CO 487: Applied Cryptography

Chapter 1: Symmetric-Key Encryption

November 27, 2019

Contents

1 Symmetric-Key Encryption Scheme (SKES)

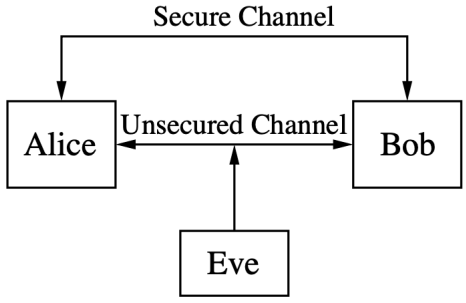
1.1 The security of SKES and the Adversary

Definition: **Symmetric-Key Encryption Scheme (SKES)** consists of:

- M: the plaintext space,
- C: the ciphertext space,
- K: the key space,
- a family of encryption functions, $E_k : M \to C$, $\forall k \in K$,
- a family of decryption functions, $D_k : C \to M$, $\forall k \in K$, such that $D_k(E_k(m)) = m$ for all $m \in M$, $k \in K$.

Equivalently: $E : K \times M \to C$ and $D : K \times C \to M$.

How does a symmetric-key encryption scheme achieve confidentiality?



1. Alice and Bob agree on a secret key $k \in K$ by communicating over the secure channel.
2. Alice computes the ciphertext $c = E_k(m)$ and sends c to Bob over the unsecured channel.
3. Bob retrieves the plaintext by computing $m = D_k(c)$.

How does an adversary of the symmetric-key encryption scheme look like?

1. **Basic Assumption**

   The adversary knows everything about the symmetric-key encryption scheme, except the key k chosen by Alice and Bob.

2. **Adversary's Goal**

   - Insecure
     A symmetric-key encryption scheme is **totally insecure**/**totally broken** if the adversary can recover the secret key **or** systematically recover plaintext from ciphertext without learning the secret key.
   - Secure
     A symmetric-key encryption scheme is **semantically secure** if the adversary cannot learn any partial information about the plaintext from the ciphertext, except its length.

3. **Adverasry's Interaction**: Three Types of Attacks

   - Passive Attacks
     - Ciphertext-only Attack
     - Known-Plaintext Attack: The adversary also knows some plaintext and the corresponding ciphertext.
   - Active Attacks
     - Chosen-Plaintext Attack: The adversary can also choose some plaintext(s) and obtain the corresponding ciphertext(s).
     - Chosen-Ciphertext Attack: The adversary can also choose some ciphertext(s) and obtain the corresponding plaintext(s).
   - Other Attacks
     - Side-Channel Attacks: Monitoring the encryption and decryption equipment.
     - Physical Attacks: Bribery, blackmail, rubber hose, etc.

4. **Computational Power of the Adversary**

   - Infinite

     Information-Theoretic Security means the adversary has infinite computational resources.
   - Polynomial

     Complexity-Theoretic Security means the adversary has a "polynomial-time Turing machine".
   - Limited

     Computational Security means the adversary has X number of real computers, which means the adversary is "computationally bounded".

     In this course, 2^{40} operations is considered very easy (to break), 2^{56} operations is considered easy, 2^{64} operations is considered feasible,

2^{80} operations is considered barely feasible, 2^{128} operations is considered infeasible.

In summary, A **SKES** is **secure** if it is semantically secure against a chosen-plaintext attack by a computationally bounded adversary. In other words, the SKES should be:

- semantically secure: the adversary cannot learn any partial information about the plaintext from the ciphertext, except possibly its length.
- chosen-plaintext attack: the adversary can also choose some plaintext(s) and obtain the corresponding ciphertext(s).
- computationally bounded: the adversary has X number of real computers.

The adversary has to **accomplish** these things to **break a symmetric-key encryption scheme**:

1. A ciphertext c which generated by the secret key k is given to the adversary.
2. The adversary can select plaintext and obtain the corresponding ciphertext during its computation.
3. The adversary obtains some information about the plaintext corresponding to c (except the length of m) in a feasible amount of computation.

Definition: A cryptographic scheme has a **security level** of l bits if the fastest known attack on the scheme takes approximately 2^l operations. (As of 2019, a security level of 128 bits is desirable in practice.)

Requirement of a Symmetric-Key Encryption Scheme:

- Basic assumption (Kerckhoffs's principle, Shannon's maxim): Assume the adversary knows everything about the algorithm, except the secret key k.
- Algorithms: Efficient algorithms should be known for computing E_k and D_k (i.e. for encryption and decryption).
- Key's Size: The key should be small (but large enough to render exhaustive key search infeasible).
- Security: The scheme should be secure even against the designer of the system.

2 Simple Substitution Cipher

2.1 Basic Ideas

Definition: **The simple substitution cipher** is consisted of

- M: English messages,
- C: Encrypted messages,
- K: Permutations of the English alphabet,
- $E_k(m)$: Apply permutation k to m, one letter at a time,
- $D_k(c)$: Apply inverse permutation k^{-1} to c, one letter at a time.

Example

$k =$

a	b	c	d	e	f	g	h	i	j	k	l	m
D	N	X	E	S	K	O	J	T	A	F	P	Y
n	o	p	q	r	s	t	u	v	w	x	y	z
I	Q	U	B	R	Z	G	V	C	H	M	W	L

,

$m =$ the big dog,

$c = E_k$(the big dog)=GJS NTO EQO

The number of keys to try exhaustive key search is $26! \approx 4 \times 10^{26} \approx 2^{88}$, which is infeasible.

But simple substitution cipher is still **not secure** since not all letters are equally common in English text. The adversary can test out the frequency of letters and find the letters with similar frequency. It is totally insecure against a chosen-plaintext attack and mostly against a ciphertext-only attack.

The Process of Breaking Simple Substitution Cipher for a Language

1. Find the frequency of each letter in this language (on Google maybe).
2. Find the most occurring symbol and change it to the form of the most frequent letter in the whole language.
3. Find the next most occurring symbol and change it to the form of the second most frequent letter in the whole language.

   ...Until all the symbols of the cryptogram have been found.

3 Polyalphabetic Ciphers

Basic Idea: Use several permutations, so a plaintext letter is encrypted to one of several possible ciphertext letters.

3.1 Binary Messages

Notation: $\oplus$ is bitwise exclusive-or.$\Longleftrightarrow$ bitwise addition modulo 2.

Operation:

1. $x \oplus x = 0$
2. $x \oplus y = y \oplus x$
3. $x = y \oplus z \Rightarrow x \oplus y = z$

Example: $1011001010 \oplus 1001001001 = 0010000011$

	1	0	1	1	0	0	1	0	1	0
$\oplus$	1	0	0	1	0	0	1	0	0	1
	0	0	1	0	0	0	0	0	1	1

3.2 Vigenere Cipher

Basic Idea:

- The key, k, is an English word without repeated letters.
- Frequency distribution of ciphertext letterds is flatter comparing to simple substitution cipher. But still totally insecure against a chosen-plaintext attack and mostly against a ciphertext-only attack.

