CSE 484 / CSE M 584 (Winter 2013)

(Continue) Cryptography

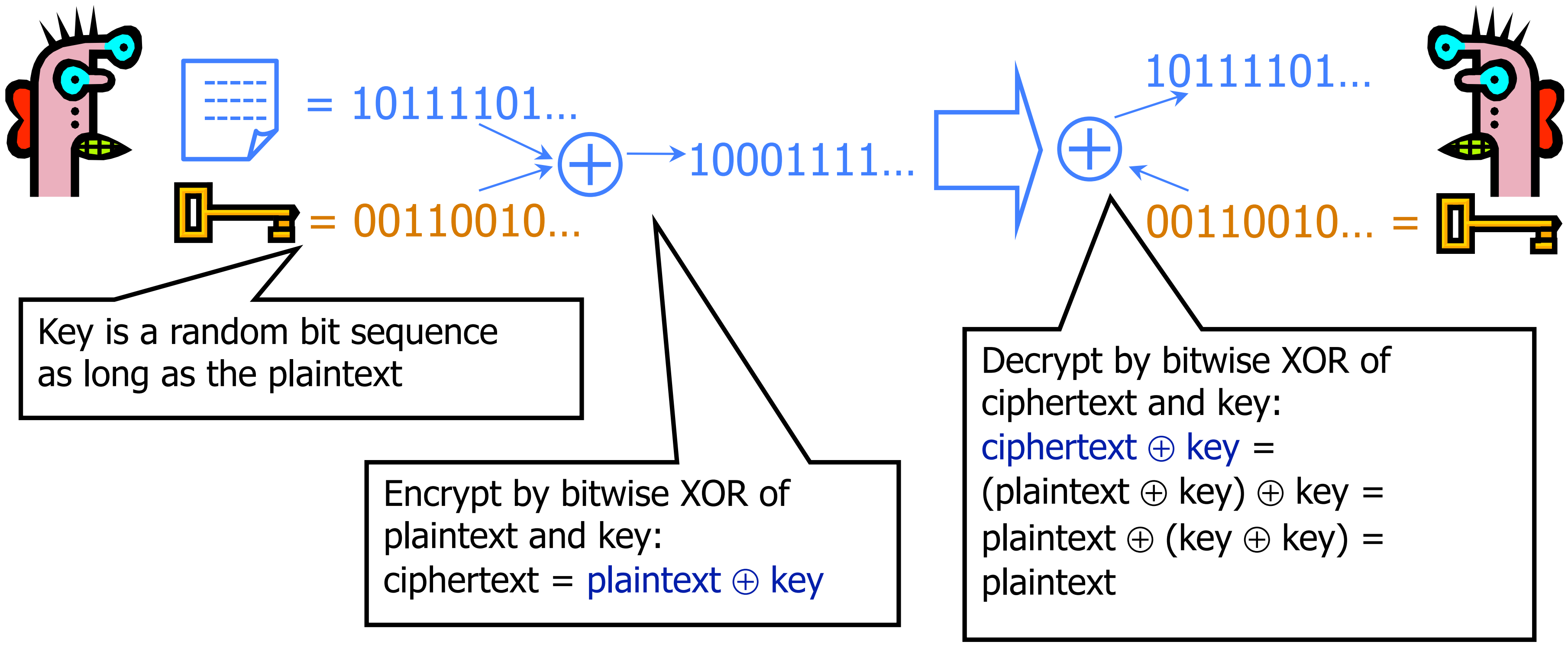
Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Goals for Today

- Cryptography

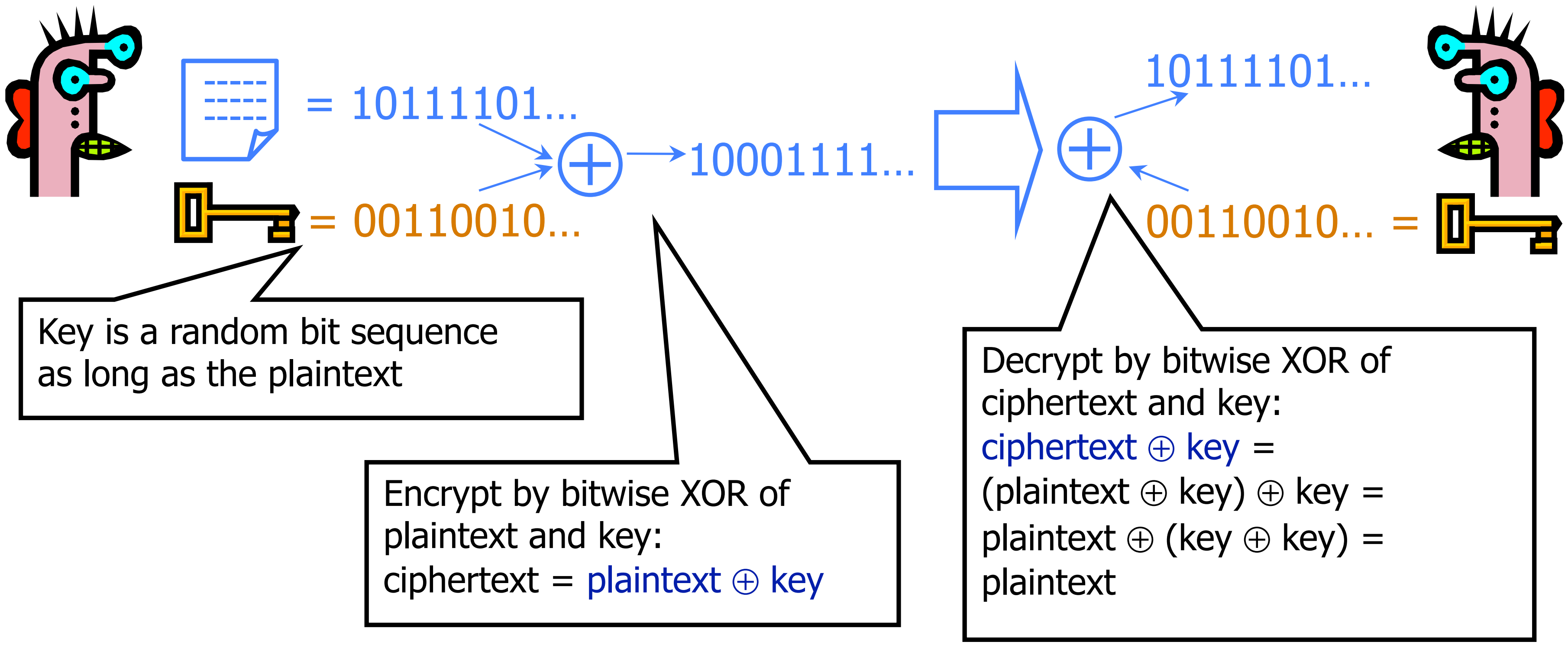
One-Time Pad


= 10111101…
= 00110010…
10001111…
10111101…
00110010… =
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

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
  - …as long as the key sequence is truly random
    - True randomness is expensive to obtain in large quantities
  - …as long as each key is same length as plaintext
    - But how does the sender communicate the key to receiver?

