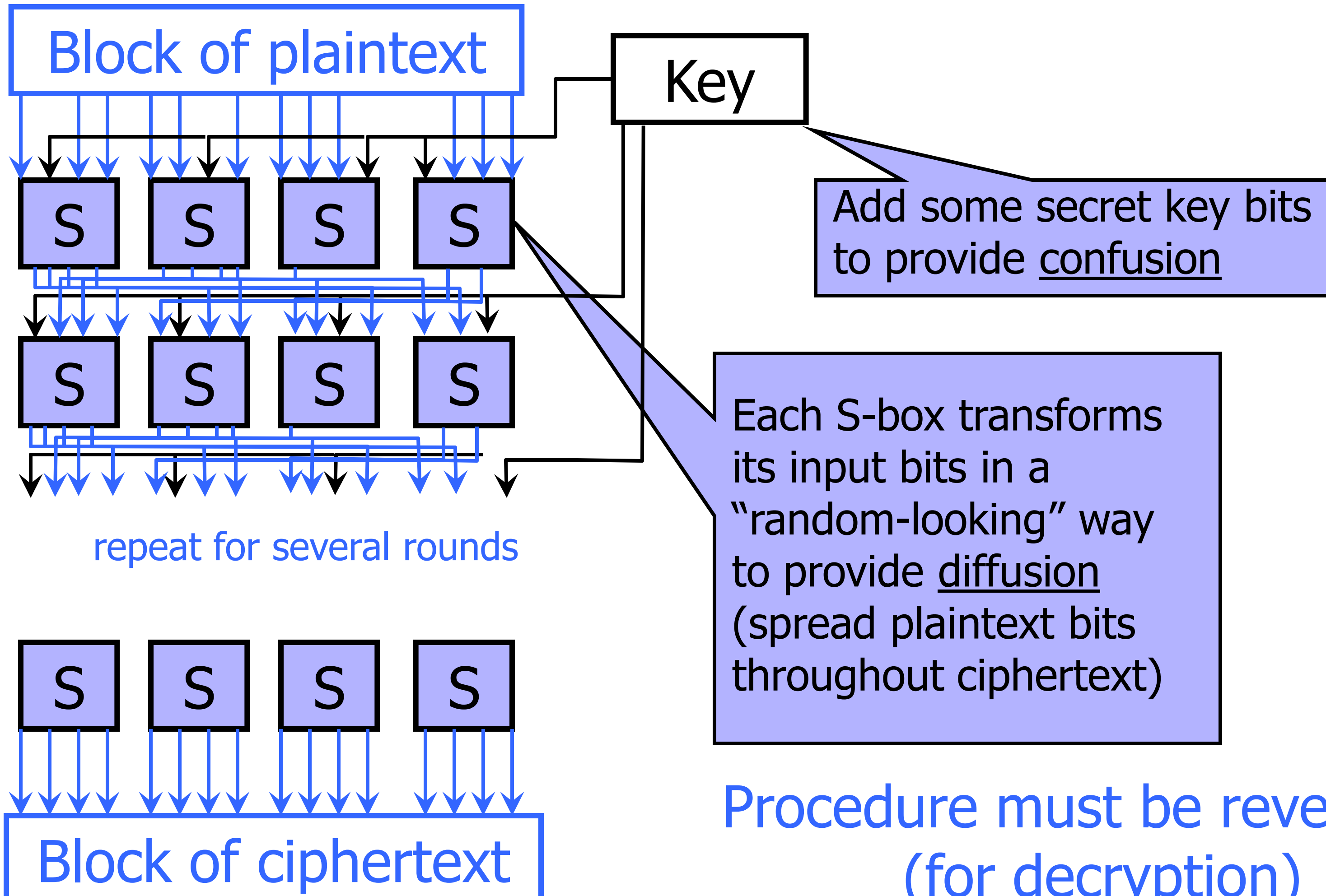
  Textbook

  - "Ladder" structure: split input in half, put one half through the round and XOR with the other half
  - After 3 random rounds, ciphertext indistinguishable from a random permutation
- DES: Data Encryption Standard

  Textbook

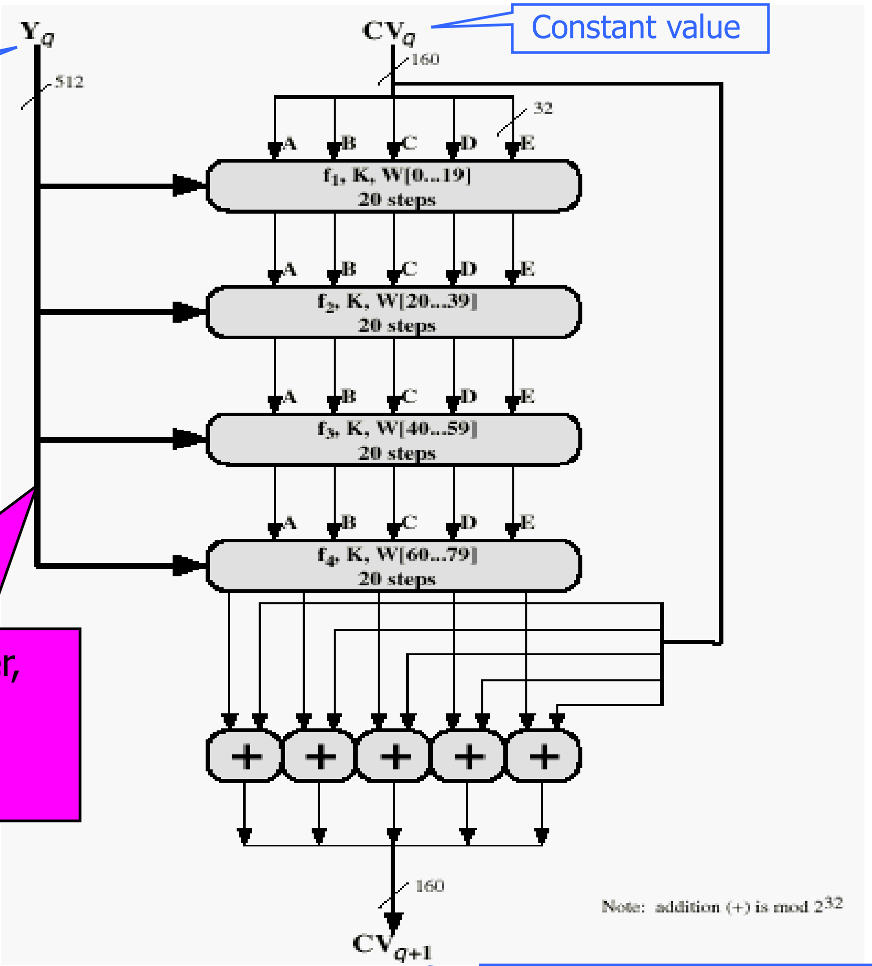
  - Invented by IBM, issued as federal standard in 1977
  - 64-bit blocks, 56-bit key + 8 bits for parity
  - Very widely used (usually as 3DES) until recently
    - 3DES: DES + inverse DES + DES (with 2 or 3 different keys)

