

# The AES encryption algorithm

En undersökning av The Advanced Encryption Standard (AES)

Klass:

**NA20**

Handledare:

**Jimmy Nylén**

Författare:

**Gabriel Lindeblad**

Program:

**Naturvetenskapsprogrammet**

# Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut lacinia ex eget sagittis congue. Nullam cursus egestas dolor, suscipit gravida magna ultrices sit amet. Nullam placerat dui eu arcu pharetra, sit amet tempor dolor convallis. Aenean sodales condimentum turpis, commodo maximus augue. Aenean vel nibh dui. Pellentesque ex libero, lacinia nec mauris vel, convallis consectetur felis. Maecenas ut nibh sed magna maximus imperdiet at id purus. In vel consequat metus. Donec non tincidunt nunc. Sed pulvinar odio ut sapien vestibulum, quis mollis arcu tempor. Maecenas ut sem leo. Sed leo risus, mollis eu ex vitae, feugiat consequat metus. Aenean interdum volutpat urna, nec tempor mi accumsan quis. Morbi blandit maximus urna non aliquet aes.

# Innehåll

<b>Ordlista</b>	<b>4</b>
<b>Akronymer</b>	<b>6</b>
<b>1 Inledning</b>	<b>7</b>
1.1 Syfte . . . . .	7
1.2 Frågeställningar . . . . .	7
1.3 Avgränsning . . . . .	7
<b>2 Bakgrund</b>	<b>8</b>
2.1 Kryptografi . . . . .	8
2.1.1 Uppkomst . . . . .	8
2.1.2 Utveckling . . . . .	8
2.2 AES Uppkomst . . . . .	8
<b>3 Teori</b>	<b>9</b>
3.1 Kryptering . . . . .	9
3.2 Blockchiffer . . . . .	9
3.2.1 Körlägen . . . . .	9
3.2.1.1 ECB . . . . .	10
3.2.1.2 CBC . . . . .	11
3.2.1.3 OFB . . . . .	13
3.3 Symetrisk & Asymmetrisk Kryptering . . . . .	14
3.4 AES . . . . .	14
3.4.1 Finite Fields . . . . .	15
3.4.2 AES S-Box . . . . .	15
3.4.3 Struktur . . . . .	15
3.4.3.1 SubBytes operation . . . . .	15
3.4.3.2 ShiftRows operation . . . . .	15
3.4.3.3 MixColumns operation . . . . .	15
3.4.3.4 AddRoundKey operation . . . . .	15
3.4.4 Nyckel utökning . . . . .	15
3.4.4.1 RotWord . . . . .	15
3.4.4.2 SubWord . . . . .	15
3.4.4.3 Rcon . . . . .	15
3.4.5 AES-128bit . . . . .	15
3.4.6 AES-192bit . . . . .	15
3.4.7 AES-256bit . . . . .	15
<b>4 Metod &amp; Genomförande</b>	<b>16</b>
4.1 Implementering . . . . .	16
4.2 Test Uppsättning . . . . .	16
4.3 Genomförande . . . . .	16
<b>5 Resultat</b>	<b>17</b>
5.1 Nyckellängds Test . . . . .	17
5.2 Körläges Test . . . . .	17

## *INNEHÅLL*

---

<b>6 Diskussion &amp; Slutord</b>	<b>18</b>
6.1 Felkällor . . . . .	18
6.2 Förbättringar . . . . .	18
6.3 Slutsats . . . . .	18
6.4 Slutord . . . . .	18
<b>Källförteckning</b>	<b>19</b>
<b>Figurer</b>	<b>21</b>

# Ordlista

## Bit

En Bit är den minsta enheten av information som kan lagras i en dator. En bit kan endast ha två värden där den antingen är 0 eller 1. I datorvetenskap pratar man dock mer vanligen om ett Byte som är 8 bits.[[Wik22a](#)]

## Byte

En Byte består av 8 Bits och är en enhet som används inom datorvetenskap. En byte kan ha 256 olika värden från 0 till 255, vilket är  $2^8$  värden. Dessa värden representerar ofta tecken eller symboler som exempelvis bokstäver, siffror och siffermässiga tecken. Tolkningen av vad en sekvens av bytes eller en enskild byte står för beror dock på vilken teckenkodning som används. Exempel på teckenkodningar har varit ASCII och ISO-8859.[[Wik20](#)]

## Caesarchiffer

Caesarchiffer är ett substitutionschiffer, vilket helt enkelt bygger på att man byter ut varje bokstav i medelandet med en annan. Ersättningens bokstaven bestäms enligt en kod, till exempel att man hoppar ett visst antal hopp i alfabetet som exempelvis 3 hopp, vilket då innebär att om man har bokstaven a så skulle den bli ett d istället.[[Wik21a](#)]

## Frekvensanalys

Frekvensanalys inom kryptografi är en metod för att knäcka ett Substitutionsschlüssel genom att analysera frekvensen av bokstäver och utnyttja de faktum att en viss bokstaver framkommer mer frekvent än andra i språket. På detta viset kan man sedan lista ut vilka bokstäver som är vilka i det krypterade medelandet.[[Wik21c](#)]

## Hashfunktion

Hashfunktion är en funktion som man kan säga delar upp en viss datamängd och genomför en serie operationer som resulterar i hashtext av godkänd längd. Längden är samma för alla hastexter som använder samma funktion medan innehållet förändras så fort något över huvud taget ändras i datan som funktionen appliceras på. Användningsområdet för dessa funktioner är bland annat när man vill kunna verifiera medelanden eller information och försäkra sig om att informationen inte är ändrat på medelandet efter att de skickats. Detta kan man då göra för att kontrollera om man skört informationen genom samma hashfunktion borde resulttera i samma hashtext och därmed vara identisk om informationen är oförändrad.[[Wik22j](#)]

## Nyckelström

En nyckelström är i kryptografin en ström av Pseudoslump karaktärer som kan kombineras med exempelvis ett medelande för att producera en skrivtext.[[Wik21b](#)]

## Polyalfabetiska skiffer

Polyalfabetiska skiffer bygger på att man använder flera olika Substitutionsschlüssel för att på så sätt undvika en ut av de största svagheterna med Substitutionsschlüssel. Detta då att dom lätt går att knäcka genom en Frekvensanalys då vissa bokstäver dyker upp mer frekvent i språket än andra. För att lösa detta så använder man polyalfabetiska skiffer flera olika substitutionsskiffer som man byter mellan medan man ändrar frekvens för att eliminera Frekvensanalysens effektivitet.[[Wik22k](#)]