Disadvantages

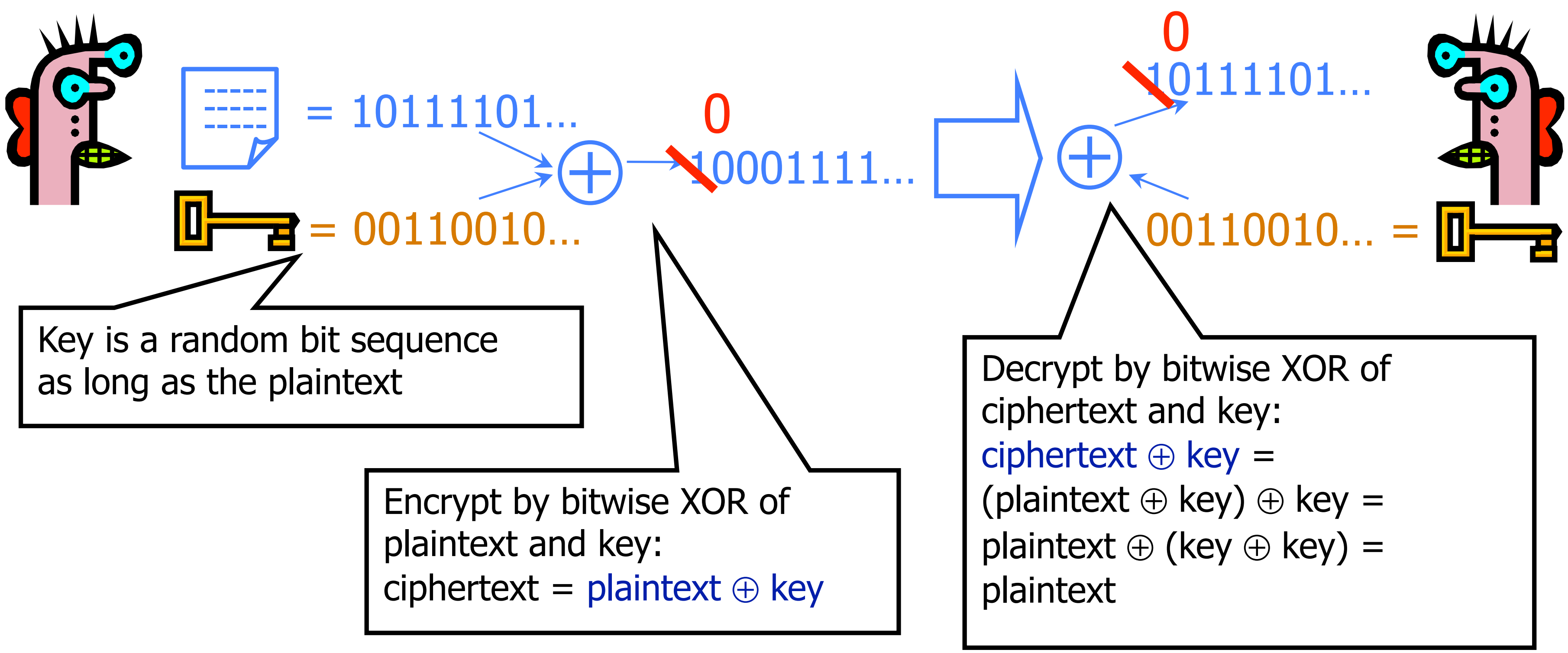


Disadvantage #1: Keys as long as messages.

Impractical in most scenarios

Still used by intelligence communities

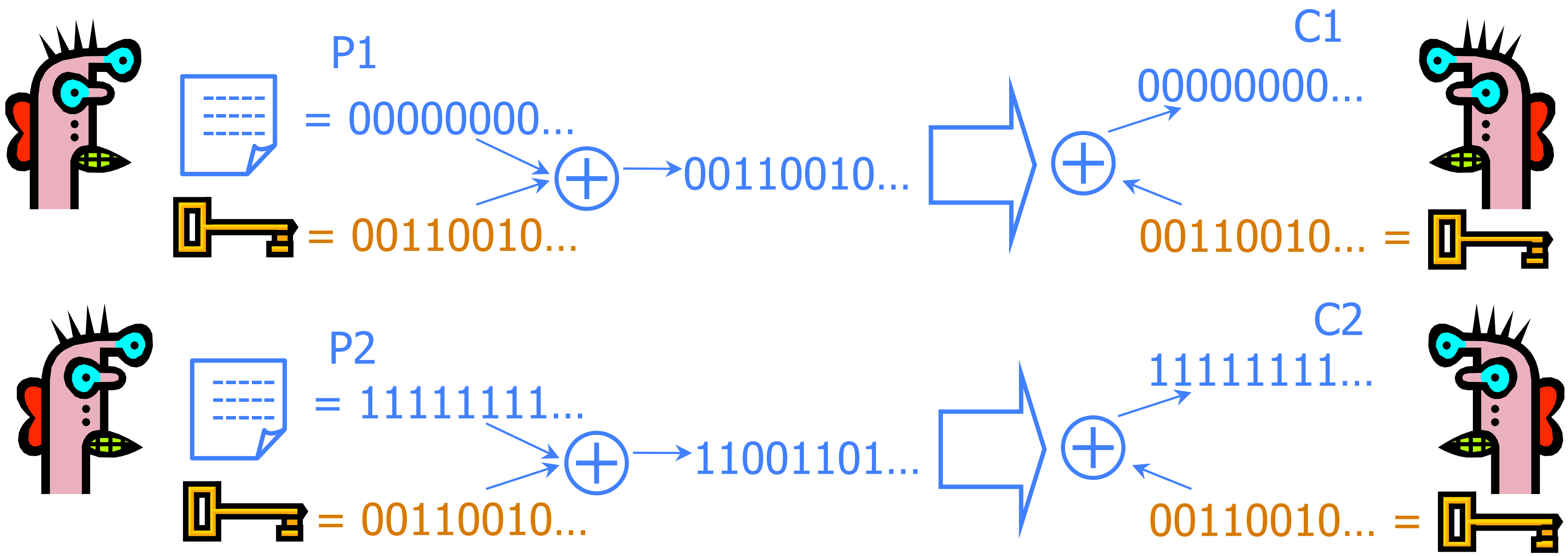
Disadvantages



Disadvantage #2: No integrity protection

Disadvantages

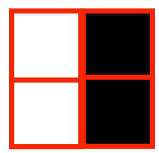
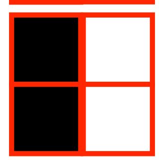
Disadvantage #3: Keys cannot be reused



Learn relationship between plaintexts:

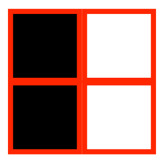
$C1 \oplus C2 = (P1 \oplus K) \oplus (P2 \oplus K) = (P1 \oplus P2) \oplus (K \oplus K) = P1 \oplus P2$

Visual Cryptography

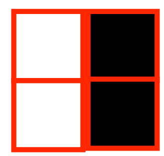
- Generate a random bitmap
- Encode 0 as: 
- Encode 1 as: 

Visual Cryptography

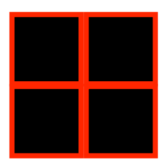
- Take a black and white bitmap image
- For a white pixel, send the same as the mask



or

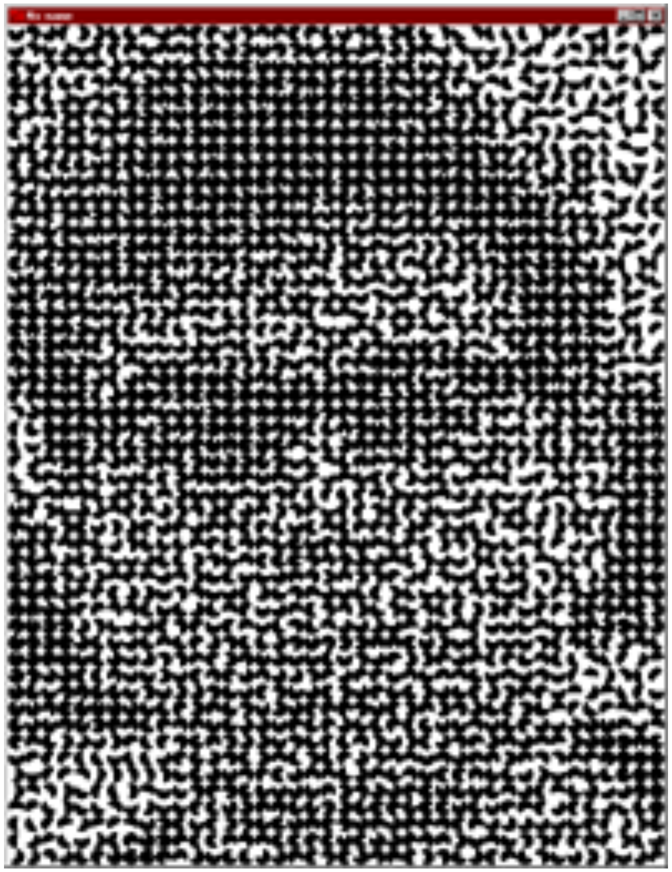


- For a black pixel, send the opposite of the mask



See also http://www.cs.washington.edu/homes/yoshi/cs4hs/cse-vc.html

Visual Cryptography



- http://www.cl.cam.ac.uk/~fms27/vck/face.gif

See also http://www.cs.washington.edu/homes/yoshi/cs4hs/cse-vc.html

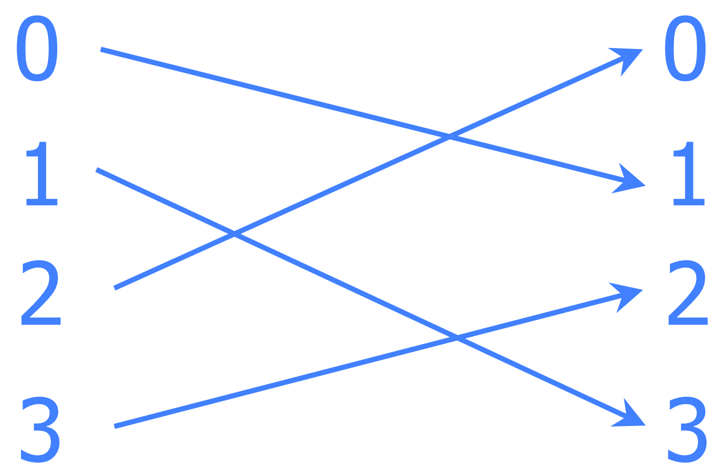
Reducing Keysize

- What do we do when we can’t pre-share huge keys?
  - When OTP is unrealistic

- We use special cryptographic primitives
  - Single key can be reused (with some restrictions)
  - But no longer provable secure (in the sense of the OTP)

