CSE 565 Computer Security
Fall 2018

Lecture 3: Symmetric Encryption II

Department of Computer Science and Engineering
University at Buffalo

Symmetric Encryption

- **So far we've covered:**
  - **what secure symmetric encryption is**
  - **high-level design of block ciphers**
  - **DES**
- **Next, we'll talk about:**
  - **AES**
  - **block cipher encryption modes**

Advanced Encryption Standard (AES)

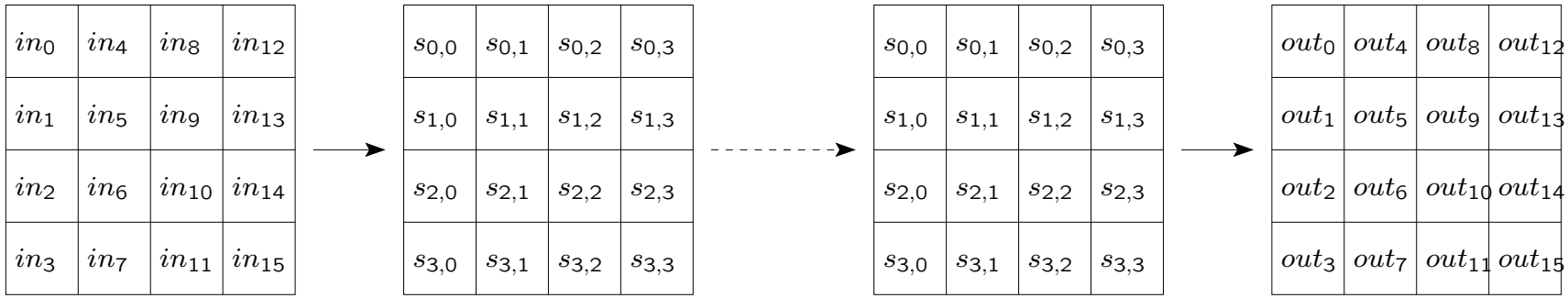
- In 1997 NIST made a formal call for an unclassified publicly disclosed encryption algorithm available worldwide and royalty-free
  - the goal was to replace DES with a new standard called AES
  - the algorithm must be a symmetric block cipher
  - the algorithm must support (at a minimum) 128-bit blocks and key sizes of 128, 192, and 256 bits
- The evaluation criteria were:
  - security
  - speed and memory requirements
  - algorithm and implementation characteristics

AES

- In 1998 15 candidate AES algorithms were announced
- They were narrowed to 5 in 1999: MARS, RC6, Rijndael, Serpent, and Twofish
  - all five were thought to be secure
- In 2001 Rijndael was adopted as the AES standard
  - invented by Belgian researchers Deamen and Rijmen
  - designed to be simple and efficient in both hardware and software on a wide range of platforms
  - supports different block sizes (128, 192, and 256 bits)
  - supports keys of different length (128, 192, and 256 bits)
  - uses a variable number of rounds (10, 12, or 14)

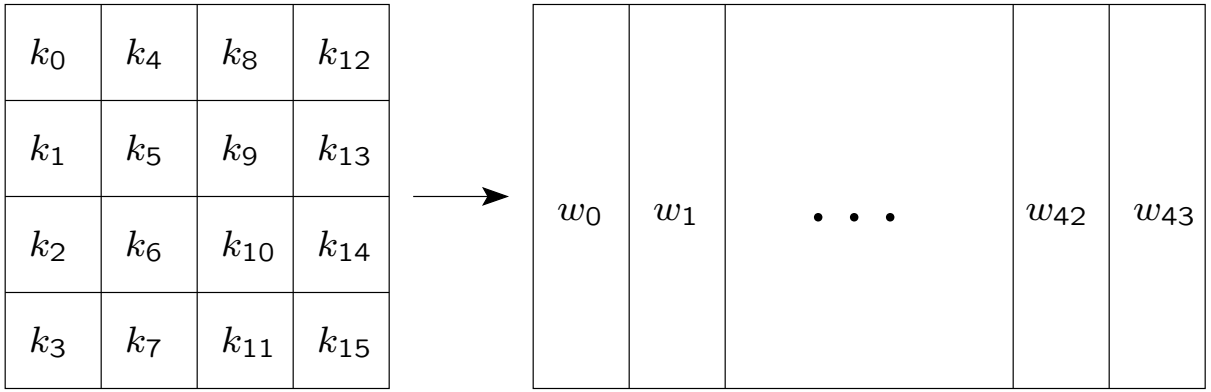
AES

- **During encryption:**
  - **the block is copied into the state matrix**
  - **the state is modified at each round of encryption and decryption**
  - **the final state is copied to the ciphertext**