Breaking the Vigenere Cipher:

1. Find the key length l.
   (a) Make a guess for l.
   (b) Check your guess as follows.
      i. Divide the ciphertext letters into l groups, $G_0, G_1, ..., G_{l-1}$, where the ith ciphertext letter is placed in group $G_{i \ mod \ l}$.
      ii. Examine the frequency distributions of letters in each group.
      iii. If each distribution "looks like" the expected distribution of letters from sensible English text, then the key length guess is probably correct.
   (c) Once the key length l is determined, then,

i. Notice that the ciphertext letters in each group G_i were obtained by applying a permutation that is a cyclic shift of the alphabet to the corresponding plaintext letters.

ii. Use the frequency counts of ciphertext letters in G_i to make guesses for the ith letter of the key word.

iii. Use the guesses for the key letters to guess the key word (keeping in mind that it is an English word).

iv. Check your guess for the key word by decrypting the ciphertext.

Example: A=0, ..., Z=25; addition of letters is mod 26. k=CRYPTO.

Encryption:

+	m =	t	h	i	s	i	s	a	m	e	s	s	a	g	e
	k=	C	R	Y	P	T	O	C	R	Y	P	T	O	C	R
	c=	V	Y	G	H	B	G	C	D	C	H	L	O	I	V

,

Decryption is subtraction modulo 26.

3.3 One-Time Pad

Basic Idea:

- The one-time pad is the Vigenere cipher with a random string of letters as a key with the same length as the plaintext.)
- The one-time pad is semantically secure against a chosen plain attack if used properly, which means no key-reuse or non-random keys.

The key should not be re-used.
If $c_1 = m_1 + k$ and $c_2 = m_2 + k$, then $c_1 - c_2 = m_1 - m_2$. $c_1 - c_2$ depends only on the plaintext (and not on the key) and hence can leak information about the plaintext. If m_1 is known, then m_2 can be easily computed.

Definition: **Perfect secrecy** means a symmetric-key encryption scheme is **totally secure** against ciphertext-only attack by an adversary with infinite computational resources. The one-time pad is one of them.
However, since a re-used key can be broken, and a non-random key can be broken, perfect secrecy is useless in practice, which means the one-time pad is also useless.

Shannon proved that if plaintexts are m-bit strings , then any symmertric-key encryption scheme with perfect secrecy must have $|K| \geq 2^m$.

For the **binary messages** in the one-time pad encryption:

- Encryption: $c = m \oplus k$
- Decryption: $m = c \oplus k$

Example: A=0, ..., Z=25; addition of letters is mod 26. k is a random string of letters with the same length as the plaintext.
Encryption:

	m =	t	h	i	s	i	s	a	m	e	s	s	a	g	e
+	k=	Z	F	K	W	O	G	P	S	M	F	J	D	L	G
	c=	S	M	S	P	W	Y	P	F	Q	X	C	D	R	K

,

Decryption is subtraction modulo 26.

4 Stream Ciphers

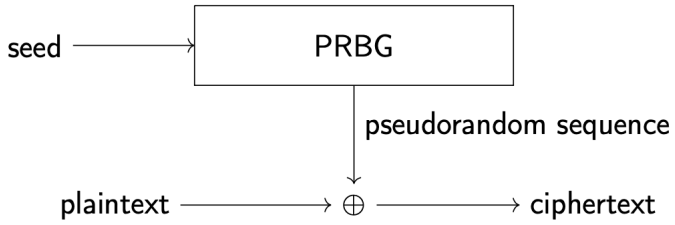
4.1 Basic Ideas

Basic Ideas

- A **stream cipher** is a symmetric-key encryption scheme the same as the one-time pad except using a pseudorandom key rather than an actual random key.
- Use a pseudorandom bit generator instead of using a random key in the one-time pad.

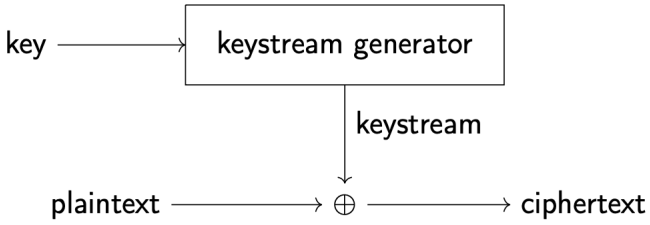
Definition: A **pseudorandom bit generator (PRBG)** is a deterministic algorithm that takes as input a (random) seed, and outputs a longer "pseudorandom" sequence called the **keystream**.

Definition: The **seed** is the secret key shared by users.



The security depends on the quality of the pseudorandom bit generator.

In the context of stream ciphers, we call PRBG as **keystream generator**.

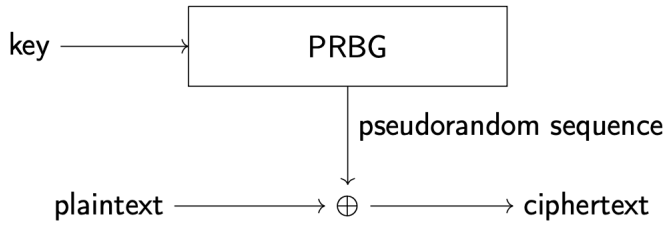


The **Security Requirements** of the PRBG **Keystream Generator**:

- Indistinguishability Requirement: The keystream should be indistinguishable from a randome sequence.
- Unpredictability Requirement: Given portions of the keystream, it should be infeasible to learn any information about the rest of the keystream.

  If an adversary knows a portion c_1 of ciphertext and the corresponding plaintext m_1, then she can easily find the corresponding portion $k_1 = c_1 \oplus m_1$ of the keystream.
- Do not use UNIX random number generators since the formula is fixed and public.

Keystream Generator with ONE input: The stream cipher has keystream generator with only one input.



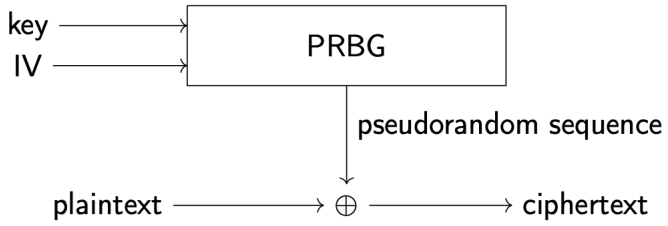
Pros:

- $Length(ciphertext) = Length(plaintext)$

Cons:

- One-to-one correspondence between keys and keystream. So the same key cannot be used to encrypt different ciphertexts.

Keystream Generator with TWO input: The stream cipher has keystream generator with two input: A key and an initialization vector (IV).



Pros:

- Can reuse the same key with different initialization vectors.

Cons:

- Need to transmit the initialization vector with the ciphertext ⇒ constant overhead

Example: **Frame-Based Encryption**

Frame-Based Encryption treats each communication as a series of frames/packets, and encrypt each one with a keystream formed from same key, but different frame number. Therefore, loss of synchronization only results in loss of remaining data in that packet, and next frame regains synchronization.