- Examples: Block ciphers, stream ciphers

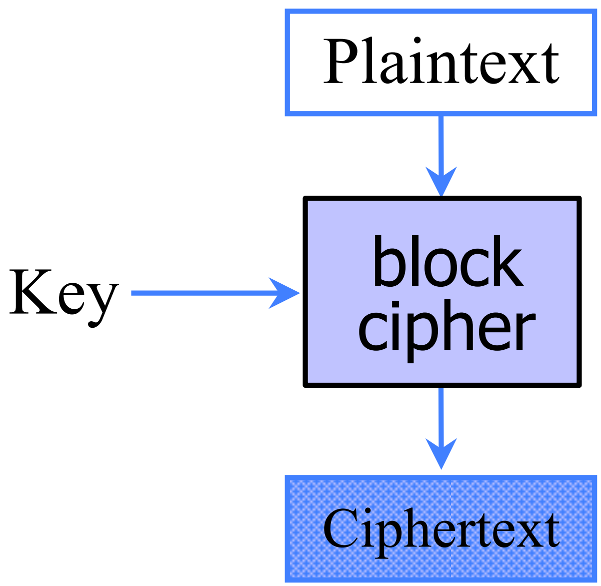
Background: Permutation



- For N-bit input, $2^N!$ possible permutations
- Idea for how to use a keyed permutation: split plaintext into blocks; for each block use secret key to pick a permutation
  - Without the key, permutation should "look random"

Block Ciphers

- Operates on a single chunk ("block") of plaintext
  - For example, 64 bits for DES, 128 bits for AES
  - Each key defines a different permutation
  - Same key is reused for each block (can use short keys)

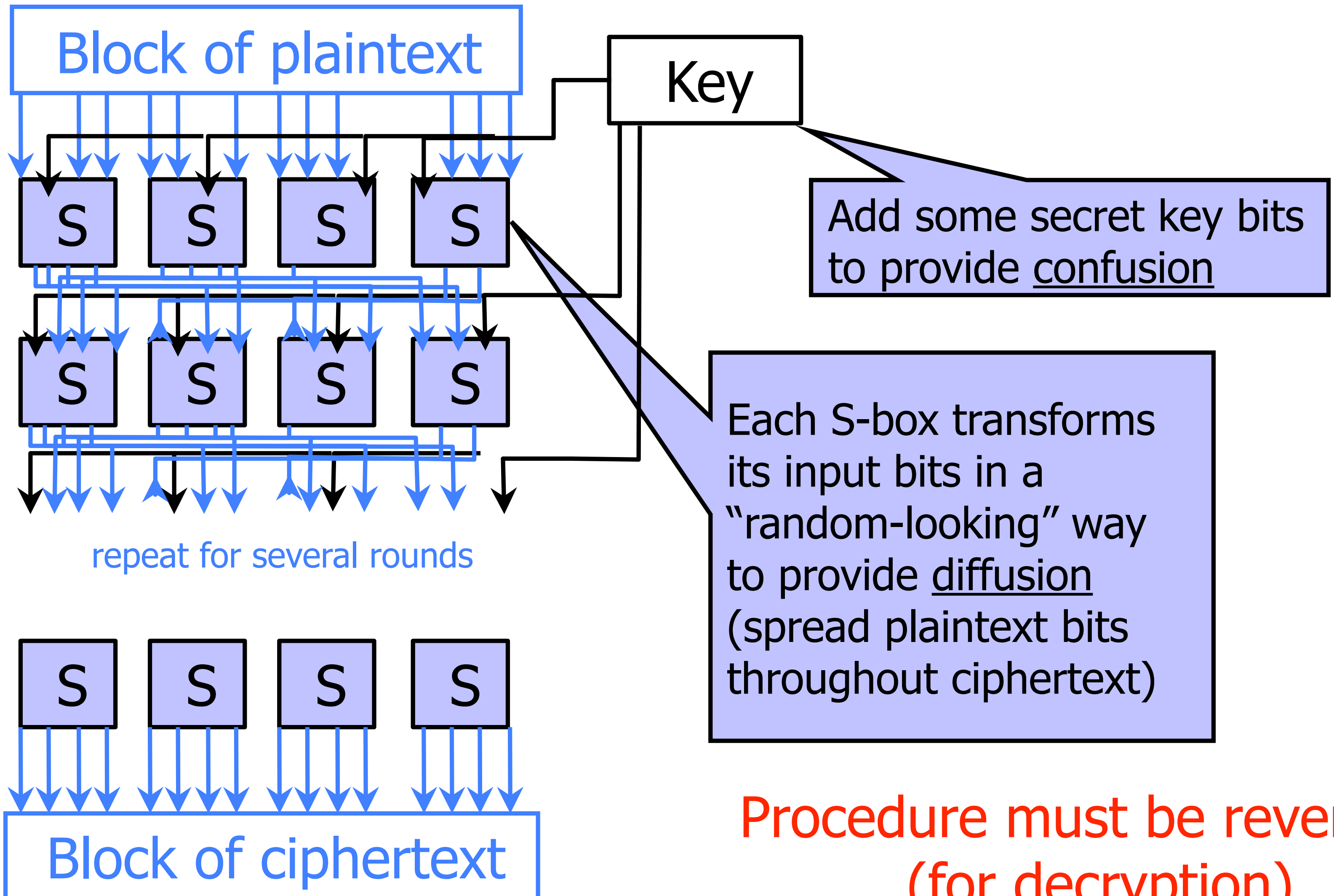


Block Cipher Security

- Result should look like a random permutation on the inputs
  - Recall: not just shuffling bits. N-bit block cipher permutes over 2^N inputs.

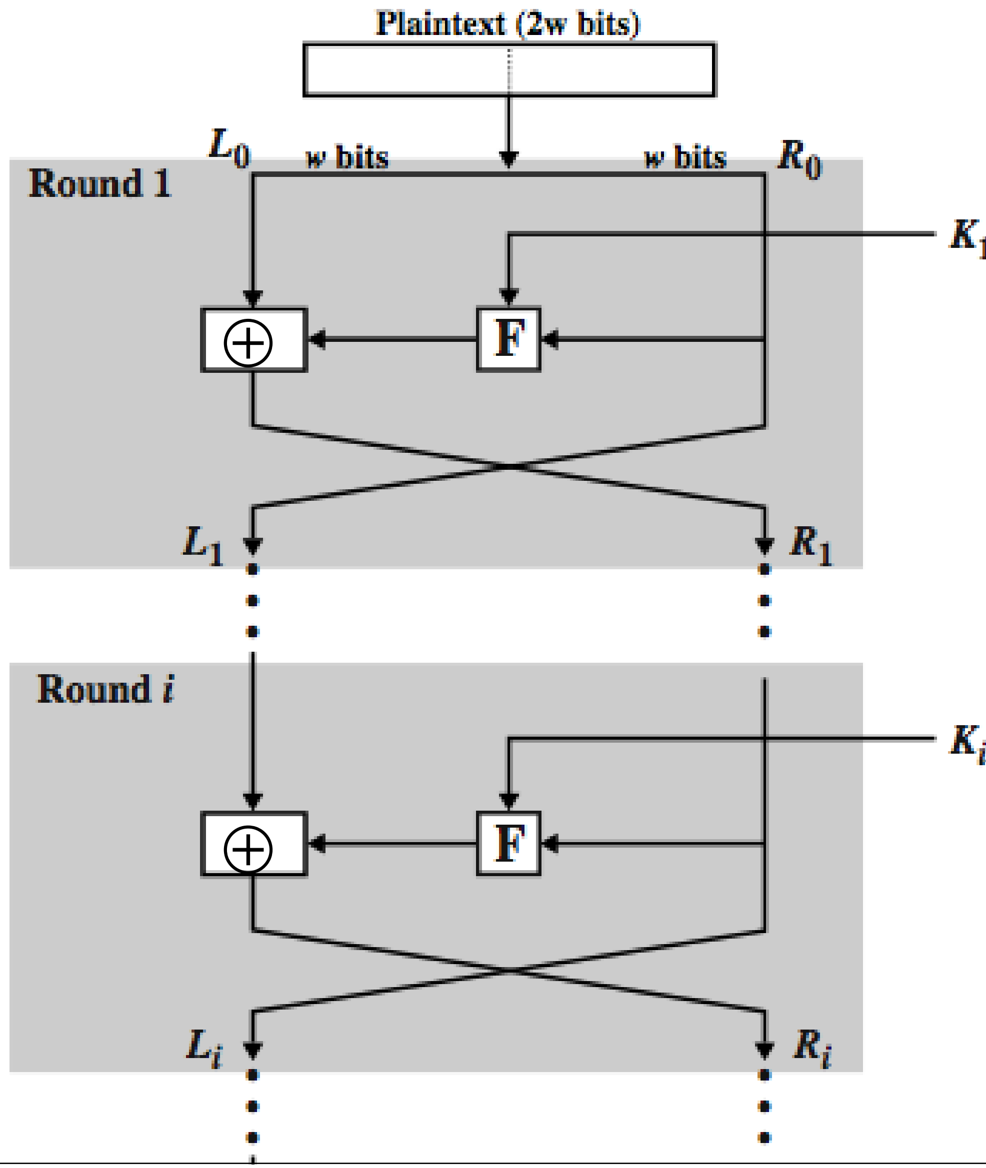
- Only computational guarantee of secrecy
  - Not impossible to break, just very expensive
    - If there is no efficient algorithm (unproven assumption!), then can only break by brute-force, try-every-possible-key search
  - Time and cost of breaking the cipher exceed the value and/or useful lifetime of protected information

