The University of Sydney
School of Computer Science
Dr Suranga Seneviratne
Senior Lecturer - Security

THE UNIVERSITY OF SYDNEY

Symmetric Cryptography

Recommended Reading

Cryptography and Network Security (7^{th} Edition - William Stallings)

- **Chapter 2** – Introduction to Number Theory
- **Chapter 3** – Classical Encryption Techniques
- **Chapter 4** – Block Ciphers and Data Encryption Standards
- **Chapter 5** – Finite Fields
- **Chapter 6** – Advanced Encryption Standards
- **Chapter 7** – Block Cipher Operation
- **Chapter 8** – Random bit Generation and Stream Ciphers

Also, please read the separate handouts given for cryptography-related math concepts.

These lecture notes are given to you to assist with understanding the lecture content better. This content is prepared based on the above book chapters. You are not allowed to upload this material to any internet source or share it with anyone else.

1 Introduction

Cryptography is an indispensable tool in the design of secure protocols and systems. It is used to ensure:

- **Data Confidentiality:** Achieved through encryption.
- **Data Integrity:** Ensured by signatures and Message Authentication Codes (MACs).
- **Authentication and Data Origin Verification:** Accomplished using signatures and Message Authentication Codes (MACs).
- **Non-repudiation:** Implemented with signatures.

A fundamental understanding of cryptography principles is crucial for many careers in computing.
In this unit, we introduce cryptography from a functional point of view with less emphasis on mathematical or formal perspectives (i.e., proofs). However, we cover a significant amount of basic math you need to know to understand foundational cryptographic concepts. Before you read this, please make sure that you read and understand the introduction to number theory and abstract algebra lecture notes.

1.1 Challenges in Cryptography

Cryptography is a double-edged sword, complex and full of subtleties. Inconsiderate application of cryptographic techniques can lead to insecure systems. Worse, these systems may appear secure to developers and users while being vulnerable to attackers. An example is the historical GSM A5/1 algorithm.

GSM A5/1 algorithm

The GSM-based second-generation cellular systems in 90's used the secret and proprietary A5 stream cipher design. This algorithm uses a 64-bit secret key and has two main variants. The A5 ciphers are responsible for encryption of the data transmitted between the mobile device and the Base Transceiver Station (BTS).

Historical records show that there were some serious cryptographic weaknesses found in 1994. By the late 90s, multiple works showed that combinations of cryptanalytic and hardware attacks could be used to conduct practical attacks against the A5/1 algorithm. The specific internal structure of the GSM communication protocol allowed the possibility of conducting the most powerful cryptographic attack, a **ciphertext-only attack**. Here, an adversary that eavesdrops a communication and gathers a sequence of consecutive encrypted frames can recover the secret key with moderate computational power.

Source: The (in)security of proprietary cryptography by Roel Verdult [**?**].

1.1.1 Implementation Challenges

Implementing cryptography requires extensive experience and knowledge of hardware and programming languages. Therefore, it is advised:

- Do not implement your own cryptography for real-world applications. Always use known and standardised methods.
- To become proficient, practice, find a mentor, and engage in discussions with other implementers. It is a long career path with no shortcuts.

1.2 Definition of Cryptography

Cryptography is the science of securely transmitting data over space and time.

- **Spatially:** Ensuring data security over physical distances. For example, this means securing communication between one of your browser tabs and a remote web server such as `www.google.com`.
- **Temporally:** Keeping data secure for a long duration into the future. For example, this might involve encrypting sensitive archival data today in a way that it remains secure and unreadable even decades from now, despite advancements in computing power and cryptographic techniques.

1.3 Core Topics in This Unit

In this unit, we focus on the following core aspects of cryptography:

- **Confidentiality:** Achieved through encryption methods.
  - Symmetric encryption
  - Asymmetric (public-key) encryption
- **Integrity:** Ensured through both symmetric and asymmetric methods.
  - Message Authentication Codes (MACs)
  - Signatures
  - Hash functions
- **Authentication:** Implemented through various protocols.

While there are other applications of cryptography such as for achieving Non-repudiation, Privacy, Anonymity, those are not covered in this unit.

2 Cryptography - Definitions

Basic Definitions and Notations

We can define the following components of a crypto-system.

Plaintext: Plain-text p is the non-encrypted, unprotected data. It is built from the symbols from the alphabet $R = \{r_1, r_2, ..., r_n\}$. For example, common English comes from the alphabet {[A-Z],[a-z],[0-9]}.

Encryption: The encryption process encodes plaintext using the cipher function, Enc and a key k_e and produces the ciphertext c. That is $Enc_{k_e}(p) = c$.

Ciphertext: Ciphertext c may not necessarily come from the same alphabet as the plain text. It may not even be of the same size as the plaintext alphabet. We denote the ciphertext alphabet as $S = \{s_1, s_2, ..., s_m\}$.

Decryption: The decryption process decodes ciphertext using the decipher function, Dec and a key k_d and produces back the plaintext p. That is $Dec_{k_d}(c) = p$.

Symmetric cryptography - In symmetric cryptography, the encryption key is the same as the decryption key, i.e., $k_e = k_d$, or k_d can be derived from k_e. This is also known as **shared key cryptography**. Symmetric cryptography algorithms usually shuffle symbols around and map them to others.

Asymmetric cryptography - In asymmetric cryptography, the encryption key is different from the decryption key, i.e., $k_e \neq k_d$ or it is infeasible to derive k_d from k_e. This is also known as **public key cryptography**. Asymmetric cryptography algorithms are based on the hardness of some mathematical problems.

3 Kerkhoff's Principle

We have discussed the Kerkhoff's Principle several times by now in our discussions on why open design is important in cybersecurity. Formally Kerkhoff's Principle can be defined as follows.

Kerkhoff's Principle

The security of a given cryptographic mechanism must depend only on the security of the cryptographic keys used, but never on the mechanism or anything else, except the keys, being a secret.

In fact, almost all the mainstream cryptographic algorithms that are used in practice are publicly disclosed. For example, we know exactly how AES (Advanced Encryption Standard) or RSA (Rivest–Shamir–Adleman) work. Their security is purely dependent on the secrecy of the used keys. Why this is important is because when the design is disclosed, many researchers and engineers can look into the algorithm and find possible flaws. In fact, when NIST (National Institute of Standards and Technology) runs competitions to establish new cryptographic standards, they invite the global cryptographic community to submit algorithms for public review and scrutiny. This process ensures that the winning algorithms have undergone rigorous analysis and testing by experts worldwide. The transparency of these competitions, like the one that led to the selection of AES, helps to ensure that the chosen standards are robust, secure, and free from hidden vulnerabilities.

There have been many historical examples of not following the Kerkhoff's principle end in disasters. We already discussed the GSM A5/1 algorithm earlier. Another example is what happened with the Content Scrambling System of DVDs.

DVD Content Scrambling System (CSS)

The CSS was put in place to prevent DVDs made for one region of the world being played in another region. It also prevented copying of DVD content, whether legal or illegal, from the DVD to any other media. It also prevented (due to the non-disclosure of the algorithm to the public) viewing of movies and other content on the DVDs on software or hardware DVD players not approved by the Copy Control Association (CCA) [**?**].

CSS relied on the secrecy of its algorithm for security. The DVD Consortium assumed that by keeping the algorithm secret, the system would remain secure. However, once the algorithm was reverse-engineered and made public, the entire copyright system was compromised.

4 Symmetric Cipher Model

Figure 1 shows the schematic overview of the symmetric cipher model, which is the focus of this lecture. Note here that the Encryption and the Decryption keys are the same.

