Cryptography (cont.)

Daniel Halperin
Tadayoshi Kohno

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

Updates Oct. 14th

- **Coffee/tea** signup sheet posted (optional)
  - Next is Tuesday @3 pm. Meet in CSE Atrium
- **Lab 1** due in 1 week
  - TA office hours Mon, Fri before class (CSE 002)
  - My office hours Mon, Wed after class (CSE 210)

Checkpoint

- **Symmetric cryptography**
  - Both sides know ***shared key***, no one else knows anything. Can ***encrypt, decrypt, sign/MAC, verify***
  - Computationally lightweight
  - ***Challenge:*** How do you privately share a key?
- **Asymmetric cryptography**
  - Everyone has a ***public*** *key* that everyone else knows; and a paired ***secret*** *key* that is private
  - Public key can ***encrypt***; only secret key can ***decrypt***
  - Secret key can ***sign/MAC***, public key can ***verify***
  - Computationally expensive
  - ***Challenge:*** How do you validate a public key?

Checkpoint

- **Where are public keys from?**
  - One solution: keys for ***Certificate Authorities*** *a priori* known by browser, OS, etc.
- **Where are shared keys from?**
  - In person exchange, snail mail, etc.
  - If we have verifiable public/private keys: ***key exchange*** protocol generates a shared key for symmetric cryptography

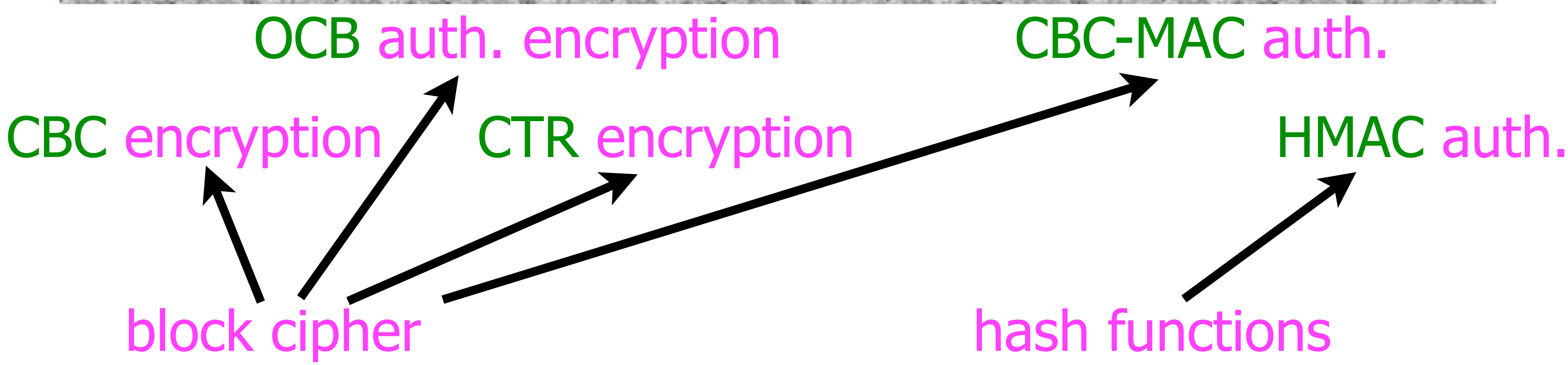
Kerckhoffs's Principle

- Security of a cryptographic object should depend **only** on the secrecy of the secret (private) key

- Security should not depend on the secrecy of the algorithm itself.

How cryptosystems work today

- Layered approach:
  - Cryptographic primitives, like block ciphers, stream ciphers, hash functions, and one-way trapdoor permutations
  - Cryptographic protocols, like CBC mode encryption, CTR mode encryption, HMAC message authentication
- Public algorithms (Kerckhoff's Principle)
- Security proofs based on assumptions (not this course)

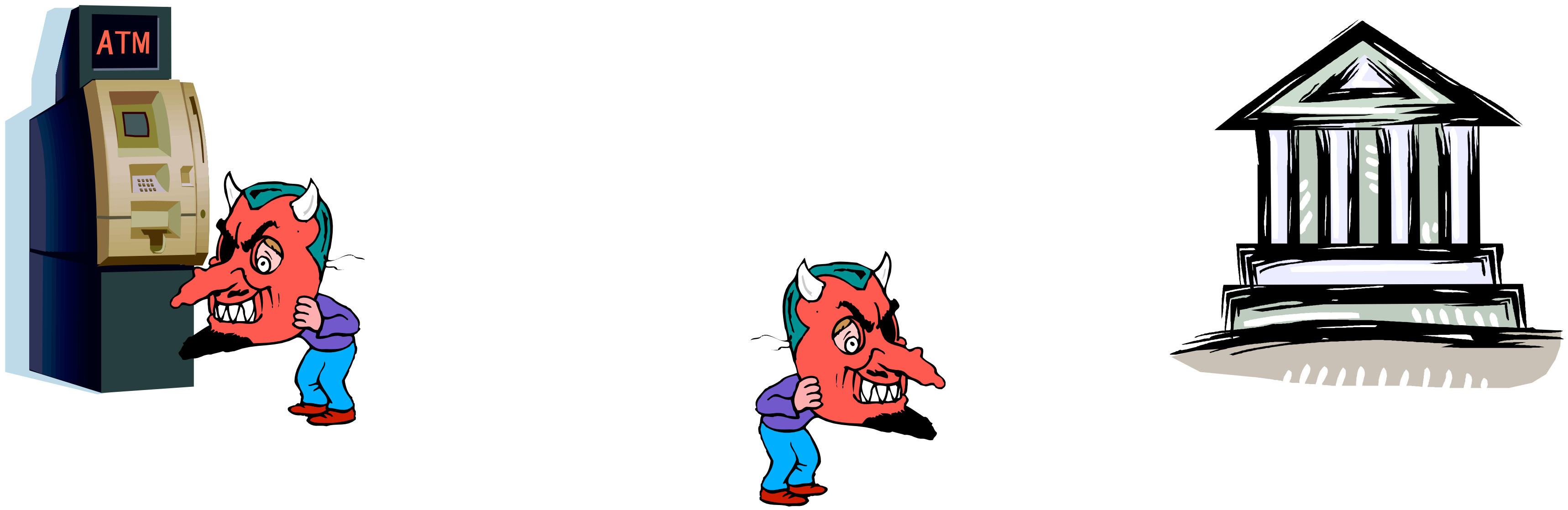


Attack Scenarios for Encryption

- Ciphertext-Only
- Known Plaintext
- Chosen Plaintext
- Chosen Ciphertext (and Chosen Plaintext)

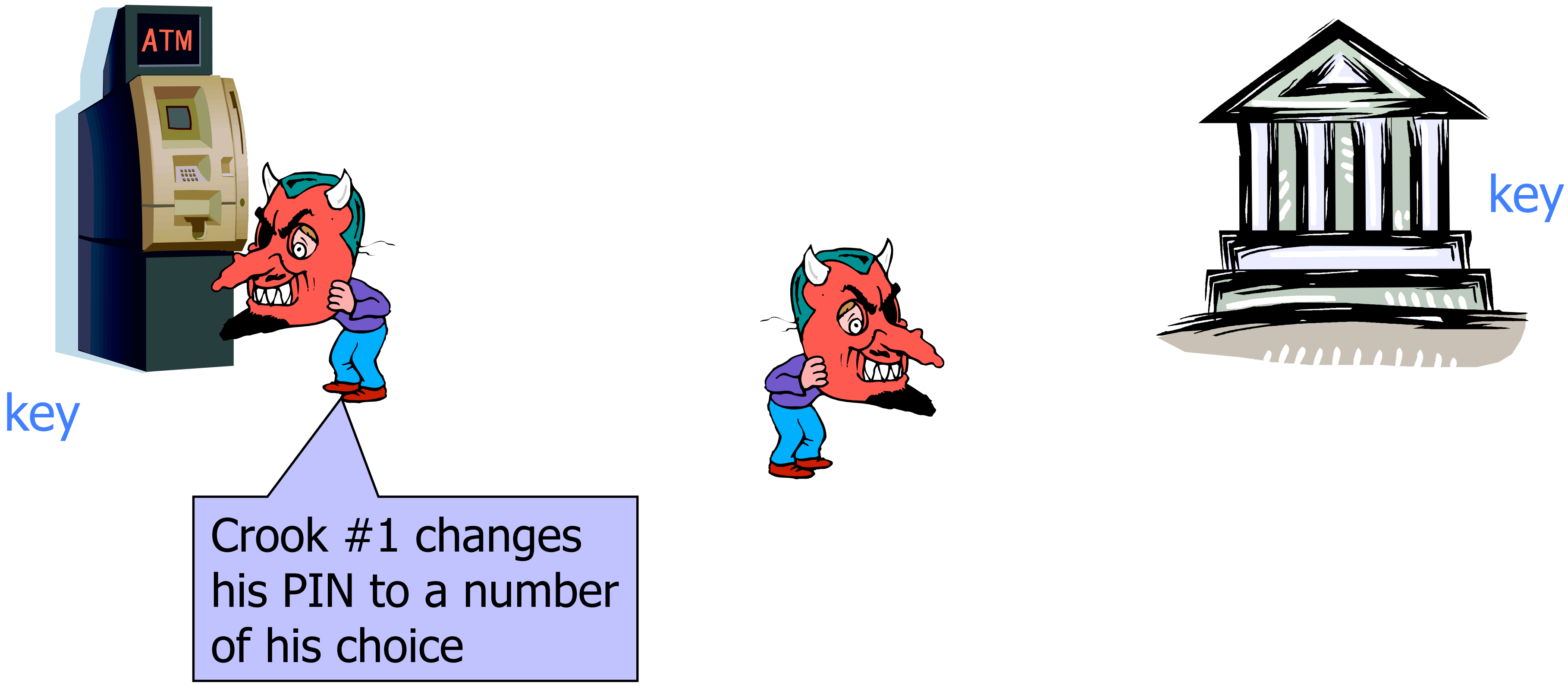
- (General advice: Target strongest level of privacy possible -- even if not clear why -- for extra “safety”)

