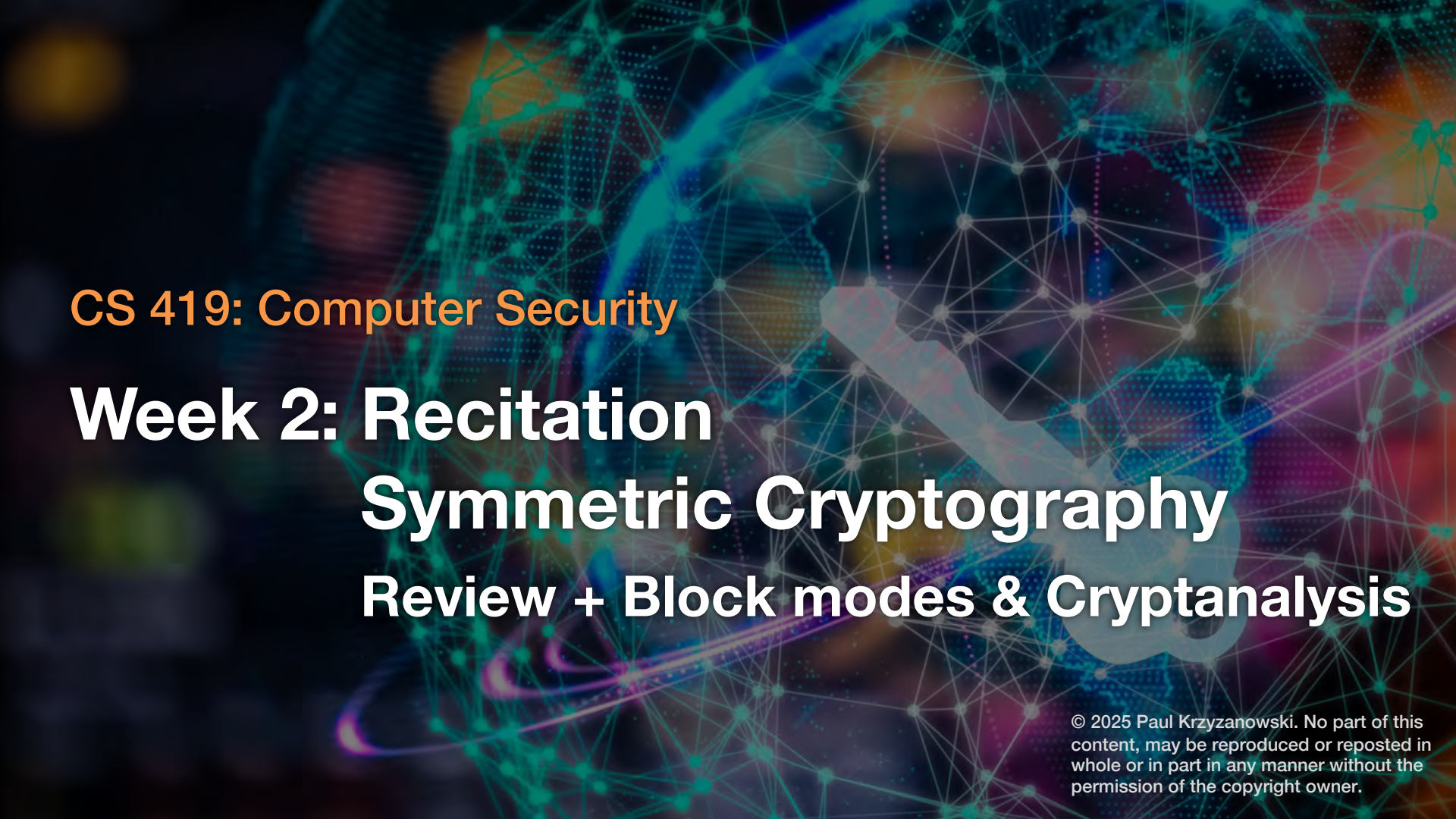


CS 419: Computer Security

Week 2: Recitation

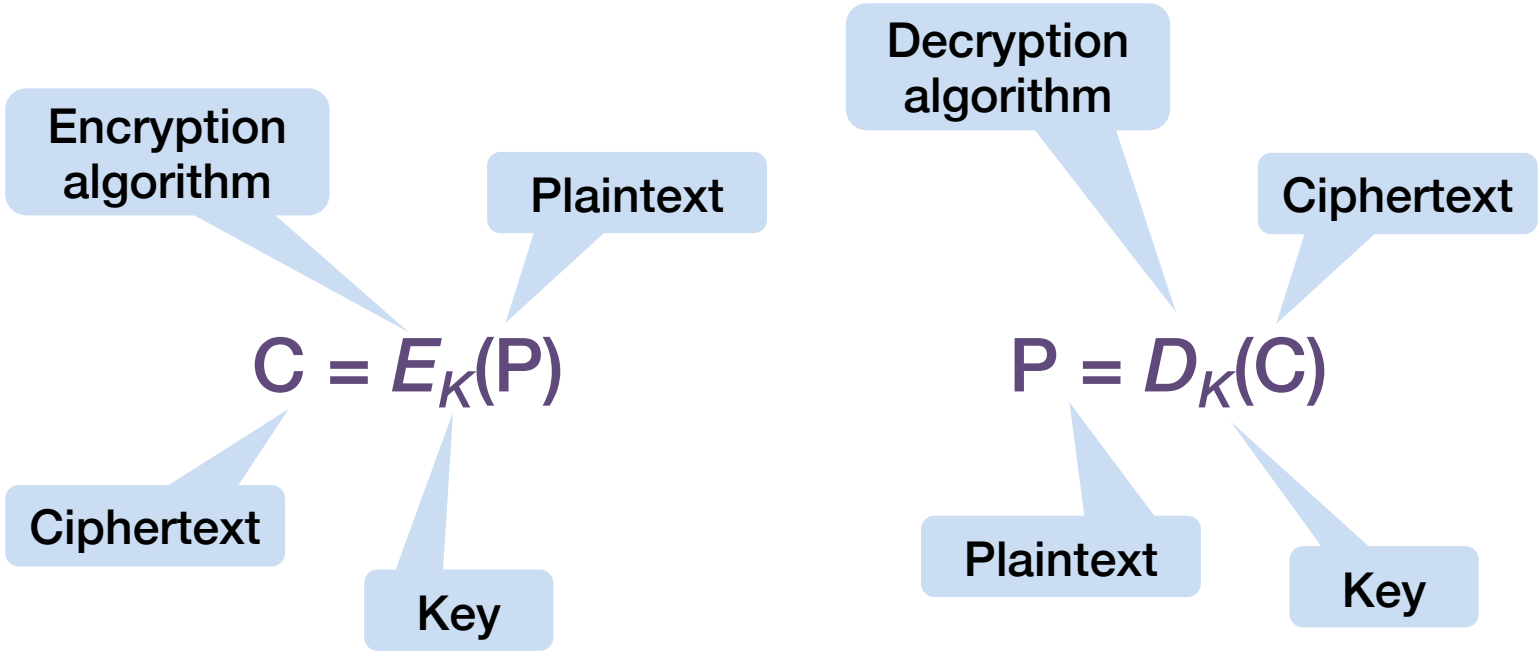
Symmetric Cryptography

Review + Block modes & Cryptanalysis

Review: Core Concepts

Symmetric Cryptography

Symmetric: one shared secret key, *K*, for encryption & decryption



Classic Cryptography

- **Monoalphabetic substitution ciphers**
  - One character in plaintext always produces the same ciphertext character

    `ABCABC` → `QMVQMV`

- **Polyalphabetic substitution ciphers**
  - The same character in plaintext may map to different characters in ciphertext

    `ABCABC` → `QMVGRS`

- **Transposition ciphers**
  - The characters are shuffled around but not substituted

    `ABCABC` → `BACCBA`

Frequency analysis

- **Definition**
  - A cryptanalysis technique used to break substitution ciphers by analyzing the frequency of letters or patterns in ciphertext
- **Assumption**
  - Certain letters and letter combinations appear more frequently in a given language
  - E.g., 'E' is the most common letter in English, occurring about 12% of the time
- **How it works**
  - Analyze the occurrence of each letter (byte) in the ciphertext
  - Match observed frequencies to typical letter distributions in the target language
  - Guess likely mappings between ciphertext and plaintext letters
  - Consider common digrams (e.g., "TH") and trigrams (e.g., "THE")
- **Mitigations**
  - Polyalphabetic ciphers (e.g., Vigenère with long keys or Enigma) disrupt frequency patterns by using multiple substitution alphabets

