

# 2. Symmetric Key

Basics

Block or Stream?

Secret Key Methods

Salting

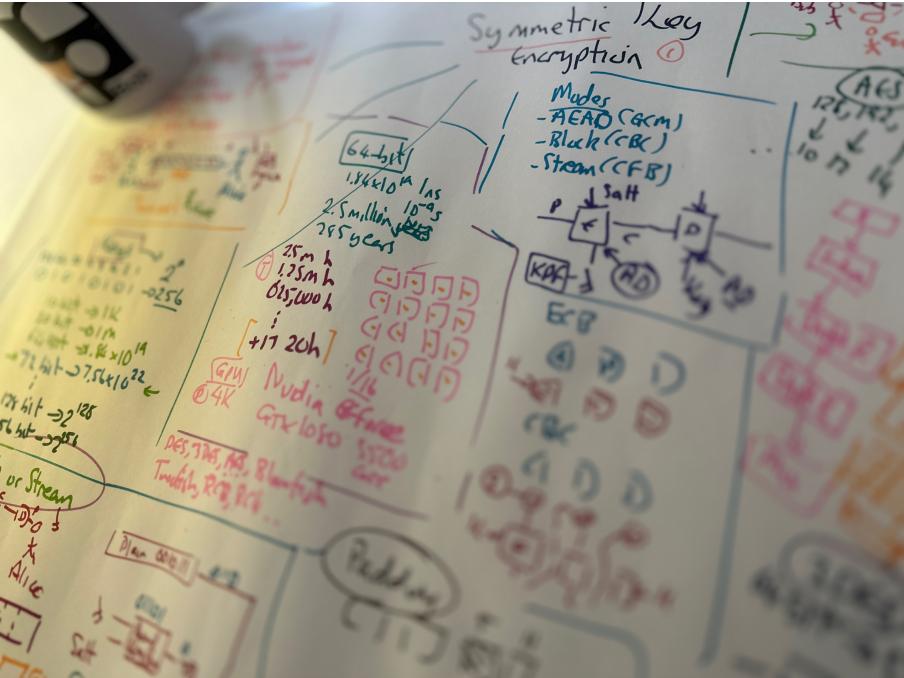
AES: Creation and Modes

3DES

ChaCha20/Poly1305 and RC4

Key Entropy

Cloud-based Symmetric Key



**Prof Bill Buchanan OBE**

<https://asecuritysite.com/symmetric>

# 2. Symmetric Key

Basics

Block or Stream?

Secret Key Methods

Salting

AES: Creation and Modes

3DES

ChaCha20/Poly1305 and RC4

Key Entropy

Cloud-based Symmetric Key

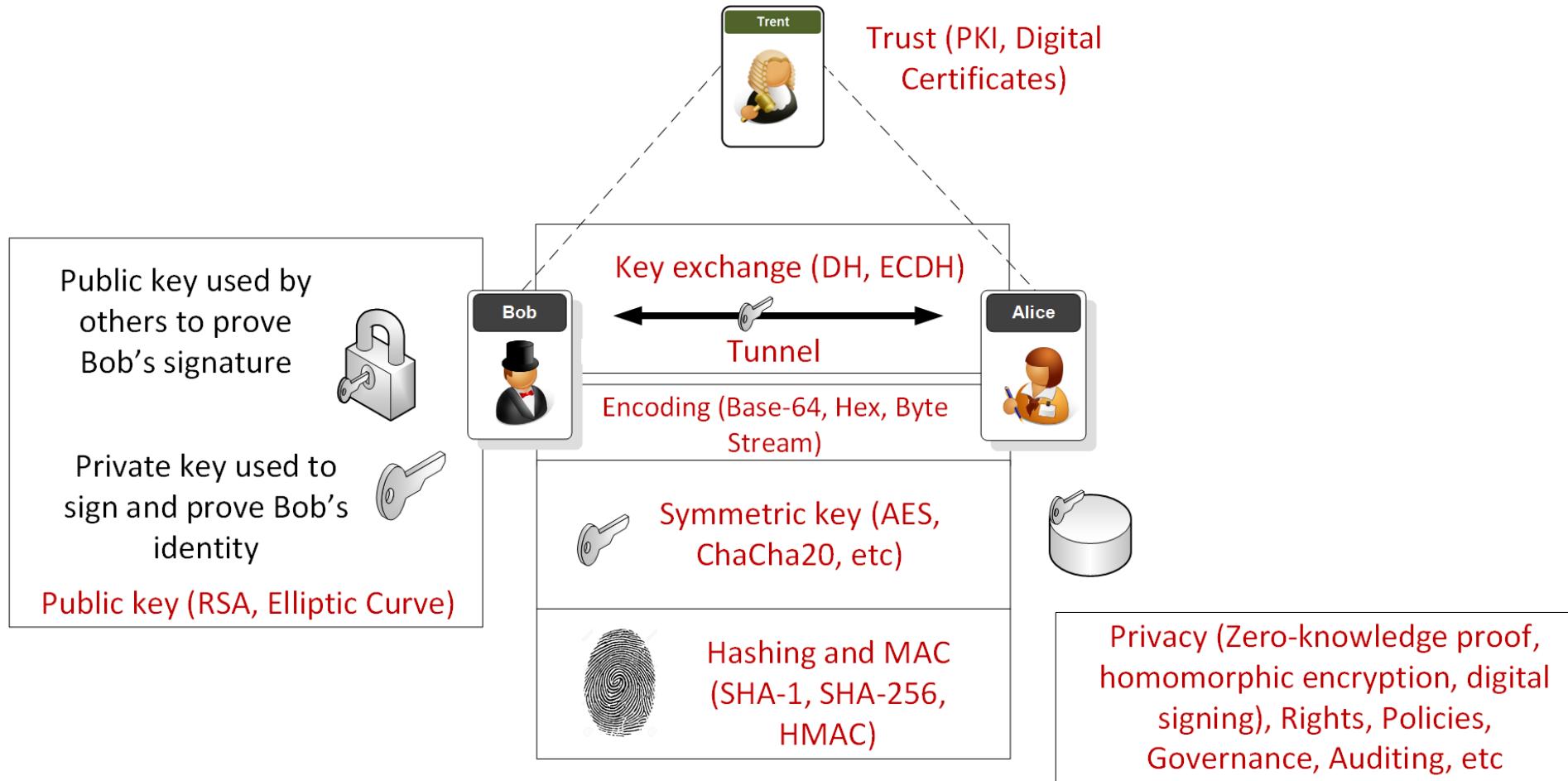


No	Date	Subject	Lab
2	23 Jan 2024	Ciphers and Fundamentals <a href="#">[Unit]</a>	<a href="#">[Lab]</a> <a href="#">[Demo]</a>
3	30 Jan 2024	Symmetric Key <a href="#">[Unit]</a>	<a href="#">[Lab]</a>
4	6 Feb 2024	Hashing and MAC <a href="#">[Unit]</a>	<a href="#">[Lab]</a> Bruce Schneier
5	13 Feb 2024	Asymmetric (Public) Key <a href="#">[Unit]</a>	<a href="#">[Lab]</a>
6	20 Feb 2024	Key Exchange <a href="#">[Unit]</a>	<a href="#">[Lab]</a> Whitfield Diffie
7	27 Feb 2024	Reading Week (Revision lecture)	Mini-project <a href="#">[Here]</a> / Coursework
8	5 Mar 2024	Digital Signatures and Certificates <a href="#">[Unit]</a>	<a href="#">[Lab]</a> Vincent Rijmen
9	12 Mar 2024	Test (Units 1-5) 40% of overall mark <a href="#">[Here]</a>	
10	19 Mar 2024	Tunnelling <a href="#">[Unit]</a>	<a href="#">[Lab]</a> Marty Hellman
11	10 Apr 2024	Blockchain <a href="#">[Unit]</a>	<a href="#">[Lab]</a>
12	16 Apr 2024	Future Cryptography <a href="#">[Unit]</a>	<a href="#">[Lab]</a>
13	23 Apr 2024	Host/Cloud Security <a href="#">[Unit]</a>	<a href="#">[Lab]</a>
14	30 Apr 2024		
15	7 May 2024	Coursework Hand-in - 60% of overall mark (15 May) <a href="#">[Coursework]</a>	

Prof Bill Buchanan OBE

<https://asecuritysite.com/symmetric-key>

# Overview



## Puzzle

For a 128-bit key and brute force, how long would it take to crack a single key for 10 billion tries per second?

- 10 days?
- 100 days?
- 1 year?
- 10 years?
- 100 years?

For a 128-bit key and brute force, how long would it take to crack a single key for 10 billion tries per second?

- ***529 million million years***
  - Boil every ocean on planet, 16,384 times



[Article](#)

# Boiling the planet

security level	volume of water to bring to a boil	bit-lengths		
		symmetric key	cryptographic hash	RSA modulus
teaspoon security	0.0025 liter	35	70	242
shower security	80 liter	50	100	453
pool security	2 500 000 liter	65	130	745
rain security	0.082 km <sup>3</sup>	80	160	1130
lake security	89 km <sup>3</sup>	90	180	1440
sea security	3 750 000 km <sup>3</sup>	105	210	1990
global security	1 400 000 000 km <sup>3</sup>	114	228	2380
solar security	-	140	280	3730

Article

# 2. Symmetric Key

Basics

Block or Stream?

