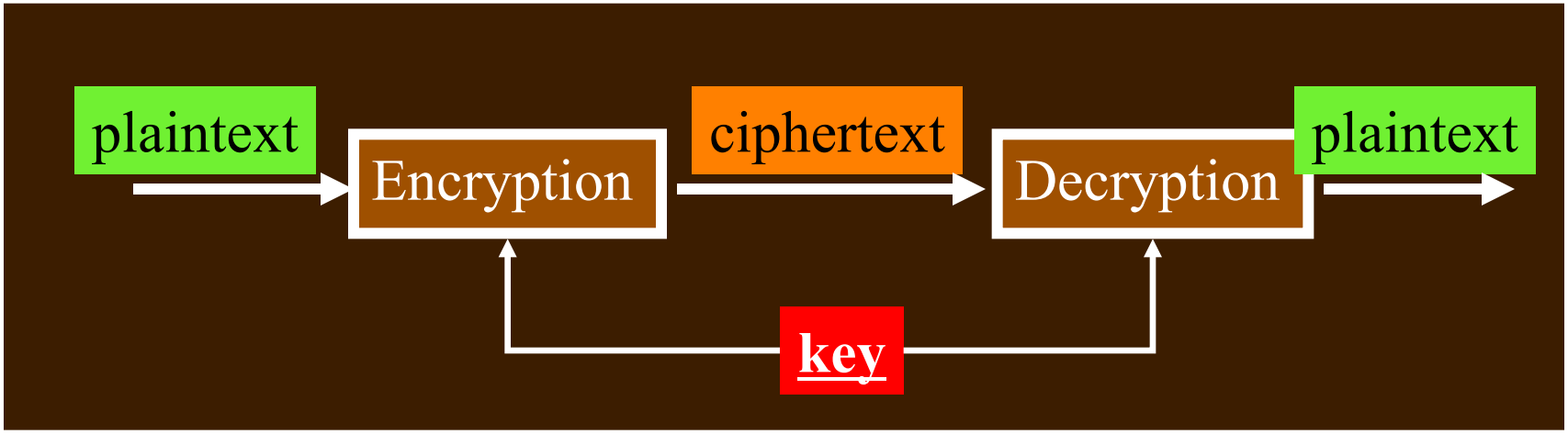
CS 4173/5173 COMPUTER SECURITY

Symmetric Cryptography in Practice

OUTLINE LAST TIME

- "Key" issues
  - Secret key cryptography
  - Public key cryptography
  - Hash
- Security Metrics
  - Perfect Security
  - Today's cryptographic systems do not achieve perfect security.
- P and NP problems
  - P: The set of questions that can be solved in polynomial time
  - NP: the set of questions for which an answer can be verified in polynomial time
  - Modern Cryptography: Without the key, the adversary must solve an NP problem, where a solution in P with polynomial time is "currently believed very hard" to find.

SECRET KEY CRYPTOGRAPHY



- Same key is used for both encryption and decryption
  - This one key is shared by two parties who wish to communicate securely
- Also known as *symmetric key* cryptography, or *shared key* cryptography

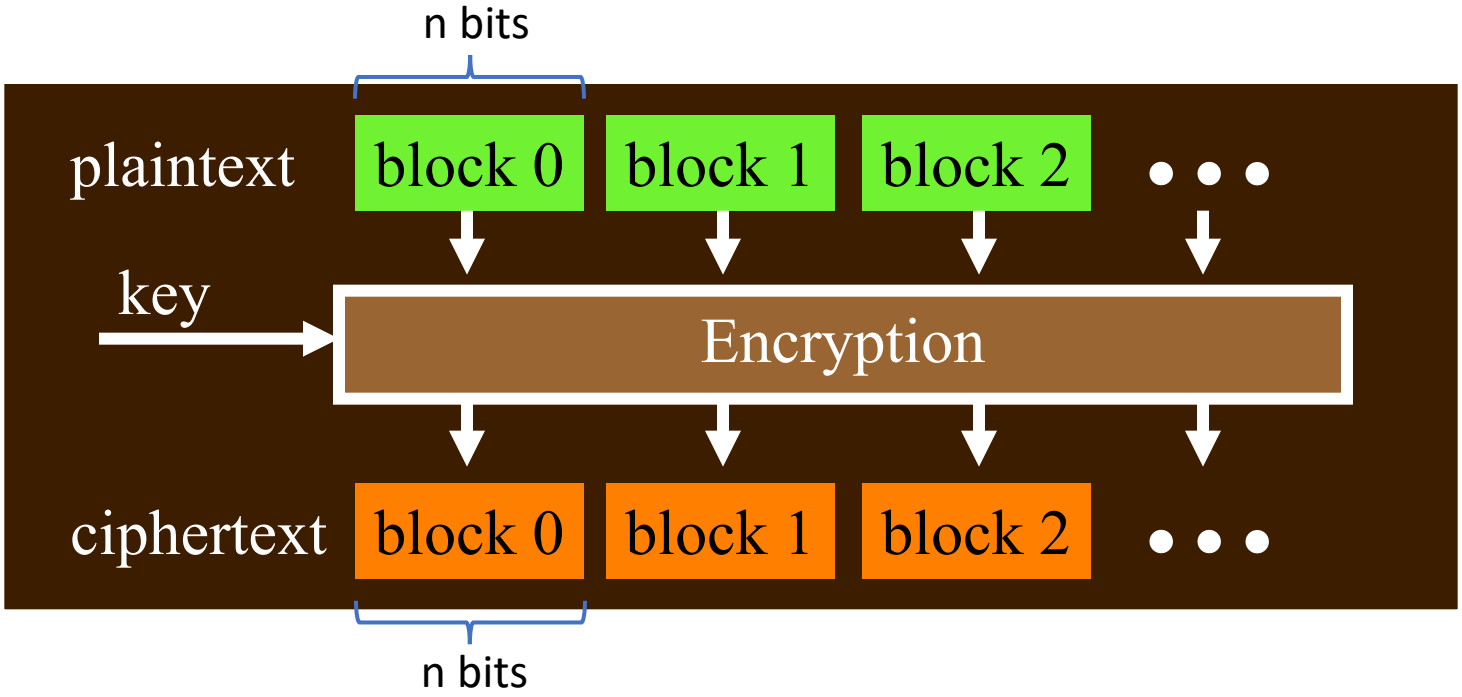
APPLICATIONS OF SECRET KEY CRYPTO

- Communicating securely over an insecure channel
  - Alice encrypts using shared key
  - Bob decrypts result using same shared key