Block Cipher Operation (Simplified)



Procedure must be reversible
(for decryption)

Feistel Structure (Stallings Fig 2.2)



DES

- Feistel structure
  - "Ladder" structure: split input in half, put one half through the round and XOR with the other half
  - Theoretical support: After 3 random rounds, ciphertext indistinguishable from a random permutation if internal F function is a pseudorandom function (Luby & Rackoff)
- DES: Data Encryption Standard
  - Feistel structure
  - Invented by IBM, issued as federal standard in 1977
  - 64-bit blocks, 56-bit key + 8 bits for parity

DES and 56 bit keys (Stallings Tab 2.2)

- 56 bit keys are quite short

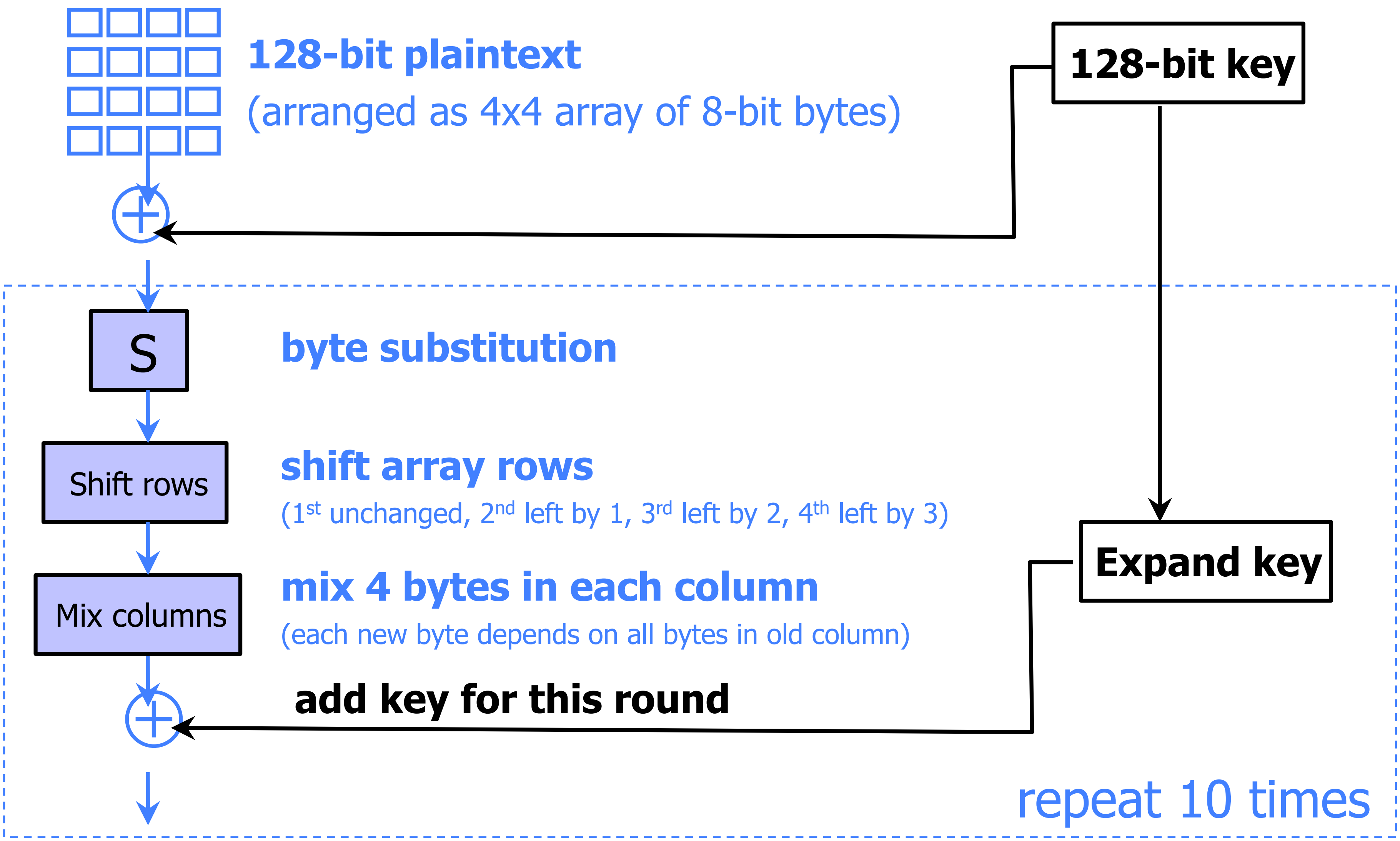
Key Size (bits)	Number of Alternative Keys	Time required at 1 encryption/μs	Time required at 10^6 encryptions/μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31}\ \mu s = 35.8$ minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55}\ \mu s = 1142$ years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\ \mu s = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167}\ \mu s = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26}\ \mu s = 6.4 \times 10^{12}$ years	6.4×10^6 years

- 1999: EFF DES Crack + distibuted machines
  - < 24 hours to find DES key
- DES ---> 3DES
  - 3DES: DES + inverse DES + DES (with 2 or 3 diff keys)

Advanced Encryption Standard (AES)

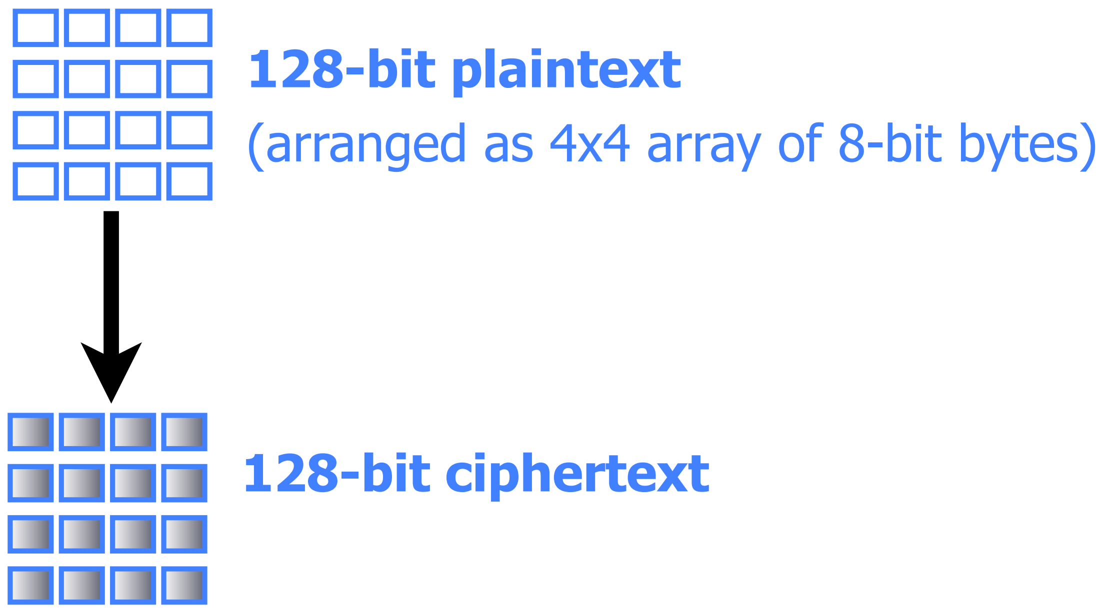
- New federal standard as of 2001
- Based on the Rijndael algorithm
- 128-bit blocks, keys can be 128, 192 or 256 bits
- Unlike DES, does not use Feistel structure
  - The entire block is processed during each round
- Design uses some very nice mathematics