Chosen-Plaintext Attack

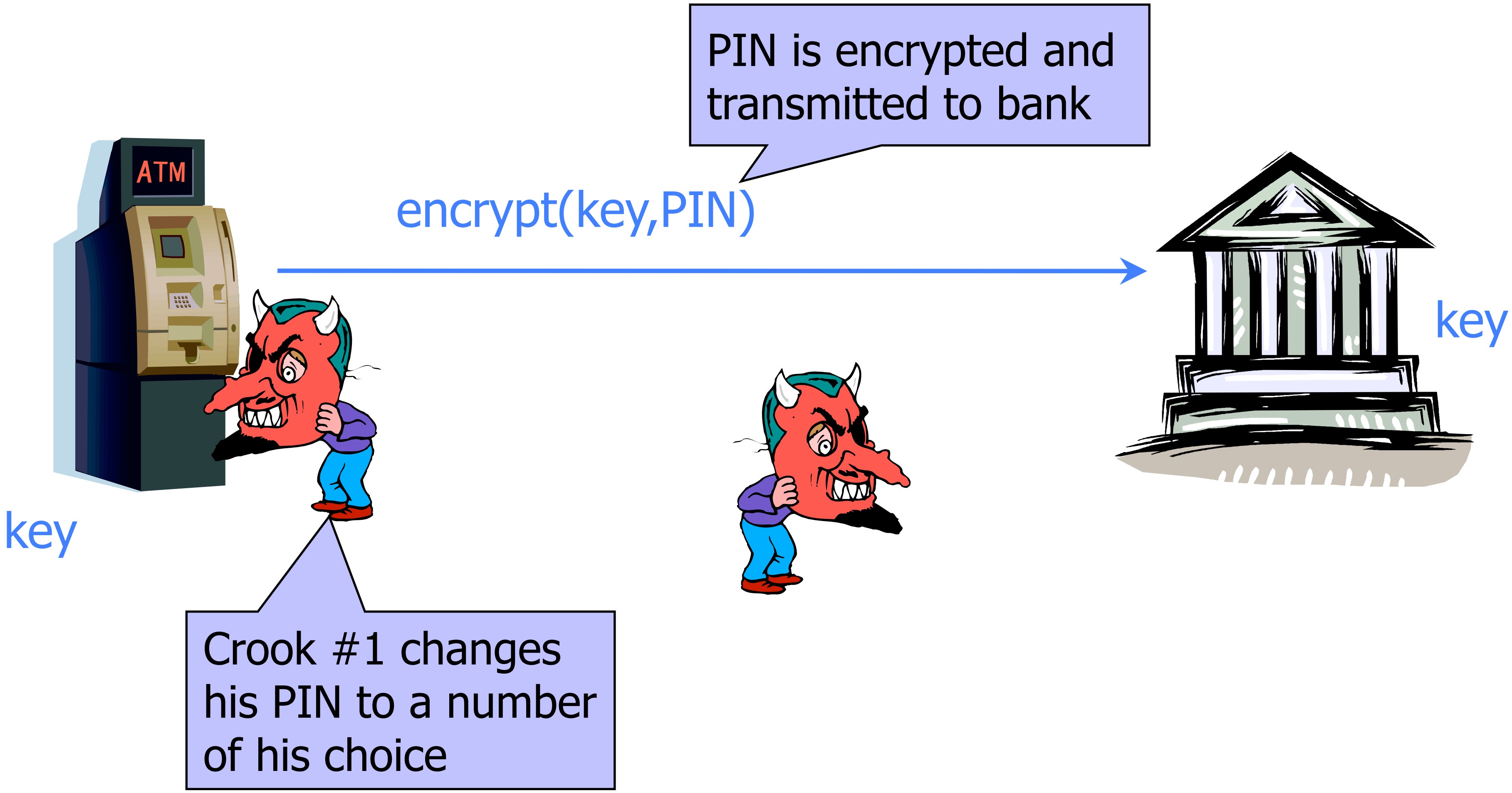


Chosen-Plaintext Attack

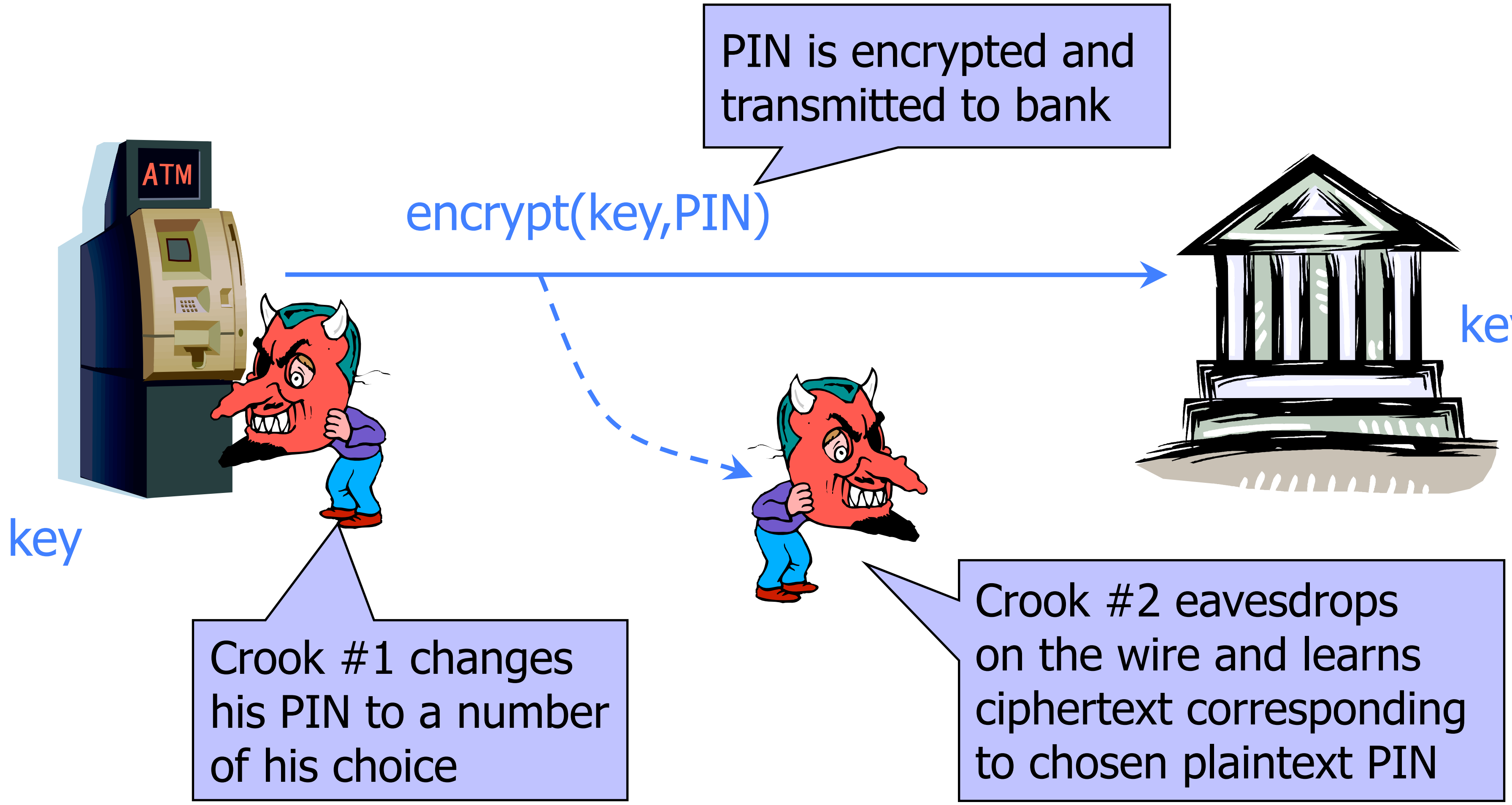
Chosen-Plaintext Attack



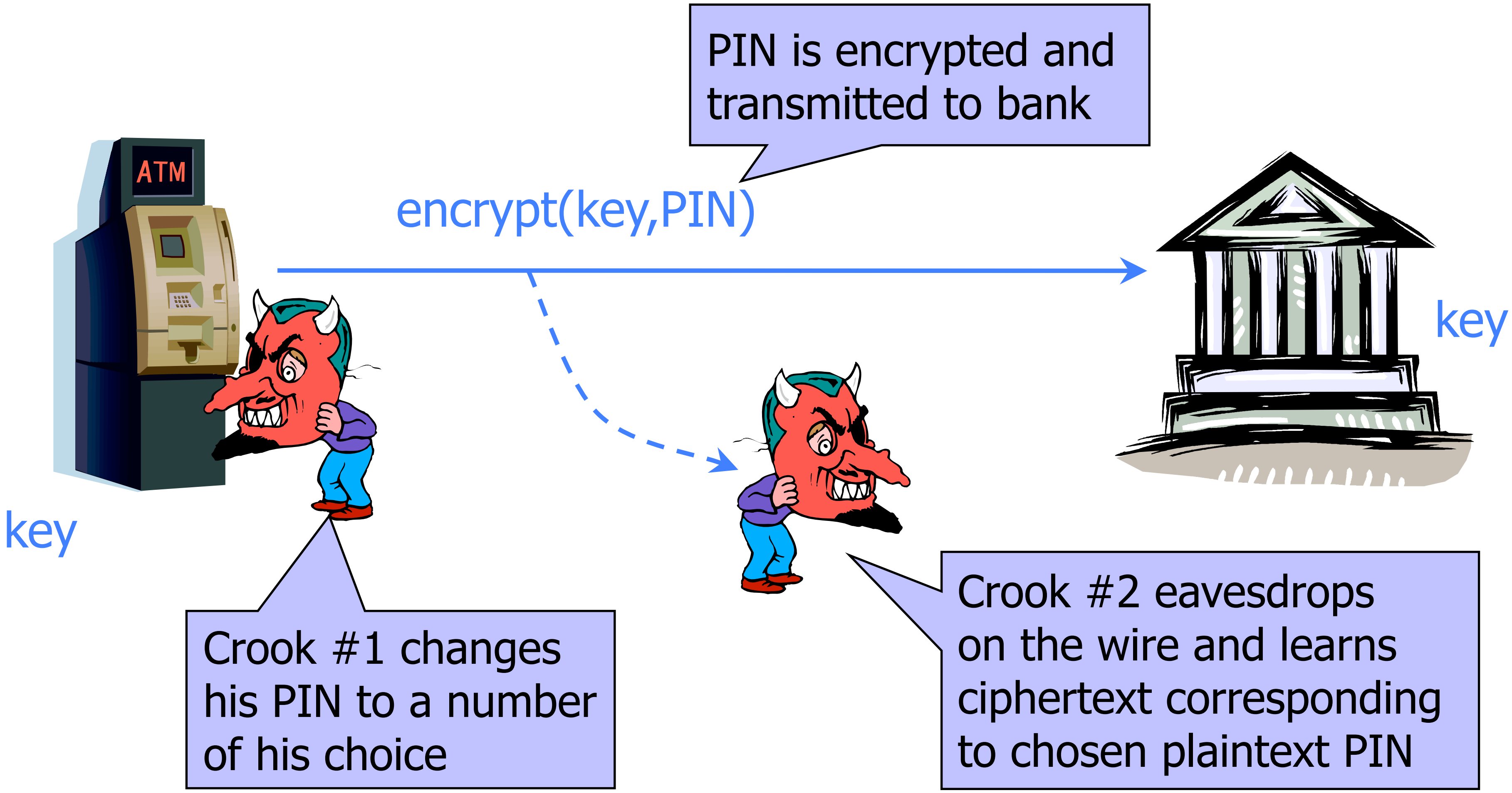
Chosen-Plaintext Attack



Chosen-Plaintext Attack



Chosen-Plaintext Attack



… repeat for any PIN value

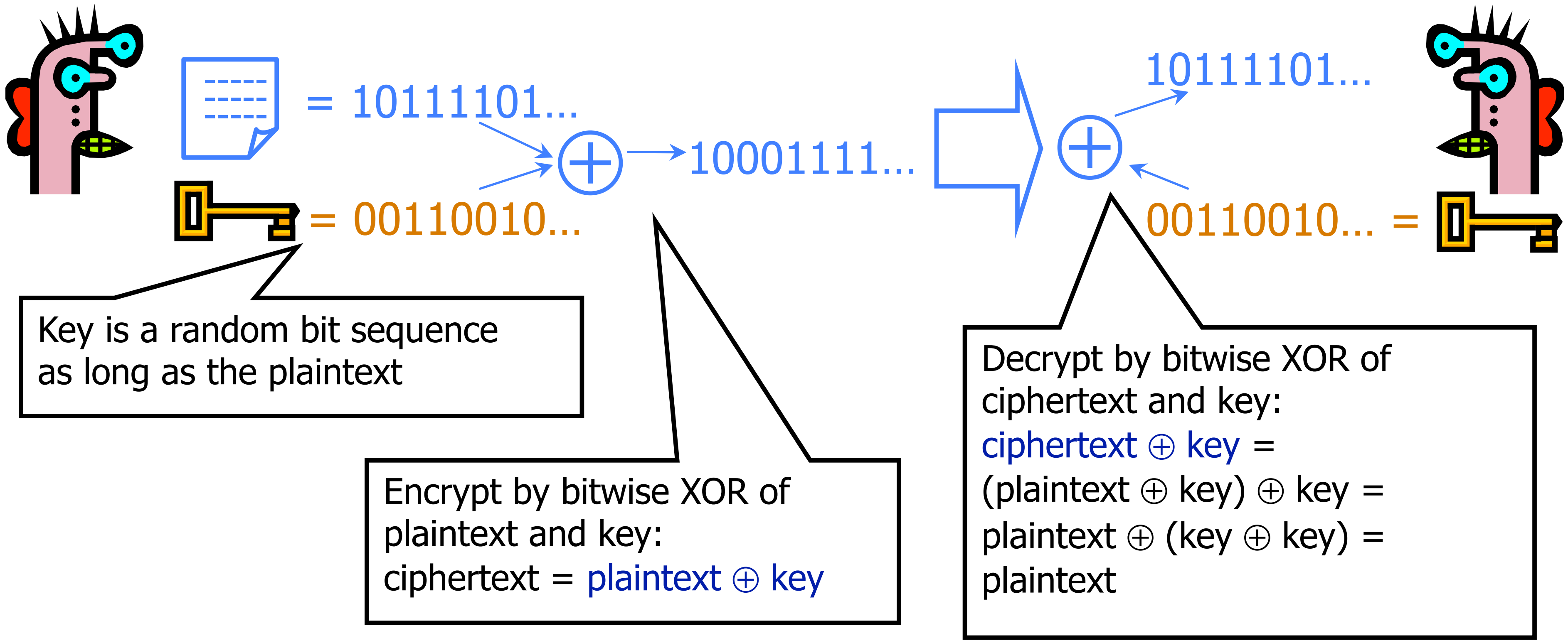
Attack Scenarios for Integrity

- What do you think these scenarios should be?

Perfect Secrecy