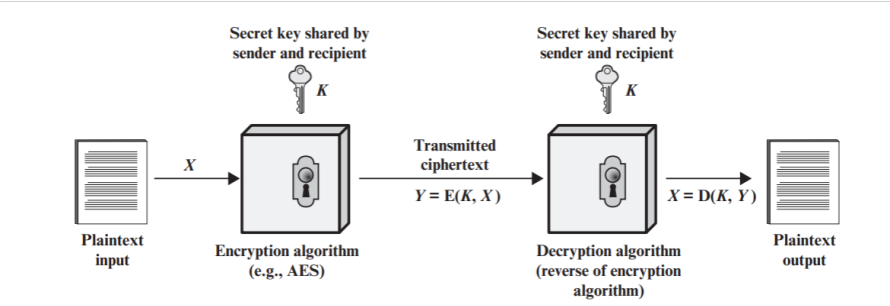


Figure 1: Basic operation of symmetric key cryptography [1])

5 Historical Ciphers

5.1 Traditional Concepts

Transposition and **substitution** are two fundamental techniques used in historical symmetric ciphers. **Transposition** involves shifting the positions of symbols in a message according to a specific rule, thereby altering the order of the symbols without changing the symbols themselves. **Substitution**, on the other hand, replaces symbols with other symbols based on a predetermined rule. Both transposition and substitution are typically governed by a key that parameterizes the rule, ensuring that only those with the correct key can correctly decipher the message (e.g., the shift). It's also common for transposition and substitution to be combined within the same cipher, enhancing the complexity and security of the encryption. Figure 2 shows two basic examples of substitution and transposition.

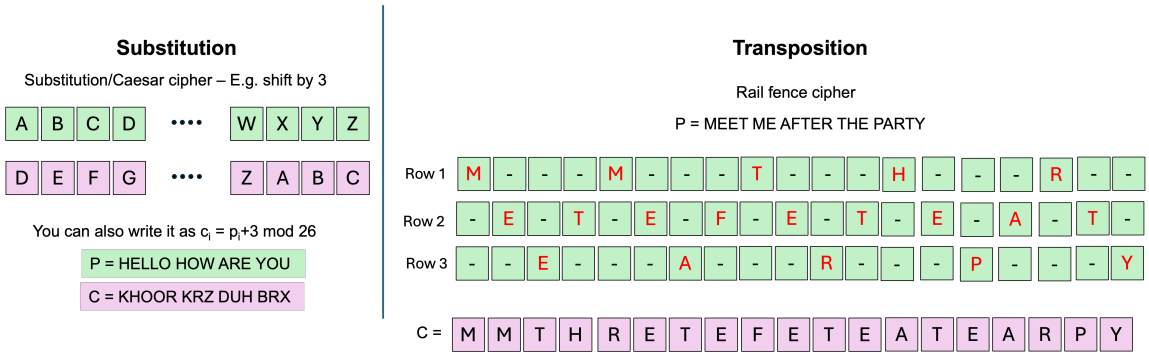


Figure 2: Examples for substitution and transposition

5.2 Breaking the Substitution/Caesar cipher

The substitution cipher is by no means secure today. First, due to its monoalphabetic nature, it still preserves the patterns of natural language. For example, in English, the most common letter is 'E'. An attacker could easily identify that the most frequent symbol after substitution is likely to be 'E' and then start guessing other letters, possibly using a dictionary lookup to assist. This idea is called *frequency analysis*. Moreover, brute-forcing a substitution cipher would take only a few seconds on a modern computer.

We can extend the idea of the Caesar cipher to build a polyalphabetic cipher.

6 Vigenère Cipher

The Vigenère Cipher is a *poly-alphabetic cipher* where each plain text character is encrypted with a different Caesar cipher. Which Caeser cipher to use is determined by the corresponding letter of the key. Here is an example. Consider our key is the word `CRYPTO` and our plaintext is `THISISAGREATDAY`. Encryption and decryption happen as below.

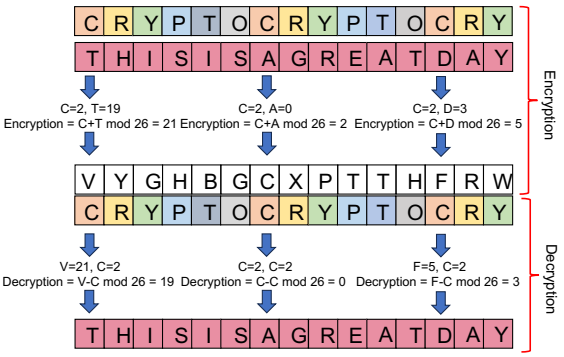


Figure 3: Vigenère Cipher - Encryption and Decryption Process

6.1 Breaking the Vigenère Cipher

If you know the key length - The key idea to note in Vigenère Cipher is that there will be positions where the plain text will be encrypted by the same character in the key. For example, in the figure, the plain text characters "T", "A", and "D" are encrypted by the key character "C". Now if you assume you have enough encrypted text, then you can use frequency analysis to guess what might be the cipher character for the letter "E".

Say you find that among all the cipher characters that were encrypted by the first character of the key is character "G". Now you can use this information to recover the first character of the key. `6-c mod 26 = 4`, where `c` is the first character in the key. This results in c = 2, meaning the first character of the key is "C". Next, you can do the same for the cipher characters that were encrypted from the second character of the key can recover it, and so on.

If you don't know the key length - You assume different key lengths and repeat the process until you get sense descriptions. However, with natural language, you can make your life easier by doing the Kasiski Examination.

6.2 Kasiski Examination

Kasiski allows us to find the key length in some circumstances. The idea is to find repeated patterns in cipher text with the hope that some will correspond to situations where the same plain text is encrypted by the same parts of the key. This is not, in particular, coincidental because the natural language has a number of repetitive patterns (e.g., the word 'the' will be repeated quite a lot in English language text) and given the short key sizes of the Vigenère Cipher, there will be occasions where the same plain text was encrypted by the same part of the key. See the below example. For simplicity, we ignore the space and period characters. We assume our key is "`KEY`" and our plain text is "`THE BELOW IS AN EXAMPLE FOR KASISKI TEST. THE TEXT IS ENCRYPTED UTILISING THE VIGENERE CIPHER.` ".

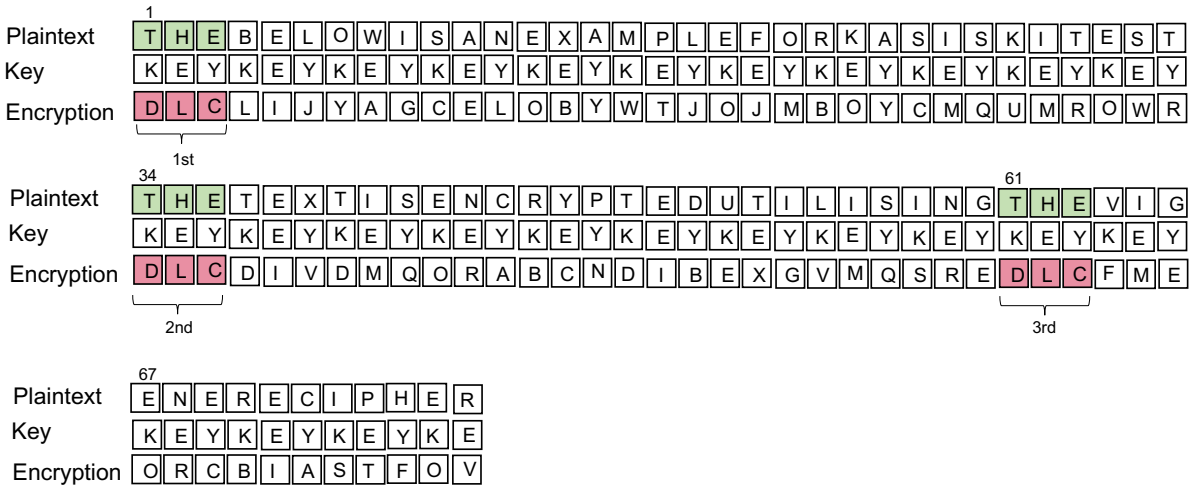


Figure 4: Kasiski Examination

The observation here is that the number of characters between the two repeated patterns is a multiple of the key length. Therefore, we can conclude the following.