Secret Key Methods

Salting

AES

3DES

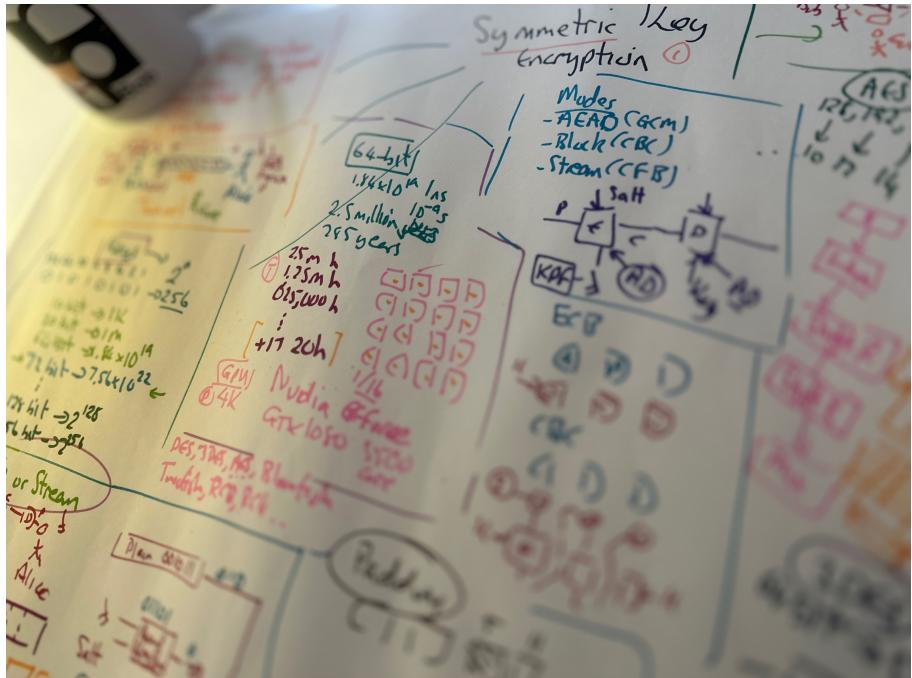
ChaCha20/Poly1305 and RC4

Key Entropy

Cloud-based Symmetric Key

**Prof Bill Buchanan OBE**

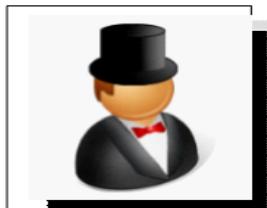
<https://asecuritysite.com/symmetric>





Intruder

Eve



Bob



**Privacy** (Private Key)  
**Identity** (Public Key)  
**Integrity** (Public/Private Key)



Alice



John

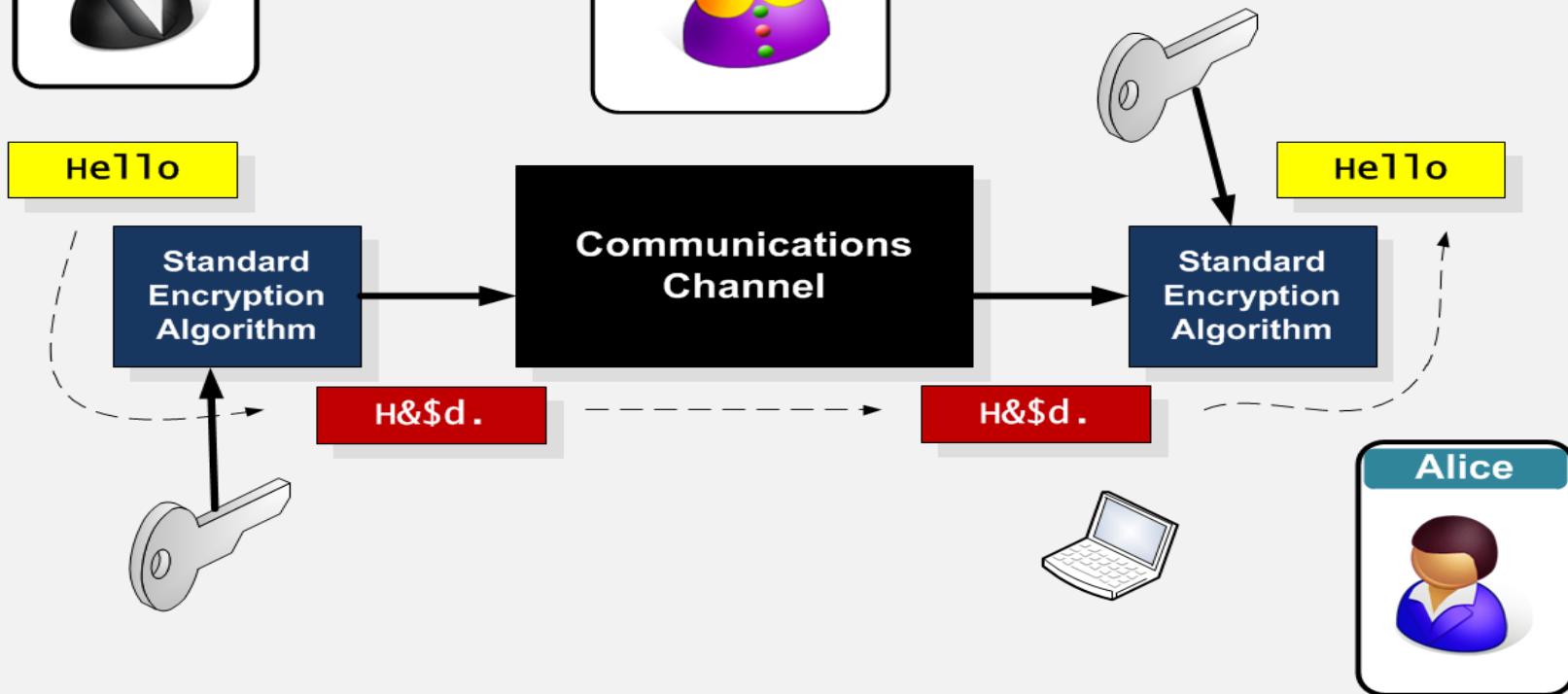


Trent

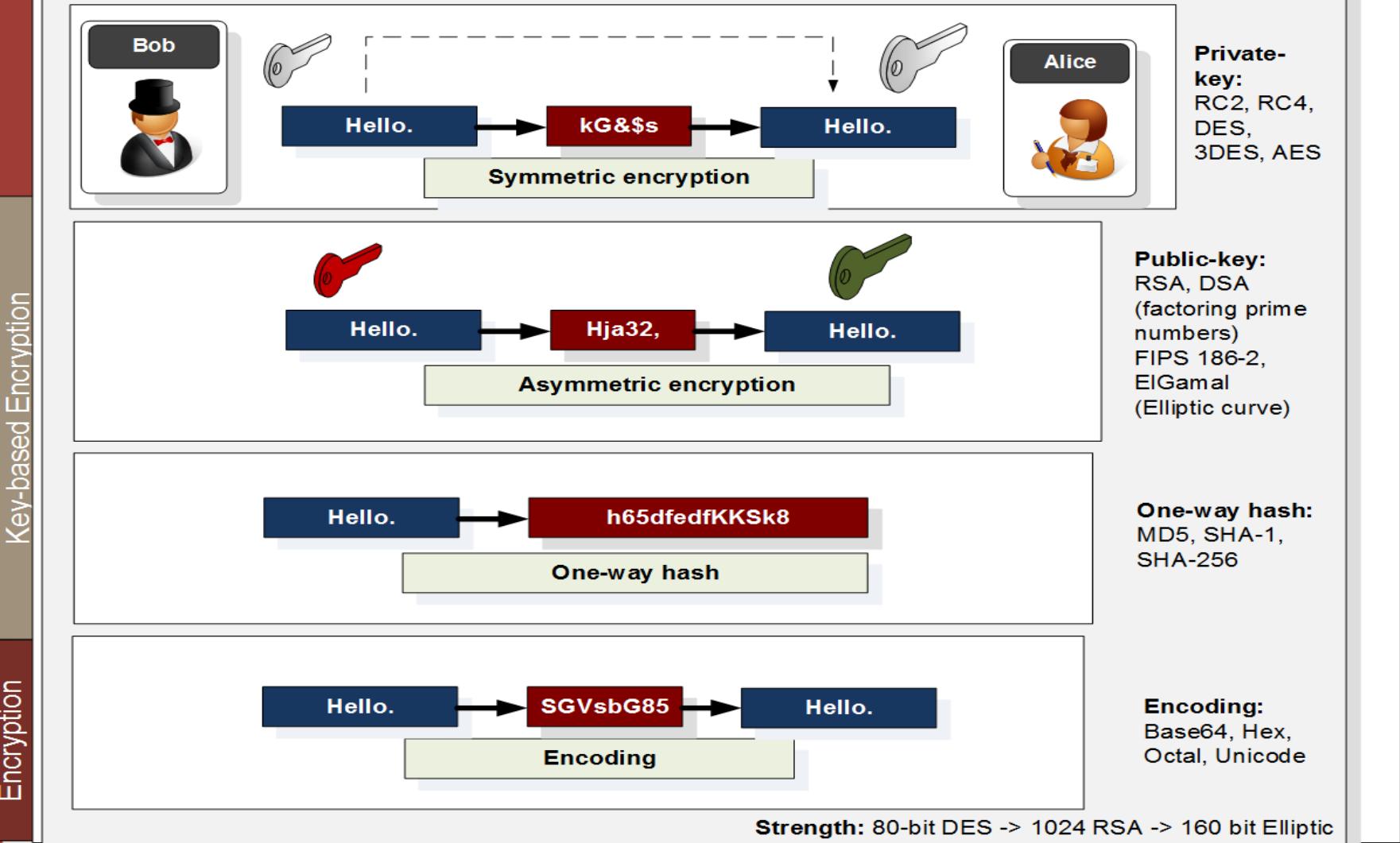
Trusted third party



The major problem is that Eve could gain the encoding algorithm.



## Encryption





## Number of keys

The larger the key, the greater the key space.

Code size	Number of keys	Code size	Number of keys	Code size	Number of keys
1	2	12	4,096	52	$4.5 \times 10^{15}$
2	4	16	65,536	56	$7.21 \times 10^{16}$
3	8	20	1,048,576	60	$1.15 \times 10^{18}$
4	16	24	16,777,216	64	$1.84 \times 10^{19}$
5	32	28	$2.68 \times 10^8$	68	$2.95 \times 10^{20}$
6	64	32	$4.29 \times 10^9$	72	$4.72 \times 10^{21}$
7	128	36	$6.87 \times 10^{10}$	76	$7.56 \times 10^{22}$
8	256	40	$1.1 \times 10^{12}$	80	$1.21 \times 10^{24}$
9	512	44	$1.76 \times 10^{13}$	84	$1.93 \times 10^{25}$
10	1024	48	$2.81 \times 10^{14}$	88	$3.09 \times 10^{26}$





Okay... we select a **64-bit key** ...  
which has  $1.84 \times 10^{19}$  combinations

### Time to crack

- It is important to understand the length of time that a message takes to crack as it may need to be secret for a certain time period.

18.4 million million million different keys  
000000000000...0000000000000000  
To  
111111111111....1111111111111111

How long will it take to crack it by brute-force (on average)?



A 64-bit key has  $1.84 \times 10^{19}$  combinations and it could be cracked by brute-force in  $0.9 \times 10^{19}$  goes.

### Time to crack

- It is important to understand the length of time that a message takes to crack as it may need to be secret for a certain time period.

If we use a fast computer such as 1GHz clock (1ns), and say it takes one clock cycle to test a code, the time to crack the code will be:

9,000,000,000 seconds (150 million minutes)  
... 2.5 million hours (285 years)



If it takes 2.5 million hours (285 years) to crack a code. How many years will it take to crack it within a day?

### Time to crack

- It is important to understand the length of time that a message takes to crack as it may need to be secret for a certain time period.

Computers typically improve their performance every year ... so assume a **doubling** of performance each year.

Date	Hours	Days	Years
2017	2,500,000	104,167	285
2018	1,250,000	52,083	143

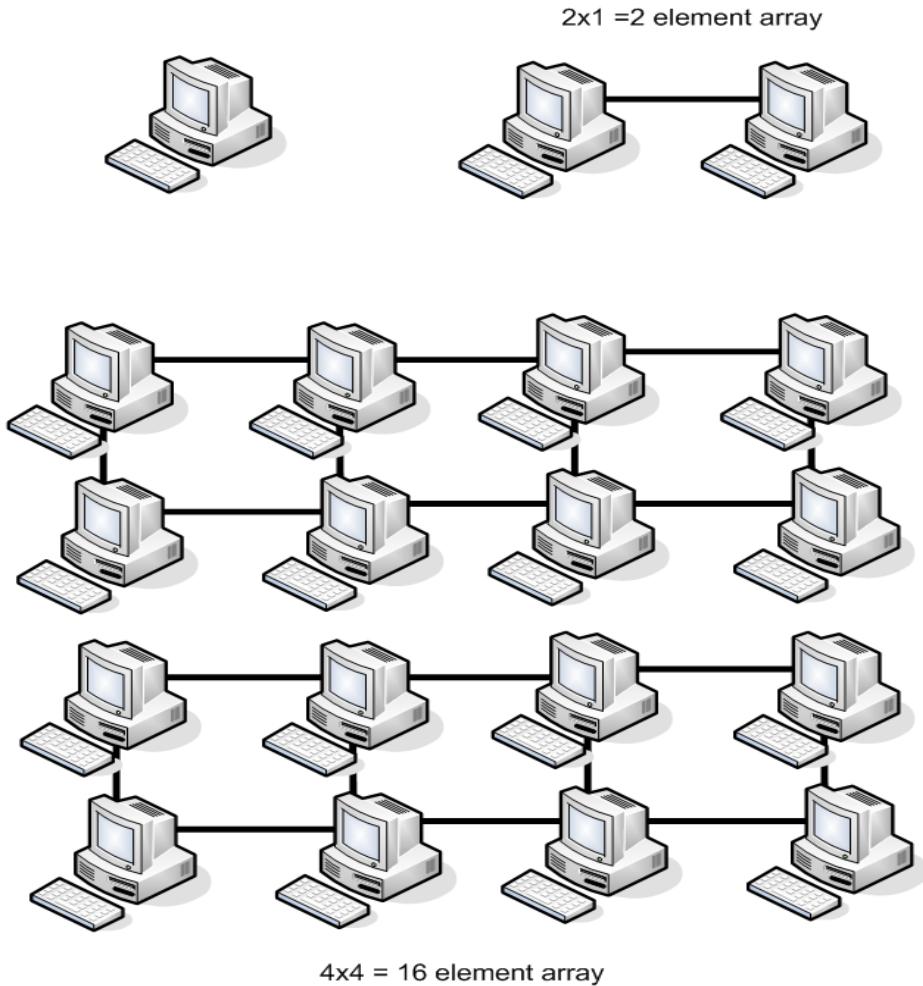


Date	Hours	Days	Years
2017	2,500,000	104,167	285
2018	1,250,000	52,083	143
2019	625,000	26,042	71
2020	312,500	13,021	36
2021	156,250	6,510	18
2022	78,125	3,255	9
2023	39,063	1,628	4
2024	19,532	814	2
+8	9,766	407	1
+9	4,883	203	1
+10	2,442	102	0.3
+11	1,221	51	0.1
+12	611	25	0.1
+13	306	13	0
+14	153	6	0
+15	77	3	0
+16	39	2	0
+17	20	1	0

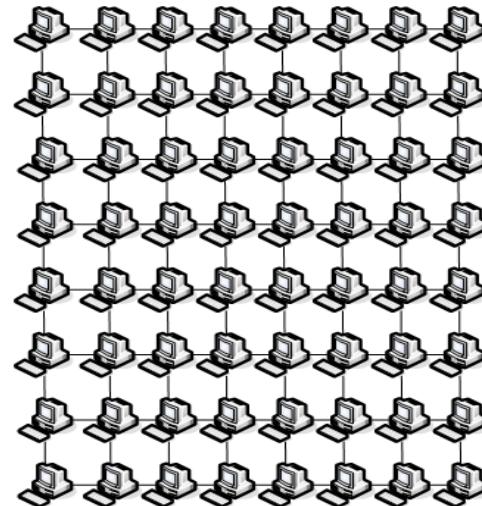
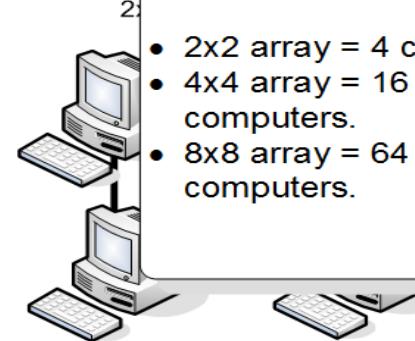
## Time to crack

- From 285 years to 1 day, just by computers increasing their computing power.

56-bit DES:  
Developed  
1975  
30 years ago!  
... now easily  
crackable



## Parallel processing

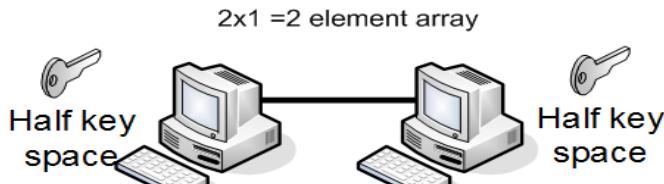


- 2x2 array = 4 computers.
- 4x4 array = 16 computers.
- 8x8 array = 64 computers.

16x16 = 256 element array

## Parallel processing

- 64-bit key --- from **104,000 days** (284 years) to one hour or less.



Brute-force

Encryption

<b>Processors</b>	<b>Year 0</b>	<b>Year 1</b>	<b>Year 2</b>	<b>Year 3</b>	<b>Year 4</b>	<b>Year 5</b>
<b>1</b>	104000 days	52000	26000	13000	6500	3250
<b>4</b>	26000	13000	6500	3250	1625	813
<b>16</b>	6500	3250	1625	813	407	204
<b>64</b>	1625	813	407	204	102	51
<b>256</b>	406	203	102	51	26	13
<b>1024</b>	102	51	26	13	7	4
<b>4096</b>	25	13	7	4	2	1
<b>16,384</b>	152hr	76hr	38hr	19hr	10hr	5hr
<b>65,536</b>	38hr	19hr	10hr	5hr	3hr	2hr
<b>262,144</b>	10hr	5hr	3hr	2hr	1hr	
<b>1,048,576</b>	2hr	1hr				