Example: A5/1 Algorithm

- Key: Each conversation encrypted using a different 64 bit key.
- Frame: Conversation broken into frames of length 228 bits.
- Keystream: Key is used with 22 bit frame number to produce 228 bits of keystream.

5 Linear Feedback Shift Registers (LFSR)

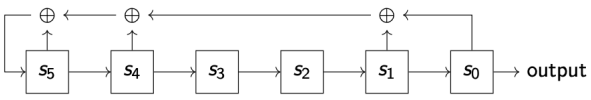
5.1 Basic Ideas

Definition: A **linear feedback shift register (LFSR)** is a common component of stream ciphers.

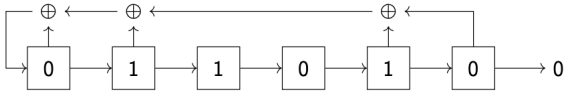
Any infinite periodic sequence (s_t) can be defined via a recurrence relation: $s_{t+n} = c_{n-1}s_{t+n-1} \oplus c_{n-2}s_{t+n-2}...c_1 s_{n+1} \oplus c_0 s_t$ where c_i are binary constants. The vector $(s_0, s_1, ..., s_{n-1})$ is called the **initial state vector**. The initial state vector together with the above equation defines all terms of the sequence.

Example: Suppose that the sequence (s_t) defined by the recurrence relation $s_{t+6} = s_{t+5} \oplus s_{t+4} \oplus s_{t+1} \oplus s_t$. Some example terms: $s_6 = s_5 \oplus s_4 \oplus s_1 \oplus s_0$, $s_7 = s_6 \oplus s_6 \oplus s_2 \oplus s_1$ and $s_8 = s_7 \oplus s_6 \oplus s_3 \oplus s_2$ and so forth.

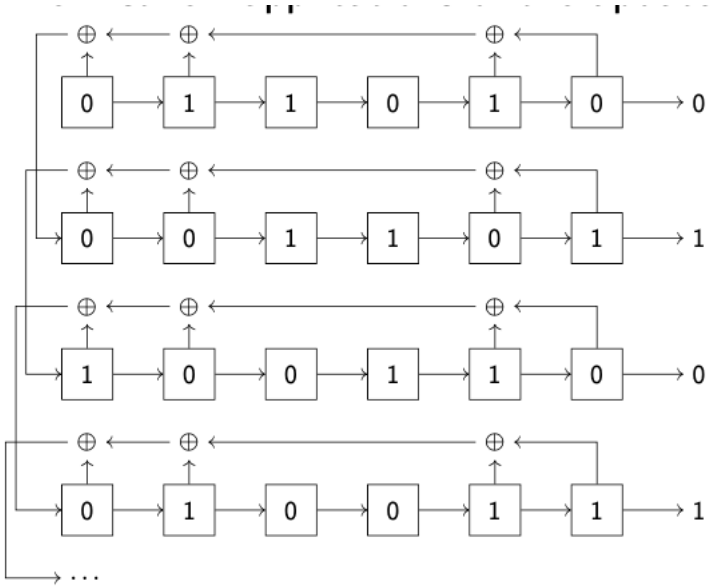
The LFSR diagram for this sequence looks like this:



If the LFSR has the initial state $(0, 1, 0, 1, 1, 0)$, then the initial setting of the LFSR is



The first few applications of the update function are as below:



The first period of the output sequence is
01011001010100100111100000110111001100011101011111101101000100 0.

There are two basic methods using an LFSR to **form a keystream for a stream cipher**.

1. Multiple outputs from the sequence can be sampled and passed through a non-linear filter.

   Example: The CRYPT01 cipher used in the MIFARE Classic contactless smart card in the Brisbane Translink Go card. CRYPT01 takes multiple outputs from one LFSR.

2. The outputs from multiple LFSRs can be passed through a non-linear combiner.

   Example: The A5 cipher used in the GSM mobile phone standard.

5.2 The Strength of a Keystream Generator

The strength of a keystream generator scales by:

1. large period
2. large linear complexity
3. random noise-like characteristics

5.2.1 Period of LFSR

A deterministic pseudorandom binary number generator produces a periodic sequence meaning it repeats after a while.

Definition: **The period of a binary sequence** $s(t)$ is said to be p if $s_{t+p} = s_t$ for all t, and p is the smallest such number.

Example: The sequence 011101110111101110111... has a period of 4.

For a good stream cipher we expect that the period of the keystream is larger than the length of plaintext.
If the period of the keystream is less than the length of a ciphertext, then two sections of message are encrypted using the same portion of keystream. By CORing these two sections, the keystream cancels and one obtains the XOR of the plaintext strings. It is possible to attack such a sum by exploiting the redundancy of the plaintext.

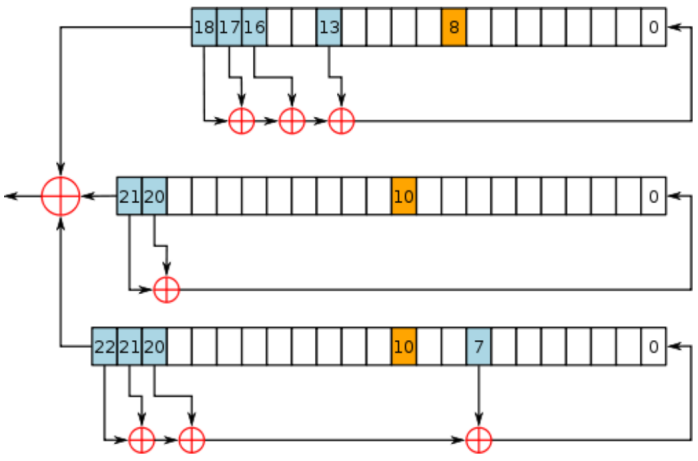
Example: **A5 Cipher and its Variants**

The **A5 cipher** is a binary synchronous stream cipher applied in most GSM mobile telephones. It has three variants: A5/1, A5/2 and A5/3. A5/1 is the original algorithm. A5/2 is a weakened version and no longer used. A5/3 is the latest one.

Definition: **A5/1 Design**

The **A5/1 algorithm** uses three linear feedback shift registers whose output is combined.

The three LSFRs are irregularly clocked which means that the overall output is non-linear. Each LFST has a clocking bit. Each LFSR is clocked if its own clocking bit is in majority agreement with the clocking bits of the other LFSRs.



Orange: Clocking bit.

Example: **The RC4 Stream Cipher**

- RC4 has two components:
  - A key scheduling algorithm
  - A keystream generator
- Do not use RC4. Instead use AES-CTR or Salsa 20/ChaCha20.

