Opinnäytetyön nimi

Mahdollinen alaotsikko

LAB-ammattikorkeakoulu

Insinööri (AMK), Tieto- ja viestintätekniikka

2023

Roope Salminen

|  |  |  |
| --- | --- | --- |
|  | | Tiivistelmä |
| Tekijä(t)  Sukunimi, Etunimi | Julkaisun laji  Opinnäytetyö, AMK | Valmistumisaika  20xx |
| Sivumäärä  xx |  |
| Työn nimi  **Opinnäytetyön otsikko**  Mahdollinen alaotsikko | | |
| Tutkinto  esim. Insinööri (AMK) | | |
| Toimeksiantajan nimi, titteli ja organisaatio  esim. Tarmo Toimeksiantaja, suunnitteluinsinööri DI, Suunnittelutoimisto Oy AB | | |
| Tiivistelmä  Tänne kirjoitetaan tiivistelmä, jossa kerrotaan …. | | |
| Asiasanat  asiasana, asiasana, asiasana | | |

|  |  |  |
| --- | --- | --- |
|  | | Abstract |
| Author(s)  Lastname, Firstname | Type of Publication  Thesis, UAS | Published  20xx |
| Number of Pages  xx |  |
| Title of Publication  **Title of the Bachelor’s Thesis**  Possible subtitle(s) | | |
| Name of Degree  e.g. Engineer (UAS) | | |
| Name, title and organization of the client  e.g. Clint Client, Design Engineer M.Sc. (Eng.), Design Office Ltd. | | |
| Abstract  Type the abstract here…. | | |
| Keywords  keyword 1, keyword 2, keyword 3 | | |

Table of Contents

[1 Introduction 1](#_Toc147995061)

[2 Block ciphers 2](#_Toc147995062)

[2.1 Common general principles 2](#_Toc147995063)

[2.2 ECB – Electronic Code Book 3](#_Toc147995064)

[2.3 CBC – Cipher Block Chaining 4](#_Toc147995065)

[2.4 CTR – Counter Mode 5](#_Toc147995066)

[2.5 GCM – Galois-Counter Mode 7](#_Toc147995067)

[3 Comparison of modes 8](#_Toc147995068)

[4 Selected flaws and vulnerabilities 9](#_Toc147995069)

[5 Conclusions 10](#_Toc147995070)

[References 11](#_Toc147995071)

Liitteet

Liite 1. Liitteen otsikko

Liite 2. Liitteen otsikko

1. Introduction

This paper will compare three different modes used in block ciphers and will demonstrate some of the most obvious weaknesses in two of them.

Here are some of the terms and abbreviations used throughout the paper.

Bit: the basis of digital data, the smallest unit of data, either 0 or 1

Byte: a collection of eight bits

XOR: exclusive or, a logical commutative operation that returns true if an odd number of its arguments are true. Here we only use two arguments, so it returns true if and only if one of them is true. The symbol used to denote the XOR operation will be .

1. Block ciphers
   1. Common general principles

A block cipher is an invertible deterministic secret-key algorithm that takes a fixed length input block and outputs a block that is the same length.[1] Deterministic here meaning that the same input together with the same key will always result in the same output and invertible meaning that for the encryption function there exists a decrypting function such that . We can write this in the form where is a plaintext block and is the secret key.

Like the name suggests, encryption and decryption are carried out by dissecting the data or message to be encrypted into smaller units called blocks which are handled by the algorithm. This is where different modes come into play as they determine how the block cipher is applied to messages whose length exceeds the block size in use. Block sizes can vary but typical sizes are 64, 128 and 256 bits.[2] Because UTF-8 encoding uses one to four bytes to represent characters, the block size dictates how many characters fit into a single block. For example, the character ‘n’ is 110 in decimal, 6E in hexadecimal and 01101110 in binary, meaning it can be represented by a single byte. If the block size was 64 bits, then eight ‘n’ characters would equal to a single block. This is also the case for most of the common letters in the English language.