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)

One-Time Pad

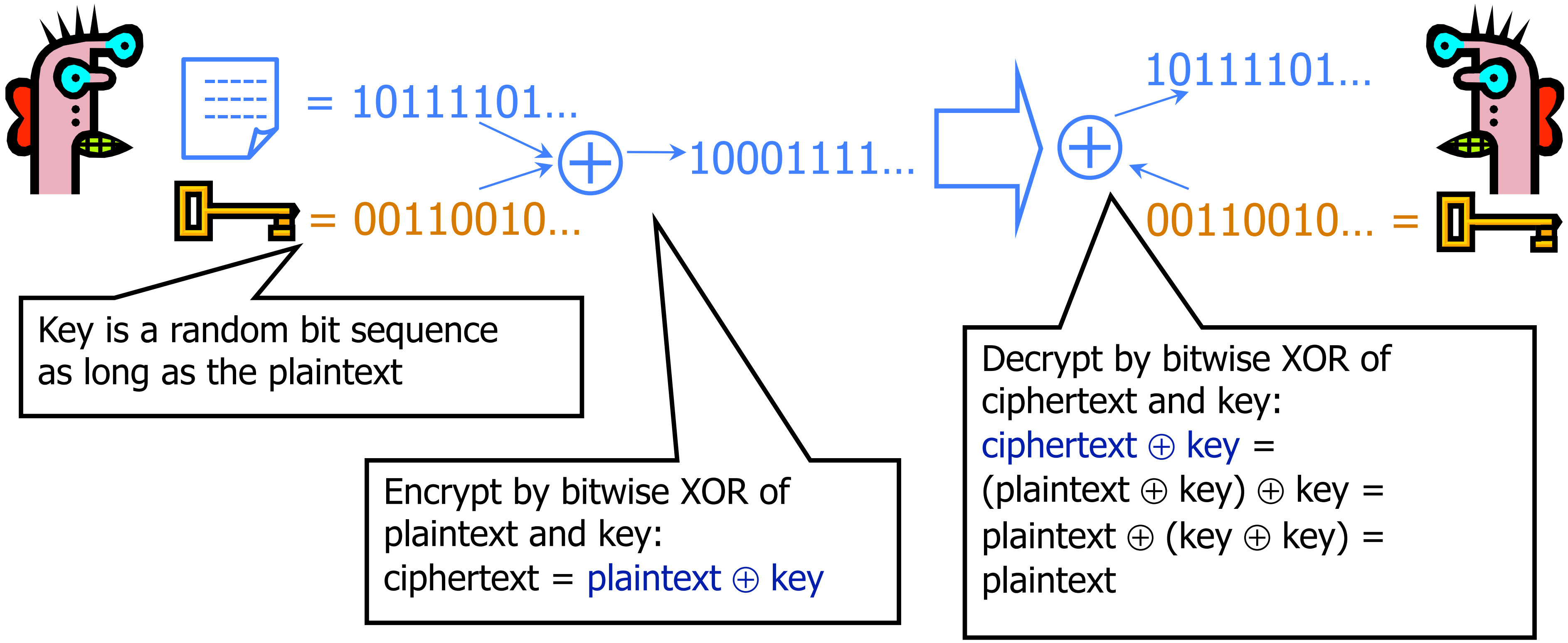


= 10111101…
= 00110010…
10001111…
10111101…
00110010… =
Key is a random bit sequence as long as the plaintext
Encrypt by bitwise XOR of plaintext and key:
ciphertext = plaintext ⊕ key
Decrypt by bitwise XOR of ciphertext and key:
ciphertext ⊕ key =
(plaintext ⊕ key) ⊕ key =
plaintext ⊕ (key ⊕ key) =
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