- The distance between the 1st and 2nd repetition is 33. Therefore the key size must be one of 1, 3, 11 and 33, which are factors of 33.
- The distance between the 2nd and 3rd repetition is 27. Therefore the key size must be one of 1, 3, 9 and 27.
- The common factors are 1 and 3. We can ignore 1 as it is a very weak key that can be brute-forced.
- So the possible key size is 3.

Once you know the key size, you identify the cipher text characters that were encrypted by the same key character and conduct frequency analysis to infer each character of the key. *This step will still include some guesswork and trial and error similar to any other case of cryptanalysis.*

7 Ideas Behind Modern Ciphers

In Vigenère Cipher, we saw how polyalphabetic ciphers addressed an observed shortcoming of monoalphabetic ciphers: that frequency analyses (in many forms) can yield the key. Modern ciphers extend these ideas further.

Claude Shannon identified two design goals for modern ciphers.

Confusion and Diffusion

Confusion: This principle ensures that the relationship between the ciphertext and the key is as complex and unpredictable as possible, i.e. *each symbol in ciphertext must depend on several symbols in key* . The goal is to make it difficult for an attacker to determine the key even if they have access to the ciphertext. Substitution techniques, where one piece of data (e.g., a letter or bit) is replaced with another, are commonly used to achieve confusion. By substituting elements in a complex way, modern ciphers introduce confusion.

Diffusion: This principle spreads the influence of each plaintext bit across multiple ciphertext bits, making it difficult to discern any patterns, *i.e., every ciphertext symbol must depend on many plaintext symbols.* Diffusion is often achieved through transposition, where the positions of bits or characters are shifted around according to a specific pattern, thus spreading the plaintext information throughout the ciphertext.

8 Attacks on Cryptographic Systems

A cryptographic system is compromised if an adversary can obtain plaintexts for certain ciphertexts or the adversary can obtain the decryption key k_d when given k_e. **Cryptanalysis** is the science of decrypting without knowledge of the key. Depending on the attacker's capabilities.

- **Ciphertext-only** - The attacker may only see ciphertexts.

  Example - During World War II, many military communications were encrypted. If an attacker intercepted these encrypted messages (ciphertext) without knowing the corresponding plaintext or having any additional information, they would be in a ciphertext-only attack scenario. The challenge would be to decrypt the intercepted messages purely by analyzing the ciphertext.

- **Known plaintext** - The attacker may see some plaintexts and corresponding ciphertexts.

  Example: During World War II, the Allies, particularly the British cryptanalysts at Bletchley Park, were able to break the Enigma cipher used by Nazi Germany because they knew certain plaintext phrases (such as weather reports or common military terms) and the corresponding ciphertexts. This known plaintext information helped them in analyzing the patterns in the Enigma's encrypted messages, leading to the decryption of other messages.

- **Chosen plaintext** - The attacker may choose a plaintext and see the corresponding ciphertext (standard model for asymmetric cryptography!).

  Example - In the BEAST attack on SSL/TLS, the attackers are choosing plaintext blocks and studying the corresponding ciphertext blocks generated by the server's encryption algorithm to uncover vulnerabilities and deduce the secret key.

- **Chosen ciphertext** - The attacker may choose ciphertext and decrypted plaintext (so-called oracle attacks).

  Example - In Bleichenbacher's attack on PKCS1 v1.5, the attackers are exploiting a decryption oracle, submitting various ciphertexts, and studying the decrypted outputs to infer details about the decryption process and ultimately to uncover the secret key.

9 Stream Ciphers vs. Block Ciphers

Symmetric ciphers can be categorised into two based on their operation.

Stream Ciphers vs. Block Ciphers

Stream Ciphers: A stream cipher encrypts data one bit or byte at a time. It takes a single bit or byte of plaintext and combines it with a corresponding bit or byte of a keystream (a sequence of bits derived from a secret key) to produce a bit or byte of ciphertext. This process continues in a stream, hence the name. E.g., RC4 (now considered insecure) and Salsa20

Block Ciphers: A block cipher encrypts data in fixed-size blocks (typically 64 or 128 bits). It takes a block of plaintext, processes it with the secret key, and outputs a block of ciphertext of the same size. If the plaintext is smaller than the block size, padding is added to fill the block. E.g. DES (now considered insecure) and AES.

10 Stream Ciphers

In stream cipher design, pseudo-random functions are employed to generate a keystream from a seed, which is typically derived from a secret key. The fundamental idea is to generate a keystream $b = b_0, b_1, b_2, \ldots, b_n$ from the secret key k by applying a cryptographic function $f(k)$. This keystream is then combined with the plaintext $p = p_0, p_1, p_2, \ldots, p_n$ using a bitwise operation, most commonly the XOR operation (denoted as $\oplus$). The resulting ciphertext is obtained as $c = p \oplus b$, where each bit of the plaintext is XORed with the corresponding bit of the keystream.

One of the primary challenges in this process is ensuring that the keystream is as unpredictable as possible without knowledge of the secret key k. Specifically, it should be computationally infeasible for an attacker to distinguish between a truly random keystream and one generated deterministically from the key k. If an attacker could identify patterns or predict the keystream, the security of the encryption would be compromised. Therefore, the design of the pseudo-random function used to generate the keystream is crucial, as it must ensure that the keystream appears random and lacks any discernible patterns that could be exploited.

Figure 5 illustrates a schematic overview of a stream cipher system.

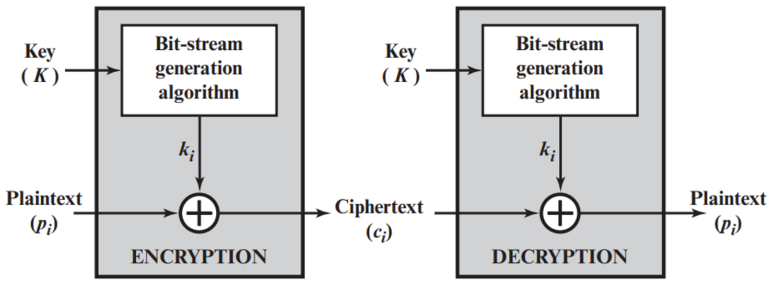


Figure 5: Basic operation of stream ciphers [1])

One-Time Pad is an example of a stream cipher. However, it is important to note that its key

stream is completely random, as opposed to the pseudo-random key streams used in other stream ciphers.

11 One-Time Pad

One-Time Pad (not to get confused with the One-Time Passwords) is a simple cipher that assumes a random key for each message you encrypt, and the key size is greater than the message itself. The encryption process is simply XORing the plain text with the key and the decryption process is vice versa. Here using XORing is not mandatory; you can also use modular arithmetic like `mod 26`.

Figure 6 shows two instances where the same plain text is encrypted at two different instances. Note the two assumptions at work; i) key length is as long as the plain text, ii) the key is random and never repeated/reused. As a result, the cipher texts in the two instances are very different - which leads to the fundamental property of One-Time Pad - the resulting cipher text doesn't show any relationship with the plain text.