In general, a 64-bit block size is nowadays considered to be too small. As a result of the birthday paradox, when the same key is used for large amounts of data, the probability of a collision happening grows too high. A collision meaning a case where two identical ciphertext block are produced. Identical outputs will in CBC mode, for example, imply that the inputs are identical, which will lead to the discovery of the result of the XOR operation of two plaintext blocks.[3] In ECB mode identical cipher blocks will always imply identical plaintext blocks. Both will make code book attacks a viable option for an attacker. As a result, most security protocols today implement larger block sizes. AES which is widely used in TLS in securing HTTPS traffic, for example, uses a 128-bit block size. One could summarize that too small of a block size results in a weak cipher and too large of a block size makes encryption and decryption cumbersome, the calculations require more time and resources. *(Lähde?)*

Key length for the algorithm is a simpler issue. As block ciphers are symmetric key encryptions, the key must be kept secret. Therefore, the most important feature of a key is that it is in practice too long to be brute-forced i.e., it will take an unreasonable amount of time and computing power to test all possible keys. AES supports key sizes of 128, 192 and 256 bits. For a 128-bit key there are possible candidates, which means it would take a huge amount of time and energy to try all the possible combinations of ones and zeroes. *(Etsi validi lähde?)* This is of course assuming that the whole possible key space is utilized by the key generating process as was not the case in the Ubuntu/Debian OpenSSL incident. *(Etsi validi lähde?)*

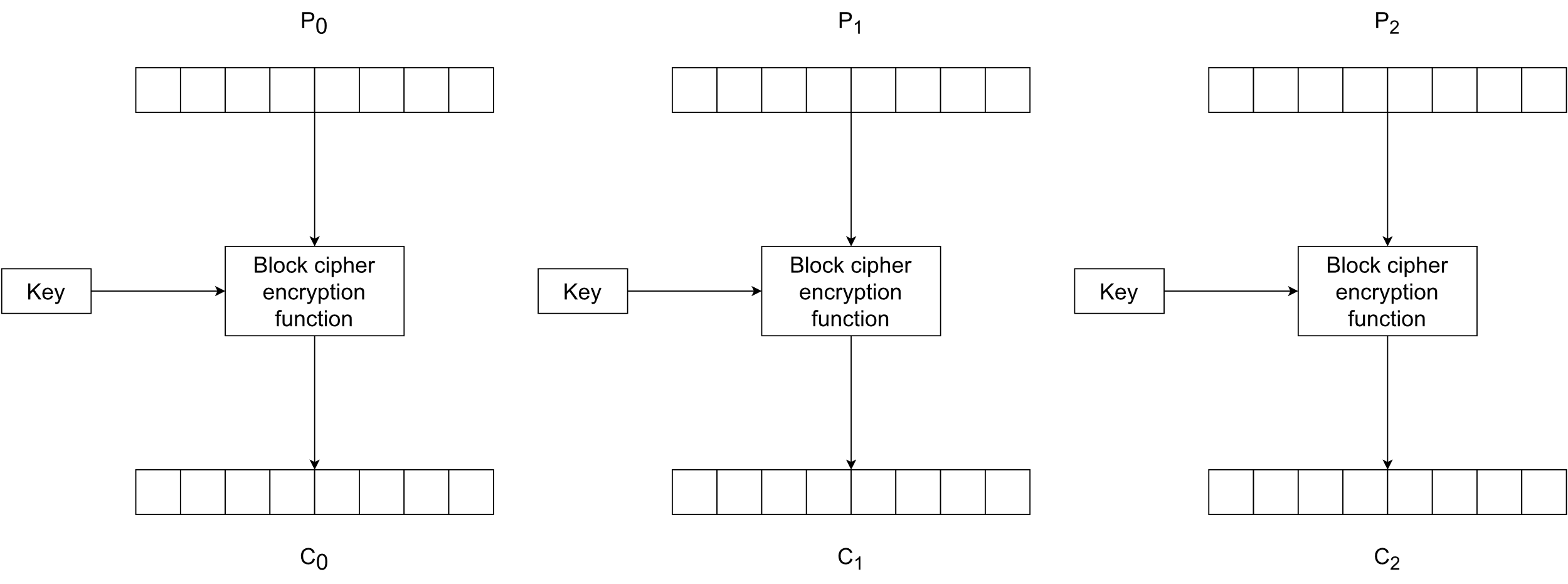
Since block cipher algorithms operate on whole blocks of data some modes require the last block of data to be padded with extra bytes to reach the block size currently in use. After decrypting the message, the extra bytes are discarded as they contain no information relevant to the original message.