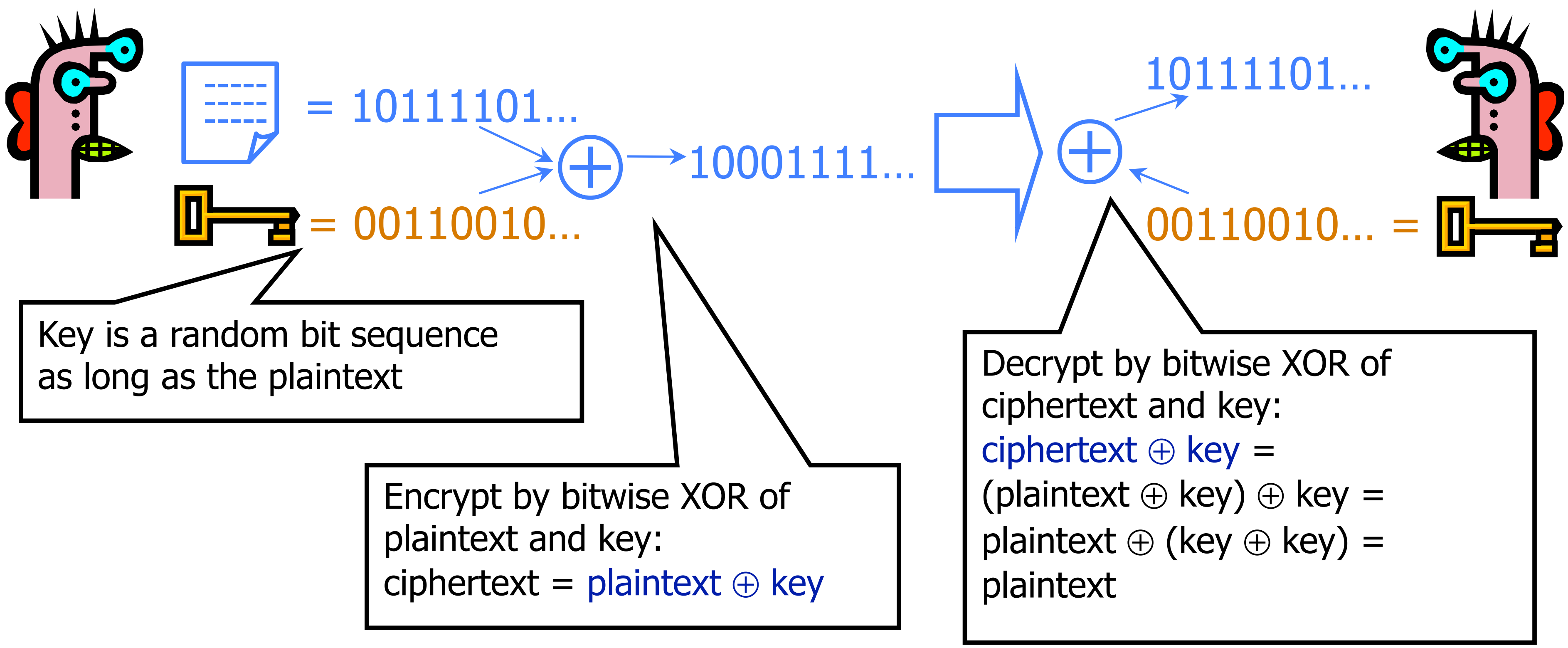


Disadvantage #1: Keys as long as messages.

Impractical in most scenarios

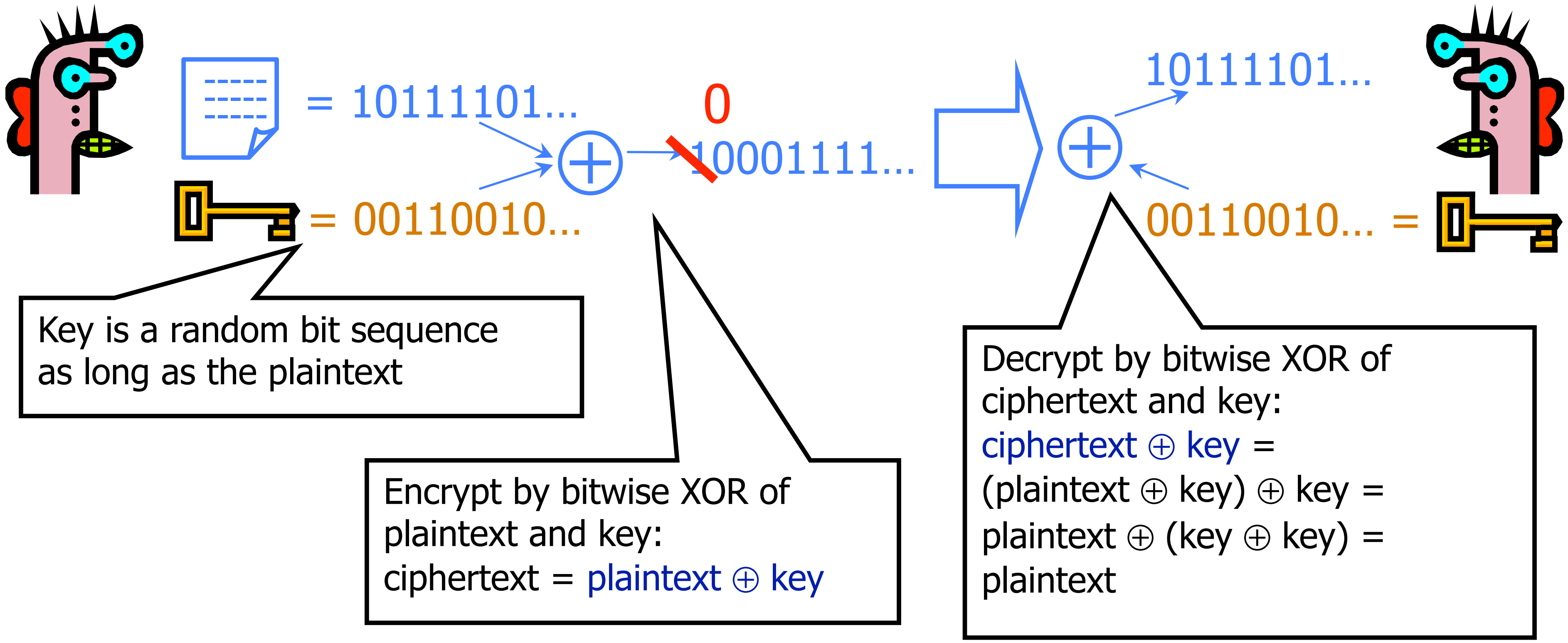
Still used by intelligence communities

Disadvantages



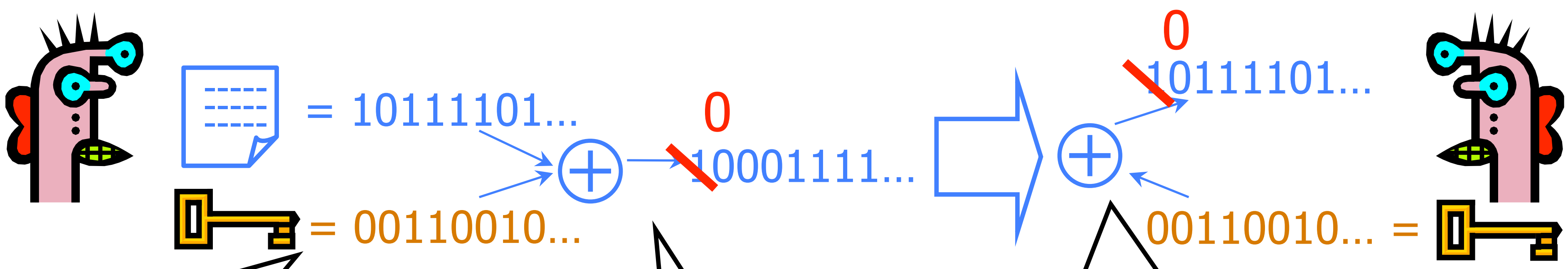
Disadvantage #2: No integrity protection

Disadvantages



Disadvantage #2: No integrity protection

Disadvantages



Key is a random bit sequence as long as the plaintext

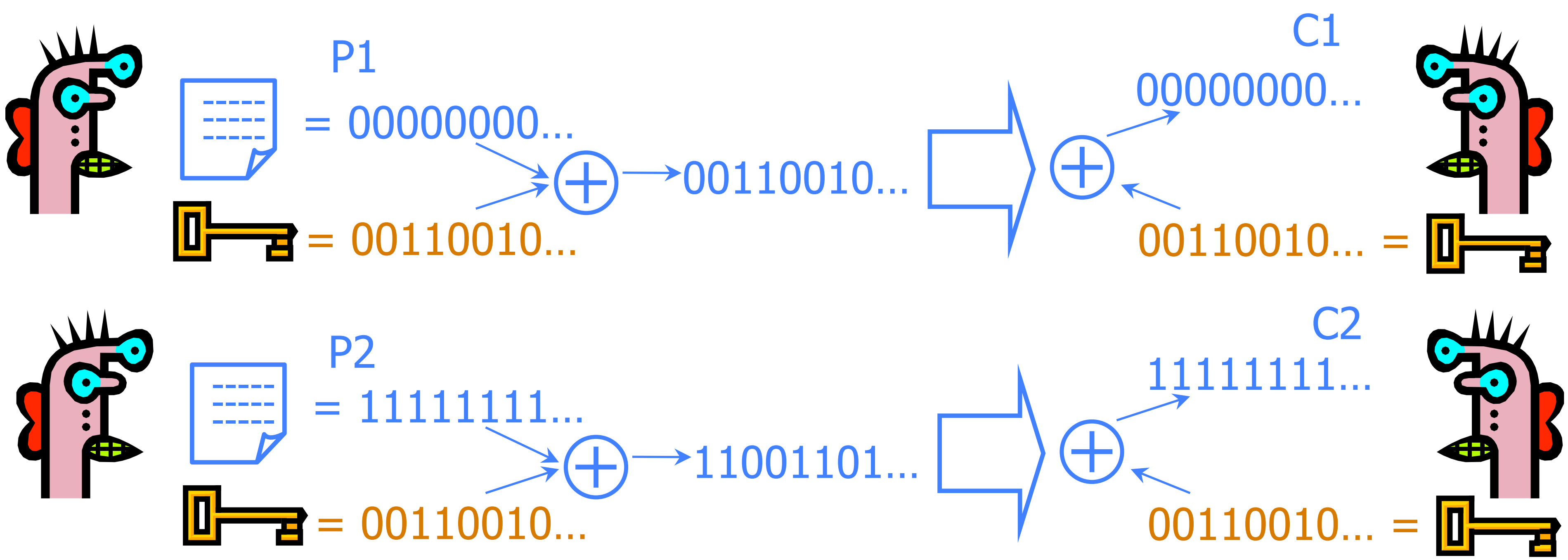
Encrypt by bitwise XOR of plaintext and key:
ciphertext = plaintext ⊕ key

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

Disadvantage #2: No integrity protection

Disadvantages

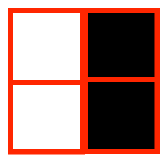
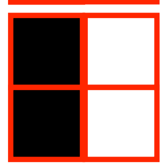
Disadvantage #3: Keys cannot be reused