- *Authentication*
  - Bob can verify if a message is generated by Alice

GENERIC BLOCK ENCRYPTION

- Converts one input plaintext block of fixed size *n* bits to an output ciphertext block also of *n* bits



KEY SIZES

- A Key should be selected from a large potential set to prevent brute force (exponential-time) attacks
  - If a key is of 3 bits, what are the possible keys?
    - 000, 001, 010, 011, 100, 101, 110,111
  - Given a pair of (plaintext, ciphertext), an attacker can do a brute force !

Question: If a key is of *n* bits, how many possible keys does a brute force attacker need to search?

KEY SIZES (CONT'D)

- Secret key sizes
  - 40 bits were considered adequate in 70's
  - 56 bits used by DES were adequate in the 80's
  - 128 bits are adequate for now
    - $2^{128} \approx 3.4\mathrm{E}+38$
    - If a computer can finish 1 billion searches per second, it requires 393845332084419517897424 3 years
  - 256 bits are also used now

KEY SIZES (CONT'D)

- Brute-force attack: try all combinations
  - NP problem: currently, the time increases exponentially as the key size increases.
- The computing power
  - Increases approximately linearly
- If computers increase in power by 40% per year, need roughly 5 more key bits per decade to stay "sufficiently" hard to break

SOME MOTIVATIONS

- How to design the ciphers?

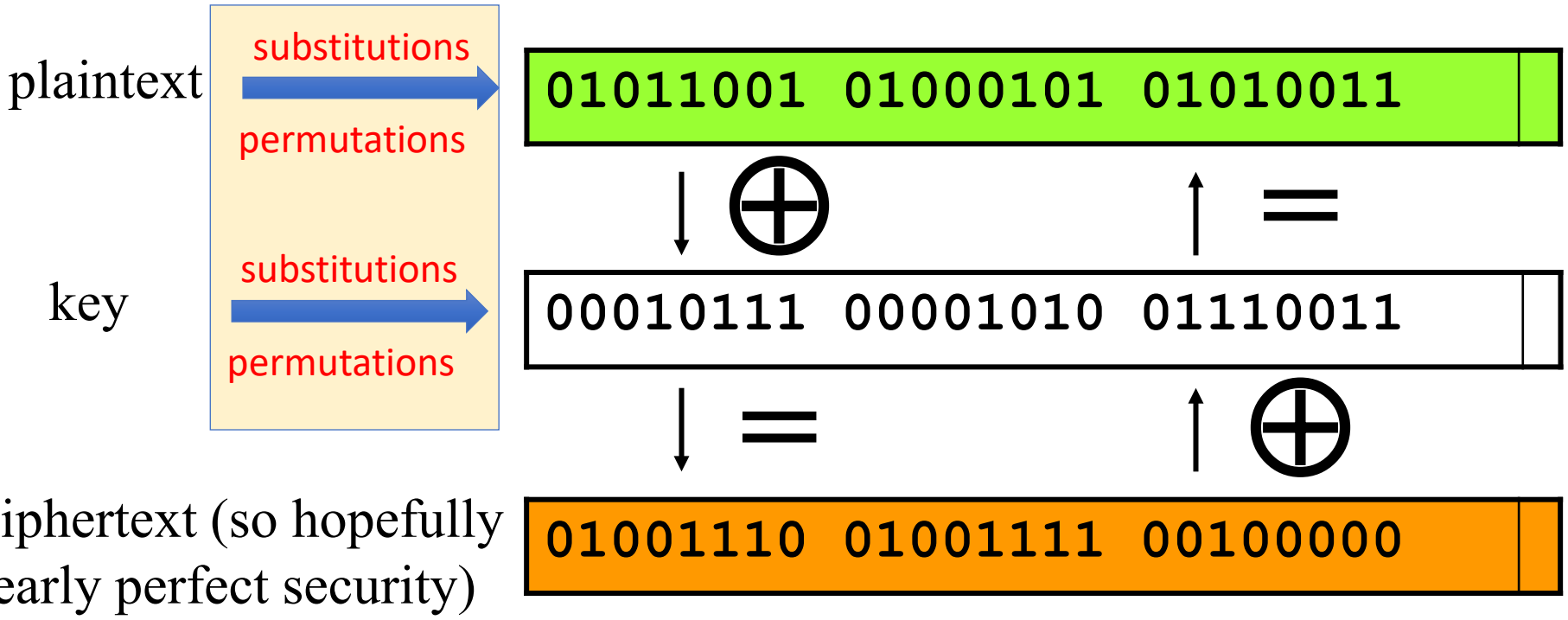
SOME MOTIVATIONS

- Early ciphers:
  - Using substitutions:
    - Ceasar variant, mono-alphabetic, Vigenere, …
  - Using permutations:
    - permutation ciphers

- One time pad:
  - theory-supported
  - Using XOR
    - plaintext XOR a random key

FOLLOW SHANNON'S GUIDELINE

make them pseudo random sequences



Q: Why also need to make the plaintext pseudo random?