Definition: **RC4 Key Scheduling Algorithm**

- Input: Secret key $K[0]$, $K[1]$,...,S[255]. (Key length is 8d bits.)
- Output: 256-long array: $S[0]$, $S[1]$,...,$S[255]$.

```
for i from 0 to 255 do
    S[i] ← i
    K̄[i] ← K[i mod d]
end for
j ← 0
for i from 0 to 255 do
    j ← (K̄[i] + S[i]+j) mod 256
    Swap(S[i], S[j]
end for
```

- Idea: S is a random-looking permutation of $\{0, 1, 2, ..., 255\}$ that is generated from the secret key.

Definition: **RC4 Keystream Generator**

- Input: 256-long byte array: $S[0]$, $S[1]$, ..., $S[255]$ produced by the RC4 Key Scheduling Algorithm.
- Output: Keystream.

```
i ← 0; j ← 0
while keystream bytes are required do
    i ← (i+1) mod 256
    j ← (S[i]+j) mod 256
    Swap(S[i], S[j])
    t ← (S[i]+S[j]) mod 256
    Output(S[t])
end while
```

- Encryption: The keystream bytes are XORed with the plaintext bytes to produce ciphertext bytes.

6 Wired Equivalent Privacy (WEP)

6.1 Basic Ideas

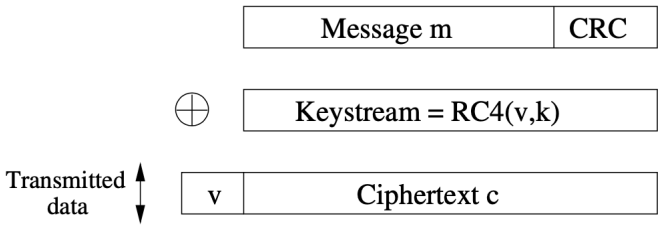
Definition: **Wired Equivalent Privacy (WEP)** is a protocol included in IEEE 802.11 standard for wireless LAN communications.

It is designed to protect link-level data during wireless transmission between mobile stations and access point. Mobile station shares a secret key k with access point. k is either 40 bits or 104 bits in length. In practice, one shared key per LAN is common. This key is manually injected into each access point and mobile station.

The **Goals** of Wired Equivalent Privacy:

1. Confidentiality: Prevent casual eavesdropping.
2. Data Integrity: Prevent tampering with transmitted messages.
3. Access Control: Protect access to a wireless network infrastructure.

6.2 How Wired Equivalent Privacy Protocol Works



1. Messages are divided into packets of some fixed length. Wired Equivalent Privacy uses a per-packet 24-bit initialization vector v to process each packet.
   Wired Equivalent Privacy does not specify how the initialization vectors are managed. In practice, a random initialization vector is generated for each packet, or the initialization vector is set to 0 and incremented by 1 for each use.
2. Send the packets.
   (a) Select a 24-bit initialization vector v.
   (b) Compute a 32-bit checksum: $S = CRC(m)$. CRC-32 is linear. That is, for any two messages m_1 and m_2 of the same bitlength, $CRC(m_1 \oplus m_2) = CRC(m_1) \oplus CRC(m_2)$.
   (c) Compute $c = (m||S) \oplus RC4(v||k)$. $||$ denotes concatenation. $(v||k)$ is the key used in the RC4 stream cipher.

(d) Send $\underline{(v, c)}$ over the wireless channel.

3. The receiver would compute $\underline{(m||S = c \oplus RC4(v||k))}$ and $\underline{S' = CRC(m)}$. The packet will be rejected if $S' \neq S$.

6.3 The Problems of Wired Equivalent Privacy

1. **No High Degree of Confidentiality: Initialization Vector Collision**

   Suppose that two packets (v, c) and (v, c') use the same initialization vector v. Let m, m' be the corresponding plaintexts. Then $c \oplus c' = (m||S) \oplus (m'||S')$. The eavesdropper can compute $m \oplus m'$.

   - If m is known, then m' is immediately available.
   - If m is not known, then one may be able to use the expected distribution of m and m' to discover information about them.

   Since there are only 2^{24} choices for the IV, collisions are guaranteed after enough time. If initialization vectors are randomly selected, then one can expect a collision after about 2^{12} packets. Collisions are more likely if keys k are long-lived and the same key is used for multiple mobile stations in a network.

2. **No Data Integrity: Checksum is Linear**

   CRC-32 is used to check integrity. This is fine for random errors, but not for deliberate ones.

   It is easy to make controlled changes to (encrypted) packets:

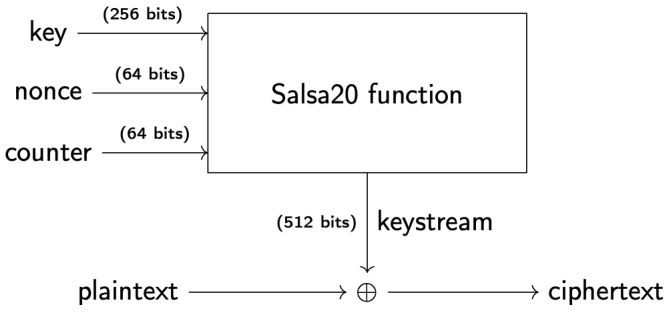
   - Suppose (v, c) is an encrypted packet.
   - Let $c = RC4(v||k) \oplus (m||S)$, where k, m, S are unknown.
   - Let $m' = m \oplus \triangle$, where $\triangle$ is a bit string. (The 1's in $\triangle$ correspond to the bits of m an attacket wishes to change.)
   - Let $c'c \oplus (\triangle||CRC(\triangle))$.
   - Then (v, c') is a valid encrypted packet for m'.

3. **No Access Control: Integrity Function is Unkeyed**

   Suppose that an attacker learns the plaintext m corresponding to a single encrypted packet (v, c). Then, the attacker can compute the RC4 keystream $RC4(v||k) = c \oplus (m||CRC(m))$. Henceforth, the attacker can compute a valid encrypted packet for any plaintext m' of her choice: (v, c'), where $c' = RC4(v||k) \oplus (m'||CRC(m'))$. Therefore, WEP does not provide access control.

7 Salsa20

Basic Ideas: Salsa20 is a modern stream cipher.



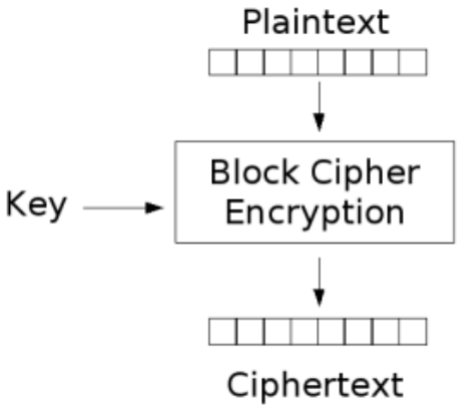
How Salsa20 Works

1. Load the key, nonce, and counter into the state.
2. Make a copy of the initial state.
3. Do the following time times.
4. XOR the current state with the copy of the initial state.
5. Output that as the key stream.