Shannon Entropy in Cryptography

- **Definition**
  - A measure of randomness in a dataset
  - High entropy in ciphertext and cryptographic keys ensures unpredictability, making attacks harder
- **Formula** $$H(X) = -\sum P(x) \log_2 P(x)$$
  - H(x) = Entropy measured in bits
  - P(x) = Probability of symbol x occurring in the message
- **Outcomes**
  - **Low entropy** → Predictable data (easier to compress or attack)
  - **High entropy** → Random, unpredictable data (what we want for cryptography)
    - High entropy in ciphertext makes it statistically indistinguishable from random noise
  - Polyalphabetic ciphers increase entropy

Perfect secrecy

- **Definition**
  - A cryptographic system achieves perfect secrecy if the ciphertext provides no information about the plaintext, regardless of the adversary's computational power
  - The probability of guessing the plaintext is the same whether the attacker sees the ciphertext or not: $P(M|C) = P(M)$
- **One-Time Pad: the only cipher that provides perfect secrecy**
  - The key must be random – as long as the message and only used once
  - Encryption: Ciphertext = Plaintext $\oplus$ Key (bitwise XOR)
  - Decryption: Plaintext = Ciphertext $\oplus$ Key (bitwise XOR).
- **Why it's perfect**
  - Any plaintext of the same length could have resulted in the same ciphertext
  - Without the key, the ciphertext is indistinguishable from random noise

Properties of a good cryptosystem

1. The secrecy should reside in the key, not in the the algorithm (algorithms should be public) ⇒ **Kerckhoffs's Principle**
2. Ciphertext should be indistinguishable from random values ⇒ **high entropy**
3. There should be no way to extract the **original plaintext** or the **key** short of enumerating all possible keys ⇒ **not invertible**
4. The keys should be large enough that **an exhaustive search is not feasible**
5. The cipher should not contain **weak keys**

Additional properties of a good cryptosystem

6. Encryption should be **efficient** – don't hesitate to use it
7. Keys and algorithms should be **as simple as possible** and work on **any data**
8. The size of the ciphertext should be the same as the plaintext, not a multiple of it – no bandwidth/storage penalty
9. The algorithm has been **extensively analyzed** – high confidence that it does not have weaknesses

Confusion & Diffusion

- **Two fundamental principles for designing secure ciphers**
  - They help hide the relationship between plaintext, ciphertext, and the encryption key, making attacks more difficult
- **Confusion**
  - Makes the relationship between the encryption key and ciphertext as complex as possible
  - Prevents attackers from determining the key even if they analyze multiple ciphertexts
  - **Effect**: small changes in the key cause unpredictable changes in the ciphertext
    - Examples: substitution ciphers, s-boxes
- **Diffusion**
  - Spreads out plaintext influence over the entire ciphertext
  - Ensures that changing a single plaintext bit affects many bits of ciphertext
  - **Effect**: Prevents frequency analysis and patterns appearing in ciphertext
    - Examples: transposition ciphers, permutation layers in AES, DES