DES Operation (Simplified)



Procedure must be reversible
(for decryption)

Remember SHA-1?



Current **message block**

Constant value

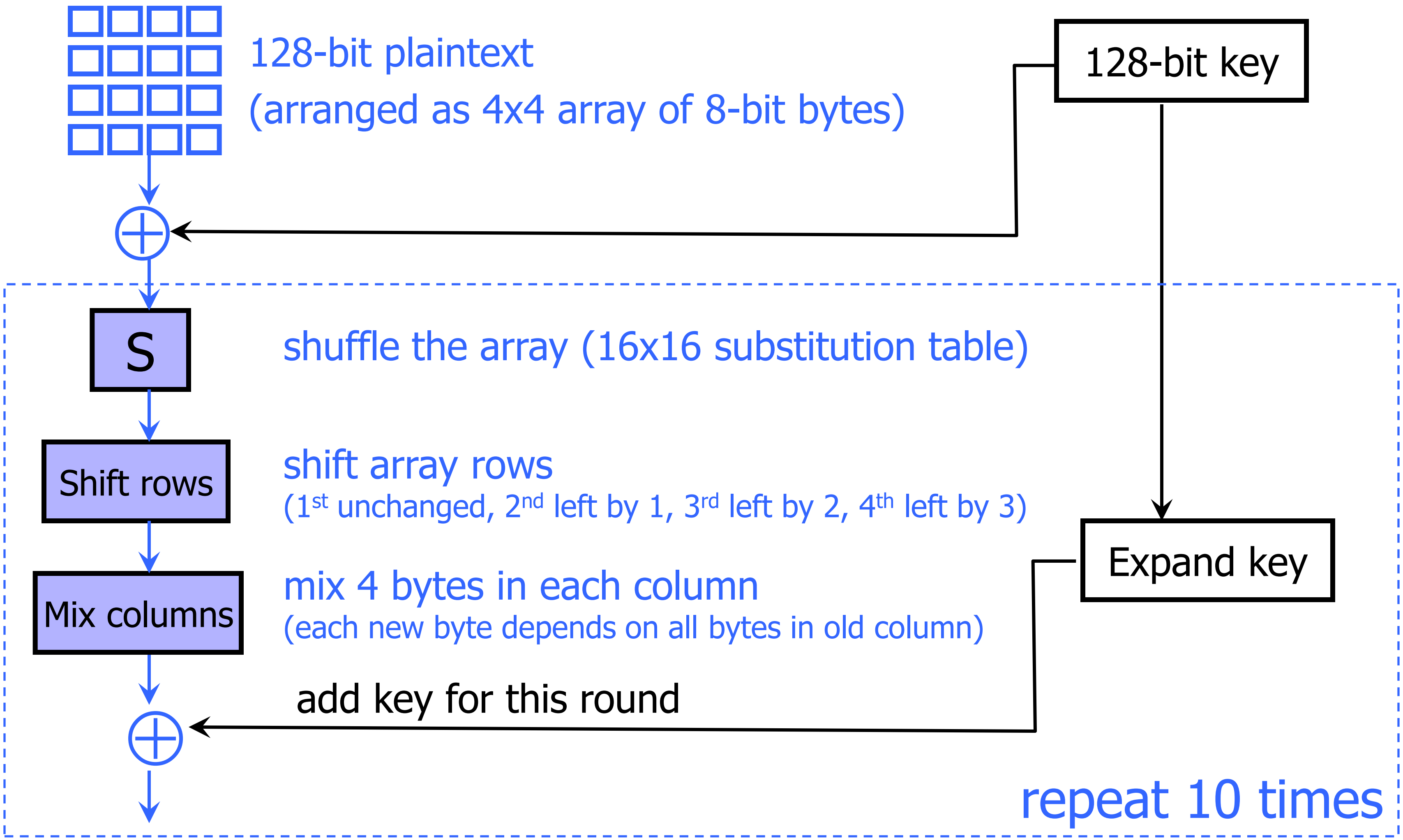
Very similar to a block cipher, with message itself used as the key for each round

Buffer contains final hash value

Advanced Encryption Standard (AES)

- US federal standard as of 2001
- Based on the Rijndael algorithm
- 128-bit blocks, keys can be 128, 192 or 256 bits
- Unlike DES, does not use Feistel structure
  - The entire block is processed during each round
- Design uses some clever math
  - See section 8.5 of the textbook for a concise summary

