Symmetric Key

Basics
Block or Stream?
Secret Key Methods
Salting
AES
Key Entropy

Prof Bill Buchanan OBE

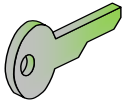
https://asecuritysite.com/symmetric

Typical application:
Diffie-Hellman used to generate private-key.
Public-key used for authentication.
Private-key used for encryption.

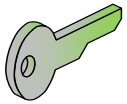
Encryption/ Decryption

Communications Channel

Encryption/ Decryption

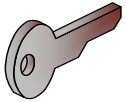


Key exchange (Diffie-Hellman)

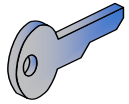


Secret key used to encrypt/decrypt
(DES/3DES/AES)

Public key



Used to authenticate (RSA)



Private key

Private key

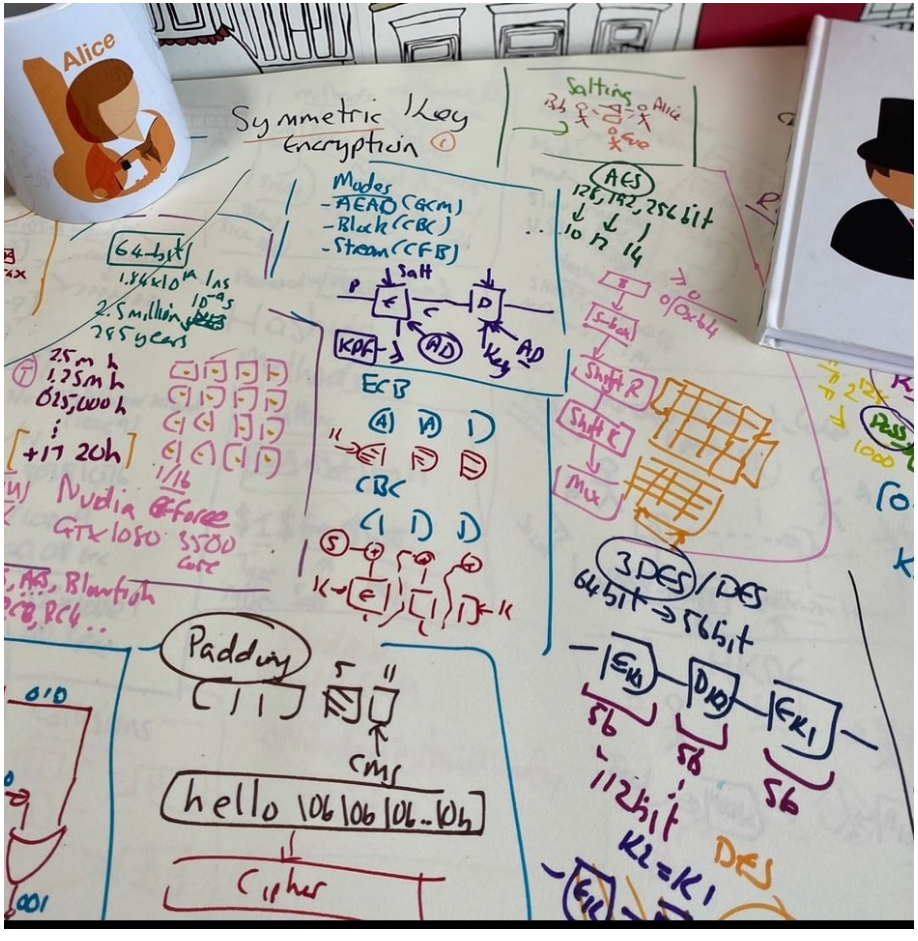
RSA 2048 bits
Replace by:
ElGamal 160bits