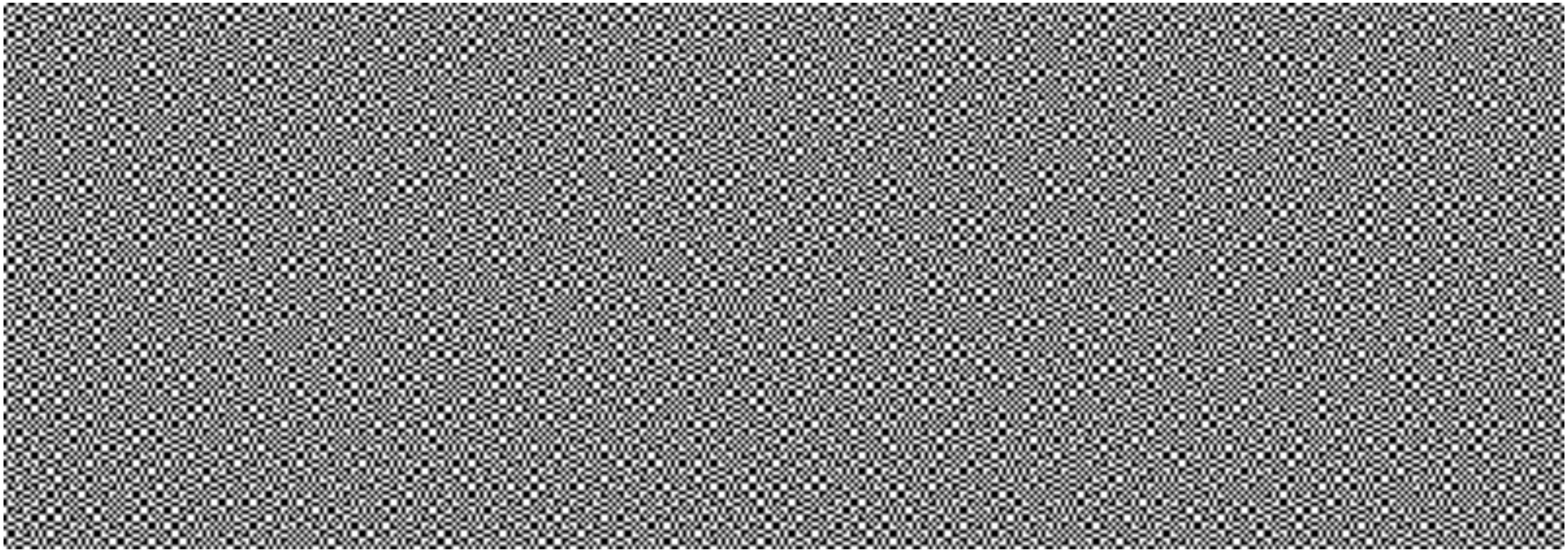


Learn relationship between plaintexts:

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

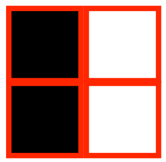
Visual Cryptography

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

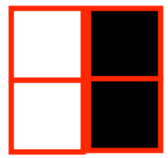


Visual Cryptography

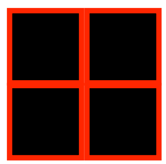
- Take a black and white bitmap image
- For a white pixel, send the same as the mask



or

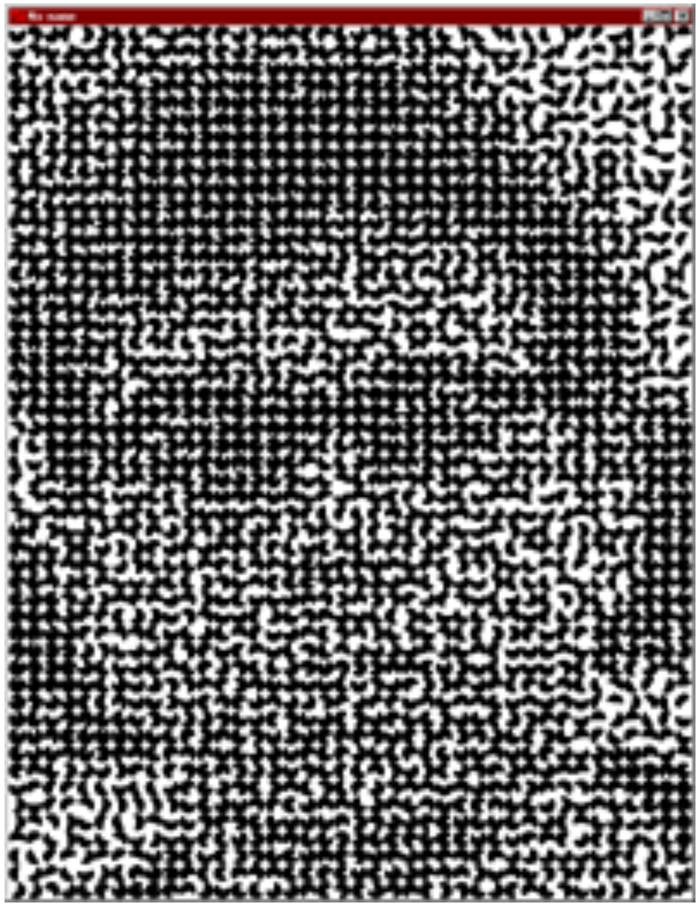


- For a black pixel, send the opposite of the mask



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

Visual Cryptography



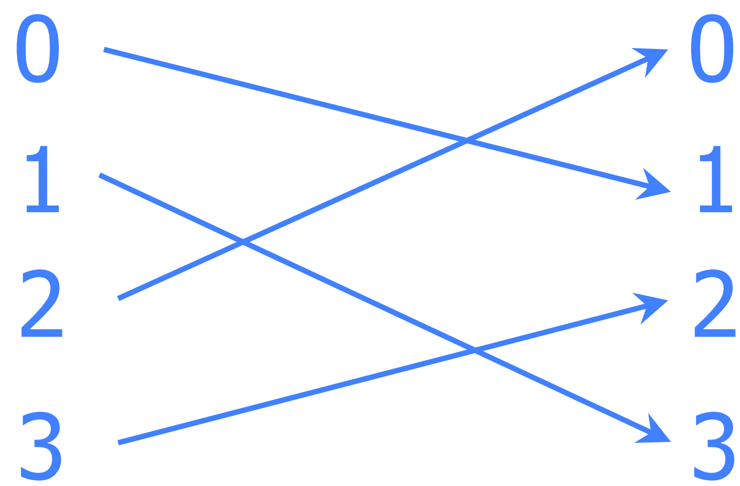
- http://www.cl.cam.ac.uk/~fms27/vck/face.gif

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

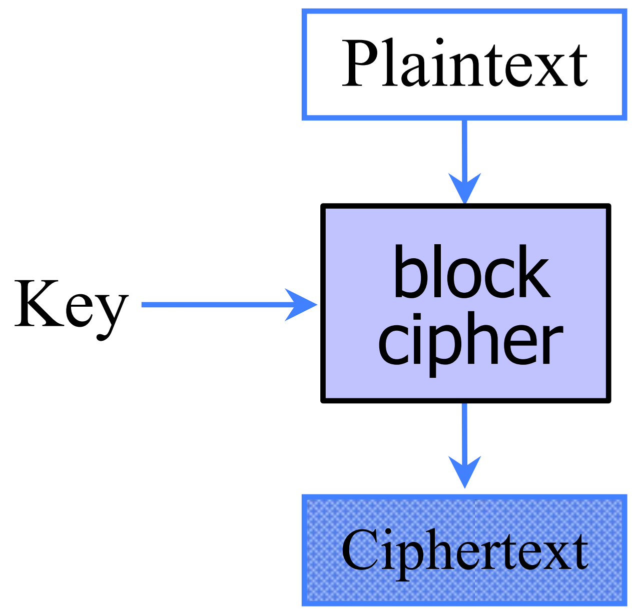
Background: Permutation



- 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)