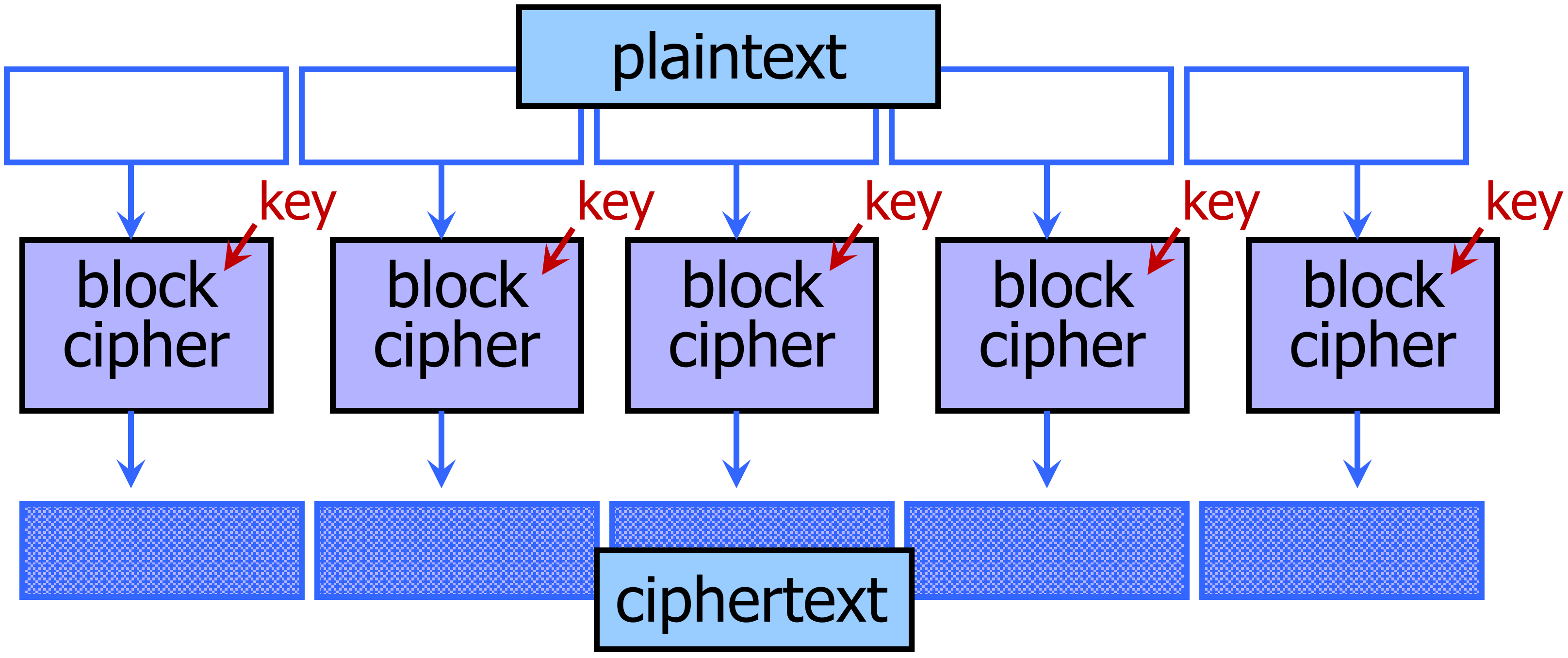
Basic Structure of Rijndael



Encrypting a Large Message

- So, we've got a good block cipher, but our plaintext is larger than 128-bit block size
- Electronic Code Book (ECB) mode
  - Split plaintext into blocks, encrypt each one separately using the block cipher
- Cipher Block Chaining (CBC) mode
  - Split plaintext into blocks, XOR each block with the result of encrypting previous blocks
- Also various counter modes, feedback modes, etc.

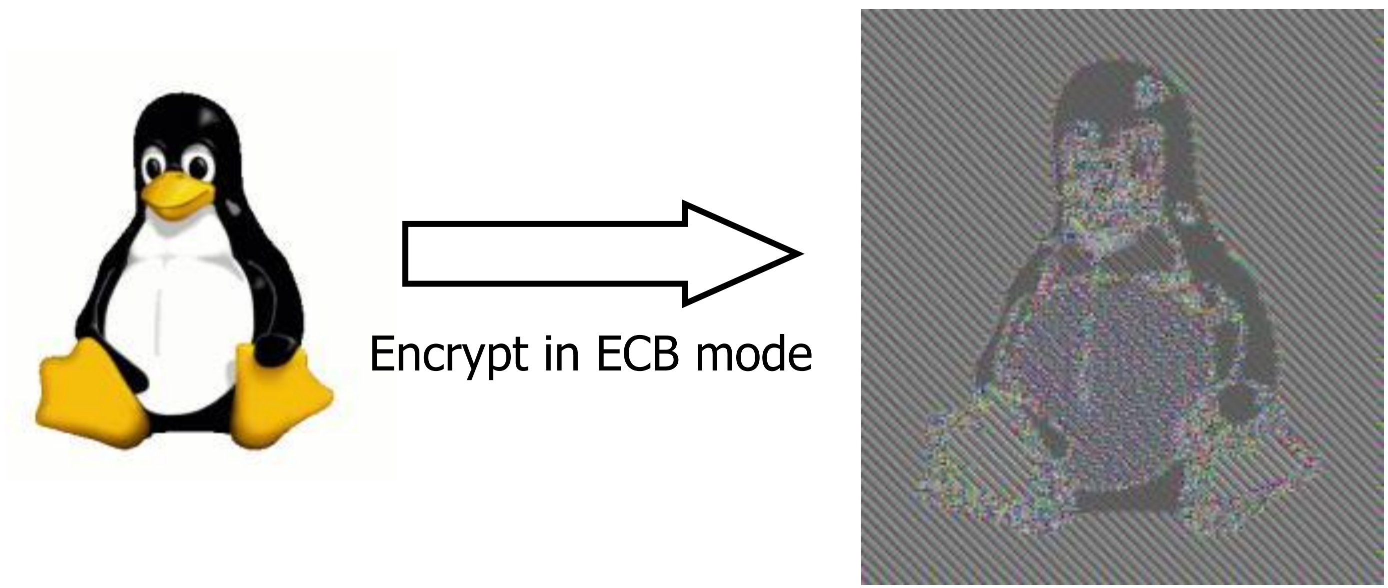
ECB Mode



- Identical blocks of plaintext produce identical blocks of ciphertext
- No integrity checks: can mix and match blocks

