

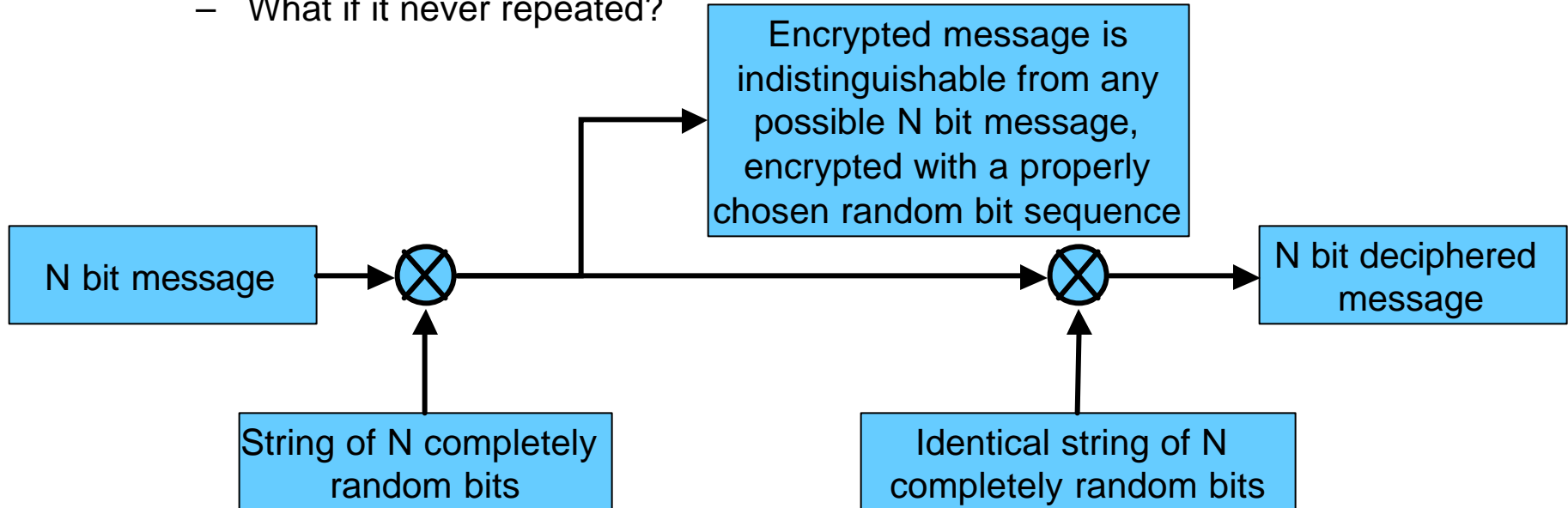
# Week 4: More Security Topics

# Evolution of Cryptography

- Monoalphabetic substitution, e.g.,
  - Caesar cipher {a,b,c,d,e,f,...,x,y,z} -> {b,c,d,e,f,g,...,y,z,a}
  - Atbash cipher {a,b,c,d,e,f,...,x,y,z} -> {z,y,x,w,v,u,t,s,r,q,p,o,n,m,l,k,j,i,h,g,f,e,d,c,b,a}
  - Any permutation of the alphabet
  - Easily solved by observing single and double letter frequencies
  - English (like most other non-ideograph languages) have distinct letter frequencies over a small alphabet.
  - Encoding English letters requires  $\log_2(26) \sim 4.7$  bits/letter,
  - but information content in English text is
$$\sum p \log_2(p)$$
With unequal letter probabilities, actual information content is much lower.  
Equivocation of source is the effective information content
- Polyalphabetic substitution:
  - thisisamessagetobeencrypted** - plaintext
  - badbadbadbadbadbadbadbadbad** - key stream
  - vimujwcniteifxqcigogtztvfh** - ciphertext
  - Correlation-like techniques find the length of the key stream, k
  - Problem then reduces to solving k monoalphabetic ciphers
  - Using running text (e.g., from an agreed to book) makes solution harder, but with enough ciphertext, both the plaintext as well as the key stream are easily found

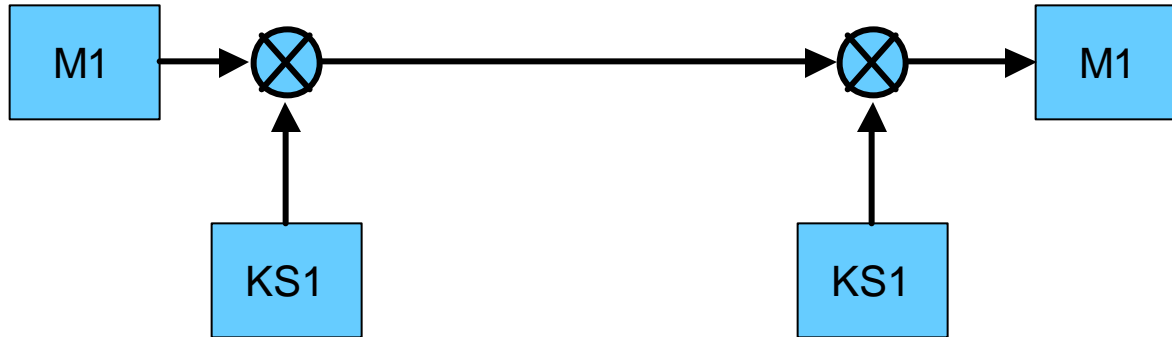
## Evolution of Cryptography - 2

- Weakness of polyalphabetic cipher is repetition of the key stream
  - What if it never repeated?

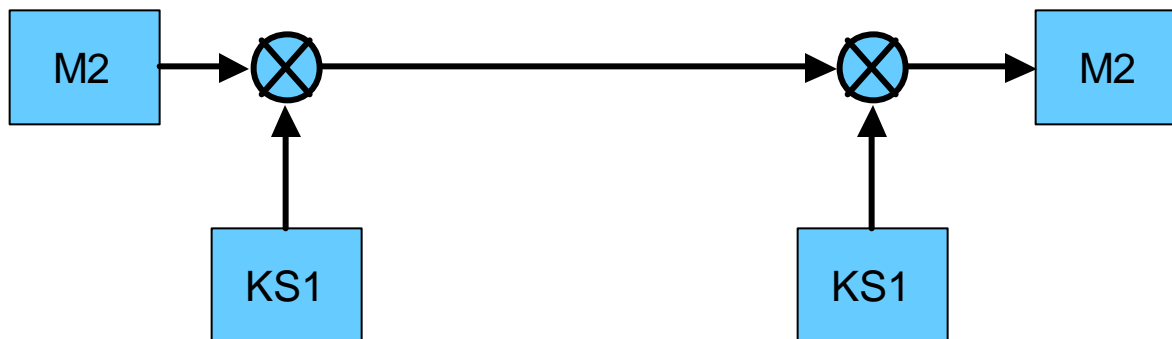


- One-time-pad is the only provably secure cryptographic system
  - What happens if key sequence is (accidentally) reused?