PRINCIPLES FOR CIPHER DESIGN

- Confusion:
  - Make the relationship between the <plaintext, key> input and the <ciphertext> output as complex (non-linear) as possible
  - Confusion is mainly accomplished by **substitutions**

- Diffusion:
  - Spread the influence of each input bit across many output bits
  - Diffusion is mainly accomplished by **permutations**

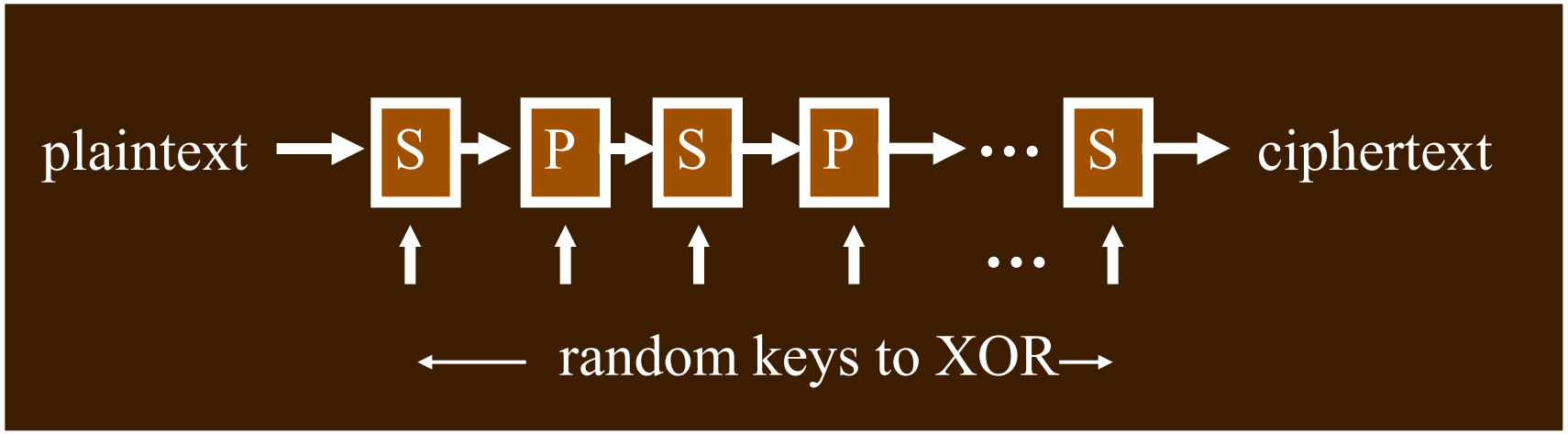
EXPLOITING THE PRINCIPLES

- Idea: use multiple, alternating Permutations and Substitutions, e.g.,
  - S→P→S→P→S→…
  - P→S→P→S→P→…

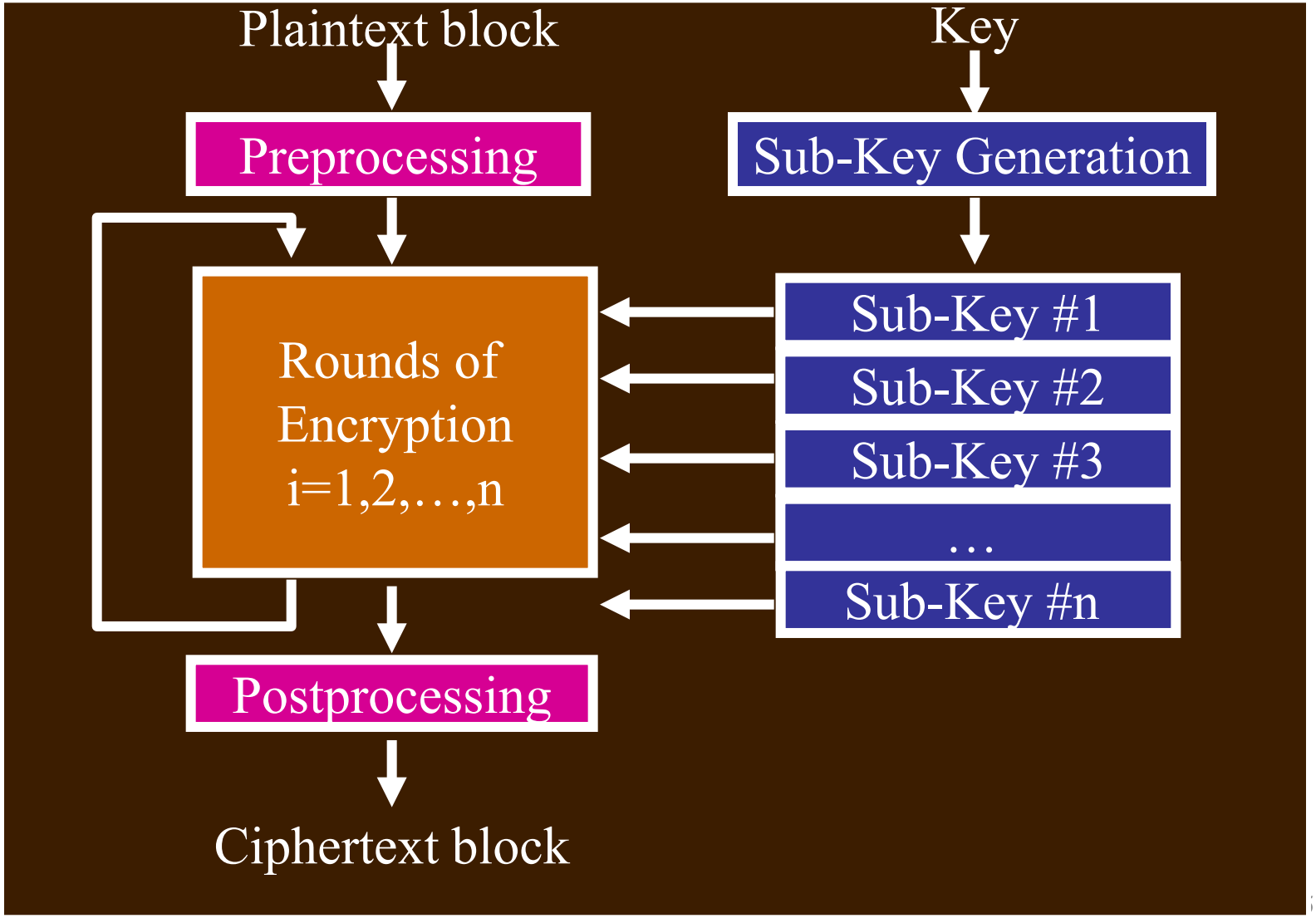
- Do P and S have to alternate? e.g.…
  - S→S→S→P→P→P→S→S→…?
  - Consecutive Ps or Ss do not improve security
  - Example?