key  
space

16x16 = 256 element array

4x4 = 16 element array

Author: Prof Bill Buchanan

## Nvidia GeForce GTX 1080 Ti Specifications:

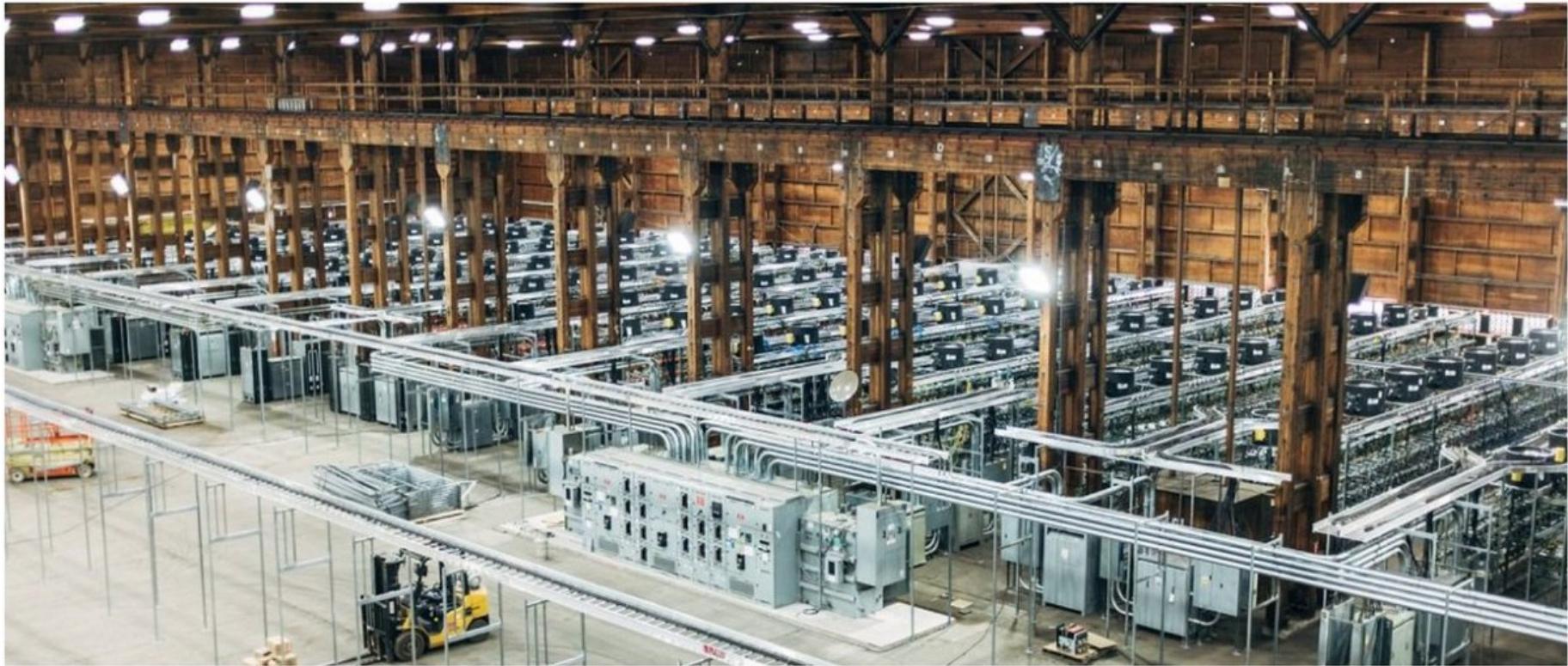
- Graphics Processing Clusters: 6
- Streaming Multiprocessors: 28
- CUDA Cores (single precision): 3584
- Texture Units: 224
- ROP Units: 88
- Base Clock: 1480 MHz
- Boost Clock: 1582 MHz
- Memory Clock: 5505 MHz
- Memory Data Rate: 11 Gbps
- L2 Cache Size: 2816K
- Total Video Memory: 11264MB GDDR5X
- Memory Interface: 352-bit
- Total Memory Bandwidth: 484 GB/s
- Texture Rate (Bilinear): 331.5 GigaTexels/sec
- Fabrication Process: 16 nm
- Transistor Count: 12 Billion
- Connectors: 3 x DisplayPort, 1 x HDMI
- Power Connectors: One 6-pin, One 8-pin
- Recommended Power Supply: 600 Watts



## Nvidia GeForce GTX 1080 Ti Specifications:

- Graphics Processing Clusters: 6
- Streaming Multiprocessors: 28
- CUDA Cores (single precision): 3584
- Texture Units: 224

ROP Units: 88



- Recommended Power Supply: 600 Watts



# Nvidia GeForce GTX 970 Ethereum Mining GPU

Price

**520 USD**

Payback period

**367 days**



**Buy now**

Power	Power cost per day	Return Per Week	Cost per KH/s
145	\$ 0.4176	\$ 9.90	\$ 0.03270
Hash Rate	Return Per Day	Return Per Month	Payback period
15,900.0 KH/s	\$ 1.41	\$ 42.42	367 days
Mines	Profit Ratio	Return Per Year	Annual Return Percentage
Ethereum	338%	\$ 516.12	99%

One of the highest powered GeForce Graphics cards on the market. The GeForce® GTX 970 is a high-performance graphics card designed for serious gaming. Powered by new NVIDIA® Maxwell™ architecture, it features advanced technologies and class-leading graphics for incredible gaming experiences.

You can also mine Ether through a cloud mining contract with Hashflare or Genesis Mining.

**Disclosure:** Mining equipment metrics are calculated based on a network hash rate of **182,293 GH/s** and using a ETH - USD exchange rate of **1 ETH = \$ 1215.26**. These figures vary based on the total network hash rate and on the ETH to USD conversion rate. Equipment cost can vary, block reward is fixed at **3 ETH** and future block reward reductions are not taken into account. The electricity price used in generating these metrics is \$ 0.12 per kWh. Network hash rate varies over time, this is just an estimation based on current values.

# 2. Symmetric Key

Basics

Block or Stream?