<b>Pseudoslump</b>	Pseudoslump är en rad av nummer som kan se ut att vara helt slumpmässiga men har blivit framställda genom en upprepbar process.[Wik22l]
<b>Python</b>	Python är ett högnivå programmerings språk byggt på programmerings språket C. De är skapat av Guido van Rossum och släpptes i Februari 1991.[Pyt22]
<b>RSA</b>	Rivest-Shamir-Adleman (RSA) är en av de mest välkända krypteringsalgoritmena och var en av de första algorithmerna som byggde på en asymetrisk kryptering. RSA bygger på multiplikation av stora primtal där primtalen är nycklarna.[Wik22m]
<b>Strömskiffer</b>	Strömskiffer, ett symmetriskt nyckel skiffer där man använder en Pseudoslumpmässig skiffer ström (Nyckelström) som sedan en Bit i taget kombineras med de som ska krypteras. Den kombinerande operationen som används strömskiffer är ofta en XOR-operation.[Wik22n] ... [Wik22o]
<b>VSCode</b>	Visual Studio Code är en programutvecklingsmiljö som är skapad av Microsoft. Det är ett öppet källkods projekt som är tillgängligt för det flesta operativsystem och kan användas för att skriva kod i flera olika språk.[Wik21c]
<b>XOR</b>	Ett logisk operation inom datorvetenskap som fungerar ungefär som + uttrycket, med den enda skillnaden att $1 \oplus 1 = 0$ . Detta samt att xor är en binär operation, vilket innebär att termerna bara kan vara 0 eller 1 och resultatet därför samma.[LEW12]

# Akronymer

<b>AES</b>	Advanced Encryption Standard
<b>ASCII</b>	American Standard Code for Information Interchange
<b>CBC</b>	Cipher Block Chaining läge
<b>DES</b>	Data Encryption Standard
<b>ECB</b>	Electronic Code Book läge
<b>IV</b>	Initialization Vector
<b>OFB</b>	Output Feedback läge
<b>SSL</b>	Secure Socket Layer
<b>TLS</b>	Transport Layer Security

# 1 Inledning

Kryptering, en bärande grundsten i dagens digitaliserade samhälle. De är väggen mellan oss och resten av världen, ett läs runt våra liv. Kryptering bygger på ett simpelt koncept, att dölja informationen från all förutom den menade mottagaren. Ett koncept som exempelvis fanns redan för 2000 år sedan när Julius Caesar använde de vi idag kallar Caesarchiffer för att skicka hemliga meddelanden.<sup>1</sup>

Sedan dess har kryptografi självklart utvecklats enormt och vi har gått från de på ett sätt simpla men även eleganta Caesarchiffer som användes då till moderna algoritmer så som Advanced Encryption Standard och Data Encryption Standard. Dessa algoritmer har samma syfte som Caesarchiffer men har utvecklats under en tid där datorer står som de dominerande informationshanteringsverktyget, vilket även är vad som används i denna rapport för att undersöka just en av dessa algoritmer.

## 1.1 Syfte

Syftet med denna undersökning är att undersöka krypterings algoritmen AES, för att utveckla en förståelse för mer avancerade krypterings algoritmer. Samt att bygga en uppfattning om hur man på olika sätt kan implementera krypterings algoritmer och vad de får för betydelse för deras säkerhet och hastighet.

## 1.2 Frågeställningar

- Hur påverkas tiden de tar att kryptera något mellan de olika nyckel längderna 128-bit, 192-bit och 256-bit nyckel?
- Hur påverkas skifertexten av de olika körlägen och vilken betydelse får de för den resultatet?
- Hur förändras tiden det tar att kryptera något beroende på ifall algoritmen körs i ECB, CBC eller OFB samt vilken betydelse det får ur ett tillämpningsperspektiv?

## 1.3 Avgränsning

Denna rapport är en avgränsad utvärdering av AES och dess användning som fokuserar på hur nyckellängd och körläge påverkar krypteringstiden. Detta samt hur den resulterande skiffer texten påverkas av vissa körlägen och hur detta i sin tur kan påverka säkerheten.

Denna analys av algoritmens säkerhet utelämnar faktorer så som möjliga attacker där ibland exempelvis Brute-Force<sup>2</sup> & Side-Channel<sup>3</sup> attacker. Undersökningen är även begränsad till en mjukvaruimplementering och tar inte hänsyn till möjliga skillnader som kan uppstå när algoritmen implementeras på en hårdvarunivå.

---

<sup>1</sup>Dennis Luciano och Gordon Prichett. “Cryptology: From Caesar ciphers to public-key cryptosystems”. I: *The College Mathematics Journal* 18.1 (1987), s. 2–17.

<sup>2</sup>Neeraj Kumar. “Investigations in brute force attack on cellular security based on des and aes”. I: *IJCEM International Journal of Computational Engineering & Management* 14 (2011), s. 50–52.

<sup>3</sup>“Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA”. I: *Cryptographic Hardware and Embedded Systems - CHES 2009*. Utg. av Christophe Clavier och Kris Gaj. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, s. 97–111. ISBN: 978-3-642-04138-9.

# 2 Bakgrund

## 2.1 Kryptografi

Ordet kryptografi härstammar från de två grekiska orden kryptos som betyder gömd och grafein som betyder skrift.<sup>4</sup> I sin simplaste form handlar kryptografi alltså om att gömma information. Detta är något som har visat sig på många olika sätt genom historien från något så simpelt som att skriva ett medelande i text då många i början inte kunde läsa till att idag istället använda komplexa algoritmer.<sup>5</sup> Begreppet kryptografi har dock också fått en utökade betydelse med tiden då det idag även inkluderar olika metoder för att säkerställa autenticiteten av informationen och avsändaren.<sup>6</sup>

### 2.1.1 Uppkomst

Kryptografins historia kan man nästan säga börjar vid den tidigaste formen av skrift, vilket grundar sig i de faktum att de flesta inte kunde läsa. Detta är ju såklart något som förändrats på senare tid och i takt med de så har även kryptografin utvecklats. Exempel på utvecklingen går att se så tidigt som 1900 f.Kr då vissa egyptiska skribenter använde sig utav hieroglyfer på ett avvikande sätt, vilket troligen då gjordes i syfte att dölja informationen från dom som inte visste vad det skulle betyda.<sup>7</sup>

Den tidiga kryptografin är även något som kan observeras hos romarna där man använde Caesarchiffer och hos grekerna. Där grekernas metod byggde på att man virade en tejpbil runt någon form av ett cylinderformat objekt och sedan skrev medelandet på tejpen. När tejpen sedan togs av så är texten oläslig och mottagaren behövde vira upp tejpen på ett cylinderformat objekt med samma diameter för att läsa det.<sup>8</sup>

### 2.1.2 Utveckling

Utvecklingen av kryptografin som en vetenskap och teknik såg dock inga större framsteg ända till medeltiden. När utvecklingen ändå började ta fart igen så använde bland annat nästan alla Europeiska nationer någon form av kryptografi för att dölja medelande och hemlig kommunikation. Under den här tiden utvecklades bland annat Polyalfabetiska skiffer där ett av dom tidigaste skapades av Leon Battista Alberti.<sup>9</sup>

...<sup>10</sup>

## 2.2 AES Uppkomst

---

<sup>4</sup>Wikipedia. *Kryptografi*. 2020. URL: <https://sv.wikipedia.org/w/index.php?title=Kryptografi&oldid=48532107> (hämtad 2022-09-07).

<sup>5</sup>Tony M Damico. "A brief history of cryptography". I: *Inquiries Journal* 1.11 (2009).