AES

- **The key schedule in AES**
  - **the key is treated as a 4×4 matrix as well**
  - **the key is then expanded into an array of words**
  - **each word is 4 bytes and there are 44 words (for 128-bit key)**
  - **four distinct words serve as a round key for each round**



AES

- **Rijndael doesn't have a Feistel structure**
  - **2 out of 5 AES candidates (including Rijndael) don't use Feistel structure**
  - **they process the entire block in parallel during each round**
- **The operations are (3 substitution and 1 permutation operations):**
  - SUBBYTES: **byte-by-byte substitution using an S-box**
  - SHIFTROWS: **a simple permutation**
  - MIXCOLUMNS: **a substitution using mod 2^8 arithmetics**
  - ADDROUNDKEY: **a simple XOR of the current state with a portion of the expanded key**

AES

- **At a high-level, encryption proceeds as follows:**
  - **set initial state $s_0 = m$**
  - **perform operation AddRoundKey (XORs k_i and s_i)**
  - **for each of the first $Nr - 1$ rounds:**
    - **perform a substitution operation SubBytes on s_i and an S-box**
    - **perform a permutation ShiftRows on s_i**
    - **perform an operation MixColumns on s_i**
    - **perform AddRoundKey**
  - **the last round is the same except no MixColumns is used**
  - **set the ciphertext $c = s_{Nr}$**

AES

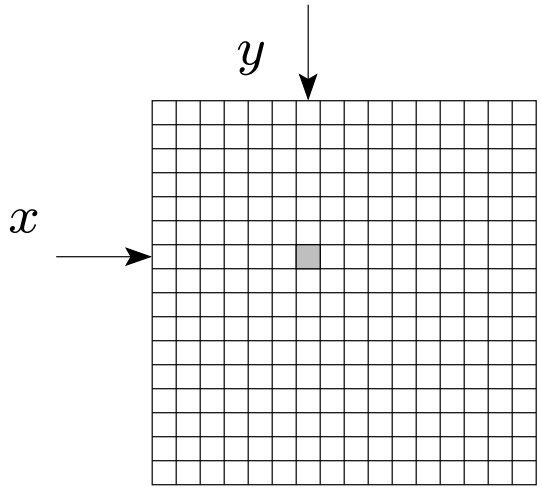
- **More about Rijndael design...**
  - **ADDROUNDKEY is the only operation that uses key**
    - **that's why it is applied at the beginning and at the end**
  - **all operations are reversible**
  - **the decryption algorithm uses the expanded key in the reverse order**
  - **the decryption algorithm, however, is not identical to the encryption algorithm**

AES

- **The SUBBYTES operation**
  - **maps a state byte $s_{i,j}$ to a new byte $s'_{i,j}$ using S-box**
  - **the S-box is a 16×16 matrix with a byte in each position**
    - **the S-box contains a permutation of all possible 256 8-bit values**
    - **the values are computed using a formula**
    - **it was designed to resist known cryptanalytic attacks (i.e., to have low correlation between input bits and output bits)**

AES

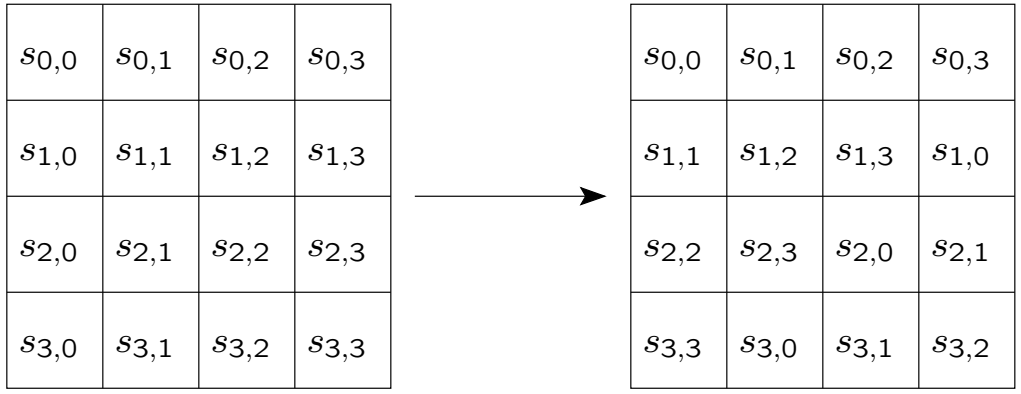
- **The SUBBYTES operation**
  - **to compute the new $s'_{i,j}$:**
    - **set x to the 4 leftmost bits of $s_{i,j}$ and y to its 4 rightmost bits**
    - **use x as the row and y as the column to locate a cell in the S-box**
    - **use that cell value as $s'_{i,j}$**



