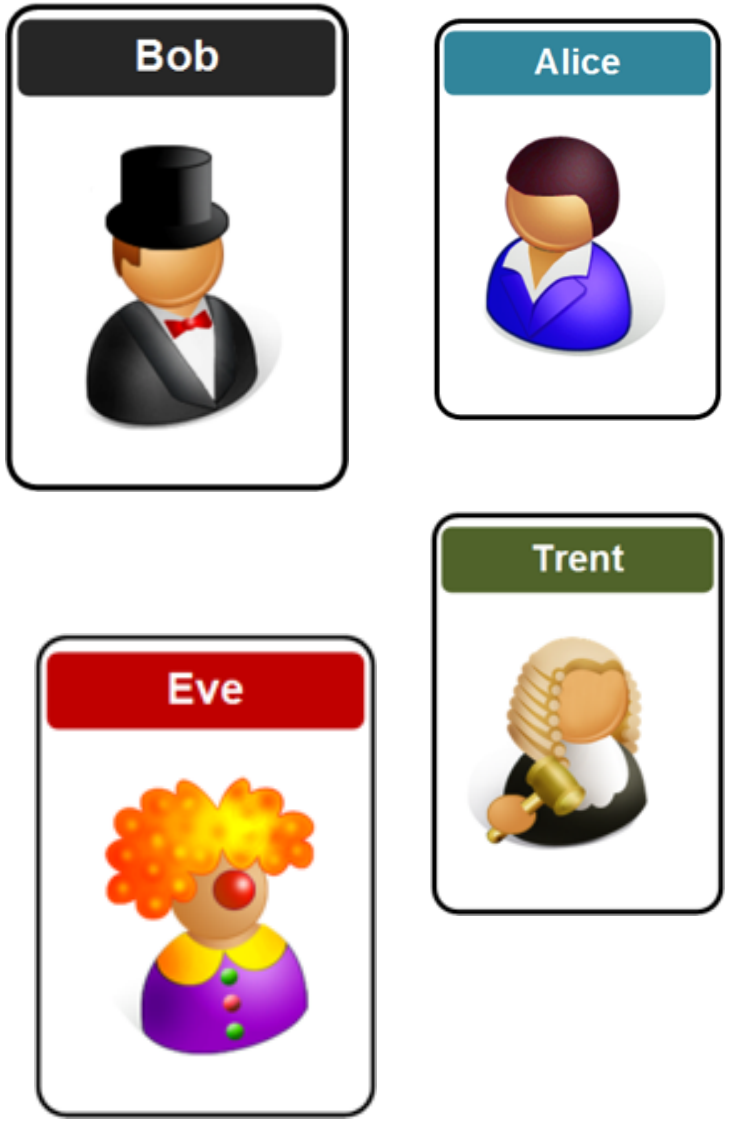
2. Symmetric Key

Basics
Block or Stream?
Secret Key Methods
Salting
AES: Creation and Modes
3DES
ChaCha20/Poly1305 and RC4
Key Entropy

Prof Bill Buchanan OBE
https://asecuritysite.com/encryption



No	Date	Subject	Lab
2	29 Jan 2021	Ciphers and Fundamentals Unit	Lab [Link] Demo [Link]
3	5 Feb 2021	Symmetric Key Unit	Lab [Link] Demo [Link]
4	12 Feb 2021	Hashing and MAC Unit	Lab [Link]
5	19 Feb 2021	Asymm	
6	26 Feb 2021	Key Ex	
7	5 Mar 2021	Guest l	
8	12 Mar 2021	Trust a	
9	19 Mar 2021	Test (U	
10	26 Mar 2021	Tunnel	
11	23 Apr 2021	Blockc	
12	30 Apr 2021	Future	
13	7 May 2021	Tokens	
14	14 May 2021	Trusted	
15	21 May 2021	Coursework Hand-in [Draft]	

main appliedcrypto / unit02_symmetric / Go to file Add file ...

billbuchanan Merge branch 'main' of https://github.com/billbuchanan/appliedcrypto ... d0fb8bf 2 minutes ago History

..		
lab	Merge branch 'main' of https://github.com/billbuchanan/appliedcrypto ...	2 minutes ago
lecture	Update	15 days ago
src	Update d_01.py	1 hour ago
.DS_Store	Update	2 minutes ago
README.md	Update	15 days ago

README.md

Unit 2: Symmetric Key

Hashing methods

Benchmarks and Password strength | Theory | MD5, SHA-1, etc | Other hash methods | Similarity hashes | SHA-3
Key derivation function | Non-crypto methods | One Time Passwords | SHAKE | MACs | Quantum hashes | Hash to ECC
Light-weight hash

Symmetric key

Principles | AES | Block encryption methods | Stream encryption | Hybrid methods | Encryption cracking (AES)
Mobile encryption | Disk Encryption

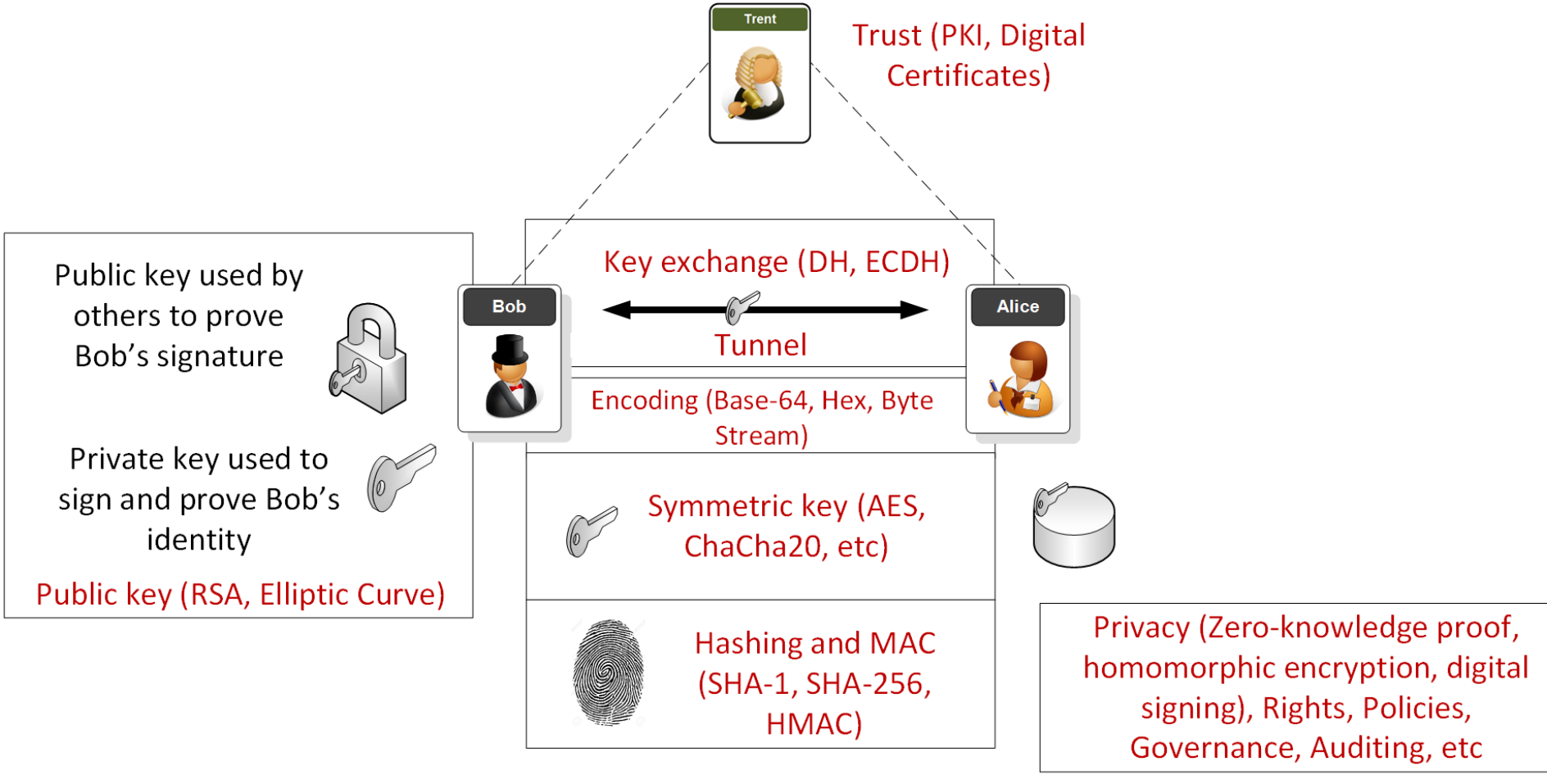
Zero knowledge

Zero-knowledge proof (Fiat Shamir and Schnorr) | Zero-knowledge proof | Zero-knowledge proof (Range proof)
Zero-knowledge proof (zkSnark) | Proof of Knowledge | Millionaire Problem | Bulletproofs | Secure Function Evaluation/MPC
Oblivious Transfer

https://asecuritysite.com/subjects/chapter122

Examples

Overview


Trent
Trust (PKI, Digital Certificates)
Public key used by others to prove Bob's signature
Private key used to sign and prove Bob's identity
Public key (RSA, Elliptic Curve)
Bob
Alice
Key exchange (DH, ECDH)
Tunnel
Encoding (Base-64, Hex, Byte Stream)
Symmetric key (AES, ChaCha20, etc)
Hashing and MAC (SHA-1, SHA-256, HMAC)
Privacy (Zero-knowledge proof, homomorphic encryption, digital signing), Rights, Policies, Governance, Auditing, etc