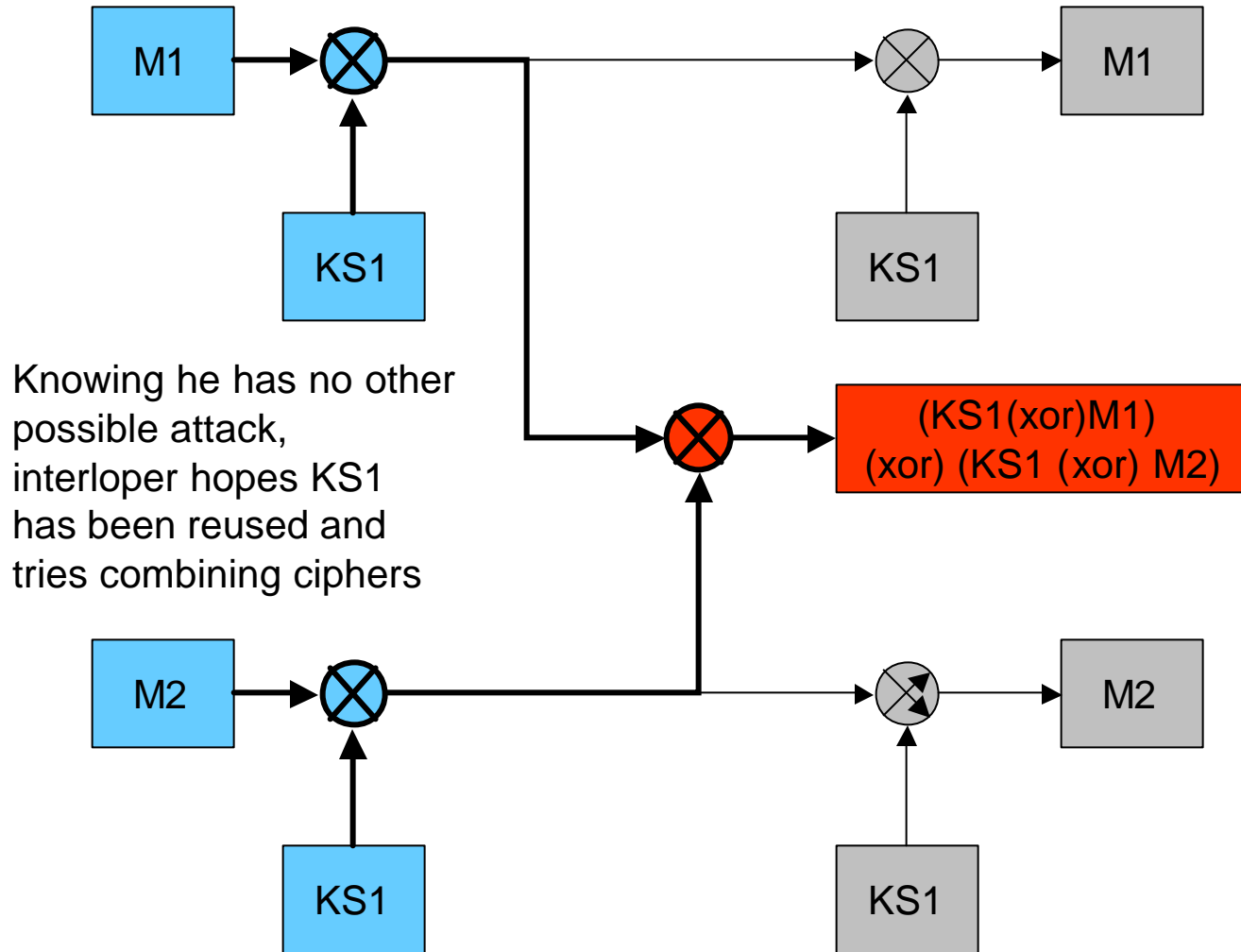
# One-bit-pad Key Reuse



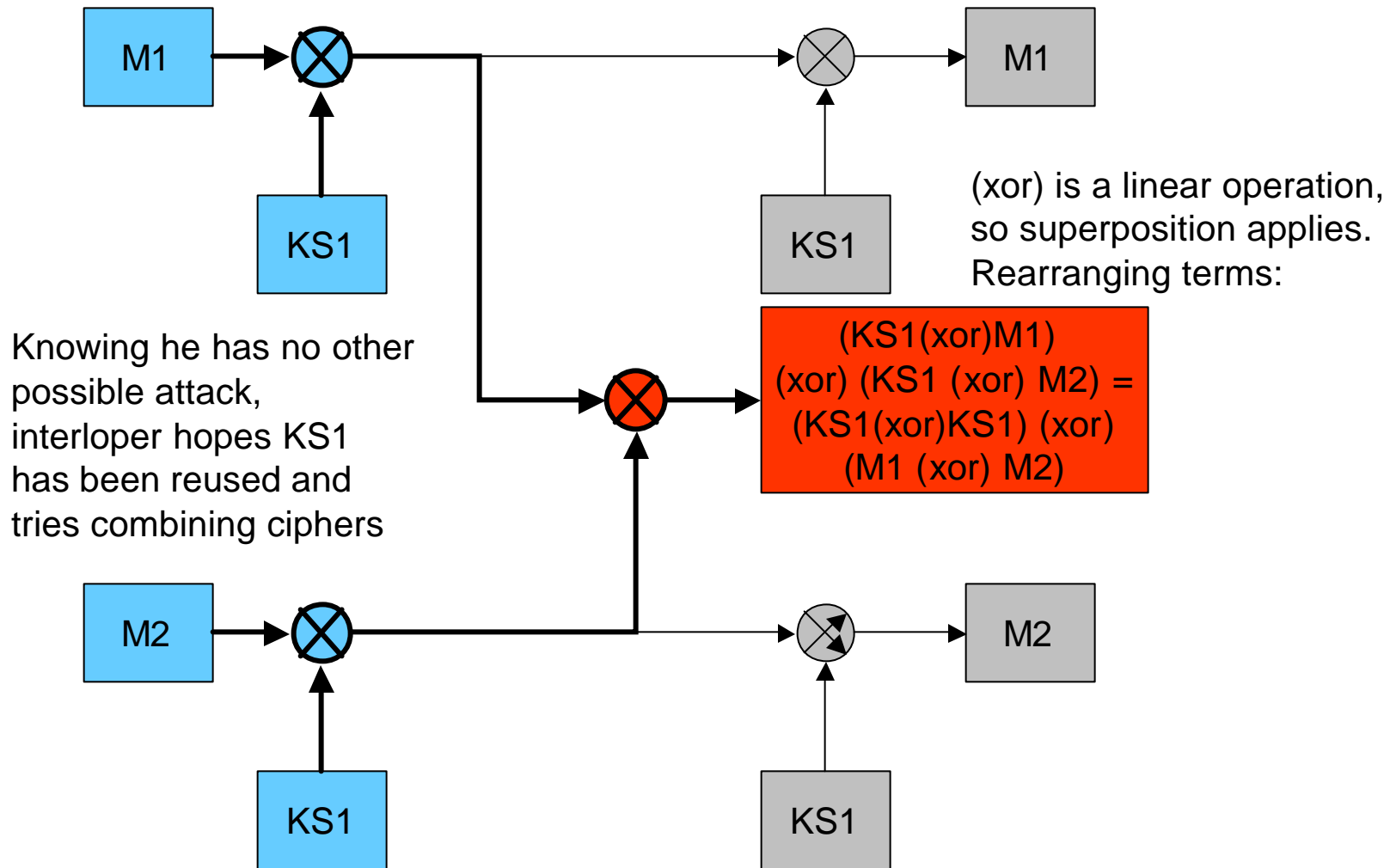
Sender (or receiver) accidentally sent M2, reusing KS1, previously used for M1