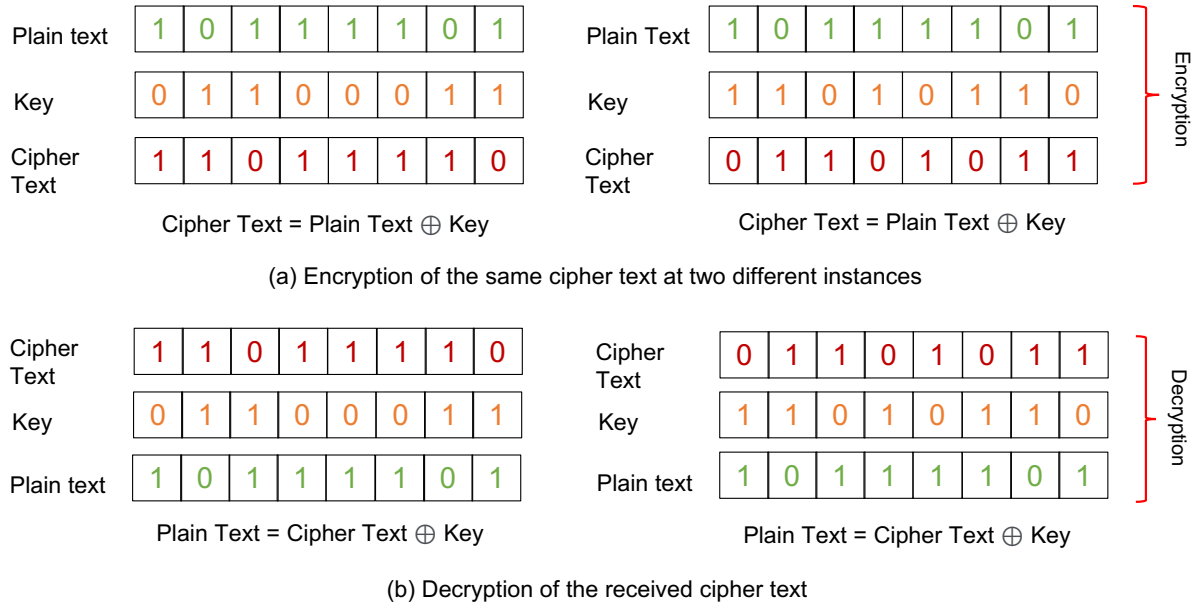


Figure 6: Operation of One-Time Pad

We call this property as *"perfect secrecy"*. That is, the cipher text does not convey any information about the content of the plain text. Essentially, this means no matter how much cipher text you have, it does not convey anything about what the plain text and the key were. This is why there is an emphasis in One Time Pad that the key can not be repeated. In fact, it can be proved that any *"perfect secrecy"* scheme must use at least as much key material as there is plain text to encrypt. In terms of probabilities, it means that the probability distribution of the possible plain texts is independent of the cipher text.

Nonetheless, the One-Time Pad is not that practical due to two reasons.

- The key distribution is an issue - How can Alice and Bob securely exchange the key? If

they already have the means to do that, the One-Time Pad is no longer required.

- Related to that is the generation of that much of a volume of truly random keys. One-Time Passwords will not be able to handle the requirements of common applications that require frequent communications.

11.1 Repeating the key in One-Time Pad

When the key is repeated (i.e., many time pads or looping one-time pads), the One-Time Pad can be broken similarly to how we broke the Vigenère Cipher earlier. However, there are other methods as well. An example of such a method is called "crib dragging".

- Let's consider a One-Time Pad that uses XOR.
- And consider two plain texts P_1 and P_2 encrypted by the same key K
- So we get $C_1 = P_1 \oplus K$ and $C_2 = P_2 \oplus K$
- Now assume the attacker gets hold of the two cipher texts, C_1 and C_2. The attacker can build a new string P where $P = C_1 \oplus C_2$, which can be simplified.

  $P = C_1 \oplus C_2$

  $P = (P_1 \oplus K) \oplus (P_2 \oplus K)$

  $P = P_1 \oplus P_2$ (because $K \oplus K$ is all zeros)

  This also means that $P_1 = P \oplus P_2$ and $P_2 = P \oplus P_1$
- Using this information, the attacker can try "crib dragging". The idea is to guess a word that can appear in one of the plan texts. For example, if it is English, and the text is sufficiently large, it is reasonable to assume the word "the" will be there somewhere in the plain text. So the idea is to try "the" in all possible positions in one of the plain texts, XOR them with P and see whether some readable can be obtained. The process continues like this.
- A worked example of the idea can be found here - http://travisdazell.blogspot.com/2012/11/many-time-pad-attack-crib-drag.html

12 Block Ciphers

As mentioned before, block cyphers encrypt data in fixed-size blocks (typically 64 or 128 bits) as shown in Figure 7. It takes a block of plaintext, processes it with the secret key, and outputs a block of ciphertext of the same size. If the plaintext is smaller than the block size, padding is added to fill the block.

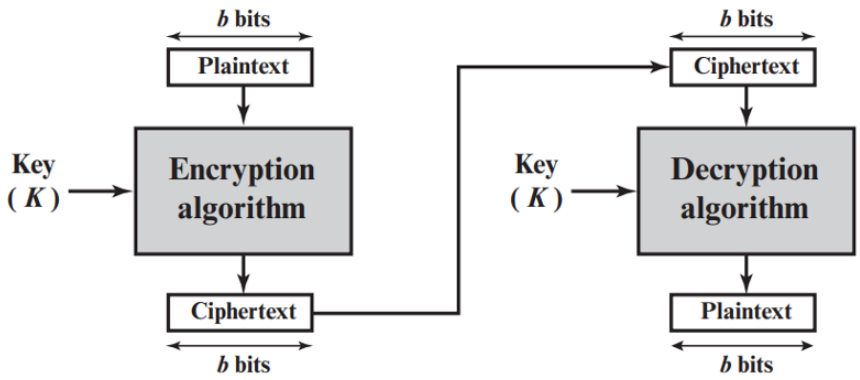


Figure 7: Basic operation of block ciphers [1]

In block ciphers, substitution is implemented through **Substitution boxes (S-boxes)**, while permutation (transposition) is carried out using **Permutation boxes (P-boxes)**. These components are executed across multiple rounds of the algorithm, which helps to enhance the diffusion and confusion properties. The rounds are structured in specific arrangements, often referred to as **'networks'**, to ensure the effective mixing of the plaintext into ciphertext.

13 Feistel Cipher

The Feistel cipher is a common structure used in block ciphers like DES, Blowfish, Twofish, and others. It divides the data block into two halves and processes them in multiple rounds. Each round follows the same basic structure as illustrated in Figure 8:

1. **Splitting the Data:**

   $$\text{Given a block of data } X, \text{ split it into two halves: } L_n \text{ and } R_n$$

   $$X = L_n \,||\, R_n$$

2. **Round Operations (for each round n) :** The right-hand side half goes through a keyed function F, and the output is XORed with the left-hand side, i.e. R_{n+1}. The direct output of the function F becomes L_{n+1}. This process happens multiple times. We will learn what exactly happens inside F under DES - basically, it consists of an arrangement with S and P boxes.

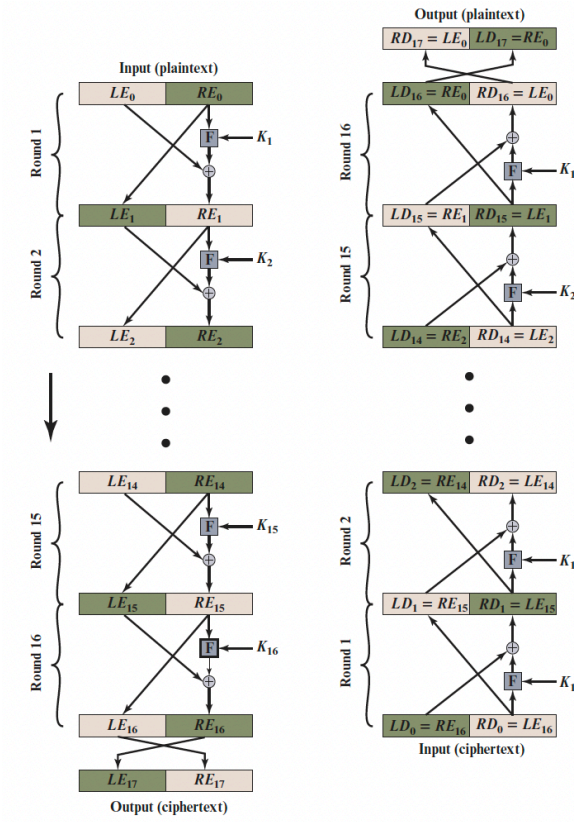


Figure 8: Example Feistel Network [1]

Beyond Feistel networks, there are other ways of arranging S-boxs and P-boxes, such as the Substitution Permutation Network (SPN) used in AES.