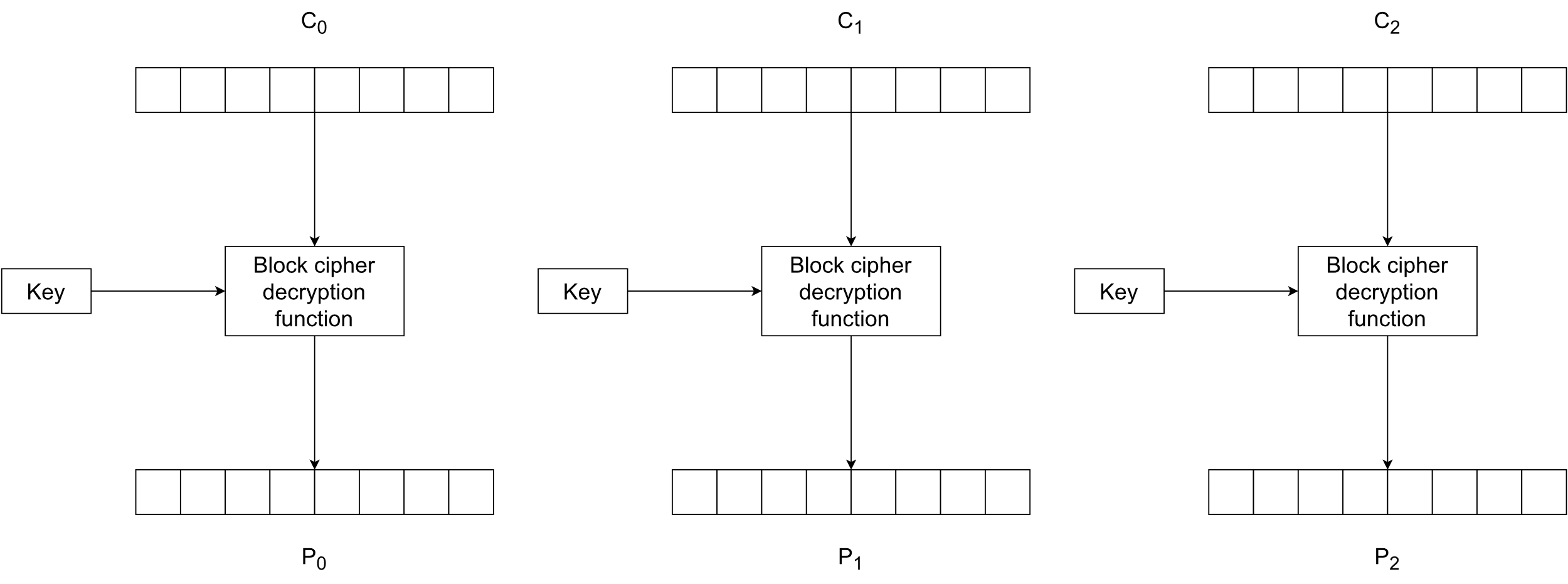
* 1. ECB – Electronic Code Book

This is the simplest of the modes and its name hints to days before digitalization where actual physical code books were used in encrypting messages. ECB is entirely deterministic and lacks diffusion, which means that all blocks are processed individually using the same key. Messages encrypted with the same key produce identical ciphertext blocks for identical plaintext blocks. This results in easy to recognize patterns in the ciphertext as messages often adhere to a standard format. Traditional letters and emails often begin with greetings and end in well-wishes. HTTP requests also follow a standard format so it can be relatively simple to start piecing together what are the plaintext counterparts of certain ciphertext blocks.