Public key

Author: Prof Bill Buchanan

Secret Key

Basics
Block or Stream?
Secret Key Methods
Salting
AES
Key Entropy



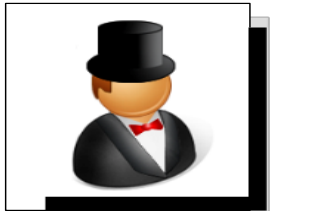
Prof Bill Buchanan OBE

https://asecuritysite.com/symmetric

Intruder

Eve

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



Bob

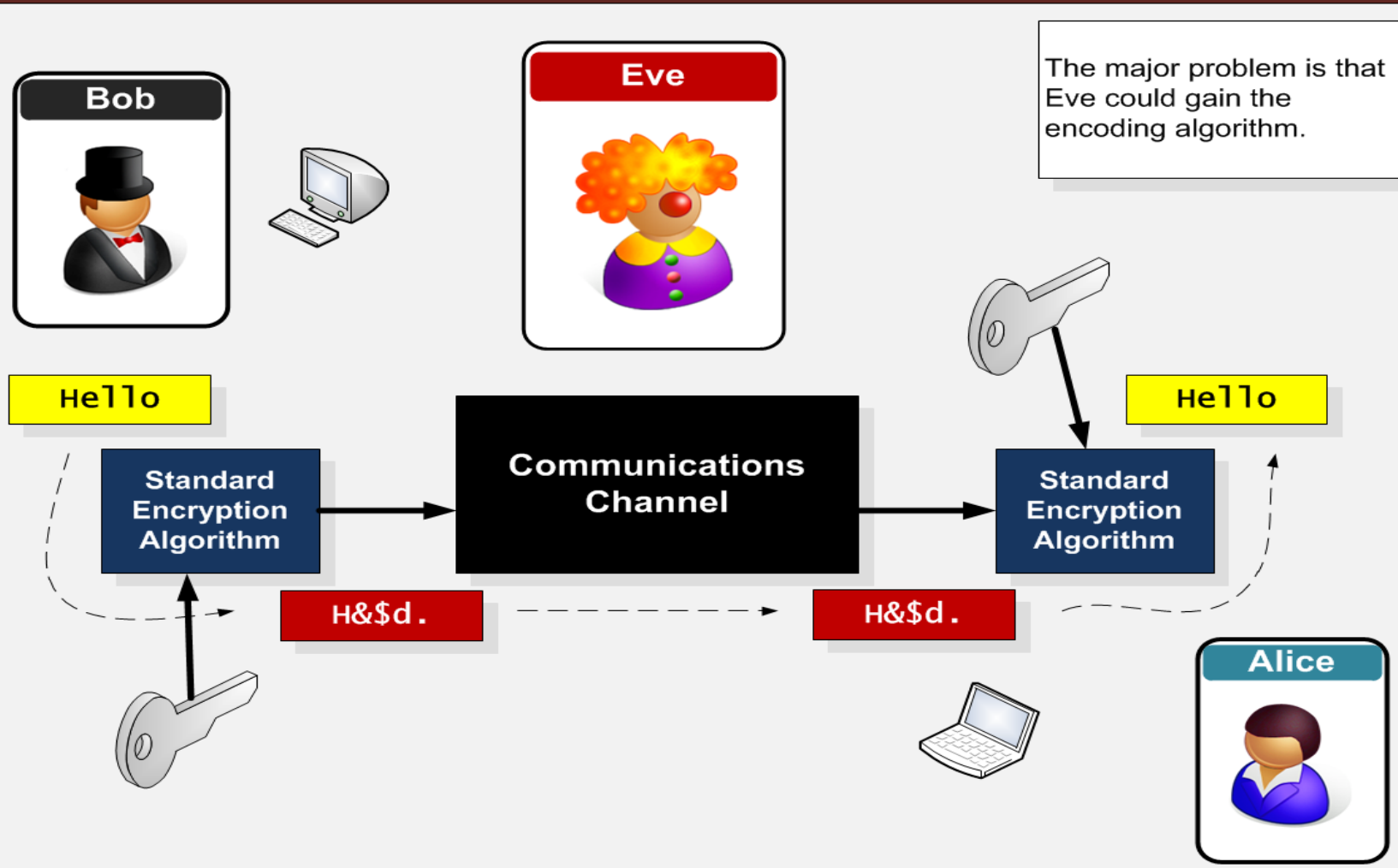
Alice

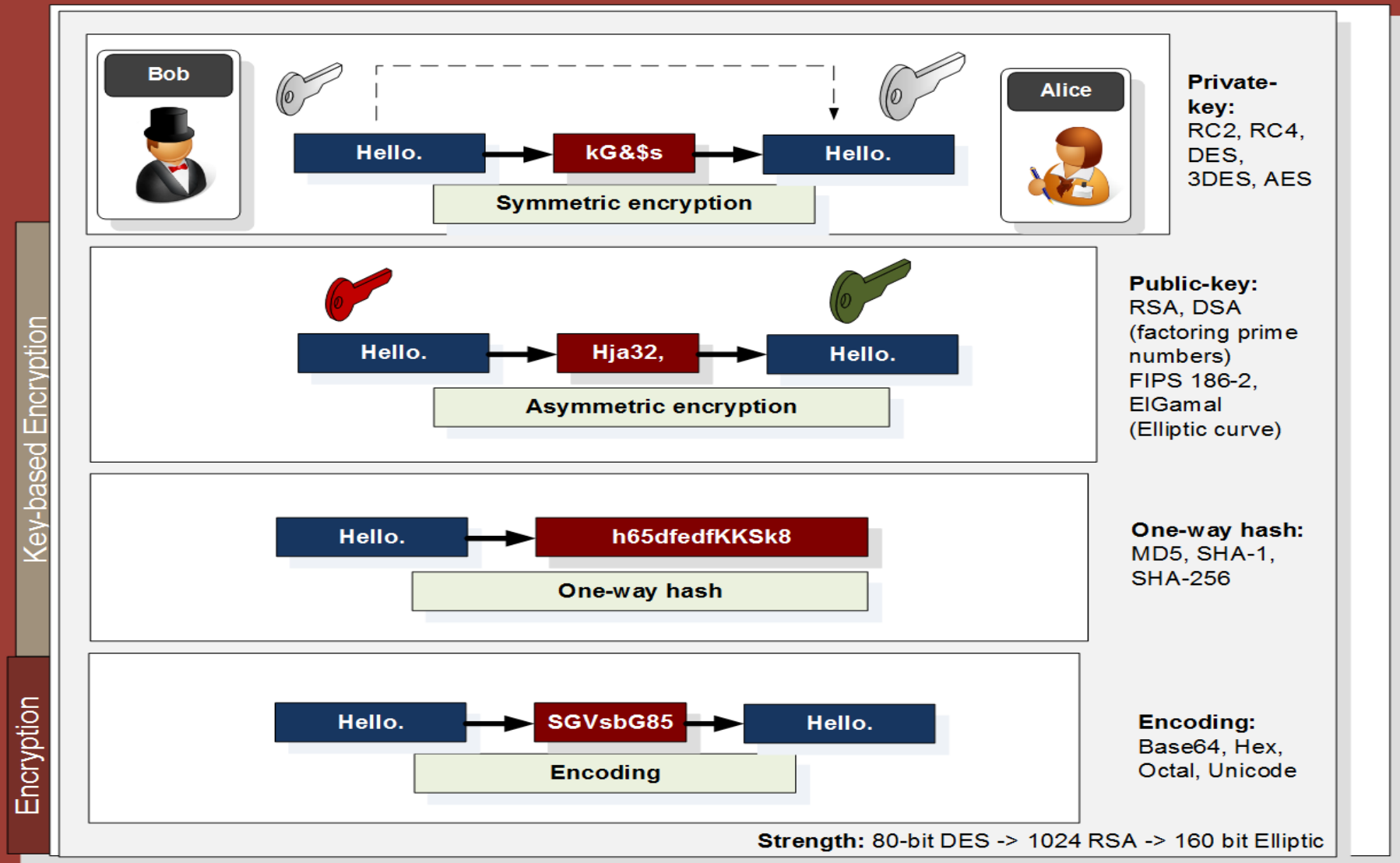
John

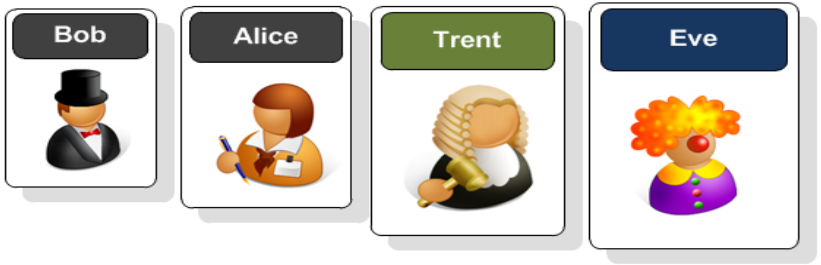
Trusted third party

Trent

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.

security level	volume of water to bring to a boil	bit-lengths: symmetric key	bit-lengths: cryptographic hash	bit-lengths: 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

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

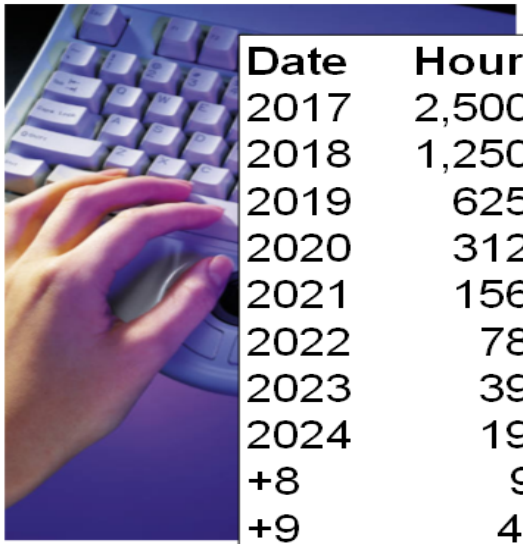
- It is important to understand the length of time that a message takes to crack as it may need to be secret for a certain time period.

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.

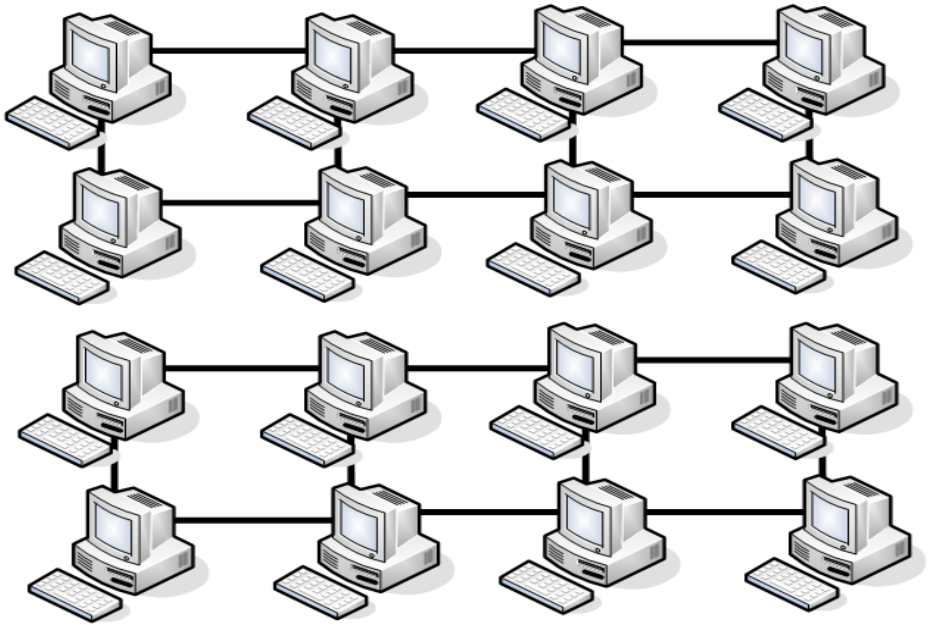
56-bit DES:
Developed
1975
30 years ago!
… now easily
crackable