14 Data Encryption Standards (DES)

DES encryption algorithm, originally published in 1977 and standardized in 1979, uses a key that, in each round, is transformed into a subkey derived from the actual key (round key). The algorithm includes a total of eight S-boxes, with the outputs from these S-boxes being sent through one P-box to ensure good diffusion across the S-boxes in the subsequent round. It is important to note that the 16 rounds utilized in this process are necessary to achieve the desired level of security. The algorithm employs a Feistel structure between rounds. As we discuss later, DES no longer considered secure.

14.1 S-Boxes

S-boxes are basically look-up tables. They map a block of bits and output a block of bits. The mapping functions are carefully chosen. Usually, changing one bit in the input of the S-box will map to an output block where, on average, half of all bits are flipped (diffusion). There are two options to make the decryption possible.

Invertible S-Boxes: If an S-box is invertible, it means that for every possible output of the S-box, there is a unique input that could have produced it. This property allows the decryption process to reverse the transformation applied during encryption.

Invertible Arrangement: Even if the individual S-boxes aren't invertible, the way they are combined and used in the overall encryption algorithm (such as in combination with other transformations) can still allow the entire encryption process to be reversed. This means the cipher as a whole is designed in such a way that decryption is possible, even if some components (like S-boxes) are not individually invertible. This is what DES uses.

14.1.1 Example S-box - S[0] from DES

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Figure 9: S[0] in DES

Use like this:

$$\text{input string} = (b_0, b_1, b_2, b_3, b_4, b_5)$$

b_0 and b_5 give the row; (b_1, b_2, b_3, b_4) give the column.

Interpret the number in that cell as binary.

Example:

$$\text{Input} = (1, 1, 0, 1, 1, 1) \rightarrow (110111)_2$$

$$\text{Row: } (11)_2 = (3)_{10}; \quad \text{Column: } (1011)_2 = (11)_{10}$$

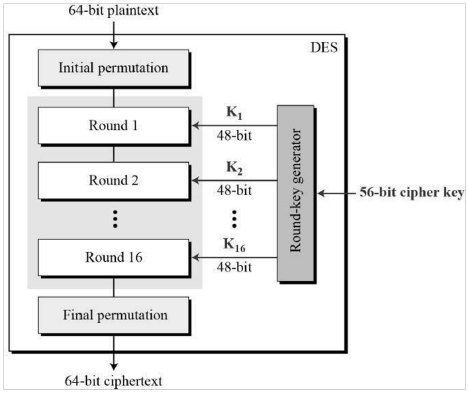
$$\text{Output: } (14)_{10} = (1110)_2 \rightarrow \mathbf{1110}$$

DES has eight S-boxes like this one.

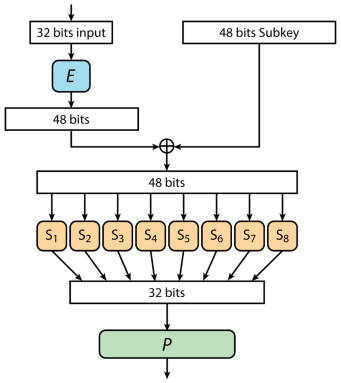
14.2 P-Boxes

Transposition in encryption is handled by P-boxes, which are simple index permutations or re-orderings of bits. P-boxes are often used in conjunction with S-boxes to create a chain where the output from S-boxes is passed through a P-box before being fed into subsequent S-boxes. The P-box redistributes the bits from one S-box across as many of the following S-boxes as possible, ensuring effective diffusion throughout the encryption process.

14.3 Operation of DES



https://www.tutorialspoint.com/cryptography/data_encryption_standard.htm



Source: Wikipedia

Figure 10: Building blocks of DES

- DES accepts data blocks of 64-bits, and its key size is 56-bits. The round-key generator generates 16 48-bit round keys to use in the 16 rounds.
- Inside each round is the Feistel arrangement where the function F looks like the figure on the right.
- Inside one round, the 64-bit input is split into two halves, left hand and right hand, 32 bits each. The right-hand side goes through the function F. The First 32 bits are expanded to 48 bits using the expansion box so that it is the same size as the round key. Next, the expanded output is XORed with the round key and resulting 48 bits intermediate output is sent to 8 S-Boxes. Note from the previous example that each S-box accepts 6-bit input, so the 48-bit output is split evenly among the 8 s-boxes. Recall also that the S-box output is 4 bits each, so the the 8 S-boxes result in 32 bits, making that output suitable for the next round. This process repeats 16 times.

14.4 Security of DES

As it was mentioned earlier, DES is no longer considered secure. The major limitation of DES comes from its shorter key length. In DES there are 2^{56} possible keys. That means the key space is $\sim 7.2 \times 10^{16}$. At a rate of 10^9 keys per second (a multicore computer) or 10^{13} keys per second (a supercomputer), DES can be broken in ≈ 1 year or ≈ 1 hour, respectively. As a result, we use AES that has larger key sizes of 128, 192, and 256 these days. The table in Figure 11 below provides a summary of the times taken to break each crypto scheme.

Key Size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 Decryptions/s	Time Required at 10^{13} Decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years
26 characters (permutation)	Monoalphabetic	$2! = 4 \times 10^{26}$	2×10^{26} ns = 6.3×10^{9} years	6.3×10^{6} years

Figure 11: Strength of DES and AES [1]

Example Calculation

DES key space $= 2^{56} = 7.2 \times 10^{16}$

If you are brute forcing the key, trying all possible combinations of the keys is the worst case. Usually, you will find the key much earlier than that. **So we assume that *on average* once we tried half the amount of keys, we will get a hit.**

This means we have to only try 2^{55} keys.

At 10^9 keys per second we need $2^{55}/10^9$ seconds.

This is equal to $\approx 36,028,797$ seconds, and that is ≈ 1.2 years.

At 10^{13} keys per second we need $2^{55}/10^{13}$ seconds. Which is $\approx$ 3,600 seconds, which is only an hour.

The last row of the table has a typo. I hope you can figure it out !!

14.5 3DES

As an intermediate solution to the risks of DES, 3DES was introduced. It is basically applying DES three times, with a slight difference as illustrated in Figure 12.

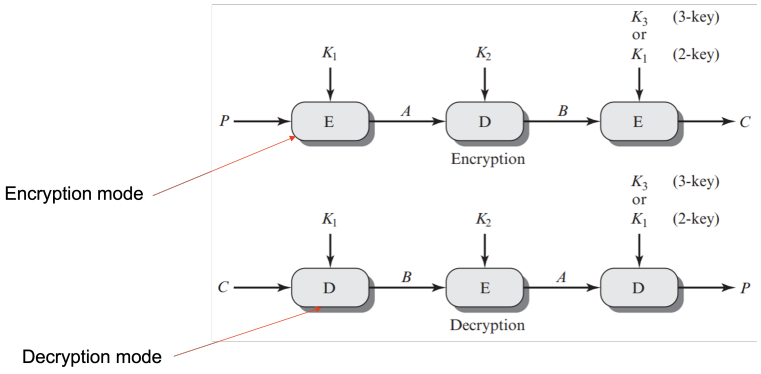


Figure 12: Encryption and Decryption in 3DES [1]

3DES uses two keys (K_1, K_2) or three keys (K_1, K_2, K_3), giving effective key sizes of 112 bits (56 x 2) or 168 bits (56 x 3). The two options are there to balance between performance and security.

Let's follow the operation of the three key options. First, we take the plain text, p and encrypt it using K_1, i.e., we get $E_{K_1}(p)$. Next, we use the decryption function of DES with K_2 on this output. i.e., we get $D_{K_2}(E_{K_1}(p))$. Note that we use the decryption function but with a different key, so we will not get the original plaintext back. Next, we do another round of encryption using K_3 to obtain $C = E_{K_3}(D_{K_2}(E_{K_1}(p)))$.

Why do we use the Encryption, Decryption, and Encryption sequence? Why not simply Encryption, Encryption, and Encryption? There are several reasons. One is backward compatibility. When using the same key in three steps, 3DES will again become DES.