Information Leakage in ECB Mode

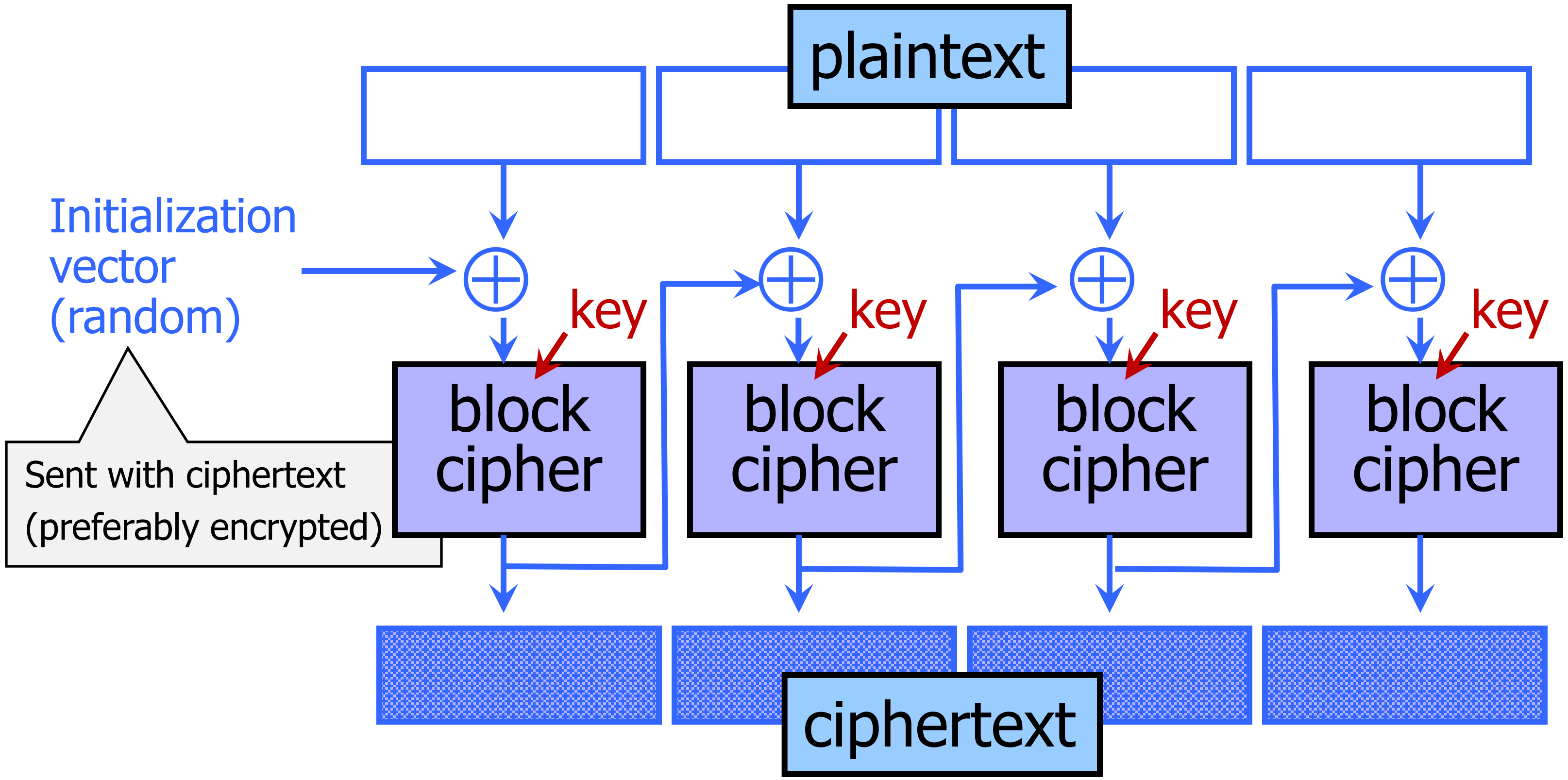
[Wikipedia]



Adobe Passwords Stolen (2013)