Author: Prof Bill Buchanan

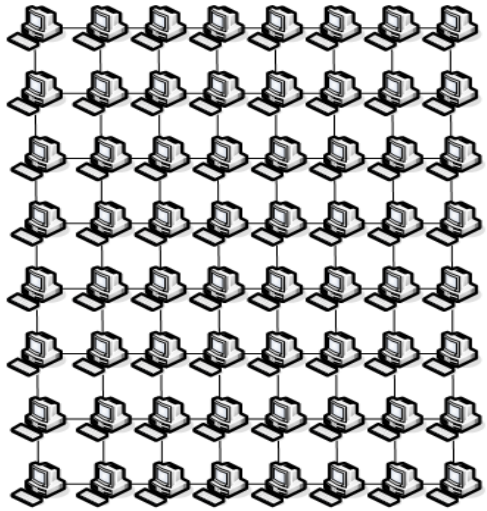
Parallel processing

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

2x1 =2 element array



4x4 = 16 element array

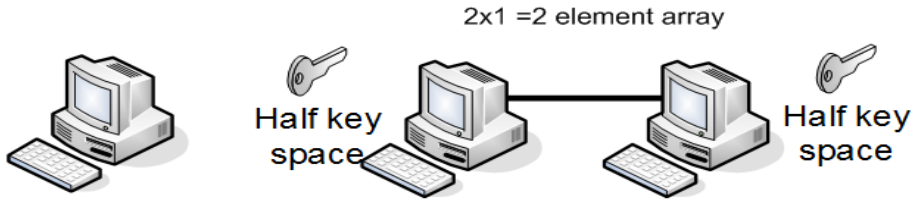


16x16 = 256 element array

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				

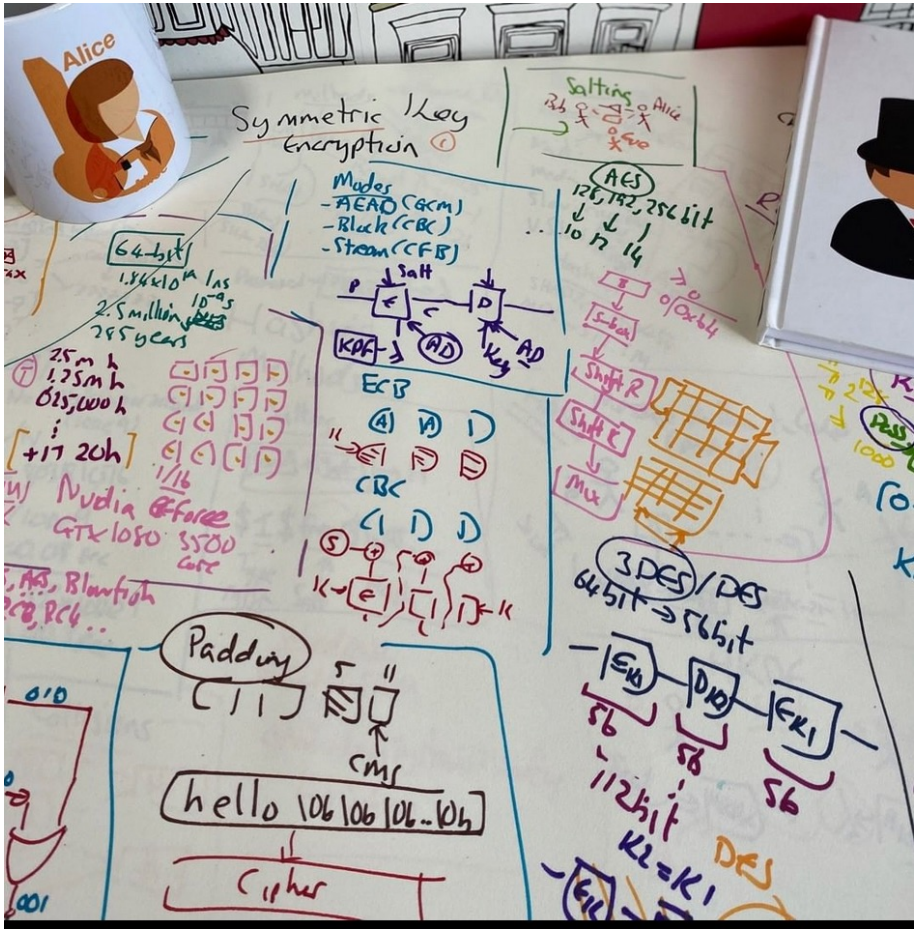
16x16 = 256 element array

4x4 = 16 element array

Author: Prof Bill Buchanan

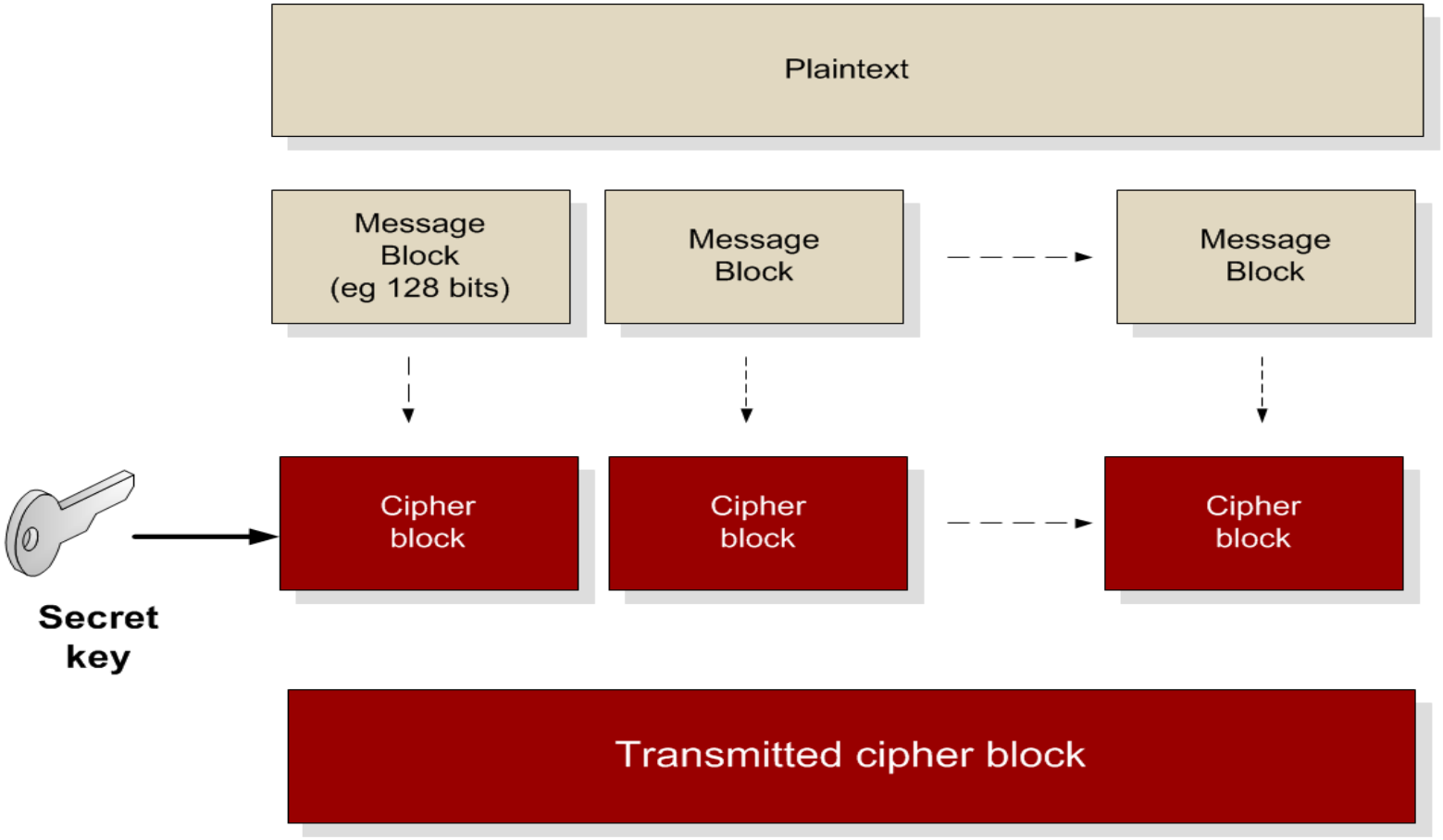
Symmetric Key

Basics
Block or Stream?
Secret Key Methods
Salting
AES
Key Entropy



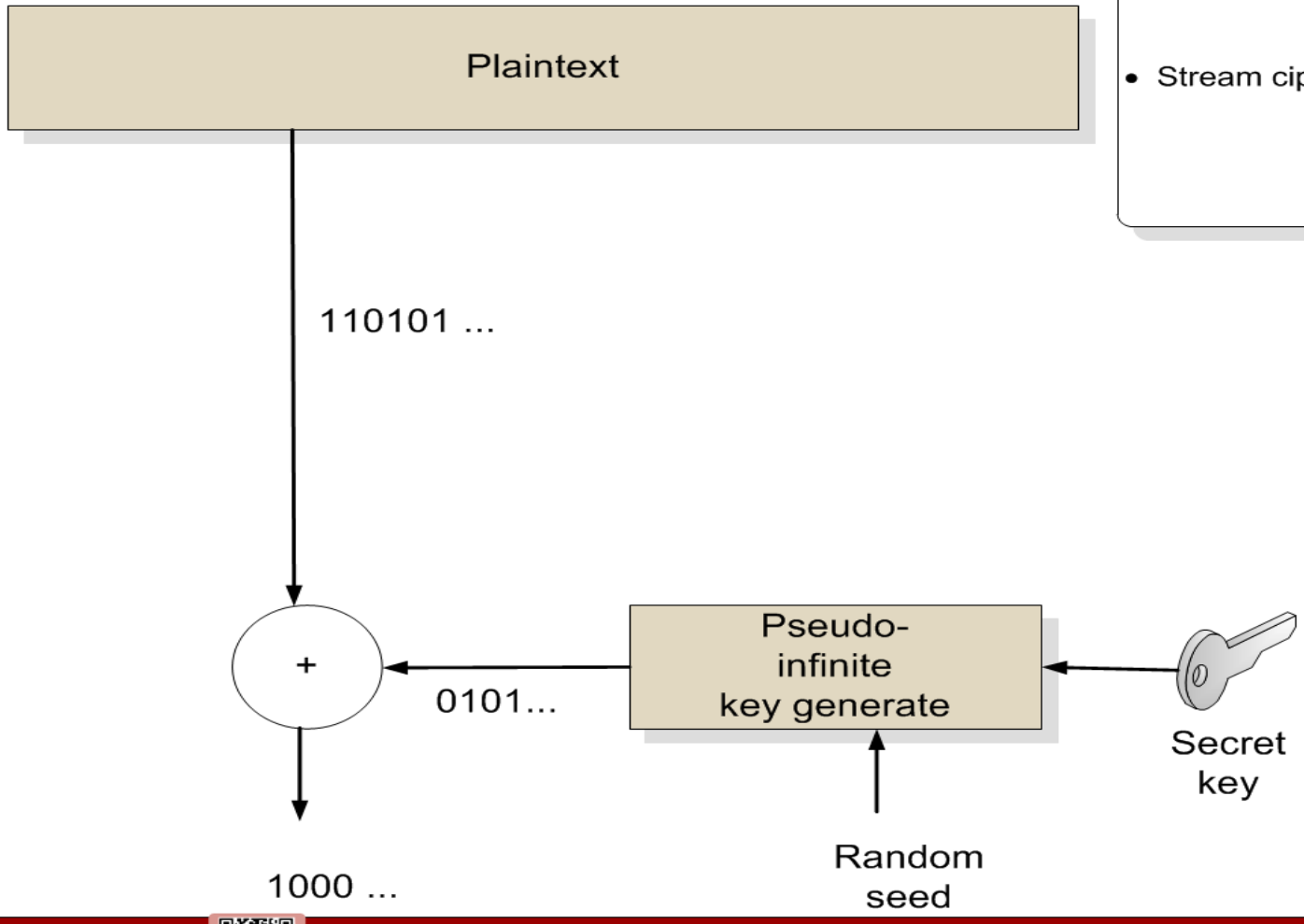
Prof Bill Buchanan OBE

https://asecuritysite.com/symmetric

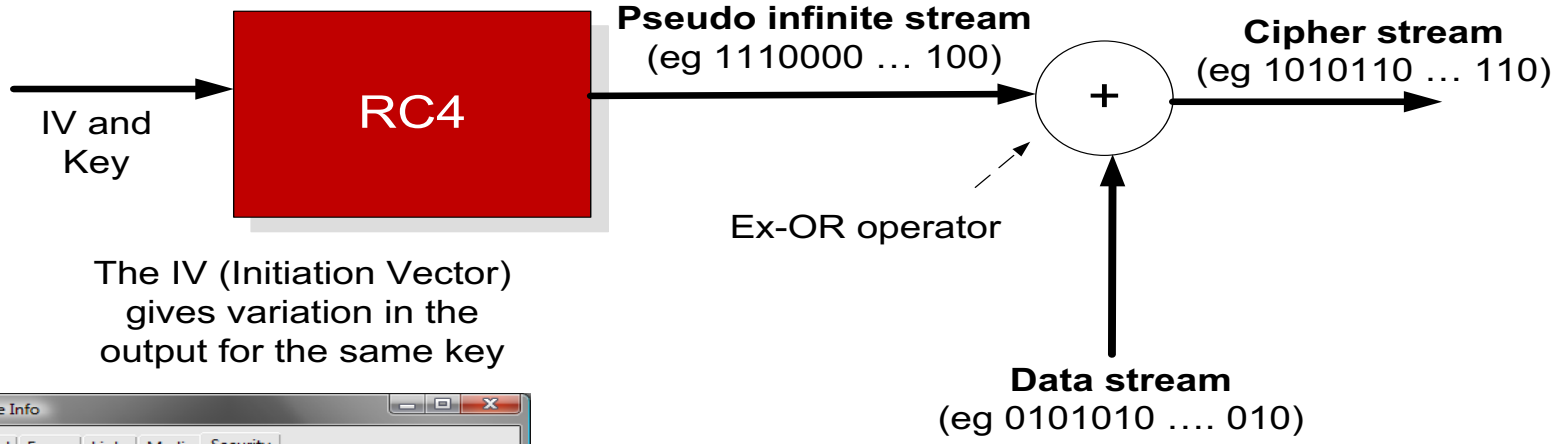

Plaintext
Message Block (eg 128 bits)
Message Block
Message Block
Cipher block