Images 1 and 2 demonstrate the principle of encrypting and decrypting a message. The message here is too big to fit into a single block so it is divided into three eight-byte blocks which concatenated together form the original message plus possible padding bytes in . The blocks are each individually run through the encryption algorithm using the same key and transformed into corresponding ciphertext blocks and which concatenated together form the ciphertext.

The decryption process is identical, only now the inputs to the decryption function are the ciphertext blocks, and of course the same key. Once the message is decrypted, the padding is removed, and the original message has now been recovered.





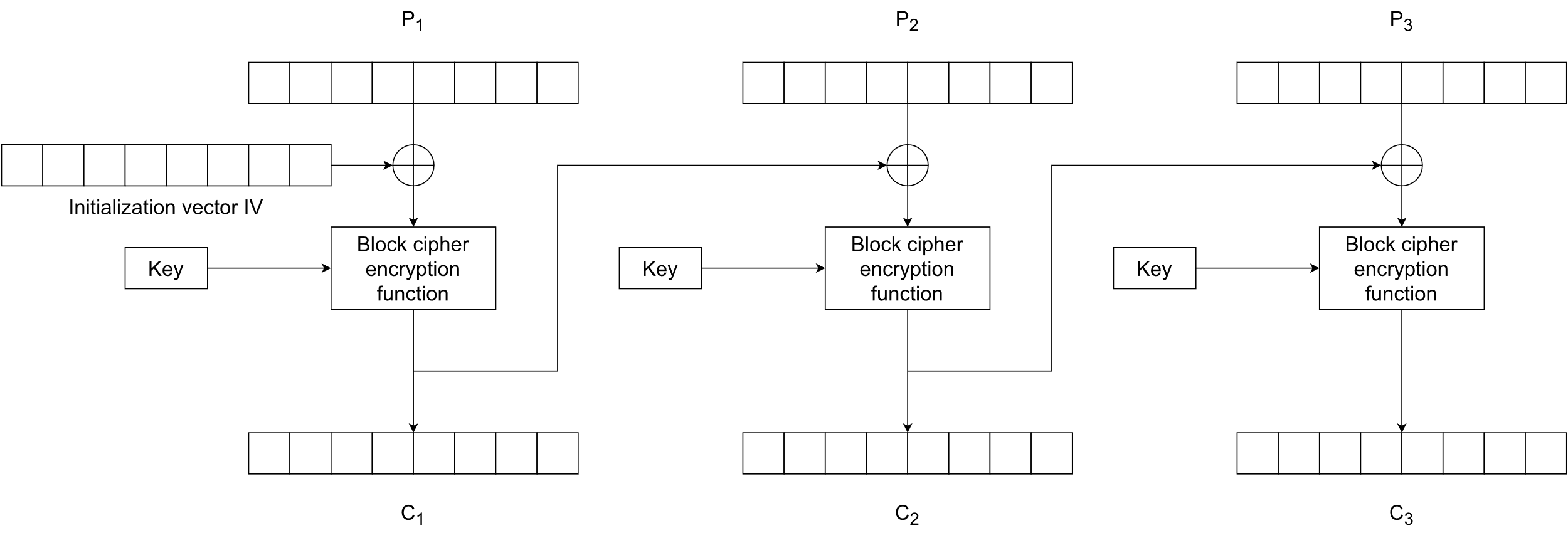
ECB is a mode that should never be used if any sort of proper secrecy is desired. Using even the most robust cipher algorithm will lead to weak encryption if combined with an unsecure mode of operation like ECB.