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

Article

Boiling the planet

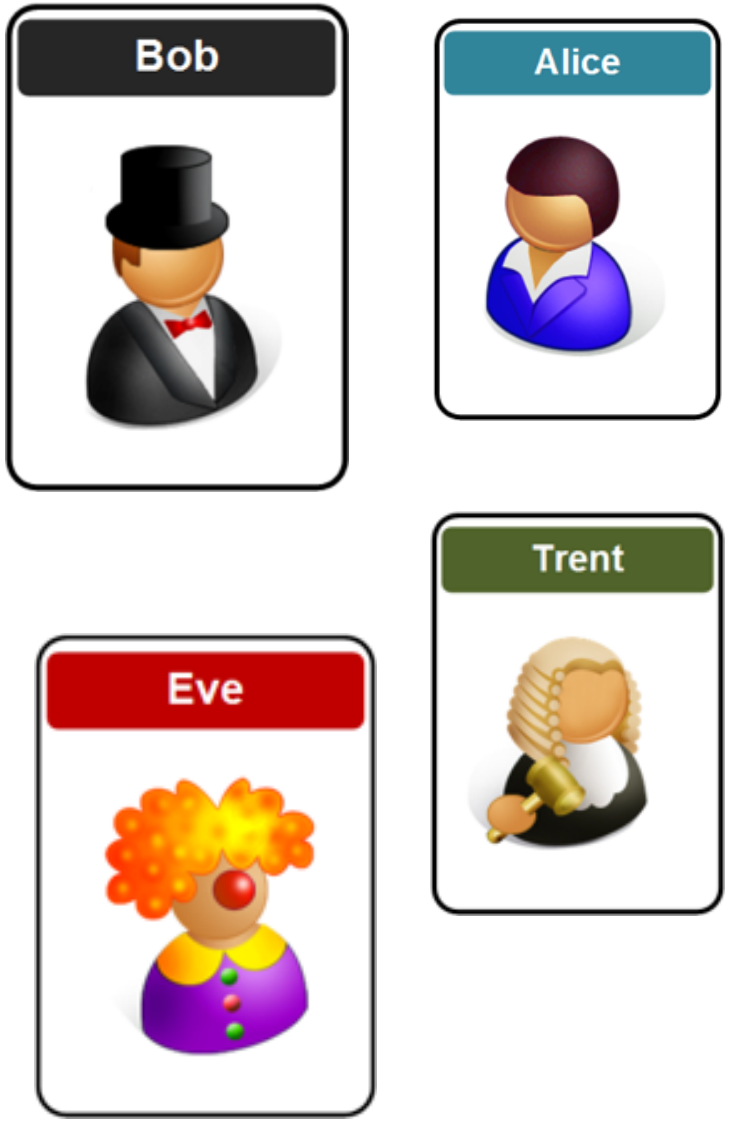
security level	volume of water to bring to a boil	bit-lengths		
		symmetric key	cryptographic hash	RSA modulus
teaspoon security	0.0025 liter	35	70	242
shower security	80 liter	50	100	453
pool security	2 500 000 liter	65	130	745
rain security	0.082 km^3	80	160	1130
lake security	89 km^3	90	180	1440
sea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

Article

2. Symmetric Key

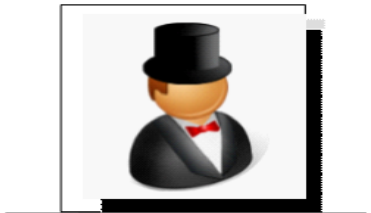
Basics
Block or Stream?
Secret Key Methods
Salting
AES
3DES
ChaCha20/Poly1305 and RC4
Key Entropy

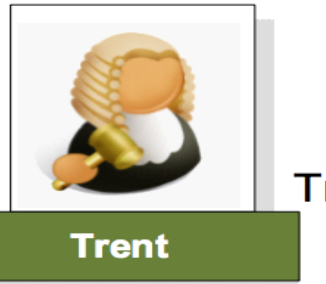
Prof Bill Buchanan OBE
https://asecuritysite.com/encryption



Intruder

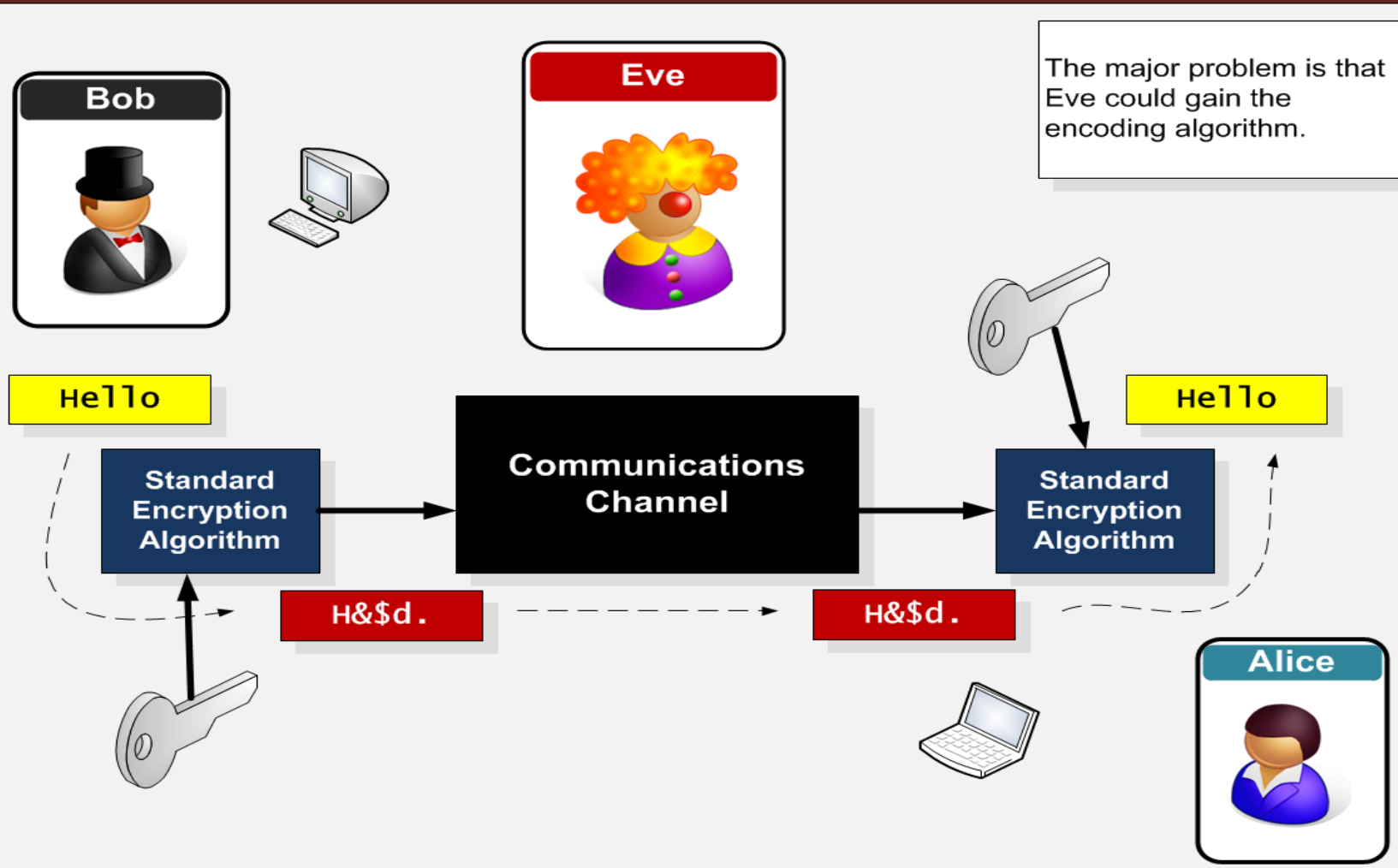
Privacy (Private Key)
Identity (Public Key)
Integrity (Public/Private Key)