<sup>6</sup>Nationalencyklopedin. *kryptografi*. 2022. URL: <http://www.ne.se/uppslagsverk/encyklopedi/lng/kryptografi> (hämtad 2022-09-07).

<sup>7</sup>Dam09.

<sup>8</sup>Dam09.

<sup>9</sup>Dam09.

<sup>10</sup>Dam09.

# 3 Teori

## 3.1 Kryptering

Kryptering handlar om att gömma information för att förhindra att andra än den menade mottagaren kan läsa informationen. Detta genomförs i dagens samhälle genom olika typer av krypteringsalgoritmer. Dessa algoritmer kan ses som både komplicerade och förvirrande men bygger på enkla principer. En krypteringsalgoritm tar in en text och en nyckel och ger sedan tillbaka en skiftext, vilket då är en krypterad version av den ursprungliga texten.<sup>11</sup>

För att den menade mottagaren ska kunna läsa skiftexten sedan så behöver hen ha en nyckel samt köra samma krypteringsalgoritm i dekrypterings läge. Bland de vanligaste typerna av krypteringsalgoritmer finns Symetrisk & Asymmetrisk Kryptering, vilket bland annat innehållar algoritmer som AES och RSA.<sup>12</sup>

## 3.2 Blockchiffer

Blockchiffer är en krypteringsalgoritm som verkar på block av data med en fast storlek. blockchiffer används bland annat som en av grundkomponenterna i kryptografiska protokoll som Transport Layer Security (TLS) och Secure Socket Layer (SSL) som används för att kryptera data som skickas över internet bland annat.<sup>13</sup>

En ut av de största problemen med Blockchiffer är dock att oavsett hur säkra dom är så passar det bara att använda för kryptering av enskilda block med en nyckel, vilket skiljer sig från något som ett Strömskiffer. På grund av detta har en mängd olika körlägen utvecklats för att på så sätt göra det möjligt att utnyttja blockchiffer för att kryptera större mängder data med en nyckel.<sup>14</sup>

Blockchiffer används däremot inte bara för kryptering utan kan även användas i olika typer av Hashfunktioner och Pseudoslämp nummer generatorer bland annat då blockchifferets resulterande skiftext ser ut att vara slumpmässig.<sup>15</sup>

### 3.2.1 Körlägen

Körlägen inom kryptografin kan man se som algoritmer som appliceras i användningen av blockchiffer eftersom dessa endast är användbara för säker kryptering av ett litet block med bestämd längd. Körlägena gör det möjligt att istället kunna använda blockchiffer på större data mängder. Detta löser körlägen på olika sätt beroende på hur man vill använda blockchiffer algoritmen samt hur mycket man är villig att kompromissa med säkerheten i förhållande till hastighet.<sup>16</sup>

---

<sup>11</sup>Wikipedia. *Kryptering*. 2021. URL: <https://sv.wikipedia.org/w/index.php?title=Kryptering&oldid=49187134> (hämtad 2022-09-08).

<sup>12</sup>Wik21.

<sup>13</sup>Wikipedia, the free encyclopedia. *Block cipher*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Block\\_cipher&oldid=1111913955](https://en.wikipedia.org/w/index.php?title=Block_cipher&oldid=1111913955) (hämtad 2022-10-05).

<sup>14</sup>Wik22h.

<sup>15</sup>Wik22h.

<sup>16</sup>Wikipedia. *Block cipher mode of operation*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Block\\_cipher\\_mode\\_of\\_operation&oldid=1106163325](https://en.wikipedia.org/w/index.php?title=Block_cipher_mode_of_operation&oldid=1106163325) (hämtad 2022-09-25).

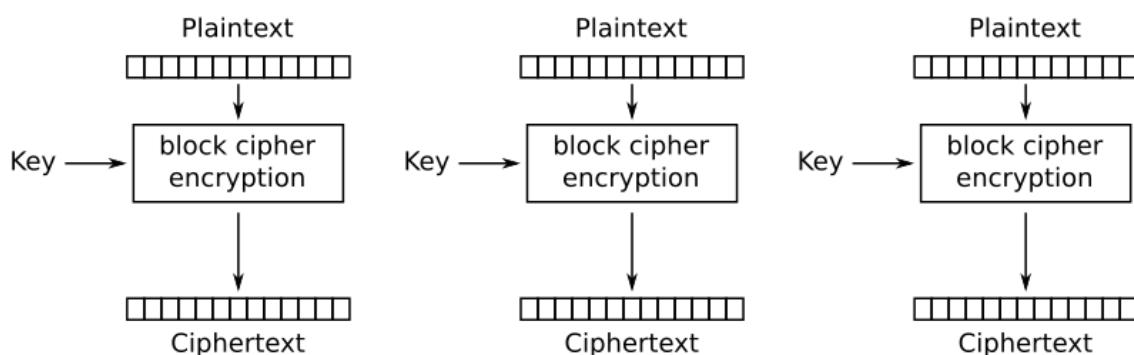
Ett ut av sätten som körlägen löser problemet med att applicera blockskiffer algoritmer på större data mängder är att dom använder sig av en så kallad Initialization Vector (IV). Det är en unik sekvens av bytes som används för att säkerställa att samma data mängd aldrig kommer att generera samma krypterade skifffertext.<sup>17</sup>

IV kan användas på många sätt från att introduceras genom en XOR-operation med första blocket och sedan kedja ihop och göra samma sak med nästkommande block fast med resultatet från den första operationen precis som i CBC. Detta gör att blocken blir beroende av varandra plus att samma information som krypteras flera gånger inte kommer ge samma resultat även fast man använder samma nyckel.<sup>18</sup>

### 3.2.1.1 ECB

Electronic Code Book läge (ECB) är en av det enklaste blockchiffer körlägena som finns. ECB i sig är ganska lätt att förstå och bygger i huvudsak bara på att man delar upp den data man vill kryptera i delar kallade block och tar sedan varje block för sig och kör genom algoritmen, vilket tydligt visas i figur 3.1 & 3.2.<sup>19</sup>

Figur 3.1 visar hur ECB fungerar vid kryptering. Här visas hur varje block för sig krypteras med hjälp av en blockchiffer algoritm tillsammans med den givna nyckeln.



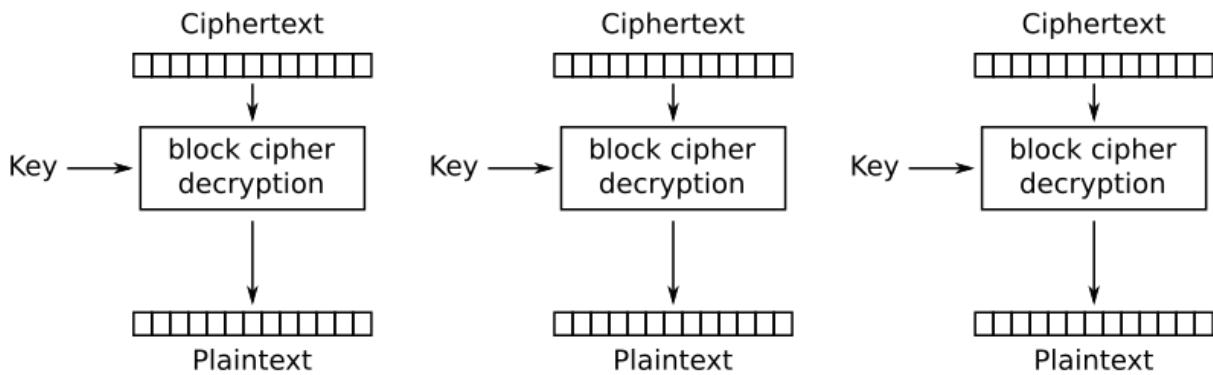
**Figur 3.1:** Electronic Code Book läge kryptering [[Wik22b](#)]

Figur 3.2 visar istället hur ECB fungerar vid dekryptering, vilken är en till stort sett identisk operation med det enda undantaget att blockchiffret körs i dekrypterings läge istället för krypterings läge.