Secret Key Methods

Salting

AES

3DES

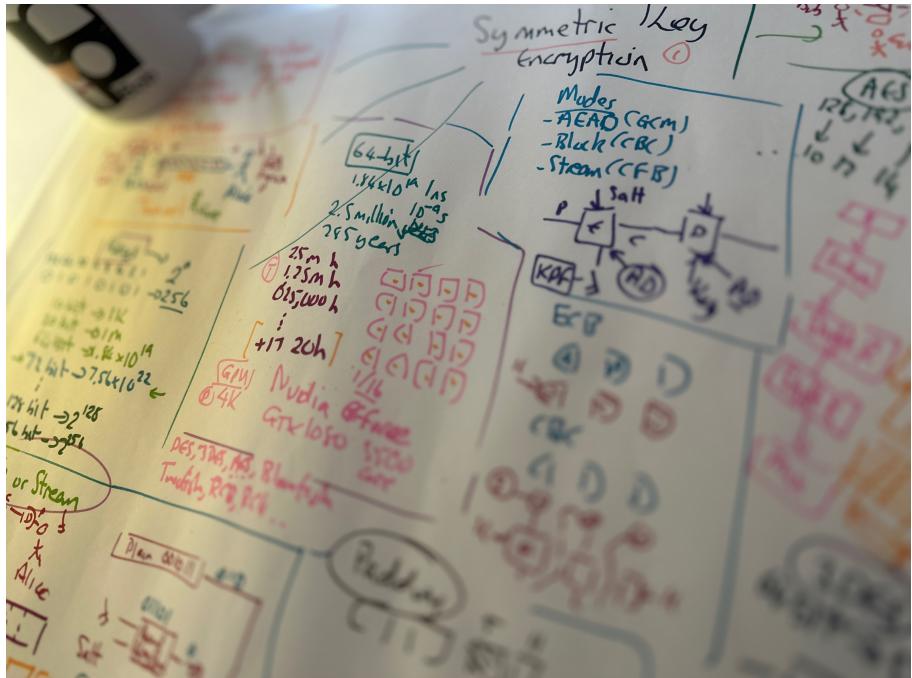
ChaCha20/Poly1305 and RC4

Key Entropy

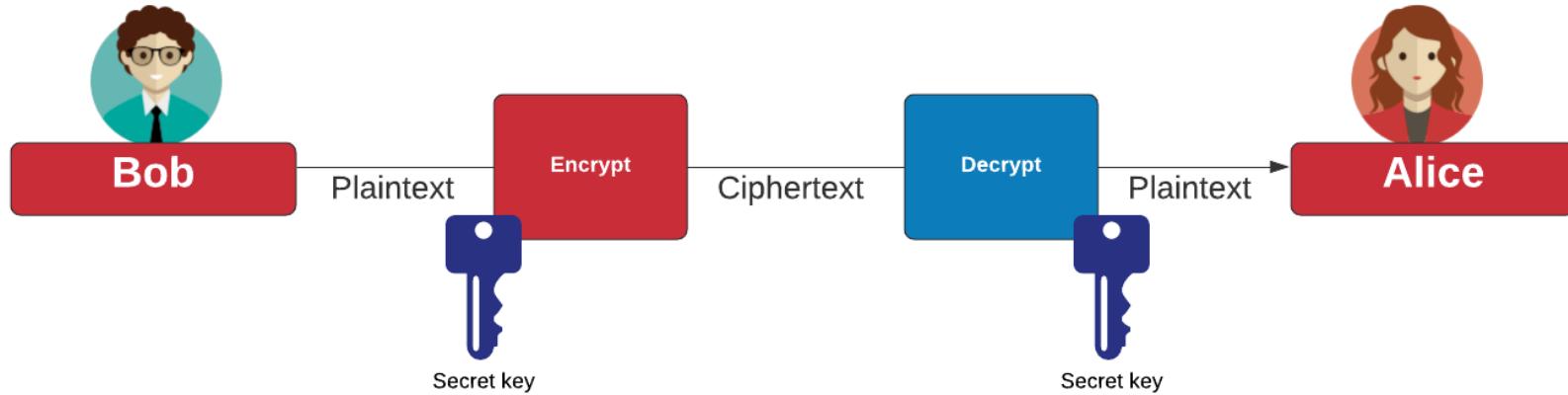
Cloud-based Symmetric Key

Prof Bill Buchanan OBE

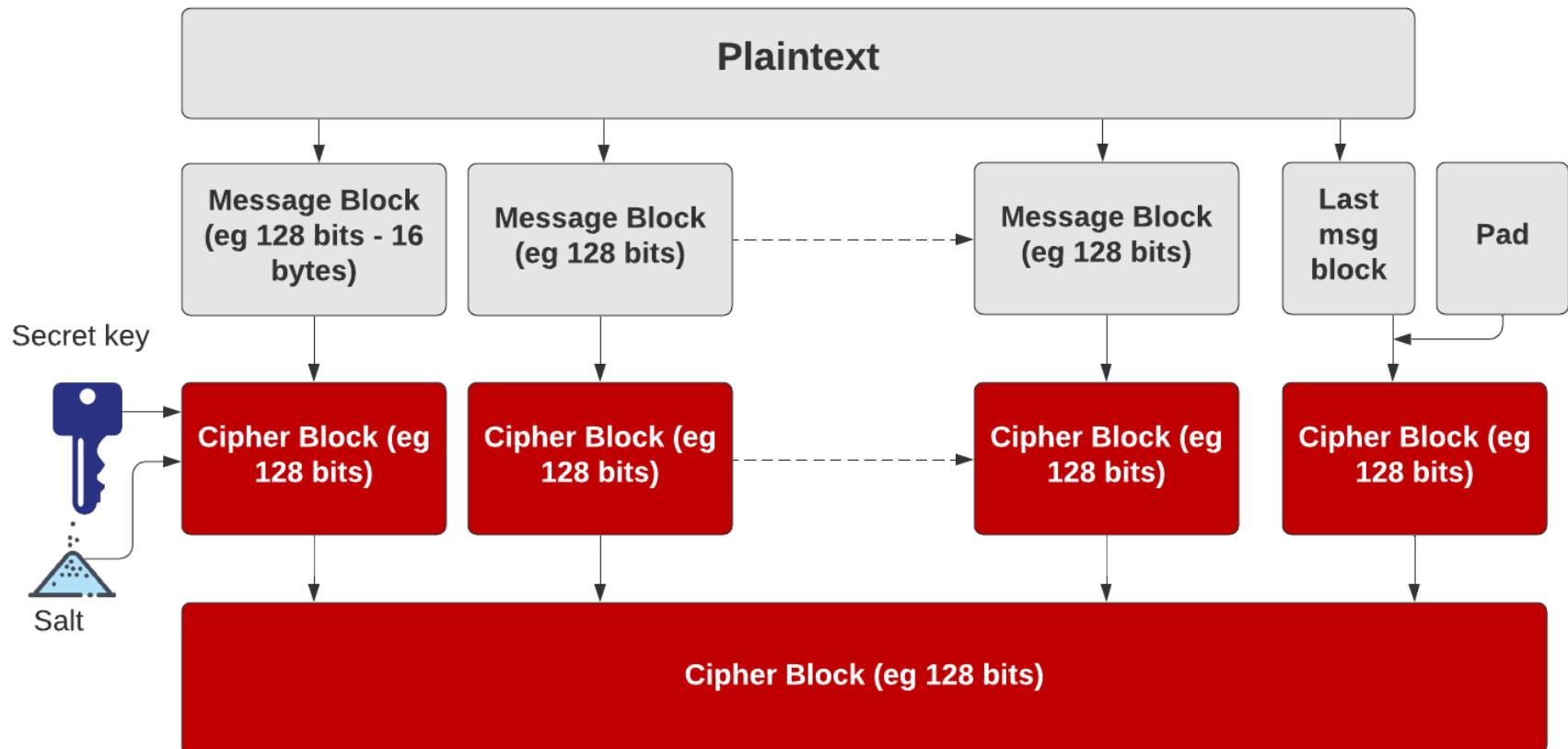
<https://asecuritysite.com/symmetric>



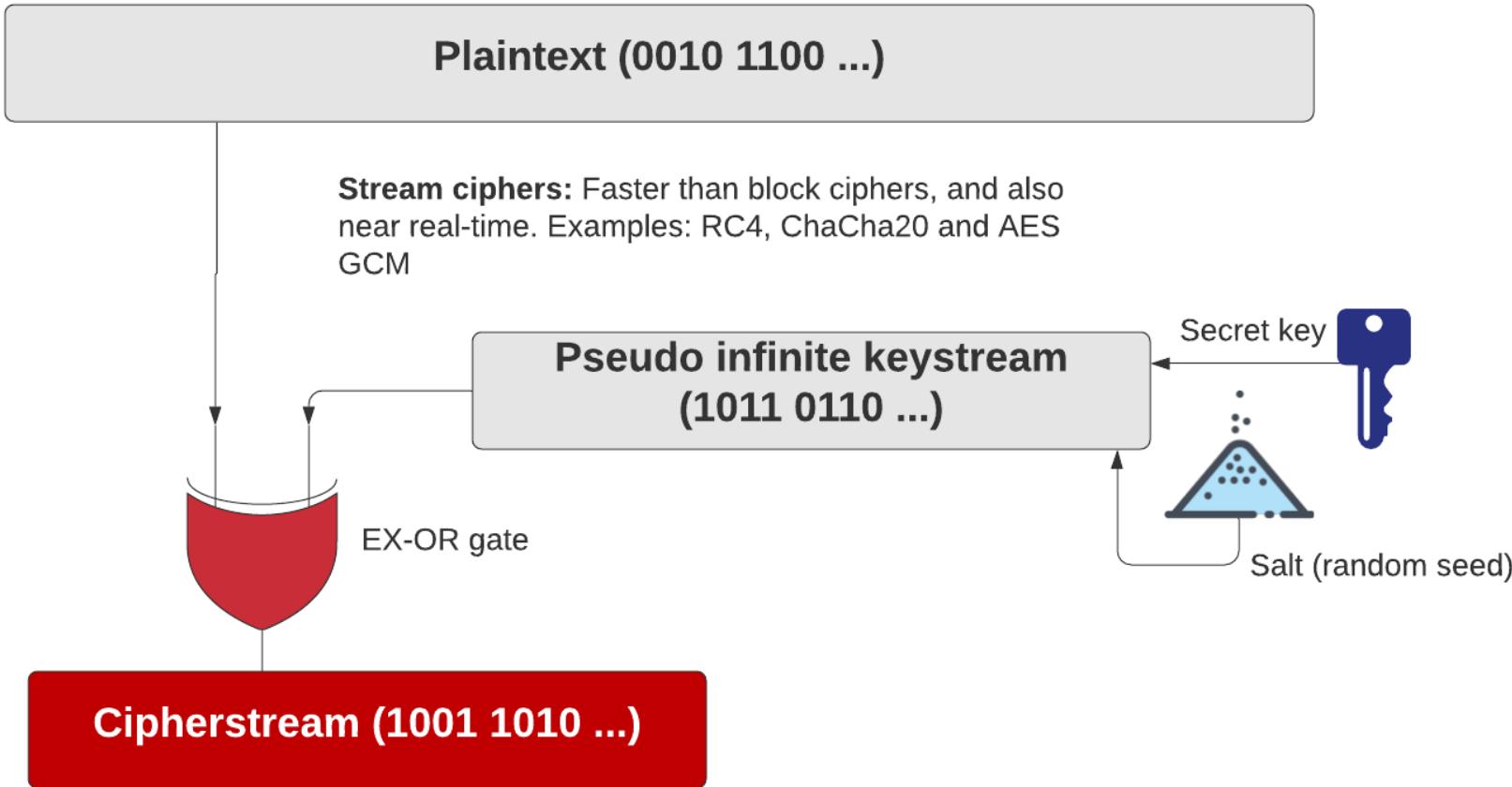
# Symmetric Key



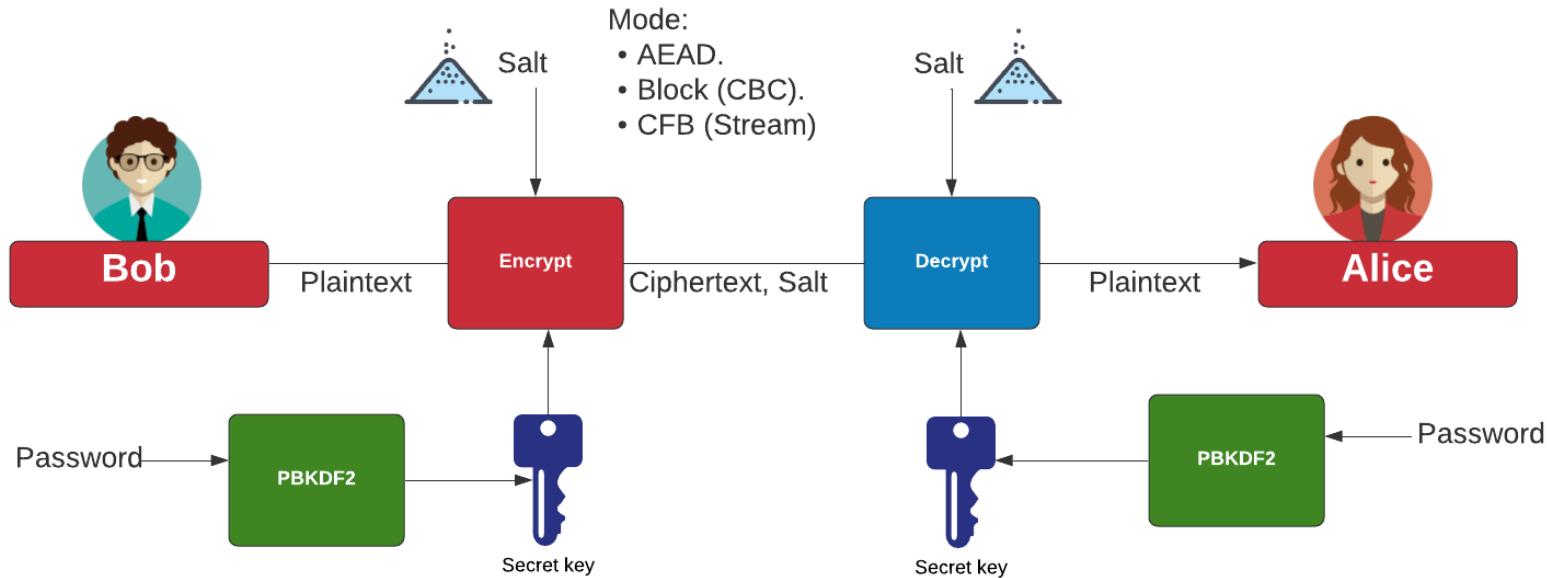
# Symmetric Key – Block cipher



# Symmetric Key – Stream cipher



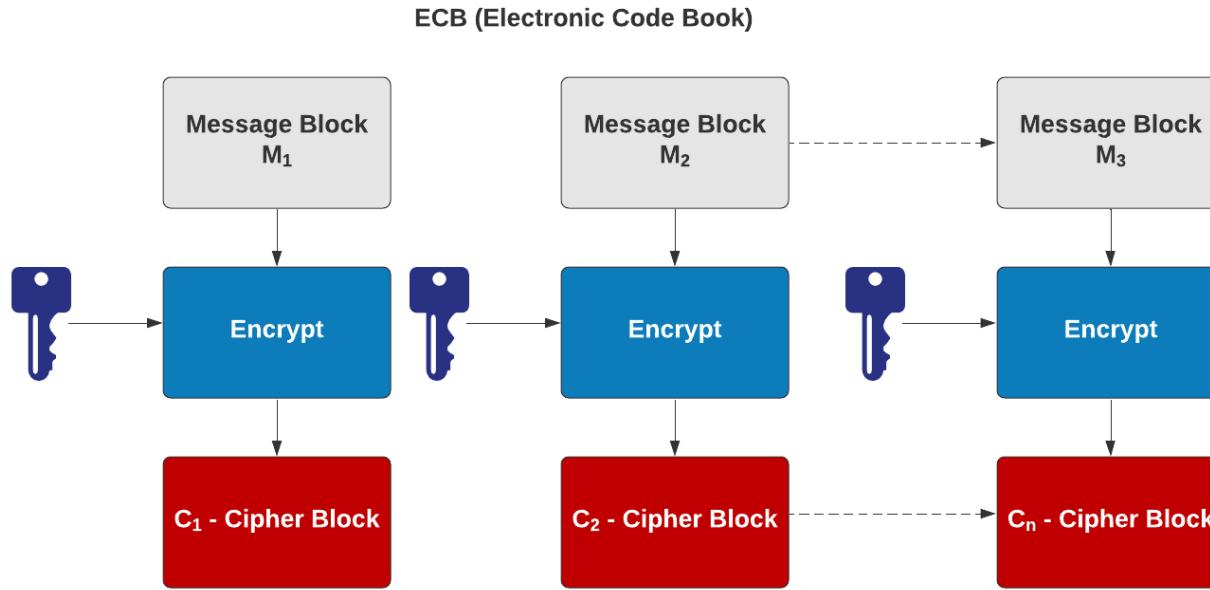
# Symmetric Key - Modes



AES Block cipher modes: ECB (No salt), CBC (Cipher Block Chaining).

AES Stream modes: CFB (Cipher Feedback), OFB (Output Feedback), CTR (Counter), GCM (Galois Counter Mode).

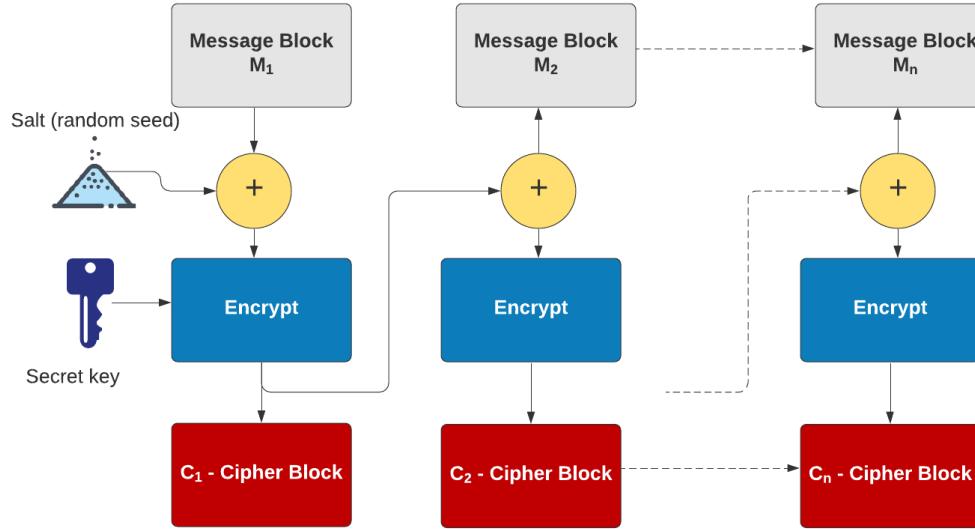
# Symmetric Key – Electronic Code Book (ECB)



AES Block cipher modes: ECB (No salt), CBC (Cipher Block Chaining).

AES Stream modes: CFB (Cipher Feedback), OFB (Output Feedback), CTR (Counter), GCM (Galois Counter Mode).

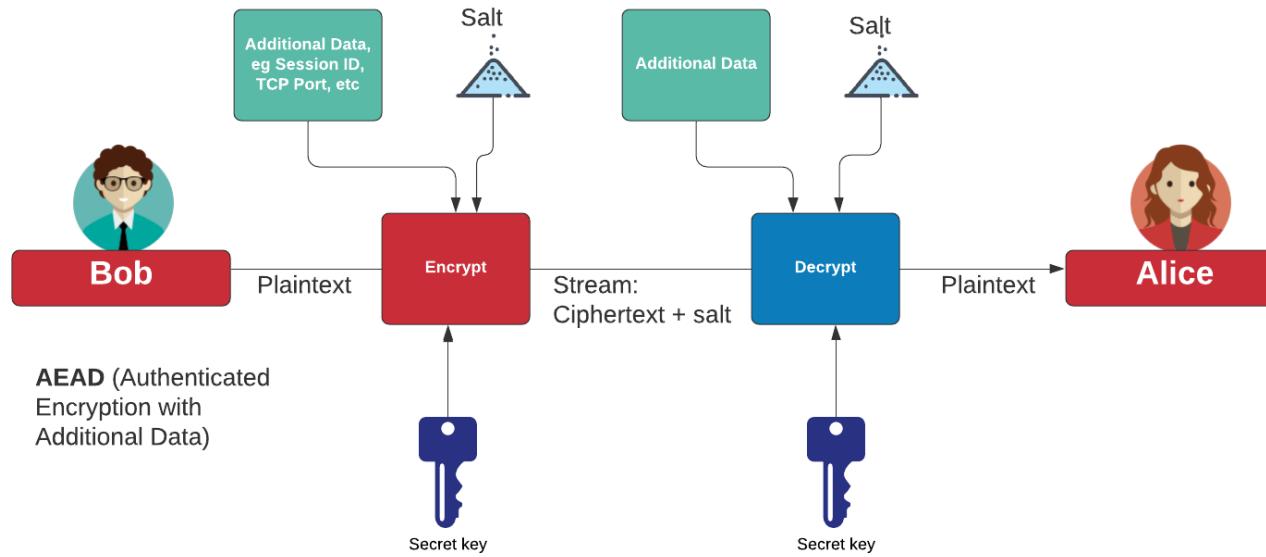
# Symmetric Key – Cipher Block Chaining (CBC)



AES Block cipher modes: ECB (No salt), CBC (Cipher Block Chaining).

AES Stream modes: CFB (Cipher Feedback), OFB (Output Feedback), CTR (Counter), GCM (Galois Counter Mode).

# Authenticated Encryption with Additional Data (AEAD)



AES Block cipher modes: ECB (No salt), CBC (Cipher Block Chaining).

AES Stream modes: CFB (Cipher Feedback), OFB (Output Feedback), CTR (Counter), GCM (Galois Counter Mode).

# Padding

- **CMS** (Cryptographic Message Syntax). This pads with the same value as the number of padding bytes. Defined in RFC 5652, PKCS#5, PKCS#7 and RFC 1423 PEM.
- **Bits**. This pads with 0x80 (10000000) followed by zero (null) bytes. Defined in ANSI X.923 and ISO/IEC 9797-1.
- **ZeroLength**. This pads with zeros except for the last byte which is equal to the number (length) of padding bytes.
- **Null**. This pads will NULL bytes. This is only used with ASCII text.
- **Space**. This pads with spaces. This is only used with ASCII text.
- **Random**. This pads with random bytes with the last byte defined by the number of padding bytes.

# Padding

- After padding (CMS): 68656c6c6f0b0b0b0b0b0b0b0b0b0b0b  
Cipher (ECB): 0a7ec77951291795bac6690c9e7f4c0d
- After padding (Bit): 68656c6c6f80000000000000000000000000000000  
Cipher (ECB): 731abffc2e3b2c2b5caa9ca2339344f9
- After padding (ZeroLen): 68656c6c6f00000000000000000000000000000000a  
Cipher (ECB): d28e2f7e8e44e068732b292bde444245
- After padding (Null): 68656c6c6f00000000000000000000000000000000  
Cipher (ECB): 444797422460453d95856eb2a1520ece
- After padding (Random): 68656c6c6fffc6ecfd884a38798d62a**0a**  
Cipher (ECB): c2c88b4364d2c2dc6f2cac9ab73c995d

[https://asecuritysite.com/symmetric/go\\_pad](https://asecuritysite.com/symmetric/go_pad)