Basic Structure of Rijndael



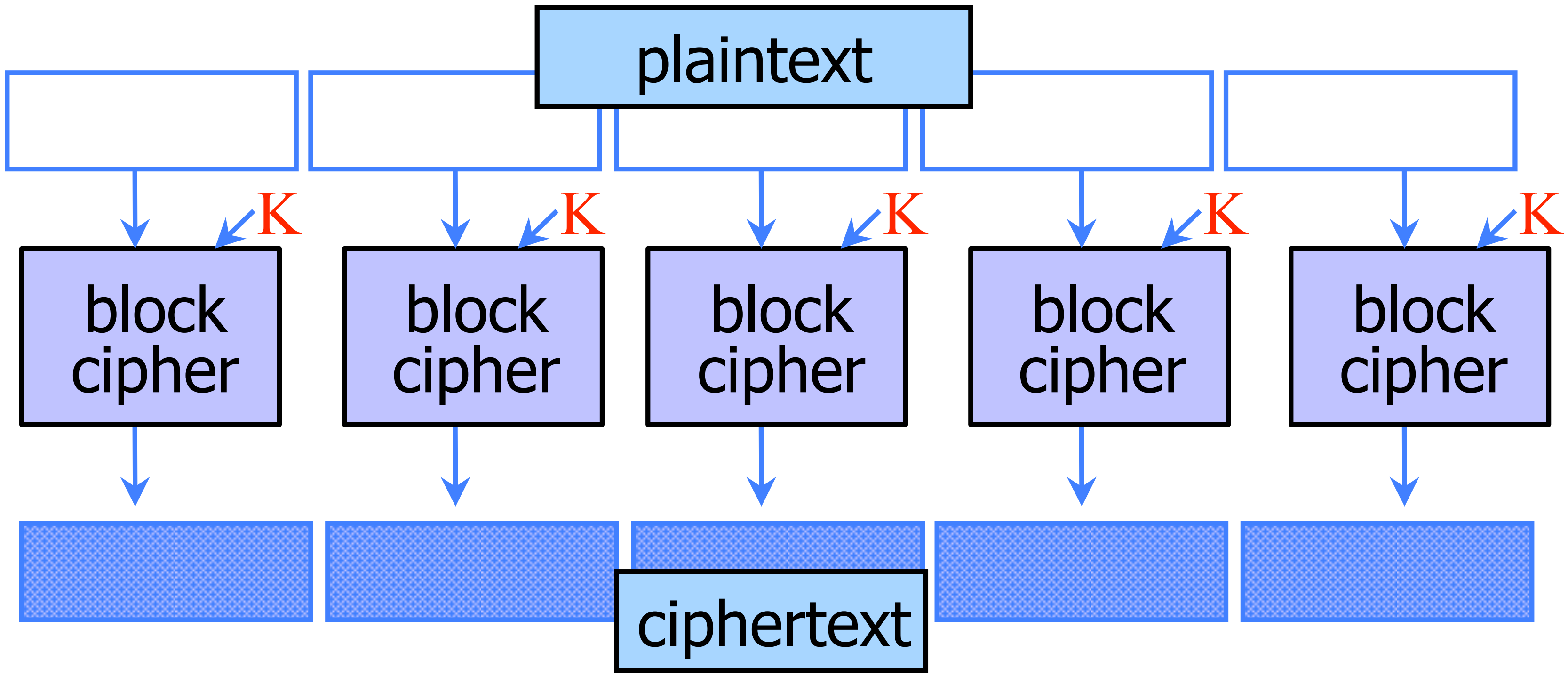
Encrypting a Large Message

- So, we've got a good block cipher, but our plaintext is larger than 128-bit block size



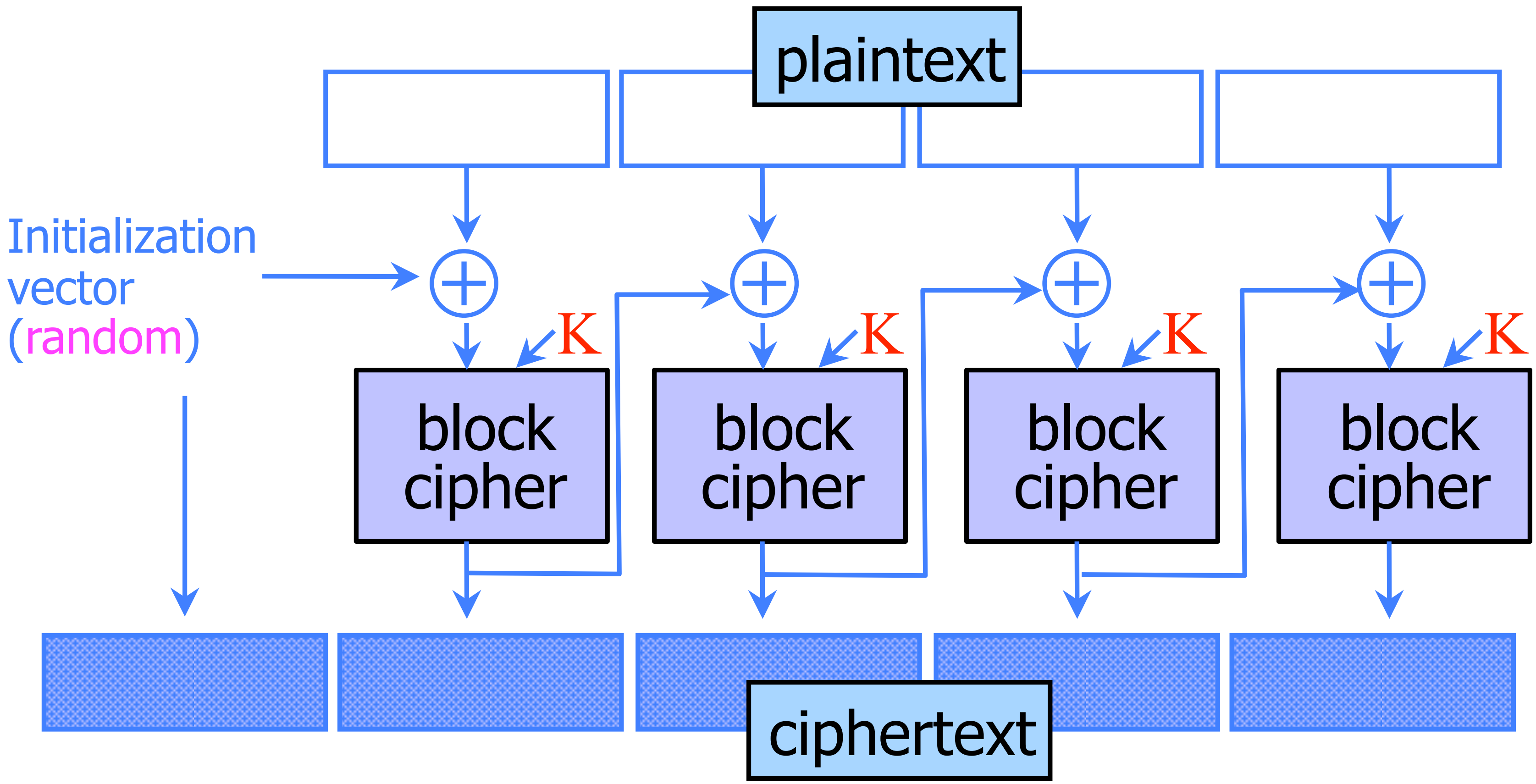
- What should we do?