Cipher block
Cipher block
Secret key
Transmitted cipher block

Stream cipher

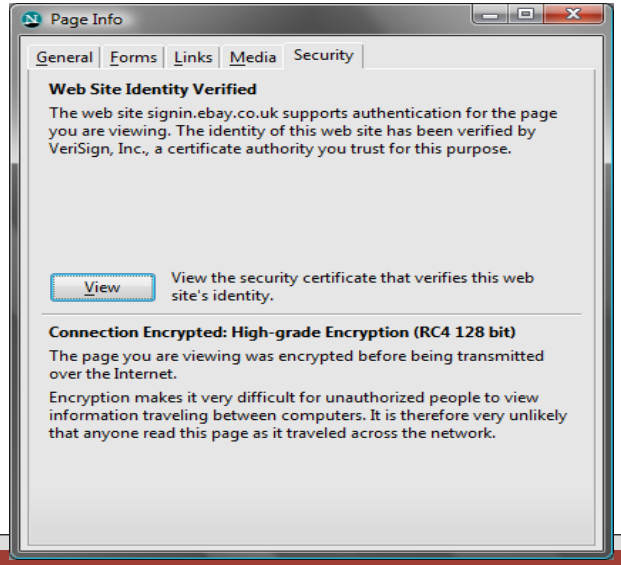
- Stream cipher (RC4)



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): 68656c6c6f80000000000000000000
  Cipher (ECB): 731abffc2e3b2c2b5caa9ca2339344f9
- After padding (ZeroLen): 68656c6c6f0000000000000000000a
  Cipher (ECB): d28e2f7e8e44e068732b292bde444245
- After padding (Null): 68656c6c6f00000000000000000000
  Cipher (ECB): 44479742246045 3d95856eb2a1520ece
- After padding (Space): 68656c6c6f20202020202020202020
  Cipher (ECB): 44479742246045 3d95856eb2a1520ece
- After padding (Random): 68656c6c6*ffc6ecfd884a38798d62*a**0a**
  Cipher (ECB): c2c88b4364d2c2dc6f2cac9ab73c995d

Symmetric Key

Basics

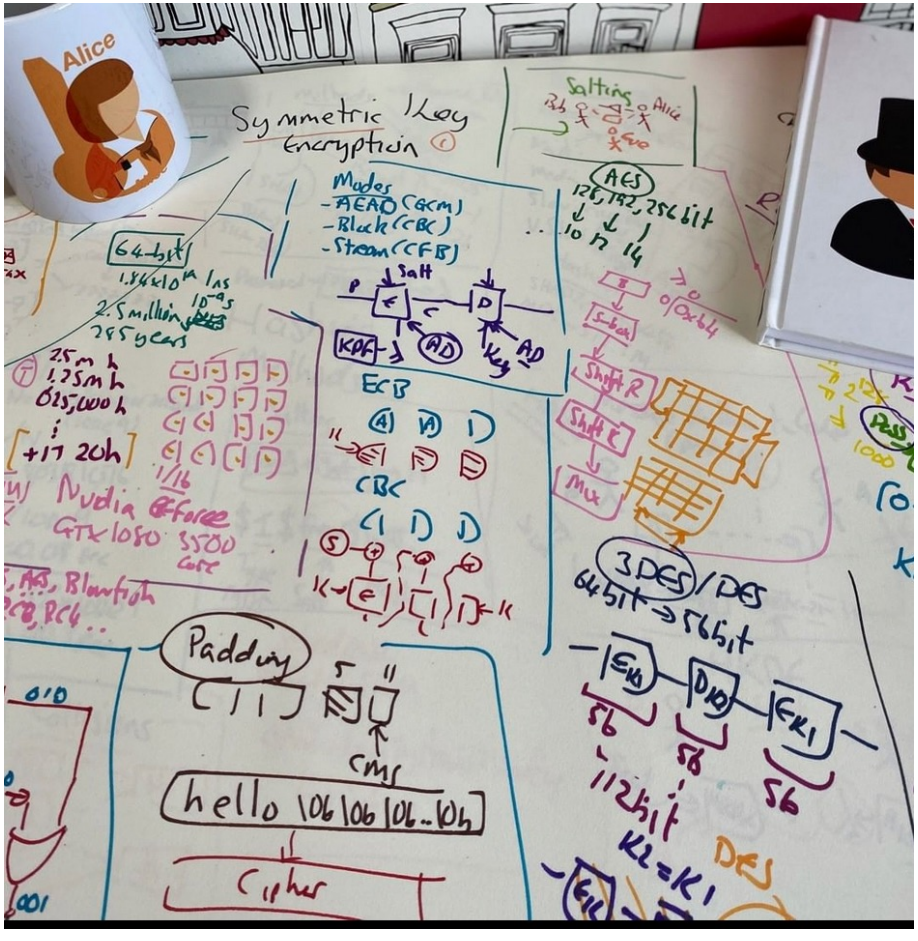
Block or Stream?