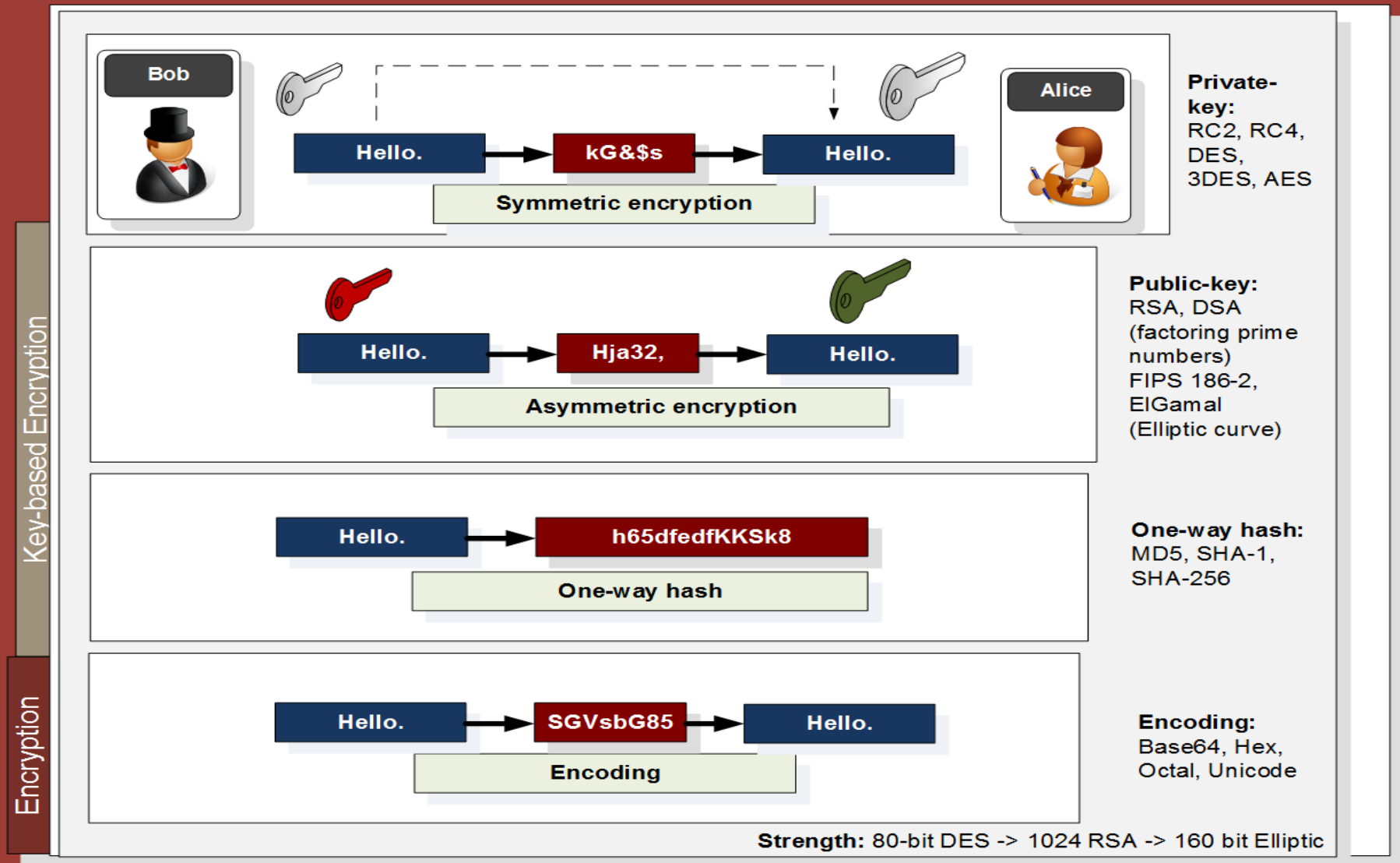


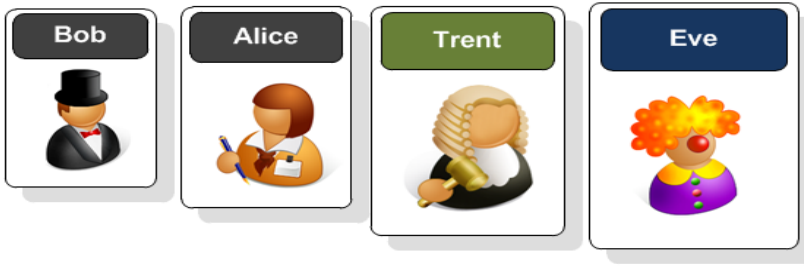


Trusted third party

Author: Prof Bill Buchanan


Bob
Eve
The major problem is that Eve could gain the encoding algorithm.
Hello
Standard Encryption Algorithm
Communications Channel
Standard Encryption Algorithm
Hello
H&$d.
H&$d.
Alice


Encryption
Key-based Encryption
Bob
Alice
Hello.
kG&$s
Hello.
Symmetric encryption
Private-key:
RC2, RC4, DES, 3DES, AES
Hello.
Hja32,
Hello.
Asymmetric encryption
Public-key:
RSA, DSA (factoring prime numbers)
FIPS 186-2, ElGamal (Elliptic curve)
Hello.
h65dfedfKKSk8
One-way hash
One-way hash:
MD5, SHA-1, SHA-256
Hello.
SGVsbG85
Hello.
Encoding
Encoding:
Base64, Hex, Octal, Unicode
Strength: 80-bit DES -> 1024 RSA -> 160 bit Elliptic



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

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.

Okay… we select a **64-bit key** …
which has 1.84×10^{19} combinations

18.4 million million million different keys
000000000000….0000000000000000000
To
111111111111….1111111111111111111

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

Author: Prof Bill Buchanan

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.

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

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)

Author: Prof Bill Buchanan

Time to crack

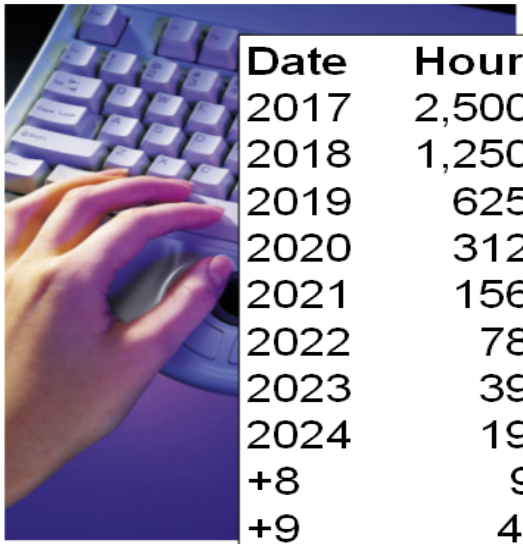
- It is important to understand the length of time that a message takes to crack as it may need to be secret for a certain time period.

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

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

Author: Prof Bill Buchanan



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.

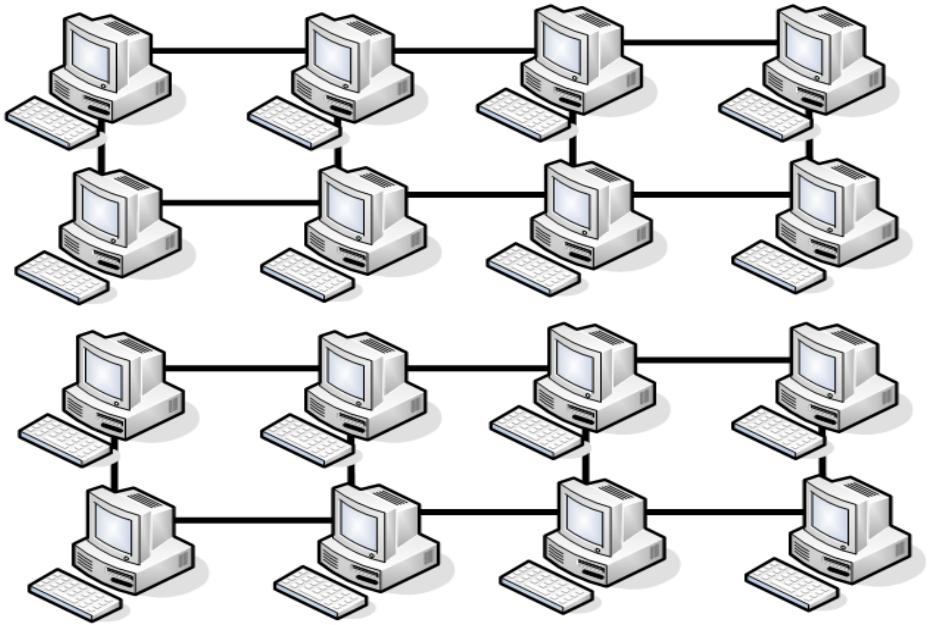
56-bit DES:
Developed
1975
30 years ago!
… now easily
crackable

Author: Prof Bill Buchanan

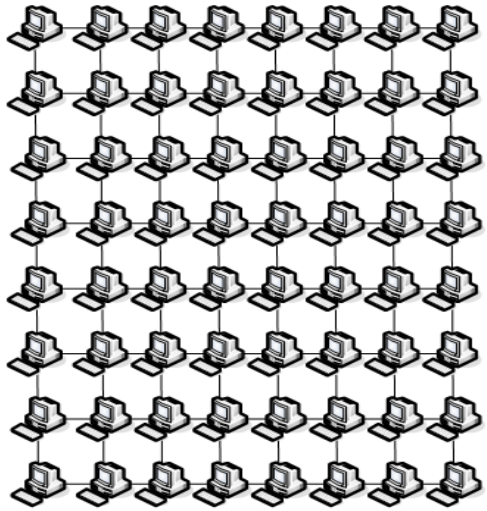
Parallel processing

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

2x1 =2 element array



4x4 = 16 element array

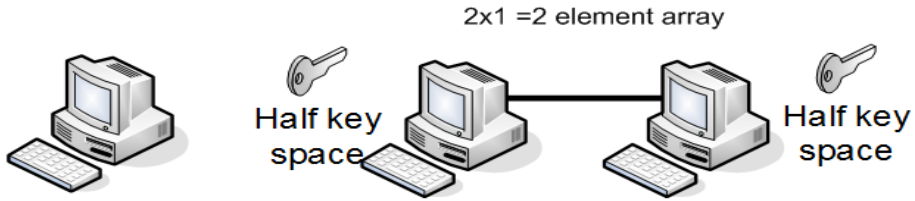


16x16 = 256 element array

Author: Prof Bill Buchanan

Parallel processing

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



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

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

– 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.

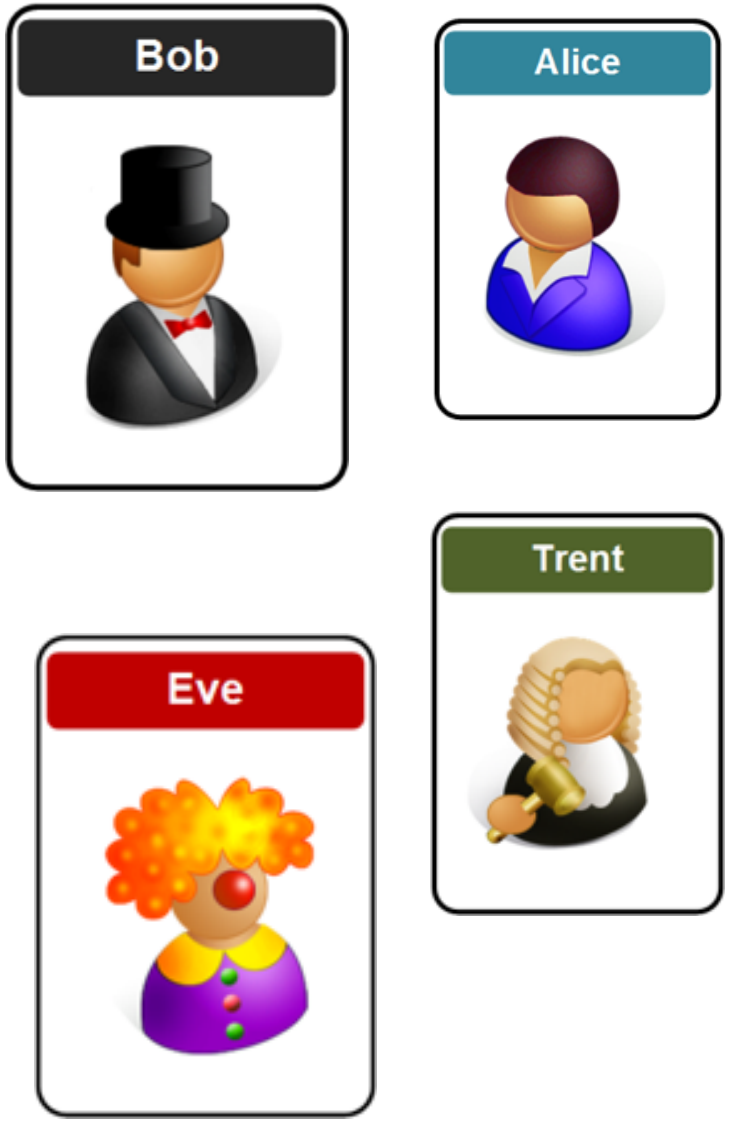
You can also mine Ether through a cloud mining contract with Hashflare or Genesis Mining.