## 2. Symmetric Key

## Basics

# Block or Stream?

# Secret Key Methods

# Salting

AES

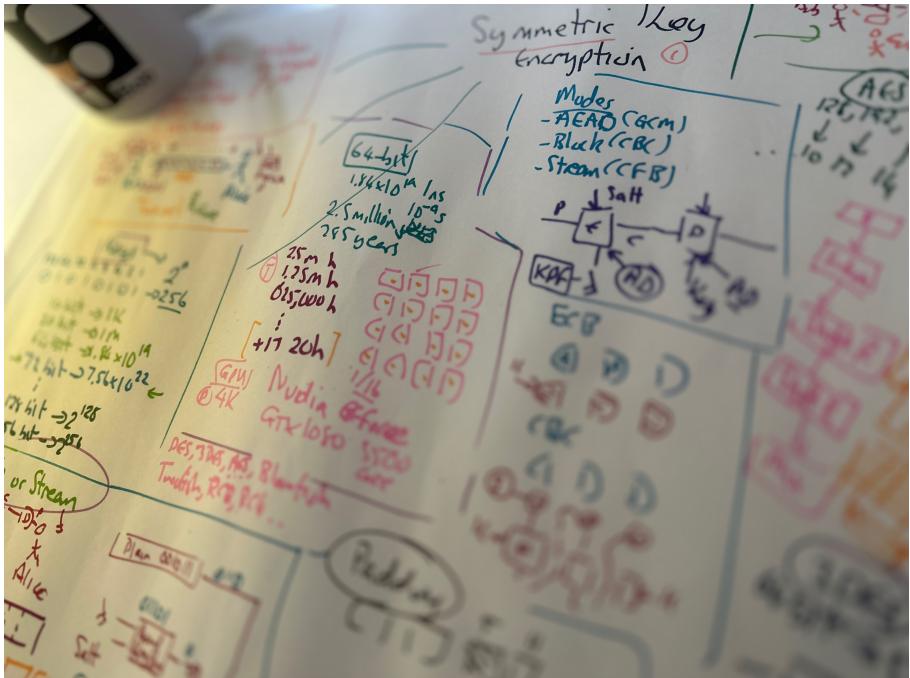
# 3DES

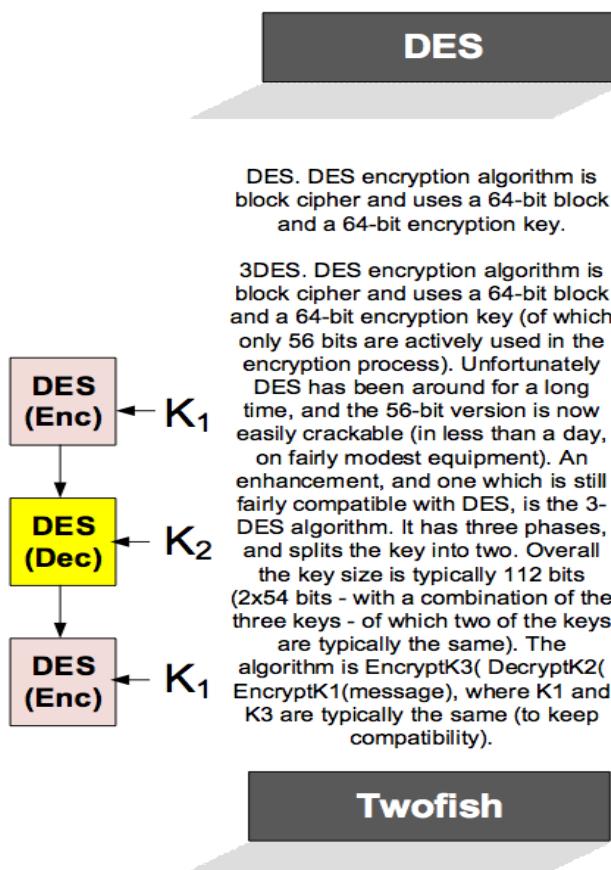
# ChaCha20/Poly1305

# Key Entropy

# **Prof Bill Buchanan OBE**

**<https://asecuritysite.com/encryption>**





Bruce Schneier created Twofish with a general-purpose private key block cipher encryption algorithm.

## DES

**DES.** DES encryption algorithm is block cipher and uses a 64-bit block and a 64-bit encryption key.

**3DES.** DES encryption algorithm is block cipher and uses a 64-bit block and a 64-bit encryption key (of which only 56 bits are actively used in the encryption process). Unfortunately DES has been around for a long time, and the 56-bit version is now easily crackable (in less than a day, on fairly modest equipment). An enhancement, and one which is still fairly compatible with DES, is the 3-DES algorithm. It has three phases, and splits the key into two. Overall the key size is typically 112 bits ( $2 \times 54$  bits - with a combination of the three keys - of which two of the keys are typically the same). The algorithm is  $\text{Encrypt}_K_3(\text{Decrypt}_{K_2}(\text{Encrypt}_{K_1}(\text{message}))$ , where  $K_1$  and  $K_3$  are typically the same (to keep compatibility).

## AES

**AES.** AES (or Rijndael) is a new block cipher, and is the new replacement for DES, and uses 128-bit blocks with 128, 192 and 256 bit encryption keys. It was selected by NIST in 2001 (after a five year standardisation process). The name Rijndael comes from its Belgium creators: Joan Daemen and Vincent Rijmen.

## Blowfish

**Blowfish.** Bruce Schneier created Blowfish with a general-purpose private key block cipher encryption algorithm. Blowfish (with CBC). Blowfishcbc. With CBC we split the message into blocks and encrypt each block. The input from the first stage is the IV (Initialisation Vector), and the input to the following stages is the output from the previous stage. In this example we will use Blowfish to encrypt, using CBC.

## RC2

**RC2.** RC2 ("Rivest Cipher") is a block cipher, and is seen as a replacement for DES. It was created by Ron Rivest in 1987, and is a 64-bit block code and can have a key size from 40 bits to 128-bits (in increments of 8 bits). The 40-bit key version is seen as weak, as the encryption key is so small, but is favoured by governments for export purposes, as it can be easily cracked. In this case the key is created from a Key and an IV (Initialisation Vector). The key has 12 characters (96 bits), and the IV has 8 characters (64 bits), which go to make the overall key.

## Others

- **Skipjack.** Skip jack. Skipjack is a block cipher, using private-key encryption algorithm, and designed by NSA.
- **Camellia.** Camillia is a block cipher created by Mitsubishi and NTT.
- **RC4.** RC4 is a stream cipher used in WEP (in wireless encryption).
- **Affine.** Affine is a stream cipher which uses an equation to encrypt.

**3-DES.** The DES encryption algorithm uses a **64-bit block** and a 64-bit encryption key (of which only **56 bits** are actively used in the encryption process). Unfortunately DES has been around for a long time, and the 56-bit version is now easily crackable (in less than a day, on fairly modest equipment). An enhancement, and one which is still fairly compatible with DES, is the 3-DES algorithm. It has three phases, and splits the key into two. Overall the key size is typically **112 bits** (2x54 bits - with a combination of the three keys - of which two of the keys are typically the same). The algorithm is:

$\text{Encrypt}_{K_3}(\text{Decrypt}_{K_2}(\text{Encrypt}_{K_1}(\text{message})))$

<http://asecuritysite.com/encryption/threedes>

where K1 and K3 are typically the same (to keep compatibility).



**RC-2.** RC2 ("Rivest Cipher") is seen as a replacement for DES. It was created by Ron Rivest in 1987, and is a **64-bit block code** and can have a key size from 40 bits to 128-bits (in increments of 8 bits). The 40-bit key version is seen as weak, as the encryption key is so small, but is favoured by governments for export purposes, as it can be easily cracked. In this case the key is created from a Key and an IV (Initialisation Vector). The key has 12 characters (96 bits), and the IV has 8 characters (64 bits), which go to make the overall key.

<http://asecuritysite.com/encryption/rc2>



**AES/Rijndael.** AES (or Rijndael) is the new replacement for DES, and uses **128-bit blocks** with 128, 192 and 256 bit encryption keys. It was selected by NIST in 2001 (after a five year standardisation process). The name Rijndael comes from its Belgium creators: Joan Daemen and Vincent Rijmen.

<http://asecuritysite.com/encryption/aes>



	Type	16 bytes (K)	64 bytes (K)	256 bytes (K)	1024 bytes (K)	8196 bytes (K)	16384 bytes (K)	Speed
1	AES-128-ECB	843274.01	3195474.22	6318487.38	7930922.12	8399568.9	8326989.91	1
2	AES-128-CTR	678368.08	2275109.84	5279553.62	7207868.87	7979423.06	8043986.94	0.966
3	AES-128-XTS	619899.27	2228173.29	3994753.2	5486256.37	6130715.31	6191415.3	0.7435
4	AES-128-GCM	605277.84	1783068.76	2638697.4	2892877.82	2965774.34	3064684.74	0.368
5	ChaCha20	249952.56	324520.11	1256085.5	1313651.24	1325465.6	1328841.03	0.1596
6	AES-128-CBC	835963.24	981633.37	1014977.93	1017791.15	1023836.16	1000585.67	0.1202
7	AES-128-CCM	194577.35	508297.61	777401.86	922077.87	969897.18	970402.47	0.1165
8	AES-128-OFB	766322.15	810347.52	830004.31	833980.07	838430.65	829145.09	0.0996
9	AES-128-CFB	613249.97	631815.15	633397.59	634749.86	633498.28	635611.53	0.0763
10	AES-128-SIV	25337.67	91657.55	251881.82	447110.12	579590.85	591571.63	0.071
11	ARIA-128-ECB	145433.91	160134.46	161871.53	163339.26	164381.6	161884.84	0.0194
12	CAMELLIA-128-ECB	151809.27	161130.12	160144.47	162545.32	163203.15	158924.8	0.0191
13	ARIA-128-CBC	148219.16	150627.7	151202.59	152393.39	152715.26	153587.67	0.0184
14	ARIA-128-OFB	145373.8	148079.87	147615.57	151164.25	152974.68	153318.31	0.0184
15	CAMELLIA-128-CBC	129253.17	144212.82	150517.59	152886.94	154768.73	152229.21	0.0183
16	ARIA-128-CFB	148310.75	147904.41	148112.95	153218.05	153820.55	151852.37	0.0182
17	CAMELLIA-128-OFB	148960.9	153933.95	153938.16	155428.23	154364.59	151097.49	0.0181
18	CAMELLIA-128-CFB	140449.91	143168	141132.2	139254.47	140323.5	141568.68	0.017
19	CAMELLIA-128-CTR	135561.18	133207.95	133864.87	135731.03	136098.79	137341.61	0.0165
20	ARIA-128-CTR	143648.3	143381.1	142180.27	139979.09	137287.61	130030.01	0.0156
21	SM4-ECB	88777.61	97180.69	97532.75	98426.2	99046.49	98604.37	0.0118
22	SM4-OFB	88510.91	87200.6	86347.6	88995.84	88831.32	89301.02	0.0107
23	SM4-CFB	85791.93	87920.43	88176.42	88301.34	87864.66	88292.77	0.0106
24	SM4-CTR	88356.77	88737.5	88368.73	86952.33	85805.04	88042.15	0.0106
25	SM4-CBC	86424.09	86264.68	84818.86	88130.44	88243.46	87535.8	0.0105
26	SM4-CBC	86832.73	87983.03	88017.08	87945.16	87834.62	86665.33	0.0104
27	AES-128-CFB8	35902.5	34952.05	34891.61	35012.58	35979.26	35891.88	0.0043
28	DES-EDE-ECB	28031.87	28280.87	28453.16	28339.88	28291.17	28488.43	0.0034
29	DES-EDE3-ECB	28032.97	28282.9	27888.02	27517.52	28420.78	28448.09	0.0034
30	DES-EDE3-CBC	26749.86	27047.49	26902.7	27013.46	27058.18	27124.01	0.0033

31	DES-EDE-CBC	26791.57	26877.63	26609.53	26127.41	27017.22	27091.14	0.0033
32	DES-EDE3-CBC	26739.82	26988.71	27121.36	27037.01	26659.5	26992.5	0.0032
33	DES-EDE3-OFB	26722.09	26930.01	27100.21	27086.85	26411.01	26345.91	0.0032
34	DES-EDE-OFB	26577.84	26820.81	26986.85	27026.43	26614.41	26176.04	0.0031
35	DES-EDE3-CFB	25305.46	24802.01	25782.45	25680.9	25646.42	25765.07	0.0031
36	DES-EDE-CFB	25371.22	24892.85	25769.53	25690.11	25657.34	25743.15	0.0031
37	ARIA-128-CFB8	9171.21	9387.72	9377.79	9465.13	9459.03	9366.19	0.0011
38	CAMELLIA-128-CFB8	8713.77	8771.57	8795.31	8661.67	8573.02	8650.75	0.001
39	AES-128-CFB1	3840.59	3846.53	3857.04	3752.62	3739.83	3822.93	0.0005
40	ARIA-128-CFB1	1128.92	1142.24	1138.18	1144.55	1125.03	1104.97	0.0001
41	CAMELLIA-128-CFB1	1055.47	1037.8	1026.14	1016.12	1026.76	1057.01	0.0001



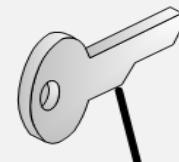


The major problem is that Eve could gain the encoding algorithm.

Hello

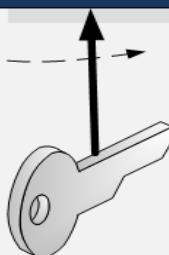
Standard  
Encryption  
Algorithm

Communications  
Channel



Hello

Standard  
Encryption  
Algorithm



H&\$d.

H&\$d.



Alice

## 2. Symmetric Key

## Basics

# Block or Stream?

# Secret Key Methods

# Salting

AES

## 3DES

# ChaCha20/Poly1305

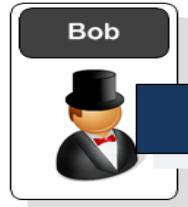
# Key Entropy

# Cloud-based Symmetric Key

# **Prof Bill Buchanan OBE**

**<https://asecuritysite.com/symmetric>**





Hello. How are you?



kG&\$s &FDsaf \*fd\$

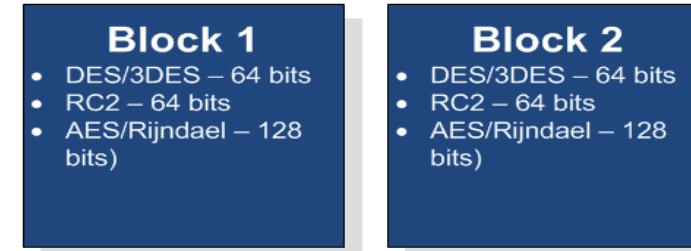


kG&\$s &FDsaf \*fd\$



The solution is to add **salt** to the encryption key, as that it changes its operation from block-to-block (for block encryption) or data frame-to-data frame (for stream encryption)

A major problem in encryption is playback where an intruder can copy an encrypted message and play it back, as the same plain text will always give the same cipher text.

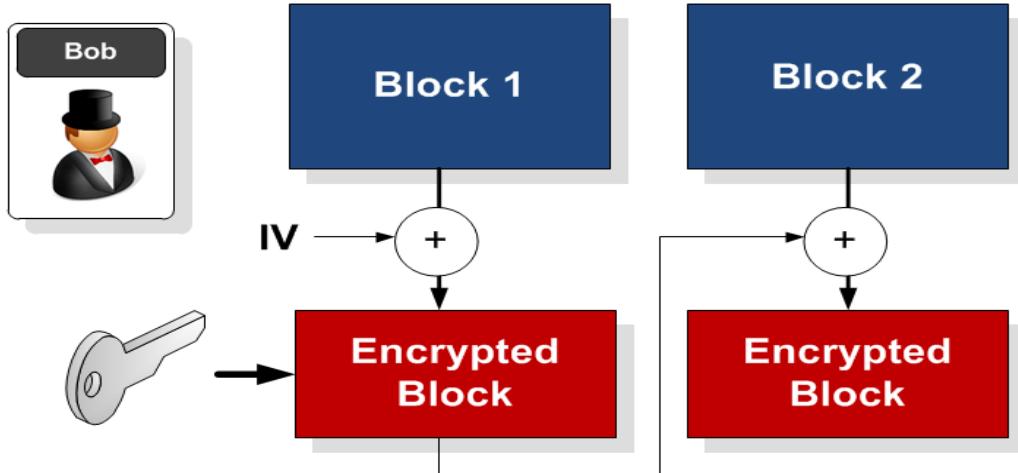


**Electronic Code Book (ECB)** method. This is weak, as the same cipher text appears for the same blocks.

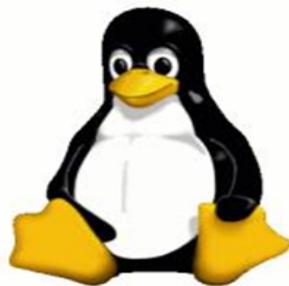
Hello → 5ghd%43f=

Hello → 5ghd%43f=

**Adding salt.** This is typically done with an IV (Initialisation Vector) which must be the same on both sides. In WEP, the IV is incremented for each data frame, so that the cipher text changes.