  - **the same procedure is performed on each byte of the state**

AES

- The SHIFTROWS operation
  - performs circular left shift on state rows
    - 2nd row is shifted by 1 byte
    - 3rd row is shifted by 2 bytes
    - 4th row is shifted by 3 bytes



  - important because other operations operate on a single cell

AES

- **The MIXCOLUMNS operation**
  - **multiplies the state by a fixed matrix**

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

  - **was designed to ensure good mixing among the bytes of each column**
  - **the coefficients 01, 02, and 03 are for implementation purposes (multiplication involves at most a shift and an XOR)**

AES

- **Decryption:**
  - **inverse S-box is used in SUBBYTES**
  - **inverse shifts are performed in SHIFTROWS**
  - **inverse multiplication matrix is used in MIXCOLUMNS**
- **Key expansion:**
  - **was designed to resist known attacks and be efficient**
  - **knowledge of a part of the key or round key doesn't enable calculation of other key bits**
  - **round-dependent values are used in key expansion**

AES

- **Summary of Rijndael design**
  - **simple design but resistant to known attacks**
  - **very efficient on a variety of platforms including 8-bit and 64-bit platforms**
  - **highly parallelizable**
  - **had the highest throughput in hardware among all AES candidates**
  - **well suited for restricted-space environments (very low RAM and ROM requirements)**
  - **optimized for encryption (decryption is slower)**

AES Hardware Implementation

- It's been long known that hardwared implementations of AES are extremely fast
  - the speed of encryption is compared with the speed of disk read
- Harware implementations however remained unaccessible to the average user
- Recently Intel introduced new AES instruction set (AES-NI) in its commodity processors
  - other processor manufacturers support it now as well
  - hardware acceleration can be easily used on many platforms

Secure Encryption

- **Using a strong block cipher is not enough for secure encryption!**
  - **if you need to send more than 1 block (i.e., 16 bytes) over the key lifetime, applying plain block cipher to the message as**

    $$\mathsf{Enc}_k(b_1), \mathsf{Enc}_k(b_2), \ldots$$

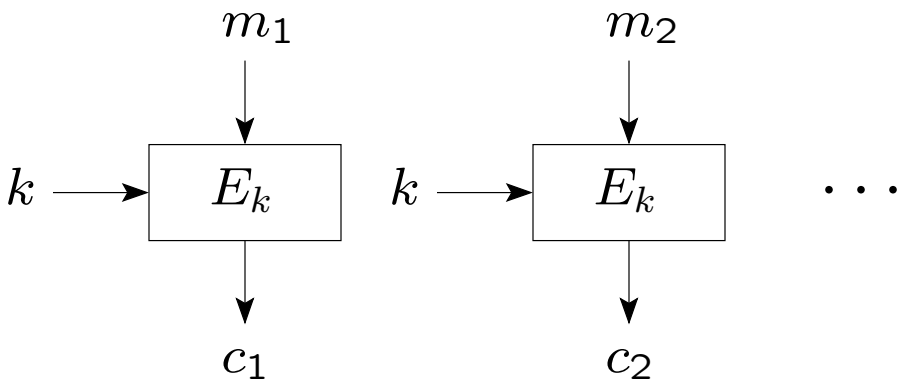
    will fail even weak definitions of secure encryption
  - **no deterministic encryption can be secure if multiple blocks are sent**

Encryption Modes

- **Encryption modes indicate how messages longer than one block are encrypted and decrypted**
- **4 modes of operation were standardized in 1980 for Digital Encryption Standard (DES)**
  - **can be used with any block cipher**
  - **electronic codebook mode (ECB), cipher feedback mode (CFB), cipher block chaining mode (CBC), and output feedback mode (OFB)**
- **5 modes were specified with the current standard Advanced Encryption Standard (AES) in 2001**
  - **the 4 above and counter mode**

Encryption Modes

- **Electronic Codebook (ECB) mode**
  - **divide the message m into blocks $m_1 m_2 \ldots m_\ell$ of size n each**
  - **encipher each block separately: for $i = 1, \ldots, \ell$, $c_i = E_k(m_i)$, where E denotes block cipher encryption**
  - **the resulting ciphertext is $c = c_1 c_2 \ldots c_\ell$**