<sup>17</sup>[Wik22a](#).

<sup>18</sup>[Wik22a](#).

<sup>19</sup>[Wik22a](#).



**Figur 3.2:** Electronic Code Book läge dekryptering [Wik22c]

På grund av ECB körlägets simplicitet så finns det dock även ett ganska stort problem med detta körläge. Det handlar om att ECB inte på något sätt förhindrar att två block med samma innehåll som krypteras inte resulterar i ett identiskt krypterat block.<sup>20</sup>

Vad detta innebär är att för större mängder data är att det börjar bildas mönster i skiffertexten. Detta är något som väldigt tydligt visar sig ifall man krypterar en bild, vilket går att se när man jämför figur 8.1 & 8.2. Det här faktumet är även varför ECB inte är ett säkert körläge och därför inte används näst intill aldrig i praktiken.<sup>21</sup>

ECB har dock ändå sina fördelar då de bland annat kan parallelliseras både när de gäller krypteringen och dekrypteringen. Detta samt att ECB även gör det möjligt att slumprässigt dekryptera enskilda block av en skiffertext utan att man behöver dekryptera hela texten.<sup>22</sup>

### 3.2.1.2 CBC

Cipher Block Chaining läge är ett av de mest vanligen använda körlägena för många blockchiffer. Till skillnad från ECB så förhindrar CBC att två block med samma innehåll kan ge samma krypterade block. Detta gör CBC genom att lägga till ett extra steg utöver vad som finns i ECB. Steget är en XOR-operation mellan det krypterade blocket näckommande block innan de körs genom blockchiffer algoritmen.<sup>23</sup> Matematiskt sett kan detta formuleras såhär:

$$\begin{aligned} S_i &= K_n(B_i \oplus S_{i-1}) \\ S_0 &= IV \end{aligned}$$

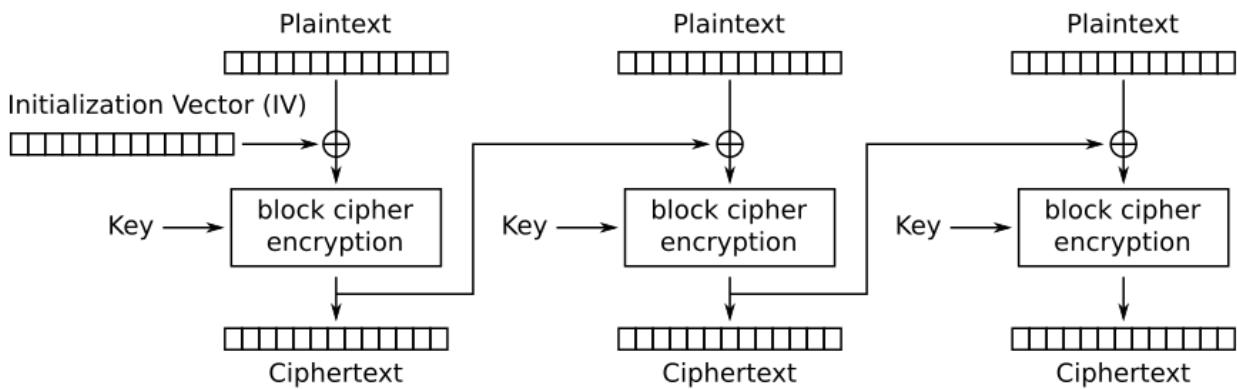
Där  $S_i$  är det krypterade blocket(skiffertexten),  $B_i$  är det blocket som ska krypteras,  $K_n$  är blockchiffer algoritmen där  $n$  står för nyckeln och  $S_{i-1}$  är det krypterade blocket före det blocket som ska krypteras. IV är en Initialization Vector (IV) som används vid krypteringen av de första blocket då de inte finns något föregående block att använda.  $i$  står för index där de första blocket har index värdet 1. Hela den här processen kan även ses i figur 3.3.

<sup>20</sup>Wik22a.

<sup>21</sup>Wik22a.

<sup>22</sup>Wik22a.

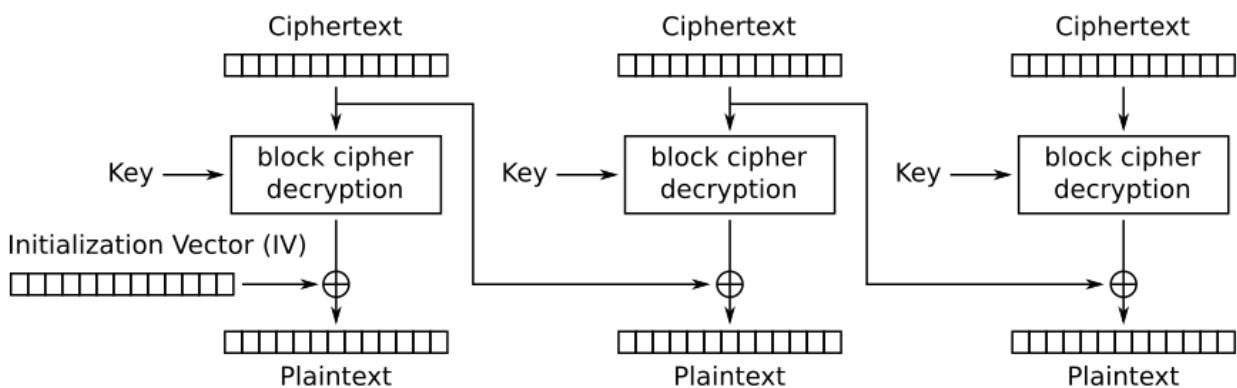
<sup>23</sup>Wik22a.