Modern Ciphers: Block vs. Stream

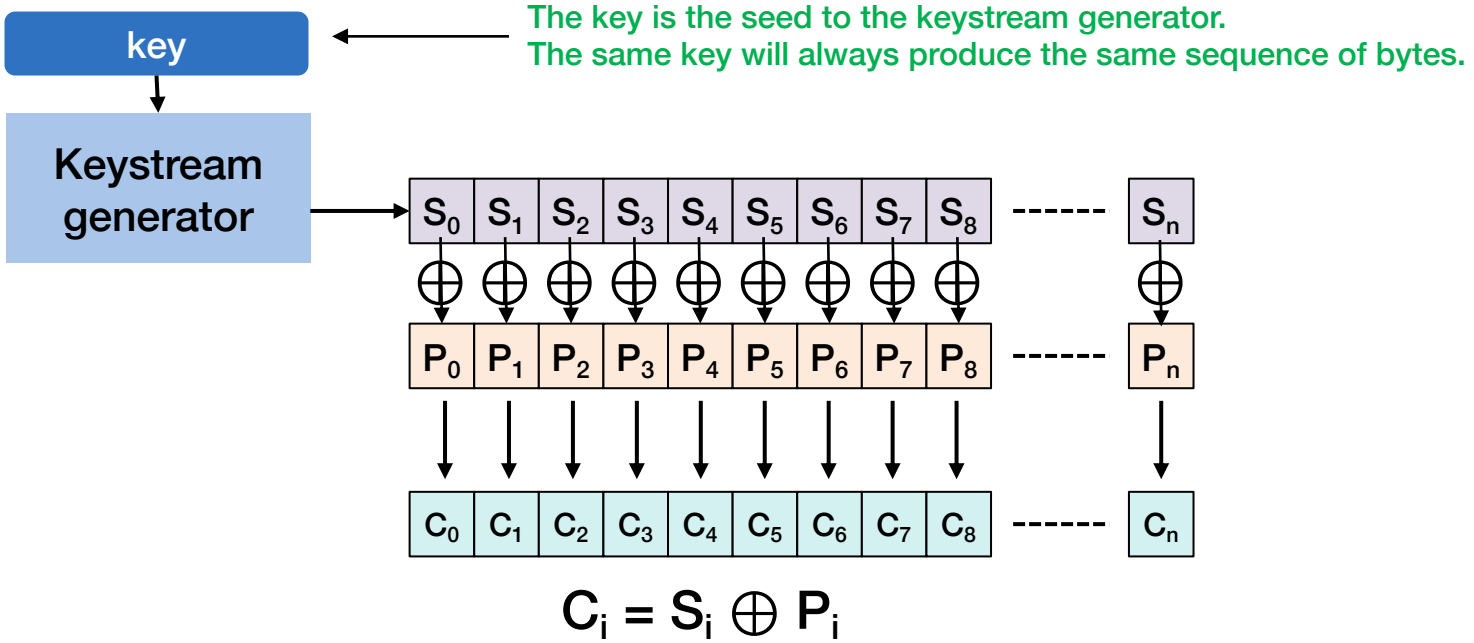
Symmetric ciphers are either block or stream ciphers

- **Block ciphers**
  - Operate on a set of bytes (a block)
  - Enables diffusion within the block (substitution and transposition)
  - Popular block ciphers: DES, AES
- **Stream ciphers**
  - Operate on one byte at a time
  - They perform substitution only
  - Popular stream cipher: Chacha20

Stream ciphers (e.g., ChaCha20)

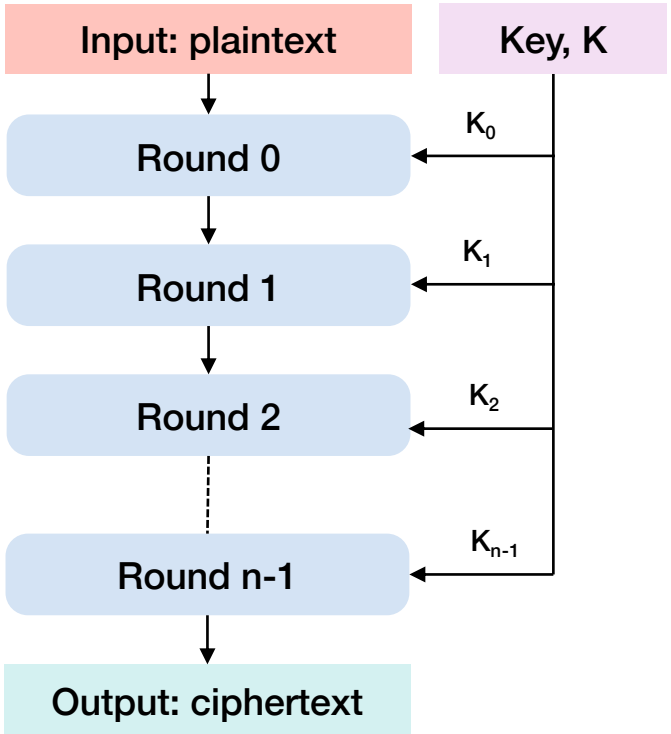
Key stream generator produces a sequence of **pseudo-random** bytes