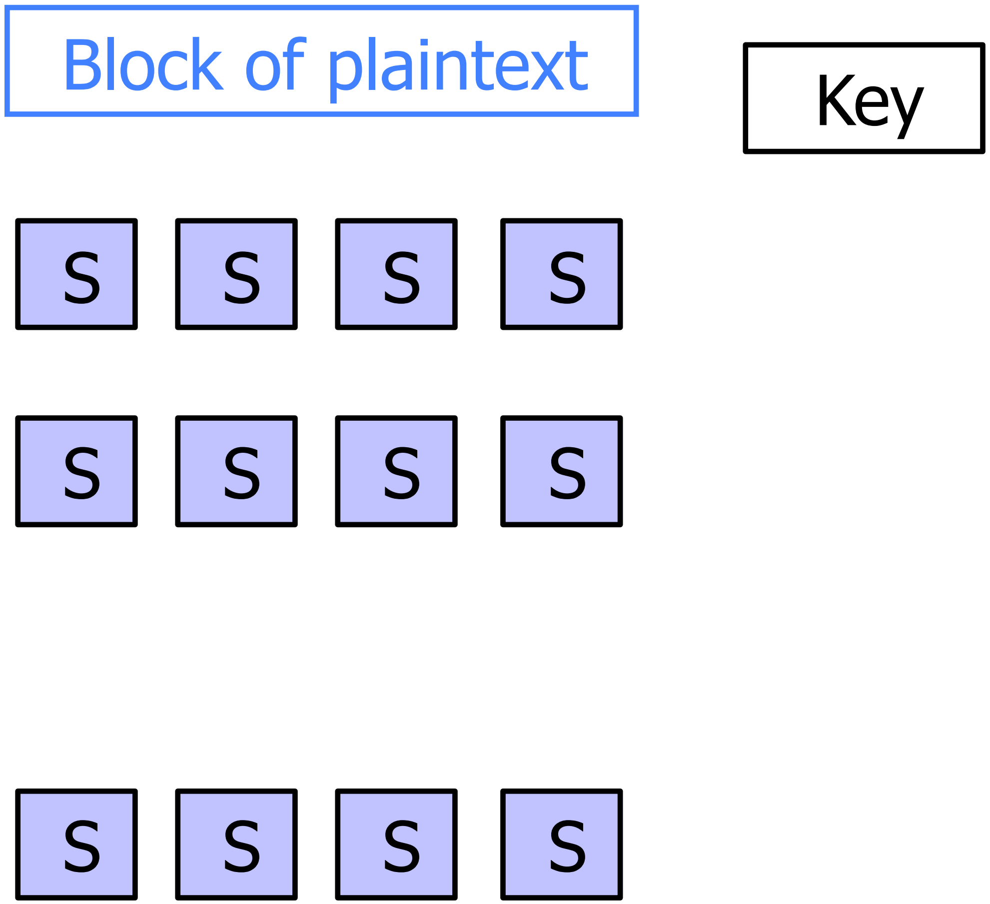


Block Cipher Security

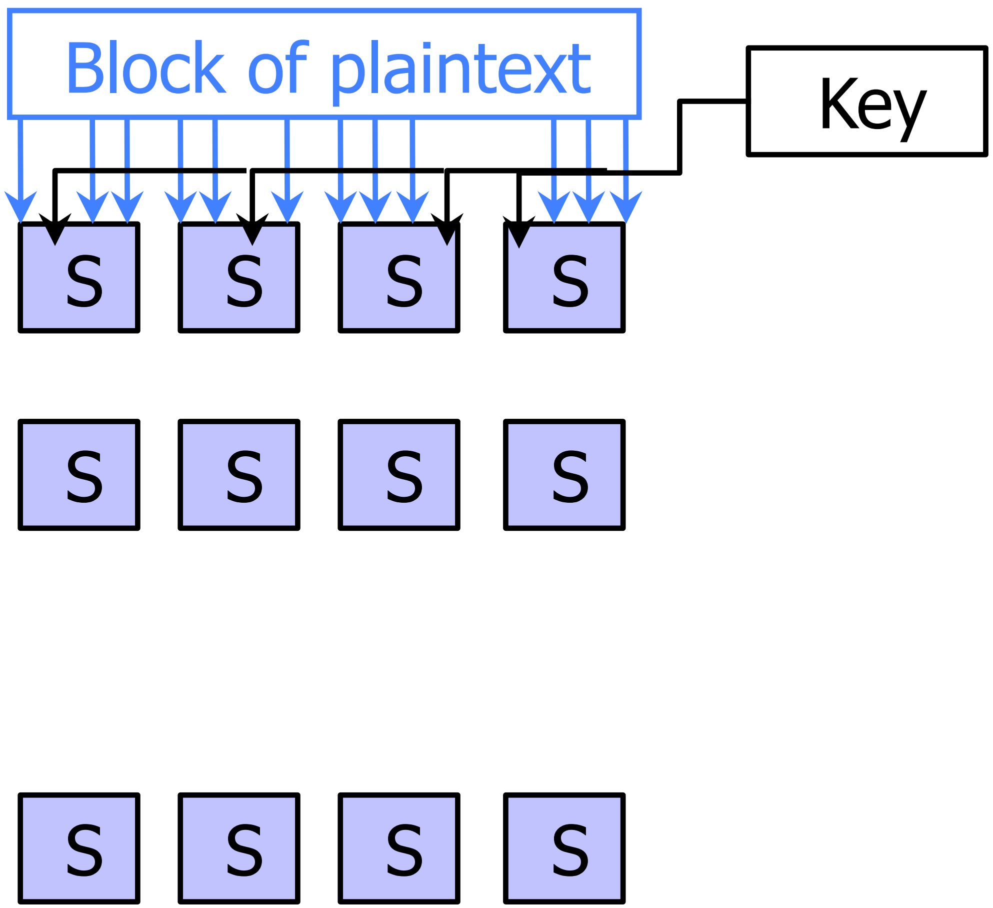
- Result should look like a random permutation on the inputs
  - Recall: not just shuffling bits. N-bit block cipher permutes over 2^N inputs.

- 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

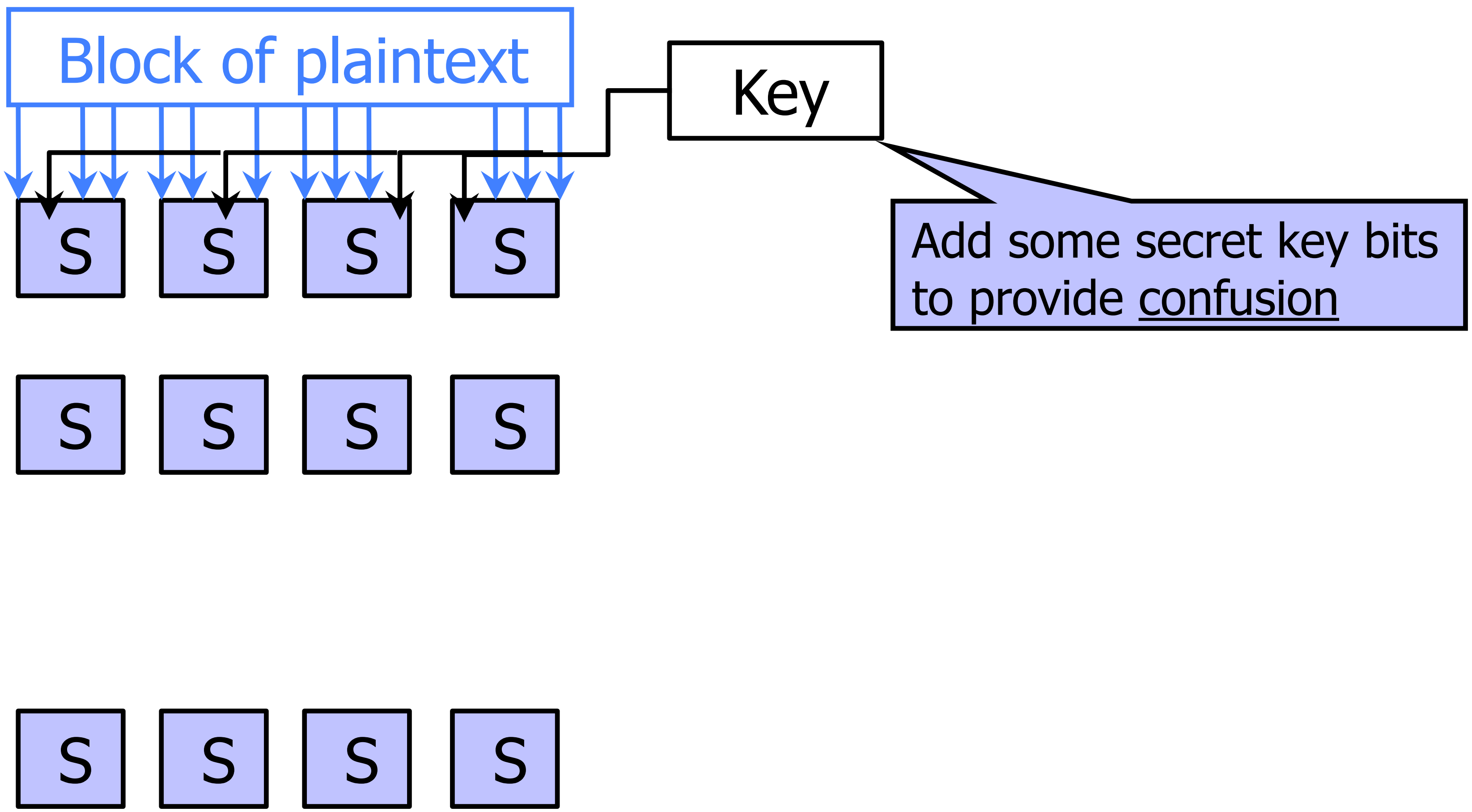
Block Cipher Operation (Simplified)



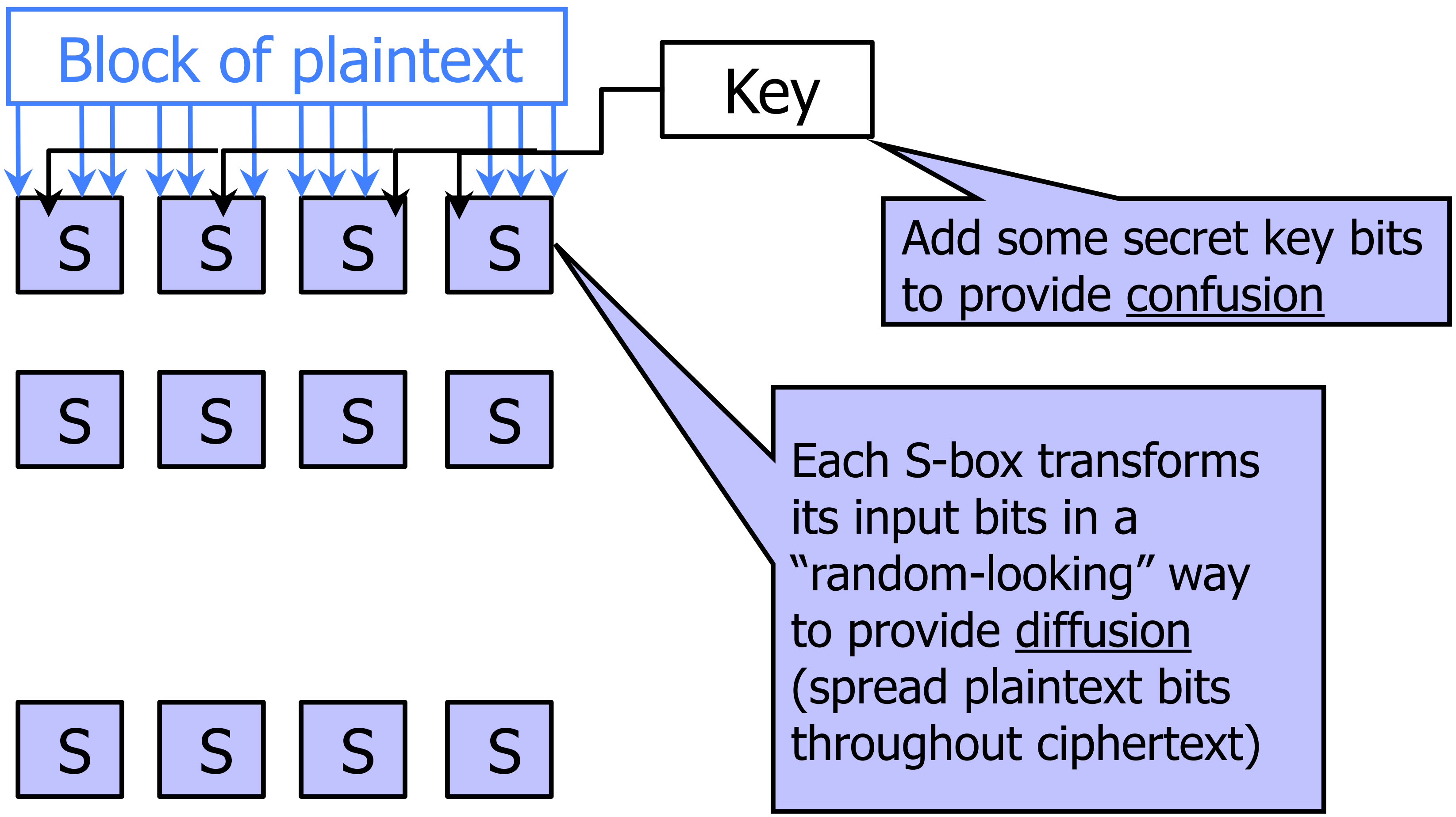
Block Cipher Operation (Simplified)