Secret Key Methods

Salting

AES

Key Entropy



Prof Bill Buchanan OBE

https://asecuritysite.com/symmetric

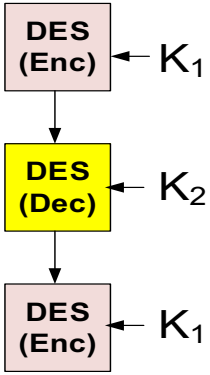
Encryption

Private key

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).

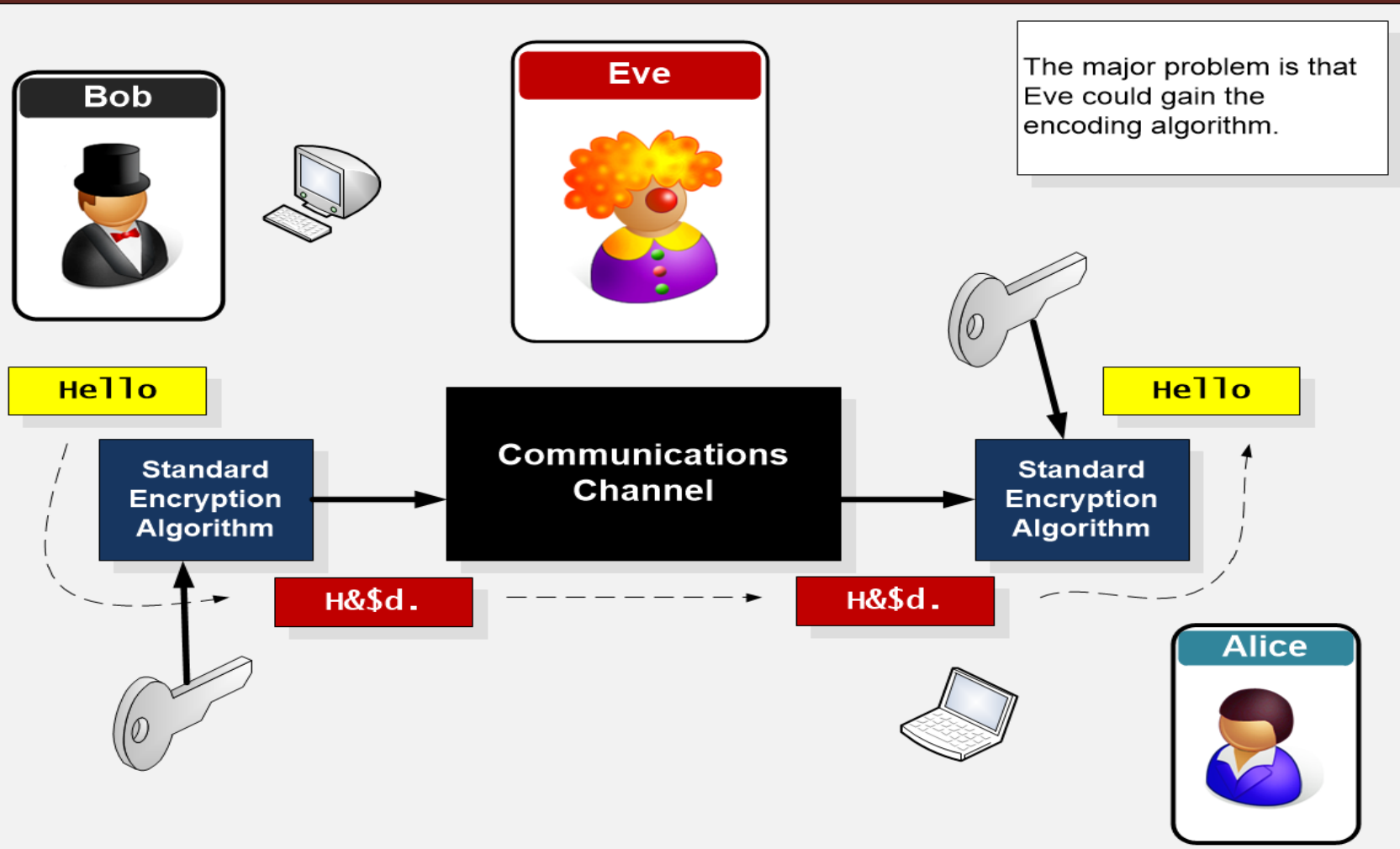
http://asecuritysite.com/encryption/threedes

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

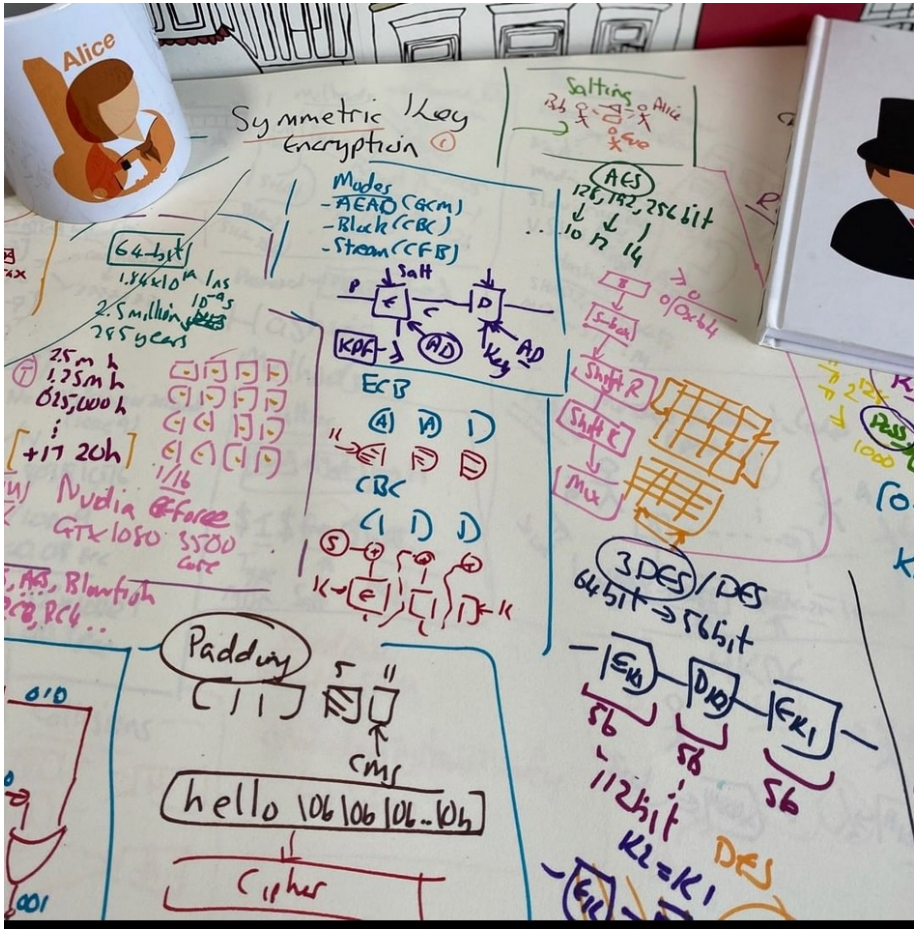

Bob
Eve
The major problem is that Eve could gain the encoding algorithm.
Hello
Standard Encryption Algorithm
Communications Channel
Standard Encryption Algorithm
Hello
H&$d.
H&$d.
Alice