**Cipher Block Chaining (CBC).** This method uses the IV for the first block, and then the results from the previous block to encrypt the current block.



Original image



Image with AES using ECB

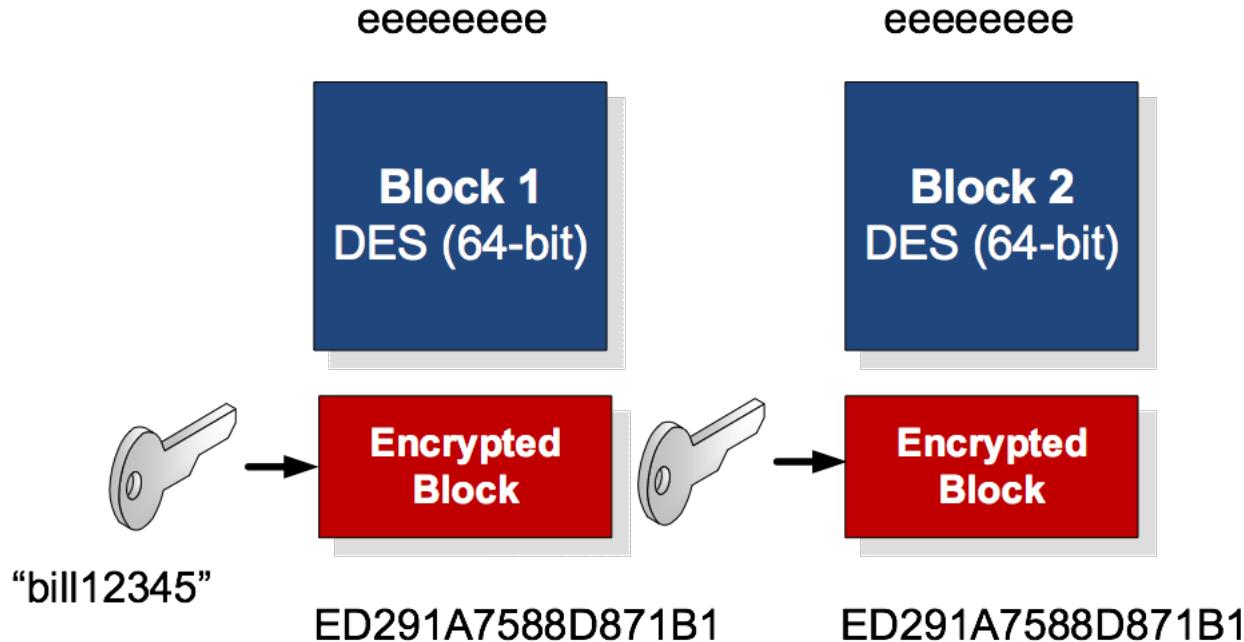


Image with AES using CBC

Image ref: [http://en.wikipedia.org/wiki/Block\\_cipher\\_modes\\_of\\_operation](http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation)

eeeeeeeeeeeeeeeeeeeeeeeeeeee  
eeeeeeeeeeeeeeeeeeeeeeee

[eeeeeeee] [eeeeeeee] [eeeeeeee][eeeeeeee] [eeeeeeee] [eeeeeeee][eeeeee <PADDING>]



ED291A7588D871B1ED291A7588D871B1ED291A7588D  
871B1ED291A7588D871B1ED291A7588D871B1ED291A  
7588D871B18D6DF6795DDEDACD



# Cipher Cracking



## Hazmat Symmetric Key - Bit-flipping Ciphertext

[AES Home][Home]

Some AES modes are vulnerable to bit flipping attacks. In this case, we will use AES CTR mode, and flip the bits of the nth character in the ciphertext, and where we specify the character to target (by default, the program will focus on changing the 9th character).



AES

aka Rijndael  
@asecuritysite.com

Data:

Pay Bob 1 Dollar

Character to flip:

1st

Determine

Message: Pay Bob 1 Dollar

Key: b'f8cc5755683defe0ff480d74f39b878933e9383e4234c60b9607bbedaf980ecd'

IV: b'54a49cb334426485d606667566a362e8'

==== AES CTR ===

Flipping two least significant bits in character 9

Before bitflip: ccacd42f76cf3cfcc039ac5b229c512a

After bitflip: ccacd42f76cf3cfcc339ac5b229c512a

Cipher: b'ccacd42f76cf3cfcc339ac5b229c512a'

Decrypted: Pay Bob 2 Dollar

# 2. Symmetric Key

Basics

Block or Stream?

Secret Key Methods

Salting

AES

3DES

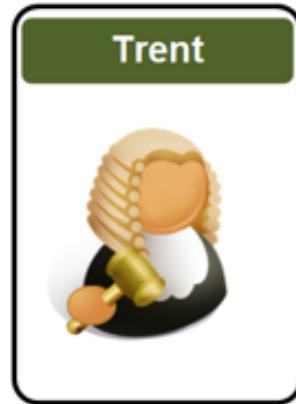
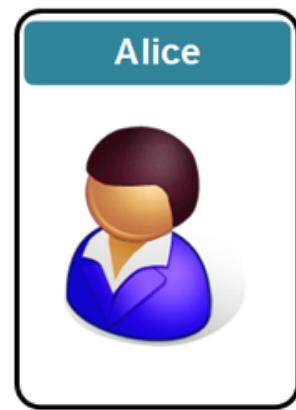
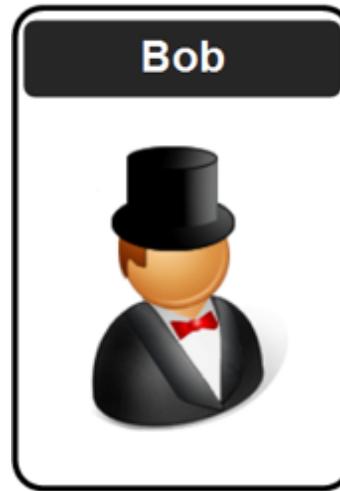
ChaCha20/Poly1305 and RC4

Key Entropy

Cloud-based Symmetric Key

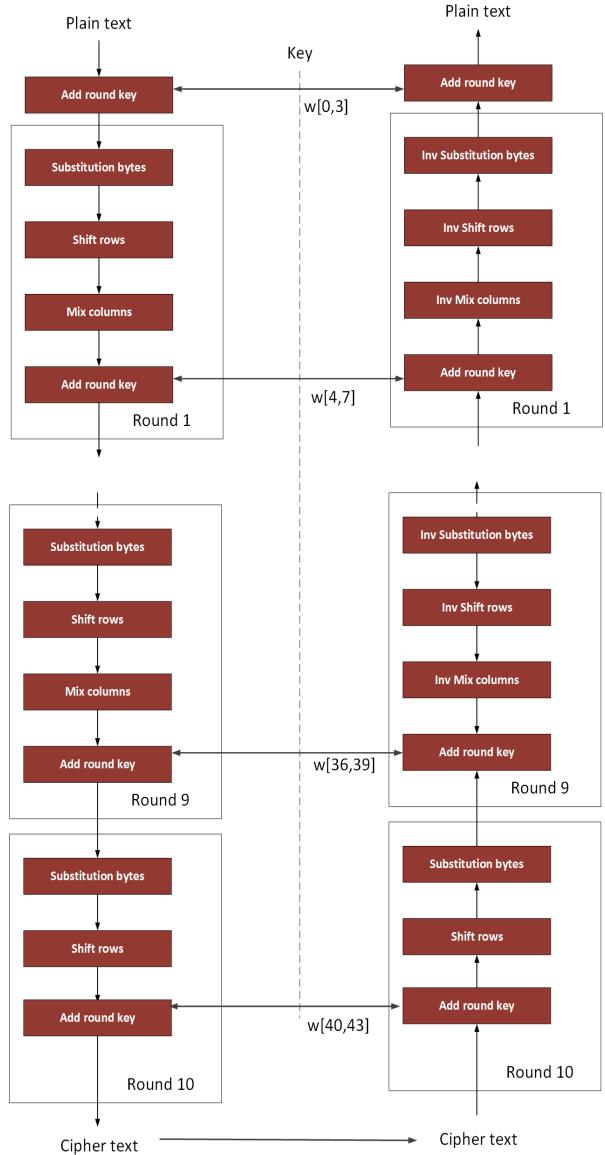
**Prof Bill Buchanan OBE**

<https://asecuritysite.com/symmetric>



# AES

NIST



	0x	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	0x63	0x7c	0x77	0x7b	0xf2	0x6b	0x6f	0xc5	0x30	0x01	0x67	0x2b	0xfe	0xd7	0xab	0x76
1x	0xca	0x82	0xc9	0x7d	0xfa	0x59	0x47	0xf0	0xad	0xd4	0xa2	0xaf	0x9c	0xa4	0x72	0xc0
2x	0xb7	0xfd	0x93	0x26	0x36	0x3f	0xf7	0xcc	0x34	0xa5	0xe5	0xf1	0x71	0xd8	0x31	0x15
3x	0x04	0xc7	0x23	0xc3	0x18	0x96	0x05	0x9a	0x07	0x12	0x80	0xe2	0xeb	0x27	0xb2	0x75
4x	0x09	0x83	0x2c	0x1a	0x1b	0x6e	0x5a	0xa0	0x52	0x3b	0xd6	0xb3	0x29	0xe3	0x2f	0x84
5x	0x53	0xd1	0x00	0xed	0x20	0xfc	0xb1	0x5b	0x6a	0xcb	0xbe	0x39	0x4a	0x4c	0x58	0xcf
6x	0xd0	0xef	0xaa	0xfb	0x43	0x4d	0x33	0x85	0x45	0xf9	0x02	0x7f	0x50	0x3c	0x9f	0xa8
7x	0x51	0xa3	0x40	0x8f	0x92	0x9d	0x38	0xf5	0xbc	0xb6	0xda	0x21	0x10	0xff	0xf3	0xd2
8x	0xcd	0x0c	0x13	0xec	0x5f	0x97	0x44	0x17	0xc4	0xa7	0x7e	0x3d	0x64	0x5d	0x19	0x73
9x	0x60	0x81	0x4f	0xdc	0x22	0x2a	0x90	0x88	0x46	0xee	0xb8	0x14	0xde	0x5e	0x0b	0xdb
Ax	0xe0	0x32	0x3a	0x0a	0x49	0x06	0x24	0x5c	0xc2	0xd3	0xac	0x62	0x91	0x95	0xe4	0x79
Bx	0xe7	0xc8	0x37	0x6d	0x8d	0xd5	0x4e	0xa9	0x6c	0x56	0xf4	0xea	0x65	0x7a	0xae	0x08
Cx	0xba	0x78	0x25	0x2e	0x1c	0xa6	0xb4	0xc6	0x88	0xdd	0x74	0x1f	0x4b	0xbd	0x8b	0x8a
Dx	0x70	0x3e	0xb5	0x66	0x48	0x03	0xf6	0x0e	0x61	0x35	0x57	0xb9	0x86	0xc1	0x1d	0x9e
Ex	0x1e	0xf8	0x98	0x11	0x69	0xd9	0x8e	0x94	0x9b	0x1e	0x87	0x9e	0xce	0x55	0x28	0xdf
Fx	0x8c	0xa1	0x89	0x0d	0xbf	0xe6	0x42	0x68	0x41	0x99	0x2d	0x0f	0xb0	0x54	0xbb	0x16

	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
0x	0x52	0x09	0x6a	0xd5	0x30	0x36	0xa5	0x38	0xbf	0x40	0xa3	0x9e	0x81	0xf3	0xd7	0xfb
1x	0x7c	0x03	0x39	0x82	0x9b	0x2f	0xff	0x87	0x34	0x8e	0x43	0x44	0xc4	0xde	0x9	0xcb
2x	0x54	0x7b	0x94	0x32	0xa6	0xc2	0x23	0x3d	0xee	0x4c	0x95	0x0b	0x42	0xfa	0xc3	0x4e
3x	0x08	0x2e	0xa1	0x66	0x28	0xd9	0x24	0xb2	0x76	0x5b	0xa2	0x49	0x6d	0xb8	0xd1	0x25
4x	0x72	0xf8	0x0f	0x64	0x86	0x68	0x98	0x16	0xd4	0xa4	0x5c	0xcc	0x5d	0x65	0xb6	0x92
5x	0x6c	0x70	0x48	0x50	0xfd	0xed	0xb9	0xda	0x5e	0x15	0x46	0x57	0xa7	0x8d	0x9d	0x84
6x	0x90	0xd8	0xab	0x00	0x8c	0xbc	0xd3	0xa0	0xf7	0xe4	0x58	0x05	0xb8	0x3b	0x45	0x06
7x	0xd0	0x2c	0x1e	0x8f	0xca	0x3f	0x0f	0x02	0xc1	0xaf	0xbd	0x03	0x01	0x13	0x8a	0x6b
8x	0x3a	0x91	0x11	0x41	0x4f	0x67	0xdc	0xea	0x97	0xf2	0xcf	0xce	0xf0	0xb4	0xe6	0x73
9x	0x96	0xac	0x74	0x22	0xe7	0xad	0x35	0x85	0xe2	0xf9	0x37	0xe8	0x1c	0x75	0xdf	0x6e
Ax	0x47	0xf1	0x1a	0x71	0x1d	0x29	0xc5	0x89	0x6f	0xb7	0x62	0x0e	0xaa	0x18	0xbe	0x1b
Bx	0xfc	0x56	0x3e	0x4b	0xc6	0xd2	0x79	0x20	0x9a	0xdb	0xc0	0xfe	0x78	0xcd	0x5a	0xf4
Cx	0x1f	0xdd	0xa8	0x33	0x88	0x07	0xc7	0x31	0xb1	0x12	0x10	0x59	0x27	0x80	0xec	0x5f
Dx	0x60	0x51	0x7f	0xa9	0x19	0xb5	0xa4	0x0d	0x2d	0xe5	0x7a	0x9f	0x93	0x9c	0xef	0x61
Ex	0xa0	0xe0	0x3b	0x4d	0xae	0x2a	0xf5	0xb0	0xc8	0xeb	0x3c	0x83	0x53	0x99	0x61	0x21
Fx	0x17	0x2b	0x04	0x7e	0xba	0x77	0xd6	0x26	0xe1	0x69	0x14	0x63	0x55	0x0c	0x2d	0x7d

# AES Modes

IV: 000000000000000b

Cipher (ECB): 0a7ec77951291795bac6690c9e7f4c0d

[Cn7HeVEpF5w6xmkMnn9MDQ==]

decrypt: hello

Cipher (CBC): 81466cb66599fc317dce1ebfa1f4e9ab

[gUZstmWZ/DF9zh6/ofTpqw==]

decrypt: hello

Cipher (CFB): 79aa48d5b5a07feba6d89d3ad5b277c8

[eapI1bwgf+um2J061bj3yA==]

decrypt: hello

Cipher (OFB): 794a54871f0473bd0556fdafed76de59

[eupuhx8Ec70Fvv2v7xbewQ==]

decrypt: hello

Cipher (CTR): 3abfc1269139221192ff31301a899791

[Or/B]pE5IhGS/zEwGomXkQ==]

decrypt: hello

## Example

ECB



# 2. Symmetric Key

Basics

Block or Stream?