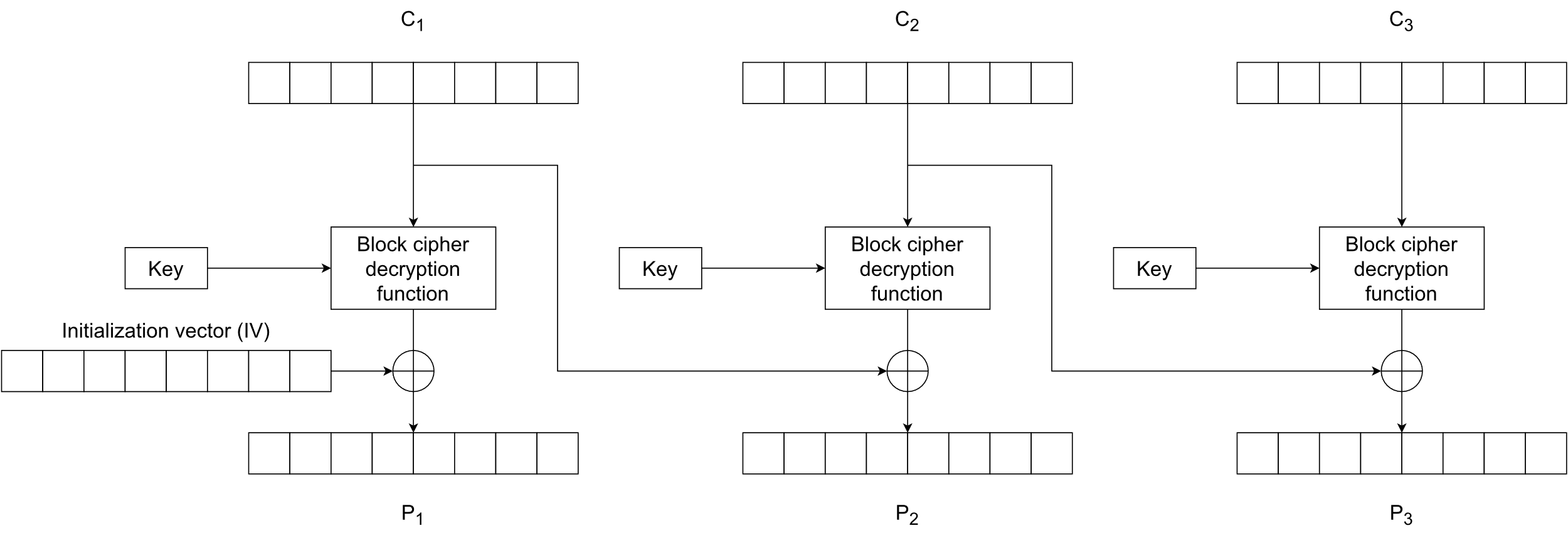
* 1. CBC – Cipher Block Chaining

In CBC each encrypted cipher block is dependant on the previous blocks. This is because before encrypting a plaintext block it is first XORed with the previous ciphertext block and the result is then encrypted. For the first plaintext block there isn’t a previous cipher block, so a random block called the initialization vector (IV) is used instead. The formula for encryption is and for decryption where . CBC also requires all of the blocks to be of the same length so the last block may contain padding bytes. Images 3 and 4 visualize the encryption and decryption processes with 8-byte blocks.

The chaining of blocks with the XOR operation causes an avalanche effect in the ciphertext so that the same message encrypted with the same key results in a totally different ciphertext if a different IV is used. Of course, re-using the same IV reduces the security of CBC to that of ECB mode so the IV has a critical role in CBC and should therefore be generated with care.