**Figur 3.3:** Cipher Block Chaining läge kryptering [Wik22d]

När de gäller dekrypteringsprocessen för CBC så bär den precis som för ECB stora likheter med krypteringsprocessen. Det två skillnaderna som finns är att blockchiffert körs i dekrypteringsläge istället för krypteringsläge. Samt att för varje block så genomförs en XOR-operation mellan det dekrypterade blocket och föregående block innan dekrypteringen av blocket.<sup>24</sup> Även detta går att både matematiskt formulera och visuellt visa så här:

$$\begin{aligned} B_i &= K_n(S_i) \oplus S_{i-1} \\ S_0 &= IV \end{aligned}$$



**Figur 3.4:** Cipher Block Chaining läge dekryptering [Wik22e]

Fördelarna som kommer från den extra operationen i CBC till skillnad från ECB är då att varje block blir beroende av föregående block. Detta innebär att dom mönster som kunde dyka upp i ECB inte längre kan uppstå, vilket då gör CBC till ett mer säkert körläge än ECB. Dock kräver cbc en ytterligare faktor för att se till så att inte olika medelanden kan ge samma krypterade block. Därför så krävs en Initialization Vector (IV) som används vid första blocket.<sup>25</sup>

CBC är dock inte prefekt och har i sig också några nackdelar. Där ibland exempelvis de faktum att en incorrect IV leder till att de första blocket inte kan dekrypteras korrekt, detta påverkar dock inte de resterande blocken. På grund av det så kan man exempelvis lösa problemet genom

<sup>24</sup>Wik22a.

<sup>25</sup>Wik22a.

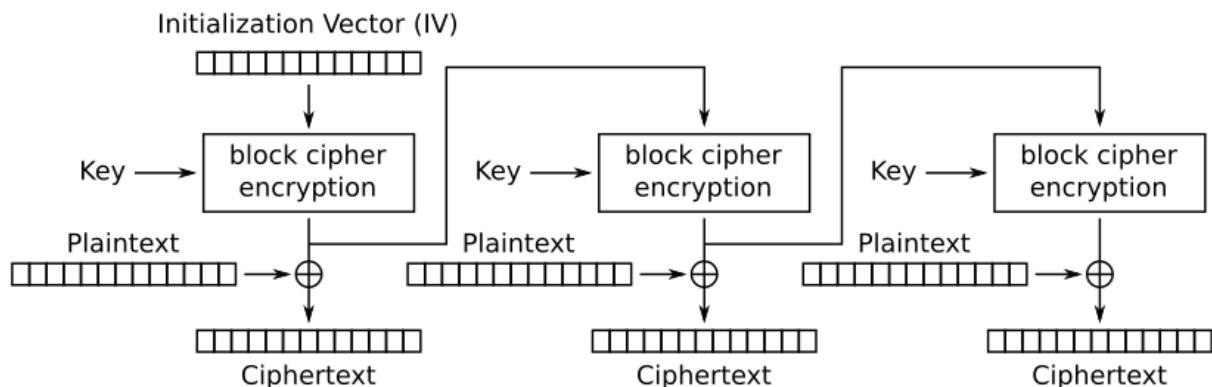
att första blocket bara innehåller någon typ av fyllnad, vilket då gör dekrypteringen möjlig utan tillgång till IV.<sup>26</sup>

Utöver detta så begränsas även CBC till att bara vara parallellisierbar under dekrypteringen och inte krypteringen, vilket är en konsekvens av att varje block i CBC är beroende av föregående block. CBC behåller dock fortfarande möjligheten som ECB har att slumpmässigt dekryptera enskilda block utan att behöva dekryptera hela skiftexten.<sup>27</sup>

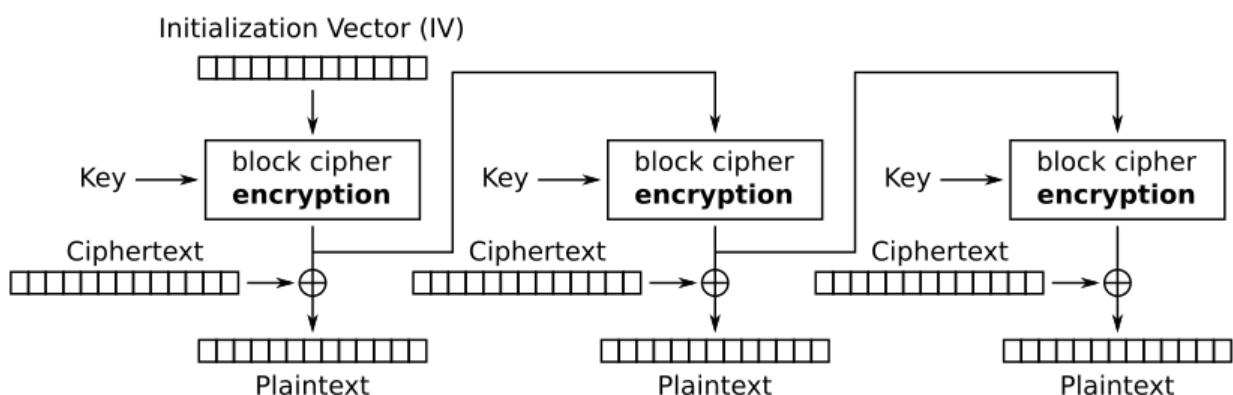
### 3.2.1.3 OFB

Output Feedback läge är ett ytterligare körläge som skiljer sig en del från ECB och CBC som redan presenterats. Den största skillnaden från det andra körlägena är att OFB inte använder blockchiffer algoritmen för att kryptera eller dekryptera blocken. Istället så körs IV genom blockchiffer algoritmen och den resulterande Nyckelström tillförs sedan genom en XOR-operation till blocket som ska krypteras eller dekrypteras.<sup>28</sup>

Tack vare XOR-operationens symmetriska natur så är så väl krypteringen som dekrypteringen av OFB identisk, vilket även visas i figur 3.5 & 3.6:



**Figur 3.5:** Output Feedback läge kryptering [Wik22f]



**Figur 3.6:** Output Feedback läge dekryptering [Wik22g]

Utöver detta kan man även matematiskt beskriva OFB, vilket visas i ekvationen nedan:

<sup>26</sup>Wik22a.

<sup>27</sup>Wik22a.

<sup>28</sup>Wik22a.

$$\begin{aligned}
 S_i &= B_i \oplus O_i \\
 B_i &= S_i \oplus O_i \\
 O_i &= K_n(I_i) \\
 I_i &= O_{i-1} \\
 I_0 &= IV
 \end{aligned}$$

Här visas OFB körläget matematiskt där  $S_i$  är det krypterade blocket,  $B_i$  är blocket som ska krypteras och  $I_0$  är IV. Men här finns även  $O_i$  som man kan säga är själva Nyckelström som används för att kryptera eller dekryptera blocket.  $O_i$  i sin tur bygger då på att  $O_{i-1}$  körs genom blockchiffer algoritmen igen och sedan används för nästa blocks kryptering.