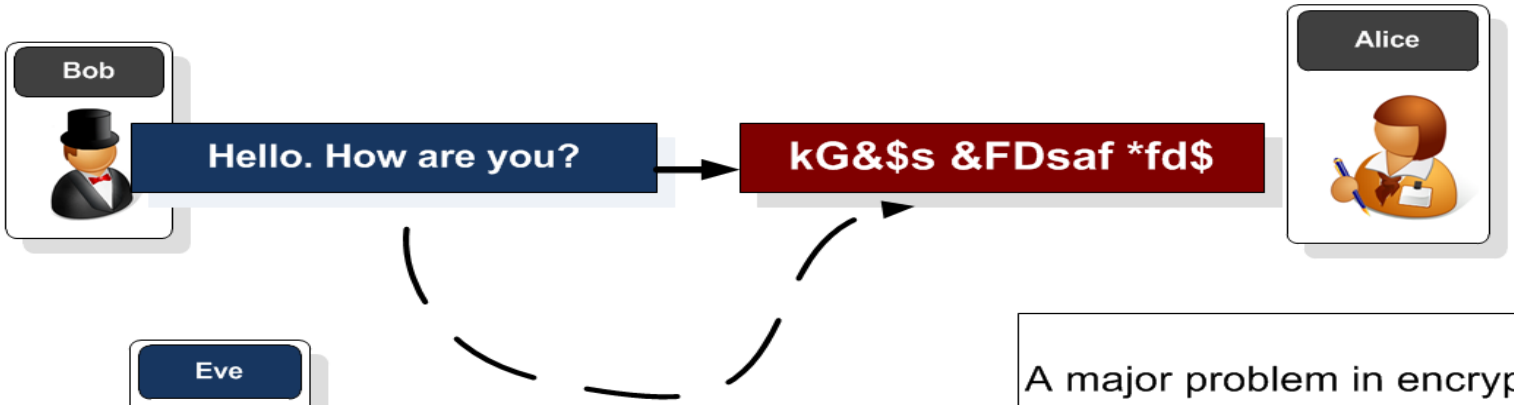
Symmetric Key

Basics
Block or Stream?
Secret Key Methods
Salting
AES
Key Entropy

Prof Bill Buchanan

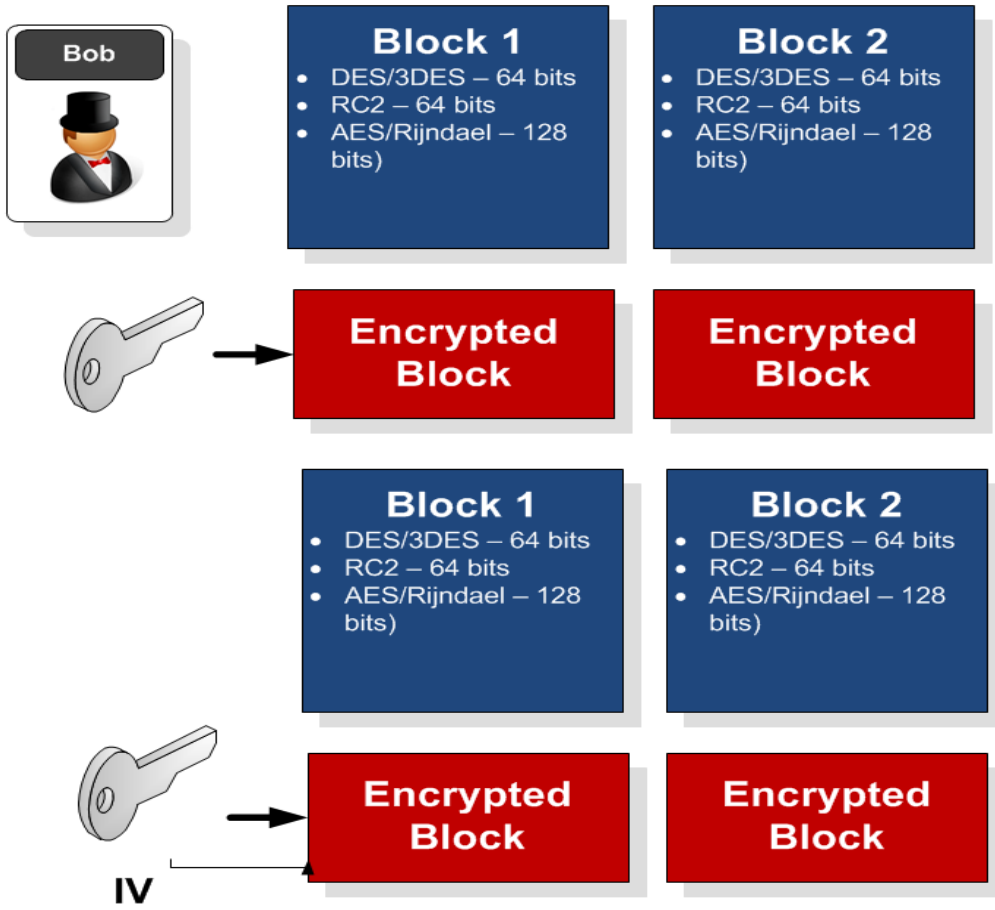
https://asecuritysite.com/symmetric





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.

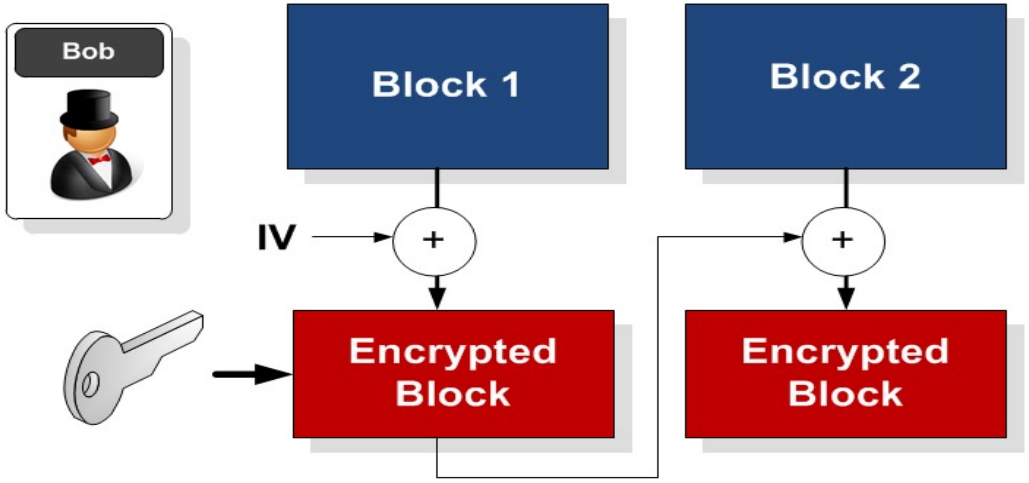
The solution is to add **salt** to the encryption key, as that it changes its operation from block-to-block (for block encryption) or data frame-to-data frame (for stream encryption)