8 Block Ciphers

8.1 Block Ciphers

Basic Idea: A **block cipher** a symmetric-key encryption scheme in which a fixed-length block of plaintext determines an equal-sized block of ciphertext.



Design Principles of Block Ciphers:

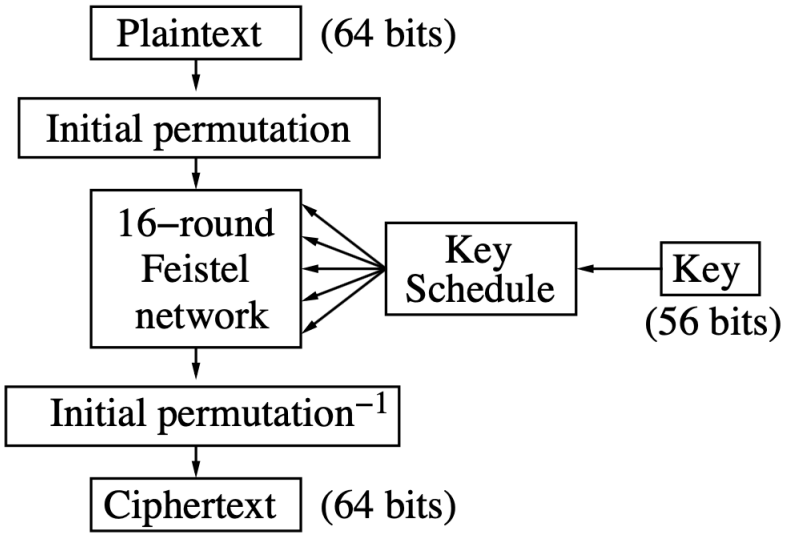
- Security
  - Diffusion: Each ciphertext bit should depend on all plaintext bits.
  - Confusion: The relationship between key and ciphertext bits should be complicated.
  - Key length: Should be small, but large enough to preclude exhaustive key search.
- Efficiency
  - Simplicity
  - High encryption and decryption rate
  - Suitability for hardware or software

8.2 DES

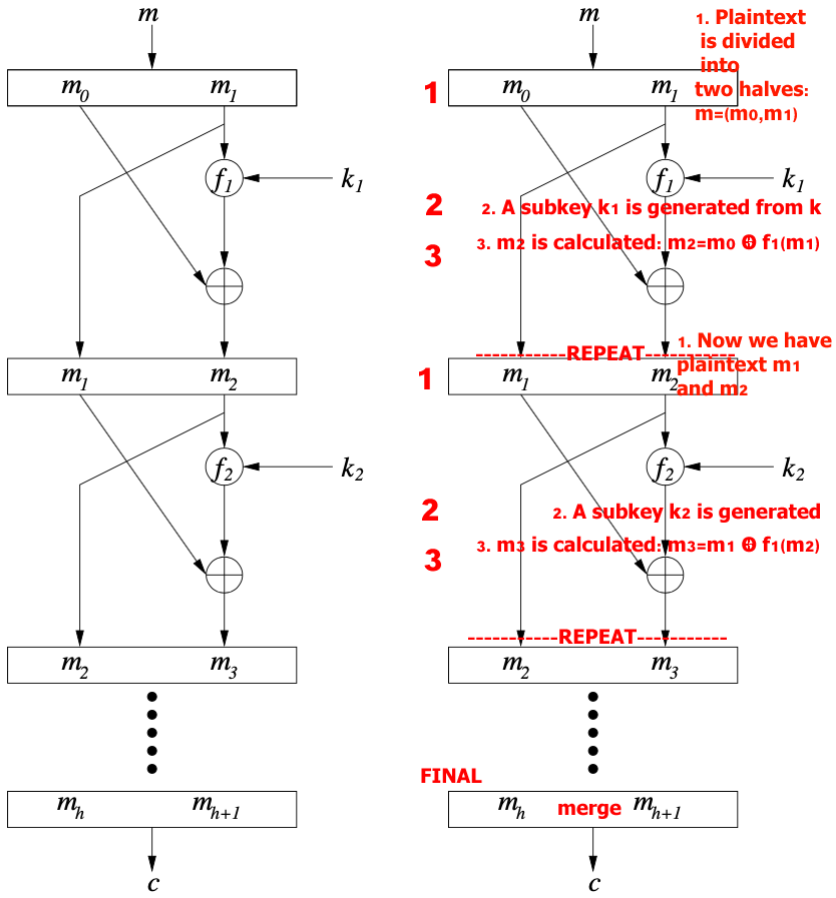
8.2.1 DES

Basic Idea:

- **DES** is a block cipher with 64-bit blocks, 56-bit key, and 16 rounds of operation.
- DES is secure against everything except brute-force key search.



8.2.2 The Feistel Network Design & Feistel Ciphter



NO Description & WITH Description

m: Plaintext
$k_n, 1 \leq n \leq h$: A subkey generated by the key k
f_i: A component function whose output value depends on k_i and m_i

How a Feistel Cipher Works: Encryption takes h hounds.

- Parameters:
  - n: half the block length
  - h: number of rounds
  - l: key size
  - $M = \{0,1\}^{2n}$: the set of messages
  - $C = \{0,1\}^{2n}$: the set of ciphers

- $K = 0, 1^l$: the set of keys

- Encryption

  Plaintext is $m = (m_0, m_1)$, where $m_i \in \{0,1\}^n$.

  1. $(m_0, m_1) \to (m_1, m_2)$, where $m_2 = m_0 \oplus f_1(m_1)$
  2. $(m_1, m_2) \to (m_1, m_2)$, where $m_2 = m_0 \oplus f_1(m_1)$
     ...
  3. $(m_{h-1}, m_h) \to (m_1, m_2)$, where $m_2 = m_0 \oplus f_1(m_1)$
  4. $(m_0, m_1) \to (m_1, m_2)$, where $m_2 = m_0 \oplus f_1(m_1)$

  The ciphertext is $c = (m_h, m_{h+1})$.

  Only need to implement one round; the same code can be used for each round.

- Decryption

  Given $c = (m_h, m_{h+1})$ and k, to find $m = (m_0, m_1)$.

  Compute $m_{h-1} = m_{h+1} \oplus f_h(m_h)$.

  Similarly, compute $m_{h-2}, ..., m_1, m_0$.

  No restrictions on the function f_i in order for the encryption procedure to be invertible.

  Decryption uses the same code as for encryption. (Use subkeys in reverse order).

DES uses a Feistel Network Design. $n = 32$, $h = 16$, $l = 56$

8.2.3 DES S-Boxes

Definition: Each **Substitution-Boxes** or **S-Box** s a function taking six input bits and producing four output bits. S-boxes are the only components of DES that are non-linear.

Security of DES crucially depends on their choice. DES with randomly selected S-boxed is easy to break.