På grund av att OFB är utformat på det här sättet och att själva blocken som ska krypteras inte används fram till sista steget så är det möjligt att genomföra blockskiffer operationerna i förväg, vilket gör det möjligt att även parallellisera OFB. Dock kan OFB inte parallelliseras ifall man inte gör blockskiffer operationerna i förväg. Utöver detta saknar även OFB möjligheten att slumpmässigt dekryptera enskilda block utan att behöva dekryptera hela skiffertexten.<sup>29</sup>

### 3.3 Symmetrisk & Asymmetrisk Kryptering

Symmetrisk och asymmetrisk kryptering handlar om hur nycklar används i olika krypteringsalgoritmer. För symmetriska krypterings algoritmer så betyder detta att samma nyckel är vad som används för både kryptering och dekryptering. Medan asymmetrisk kryptering bygger på att man använder olika nycklar för kryptering och dekrypterings processerna.<sup>30</sup>

De symmetriska krypterings algoritmernas huvudsakliga nackdel ligger i de faktum att de krävs en delad känd nyckel mellan båda parter. Detta är något som asymmetriska krypterings algoritmer inte behöver, vilket har lett till att man ofta använder asymmetriska krypterings algoritmer för att sköta nyckelutbytet för de symmetriska krypterings algoritmerna. Anledningen till detta är att det symmetriska krypterings algoritmerna ofta är bättre för större data mängder då dom bland annat behöver mycket kortare nyckellängder.<sup>31</sup>

Exempel på symmetriska krypterings algoritmer är bland annat AES och DES varav AES kommer förklaras djupare senare i denna rapport.<sup>32</sup> Medan exempel på asymmetriska krypterings algoritmer är bland annat RSA.<sup>33</sup>

### 3.4 AES

Som nämnts tidigare i AES Uppkomst bygger AES standarden på Rijndael blockchiffer algoritmen skapad av ...

---

<sup>29</sup>Wik22a.

<sup>30</sup>Wikipedia. *Symmetric-key algorithm*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Symmetric-key\\_algorithm&oldid=1106743629](https://en.wikipedia.org/w/index.php?title=Symmetric-key_algorithm&oldid=1106743629) (hämtad 2022-09-25).

<sup>31</sup>Wik22b.

<sup>32</sup>Wik22b.

<sup>33</sup>Wikipedia, the free encyclopedia. *RSA*. 2022. URL: <https://sv.wikipedia.org/w/index.php?title=RSA&oldid=50280992> (hämtad 2022-10-04).

**3.4.1 Finite Fields****3.4.2 AES S-Box****3.4.3 Struktur****3.4.3.1 SubBytes operation**

SubBytes operationen bygger på ...

**3.4.3.2 ShiftRows operation****3.4.3.3 MixColumns operation****3.4.3.4 AddRoundKey operation****3.4.4 Nyckel utökning****3.4.4.1 RotWord****3.4.4.2 SubWord****3.4.4.3 Rcon****3.4.5 AES-128bit****3.4.6 AES-192bit****3.4.7 AES-256bit**

# 4 Metod & Genomförande

Metoden för denna undersökning bygger på en implementering av AES i programmeringsspråket Python. Detta tillsammans med ett antal konstruerade tester även dom implementerade i Python är vad som används för själva undersökningen av AES. Själva koden är skriven med hjälp av programmet VSCode och är byggd huvudsakligen för Python 3.10.

## 4.1 Implementering

Implementeringen av AES är uppdelad i ett antal funktioner till stor del är baserat på hur strukturen och uppdelningen av AES beskrivs i “AES proposal: Rijndael”<sup>34</sup>...

## 4.2 Test Uppsättning

Test uppsättningen går att se i filen Analyze.py ...

## 4.3 Genomförande

---

<sup>34</sup>DR99.