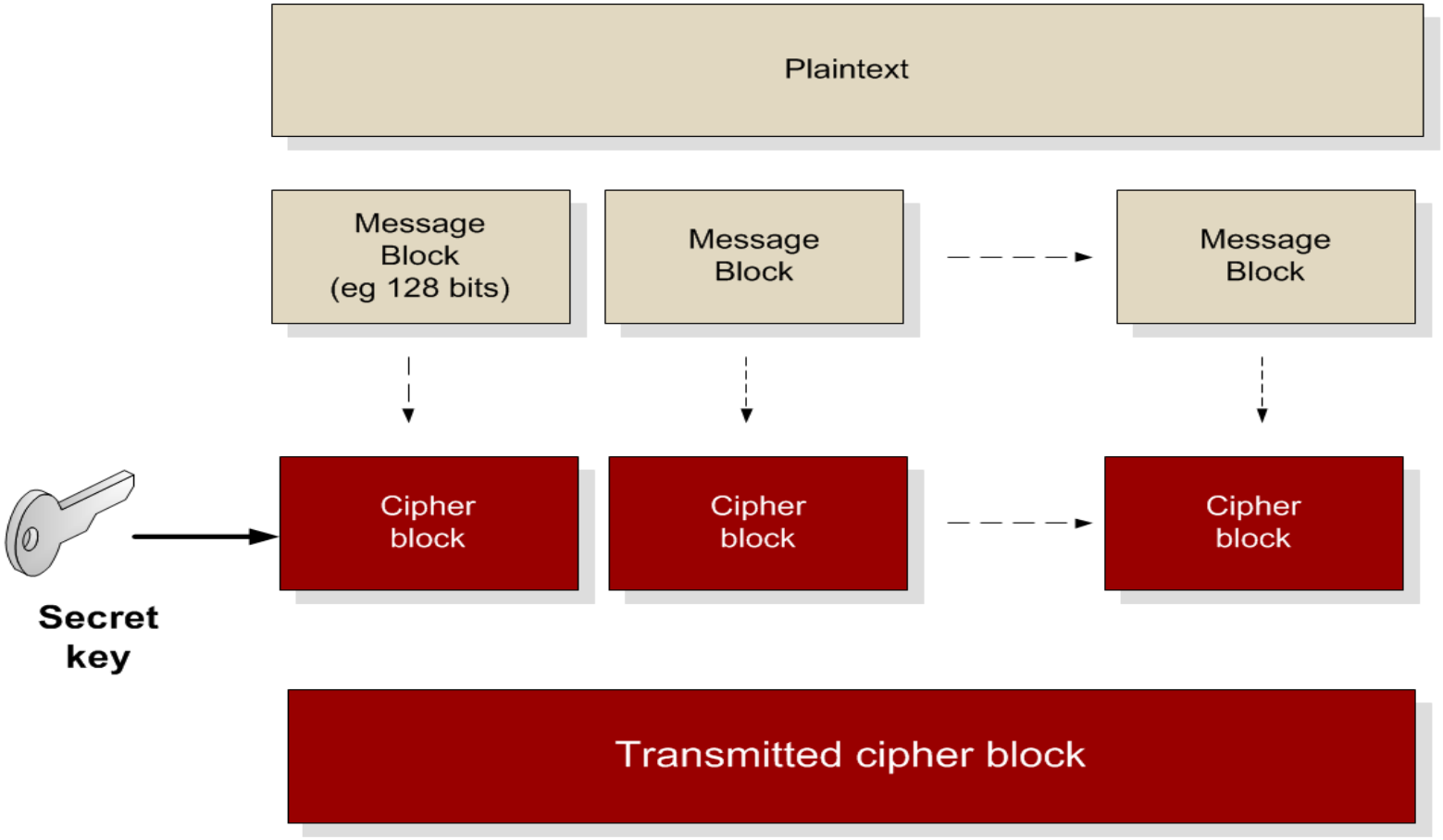
Disclosure: Mining equipment metrics are calculated based on a network hash rate of **182,293 GH/s** and using a ETH - USD exchange rate of **1 ETH = $ 1215.26**. These figures vary based on the total network hash rate and on the ETH to USD conversion rate. Equipment cost can vary, block reward is fixed 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

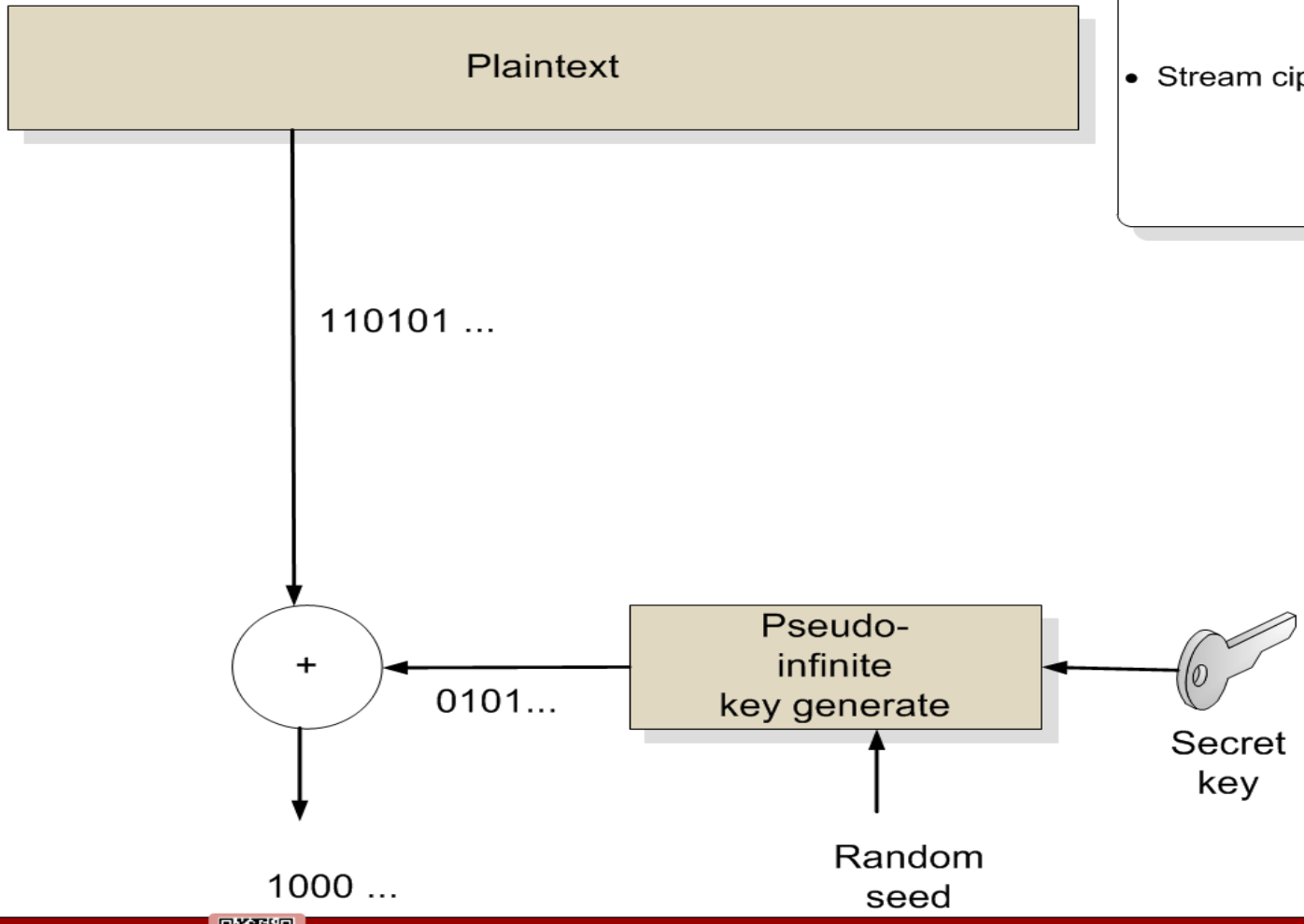
Prof Bill Buchanan OBE
https://asecuritysite.com/encryption



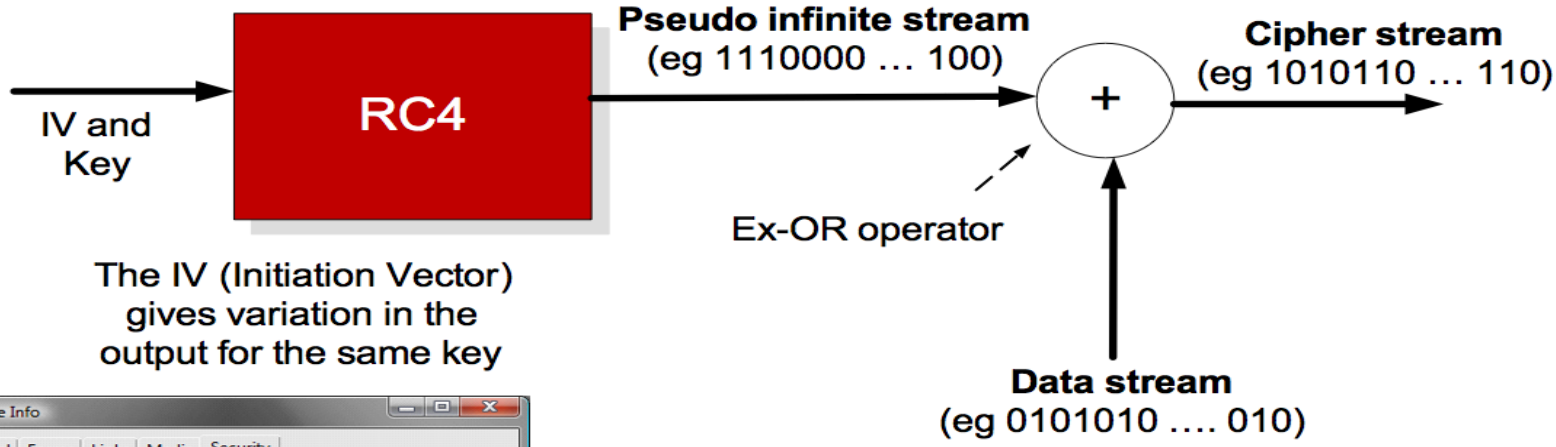

Plaintext
Message Block (eg 128 bits)
Message Block
Message Block
Cipher block
Cipher block
Cipher block
Secret key
Transmitted cipher block