The receiver of a message encrypted with CBC mode must receive the IV as well, otherwise they won’t be able to decrypt the first ciphertext block. In general, the IV used for generating a particular ciphertext is sent along the ciphertext to the receiver. For CBC to work as intended, the IV must be random or pseudorandom to prevent possible attackers from predicting or guessing the IV for the next message and it should also be authenticated so attackers can’t tamper with it and make the first block decrypt to what they want. It doesn’t however need to be secret after the encryption process is done. [3]





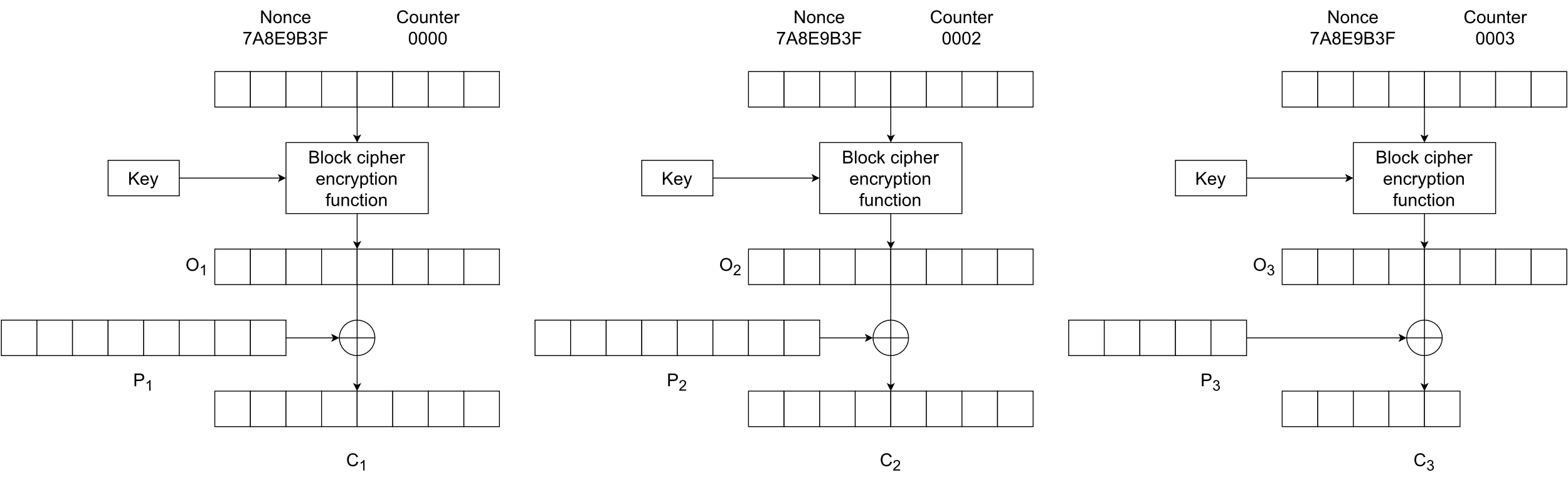
* 1. CTR – Counter Mode

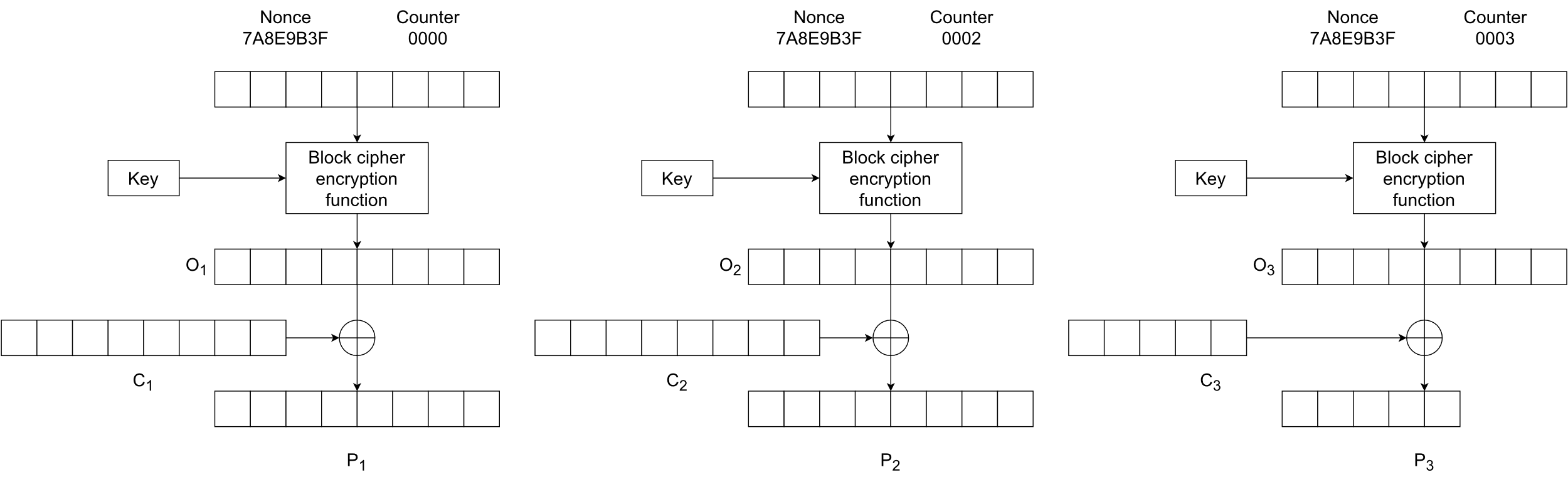
Counter mode is an older mode proposed by Whitfield Diffie and Martin Hellman in 1979. It gained interest in more recent years and was standardized by NIST in 2001. [5] It benefits from the possibility of parallel encryption and decryption since the blocks are processed independently of each other in both cases. [6]

Like CBC, CTR also needs an IV, usually called a counter, which in many cases actually consists of a nonce and a separate counter which are concatenated together, and both are bits long, where is the block length. For a given message the nonce stays the same for all blocks, while the counter is unique, and usually follows a simple increment-by-one procedure. The nonce should then be unique for all messages exchanged in a session and the counter unique for all blocks within a message. This way all nonce and counter combinations will be unique within a session. The nonce-counter combination does not need to be kept secret, only unique, and is often sent along the ciphertext block to the receiver for decryption. Like in the case with IVs in CBC mode, the purpose of the nonce-counter combination in CTR mode is to ensure identical plaintext blocks encrypt to different ciphertext blocks. The security of the cipher algorithm comes from the other input to the encryption function, namely the secret key, while the nonce-counter combination takes care of the uniqueness of the output.