15 Block Cipher Design Principles

Number of Rounds: In block cipher design, the number of rounds plays a crucial role in determining the cipher's resistance to cryptanalysis. A higher number of rounds increases the difficulty of breaking the cipher, even if the underlying function F is not particularly strong. This added complexity enhances the element of confusion, especially in Feistel ciphers, making it harder for attackers to deduce the relationship between the plaintext and ciphertext.

Design of Function F: The function F within a block cipher should be nonlinear and exhibit strong avalanche properties, such as those defined by the strict avalanche criterion (SAC). Also, it is recommended for F to satisfy Bit Independence Criterion (BIC).

Desired properties of F

Strict Avalanche Criterion (SAC): Whenever a single input bit is complemented, each of the output bits changes with a 50% probability.

Bit Independence Criterion (BIC): Output bits j and k should change independently when any single input bit i is inverted, for all i, j and k.

Key Schedule Algorithm: The Key Schedule Algorithm generates one subkey for each round from the main key. In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key. No general principles for this have yet been defined; SAC and BIC are desired here.

16 Advanced Encryption Standards (AES)

Advanced Encryption Standards (AES) is the current de-facto block cipher on the Internet. It is the successor of DES. It operates in three key sizes: 128, 192, and 256. Its block size is 128 bits. AES also uses S-boxes and P-boxes. However, in contrast to DES, AES doesn't use the Feistel Network. Rather, it is based on a substitution-permutation network (SPN).

16.1 Finite Field Arithmetic

AES uses arithmetic in the finite field $GF(2^8)$ with the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$. That means addition and multiplication are done differently. Please refer to the math background handouts to understand this further.

16.2 AES Encryption and Decryption

The high-level flow of AES is shown in Figure 13. Overall, it takes 128-bit blocks and an M-bit key (128, 192, or 256) and has 10, 12, or 14 rounds, depending on the key size. For clarity for the rest of the discussion, let's focus on 128-bit key size.

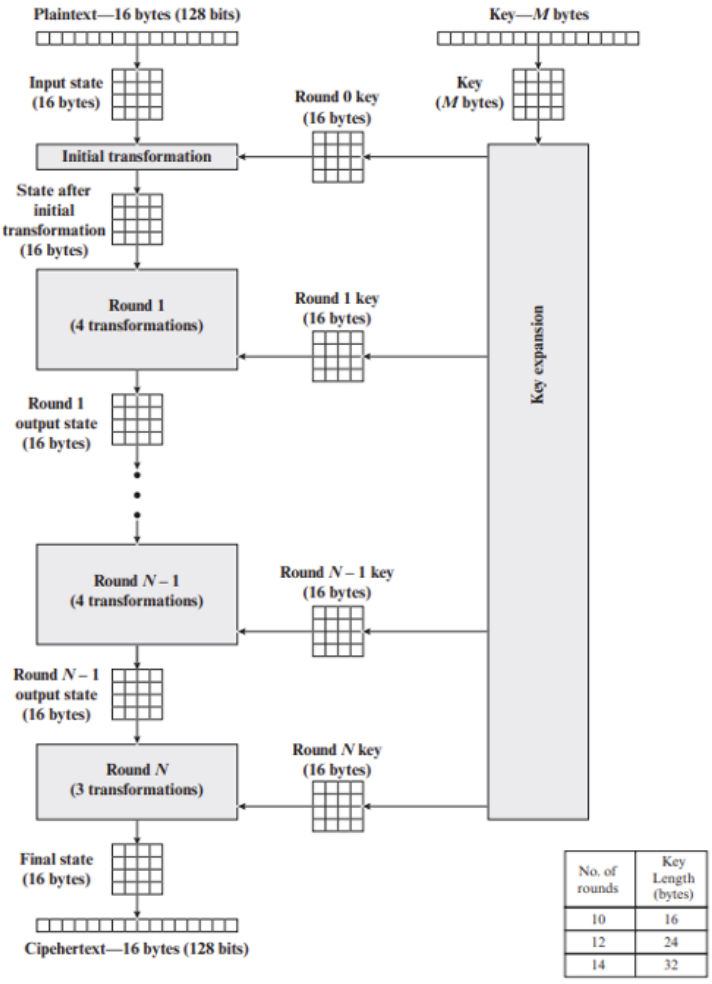


Figure 13: Encryption in AES [1]

Key Expansion Algorithm The key expansion process generates 11 round keys, each 128 bits in size, resulting in a total of 176 bytes (11×16 bytes).

Inside each round four steps are happening.

- **Substitute bytes:** Uses an S-box to perform a byte-by-byte substitution of the block.
- **ShiftRows:** A simple permutation.
- **MixColumns:** A substitution that makes use of arithmetic over $GF(2^8)$.
- **AddRoundKey:** A simple bitwise XOR of the current block with a portion of the expanded key.

17 Cipher Block Modes

Ciphers are designed to handle messages of arbitrary length by splitting them into blocks and processing these blocks according to a cipher block mode. However, if these modes of operation are not carefully designed and used, they can introduce new security vulnerabilities. Traditional block modes, such as Electronic Codebook (ECB), Cipher Block Chaining (CBC), and Counter (CTR), focus solely on data encryption. In contrast, modern modes offer authenticated encryption (AE, AEAD), which combines encryption with integrity protection. Examples of these advanced modes include Galois Counter Mode (GCM) and Counter-with-CBC-MAC (CCM).

17.1 Electronic Code Book Mode – ECB

In ECB, each plaintext block is encrypted independently, as shown in Figure 14. That is $C_i = E_i(P_i)$

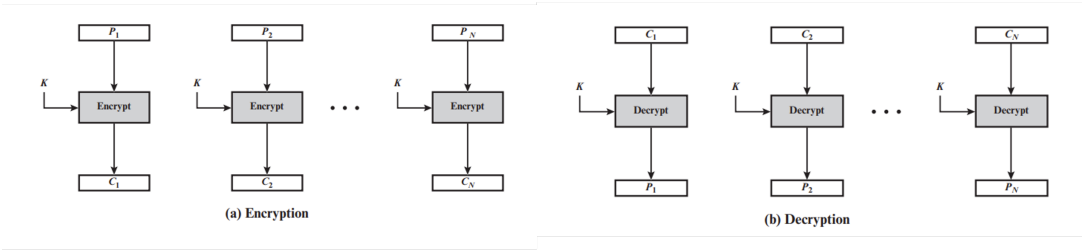


Figure 14: ECB [1]

ECB has a major limitation, and as a result, it is not used at all these days. Most of the time, plaintext will have patterns that repeat (e.g. white background of an image). In ECB, the same plain text will generate the same cipher text. As a result, some patterns in the plaintext will be visible in ciphertext as well, as shown in the example in Figure 15.

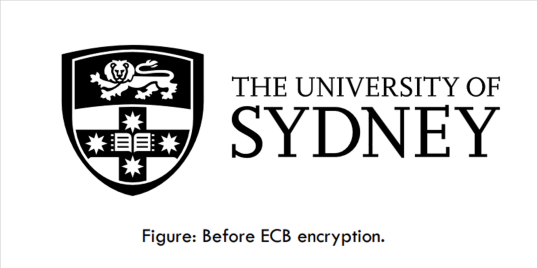


Figure: Before ECB encryption.

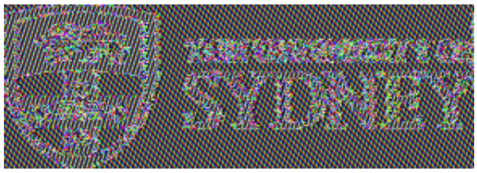


Figure: After ECB encryption.

Figure 15: ECB Encryption Example

17.2 Cipher Block Chaining – CBC

In CBC, each plaintext block is XORed with the previous ciphertext block before being encrypted. The first block is XORed with an initialization vector (IV) to ensure that even identical plaintext blocks result in different ciphertexts.