SECRET KEY… (CONT'D)

- Basic technique used in secret key ciphers: multiple applications of alternating substitutions and permutations



BASIC FORM OF MODERN BLOCK CIPHERS



FEISTEL CIPHER

- Feistel Cipher has been a very influential “template” for designing a block cipher

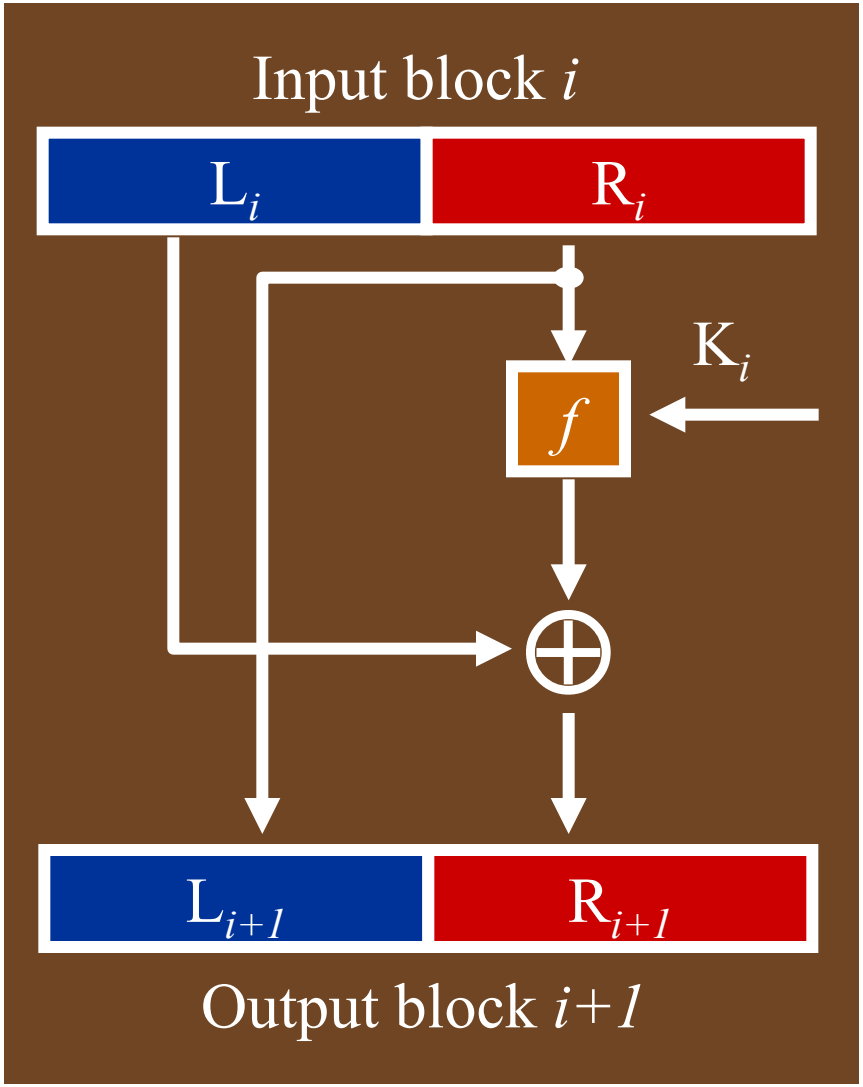
- Major benefit: Encryption and decryption take the same time
  - they can be performed on the same hardware