Stream cipher

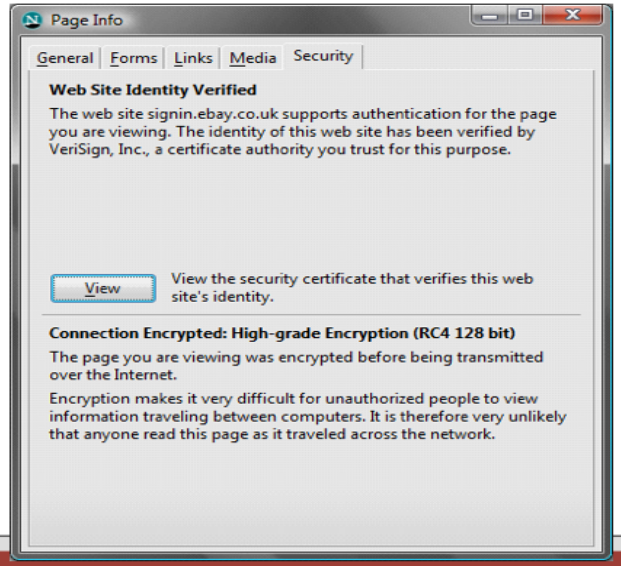
- Stream cipher (RC4)



RC4. This is a **stream** encryption algorithm, and is used in wireless communications (such as in WEP) and SSL (Secure Sockets).



The IV (Initiation Vector) gives variation in the output for the same key



Data stream	0101010 … 010
Pseudo infinite stream	1110000 … 100 ⊕
Cipher stream	1010110 … 110

Author: Prof Bill Buchanan

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.

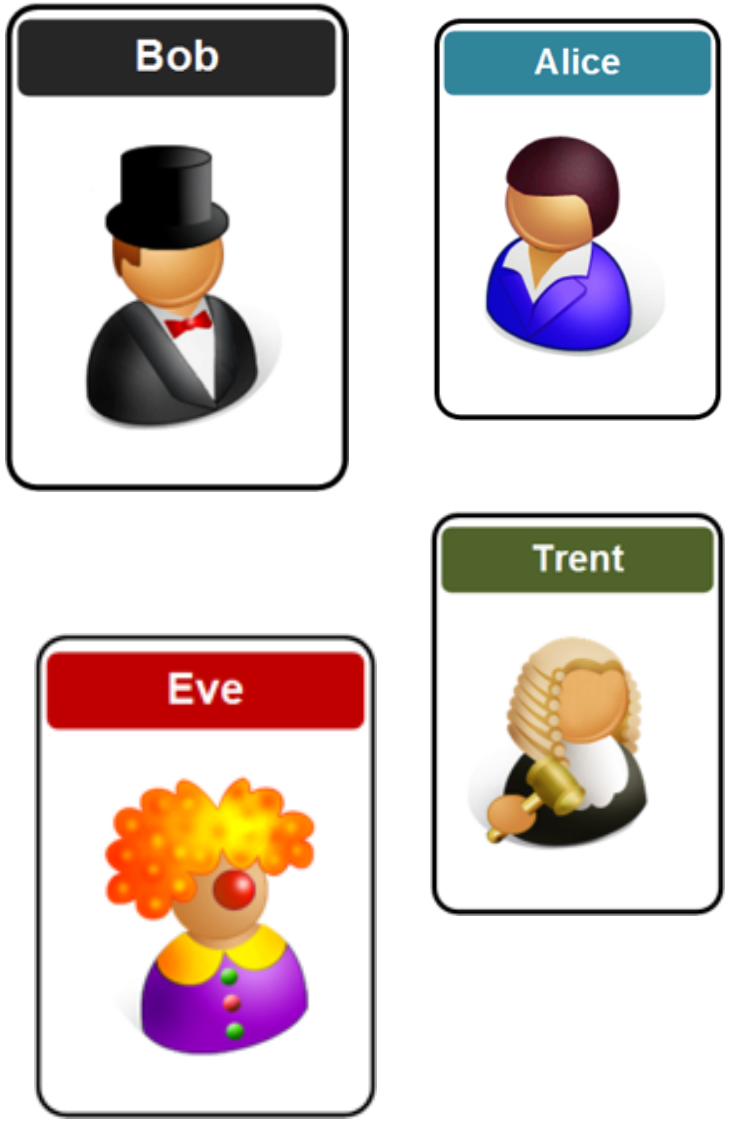
Padding

- After padding (CMS): 68656c6c6f0b0b0b0b0b0b0b0b0b0b0b
  Cipher (ECB): 0a7ec77951291795bac6690c9e7f4c0d
- After padding (Bit): 68656c6c6f800000000000000000
  Cipher (ECB): 731abffc2e3b2c2b5caa9ca2339344f9
- After padding (ZeroLen): 68656c6c6f0000000000000000000a
  Cipher (ECB): d28e2f7e8e44e068732b292bde444245
- After padding (Null): 68656c6c6f000000000000000000
  Cipher (ECB): 44479742246045 3d95856eb2a1520ece
- After padding (Random): 68656c6c6*ffc6ecfd884a38798d62*a**0a**
  Cipher (ECB): c2c88b4364d2c2dc6f2cac9ab73c995d

2. Symmetric Key

Basics
Block or Stream?
Secret Key Methods
Salting
AES
3DES
ChaCha20/Poly1305
Key Entropy

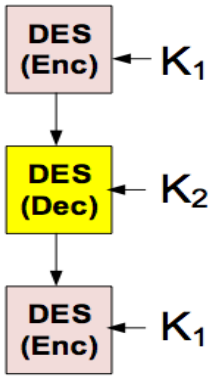
Prof Bill Buchanan OBE
https://asecuritysite.com/encryption



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 (2x54 bits - with a combination of the three keys - of which two of the keys are typically the same). The algorithm is EncryptK3(DecryptK2(EncryptK1(message), where K1 and K3 are typically the same (to keep compatibility).



Twofish

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

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.

RC2

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.

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.

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.

Author: Prof Bill Buchanan

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:

$Encrypt_{K3}(Decrypt_{K2}(Encrypt_{K1}(message)))$

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

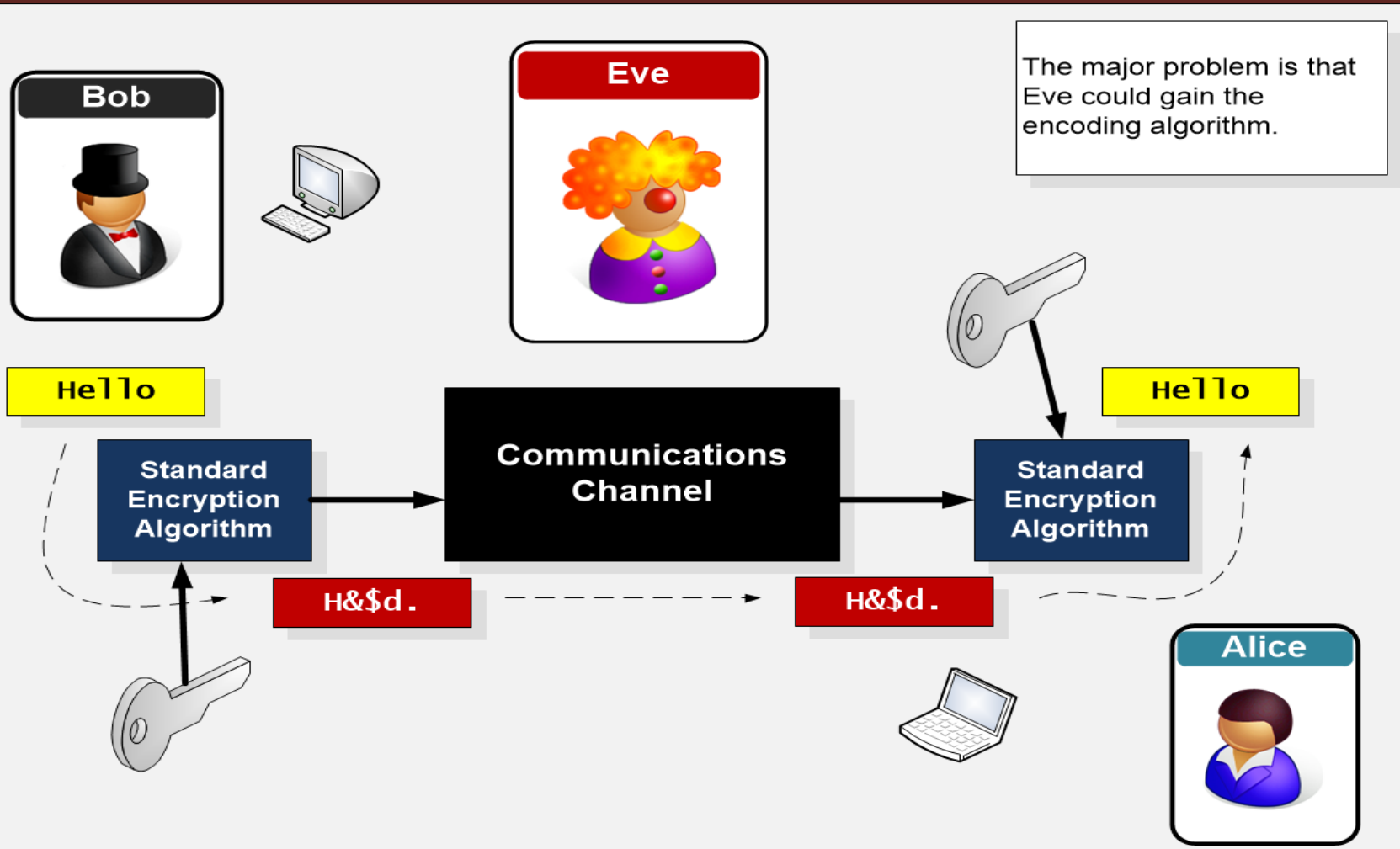
http://asecuritysite.com/encryption/threedes

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.