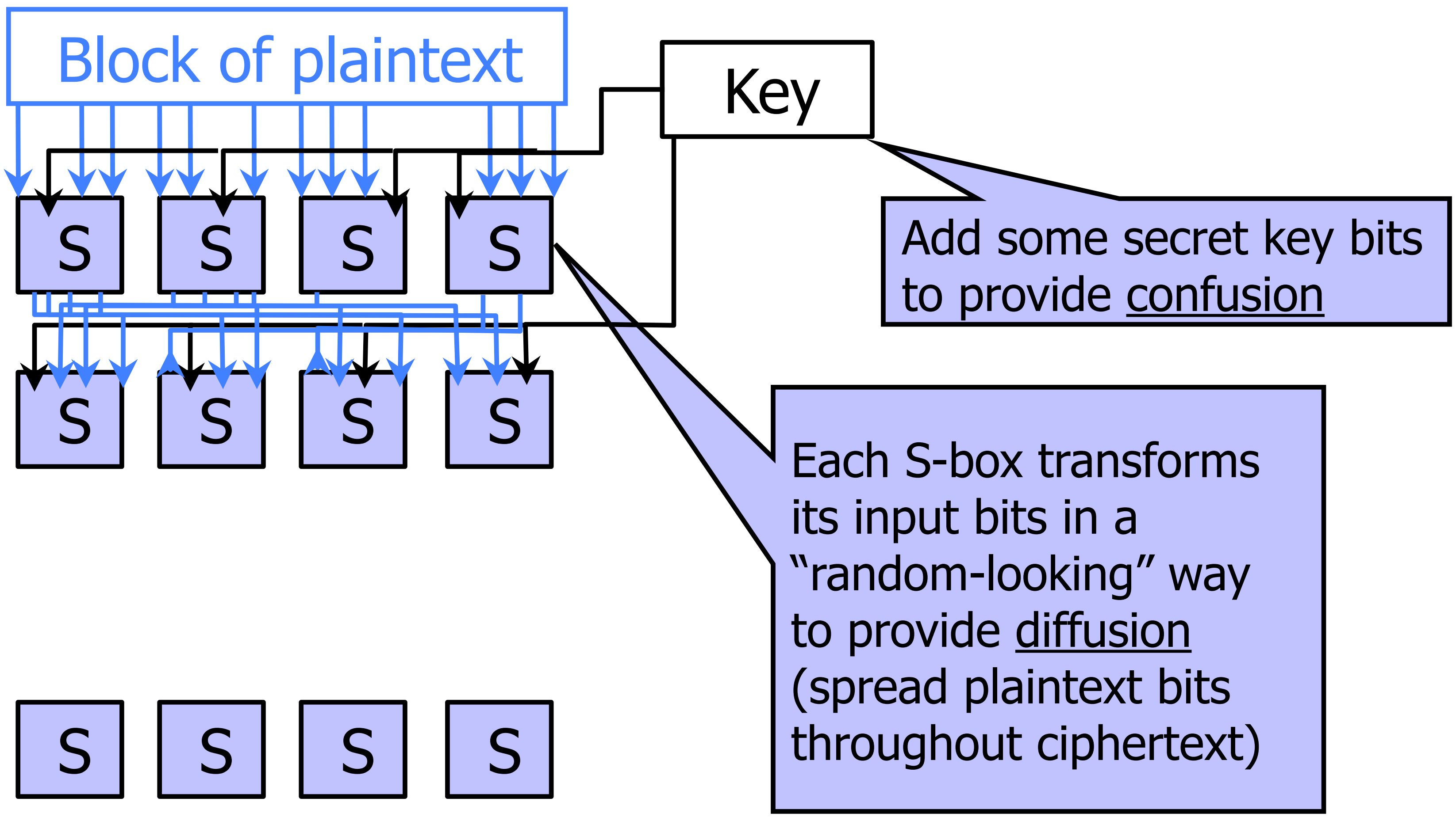
Block Cipher Operation (Simplified)



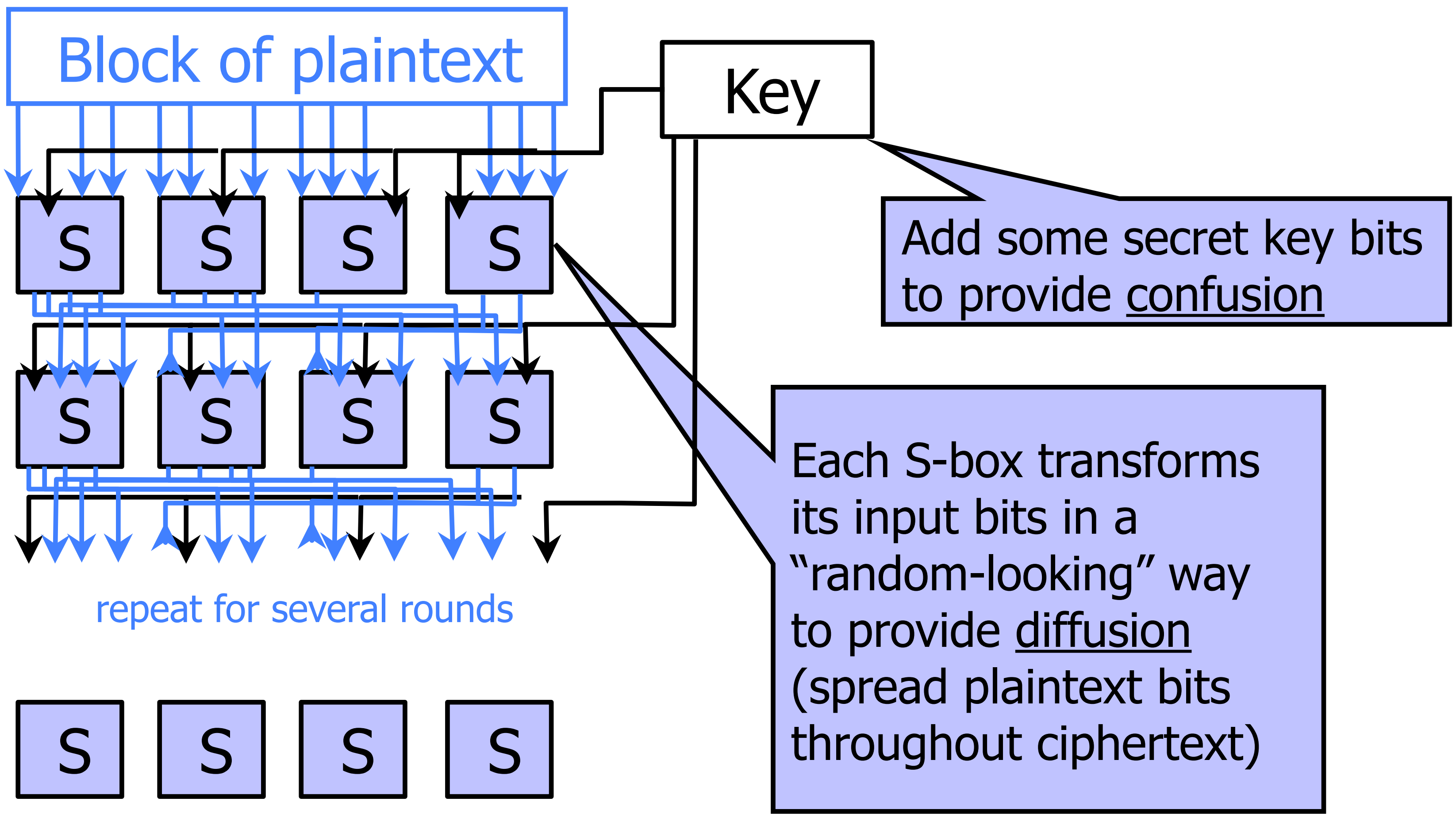
Block Cipher Operation (Simplified)