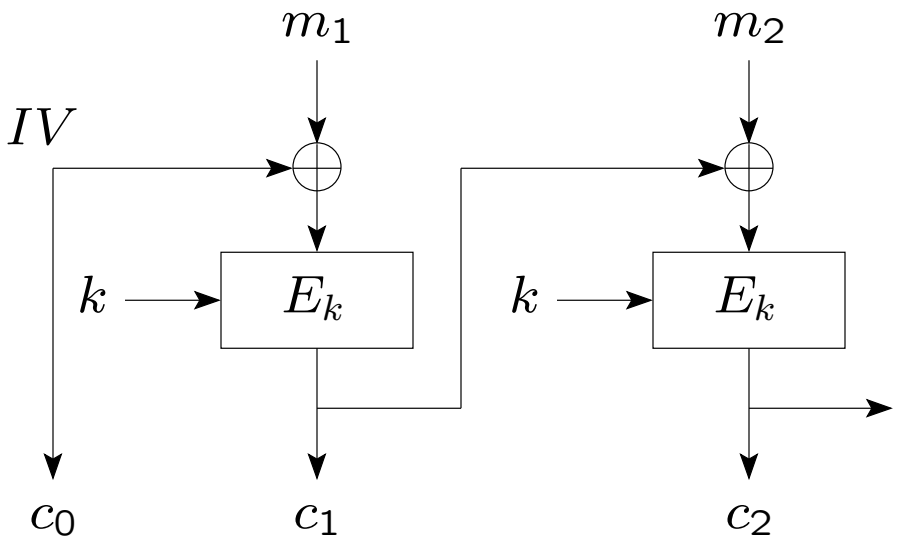


Encryption Modes

- **Properties of ECB mode:**
  - **identical plaintext blocks result in identical ciphertexts (under the same key)**
  - **each block can be encrypted and decrypted independently**
  - **this mode doesn't result in secure encryption**
- **ECB mode is a plain invocation of the block cipher**
  - **it allows the block cipher to be used in other, more complex cryptographic constructions**

Encryption Modes

- **Cipher Block Chaining (CBC) mode**
  - **set** $c_0 = IV \xleftarrow{R} \{0,1\}^n$ **(initialization vector)**
  - **encryption: for** $i = 1, \ldots, \ell$, $c_i = E_k(m_i \oplus c_{i-1})$
  - **decryption: for** $i = 1, \ldots, \ell$, $m_i = c_{i-1} \oplus D_k(c_i)$, **where** D **is block cipher decryption**



Encryption Modes

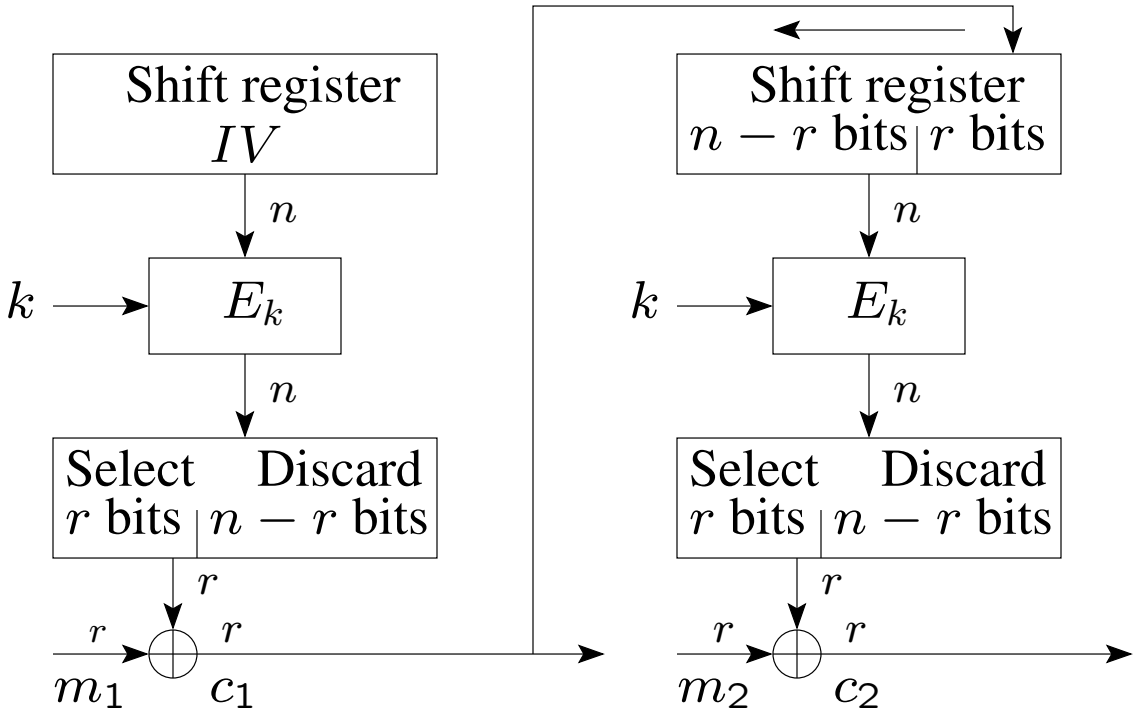
- **Properties of CBC mode:**
  - **this mode is CPA-secure (has a formal proof) if the block cipher can be assumed to produce pseudorandom output**
  - **a ciphertext block depends on all preceding plaintext blocks**
  - **sequential encryption, cannot use parallel hardware**
  - **IV must be random and communicated intact**
    - **if the IV is not random, security quickly degrades**
    - **if someone can fool the receiver into using a different IV, security issues arise**