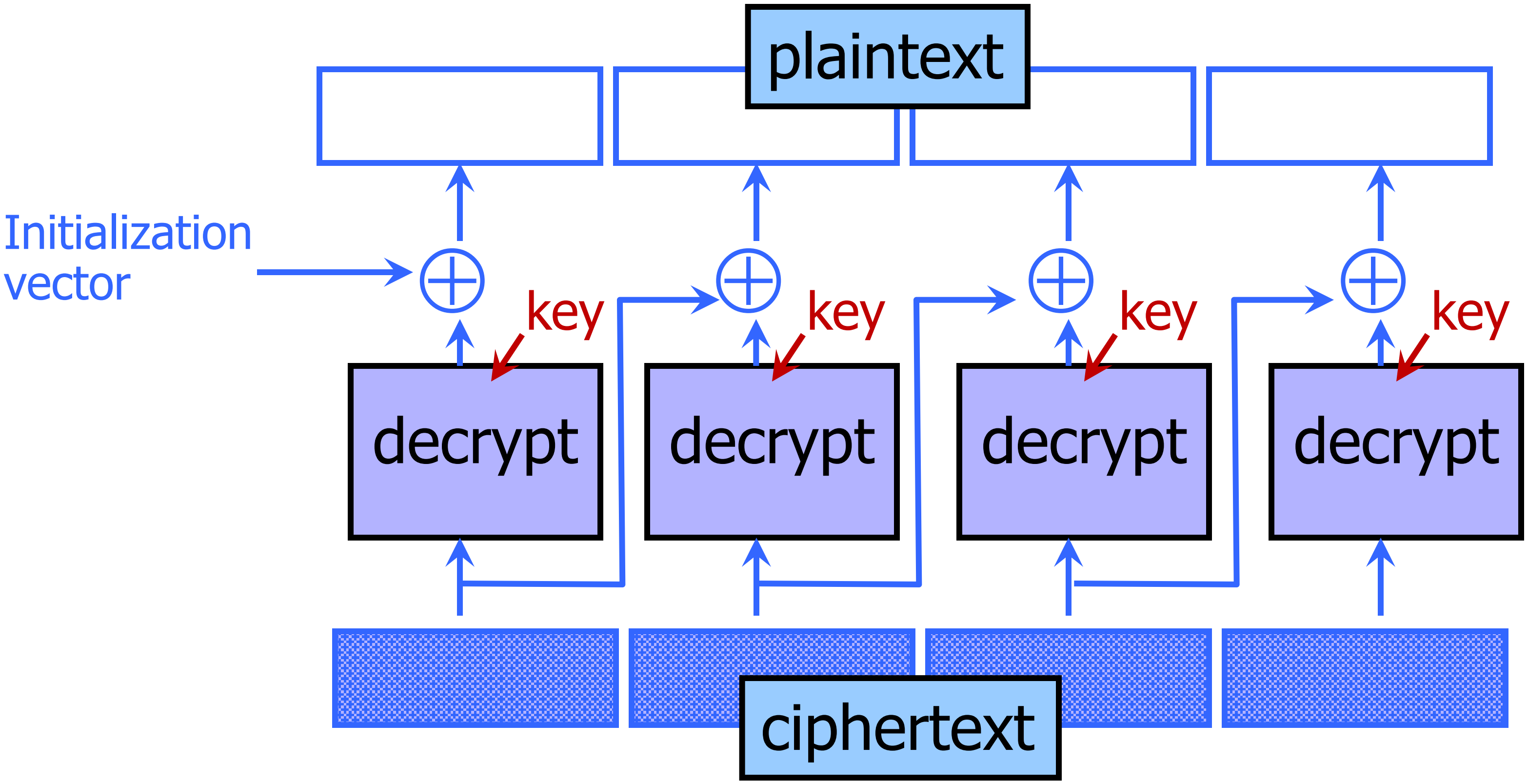
- 153 million account passwords
  - 56 million of them unique
- Encrypted using 3DES in ECB mode rather than hashed

CBC Mode: Encryption



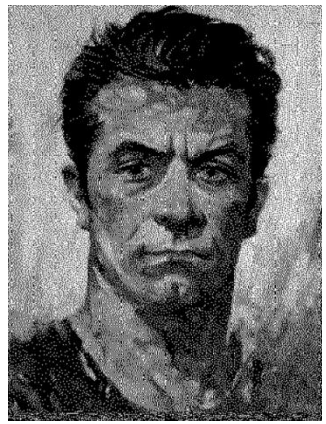
- Identical blocks of plaintext encrypted differently
- Last cipherblock depends on entire plaintext
  - Still does not guarantee integrity

CBC Mode: Decryption



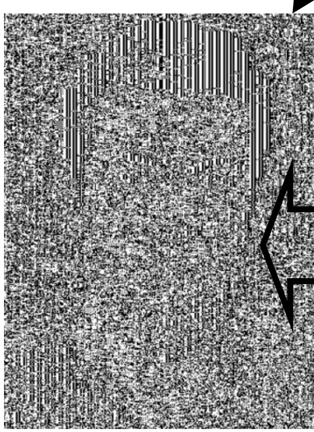
ECB vs. CBC

[Picture due to Bart Preneel]

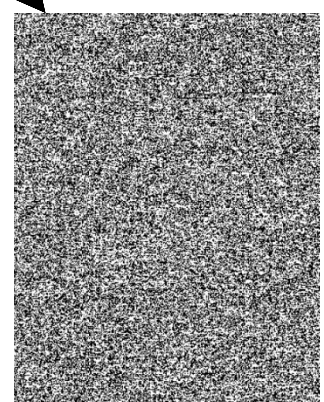


AES in ECB mode

AES in CBC mode



Similar plaintext
blocks produce
similar ciphertext
blocks (not good!)