Another useful feature of the CTR mode is that it does not require padding. This is because the last phase of encrypting is a XOR operation between output from the block cipher algorithm and a plaintext block. If the last plaintext block’s length is bits and it’s not the same as the current block size, only the most significant bits from the block cipher’s output are used for the XOR with the last plaintext block. This will result in the last ciphertext block to be of length also. The decryption process is almost identical to the encryption, only now the ciphertext blocks are XORed with the cipher’s output blocks. The same principal of only using the most significant bits of the last ciphertext block for the XOR applies. Images 5 and 6 illustrate the encryption and decryption processes.

Let the nonce/counter combinations be denoted by . The formula for encryption can then be written as , for and , where and denotes the length of the last plaintext block . For decryption it follows from the properties of the XOR operator that for and . Of note is that for both encryption and decryption using CTR mode only the block cipher encryption algorithm is needed. [5]





* 1. GCM – Galois/Counter Mode

Galois/Counter Mode uses encrypts messages in the same way that CTR does but adds a layer of authenticity to the encryption process. The result is a mode that combines the high speed and parallelization properties of CTR and also provides confidentiality and integrity of messages. [7]

Integrity of messages is achieved through calculating a message authentication tag for each encrypted message. The purpose of the tag is to tell the receiver if the received ciphertext was tampered with in transit or not. As it is implemented, the integrity of the message can be checked before decrypting the actual message since the tag is calculated from ciphertext blocks. This is an important security measure as it will help defend from chosen-ciphertext attacks.[8]