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

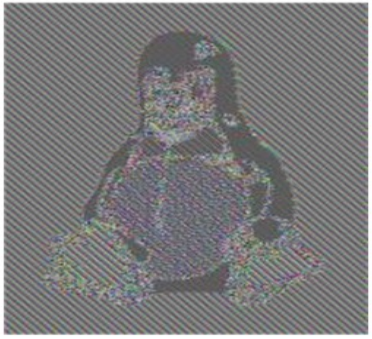


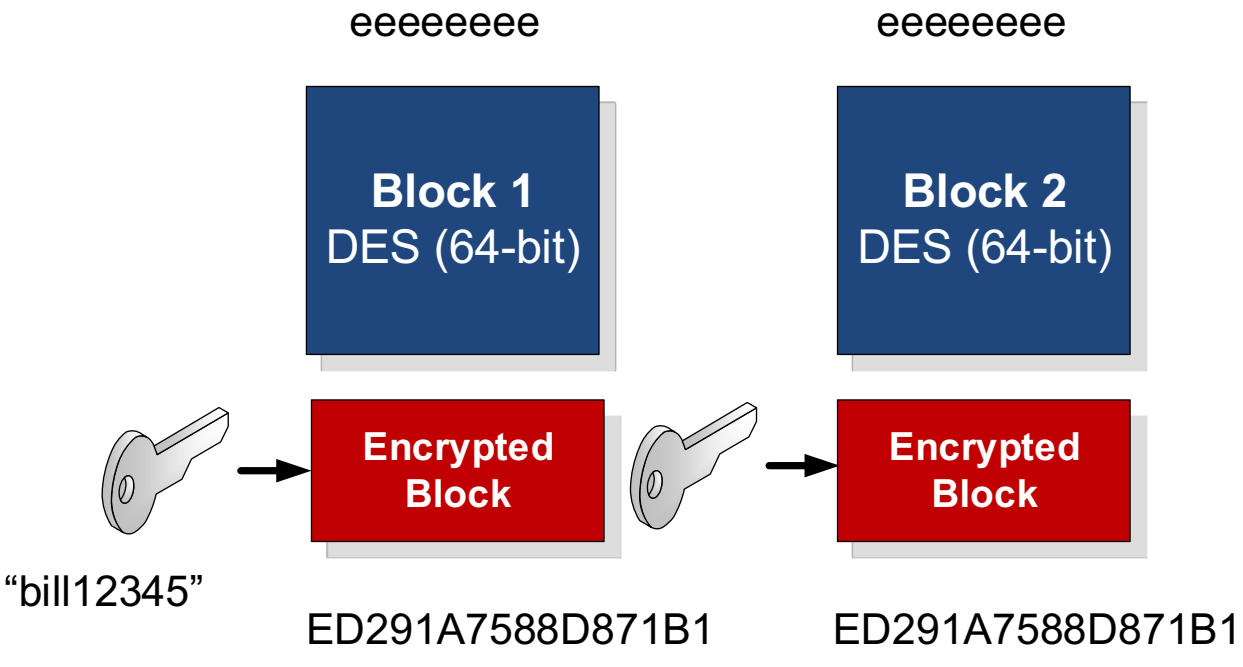
Image with AES using ECB

Image with AES using CBC

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

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

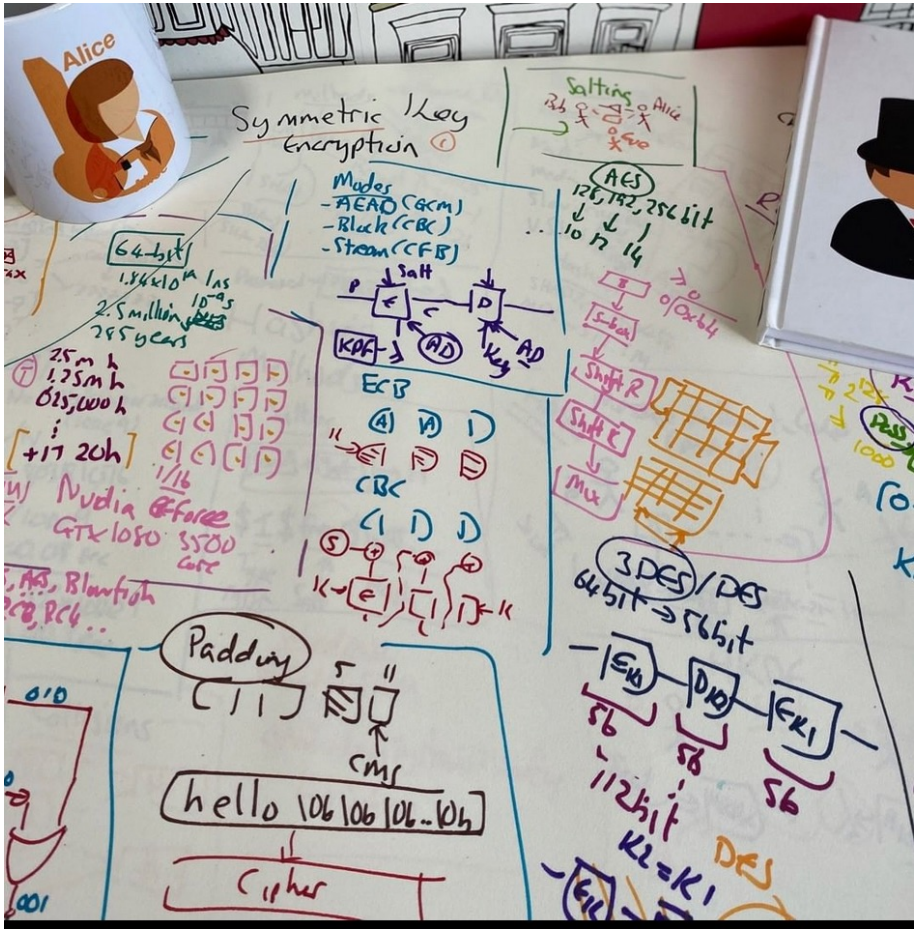
[eeeeeeee] [eeeeeeee] [eeeeeeeee][eeeeeeee] [eeeeeeee] [eeeeeeee][eeeeee <PADDING>]



ED291A7588D871B1ED291A7588D871B1ED291A7588D
871B1ED291A7588D871B1ED291A7588D871B1ED291A
7588D871B18D6DF6795DDEDACD

Symmetric Key

Basics
Block or Stream?
Secret Key Methods
Salting
AES
Key Entropy



Prof Bill Buchanan

https://asecuritysite.com/symmetric

Key Entropy

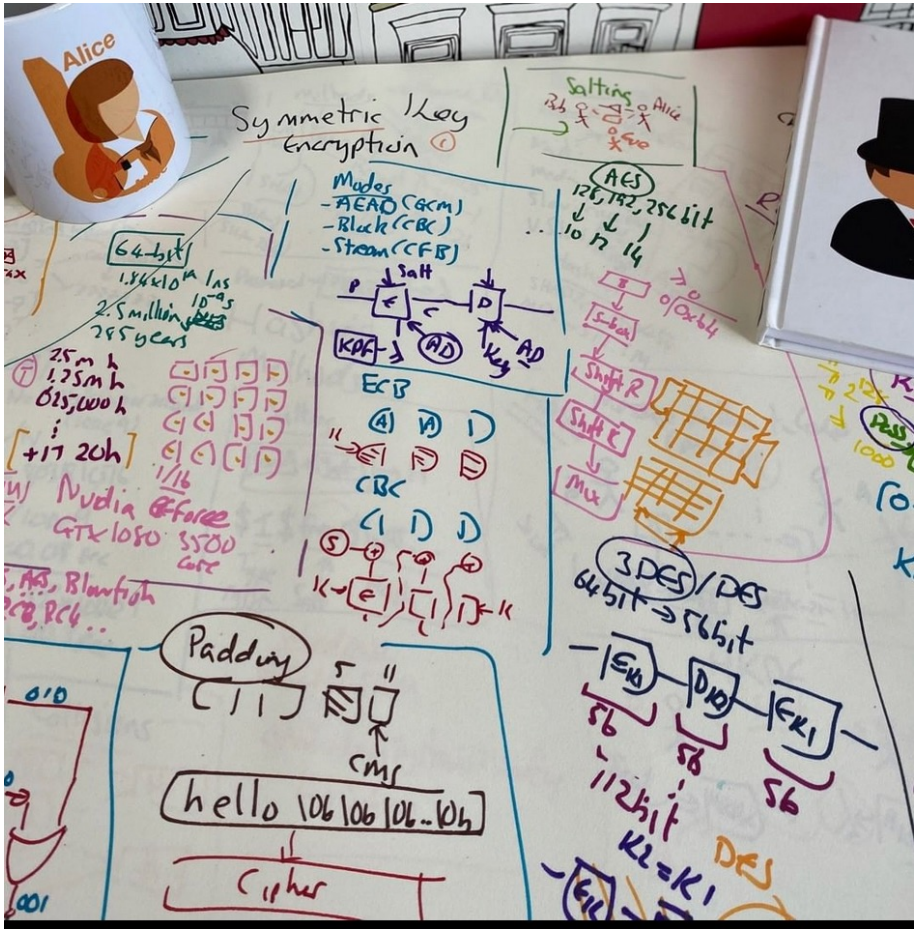
$$Key\ Entropy = log_2(Phrases) = \frac{log_{10}(Phrases)}{log_{10}(2)}$$

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

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

Symmetric Key

Basics
Block or Stream?
Secret Key Methods
Salting
AES
Key Entropy



Prof Bill Buchanan OBE

https://asecuritysite.com/symmetric

Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.

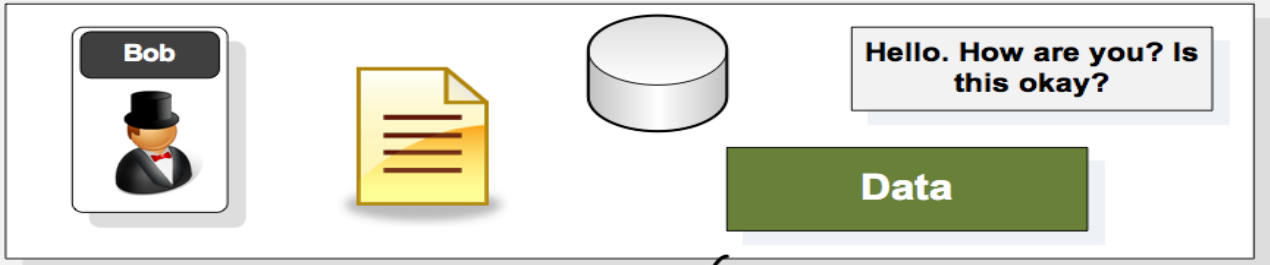
Prof Bill Buchanan OBE
https://asecuritysite.com/hash

Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.
Secret Shares.

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

How do we get a finger-print for data?



With a fingerprint we can hopefully tell if Eve has modified any of the data

Solved by Prof Ron Rivest with the MD5 hash signature.

hello	------>	XUFAKrxLKna5cZ2REBfFkg
Hello	------>	ixqZU8RhEpaoJ6v4xHgE1w
Hello. How are you?	------>	CysDE5j+ZOUbCYZtTdsFiw
Napier	------>	j4NXH5Mkrk4j13N1MFXHtg

Base-64

hello	->	5D41402ABC4B2A76B9719D911017C592
Hello	->	8B1A9953C4611296A827ABF8C47804D7
Hello. How are you?	->	CC708153987BF9AD833BEBF90239BF0F
Napier	->	8F83571F9324AE4E23D773753055C7B6

Hex

hello	→	qvTGHdzF6KLavt4PO0gs2a6pQ00=
Hello	→	9/+ei3uy4Jtwk1pdeF4MxdnQq/A=
Hello. How are you?	→	Puh2Am76bhjqE5lbTWtwsqbdFC8=
Napier	→	v4GxNaVod2b09GR2Tqw4yopOuro=

Base-64

hello	→	AAF4C61DDCC5E8A2DABEDE0F3B482CD9AEA9434D
Hello	→	F7FF9E8B7BB2E09B70935A5D785E0CC5D9D0ABF0
Hello. How are you?	→	3EE876026EFA6E18EA13995B4D6B70B2A6DD142F
Napier	→	BF81B135A5687766F4F464764EAC38CA8A4EBABA

Hex

Message Hash

Authentication

Security and mobility are two of the most important issues on the Internet, as they will allow users to secure their data transmissions, and also break their link with physical connections.