Simulates a one-time pad



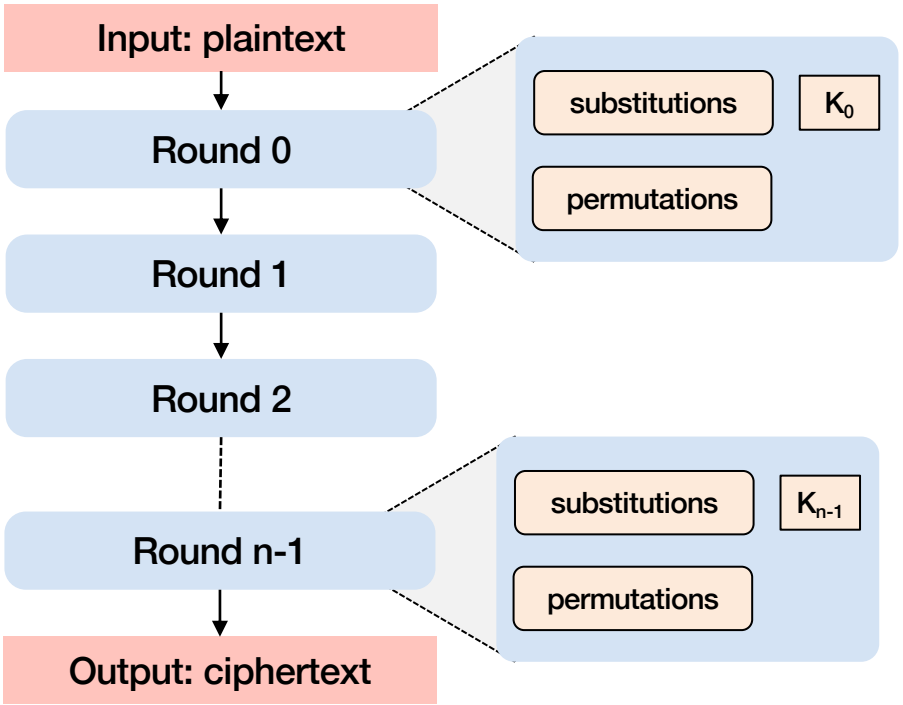
Block ciphers

- **Block ciphers encrypt a *block* of plaintext at a time**
- **DES & AES are two popular block ciphers**
  - DES: uses 64-bit blocks
  - AES: uses 128-bit blocks
- **Block ciphers are usually iterative ciphers**
  - The encryption process is an iteration through several ***round*** operations
  - A single round does not provide perfect confusion or diffusion
  - Each round uses a **subkey** derived from the key
    - Determines what substitutions & transpositions take place



Block cipher: SPN (substitution-permutation network)

Each round consists of substitutions & permutations = SP Network



Substitution = S-box

- Table lookup
- Converts a small block of input to a block of output

Permutation

- Scrambles the bits in a prescribed order

Key application per round

- **Subkey**, K_n, per round derived from the key
- Can drive behavior of s-boxes
- May be XORed with the output of each round

Each round increases the amount of confusion and diffusion

Cipher modes

Cipher modes define how block ciphers (e.g., AES, DES) encrypt data larger than a single block (typically 128 bits).

Not a good idea to use block ciphers directly

A stream of data is broken into *k*-byte blocks

- Each block is encrypted separately
- This is called **Electronic Codebook** (**ECB**)

Problems

1. The same plaintext results in identical encrypted blocks
   - Attackers can build up a code book of plaintext/ciphertext matches
   - Even without decrypting, attackers can see patterns of identical blocks
2. Attackers can add/delete/replace blocks from other messages encrypted with the same key and you may not know

Goal of block cipher modes beyond Electronic Codebook (ECB)