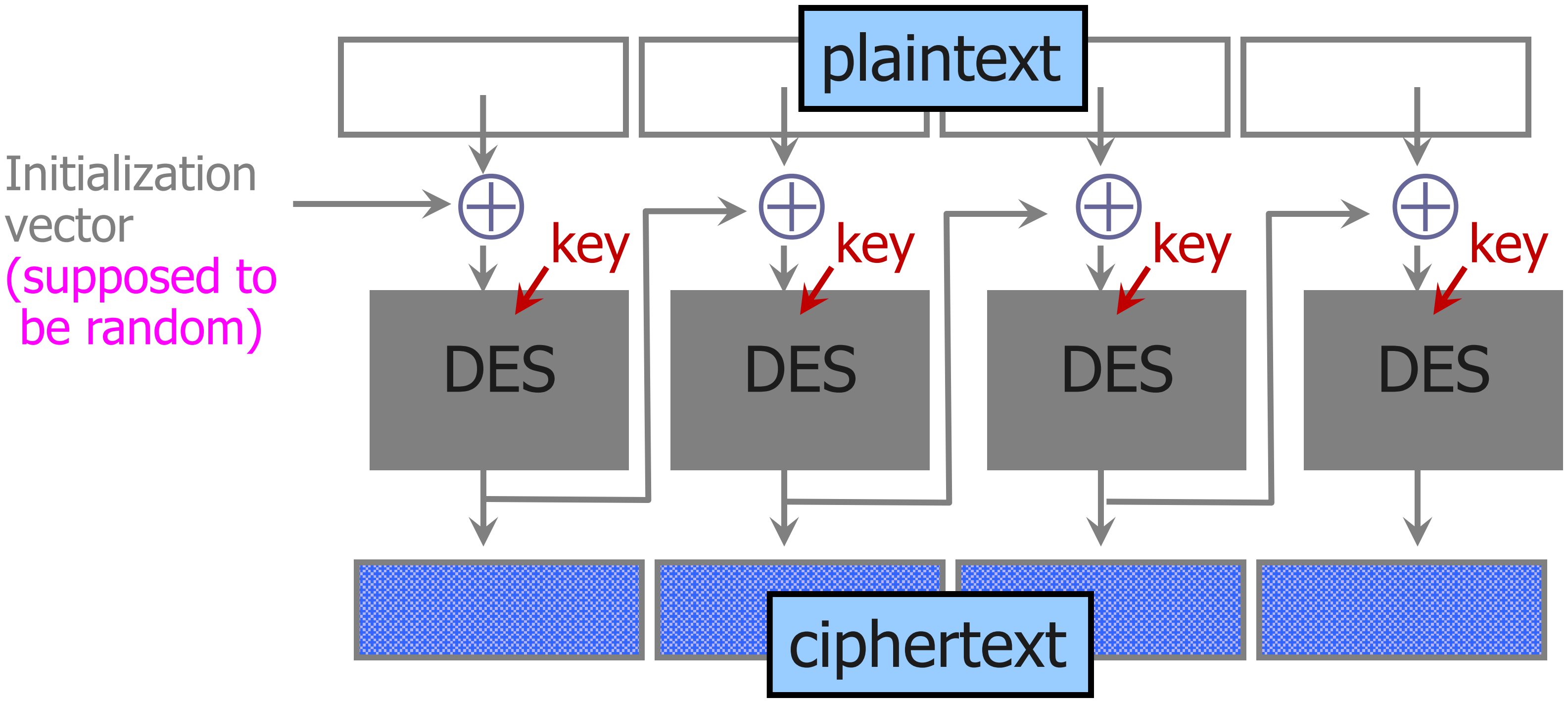


Choosing the Initialization Vector

- Key used only once
  - No IV needed (can use IV=0)
- Key used multiple times
  - Best: fresh, random IV for every message
  - Can also use unique IV (eg, counter), but then the first step in CBC mode must be IV′ $\leftarrow$ E(k, IV)
    - Example: Windows BitLocker
    - May not need to transmit IV with the ciphertext
- Multi-use key, unique messages
  - Synthetic IV: IV $\leftarrow$ F(k′, message)
    - F is a cryptographically secure keyed pseudorandom function

CBC and Electronic Voting

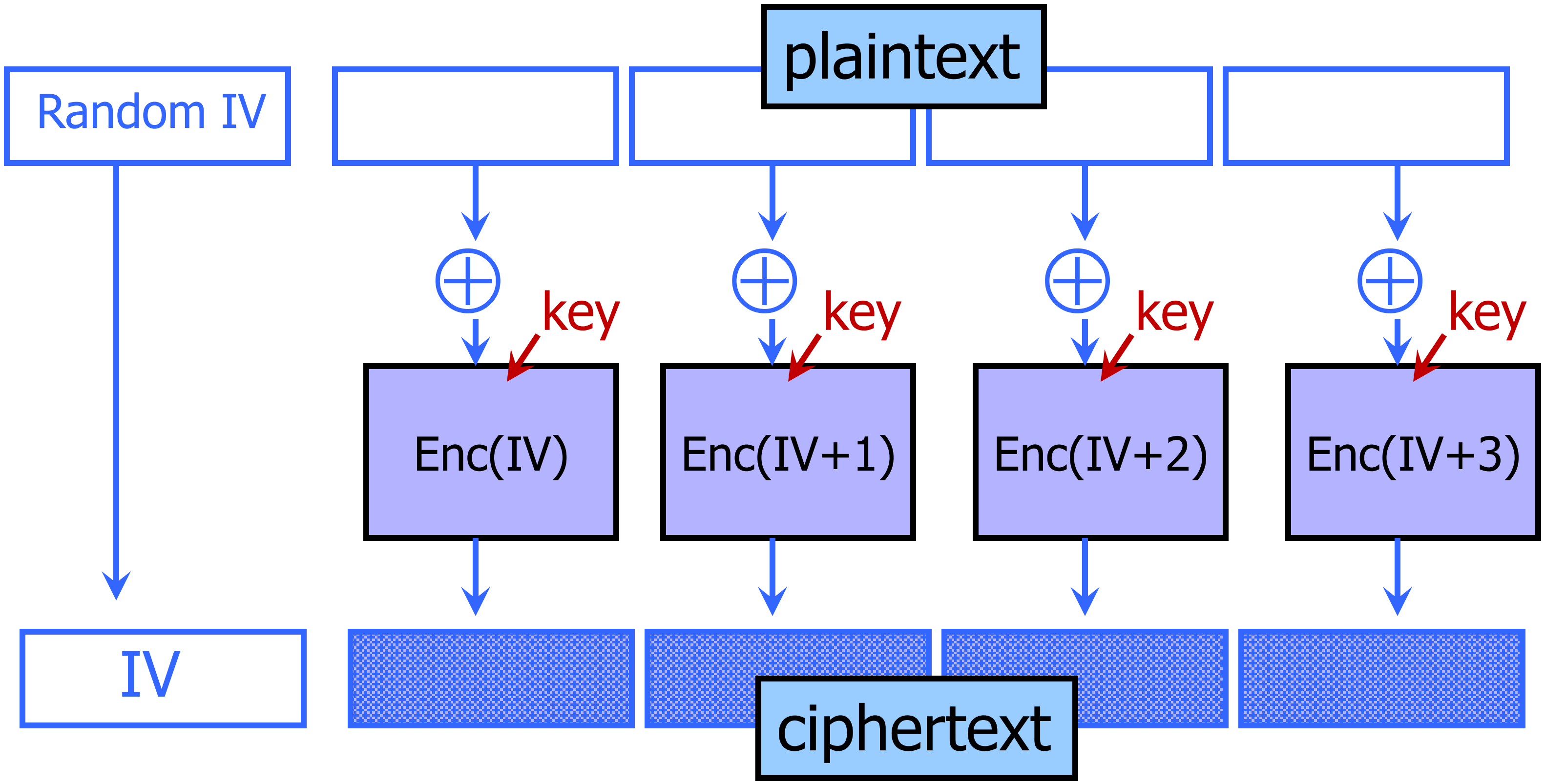
[Kohno, Stubblefield, Rubin, Wallach]



Found in the source code for Diebold voting machines:

```
DesCBCEncrypt((des_c_block*)tmp, (des_c_block*)record.m_Data,
        totalSize, DESKEY, NULL, DES_ENCRYPT)
```

CTR (Counter Mode)



- Still does not guarantee integrity
- Fragile if counter repeats

When Is a Cipher "Secure"?