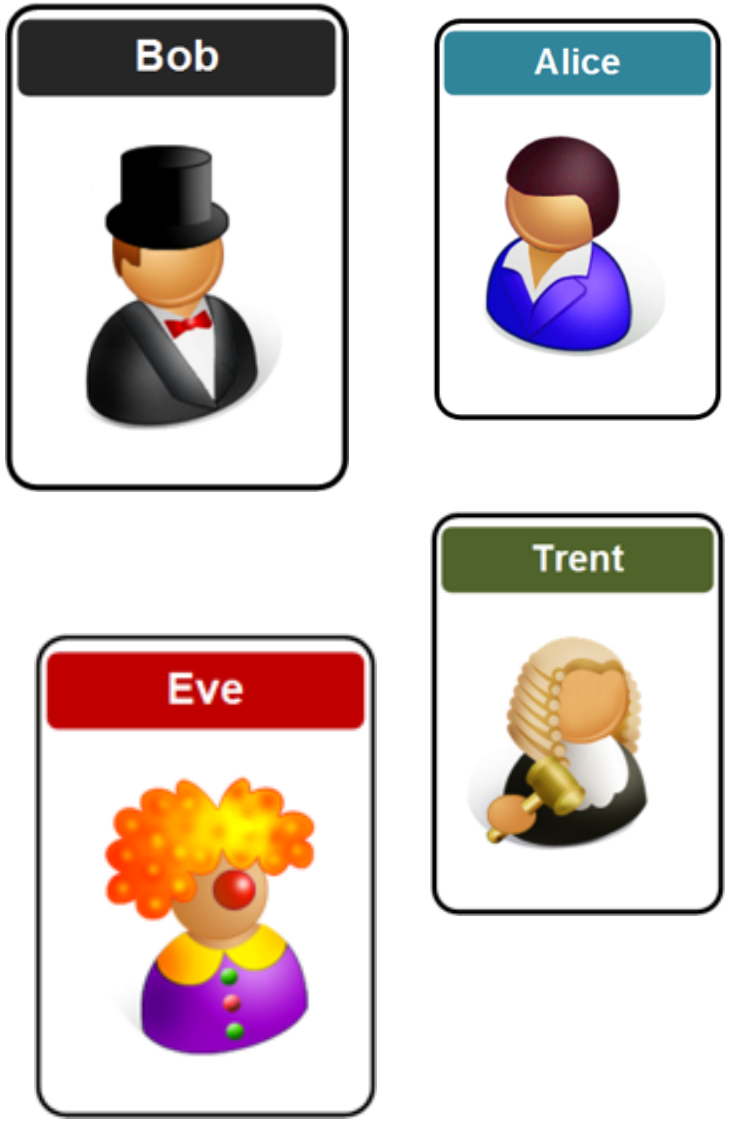
http://asecuritysite.com/encryption/aes

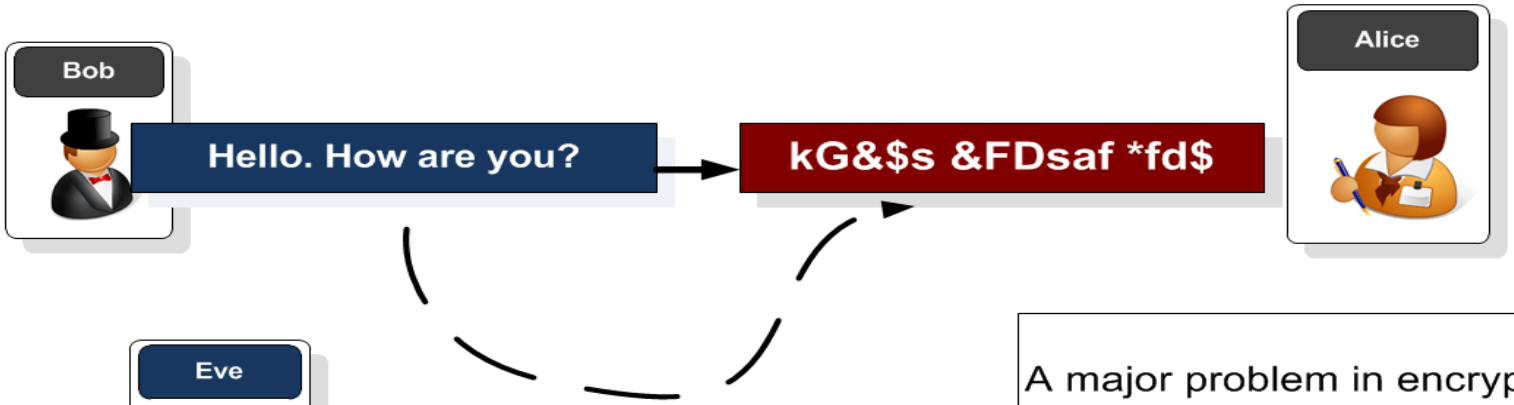

Bob
Eve
The major problem is that Eve could gain the encoding algorithm.
Hello
Standard Encryption Algorithm
Communications Channel
Standard Encryption Algorithm
Hello
H&$d.
H&$d.
Alice

2. Symmetric Key

Basics
Block or Stream?
Secret Key Methods
Salting
AES
3DES
ChaCha20/Poly1305
Key Entropy

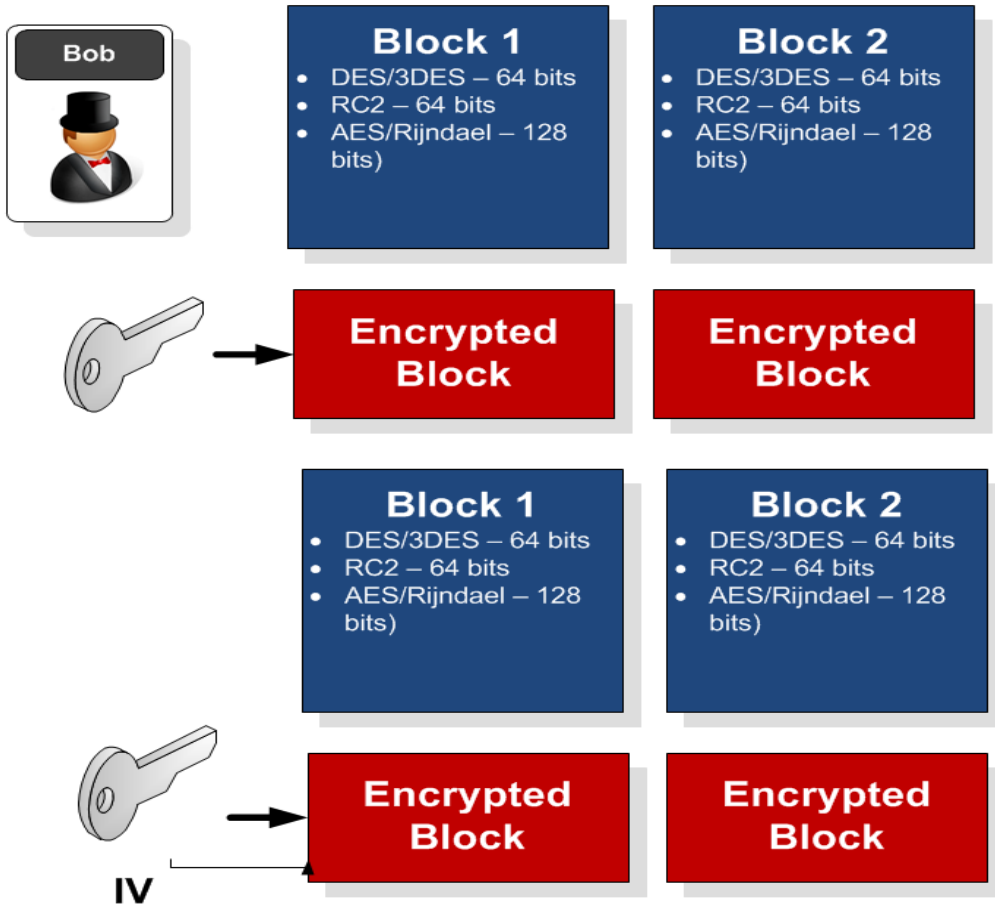
Prof Bill Buchanan OBE
https://asecuritysite.com/encryption





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.

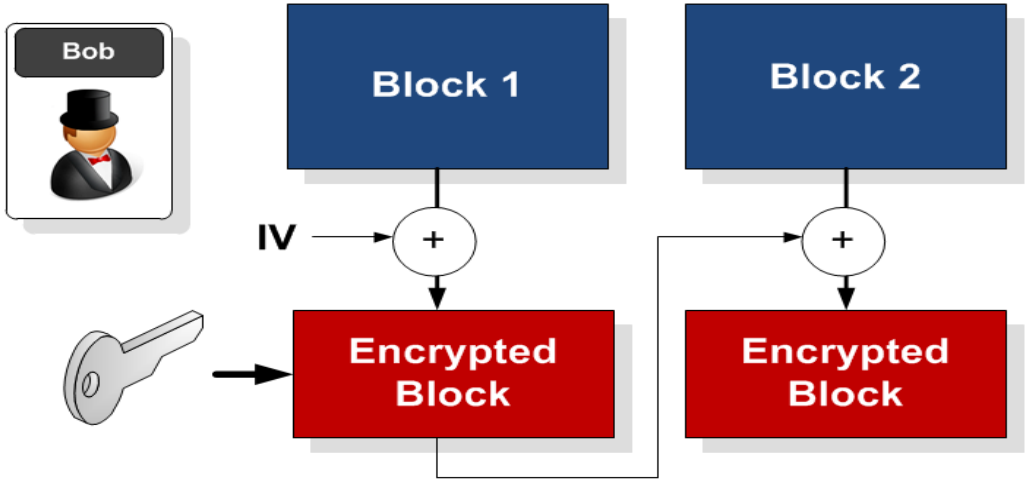
The solution is to add **salt** to the encryption key, as that it changes its operation from block-to-block (for block encryption) or data frame-to-data frame (for stream encryption)