This approach avoids the problem in Electronic Codebook (ECB), where identical plaintext blocks are encrypted into identical ciphertext blocks, making patterns in the plaintext easily detectable. By chaining the blocks together, CBC ensures that the encryption of each block depends on the previous one, effectively masking patterns and providing better security. The operation of CBC is shown in Figure 16. Now, the problem of the same plaintext being mapped to the same cypher text is no longer there, as shown in Figure 17.

CBC encrypt: $c_i = Enc_k(c_{i-1} \oplus m_i)$ CBC decrypt: $m_i = Dec_k(c_i) \oplus c_{i-1}$

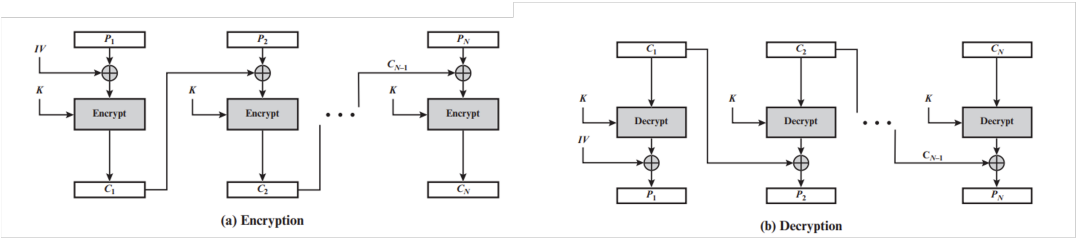


Figure 16: CBC [1]

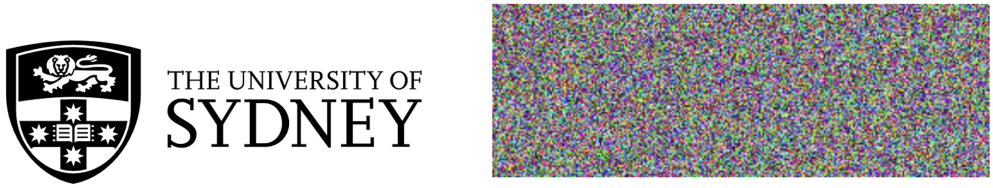


Figure: Before ECB encryption.

Figure: After CBC encryption.

Figure 17: CBC Encryption Example

Initialization Vector (IV): The IV is a random or pseudo-random value that is used as the first input in the encryption process. Its primary purpose is to introduce randomness into the encryption, ensuring that even if the same plaintext is encrypted multiple times with the same

key, the resulting ciphertext will be different each time.

The IV does not need to be kept secret. It is often transmitted along with the ciphertext to allow the decryption process to correctly reconstruct the original plaintext. Since the IV is not a secret, its role is not to provide security through secrecy but through the introduction of unpredictability.

It is critical that the IV is never repeated when using the same encryption key. If the same IV is used with the same key for encrypting identical plaintext blocks, the resulting ciphertext blocks will also be identical. This can reveal patterns in the plaintext and compromise security, similar to the vulnerabilities seen in the ECB mode.

Wired Equivalent Privacy (WEP)

WEP, an early security protocol for wireless networks, uses the RC4 stream cipher, which relies on a combination of a secret key and an Initialization Vector (IV) to encrypt data.

In WEP, the IV is only 24 bits long, which is relatively small. This small size means that after a certain amount of data is transmitted, the same IV is likely to be reused. In busy networks, IVs can start repeating very quickly.

When the same IV is reused with the same secret key, RC4 generates the same key stream. If an attacker can capture multiple packets with the same IV, they can observe how the key stream interacts with different plaintexts.

Plaintext Recovery: By analyzing these repeated key streams, especially if parts of the plaintext are known or predictable (like certain headers or protocol-specific information), an attacker can start to recover the plaintexts of other packets. Once enough packets are captured, the attacker can deduce the key stream and, eventually, the secret key itself.

17.3 Counter Mode - CTR

Though CBC overcomes the limitations of ECB, it is inefficient. The encryption process is sequential, and each step is dependent on the output of the previous step. As a result, CBC can't be parallelised. Counter mode provides an easy solution for this.

In CTR mode, a counter is incremented for each block, and this counter, combined with a nonce, is encrypted to produce a keystream that is then XORed with the plaintext, ensuring both efficiency and security. The operation of CTR mode is shown in Figure 18.

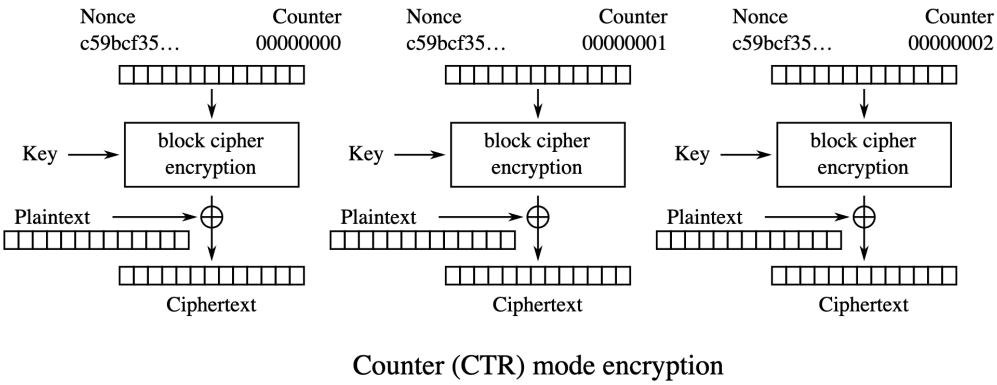


Figure 18: CTR

The nonce in CTR mode serves a similar role to the Initialization Vector (IV) in CBC. Like an IV, the nonce does not need to be kept secret, but it is crucial that it is never reused with the same key. Reusing a nonce with the same key would result in the same keystream being generated, which could allow an attacker to recover the plaintext by analyzing the repeated patterns in the ciphertext.

18 Pseudo Random Number Generators

Random numbers are essential in cryptography because they introduce unpredictability and ensure the security of various cryptographic operations. In encryption, random numbers are used to generate keys, Initialization Vectors (IVs), and nonces, all of which prevent patterns from emerging in encrypted data. This unpredictability is critical because if an attacker could predict or reproduce these random values, they could potentially decrypt messages or forge digital signatures. Random numbers also play a vital role in secure communication protocols, where they are used in generating session keys and ensuring that each session remains unique and secure. Without strong randomness, cryptographic systems would be vulnerable to attacks that exploit predictable patterns, ultimately compromising the confidentiality and integrity of sensitive data.

A Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) is a type of Pseudorandom Number Generator (PRNG) that produces a deterministic sequence of numbers based on an initial seed value. The output sequence of a PRNG is entirely dependent on this seed, making the seed crucial to the generator's operation. For a PRNG to be considered cryptographically secure, it must be computationally infeasible for an attacker to brute-force the seed value or predict the output sequence. Additionally, the output must be indistinguishable from true randomness, ensuring that the only viable option for an attacker is to somehow know or discover the seed.

19 Practice Quiz

Question 1: Consider the following in a simple cryptosystem where c is the cipher text, p is the plain text and k is the key, and c = p XOR k.

Assume:

p (plaintext) = 10111011

k (key) = 00111101

Which one(s) of the following is true about basic cryptographic operations? Select all that apply.

a) c = 10000110
b) c XOR p = k
c) p = c XOR k
d) The attacker only needs one plain text and cipher text pair to derive the key.

***Explanation**: We know that c = p XOR k. So c = 10111011 XOR 00111101, which is 10000110. Therefore c =10000110. => a) is correct.*

b) is correct. This is a property of XOR. You can verify this: c = 10000110 and p=10111011. c XOR p = 00111101, which is equal to k.

c) is also correct. Again this is a property of XOR. You can verify this.

Above b) shows that if the attacker can get hold of a cipher text and its corresponding plaintext, they can derive the key.

Question 2: As mentioned in the lecture, use the below link to read about using the Kasiski analysis for breaking the Vigenère Cipher (A polyalphabetic cipher).