- Hard to recover plaintext from ciphertext?
  - What if attacker learns only some bits of the plaintext? Some function of the bits? Some partial information about the plaintext?
- Fixed mapping from plaintexts to ciphertexts?
  - What if attacker sees two identical ciphertexts and infers that the corresponding plaintexts are identical?
  - What if attacker guesses the plaintext – can he verify his guess?
  - Implication: encryption must be randomized or stateful

How Can a Cipher Be Attacked?

- Attackers knows ciphertext and encryption algthm
  - What else does the attacker know? Depends on the application in which the cipher is used!

- Known-plaintext attack (stronger)
  - Knows some plaintext-ciphertext pairs
- Chosen-plaintext attack (even stronger)
  - Can obtain ciphertext for any plaintext of his choice
- Chosen-ciphertext attack (very strong)
  - Can decrypt any ciphertext except the target
  - Sometimes very realistic

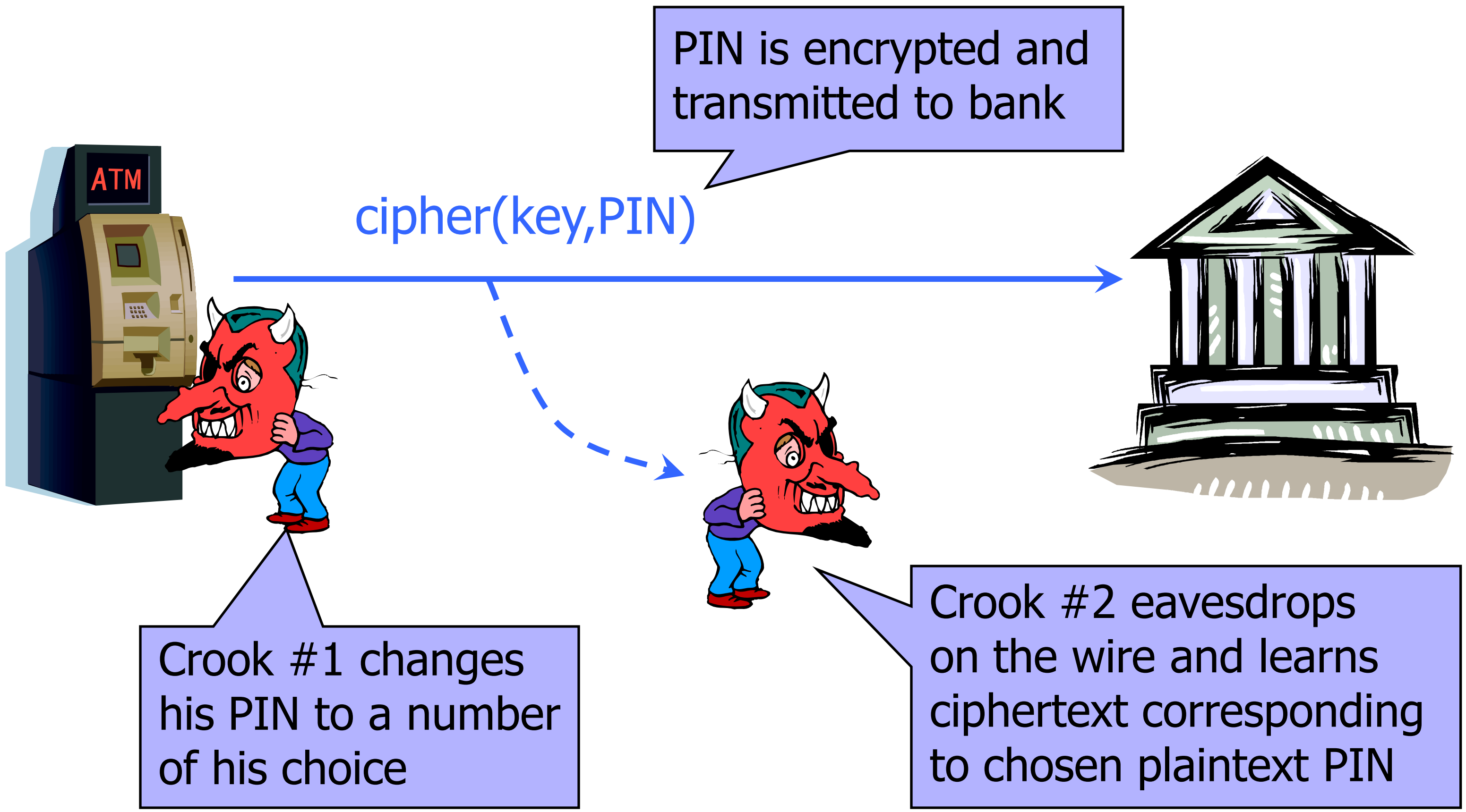
Known-Plaintext Attack

[From "The Art of Intrusion"]

Extracting password from an encrypted PKZIP file …

- "… I opened the ZIP file and found a `logo.tif' file, so I went to their main Web site and looked at all the files named `logo.tif.' I downloaded them and zipped them all up and found one that matched the same checksum as the one in the protected ZIP file"
- With known plaintext, PkCrack took 5 minutes to extract the key
  - Biham-Kocher attack on PKZIP stream cipher