8.2.4 DES Problems

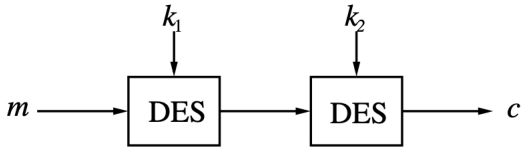
1. Small Key Size: Exhaustive search on key space take 2^{56} steps and can be easily parallelized.
2. Small Block Size:
   - If plaintext blocks are distributed "uniformly at random", then the expected number of ciphertext blocks observed before a collision occurs is $\approx 2^{32}$. Hence the ciphertext reveals some information about the plaintext.
   - Damaging to some authentication applications.

8.2.5 Multiple Encryption

Definition: **Multiple encryption** means re-encrypting the ciphertext one or more times using independent keys to increase the effective key length. (Multiple encryption does not always increase security.)

Double-DES: DES with double encryption.

- Key: $k = (k_1, k_2)$, k_1, $k_2 \in R\{0,1\}^{56}$.
  - Key size of Double-DES l is 112. So exhaustive key search takes 2^{112} steps.
- Encryption: $c = E_{k_2}(E_{k_1(m)})$. (E = DES encryption)
- Decryption: $m = E_{k_1}^{-1}(E_{k_2}^{-1}(c))$. ($E^{-1}$ = DES decryption)



Attack on Double-DES

Main Idea: Complexity of this attack is $\approx 2^{57}$.
If $c = E_{k_2}(E_{k_1}(m))$, then $E_{k_2}^{-1}(c) = E_{k_1}(m)$.

1. Given: Known plaintext pairs (m_i, c_i), $i = 1, 2, 3, ...$
2. For each $h_2 \in \{0,1\}^{56}$,
   (a) Compute $E_{h_2}^{-1}(c_1)$, and store $[E_{h_2}^{-1}(c_1), h_2]$ in a table.
3. For each $h_1 \in \{0,1\}^{56}$ do the following:
   (a) Compute $E_{h_1}(m_1)$.
   (b) Search for $E_{h_1}(m_1)$ in the table.
   (c) If $E_{h_1}(m_1) = E_{h_2}^{-1}(c_1)$
      i. Check if $E_{h_1}(m_2) = E_{h_2}^{-1}(c_2)$
      ii. Check if $E_{h_1}(m_3) = E_{h_2}^{-1}(c_3)$
      ...
      If all checks pass, then output (h_1, h_2) and STOP.

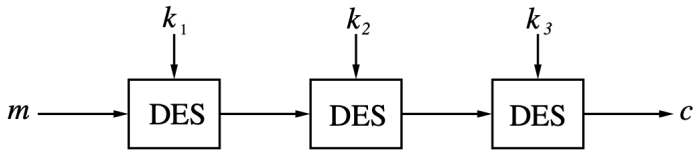
Analyzing the Meet-in-the-Middle Attack to Double-DES

- Number of known plaintext/ciphertext pairs required to avoid false keys: 2 suffice, with high probability.
- Number of DES operations is $\approx 2^{56} + 2^{56} + 2 \cdot 2^{48} \approx 2^{57}$.
- Space requirements $2^{56}(64 + 56)$ bits $\approx 1,080,863$ Tbytes.
- Time-Memory Tradeoff: The attack can be modified to decrease the storage requirements at the expense of time.
  - Time: 2^{56+s} steps
  - Memory: 2^{56-s} units, $1 \leq s \leq 55$.

Therefore, Double-DES has the same effective key length as DES, and not much more secure than DES.

Triple-DES: DES with triple encryption. No proof that Triple-DES is more secure than DES though.

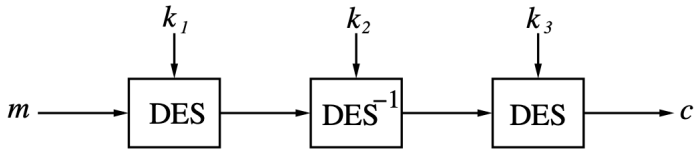
- Key: $k = (k_1, k_2, k_3)$, k_1, $k_2, k_3 \in R\{0,1\}^{56}$.
  - Key size of triple-DES l is 168. So exhaustive key search takes 2^{168} steps.
- Encryption: $c = E_{k_3}(E_{k_2}(E_{k_1}(m)))$. ($E$ = DES encryption)
- Decryption: $m = E_{k_1}^{-1}(E_{k_2}^{-1}(E_{k_3}^{-1}(c)))$. ($E^{-1}$ = DES decryption)



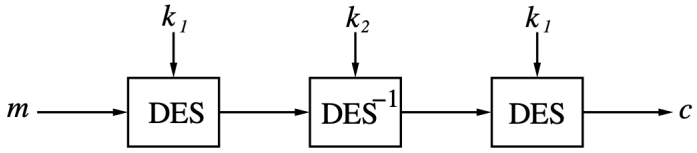
Analyzing the Meet-in-the-Middle Attack to Triple-DES

- Meet-in-the-middle attack takes $\approx 2^{112}$ steps. So the effective key length of Triple-DES against exhaustive key search is ≤ 112 bits.
- Block length is 64 bits, and now forms the weak link. Adversary stores a large table of size $\leq 2^{64}$ of (m, c) pairs (dictionary attack). To prevent this attack, change secret keys frequently.

EDE Triple-DES: for backward compatibility with DES



Two-Key Triple DES



8.2.6 Substitution-Permutation Networks

Definition: A **substitution-permutation network** is a type of iterated block cipher where a round consists of

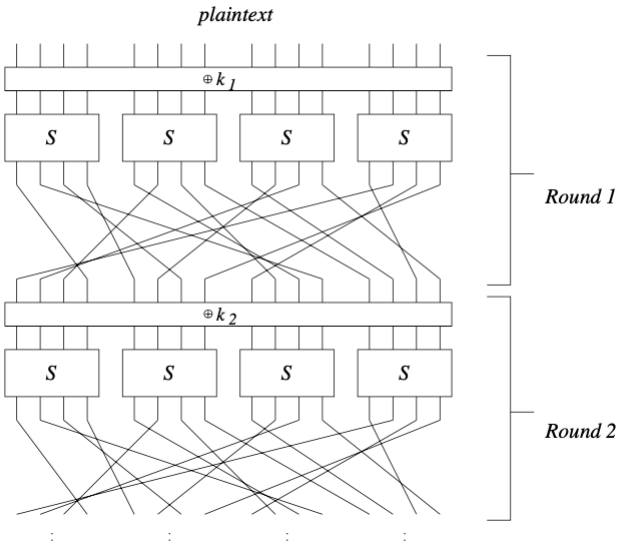
- A substitution operation, followed by
- A permutation operation.

8.3 AES: The Advanced Encryption Standard

8.3.1 Substitution-Permutation Networks

Definition: A **substitution-permutation network (SPN)** is a multiple-round iterated block cipher where each round consists of

1. a substitution operation, followed by
2. a permutation operation.



AES is an SPN where the permutation operation consists of two linear transformations, one of which is a permutation.