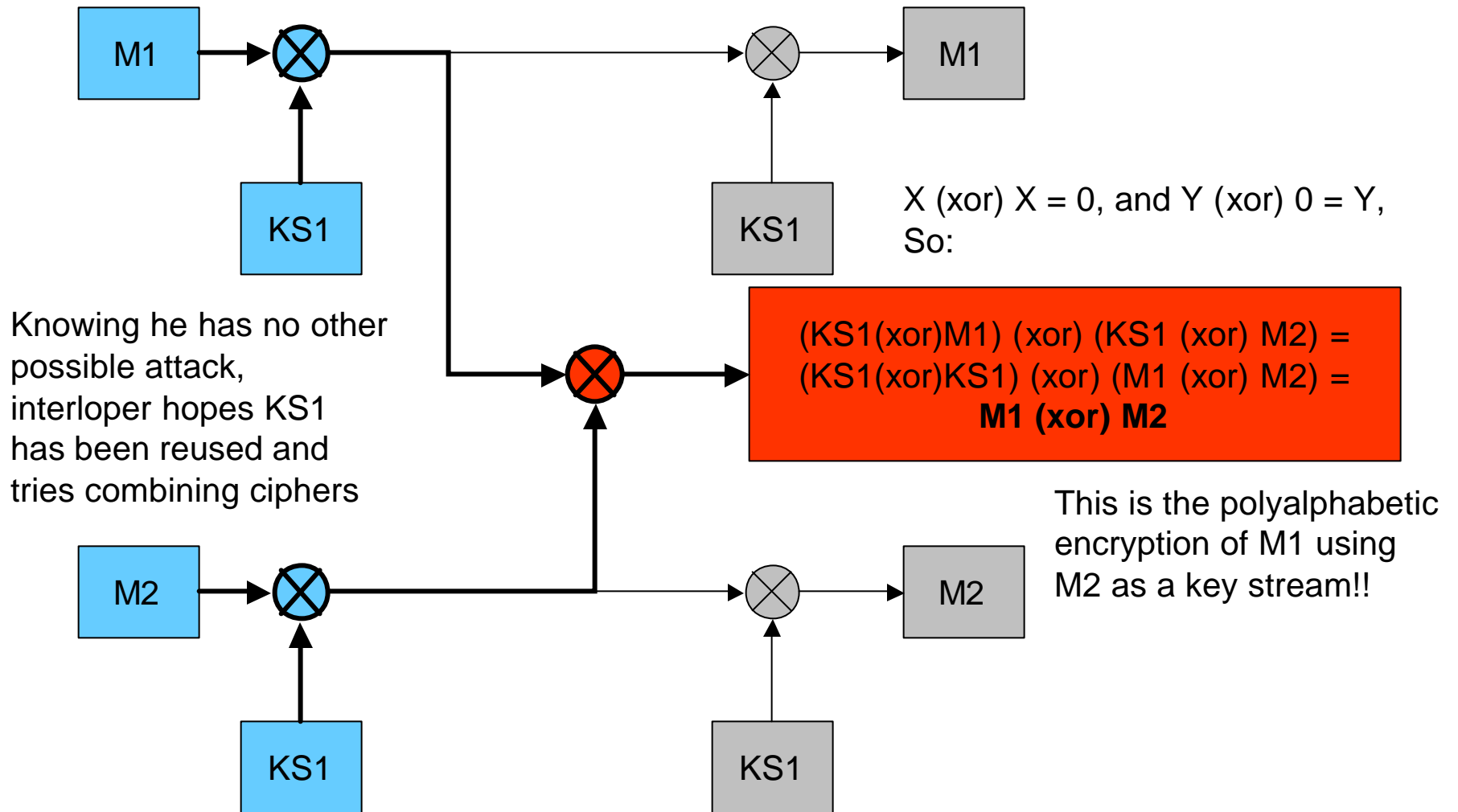
# One-bit-pad Key Reuse



# One-bit-pad Key Reuse

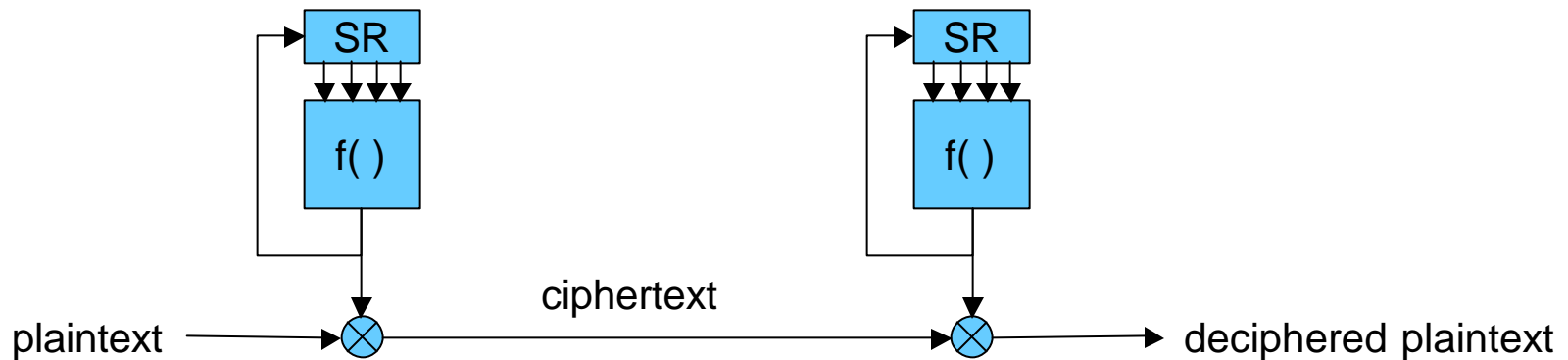
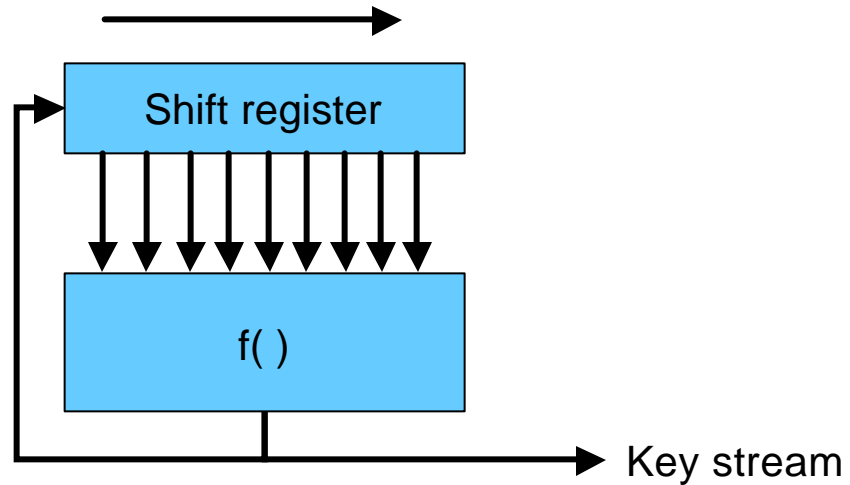