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

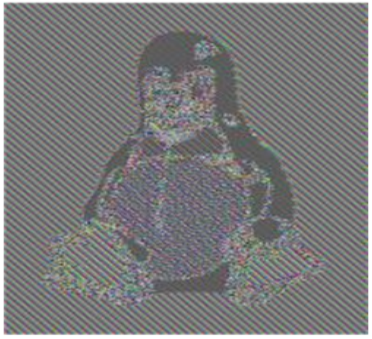


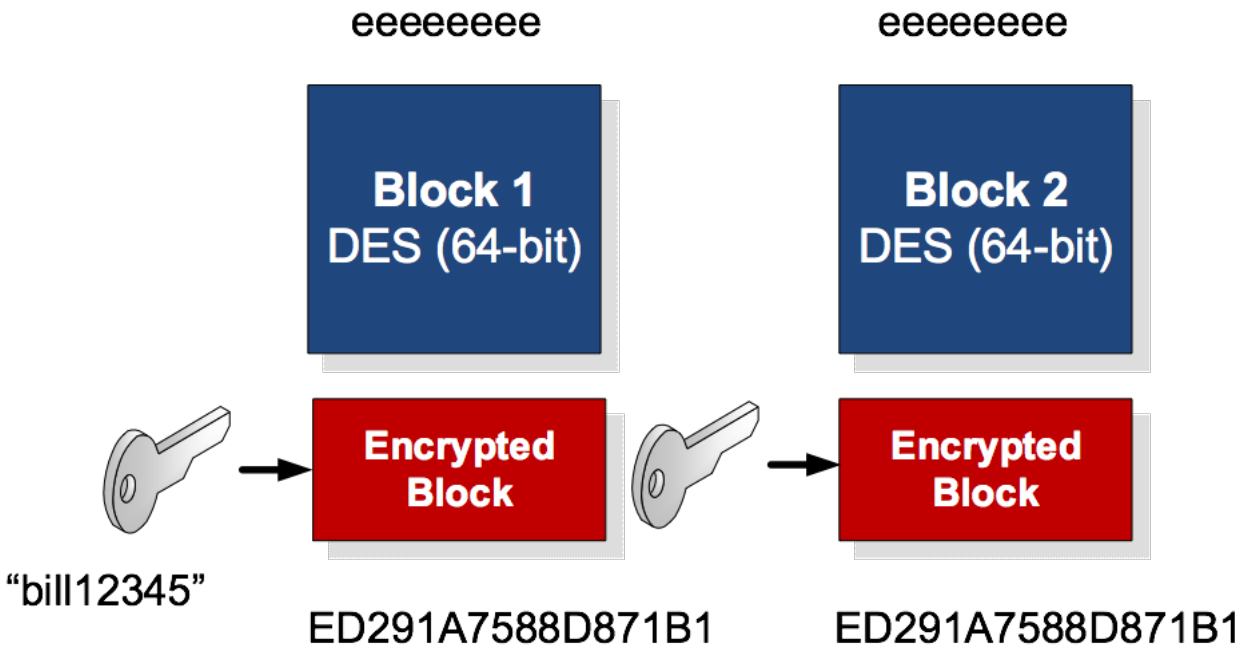
Image with AES using ECB

Image with AES using CBC

Image ref: http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

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



ED291A7588D871B1ED291A7588D871B1ED291A7588D
871B1ED291A7588D871B1ED291A7588D871B1ED291A
7588D871B18D6DF6795DDEDACD

AES Modes

```
IV: 000000000000000b
Input data (CMS): 68656c6c6f0b0b0b0b0b0b0b0b0b0b0b
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/BJpE5IhGS/zEwGomXkQ==]
  decrypt: hello
```

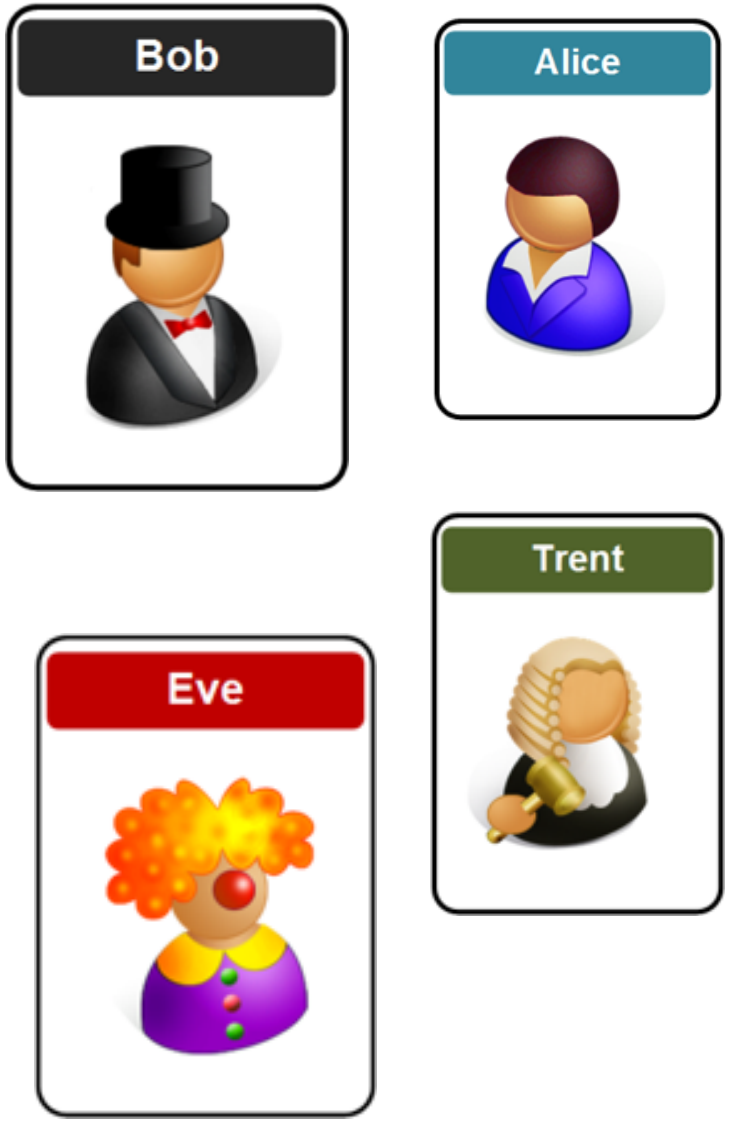
Example

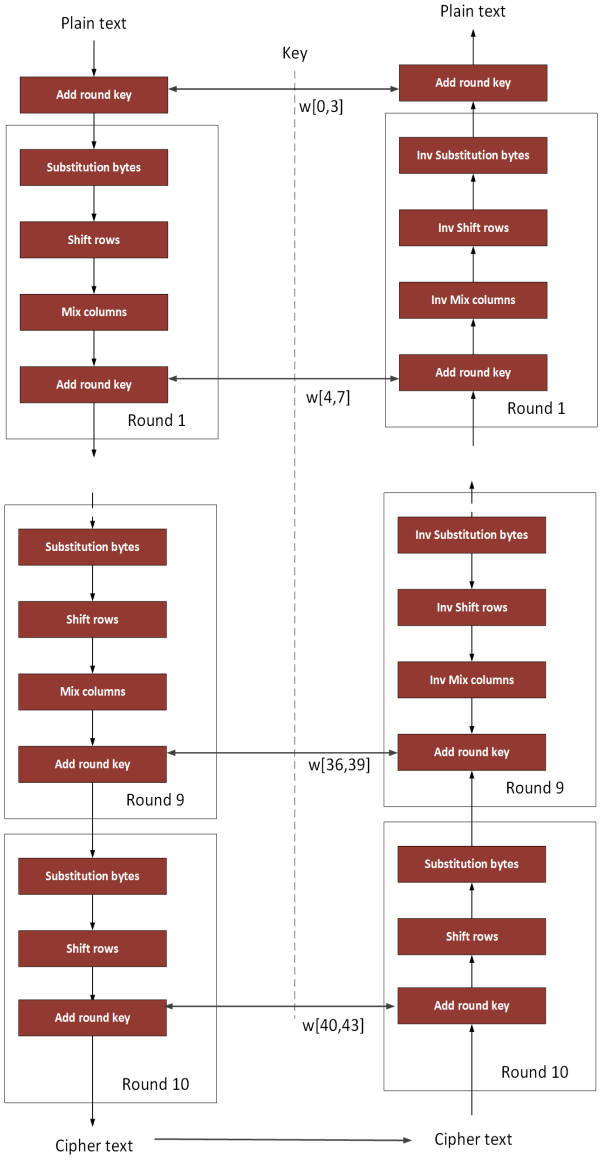
ECB

2. Symmetric Key

Basics
Block or Stream?
Secret Key Methods
Salting
AES
3DES
ChaCha20/Poly1305 and RC4
Key Entropy