Electronic Code Book (ECB) Mode



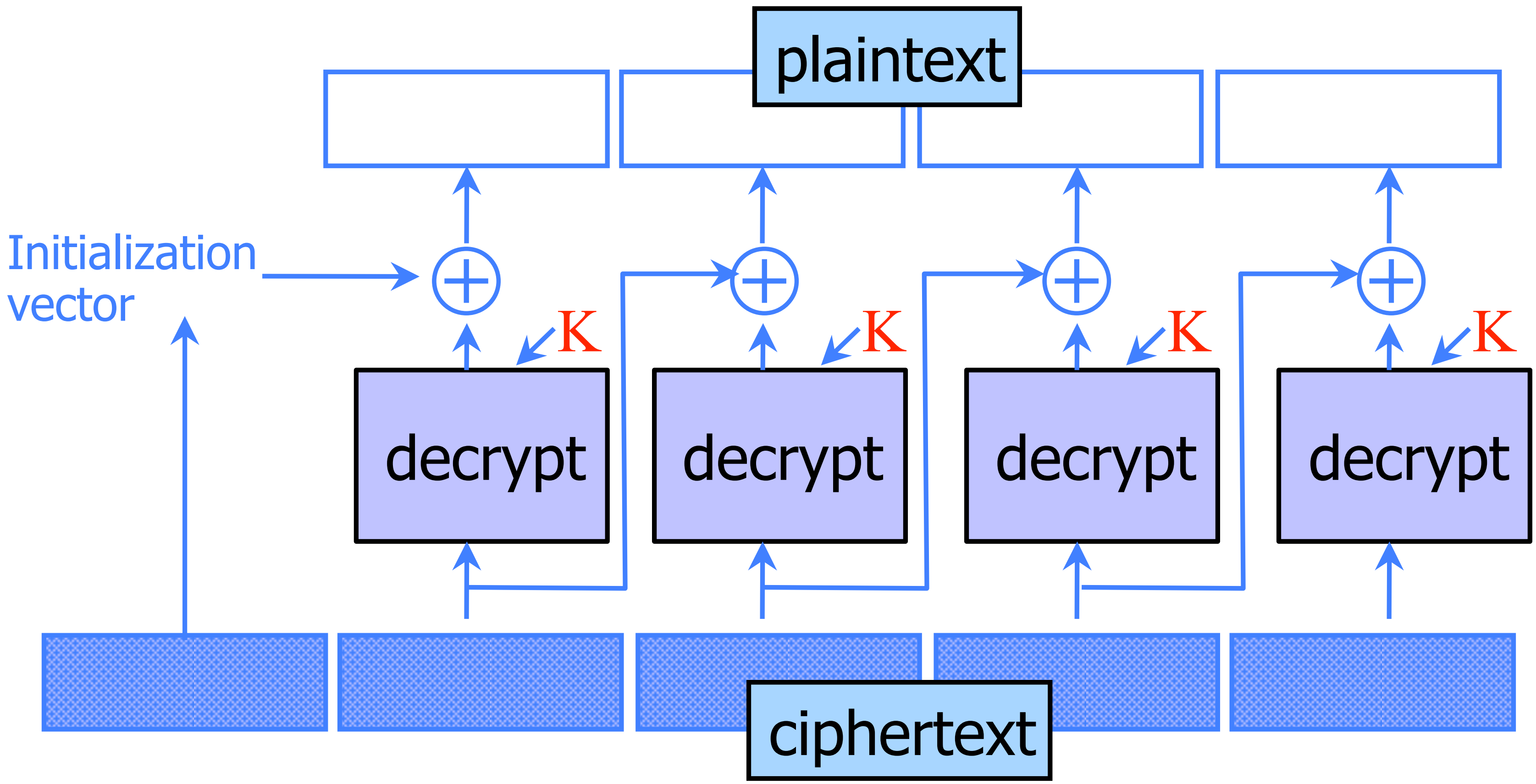
- Identical blocks of plaintext produce identical blocks of ciphertext
- No integrity checks: can mix and match blocks

Cipher Block Chaining (CBC) Mode: Encryption



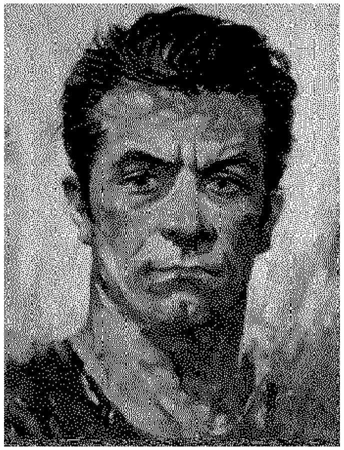
- Identical blocks of plaintext encrypted differently
- Last cipherblock depends on entire plaintext
  - Still does not guarantee integrity

CBC Mode: Decryption



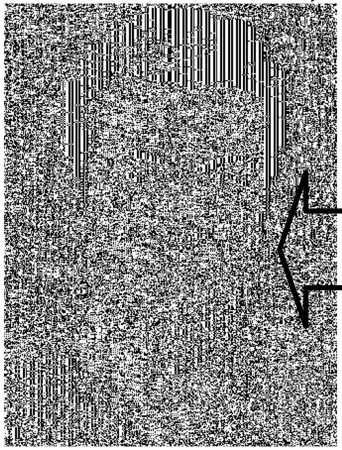
ECB vs. CBC

[Picture due to Bart Preneel]

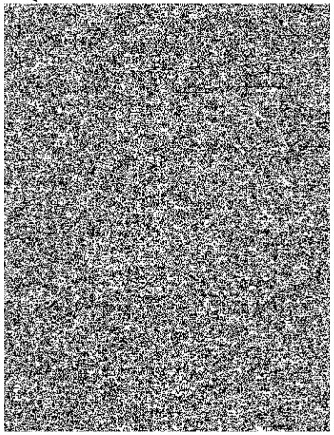


AES in ECB mode

AES in CBC mode

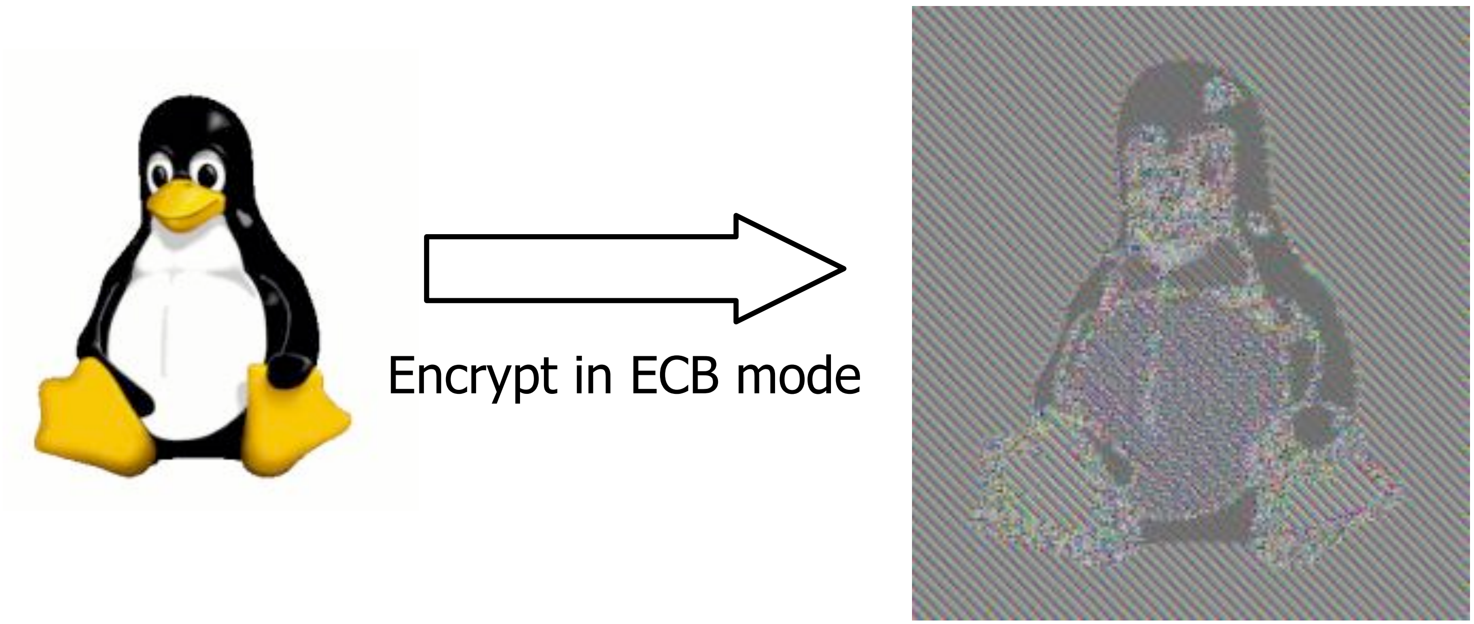


Similar plaintext blocks produce similar ciphertext blocks (not good!)