# One-bit-pad Key Reuse



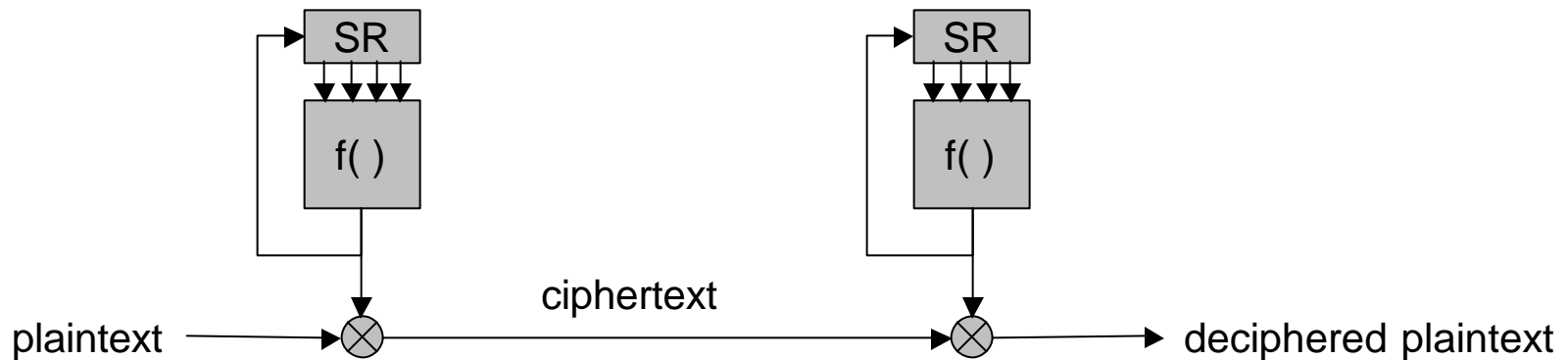
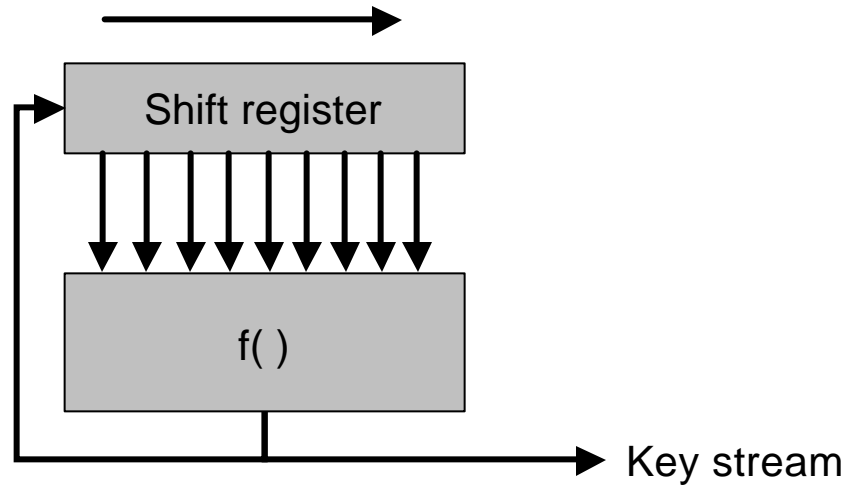
# Generating Long Pseudo-Random Sequences

- For N bit register, if  $f(x)$  is linear,  $2^N - 1$  bits of key stream are sufficient to find  $f()$
- So,  $f()$  must be nonlinear



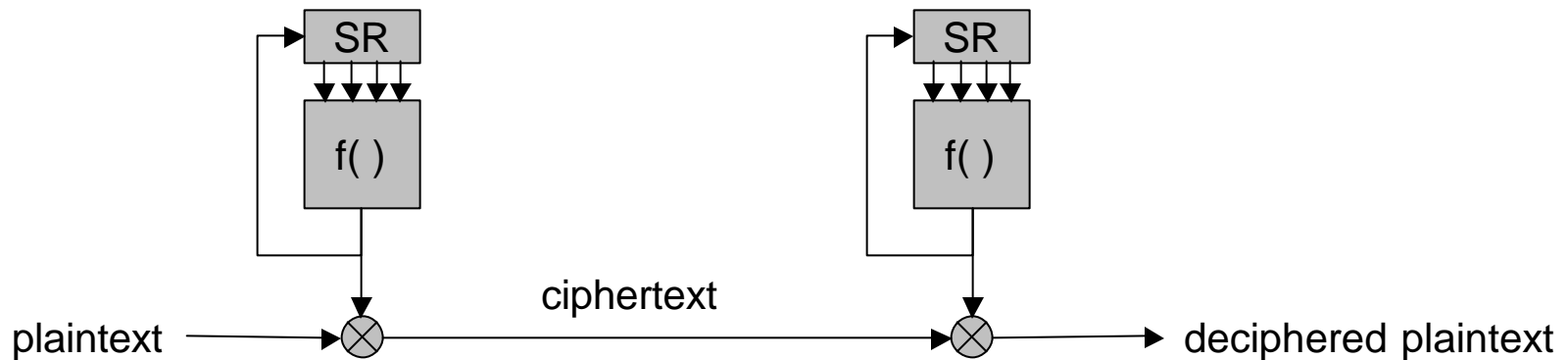
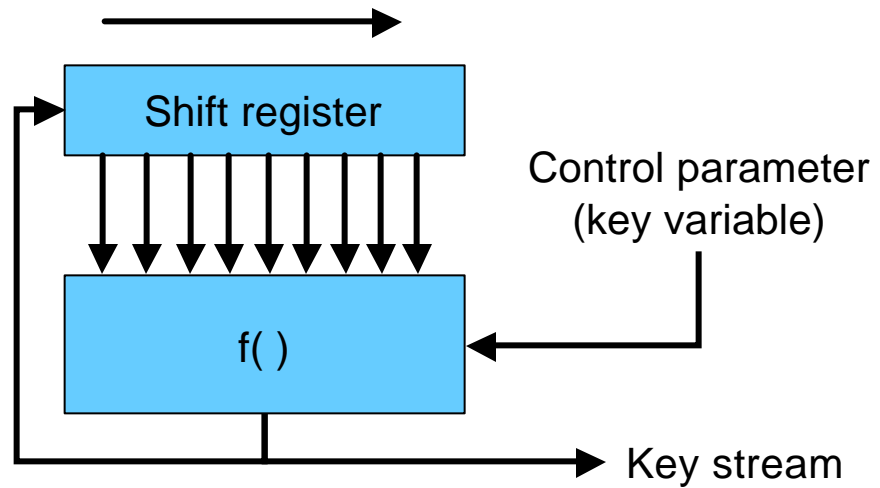
# Generating Long Pseudo-Random Sequences

- How to make  $f()$  easy to change?
- What about synchronization?



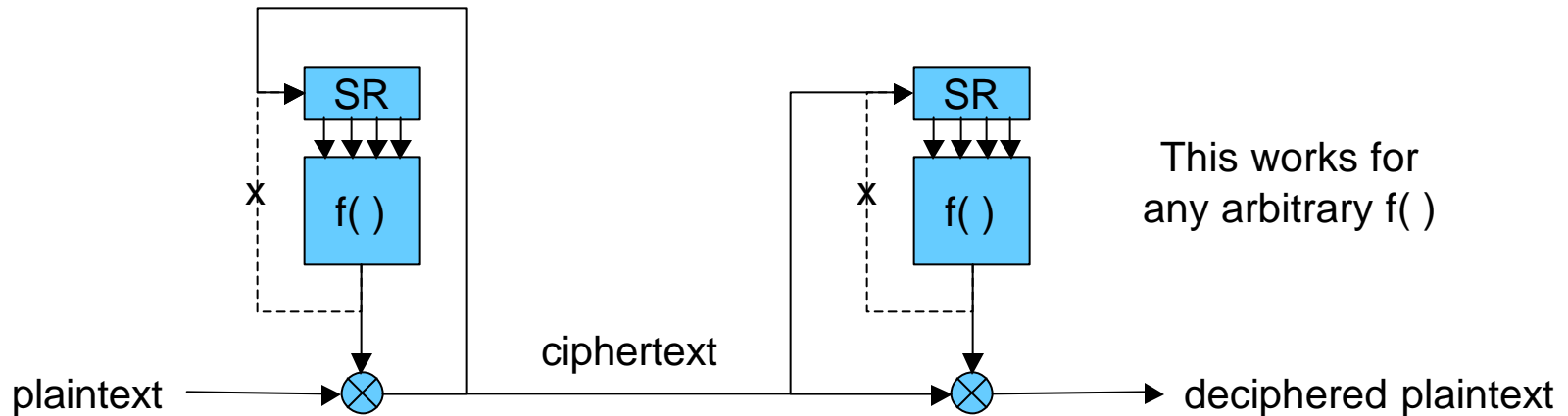
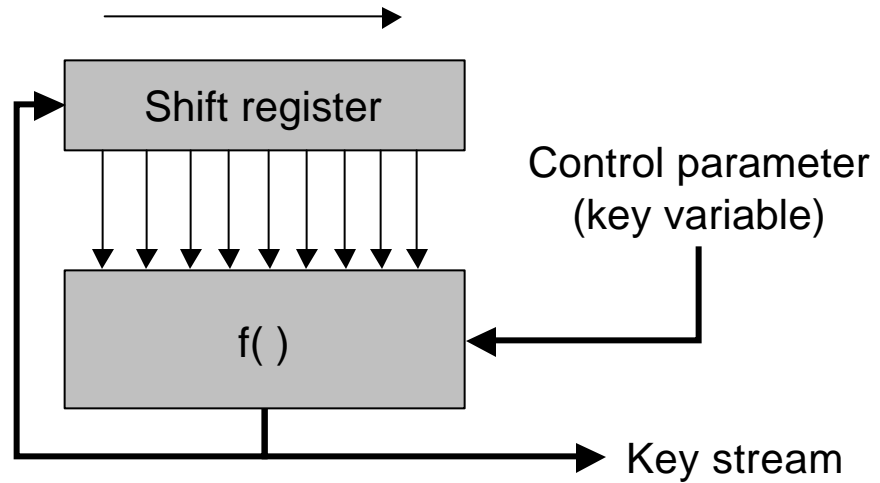
# Generating Long Pseudo-Random Sequences