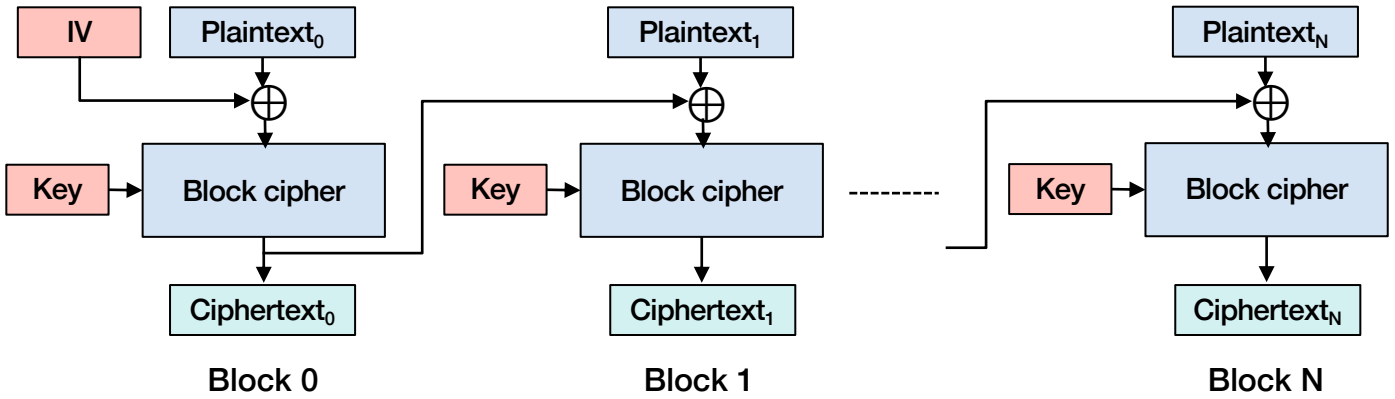
Transform each block so that its ciphertext is dependent on its position in the input stream

We will only look at two modes – there are more.

Cipher Block Chaining (CBC) mode

- **Random initialization vector (IV) = bunch of *k* random bits**
  - Non-secret: both parties need to know this – usually added to the start of the message
- **Exclusive-or with first plaintext block – then encrypt the block**
- **Take exclusive-or of the result with the next plaintext block**

$$c_i = E_K(m_i) \oplus c_{i-1}$$

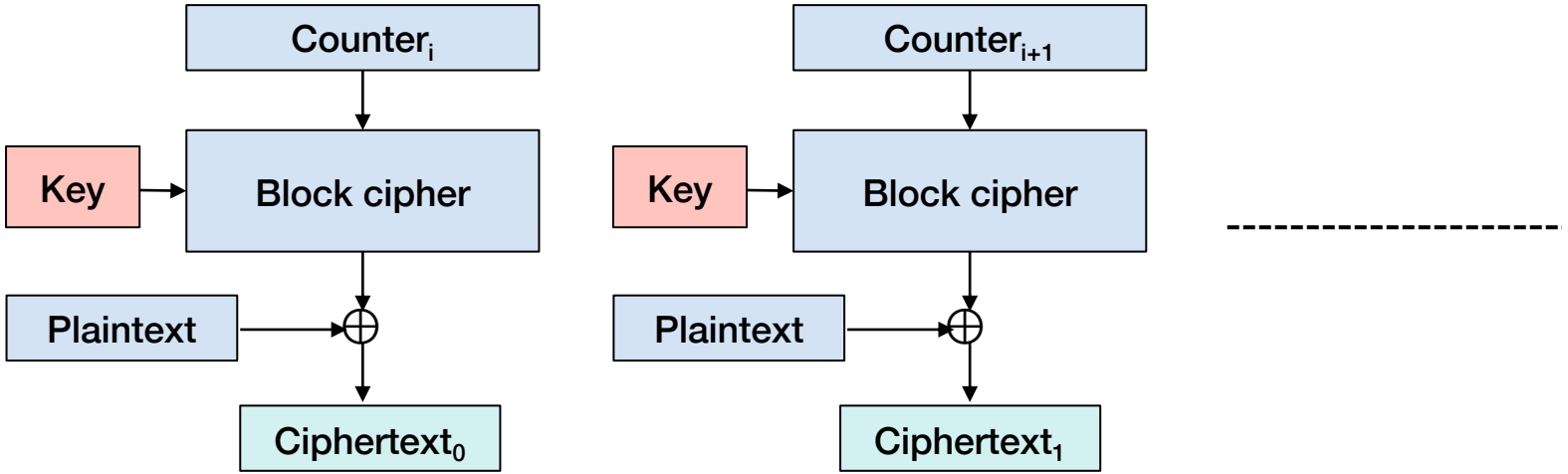


CBC Observations

- **Identical plaintext does not produce the same ciphertext**
  - Unless all previous blocks and the initialization vector are identical
- **Each block is a function of all previous blocks**
  - Great for diffusion
  - Not great for parallel processing