9 Differential Cryptanalysis

9.1 Basic Ideas

operation of the attack

- P and P': plaintexts
- C and C': encryptions

The idea is to look for value $\triangle C = C \oplus C'$ which occur with abnormally high probability for a given $\triangle P = P \oplus P'$.

10 Encryption Bulk Data

10.1 Basic Ideas

Block Ciphers versus Stream Ciphers

	Stream Cipher	**Block Cipher**
Definition	a symmetric-key encryption scheme in which each successive character of plaintext determines a single character of ciphertext	a symmetric-key encryption scheme in which a fixed-length block of plaintext determines an equal-sized block of ciphertext
Encrypting Large Quantities of Data	Encrypt each character	There are some complication if 1. the input is larger than one block 2. the input does not fill an integer number of blocks

Definition: We use **a mode of operation**, which means a specification for how to encrypt multiple and/or partial data blocks using a block cipher, to deal the problems above.

Some modes require the plaintext to consist of one or more complete blocks.

NIST Special Publication 800-30A suggests a padding method as follows.

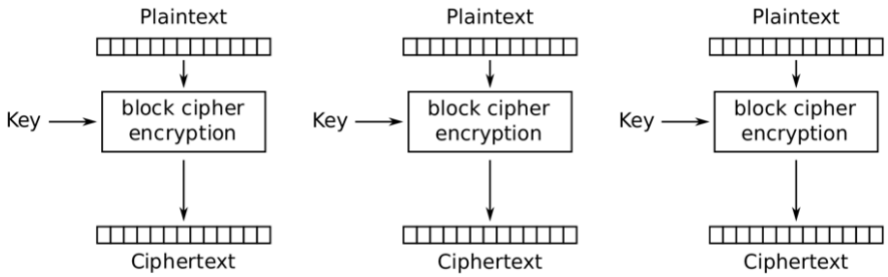
1. append a single '1' bit to the data string.
2. pad the resulting string by as few '0' bits, possibly none, as are necessary to complete the final block.

The padding bits can be removed unambiguously, if the receiver knows that this padding method is used.

1. remove all trailing '0' bits after the last '1' bit.
2. remove a single '1' bit.

10.2 Electronic Codebook Mode (ECB Mode)

This approach is to encrypt each l bits independently, where l is the block size.



Electronic Codebook (ECB) mode encryption

Electronic Codebook mode is the most basic mode of a block cipher.

- **Encryption**
  $C_t = E(P_t, K)$: Plaintext block P_t is encrypted with the key K to produce ciphertext block C_t.
- **Decryption**
  $P_t = D(C_t, K)$: Ciphertext block C_t is decrypted with the key K to produce plaintext block P_t.

Steam ciphers are secure but **block ciphers** in ECB mode are **insecure**.

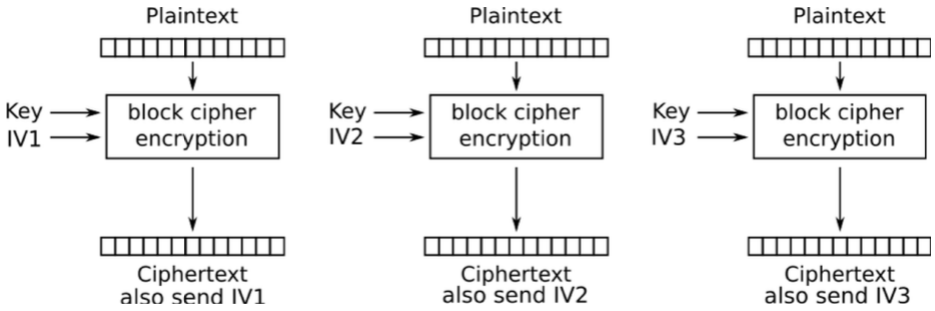
- Stateless: A block cipher, unlike a stream cipher, is stateless.
- ECB mode is equivalent to a giant substitution cipher where each l-bit block is a character.
- Semantic security is immediately violated: one can tell by inspection whether or not two clocks of ciphertext correspond to identical plaintext blocks.

Randomised	$\times$
Padding	Required
Error Propagation	Errors propagate within blocks
IV	None
Parallel Encryption?	$\surd$
Parallel Decryption?	$\surd$

Because it is deterministic, ECB mode is not normally used for bulk encryption.

Adding IVs

Ideas: Use a new (non-secret) initialization vector for each block of plaintext.
Problem: Have to send an IV for each block of ciphertext, adding linear.

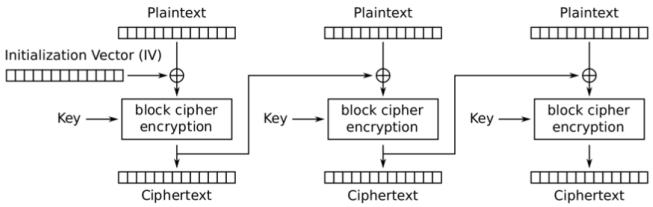


10.3 Cipher Block Chaining (CBC) Mode

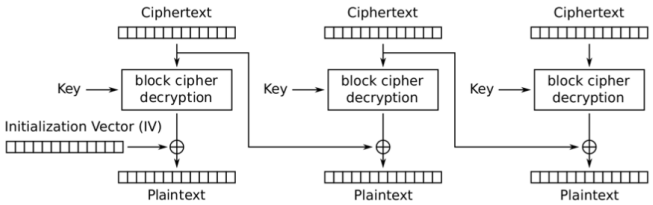
Ideas: Use a single non-secret initialize vector for the first block of plaintext, and "chain"the pseudorandom ciphertext blocks as the next block' initialization vectors.

Cipher block chaining chains the clocks together.

A random initialization vector IV is chosen and sent together with the ciphertext blocks.



Cipher Block Chaining (CBC) mode encryption

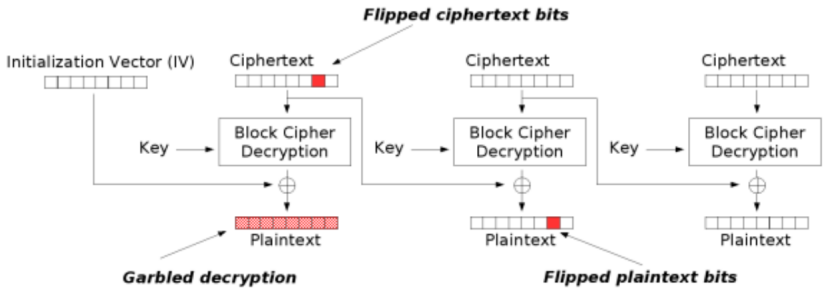


Cipher Block Chaining (CBC) mode decryption

- **Encryption**
  $C_t = E(P_t \oplus C_{t-1}, K)$, where $C_0 = IV$: P_t is XOR'd with the previous ciphertext block C_{t-1}, and encrypted with key K to produce ciphertext block C_t. For the first plaintext block IV is used for the value C_0.