- How to make  $f()$  easy to change?
- What about synchronization?

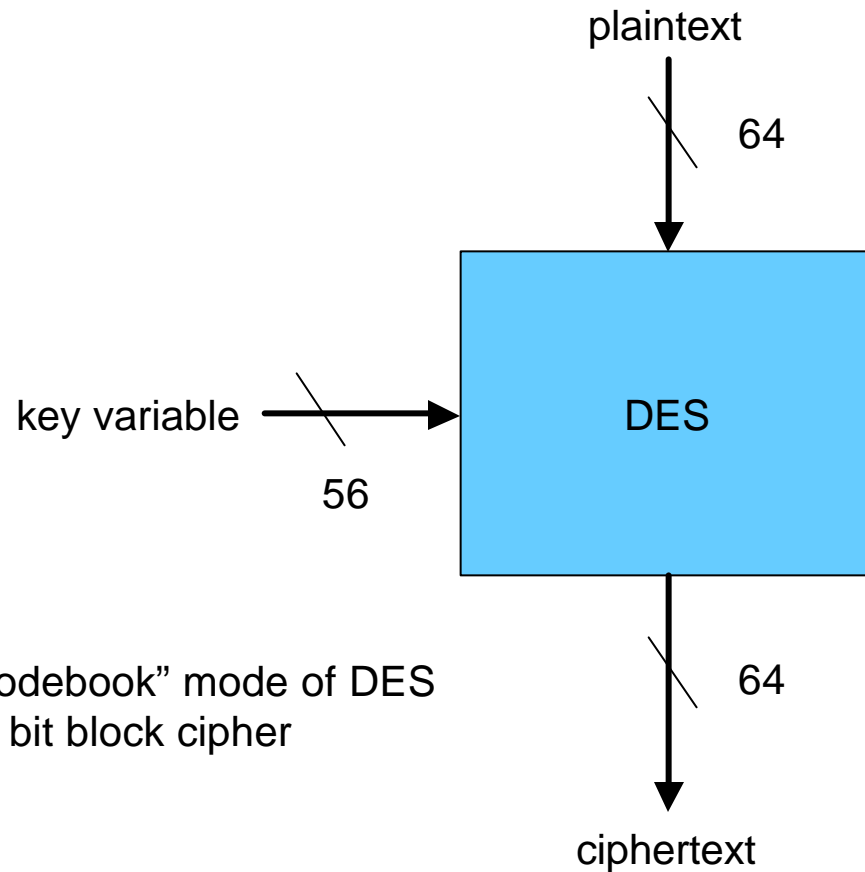


# Generating Long Pseudo-Random Sequences

- How to make  $f()$  easy to change?
- What about synchronization?