- Examples: DES, RC5

ONE "ROUND" OF FEISTEL ENCRYPTION

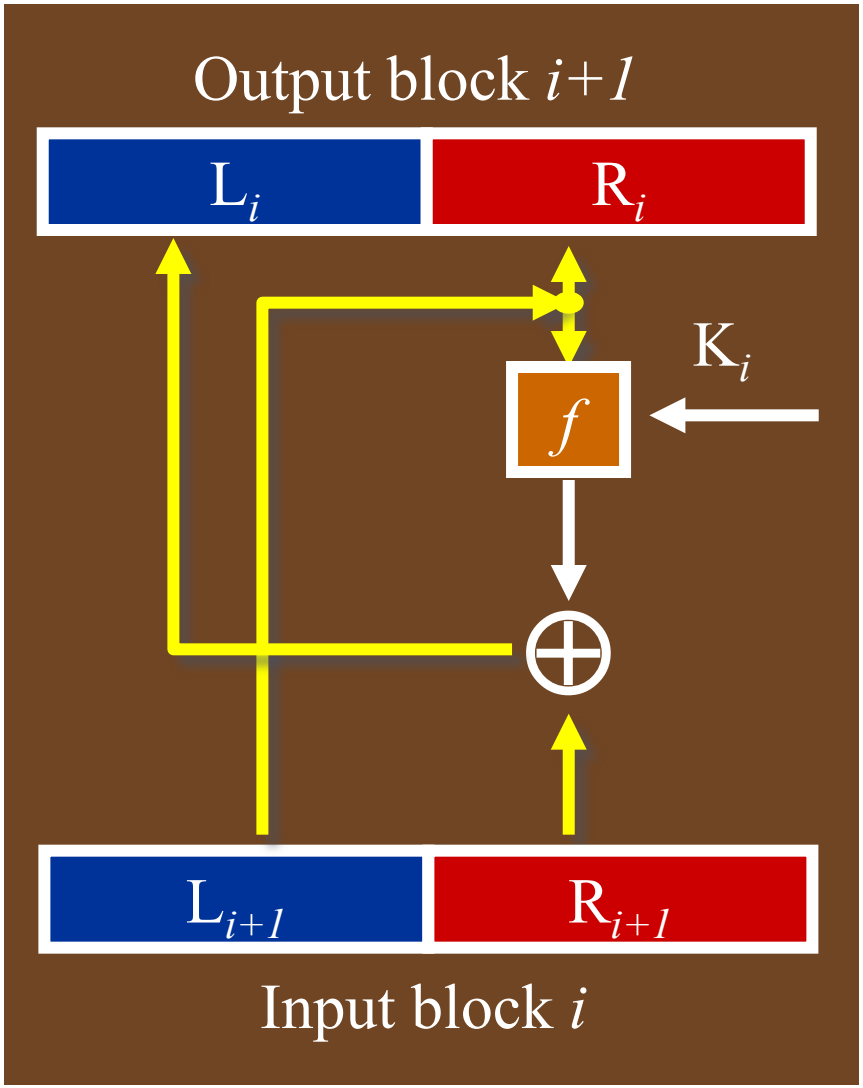
1. Break input block *i* into left and right halves L_i and R_i
2. Copy R_i to create output half block L_{i+1}
3. Half block R_i and key K_i are "scrambled" by function *f*
4. XOR result with input half-block L_i to create output half-block R_{i+1}

Key elements: substitution, XOR, and permutation



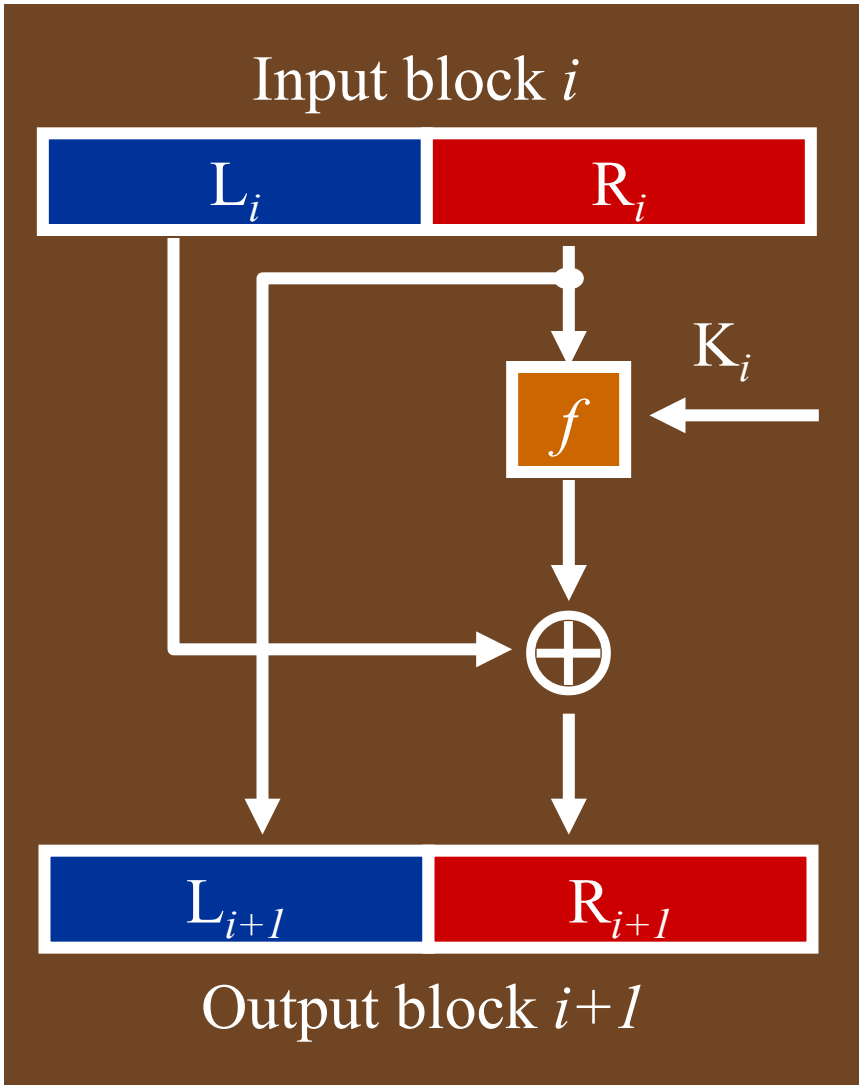
ONE "ROUND" OF FEISTEL DECRYPTION

- Just reverse the arrows!
- Why?