Encryption Modes

- **Cipher Feedback (CFB) mode**
  - **the message is XORed with the encryption of the feedback from the previous block**
  - **set initial input $I_1 = IV$**
  - **encryption: $c_i = E_k(I_i) \oplus m_i$; $I_{i+1} = c_i$**
  - **decryption: $m_i = c_i \oplus E_k(I_i)$**
- **This mode allows the block cipher to be used as a stream cipher**
  - **if our application requires that plaintext units shorter than the block are transmitted without delay, we can use this mode**
  - **the message is transmitted in r-bit units (r is often 8 or 1)**

Encryption Modes

- **Cipher Feedback (CFB) mode**
  - **input: key k, n-bit IV, r-bit plaintext blocks $m_1, \ldots$**
  - **output: r-bit ciphertext blocks $c_1, \ldots$**

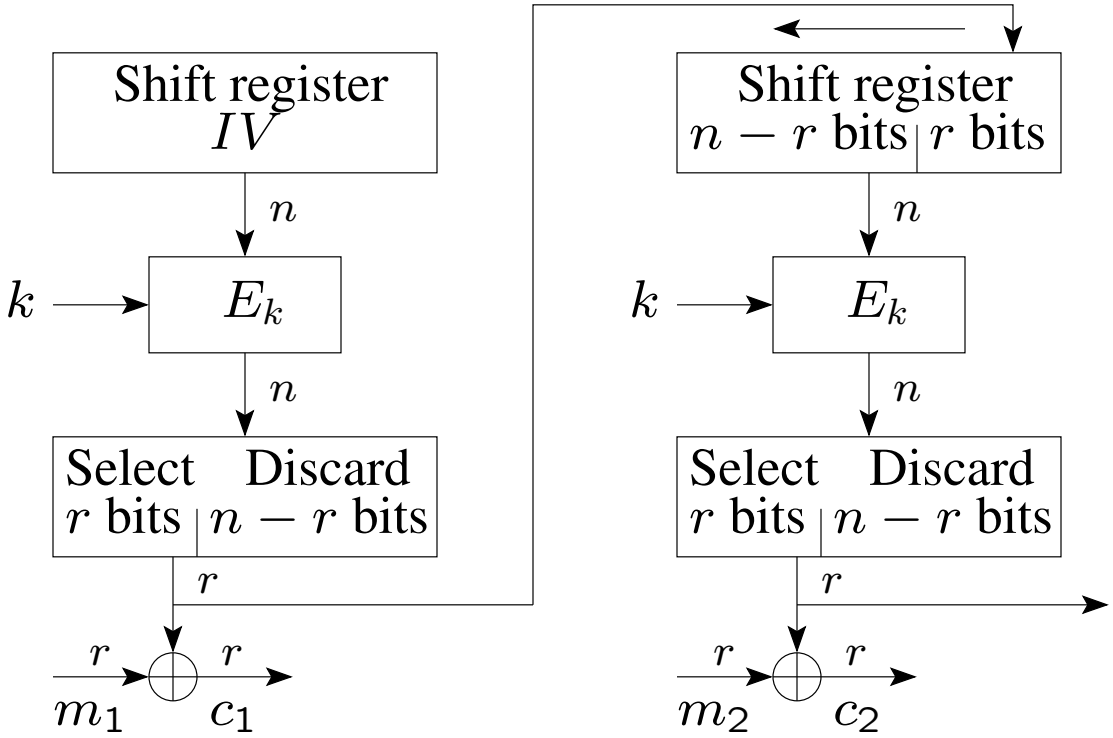


Encryption Modes

- **Properties of CFB mode:**
  - **the mode is CPA-secure (under the same assumption that the block cipher is strong)**
  - **similar to CBC, a ciphertext block depends on all previous plaintext blocks**
  - **throughput is decreased when the mode is used on small units**
    - **one encryption operation is applied per r bits, not per n bits**

Encryption Modes

- **Output Feedback (OFB) mode**
  - **similar to CFB, but the feedback is from encryption output and is independent of the message**

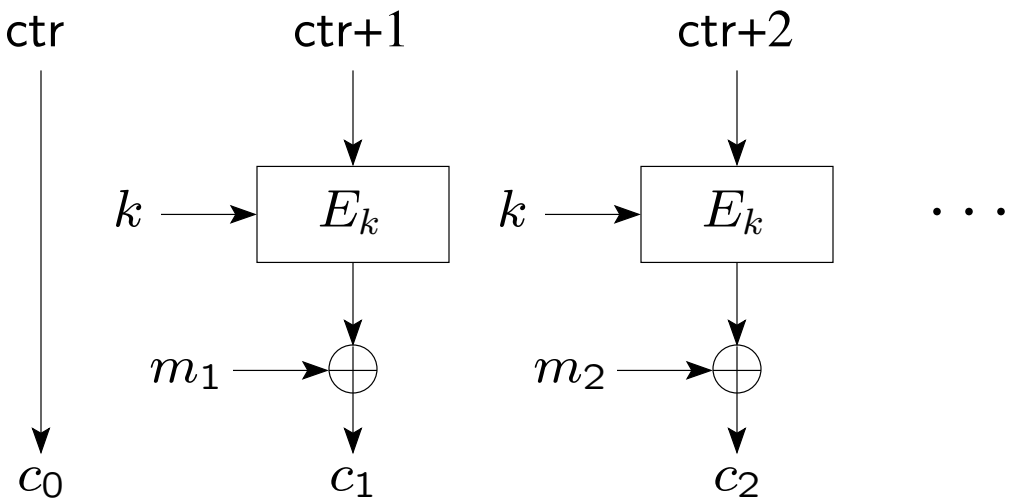


Encryption Modes

- **Output Feedback (OFB) mode:**
  - **n-bit feedback is recommended**
  - **using fewer bits for the feedback reduces the size of the cycle**
- **Properties of OFB:**
  - **the mode is CPA-secure**
  - **the key stream is plaintext-independent**
  - **similar to CFB, throughput is decreased for $r < n$, but the key stream can be precomputed**