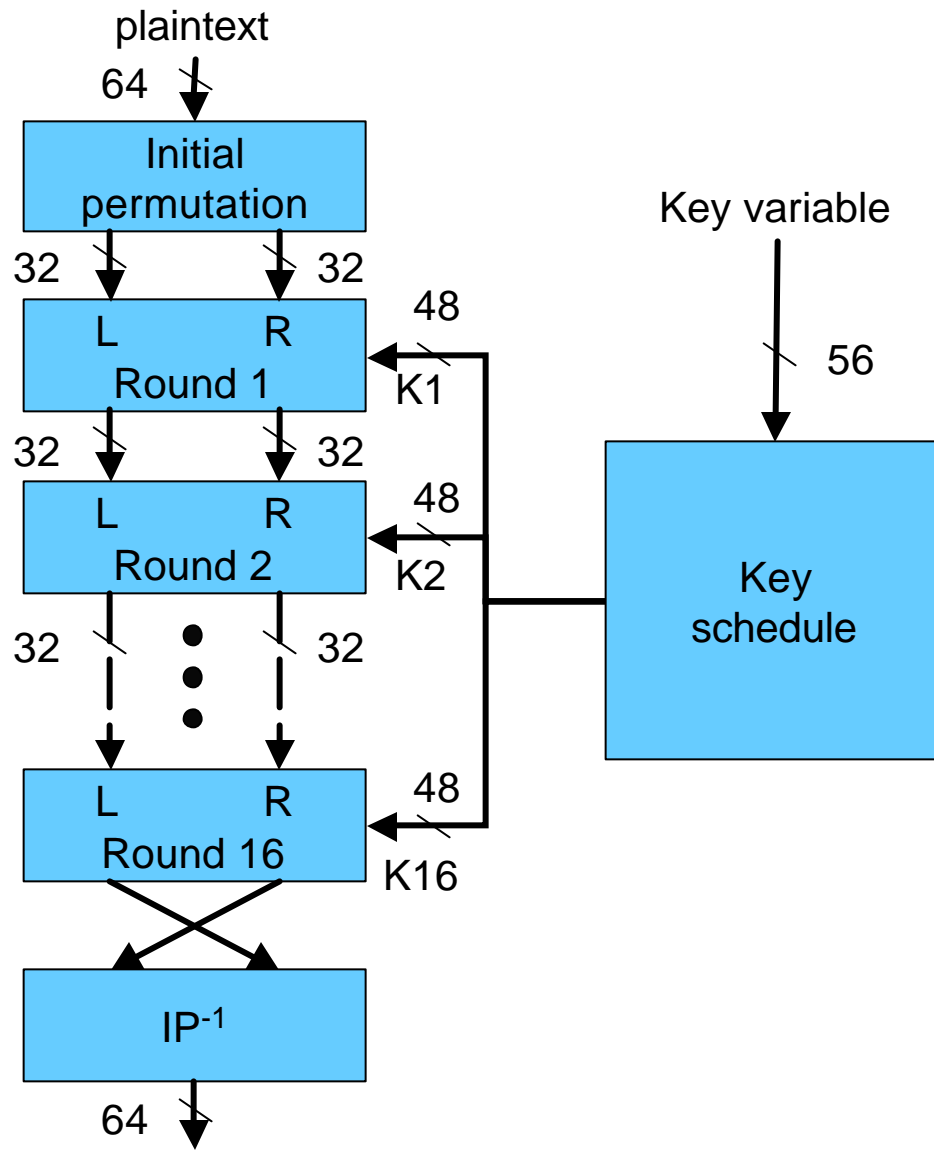


# DES as one $f()$ option

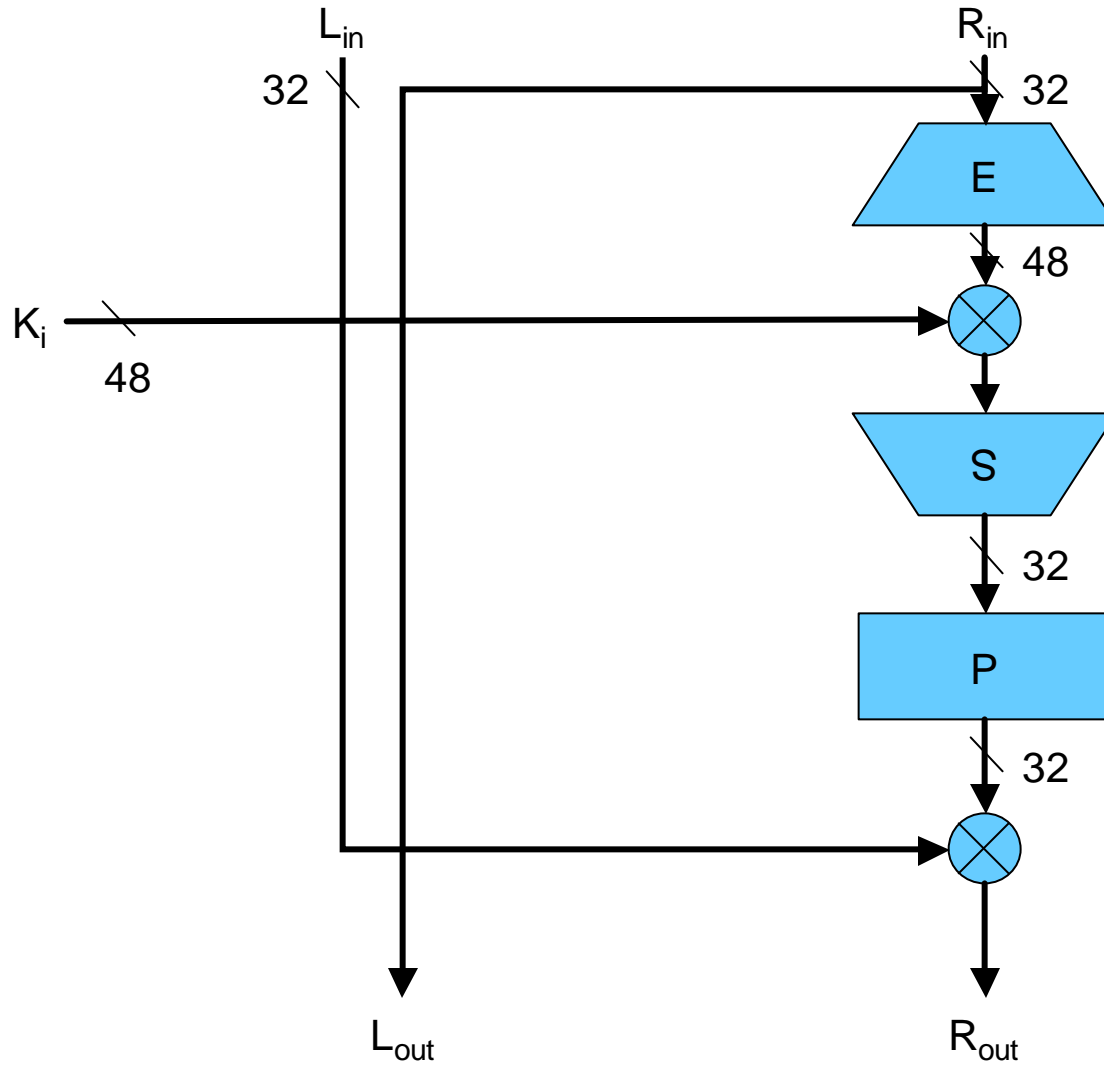


“Electronic codebook” mode of DES  
– 64 bit block cipher

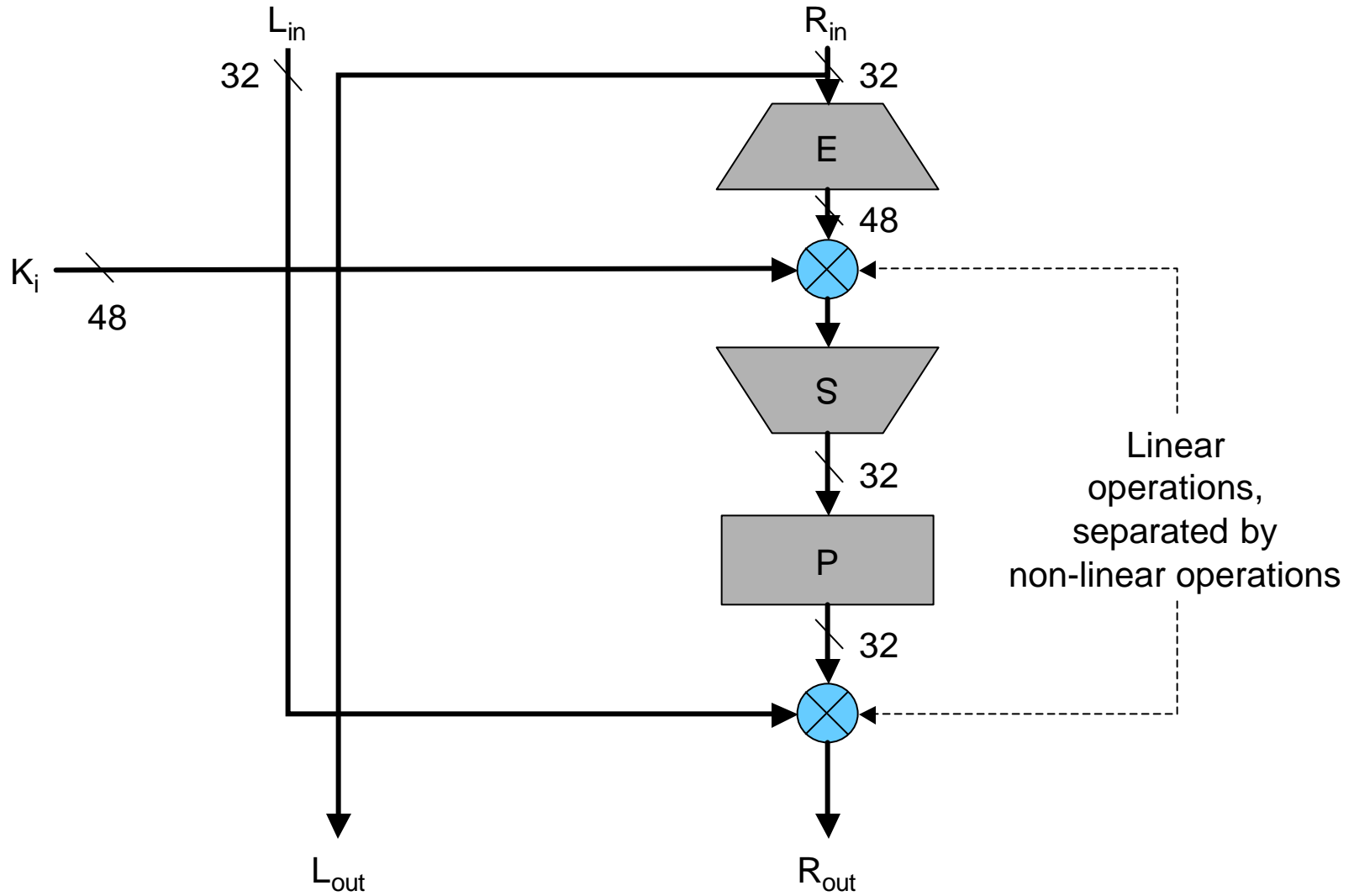
# Internal operation of DES



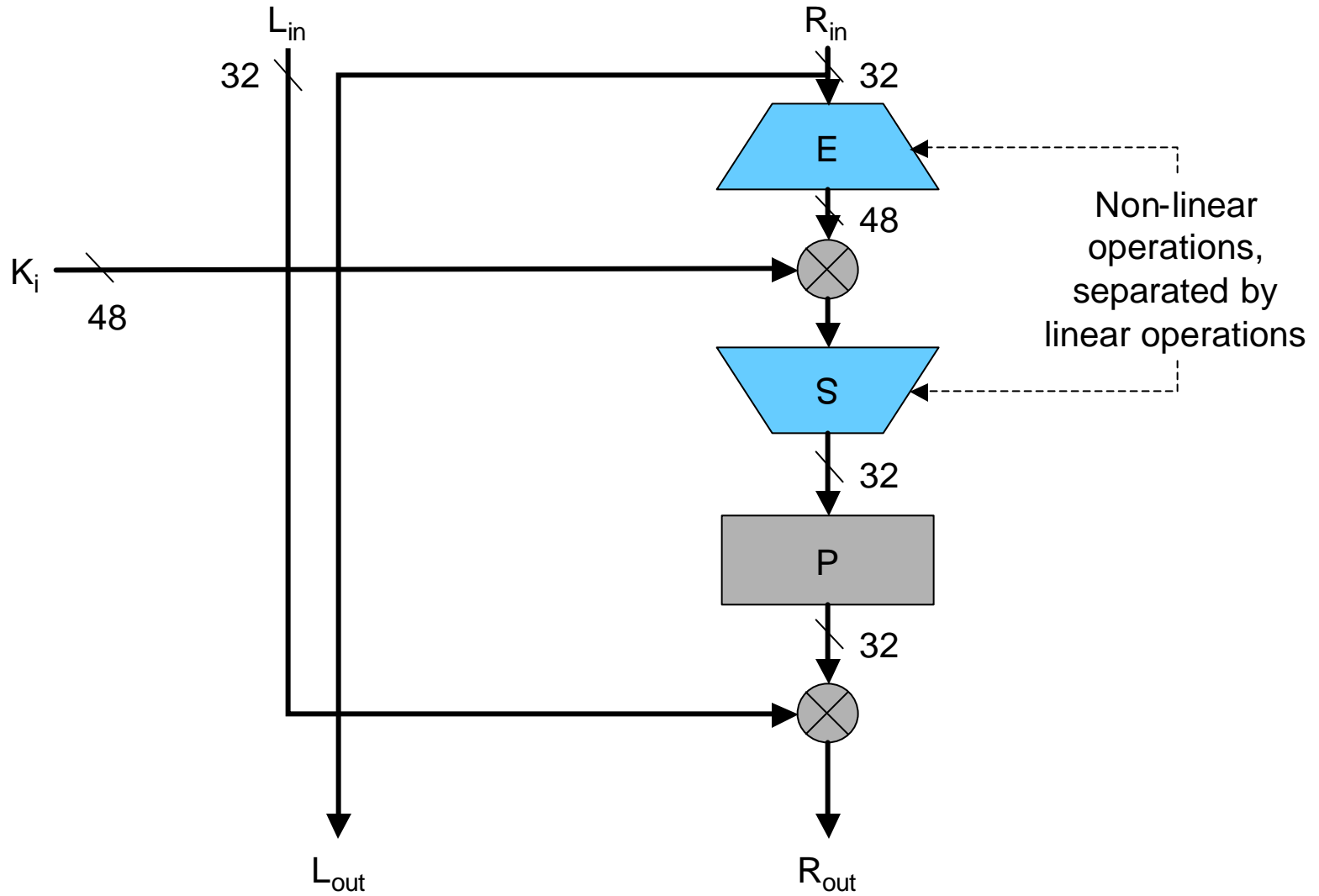
# DES Round<sub>i</sub>



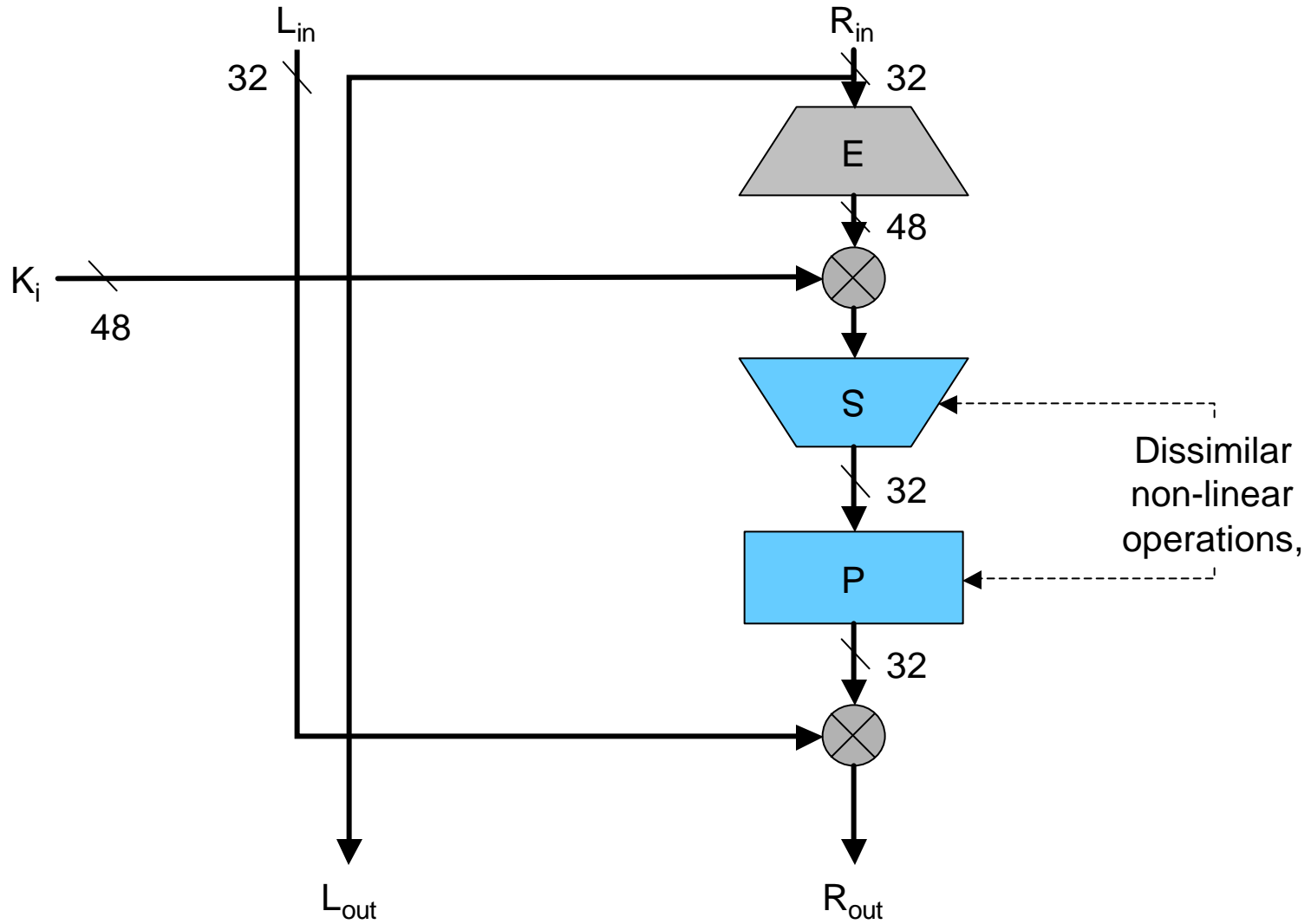
# DES Round<sub>i</sub>



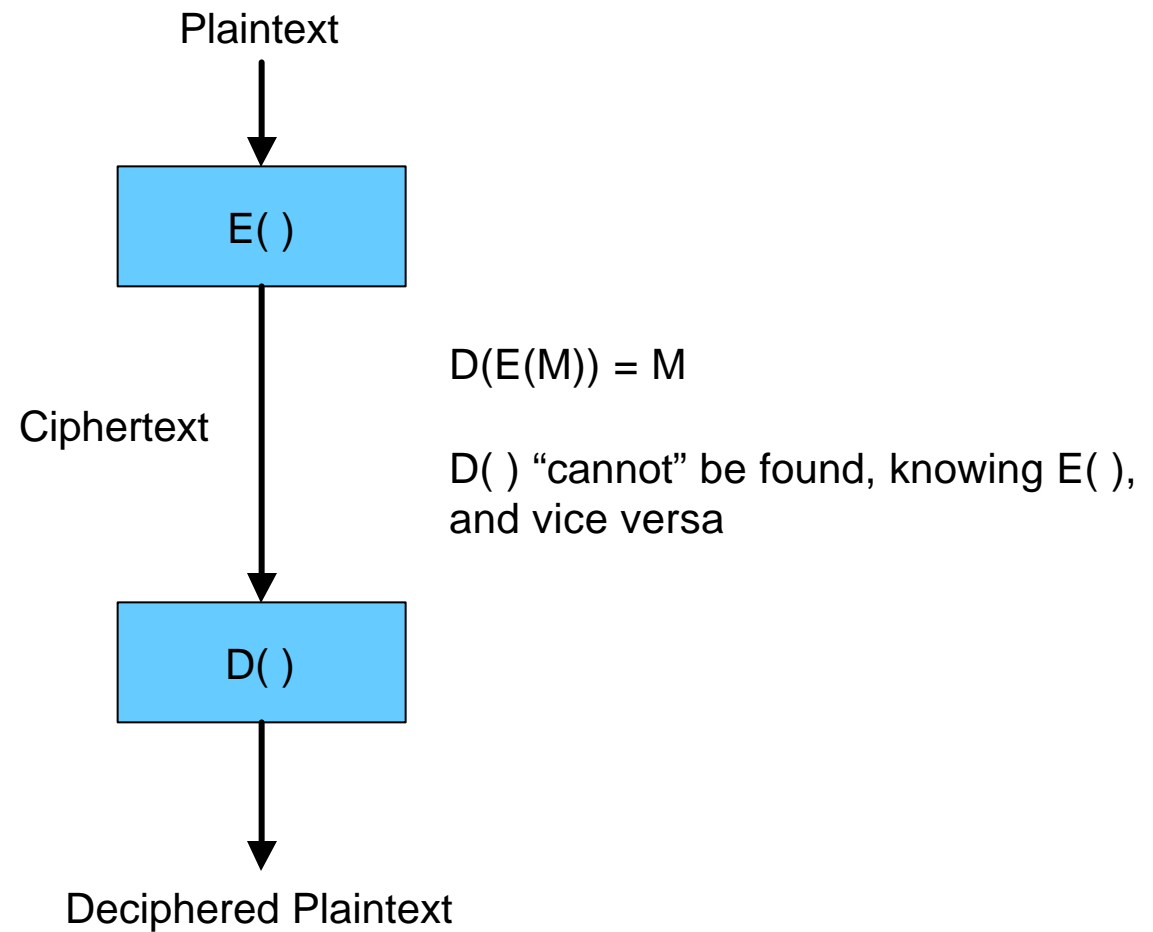
# DES Round<sub>i</sub>



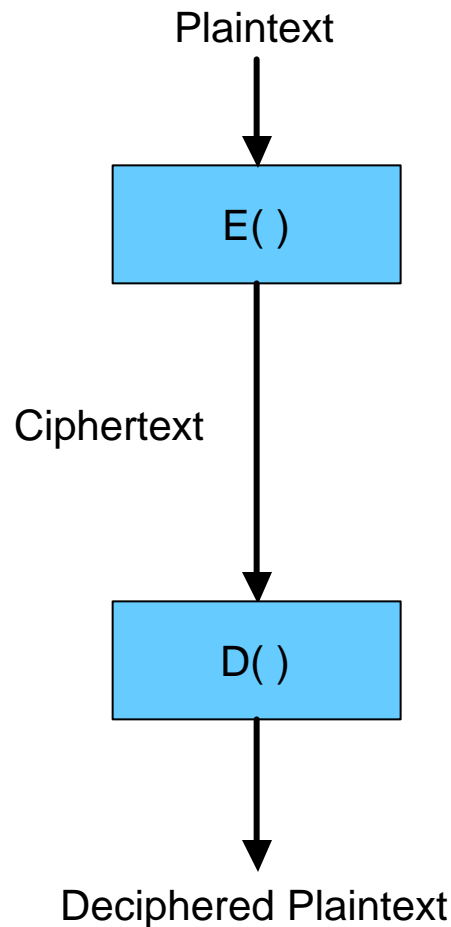