Secret Key Methods

Salting

AES

3DES

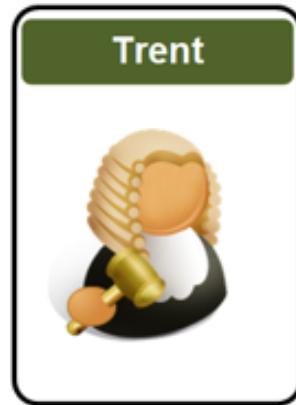
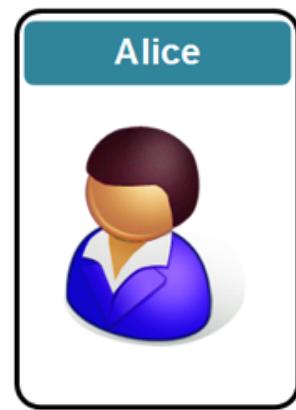
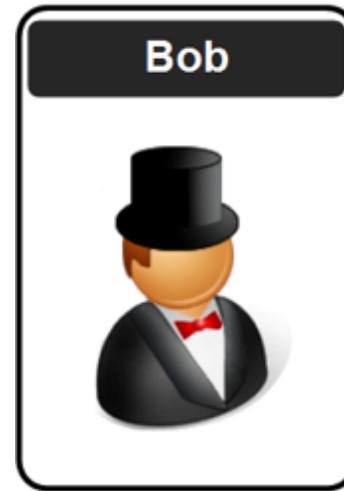
ChaCha20/Poly1305

Key Entropy

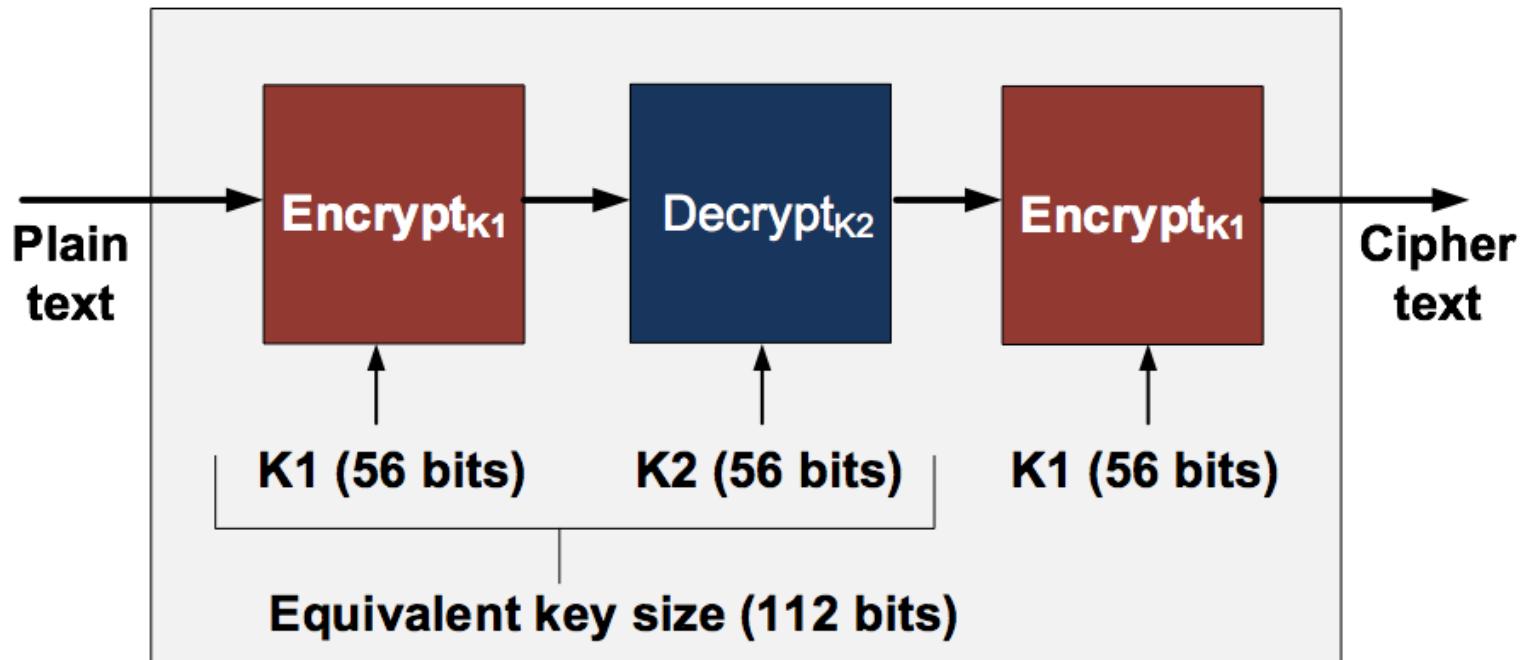
Cloud-based Symmetric Key

**Prof Bill Buchanan OBE**

<https://asecuritysite.com/symmetric>



# 3-DES



# 2. Symmetric Key

Basics

Block or Stream?

Secret Key Methods

Salting

AES

3DES

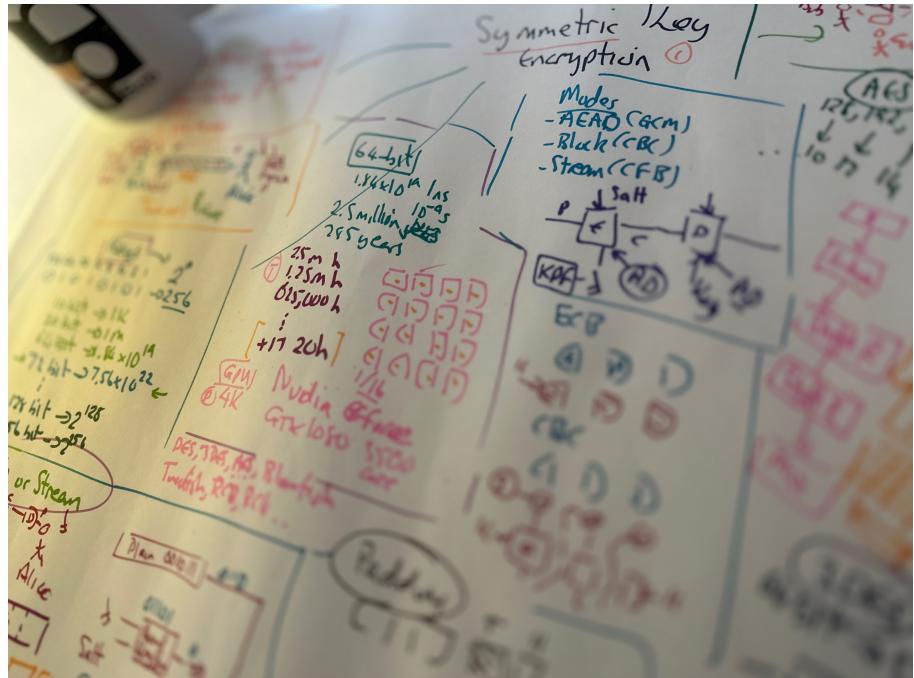
**ChaCha20/Poly1305 and RC4**

Key Entropy

Cloud-based Symmetric Key

**Prof Bill Buchanan OBE**

<https://asecuritysite.com/encryption>



INFORMATIONAL  
Errata Exist

Internet Research Task Force (IRTF)  
Request for Comments: 7539  
Category: Informational  
ISSN: 2070-1721

Y. Nir  
Check Point  
A. Langley  
Google, Inc.  
May 2015

Try!

Try!

## ChaCha20 and Poly1305 for IETF Protocols

### Abstract

This document defines the ChaCha20 stream cipher as well as the use of the Poly1305 authenticator, both as stand-alone algorithms and as a "combined mode", or Authenticated Encryption with Associated Data (AEAD) algorithm.

This document does not introduce any new crypto, but is meant to serve as a stable reference and an implementation guide. It is a product of the Crypto Forum Research Group (CFRG).

[ Docs ] [ txt | pdf ] [ draft-irtf-cfrq-c... ] [ Diff1 ] [ Diff2 ] [ Errata ]

INFORMATIONAL  
Errata Exist

Internet Research Task Force (IRTF)  
Request for Comments: 7539  
Category: Informational  
ISSN: 2070-1721

Y. Nir  
Check Point  
A. Langley  
Google, Inc.  
May 2015

Try!

Try!

## ChaCha20 and Poly1305 for IETF Protocols

### Abstract

This document defines the ChaCha20 stream cipher as well as the use of the Poly1305 authenticator, both as stand-alone algorithms and as a "combined mode", or ~~Authenticated Encryption with Associated Data~~ (AEAD) algorithm. Internet Engineering Task Force (IETF)

This document does not serve as a stable product of the Cryo-EM Standardization Working Group. Updates: [5246](#), [6347](#)  
Category: Standards Track  
ISSN: 2070-1721

A. Langley

W. Chang

Google, Inc.

giannopoulos

N. Mavrogiannopoulos

Red Hat

J. Strombergson

Secworks Sweden AB

ALKS Sweden AB  
S-1600 Trosa

S. JOSELISSON  
SJD. AB.

SJD AB

June 2016

Internet Research Task Force (IRTF)

Request for Comments: 7539

Category:

ISSN:

Abstract:

This document specifies ChaCha20 and Salsa20 as new cipher suites for Transport Layer Security (TLS).

(AES)

This document specifies ChaCha20 and Salsa20 as new cipher suites for Transport Layer Security (TLS).

ChaCha20 and Salsa take a 256-bit key (or a 128-bit version) and a 32-bit nonce. This creates a key stream, which is then XORed with the plaintext stream. In software, it is more than three times faster than AES, and is well suited to lower-powered devices and in real-time communications.

ChaCha operates on 32-bit bits with a key of 256 bits ( $K=(k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7)$ ). This output blocks of 512-bits for the key stream ( $Z$ ), and which is EX-XORed with the plaintext stream.

Try!

Try!

A. Langley  
W. Chang  
Google, Inc.

N. Mavrogiannopoulos  
Red Hat

J. Strombergson  
Secworks Sweden AB

S. Josefsson  
SJD AB

June 2016

```

def KSA(key):
    keylength = len(key)

    S = range(256)

    j = 0
    for i in range(256):
        j = (j + S[i] + key[i % keylength]) % 256
        S[i], S[j] = S[j], S[i] # swap

    return S

def PRGA(S):
    i = 0
    j = 0
    while True:
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i] # swap

        K = S[(S[i] + S[j]) % 256]
        yield K

def RC4(key):
    S = KSA(key)
    return PRGA(S)

```

[f1](#) [Diff2] [Errata]

INFORMATIONAL  
**Errata Exist**  
 Y. Nir  
 Check Point

256-bit key (or a 128-bit  
 This creates a key stream,  
 plaintext stream. In  
 times faster than AES,  
 levered devices and in

ts with a key of 256 bits  
 k7). This output blocks of  
 ), and which is EX-ORed

Try!

Try!

A. Langley  
 W. Chang  
 Google, Inc.  
 N. Mavrogiannopoulos  
 Red Hat  
 J. Strombergson  
 Secworks Sweden AB  
 S. Josefsson  
 SJD AB  
 June 2016

Cipher Suites for Transport Layer Security (TLS)

# 2. Symmetric Key

Basics

Block or Stream?

Secret Key Methods

Salting

AES

3DES

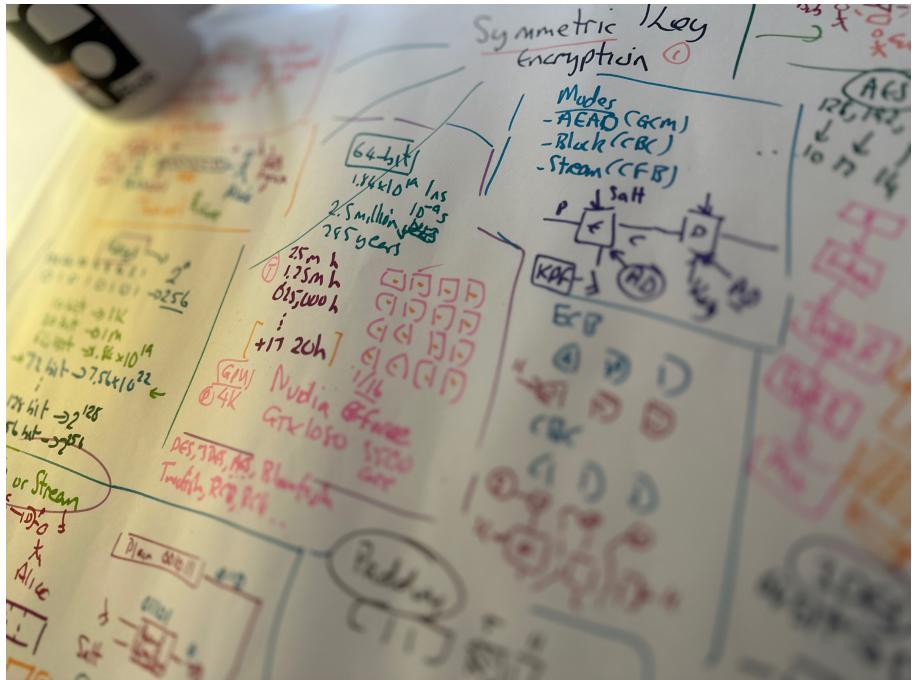
ChaCha20/Poly1305

**Key Entropy**

Cloud-based Symmetric Key

**Prof Bill Buchanan OBE**

<https://asecuritysite.com/symmetric>





# Key Entropy

$$\text{Key Entropy} = \log_2(\text{Phrases}) = \frac{\log_{10}(\text{Phrases})}{\log_{10}(2)}$$

$$\text{Key Entropy} = \log_2(26^8) = \frac{\log_{10}(26^8)}{\log_{10}(2)} = 37.6 \text{ bits}$$

[Try!](#)

Password definition	Number of possible characters	Total number of passwords	Entropy (bits)
[0-9]	10	100,000,000	26.6
[a-z]	26	2.08827x10 <sup>11</sup>	37.6
[a-zA-Z]	52	5.34597x10 <sup>13</sup>	45.6
[a-zA-Z0-9]	62	2.1834x10 <sup>14</sup>	47.6
[a-zA-Z0-9\$%!@+=]	68	4.57163x10 <sup>14</sup>	48.7

# 2. Symmetric Key

Basics

Block or Stream?