# **5 Resultat**

## **5.1 Nyckellängds Test**

## **5.2 Körläges Test**

# **6 Diskussion & Slutord**

**6.1 Felkällor**

**6.2 Förbättringar**

**6.3 Slutsats**

**6.4 Slutord**

XOR

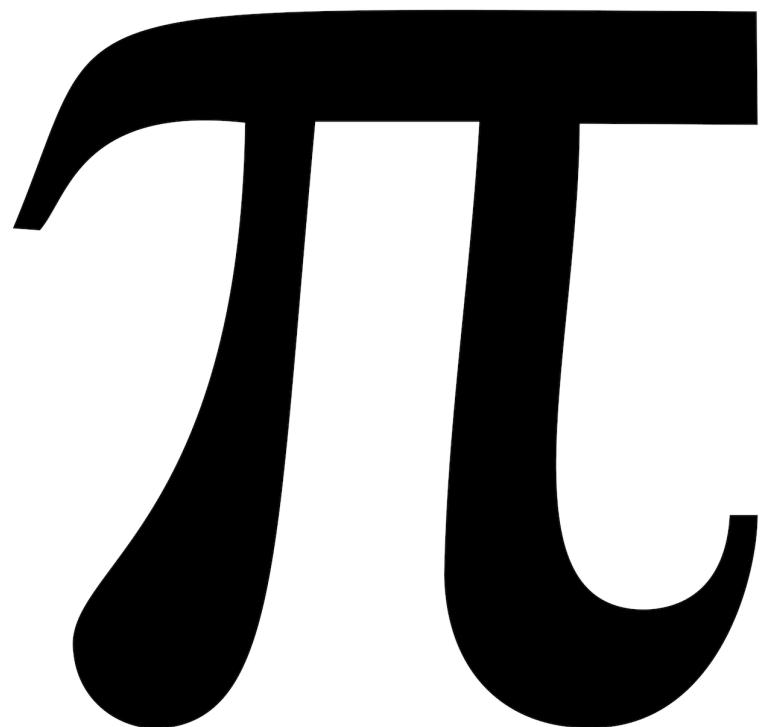
# Källförteckning

- [CG09] “Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA”. I: *Cryptographic Hardware and Embedded Systems - CHES 2009*. Utg. av Christophe Clavier och Kris Gaj. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, s. 97–111. ISBN: 978-3-642-04138-9.
- [DR99] Joan Daemen och Vincent Rijmen. “AES proposal: Rijndael”. I: (1999).
- [Dam09] Tony M Damico. “A brief history of cryptography”. I: *Inquiries Journal* 1.11 (2009).
- [Kum11] Neeraj Kumar. “Investigations in brute force attack on cellular security based on des and aes”. I: *IJCEM International Journal of Computational Engineering & Management* 14 (2011), s. 50–52.
- [LEW12] FEATURE MICHAEL LEWIN. “All about XOR”. I: *For details of ACCU, our publications and activities, visit the ACCU website: www. accu. org* (2012), s. 14.
- [LP87] Dennis Luciano och Gordon Prichett. “Cryptology: From Caesar ciphers to public-key cryptosystems”. I: *The College Mathematics Journal* 18.1 (1987), s. 2–17.
- [Nat22] Nationalencyklopedin. *kryptografi*. 2022. URL: <http://www.ne.se/uppslagsverk/encyklopedi/lng/kryptografi> (hämtad 2022-09-07).
- [Pyt22] Python Software Foundation. *What is Python?* 2022. URL: <https://docs.python.org/3/faq/general.html#what-is-python> (hämtad 2022-09-01).
- [Wik20] Wikipedia, the free encyclopedia. *Byte*. 2020. URL: <https://sv.wikipedia.org/w/index.php?title=Byte&oldid=48406212> (hämtad 2022-10-05).
- [Wik20] Wikipedia. *Kryptografi*. 2020. URL: <https://sv.wikipedia.org/w/index.php?title=Kryptografi&oldid=48532107> (hämtad 2022-09-07).
- [Wik21a] Wikipedia, the free encyclopedia. *Caesarchiffer*. 2021. URL: <https://sv.wikipedia.org/w/index.php?title=Caesarchiffer&oldid=48885737> (hämtad 2022-09-23).
- [Wik21b] Wikipedia, the free encyclopedia. *Keystream*. 2021. URL: <https://en.wikipedia.org/w/index.php?title=Keystream&oldid=1039345792> (hämtad 2022-10-04).
- [Wik21c] Wikipedia, the free encyclopedia. *Visual Studio Code*. 2021. URL: [https://sv.wikipedia.org/w/index.php?title=Visual\\_Studio\\_Code&oldid=48905230](https://sv.wikipedia.org/w/index.php?title=Visual_Studio_Code&oldid=48905230) (hämtad 2022-10-04).
- [Wik21] Wikipedia. *Kryptering*. 2021. URL: <https://sv.wikipedia.org/w/index.php?title=Kryptering&oldid=49187134> (hämtad 2022-09-08).
- [Wik22a] Wikipedia, the free encyclopedia. *Bit*. 2022. URL: <https://sv.wikipedia.org/w/index.php?title=Bit&oldid=51073011> (hämtad 2022-10-05).
- [Wik22a] Wikipedia. *Block cipher mode of operation*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Block\\_cipher\\_mode\\_of\\_operation&oldid=1106163325](https://en.wikipedia.org/w/index.php?title=Block_cipher_mode_of_operation&oldid=1106163325) (hämtad 2022-09-25).
- [Wik22b] Wikipedia, the free encyclopedia. *Block cipher mode of operation*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Block\\_cipher\\_mode\\_of\\_operation&oldid=1106163325#/media/File:ECB\\_encryption.svg](https://en.wikipedia.org/w/index.php?title=Block_cipher_mode_of_operation&oldid=1106163325#/media/File:ECB_encryption.svg) (hämtad 2022-09-25).
- [Wik22b] Wikipedia. *Symmetric-key algorithm*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Symmetric-key\\_algorithm&oldid=1106743629](https://en.wikipedia.org/w/index.php?title=Symmetric-key_algorithm&oldid=1106743629) (hämtad 2022-09-25).
- [Wik22c] Wikipedia, the free encyclopedia. *Block cipher mode of operation*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Block\\_cipher\\_mode\\_of\\_operation&oldid=1106163325#/media/File:ECB\\_decryption.svg](https://en.wikipedia.org/w/index.php?title=Block_cipher_mode_of_operation&oldid=1106163325#/media/File:ECB_decryption.svg) (hämtad 2022-09-25).

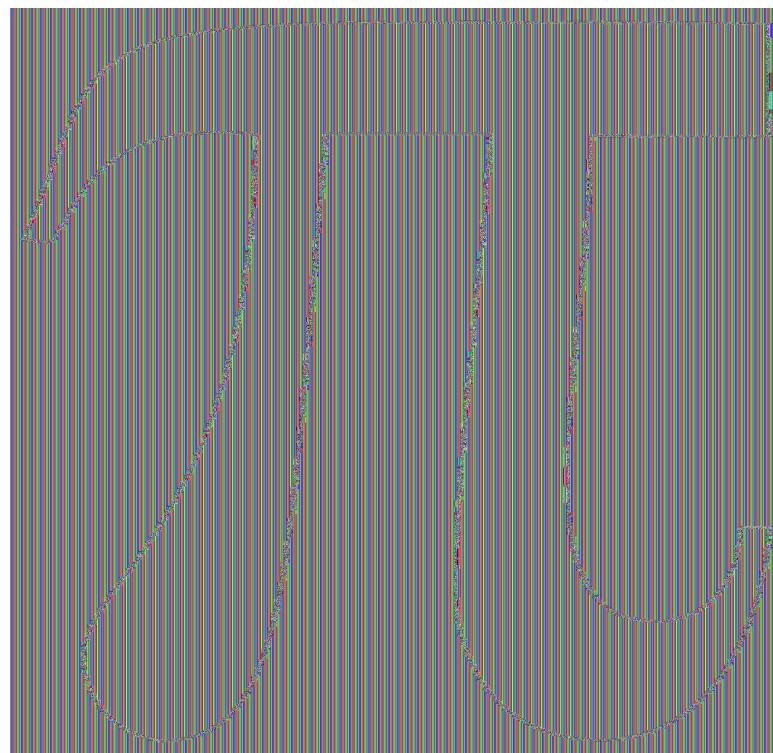
- [Wik22d] Wikipedia, the free encyclopedia. *Block cipher mode of operation*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Block\\_cipher\\_mode\\_of\\_operation&oldid=1106163325#/media/File:CBC\\_encryption.svg](https://en.wikipedia.org/w/index.php?title=Block_cipher_mode_of_operation&oldid=1106163325#/media/File:CBC_encryption.svg) (hämtad 2022-09-26).
- [Wik22e] Wikipedia, the free encyclopedia. *Block cipher mode of operation*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Block\\_cipher\\_mode\\_of\\_operation&oldid=1106163325#/media/File:CBC\\_decryption.svg](https://en.wikipedia.org/w/index.php?title=Block_cipher_mode_of_operation&oldid=1106163325#/media/File:CBC_decryption.svg) (hämtad 2022-09-26).
- [Wik22f] Wikipedia, the free encyclopedia. *Block cipher mode of operation*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Block\\_cipher\\_mode\\_of\\_operation&oldid=1106163325#/media/File:OFB\\_encryption.svg](https://en.wikipedia.org/w/index.php?title=Block_cipher_mode_of_operation&oldid=1106163325#/media/File:OFB_encryption.svg) (hämtad 2022-10-03).
- [Wik22g] Wikipedia, the free encyclopedia. *Block cipher mode of operation*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Block\\_cipher\\_mode\\_of\\_operation&oldid=1106163325#/media/File:OFB\\_decryption.svg](https://en.wikipedia.org/w/index.php?title=Block_cipher_mode_of_operation&oldid=1106163325#/media/File:OFB_decryption.svg) (hämtad 2022-10-03).
- [Wik22h] Wikipedia, the free encyclopedia. *Block cipher*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Block\\_cipher&oldid=1111913955](https://en.wikipedia.org/w/index.php?title=Block_cipher&oldid=1111913955) (hämtad 2022-10-05).
- [Wik22i] Wikipedia, the free encyclopedia. *Frequency analysis*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Frequency\\_analysis&oldid=1113626431](https://en.wikipedia.org/w/index.php?title=Frequency_analysis&oldid=1113626431) (hämtad 2022-10-10).
- [Wik22j] Wikipedia, the free encyclopedia. *Hashfunktion*. 2022. URL: <https://sv.wikipedia.org/w/index.php?title=Hashfunktion&oldid=50669121> (hämtad 2022-10-07).
- [Wik22k] Wikipedia, the free encyclopedia. *Polyalphabetic cipher*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Polyalphabetic\\_cipher&oldid=1113775685](https://en.wikipedia.org/w/index.php?title=Polyalphabetic_cipher&oldid=1113775685) (hämtad 2022-10-10).
- [Wik22l] Wikipedia, the free encyclopedia. *Pseudorandomness*. 2022. URL: <https://en.wikipedia.org/w/index.php?title=Pseudorandomness&oldid=1112429322> (hämtad 2022-10-04).
- [Wik22m] Wikipedia, the free encyclopedia. *RSA*. 2022. URL: <https://sv.wikipedia.org/w/index.php?title=RSA&oldid=50280992> (hämtad 2022-10-04).
- [Wik22n] Wikipedia, the free encyclopedia. *Stream cipher*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Stream\\_cipher&oldid=1098861130](https://en.wikipedia.org/w/index.php?title=Stream_cipher&oldid=1098861130) (hämtad 2022-10-05).
- [Wik22o] Wikipedia, the free encyclopedia. *Substitution cipher*. 2022. URL: [https://en.wikipedia.org/w/index.php?title=Substitution\\_cipher&oldid=1111925704](https://en.wikipedia.org/w/index.php?title=Substitution_cipher&oldid=1111925704) (hämtad 2022-10-10).
- [Wik99] Wikipedia, the free encyclopedia. *Advanced Encryption Standard*. 1999. URL: [https://en.wikipedia.org/wiki/Advanced\\_Encryption\\_Standard#/media/File:AES\\_\(Rijndael\)\\_Round\\_Function.png](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard#/media/File:AES_(Rijndael)_Round_Function.png) (hämtad 2022-09-02).