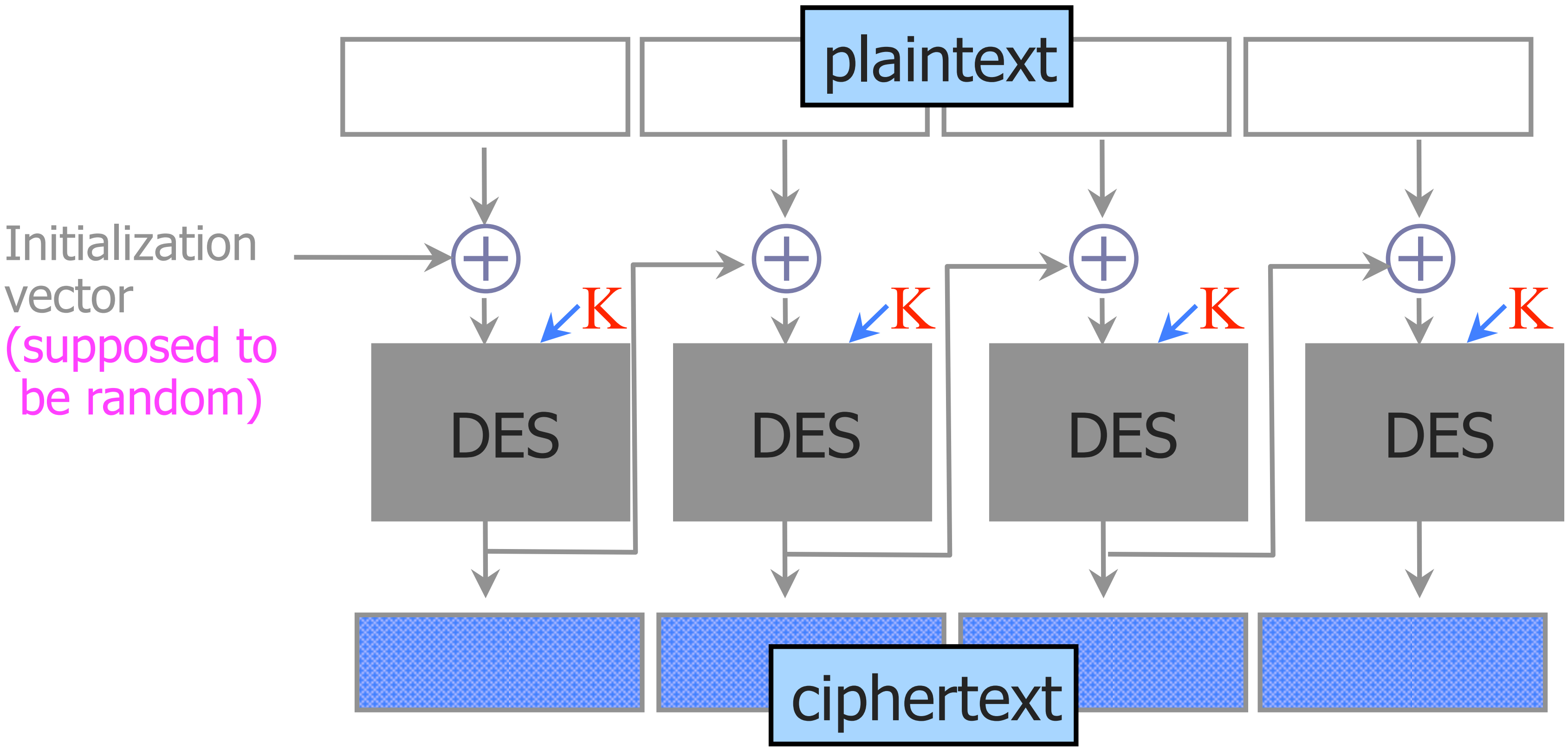


Information Leakage in ECB Mode

[Wikipedia]



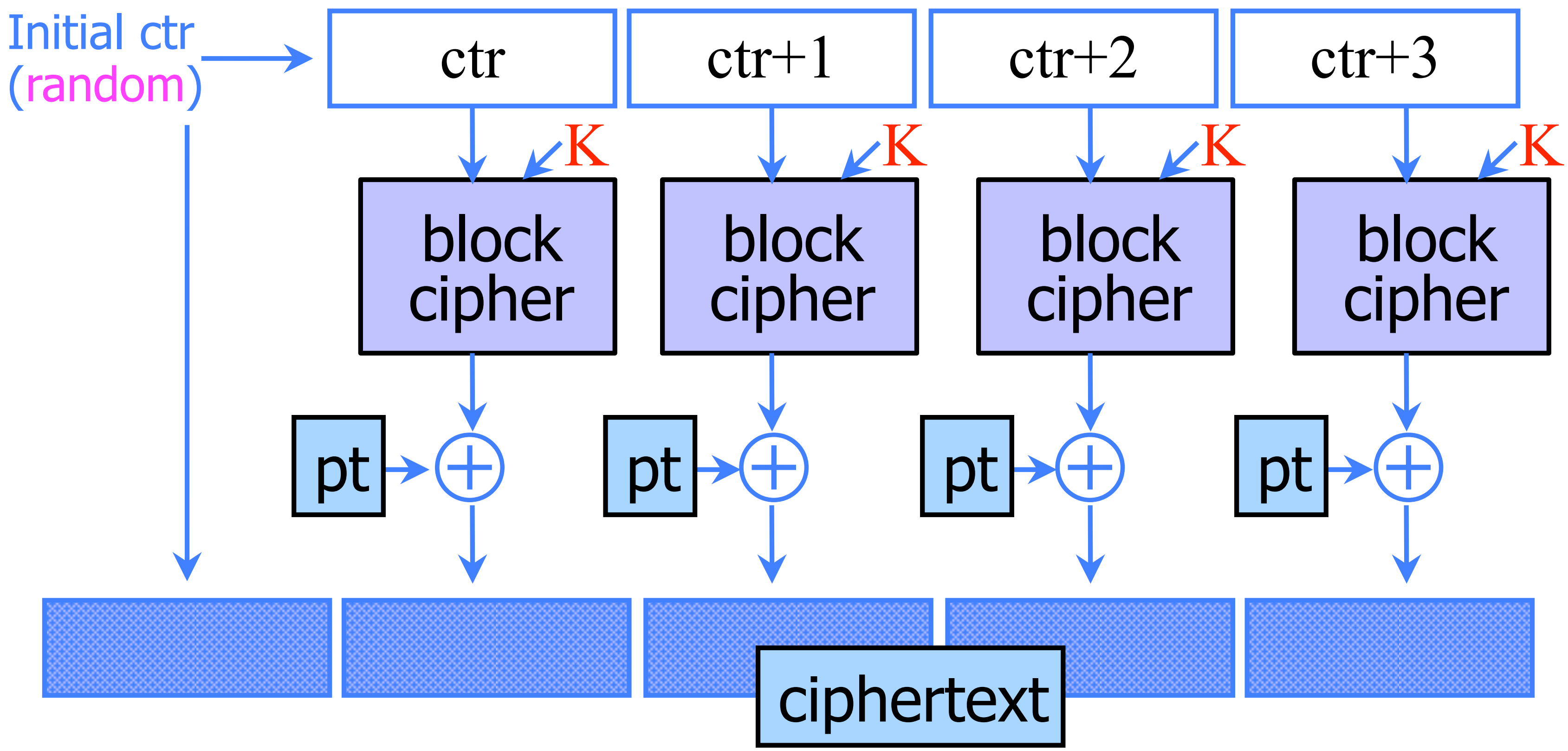
CBC and Electronic Voting



Found in the source code for Diebold voting machines:

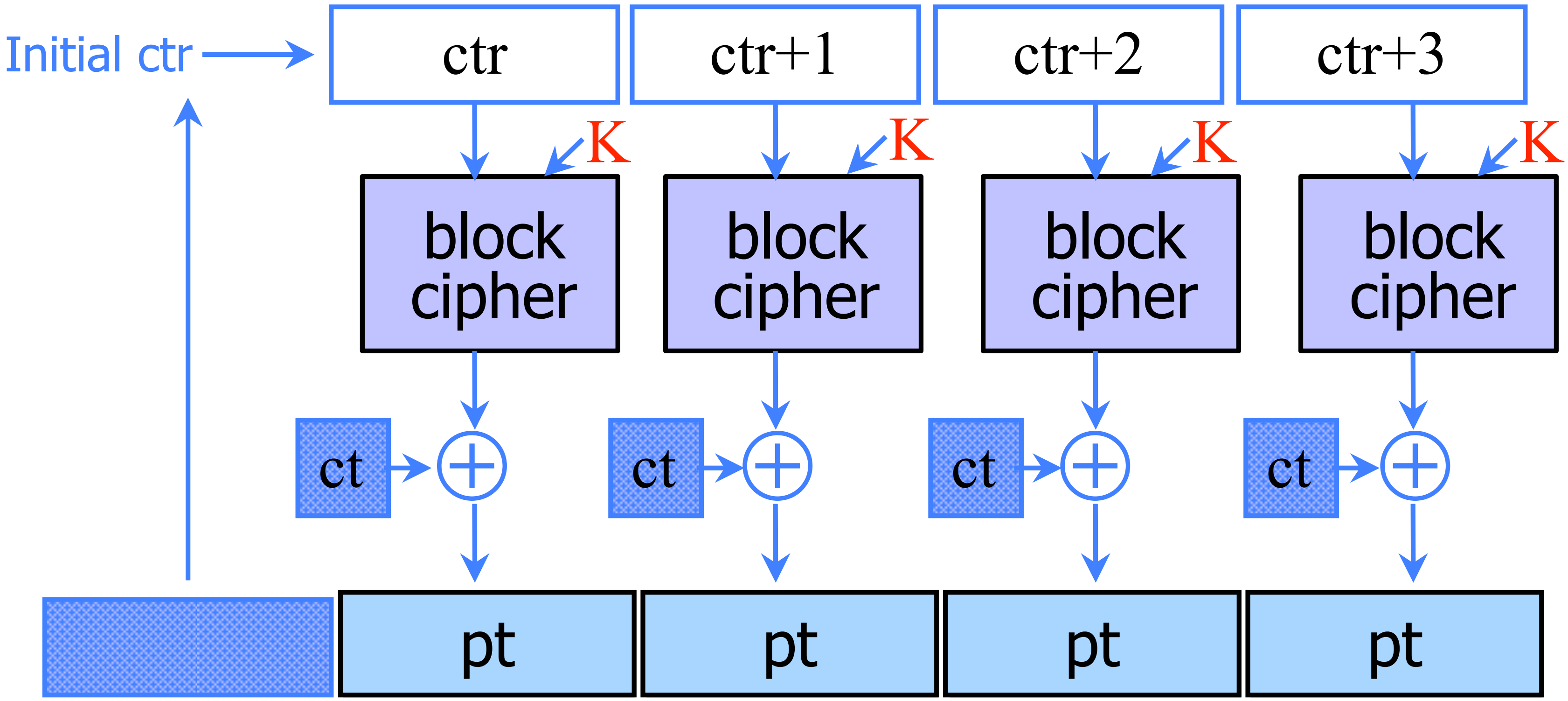
```
DesCBCEncrypt((des_c_block*)tmp, (des_c_block*)record.m_Data,
              totalSize, DESKEY, NULL, DES_ENCRYPT)
```

Counter (CTR) Mode: Encryption



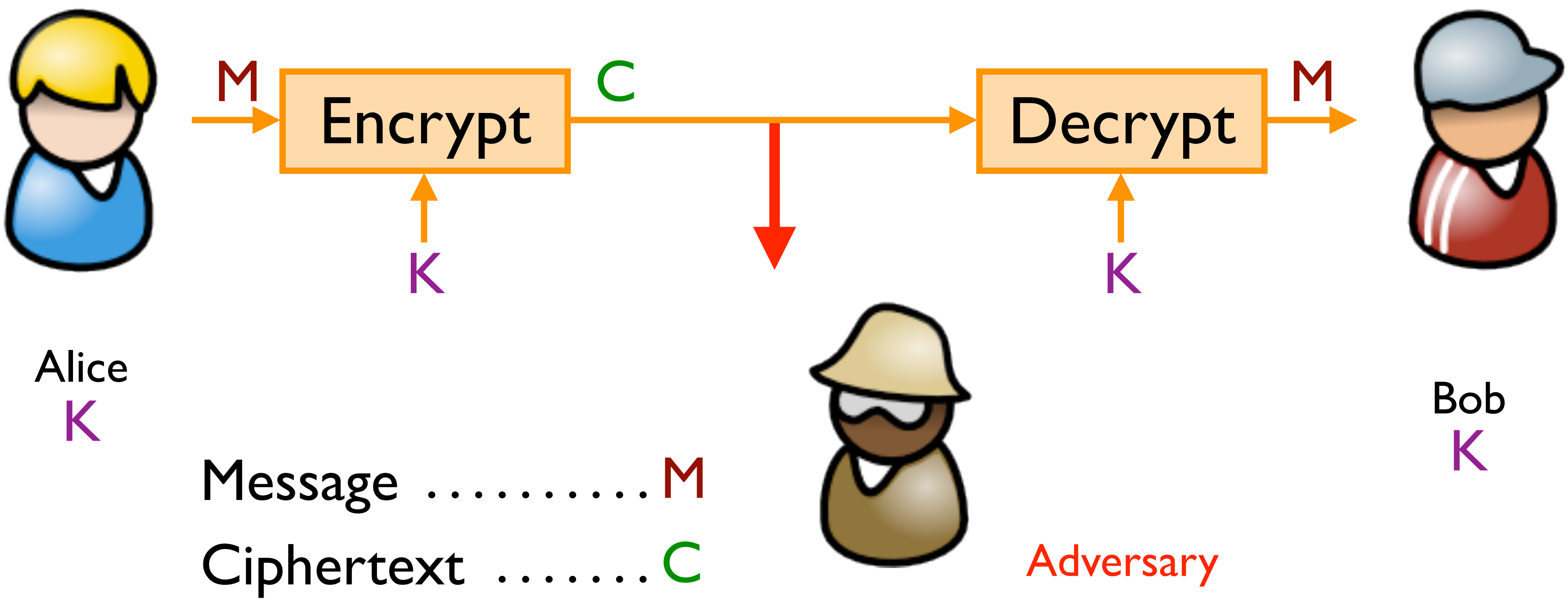
- Identical blocks of plaintext encrypted differently
- Still does not guarantee integrity
- Fragile if ctr repeats

CTR Mode: Decryption



Achieving Privacy (Symmetric)

Encryption schemes: A tool for protecting privacy.



Message . . . . M

Ciphertext C

When Is an Encryption Scheme "Secure"?

- Hard to recover the key?
  - What if attacker can learn plaintext without learning the key?
- Hard to recover plaintext from ciphertext?
  - What if attacker learns some bits or some function of bits?
- Fixed mapping from plaintexts to ciphertexts?
  - What if attacker sees two identical ciphertexts and infers that the corresponding plaintexts are identical?
  - Implication: encryption must be randomized or stateful

How Can a Cipher Be Attacked?

- Assume that the attacker knows the encryption algorithm and wants to learn information about some ciphertext
- Main question: what else does attacker know?
  - Depends on the application in which cipher is used!
- Ciphertext-only attack
- 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 <u>except</u> the target
  - Sometimes very realistic model

Defining Security (Not Required)

- Attacker does not know the key
- He chooses as many plaintexts as he wants, and learns 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

Defining Security (Not Required)

- Idea: attacker should not be able to learn even a single bit of the encrypted plaintext
- Define Enc(M_0,M_1,b) to be a function that returns encrypted M_b

0 or 1

  - Given two plaintexts, Enc returns a ciphertext of one or the other depending on the value of bit b
  - Think of Enc as a magic box that computes ciphertexts on attacker's demand. He can obtain a 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 one bit b

Chosen-Plaintext Security (Not Required)

- Consider two experiments (A is the attacker)

Experiment 0

A interacts with Enc(-,-,0) and outputs bit d

Experiment 1

A interacts with Enc(-,-,1) and outputs bit d

  - Identical except for the value of the secret bit
  - d is attacker's guess of the secret bit

If A "knows" secret bit, he should be able to make his output depend on it

- 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 (Not Required)

- 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,Y,b); $C_2 \leftarrow$ Enc(Y,Y,b);

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

- The advantage of this attacker A is 1

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

Why Hide Everything?

- Leaking even a little bit of information about the plaintext can be disastrous
- Electronic voting
  - 2 candidates on the ballot (1 bit to encode the vote)
  - If ciphertext leaks the parity bit of the encrypted plaintext, eavesdropper learns the entire vote
- Also, want a strong definition, that implies other definitions (like not being able to obtain key)