Block encryption: Counter (CTR) mode

- **Random starting counter = bunch of *k* random bits, just like IV**
  - Any function producing a non-repeating sequence (an incrementing number is a common function)
- **Encrypt the counter with the key**
- **Exclusive-or result with plaintext block**



CTR Mode Observations

- **Supports parallel processing**
  - One block does not depend on the results of the previous blocks
- **Reusing a counter can compromise security**
  - It will allow attackers to identify matching blocks from other ciphertext encrypted with the same key
- **Neither ECB, CBC, nor CTR mode provide integrity or authentication**
  - A receiver can't tell if the ciphertext has been modified

Cryptanalysis

Cryptographic attacks

Goal: Break encryption or recover plaintext from ciphertext – try to find the key

Core assumptions

You have an idea of what content you're looking for (German text, ARM instructions, JPEG files) and a likely encryption algorithm

Brute force attack: exhaustive search

- The attacker iterates through all possible keys
- This isn't feasible for large key sizes – it would take too long

Cryptanalytic attacks:

Try to find weaknesses in the cipher to recover the key without an exhaustive search

Exploit weaknesses in algorithms, implementations, or key management

We are not examining HOW to break codes but the approaches that are taken

Cryptoanalytic attacks

- **Side-Channel Attack**: Exploit physical leakage
  - Observes timing, power consumption, or electromagnetic radiation to extract cryptographic keys
  - Mitigation: constant-time algorithms regardless of key or plaintext data
- **Chosen Plaintext Attack**
  - Attacker can create plaintext and observe the corresponding ciphertext
- **Known Plaintext Attack**
  - The attacker has access to both plaintext & ciphertext but doesn't get to choose the text
- **Ciphertext-only Attack**
  - The attacker only sees ciphertext.
  - Popular in moves but not practical in real life with modern ciphers

We are not examining HOW to break codes but the approaches that are taken

Differential Cryptanalysis

Examine how changes in input affect changes in output

Analyze how differences in plaintext propagate through a cipher to impact ciphertext

- **Discover where a cipher exhibits non-random behavior**
  - These properties can be used to extract the secret key
  - Applied to block ciphers, stream ciphers, and hash functions (functions that flip & move bits vs. mathematical operations)

- **A chosen plaintext attack is normally used**
  - The attacker must be able to choose the plaintext and see the corresponding cipher text

Differential Cryptanalysis

- **Provide plaintext with known differences**
  - See how those differences appear in the ciphertext
  - The results depend on the key and the s-boxes in the algorithm
- **Do this with lots and lots of known *plaintext-ciphertext* sets**
- **Statistical differences, if found, may allow a key to be recovered faster than with a brute-force search**
  - You may identify certain bits of the key.
  - That allows you to deduce that certain keys are not worth trying

Linear Cryptanalysis

Create a predictive approximation of inputs to outputs

- **Instead of looking for differences, linear cryptanalysis attempts to come up with a linear formula (e.g., a bunch of *xor* operations) that connects certain input bits, output bits, and key bits with a probability higher than random**
  - Goal is to approximate the behavior of s-boxes
- **Part 1: construct linear equations**
  - Find high correlations
- **Part 2: guess key bits**
  - Guess enough bits so that a brute-force attack becomes feasible

Linear Cryptanalysis

It will *not* recreate the working of the cipher

- You just hope to find non-random behavior that gives you insight into what bits of the key might matter
- Linear cryptanalysis finds statistical biases in encryption – non-random variations

- **Works better than differential cryptanalysis for known plaintext**
  - Differential cryptanalysis works best with chosen plaintext

- **Linear & differential cryptanalysis will rarely recover a key but may be able to reduce the number of keys that need to be searched**

The End