Chosen-Plaintext Attack



… repeat for any PIN value

Very Informal Intuition

Minimum security requirement for a modern encryption scheme

- Security against chosen-plaintext attack
  - Ciphertext leaks no information about the plaintext
  - Even if the attacker correctly guesses the plaintext, he cannot verify his guess
  - Every ciphertext is unique, encrypting same message twice produces completely different ciphertexts
- Security against chosen-ciphertext attack
  - Integrity protection – it is not possible to change the plaintext by modifying the ciphertext

The Chosen-Plaintext Game

- Attacker does not know the key
- He chooses as many plaintexts as he wants, and receives the corresponding ciphertexts
- When ready, he picks two plaintexts M_0 and M_1
  - He is even allowed to pick plaintexts for which he previously learned ciphertexts!
- He receives either a ciphertext of M_0, or a ciphertext of M_1
- He wins if he guesses correctly which one it is

Meaning of “Leaks No Information”

- Idea: given a ciphertext, attacker should not be able to learn even a single bit of useful information about the plaintext
- Let $Enc(M_0,M_1,b)$ be a “magic box” that returns encrypted M_b

  0 or 1

  - Given two plaintexts, the box always returns the ciphertext of the left plaintext or right plaintext
  - Attacker can use this box to obtain the ciphertext of any plaintext M by submitting $M_0=M_1=M$, or he can try to learn even more by submitting $M_0 \neq M_1$
- Attacker’s goal is to learn just this one bit b

Chosen-Plaintext Security

- Consider two experiments (A is the attacker)

  Experiment 0

  A interacts with Enc(-,-,0) and outputs his guess of bit b

  Experiment 1

  A interacts with Enc(-,-,1) and outputs his guess of bit b

  - Identical except for the value of the secret bit
  - b is attacker's guess of the secret bit
- Attacker's advantage is defined as

  | Prob(A outputs 1 in Exp0) - Prob(A outputs 1 in Exp1)) |
- Encryption scheme is chosen-plaintext secure if this advantage is negligible for any efficient A

Simple Example

- Any deterministic, stateless symmetric encryption scheme is insecure
  - Attacker can easily distinguish encryptions of different plaintexts from encryptions of identical plaintexts
  - This includes ECB mode of common block ciphers!

Attacker A interacts with Enc(-,-,b)

Let X,Y be any two different plaintexts

$C_1 \leftarrow$ Enc(X,X,b); $C_2 \leftarrow$ Enc(X,Y,b);

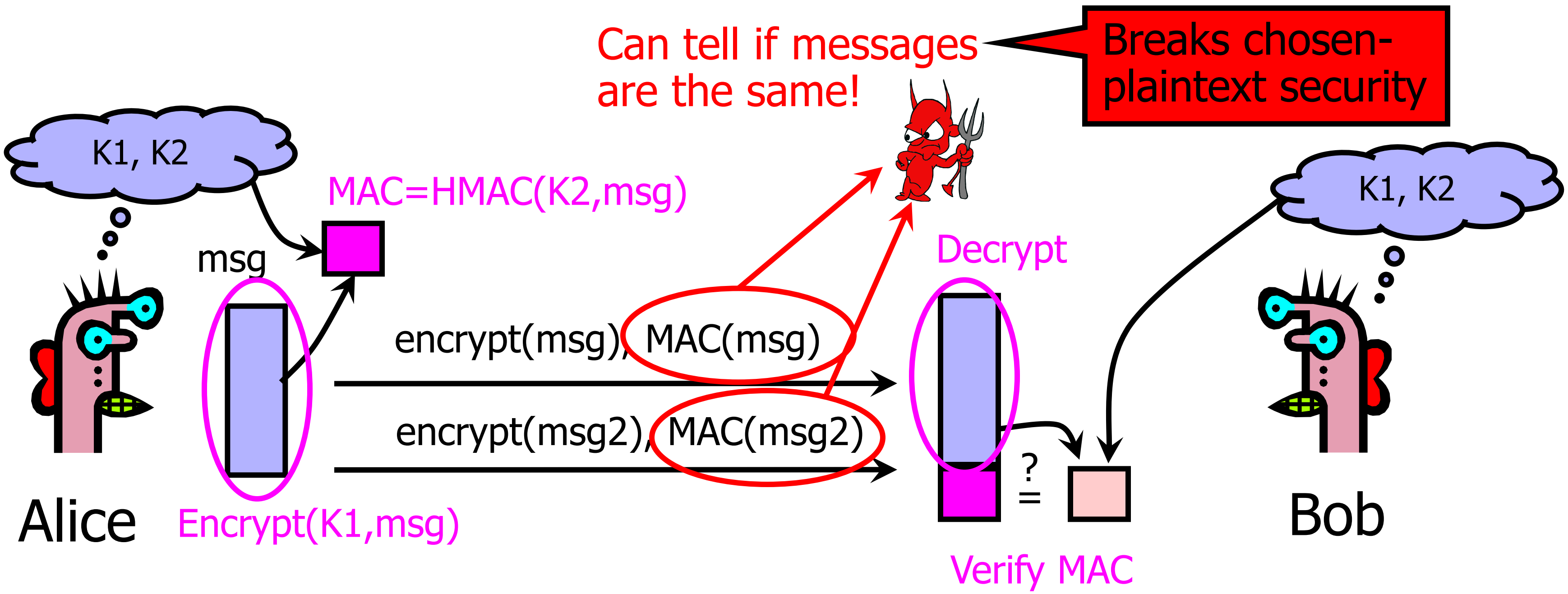
If $C_1=C_2$ then b=0 else b=1

- The advantage of this attacker A is 1

Prob(A outputs 1 if b=0)=0 Prob(A outputs 1 if b=1)=1

Encrypt + MAC

Goal: confidentiality + integrity + authentication



MAC is deterministic: messages are equal $\Rightarrow$ their MACs are equal

Solution: Encrypt, <u>then</u> MAC (or MAC, then encrypt)