# Figurer

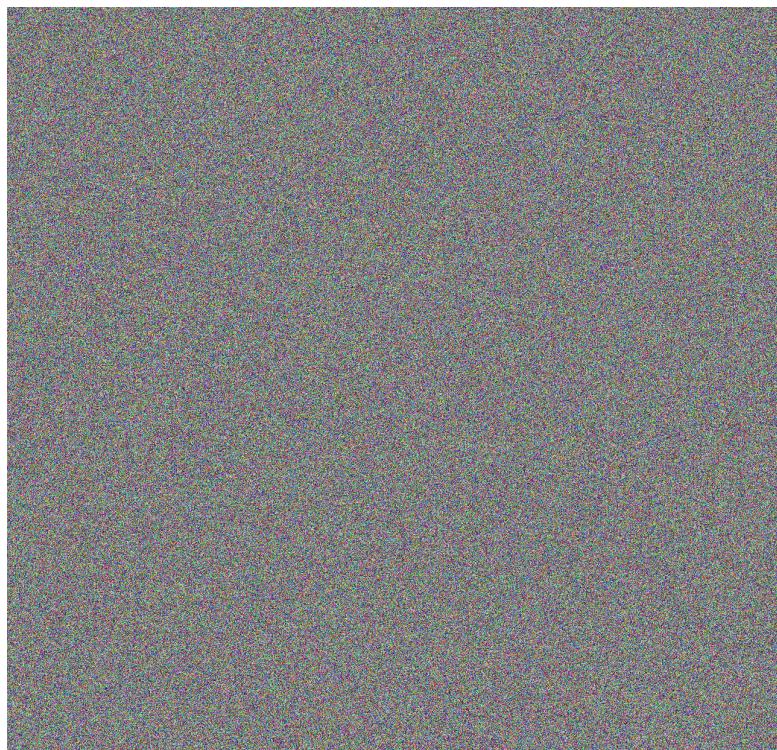
3.1	Electronic Code Book läge kryptering [Wik22b]	10
3.2	Electronic Code Book läge dekryptering [Wik22c]	11
3.3	Cipher Block Chaining läge kryptering [Wik22d]	12
3.4	Cipher Block Chaining läge dekryptering [Wik22e]	12
3.5	Output Feedback läge kryptering [Wik22f]	13
3.6	Output Feedback läge dekryptering [Wik22g]	13
8.1	Orginal bild	22
8.2	Efter ECB Kryptering	22
8.3	Efter CBC Kryptering	23
8.4	Efter OFB Kryptering	23
8.5	Uppställning av vanliga rundor	24



**Figur 8.1:** Orginal bild



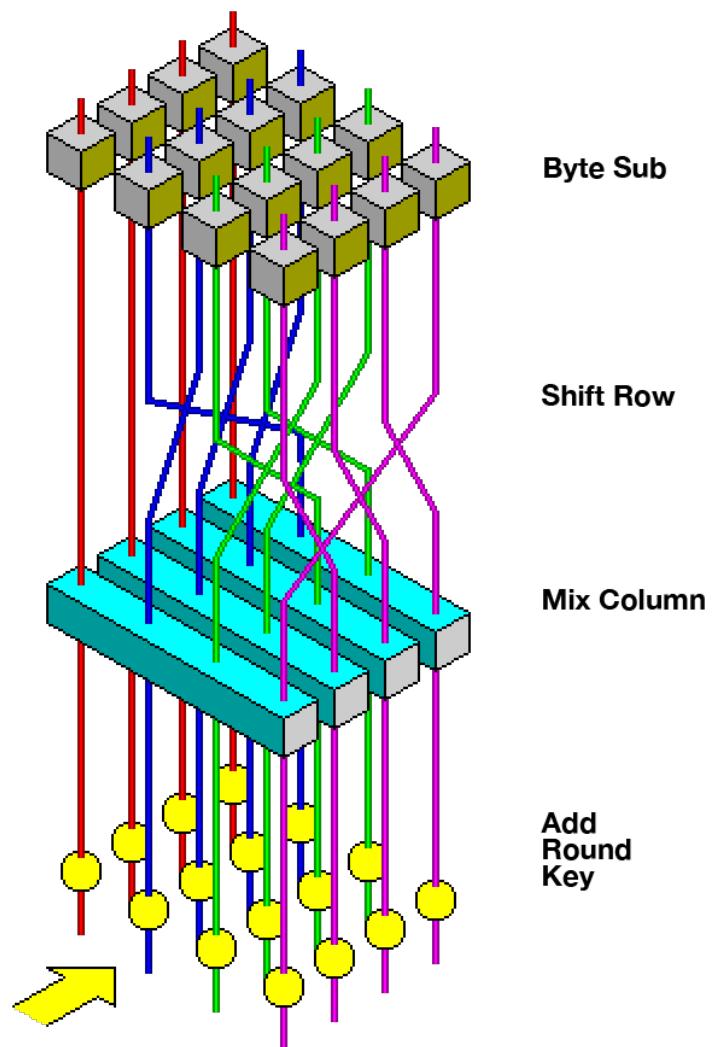
**Figur 8.2:** Efter ECB Kryptering



**Figur 8.3:** Efter CBC Kryptering



**Figur 8.4:** Efter OFB Kryptering



**Figur 8.5:** Uppställning av vanliga rundor

**Källa:** Wikipedia, the free encyclopedia. *Advanced Encryption Standard*. 1999. URL: [https://en.wikipedia.org/wiki/Advanced\\_Encryption\\_Standard#/media/File:AES\\_\(Rijndael\)\\_Round\\_Function.png](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard#/media/File:AES_(Rijndael)_Round_Function.png) (hämtad 2022-09-02)