Prof Bill Buchanan OBE
https://asecuritysite.com/encryption





AES

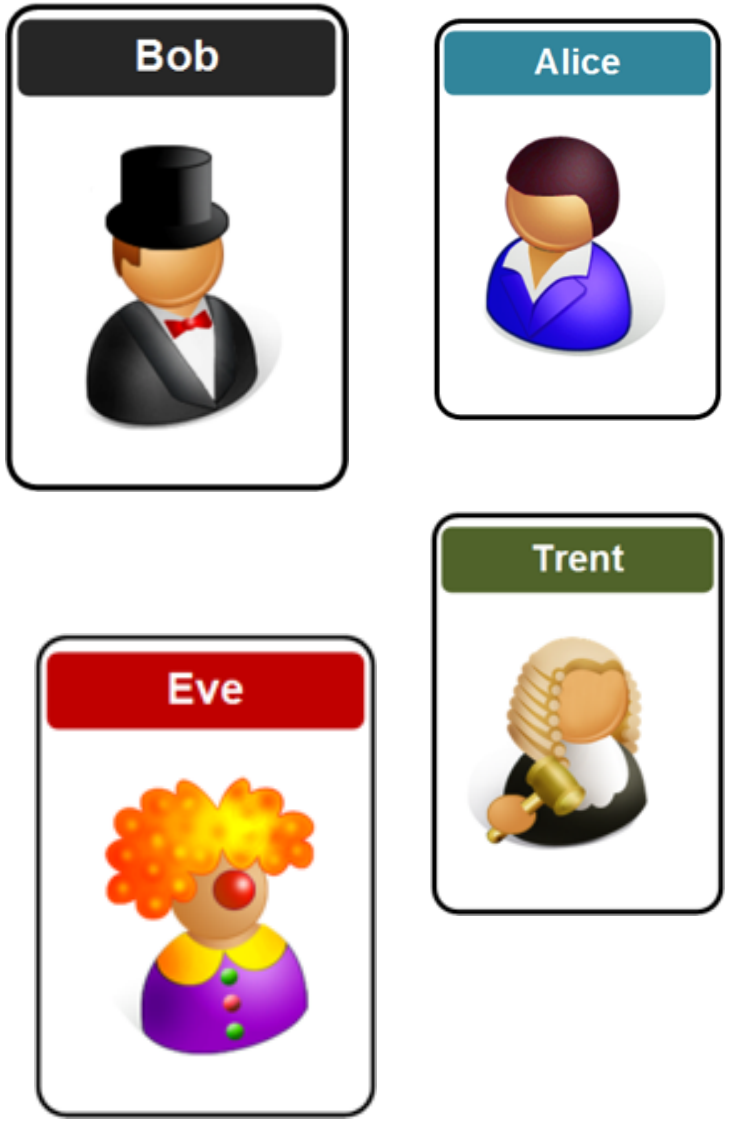
	0x	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	0x63	0x7c	0x77	0x7b	0xf2	0x6b	0x6f	0xc5	0x30	0x01	0x67	0x2b	0xfe	0xd7	0xab	0x76
1x	0xca	0x82	0xc9	0x7d	0xfa	0x59	0x47	0xf0	0xad	0xd4	0xa2	0xaf	0x9c	0xa4	0x72	0xc0
2x	0xb7	0xfd	0x93	0x26	0x36	0x3f	0xf7	0xcc	0x34	0xa5	0xe5	0xf1	0x71	0xd8	0x31	0x15
3x	0x04	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	0xe8	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	0xe1	0xf8	0x98	0x11	0x69	0xd9	0x8e	0x94	0x9b	0x1e	0x87	0xe9	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	0xe3	0x39	0x82	0x9b	0x2f	0xff	0x87	0x34	0x8e	0x43	0x44	0xc4	0xde	0xe9	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	0x8b	0xd1	0x25
4x	0x72	0xf8	0xf6	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	0x0a	0xf7	0xe4	0x58	0x05	0xb8	0xb3	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	0x4a	0x0d	0x2d	0xe5	0x7a	0x9f	0x93	0xc9	0x9c	0xef
Ex	0xa0	0xe0	0x3b	0x4d	0xae	0x2a	0xf5	0xb0	0xc8	0xeb	0xbb	0x3c	0x83	0x53	0x99	0x61
Fx	0x17	0x2b	0x04	0x7e	0xba	0x77	0xd6	0x26	0xe1	0x69	0x14	0x63	0x55	0x21	0x0c	0x7d

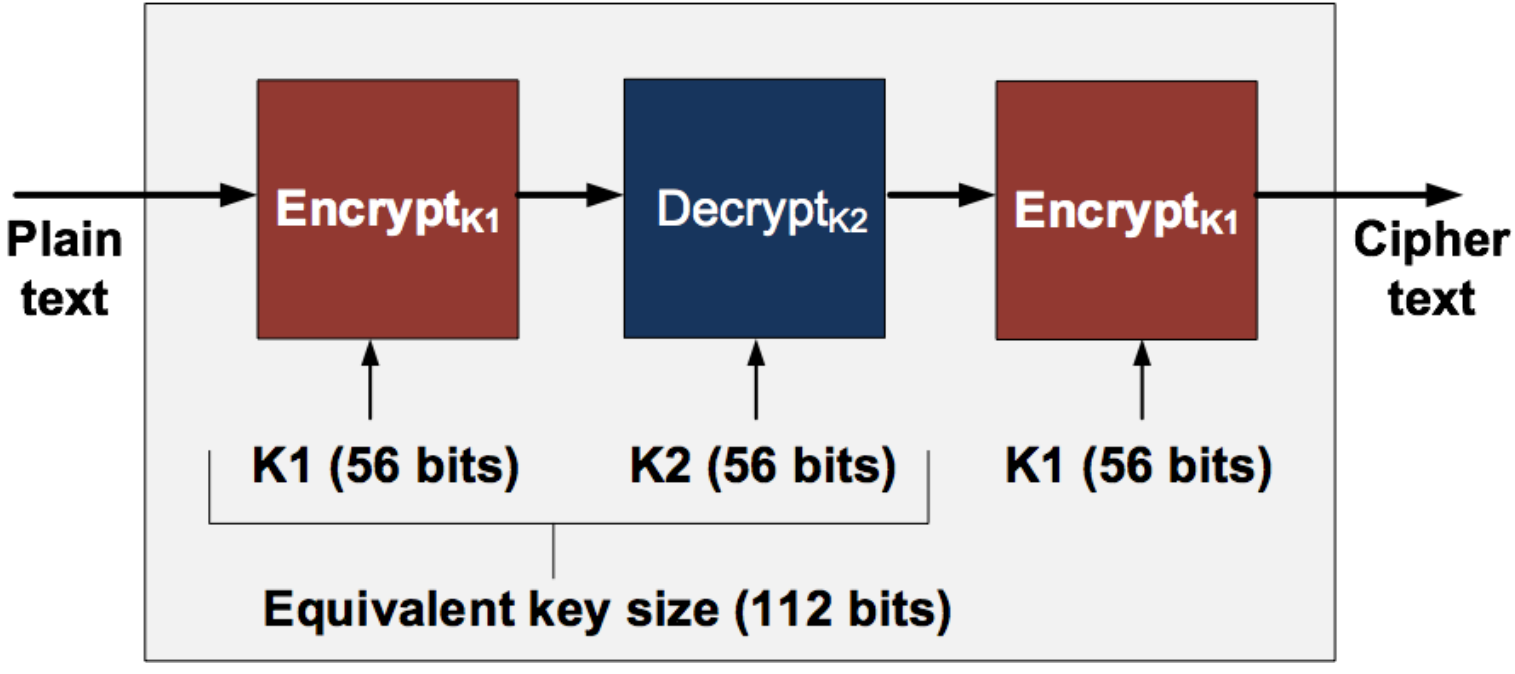
2. Symmetric Key

Basics
Block or Stream?
Secret Key Methods
Salting
AES
3DES
ChaCha20/Poly1305
Key Entropy

Prof Bill Buchanan OBE
https://asecuritysite.com/encryption



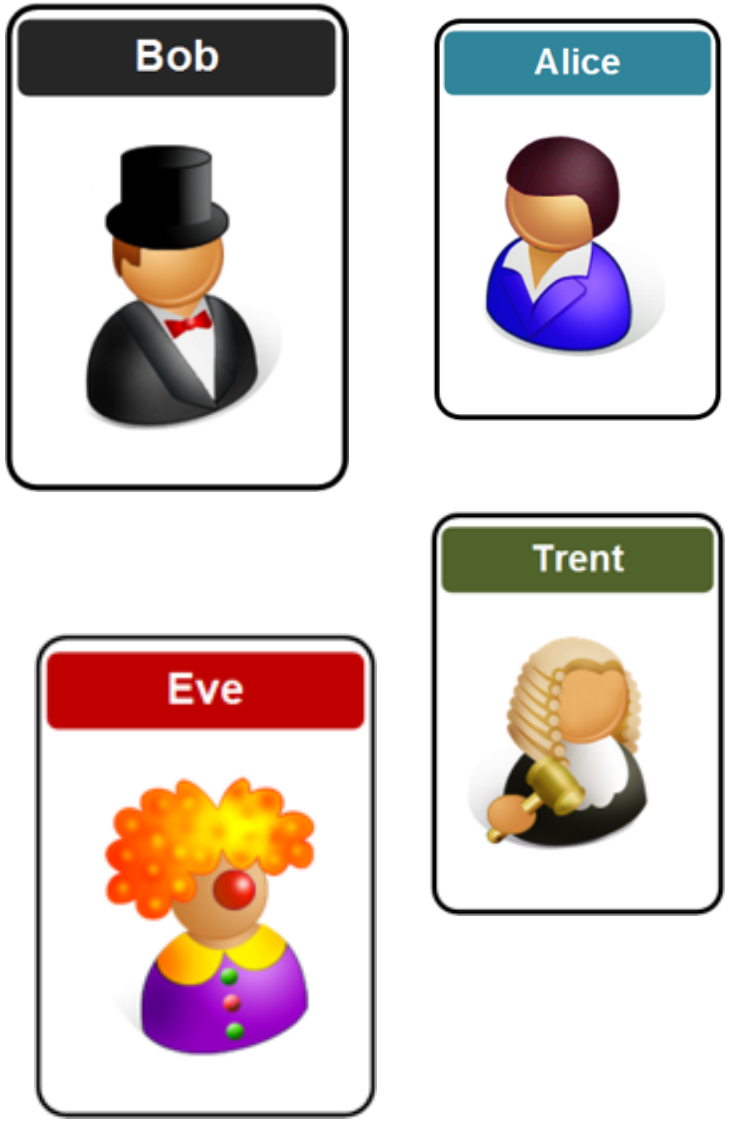
3-DES


Plain text
Encrypt$_{K1}$
Decrypt$_{K2}$
Encrypt$_{K1}$
Cipher text
K1 (56 bits)
K2 (56 bits)
K1 (56 bits)
Equivalent key size (112 bits)

2. Symmetric Key

Basics
Block or Stream?
Secret Key Methods
Salting
AES
3DES
ChaCha20/Poly1305 and RC4
Key Entropy

Prof Bill Buchanan OBE
https://asecuritysite.com/encryption



```python
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

Try!

Try!

256-bit key (or a 128-bit

This creates a key stream,

plaintext stream. In

times faster than AES,

wered devices and in real-

ts with a key of 256 bits

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

k7). This output blocks of

Cipher Suites for Transport Layer Security (TLS)

), and which is EX-ORed

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

Key Entropy

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

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

Try!

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

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