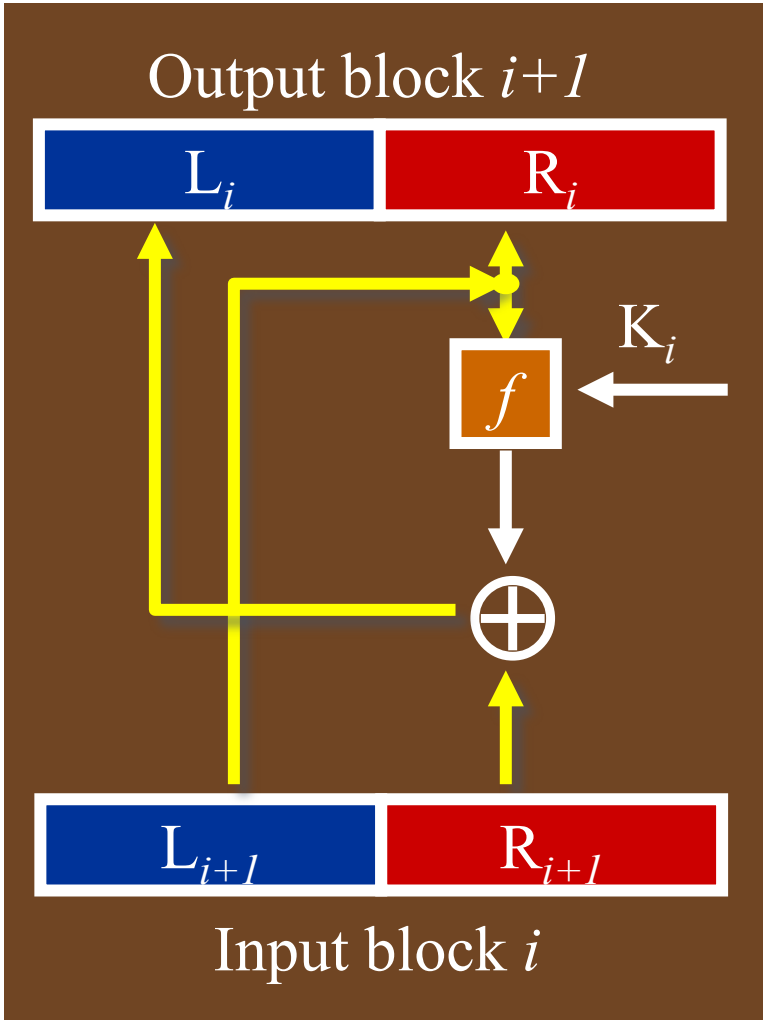
FEISTEL CIPHER: ENCRYPTION (CONT'D)

- Encryption:
- $L_{i+1} = R_i$
- $R_{i+1} = L_i \oplus f(R_i, K_i)$



FEISTEL CIPHER: DECRYPTION (CONT'D)

- Decryption:
- $\hat{R}_i = L_{i+1} = R_i$
- $\hat{L}_i = R_{i+1} \oplus f(R_i, K_i)$
  $= L_i \oplus f(R_i, K_i) \oplus f(R_i, K_i)$
  $= L_i \oplus$ all 0 bits $= L_i$



EXERCISE

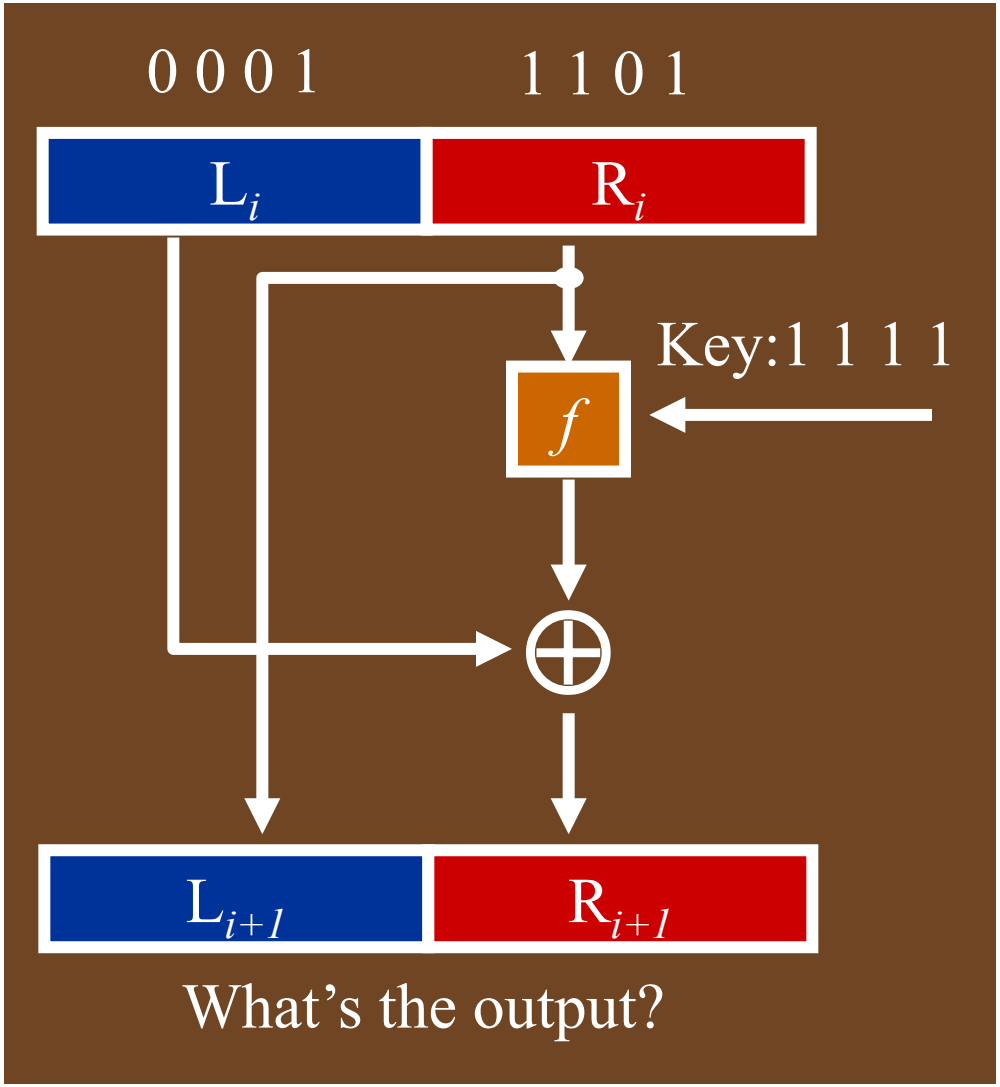
Key generation:

subkey is the key

Scrambling function

$f(K, R_i) = K \oplus R_i$

Q: What's the output?