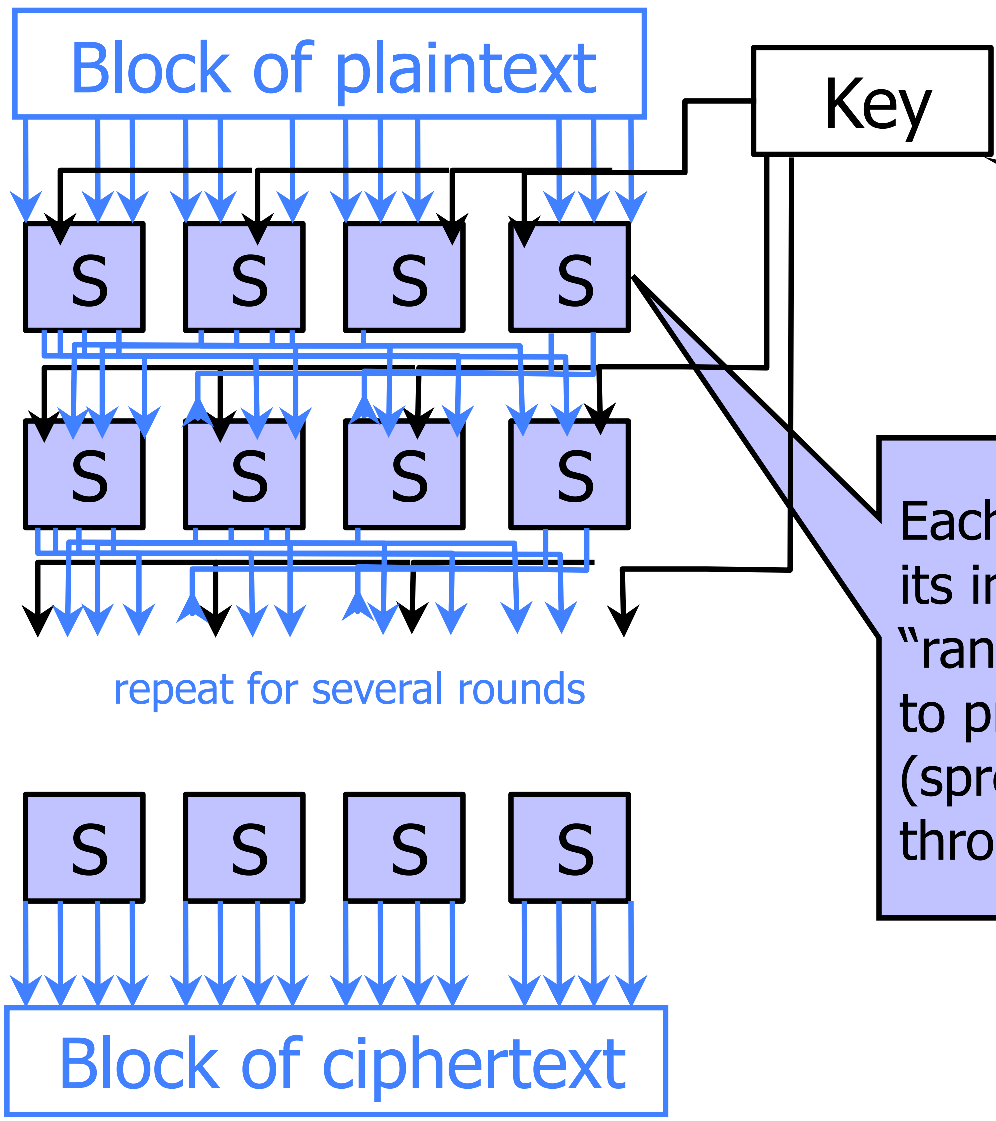
Block Cipher Operation (Simplified)



Block Cipher Operation (Simplified)



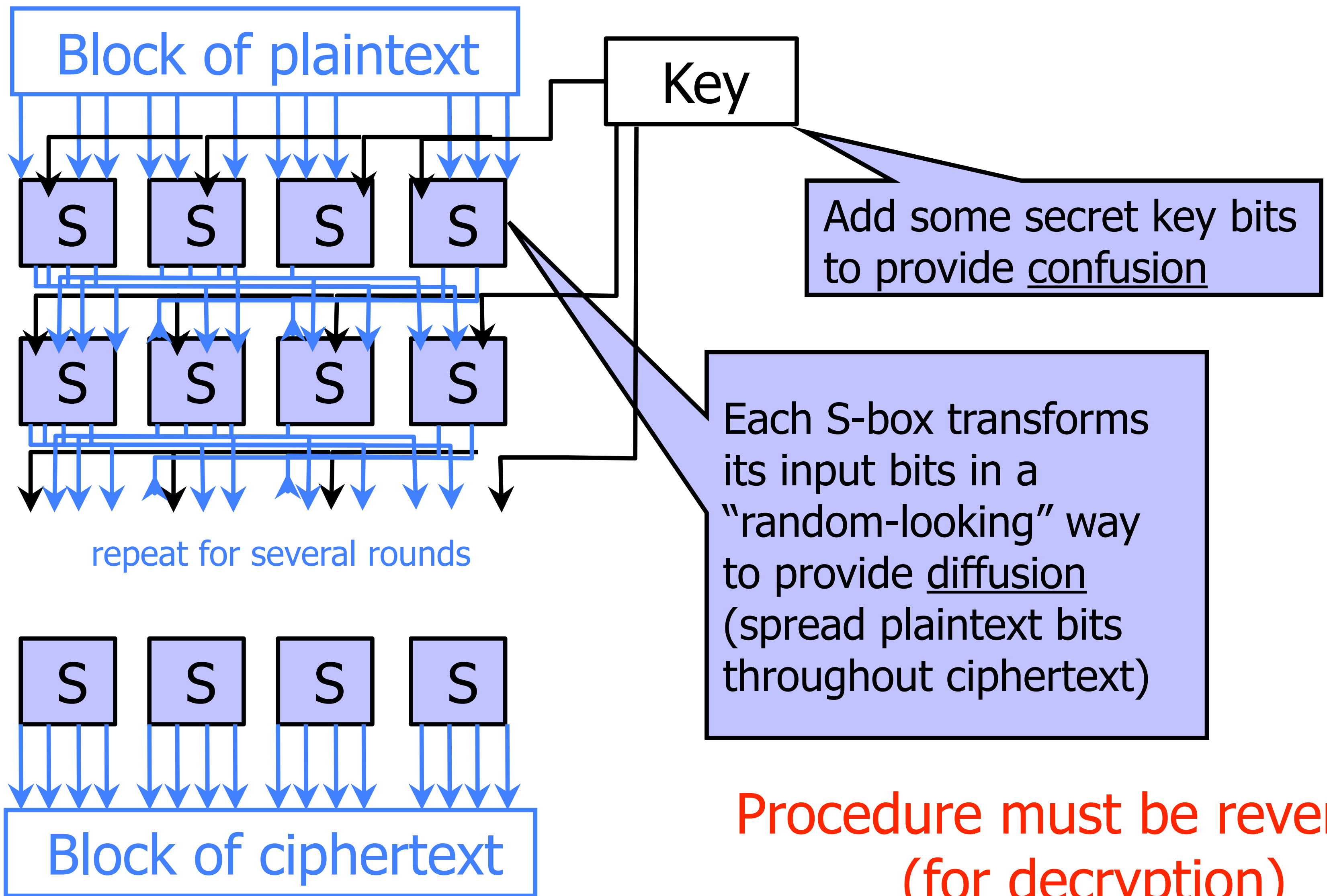
Block Cipher Operation (Simplified)



Add some secret key bits to provide confusion

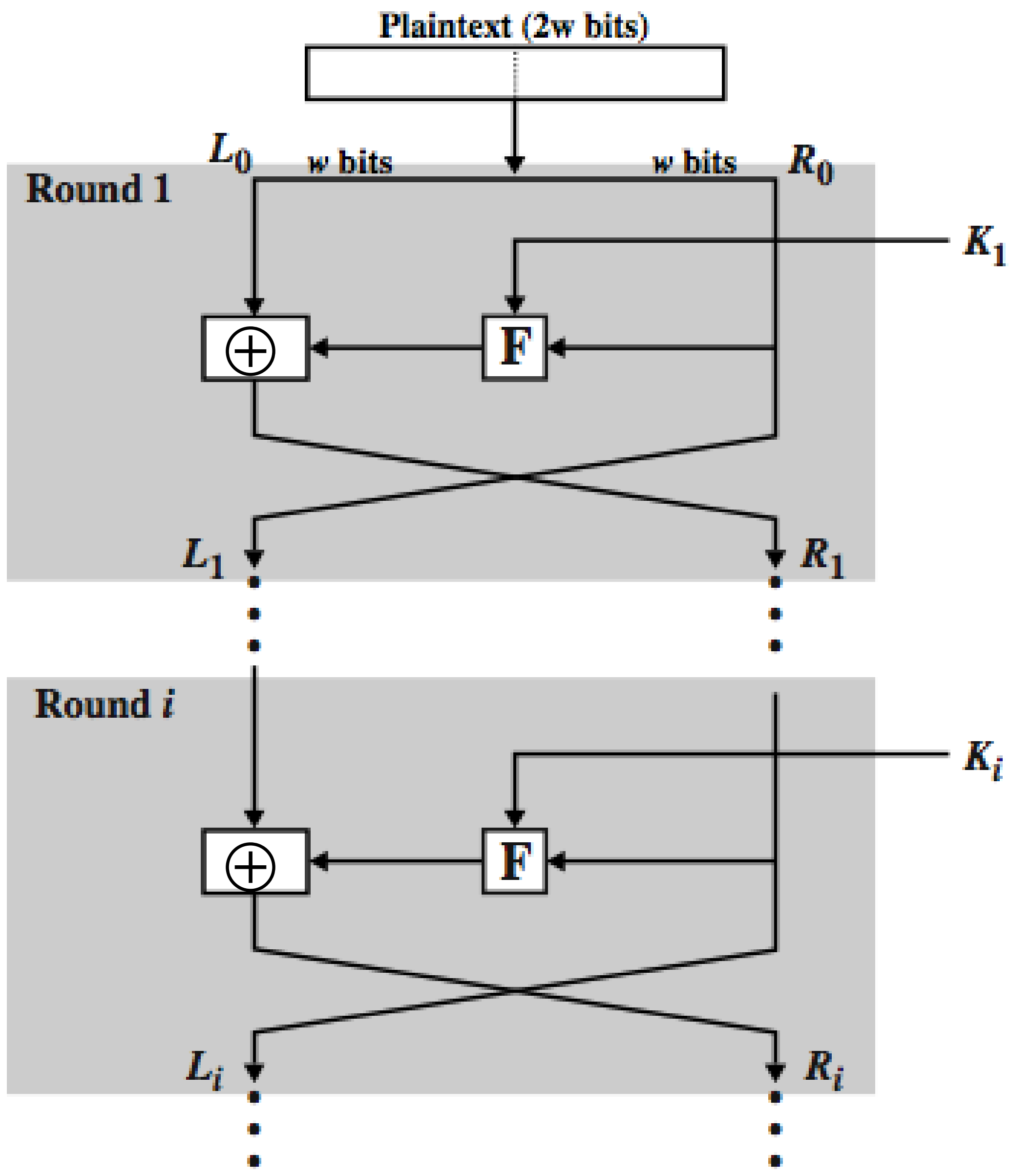
Each S-box transforms its input bits in a "random-looking" way to provide diffusion (spread plaintext bits throughout ciphertext)

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
  - 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 <u>not</u> use Feistel structure
  - The entire block is processed during each round
- Design uses some very nice mathematics