1. Comparison of modes
2. Selected flaws and vulnerabilities
3. Conclusions

References

[1] <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-107r1.pdf>

[2] <https://www.jscape.com/blog/stream-cipher-vs-block-cipher>

[3] <https://sweet32.info/>

[4] <https://defuse.ca/cbcmodeiv.htm>

[5] <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-38a.pdf>

[6] <https://www.cs.ucdavis.edu/~rogaway/papers/ctr.pdf>

[7]<https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf>

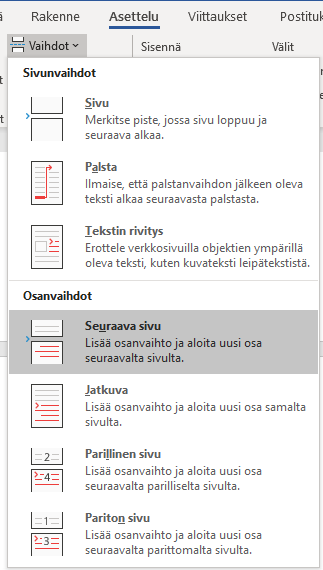
[8] <https://www.scottbrady91.com/c-sharp/aes-gcm-dotnet>

Virtanen, V. 2011. Esimerkkilähteen otsikko. LAB-ammattikorkeakoulu. Viitattu 1.1.2018. Saatavissa <http://esimerkki.fi>

Liite 1. Otsikko

Jos liite on yksisivuinen, siihen ei laiteta sivunumeroa.

Seuraavalla sivulla alkaa liite, joka on useampi sivuinen – tälle sivulle tulee tehdä osan vaihto: Asettelu/Vaihdot/Seuraava sivu.

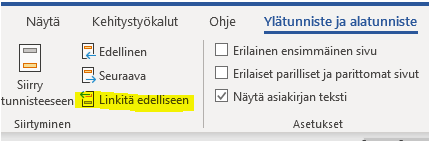


Kun piilomerkit ovat näkyvissä sivunvaihto näkyy dokumentissa seuraavasti:



Liite 2. Liitteen otsikko

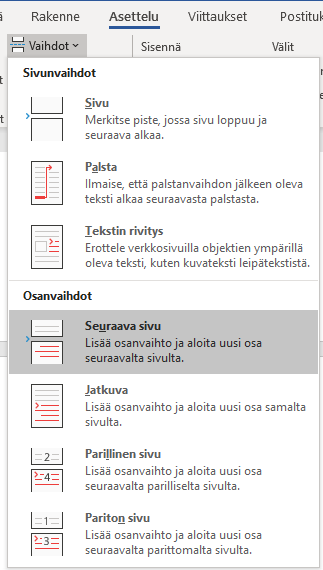
Jos liite on useampi sivuinen, sille täytyy laittaa sivunumerointi. Huom. ylätunnisteesta tulee ottaa Linkitä edelliseen pois päältä. (**Huom. edelliselle sivulle on tehty osan vaihto**)



Liite 2:n tarina jatkuu …

Liite 2:n kolmas sivu

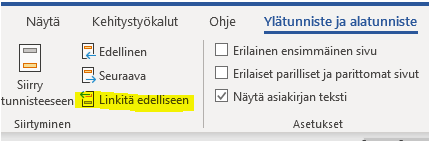
Jos tämän jälkeen tulee kolmas liite, niin tänne tulee tehdä osan vaihto. Asettelu/Vaihdot/Seuraava sivu.



Liite 3. Otsikko

Liite 3:n tarina alkaa tästä. Jos liitteessä on vain yksi sivu, sivunumeroa ei laiteta.

Tämä tarkoittaa, ylätunnisteesta otetaan pois Linkitä edelliseen



ja vasta tämän jälkeen poistetaan sivunumerointi.