EXERCISE

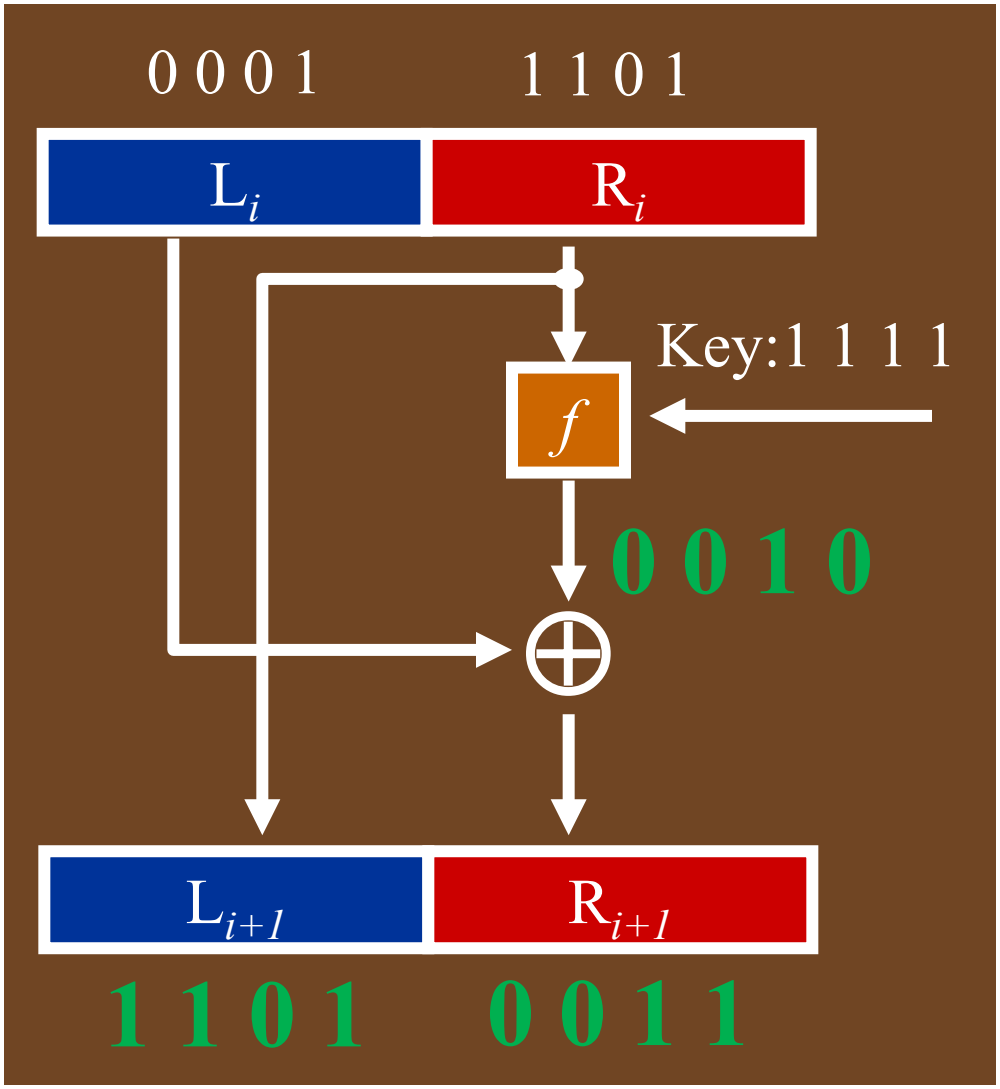
Key generation:

subkey is the key

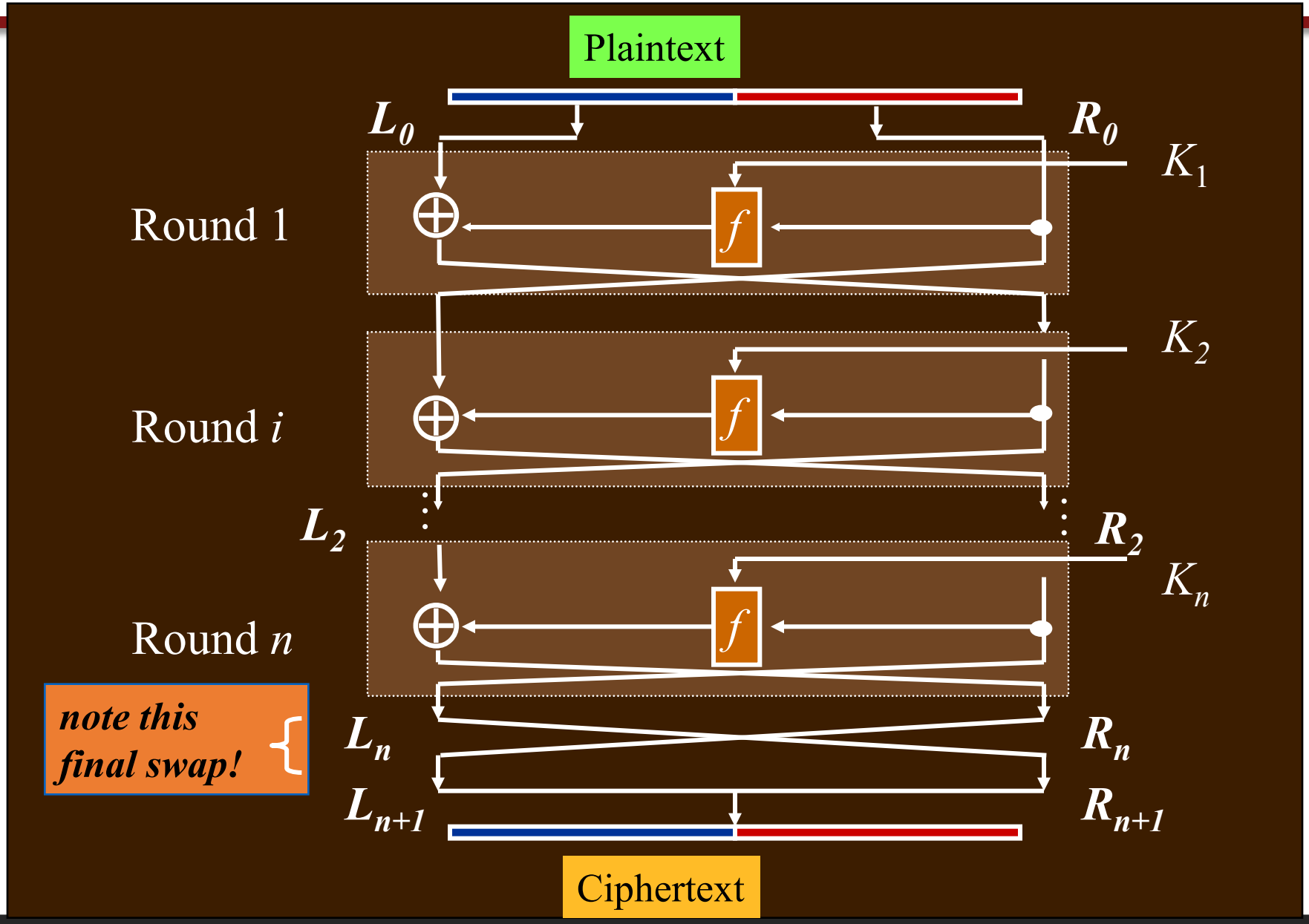
Scrambling function

$f(K, R_i) = K \oplus R_i$

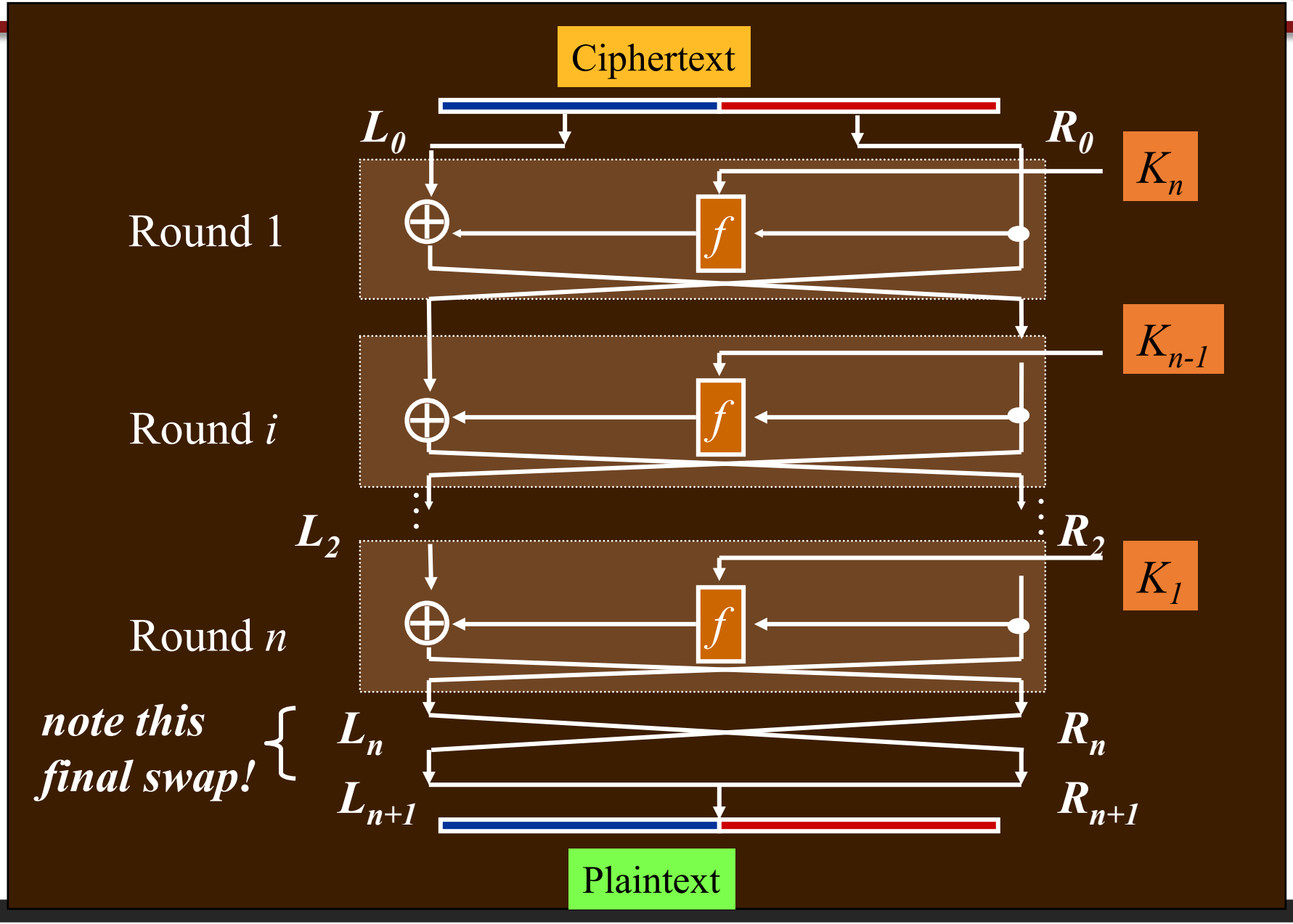
Q: What's the output?



COMPLETE FEISTEL CIPHER: ENCRYPTION



COMPLETE FEISTEL CIPHER: DECRYPTION



PARAMETERS OF A FEISTEL CIPHER

- Block size
- Key size
- Number of rounds
- Subkey generation algorithm
- “Scrambling” function *f*