# DES Round<sub>i</sub>



# Public Key Cryptosystems



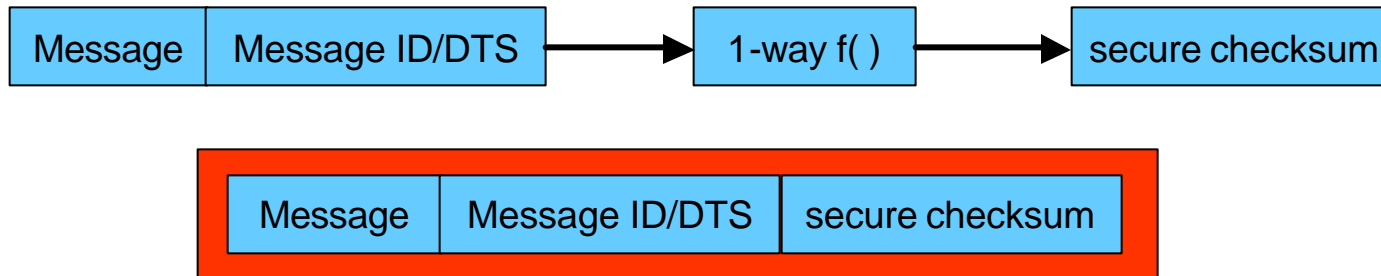
# Public Key Cryptosystems



- $D()$  and  $E()$  must be built on commutative functions:  
 $f(g(x)) = g(f(x))$
- Multiplication and exponentiation work – are there others?  
These form bases for Rivest-Shamir-Adleman (RSA) and Diffie-Hellman PKCs
- The apparent security of PKCs come from difficulty of computing logarithms and factoring composite numbers in a finite field. **Thought** to be NP-Complete problems, Which **might** make them mathematically intractable
- E.g.,  
 $E(M) = M^e$   
 $D(C) = C^d$   
 $D(E(M)) = (M^e)^d = M^{ed} = M^1$ , if  $d=e^{-1}$  in the field

# Applications of cryptography to security

- Confidentiality – the most obvious application
- Integrity



- Non-repudiation
  - Same as integrity, but seal the message: with user ID and user-specific key
- Authentication Challenge-response