Secret Key Methods

Salting

AES

3DES

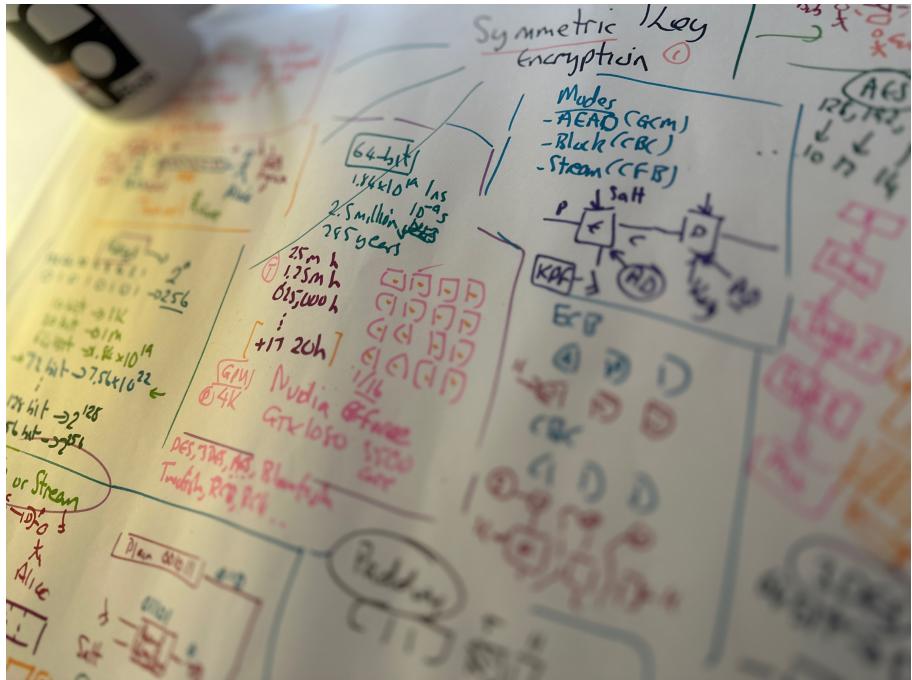
ChaCha20/Poly1305

Key Entropy

Cloud-based Symmetric Key

Prof Bill Buchanan OBE

<https://asecuritysite.com/symmetric>



# Key Storage/Transport

```
3059301306072a8648ce3d020106082a8648ce3d030107034200042927b10512bae  
3eddcfe467828128bad2903269919f7086069c8c4df6c732838c7787964eaac00e5  
921fb1498a60f4606766b3d9685001558d1a974e7341513
```

DER format

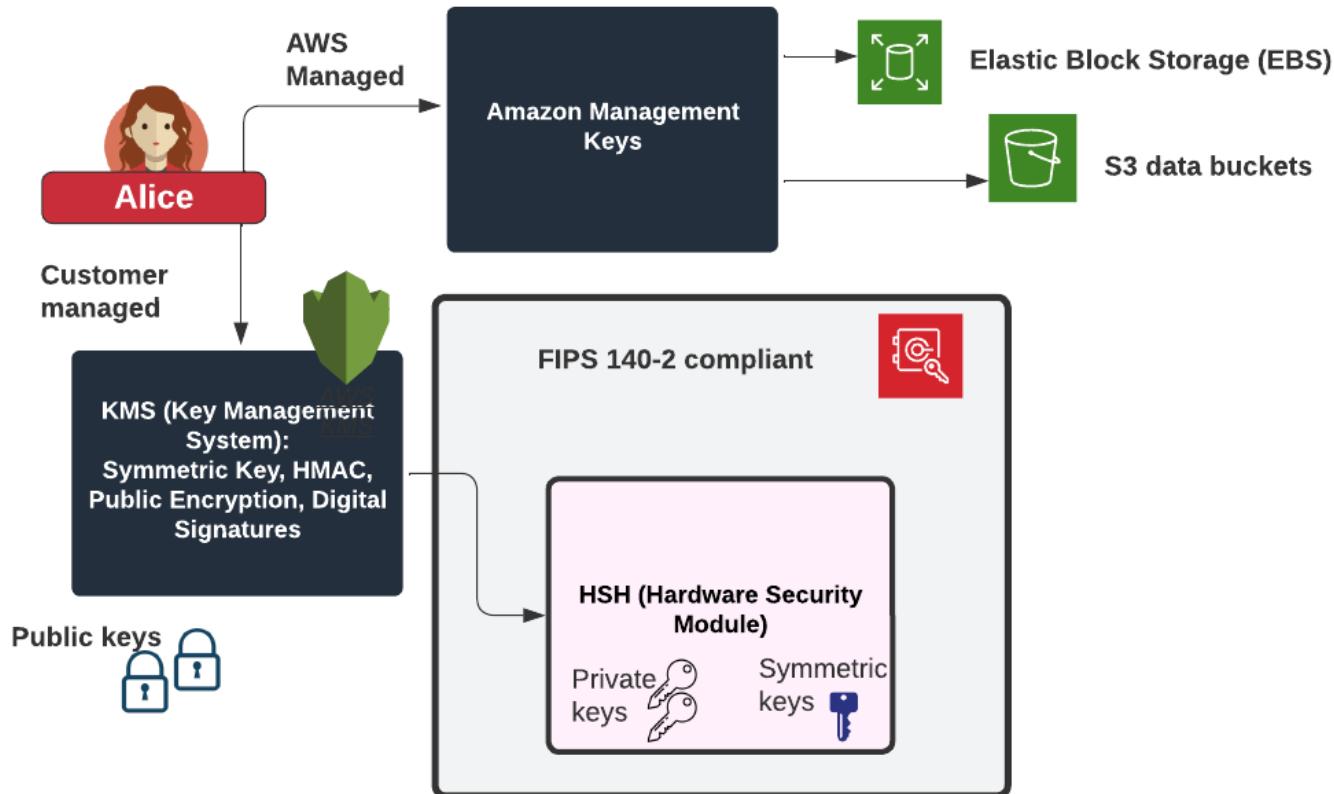
JSON format

```
{  
  "primaryKeyId": 1331912396,  
  "key": [  
    {"keyData": {  
        "typeUrl": "type.googleapis.com/google.crypto.tink.AesGcmKey",  
        "keyMaterialType": "SYMMETRIC",  
        "value": "GhBpskWWTrE27e2w67X4Tzfs"  
      },  
      "outputPrefixType": "TINK",  
      "keyId": 1331912396,  
      "status": "ENABLED"  
    }  
  ]  
}
```

PEM format

```
-----BEGIN PUBLIC KEY-----  
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIIBCgKCAQEAw51PMB  
Ulyh22zy3hEhlsNPH6/Cqg0HJorX1WbNKLfiU2aAt24jn4CC+y  
0L4XZxm14QvKKImIOM0MbLS1Te29n64HuuQ9owKLHuSMww4wiL  
HL6x7nK/Pq72eoQ/etFBkaX5nYGUD/+G+5BgAPx1mBgU5/y9+/  
Tfa6rDMSAbmJ0tkk1ghnuaq4dSoHWbW+zpHMVtjtHgzDGhX9Kj  
p2yDLULUbsd04ylacTkxyIc92ZHdZeP6Hh+KhNC04Z65zwXLEA  
xwIDAQAB  
-----END PUBLIC KEY-----
```

# KMS and HSM



# Cloud-based Symmetric Key

Key Management Service (KMS)

AWS managed keys

**Customer managed keys**

Custom key stores

The screenshot shows the AWS KMS console interface. A success message at the top indicates a new key was created with alias "MyPublicKey2" and key ID "mrk-563b89d2385b4e70899e0dfd5158ef7b". Below this, the "Customer managed keys" list is displayed, showing three existing keys: "MyPublicKey" (key ID: 68ded69b-6c19-4b34-9f91-f8c2628ee612), "BillsNewKey" (key ID: 98a90e1f-2cb5-4564-a3aa-d0c060cdcf0a), and "MyPublicKey2" (key ID: mrk-563b89d2385b4e70899e0dfd5158ef7b). The "Create key" button is visible at the top right of the list.

Aliases	Key ID	Status	Key spec	Key usage
<a href="#">MyPublicKey</a>	68ded69b-6c19-4b34-9f91-f8c2628ee612	Enabled	RSA_2048	Encrypt and decrypt
<a href="#">BillsNewKey</a>	98a90e1f-2cb5-4564-a3aa-d0c060cdcf0a	Enabled	SYMMETRIC_DEFAULT	Encrypt and decrypt
<a href="#">MyPublicKey2</a>	mrk-563b89d2385b4e70899e0dfd5158ef7b	Enabled	RSA_2048	Encrypt and decrypt

# Cloud-based Symmetric Key

Key Management Service (KMS)

AWS managed keys

**Customer managed keys**

Custom key stores

**Success**  
Your AWS KMS key was created with alias MyPublicKey2 and key ID `mrk-563b89d2385b4e70899e0dfd5158ef7b`.

**Configure key**

**Customer managed keys (3)**

Aliases	Key ID	Status
<input type="checkbox"/> MyPublicKey	<code>68ded69b-6c19-4b34-9f91-f8c2628ee612</code>	Enabled
<input type="checkbox"/> BillsNewKey	<code>98a90e1f-2cb5-4564-a3aa-d0c060cdcf0a</code>	Enabled
<input type="checkbox"/> MyPublicKey2	<code>mrk-563b89d2385b4e70899e0dfd5158ef7b</code>	Enabled

**Key type** [Help me choose](#)

**Symmetric**  
A single key used for encrypting and decrypting data or generating and verifying HMAC codes.

**Asymmetric**  
A public and private key pair used for encrypting and decrypting data or signing and verifying messages.

**Key usage** [Help me choose](#)

**Encrypt and decrypt**  
Use the key only to encrypt and decrypt data.

**Generate and verify MAC**  
Use the key only to generate and verify hash-based message authentication codes (HMAC).

**Advanced options**

**Cancel** **Next**

# Cloud-based Symmetric Key

## Key Management Service (KMS)



Your AWS KMS key was created with alias MyPublicKey2 and key ID **mrk-563b89d2385b4e70899e0dfd5158ef7b**.

[View](#)

Once we have this, we can then encrypt the file using the "aws kms encrypt" command, and then use "fileb://1.txt" to refer to the file:

AW

Cu:

Cu:

```
aws kms encrypt --key-id alias/MySymKey --plaintext fileb://1.txt --query CiphertextBlob --output text > 1.out  
cat 1.out
```

This produces a ciphertext blob, and which is in Base64 format:

```
AQICAHgTBdpVTrBrduWKdNnvMoMMUWj0bqp+GqbghUx7qa6JwEQ7F2Fzubd+pcz3I06bFuLAAAAdjB0BgkqhkiG9w0BBwagZzB1AgEAMGAGCSqGS1b3DQEHATAeBglghkgBZQMEAS4wEQQM  
g13vWRVPyL7KK3k1AgEQgDP+dQ4KsqT94hiARF8zlybFAtXJJBIucc8M952KHmkJzBGQQP4f8YQQ70DELV97ZXizzME=
```

We could transmit this in Base64 format, but we need to convert it into a binary format for us to now decrypt it. For this we use the "Base64 -d" command:

```
$ base64 -i 1.out --decode > 1.enc  
$ cat 1.enc
```

The result is a binary output:

```
$ cat 1.enc  
x:UNSjg  
00e0` 1`He.0'[3*:l[v0t *H  
]Y07+y%3u  
D_3&$.q  
i @-_ {exddd_v1_w_W3n_145559
```

Now we can decrypt this with our key, and using the command of:

```
$ aws kms decrypt --key-id alias/BillsNewKey --output text --query Plaintext --ciphertext-blob fileb://1.enc > 2.out  
$ cat 2.out
```

The output of this is our secret message in Base64 format:

```
VGhpcyBpcyBteSBzZWNyZXQgZm1sZS4K
```

and now we can decode this into plaintext:

```
$ base64 -i 2.out --decode  
This is my secret file.
```

## Symmetric

public and private key pair used for encrypting and decrypting data or signing and verifying messages.

## Generate and verify MAC

the key only to generate and verify hash-based message authentication codes (HMAC).

[Cancel](#)[Next](#)

# 2. Symmetric Key

Basics

Block or Stream?

Secret Key Methods

Salting

AES

3DES

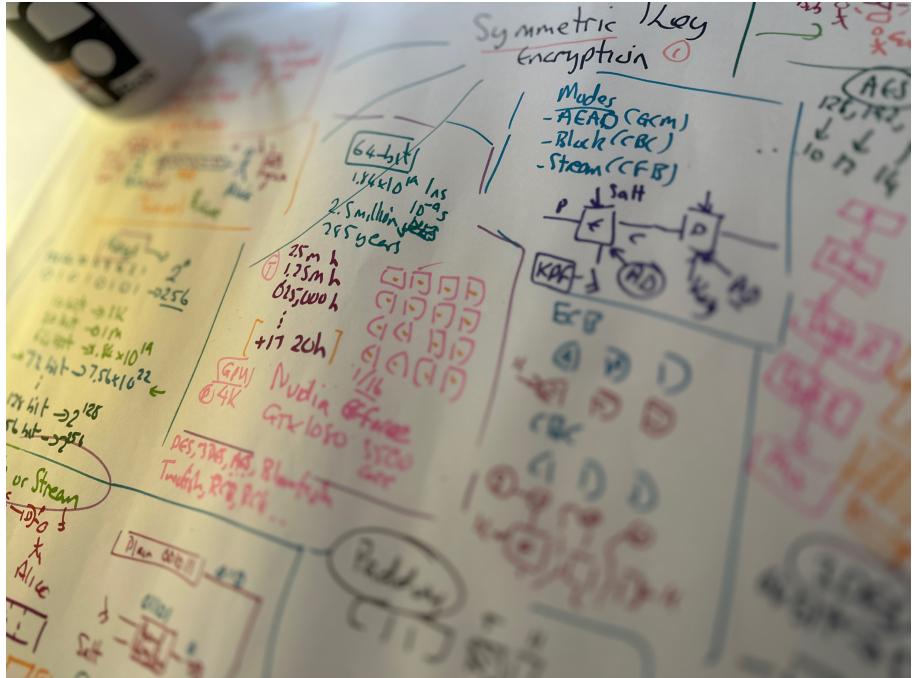
ChaCha20/Poly1305

Key Entropy

Cloud-based Symmetric Key

Prof Bill Buchanan OBE

<https://asecuritysite.com/symmetric>



# Quiz

Menti.com  
8489 7935