`https://crypto.interactive-maths.com/kasiski-analysis-breaking-the-code.html`

Now answer the following.

'BWGWBHQSJBBKNF' is a ciphertext generated from the Vigenère Cipher using a two-letter key. You also, know that the second letter of the plaintext is "I".

What is the correct plain text?

a) AIMISTHEANSWER
b) SIXISTHEANSWER
c) AIDISTHEANSWER
d) HITISTHEANSWER

***Explanation**: The key idea is that "same key" and "same plain text" will result in the same cipher text. Therefore we can build something like this, where the two-character key is repeated (See Figure 19).*

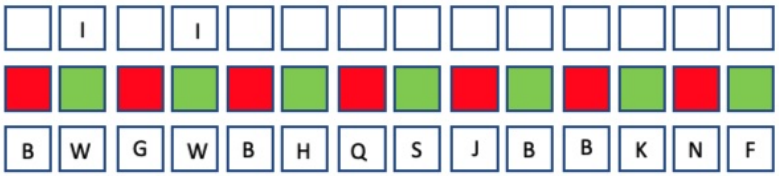


Figure 19: Key alignment in Vigenère Cipher

Given the second letter of the plaintext is I, we can conclude that the fourth letter is also I - because the cipher text is W and it is encrypted by the same key.

We can also conclude that the first, fifth, and eleventh characters in the plain text have to be the same - because they have encrypted with the same character in the key and have produced the same cipher text B (See Figure 20).

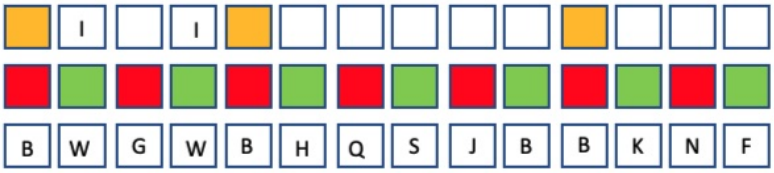


Figure 20: Same cipher text

Now we can eliminate answers

AIMISTHEANSWER
SIXISTHEANSWER
AIDISTHEANSWER
HITISTHEANSWER

Only "SIXISTHEANSWER" has the first, fifth, and eleventh characters the same.

Question 3: Which of the following statement is NOT TRUE about one-time pad (OTP)? Select all that apply.

a) Once generated OTPs can be reused.
b) OTP length has to be at least as long as the plaintext.
c) Practical use of OTP is difficult.
d) OTP gives perfect security.

***Explanation**: The key idea of OPT is that once generated, it can be used only once. Why? "if you re-use the same key, and someone has access to one message you encrypted in both plaintext and encrypted form, they can use that to find your key (i.e., known plaintext attack)*

Therefore "Once generated OTPs can be reused" is NOT true => The correct answer.

To avoid repetition, the OTP length must be greater than the plaintext. Therefore "OTP length has to be at least as long as the plaintext" is TRUE.

OTP have issues distributing the key. Therefore "Practical use of OTP is difficult" is TRUE.

OTP gives the maximum randomness in the ciphertext. The key is not repeated and the ciphertext does not provide any additional information about the plaintext.

Question 4: What is TRUE about the IV (Initialization Vector) used in symmetric cryptography?

a) IV must be kept as a secret.
b) IVs can be repeated.
c) IV ensures that the same pain text block does not generate the same cipher text.
d) IV are used in ECB (Electronic Code Block) block mode.

Explanation*: In general, the requirement IVs is that they must not be repeated, but they do not have to be a secret. Usually, they are generated at the transmitter communication and sent in plaintext to the receiver. If IVs are repeated, the same issues as OTP repetition can happen. Therefore 1) and 2) are not true. 3) explain the exact idea of IV - therefore, it is TRUE. We use ECB as the motivation to use IV. Therefore 4) is not true.*

Question 5: Which of the following are TRUE about Congrunet modulo n? Select all that apply.

a) $26 \equiv 11 \mod (5)$
b) $74 \equiv 30 \mod (11)$
c) If $a \equiv b \mod (n)$ *and* $b \equiv c \mod (n)$ then $a \equiv c \mod (n)$
d) If $a \equiv b \mod (n)$ and $b \equiv c \mod (n)$ then $a = c$

Explanation*:*

a) $26 \mod (5) = 1$ and $11 \mod (5) = 1 \implies 26 \equiv 11 \mod (5) \implies$ TRUE
b) $74 \mod (11) = 8$ and $30 \mod (11) = 8 \implies 74 \equiv 30 \mod (11) \implies$ TRUE

c)

$a \equiv b \mod (n) \implies$

$a = k_1 n + p_1$ (where $p_1 < n$)
$b = k_2 n + p_1$ (where $p_1 < n$)

Similarly,

$b = k_3 n + p_2$ (where $p_2 < n$)
$c = k_4 n + p_2$ (where $p_2 < n$)

$k_2 n + p_1 = k_3 n + p_2$
$(k2 - k3)n + p_1 = p_2$

$c = k_4 n + (k_2 - k_3)n + p_1$
$c = (k_4 + k_2 - k_3)n + p1 \implies a \equiv c \mod (n) \implies$ TRUE

d) This is not true. Giving a counter-example will be enough.

Let a = 26, b = 11, c = 16, and n = 5.

$26 \equiv 11 \mod (5)$
$11 \equiv 16 \mod (5)$,
And a is not equal to c. $\implies$ NOT TRUE

Question 6: Which of the following is NOT a finite filed under the given modulus?

a) 0,1 modulus = 2
b) 0,1,2,3,4,5 modulus = 6
c) 0,1,2,3,4,5,6 modulus = 7
d) 0,1,2,3,4,.....,30 modulus = 31

Explanation: *The key here is that we know what are prime finite fields. That is 0,1,2,3,....p-1 is a field when p is a prime. That means a), c), and d) are all finite fields.*

b) is not a field. The first observation is that the modulus is not prime. Here it can be proved that some elements in 0,1,2,3,4,5 don't have a multiplicative inverse under modulo 6.

For example, let's take 2.

2.0 mod 6 = 0
2.1 mod 6 = 2
2.2 mod 6 = 4
2.3 mod 6 = 0
2.4 mod 6 = 2
2.5 mod 6 = 4

That is, there is no such element in 0,1,2,3,4,5 such that 2.x mod 6 =1 (lacking the inverse multiplicative property). Therefore 0,1,2,3,4,5 is not a finite field.

Question 7: DES (Data Encryption Standards) is still considered secure. TRUE or FALSE.?

a) TRUE
b) FALSE

Explanation: *FALSE - DES is insecure because of the shorter key length.*

Question 8: AES uses a ..... bit block size and a key size of ..... bits.

a) 128; 128 or 256
b) 64; 128 or 192
c) 256; 128, 192, or 256

d) 128; 128, 192, or 256

Explanation: *AES is a block cipher that supports 128-bit blocks and three key sizes: 128, 192, or 256.*

Question 9: Like DES, AES also uses Feistel Structure. TRUE or FALSE.

a) TRUE
b) FALSE

Explanation: *AES does not use a Feistel structure. Instead, each full round consists of four separate functions:*

- *Byte substitution*
- *Permutation*
- *Arithmetic operations over a finite field*
- *XOR with a key*

Question 10: Compute the output of the following S-Box (as shown in the lecture). Your input is 010000.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Figure 21: Given S-Box

a) 0011
b) 0001
c) 0000
d) 1010

Explaination: *The input string is 010000.*

b_0 *and* b_5 *gives the row - that means* 00_2 -> 0_{10} *is the row.*

$b_1b_2b_3b_4$ *gives the column - that means* 1000_2 -> 8_{10} *is the column.*

At the 0th row and 8th column, the table reads 3. So the output is 0011.

References

[1] William Stallings. *Cryptography and Network Security: Principles and Practice, Global Edition.* Pearson, Upper Saddle River, NJ, 8th edition, 2022.