Encryption Modes

- **Counter (CRT) mode**
  - **a counter is encrypted and XORed with a plaintext block**
  - **no feedback into the encryption function**
  - **initially set** $\mathsf{ctr} = IV \xleftarrow{R} \{0,1\}^n$



Encryption Modes

- **Counter (CRT) mode**
  - **encryption: for** $i = 1, \ldots, \ell$**,** $c_i = E_k(\mathsf{ctr} + i) \oplus m_i$
  - **decryption: for** $i = 1, \ldots, \ell$**,** $m_i = E_k(\mathsf{ctr} + i) \oplus c_i$
- **Properties:**
  - **ciphertext can have the same length as the plaintext**
  - **if the last plaintext block is incomplete, we just truncate the last cipherblock and transmit it**

Encryption Modes

- **Advantages of counter mode**
  - **Hardware and software efficiency: multiple blocks can be encrypted or decrypted in parallel**
  - **Preprocessing: encryption can be done in advance; the rest is only XOR**
  - **Random access: ith block of plaintext or ciphertext can be processed independently of others**
  - **Security: at least as secure as other modes (i.e., CPA-secure)**
  - **Simplicity: doesn't require decryption or decryption key scheduling**
- **But what happens if the counter is reused?**

Summary

- AES is the current block cipher standard
  - it offers strong security and fast performance
- Five encryption modes are specified as part of the standard
  - ECB mode is not for secure encryption
  - any other encryption mode achieves sufficient security
    - use one of these modes for encryption even if the message is a single block
- Strong randomness is required for cryptographic purposes
  - key generation, IV generation, etc.