- **Decryption**
  $P_t = D(C_t, K) \oplus C_{t-1}$, where $C_0 = IV$: C_t is decrypted with the key K, and XOR'd with the previous ciphertext block C_{t-1} to produce plaintext block P_t. As in encryption, IV is used in place of C_0.

CBC Mode Error Propagation



Modification attack or transmission error for CBC

Randomised	√
Padding	Required
Error Propagation	Errors propagate within blocks and into specific bits of next block
IV	Must be random
Parallel Encryption?	×
Parallel Decryption?	√

CBC mode is commonly used for bulk encryption and is supported in most libraries and protocols.

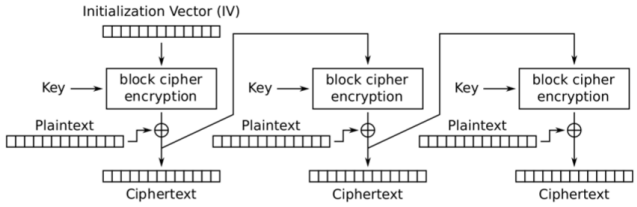
10.4 Cipher Feedback (CFB) Mode

Cipher Feedback Mode feeds the ciphertext block back into the enciphering/deciphering process, thus "chaining"the blocks together.

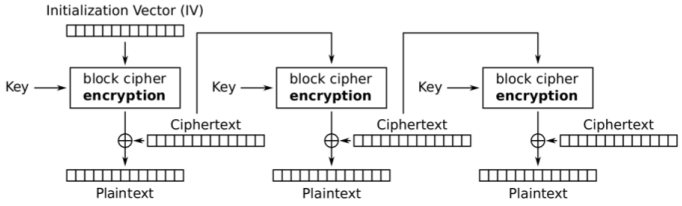
- **Encryption**
  $C_t = E(C_{t-1}, K) \oplus P_t$, where $C_0 = IV$

- **Decryption**
  $P_t = E(C_{t-1}, K) \oplus C_t$

- **Propagation of Channel Errors**
  A one-bit change in C_t produces a noe-bit change in P_t, and complete corruption of P_{t+1}.



Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

CFB is a self-synchronizing stream cipher

- Keystream depends on previous ciphertexts, which allows CFB mode to self-synchronize after processing a correct ciphertext block.
- Assume block C_t is lost in transmission, producing a loss in synchronicity between sender and receiver.
- Receiver decrypts next received block C_{t+1} as $E(C_{t-1}, K) \oplus C_{t+1}$, which is incoreect (i.e. different from P_t).
- For the next received block C_{t+2} the receiver computes $E(C_{t+1}, K) \oplus C_{t+2}$. which is the correct plaintext block P_{t+2}. The cipher is back in sync (after losing P_t and P_{t+1}.

The underlying block cipher is only used in encryption mode.

Randomised	$\surd$
Padding	Not Required
Error Propagation	Errors propagate within blocks and into specific bits of next block
IV	Must be random
Parallel Encryption?	$\times$
Parallel Decryption?	$\surd$

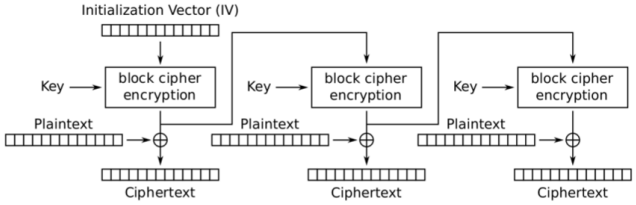
CFB mode is commonly used when self-synchronization is useful.

10.5 Output Feedback (OFB) Mode

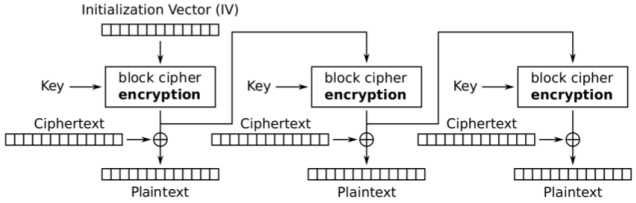
Output Feedback Mode is a synchronous cipher that feeds the output block back into the enciphering/deciphering process. The keystream is $O_t = E(O_{i-1}, K)$ where $O_0 = IV$ is chosen at random.

- **Encryption**
  $C_t = O_t \oplus P_t$
- **Decryption**
  $P_t = O_t \oplus C_t$
- **Propagation of Channel Errors**
  A one-bit change in the cipher produces a one-bit change in the plaintext at the same location.

The underlying block cipher is only used in encryption mode.



Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption

OFB mode is a synchronous stream cipher mode.

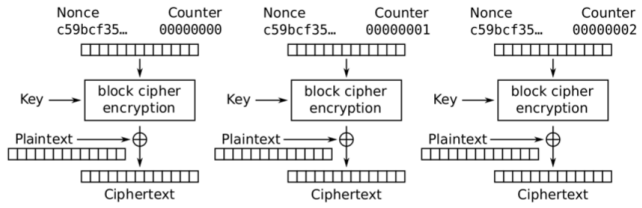
Randomised	$\surd$
Padding	Not Required
Error Propagation	Errors occur in specific bits of current block
IV	Must be unique
Parallel Encryption?	$\times$ (But keystream can be computed in advance)
Parallel Decryption?	$\times$ (But keystream can be computed in advance)

10.6 Counter (CTR) Mode

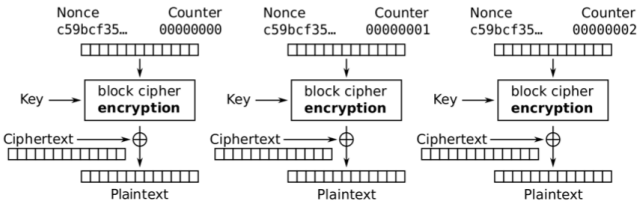
CTR is a synchronous stream cipher. The keystream is generated by encrypting successive values of a counter, initialised using a nonce (randomly chosen value) N: $O_t = E(T_t, K)$, where $T_t = N||t$ is the concatenation of the nonce and block number t.

- **Encryption**
  $C_t = O_t \oplus P_t$
- **Decryption**
  $P_t = O_t \oplus C_t$
- **Propagation of Channel Errors**
  A one-bit change in the ciphertext produces a one-bit change in the plaintext at the same location.

Choose a nonce at random during encryption. Pretend the nonce to the ciphertext.



Counter (CTR) mode encryption



Counter (CTR) mode decryption

The underlying block cipher is only used in encryption mode.

Randomised	√
Padding	Not Required
Error Propagation	Errors occur in specific bits of current block
IV	Nonce must be unique
Parallel Encryption?	√
Parallel Decryption?	√

CTR mode is a synchronous stream cipher mode and also good for access to specific plaintext blocks without decryption the whole stream.

10.6.1 Summary

	ECB	CBC	CFB	OFB	CTR
Randomised	√				
Padding	Not Required				
Error Propagation	Errors occur in specific bits of current block				
IV	Nonce must be unique				
Parallel Encryption?	√				
Parallel Decryption?	√				