F94FBED3DAE05D223E6B963B9076C4EC

+U++09rgXSI+a5Y7kHbE7A==

Base-64

Security and mobility are two of the mast important issues on the Internet, as they will allow users to secure their data transmissions, and also break their link their physical connections.

8A8BDC3FF80A01917D0432800201CFBF

iovcP/gKAZF9BDKAAgHPvw==

Hex

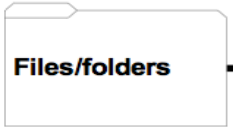
Message Hash

Authentication

OpenSSL

```
root@kali:~# echo -n “hello” | openssl md5
(stdin)= 5d41402abc4b2a76b9719d911017c592

root@kali:~# echo -n “hello” | md5sum
5d41402abc4b2a76b9719d911017c592  -

root@kali:~# openssl md5 pw
MD5(pw)= 859b6a9be3b45262c4414bd1696ba91b

root@kali:~# md5sum pw
859b6a9be3b45262c4414bd1696ba91b  pw

Hash methods supported:
md2          md4          md5          rmd160       sha
sha1
```

Author: Prof Bill Buchanan



Hashing Algorithm (MD5) - 128 bit signature

Hash signature

- Hash signatures are used to gain a signature for files, so that they can be checked if they have been changed.

```
[Path] / filename                    MD 5 sum
-------------------------------------------------------------------
[C:\Windows\System32\]
12520437.cpx                          0a0feb9eb28bde8cd835716343b03b14
12520850.cpx                         d 69ae057cd82d04ee7d311809abefb2a
8point1.wav                          beab 165fa58ec5253185f32e124685d5
aaclient.dll                         ad 45dedfdcf69a28cbaf6a2ca84b5f1e
AC3ACM.acm                            59683d1e4cd0b1ad6ae32e1d627ae25f
Ac3audio.ax                           4b87d889edf278e5fa223734a9bbe79a
ac3filter.cpl                         10b27174d46094984e7a05f3c36acd2a
accessibilitycpl.dll                 ac 4cecc86eeb8e1cc2e9fe022cff3ac1
ACCTRES.dll                           58f57f2f2133a2a77607c8ccc9a30f73
acledit.dll                           0bcee3f36752213d1b09d18e69383898
  . . .
ZSHP1020.CHM                         c 671ec
ZSHP1020.EXE                          96e45a
ZSHP1020.HLP                         a 07693
ZSPOOL.DLL                           fae 332
ZTAG.DLL                              7ca836
ZTAG32.DLL                            27b026
```

```
[Path] / filename                    MD 5 sum
-------------------------------------------------------------------
[C:\Windows\system32\]
12520437.cpx                         Cg/rnrKL3ozYNXFjQ7A7FA==
12520850.cpx                          1prgV82C0E7n0xGAmr77Kg==
8point1.wav                          vqsWX 6WOxSUxhfMuEkaF1Q==
aaclient.dll                         rUXe 39z2mijLr2osqEtfHg==
AC3ACM.acm                           WWg 9HkzQsa1q4y4dYnriXw==
Ac3audio.ax                          S 4fYie3yeOX6Ijc0qbvnmg==
```

Message Hash

Authentication

MD5 hash algorithm

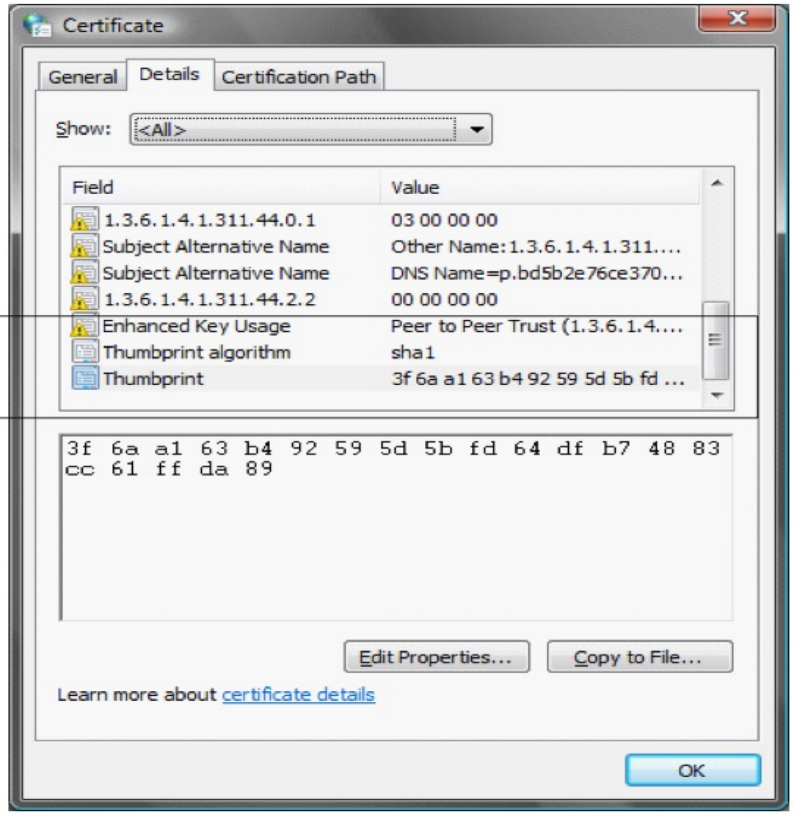
Files/folders

Hashing Algorithm (MD5) - 128 bit signature

Hash signature

- Hash signatures are used to identify that a file/certificate has not been changed.

Bob



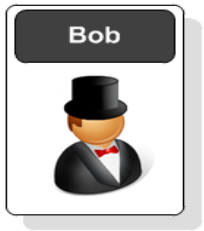
The digital certificate has an SHA-1 hash thumbprint (3f6a...89) which will be checked, and if the thumbprint is different, the certificate will be invalid.

Author: Prof Bill Buchanan

One-way hash

- Hashes are used for digital fingerprints (see the next unit) and for secure password storage.
- Typical methods are NT hash, MD4, MD5, and SHA-1.

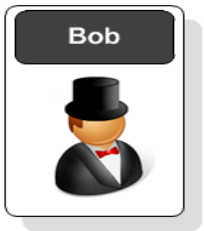
Windows login/ authentication



mypass → NT hash (MD4) → fa1bfa14fa13fa12fa10fa1ffa14fa12

NT-password hash for Windows NT, XP and Vista

Cisco password storage (MD5)



mypass → 

```
# config t
(config)# enable secret test

Current configuration : 542 bytes
!
version 12.1
no service single-slot-reload-enable
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Router
!
enable secret 5 $1$/Nwk$knsEQYxZVenGjWOGj/TGk0
```

MD5 encoded password →

One-way hash

- Hashing suffers from dictionary attacks, thus it is important that any passwords are not standard words, such as to change **password** for **pA55wOrd**.

Windows login/ authentication

mypass → NT hash (MD4) → fa1bfa14fa13fa12fa10fa1ffa14fa12

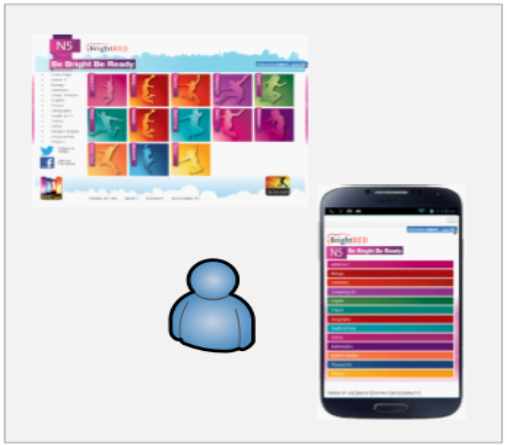
NT-password hash for Windows NT, XP and Vista

Hashing suffers from **dictionary attacks** where the signatures of well know words are stored in a table, and the intruders does a lookup on this

mypast	effahd13fa12fa10fgffa1ffa14fa144
mypass	fa1bfa14fa13fa12fa10fa1ffa14fa12
mypose	ff12189043210954defff0123444512d
test1	aabbfce023215546dfeddd0101001cd

Risk 4: One Password Fits All

Edinburgh Napier UNIVERSITY



Adobe

150 million accounts compromised

#	Count	Ciphertext	Plaintext
1.	1911938	EQ7fIpT7i/Q=	123456
2.	446162	j9p+HwtWWT86aMjgZFLzYg==	123456789
3.	345834	L8qbAD3jl3jioxG6CatHBw==	password
4.	211659	BB4e6X+b2xLioxG6CatHBw==	adobe123
5.	201580	j9p+HwtWWT/ioxG6CatHBw==	12345678
6.	130832	5djv7ZCI2ws=	qwerty
7.	124253	dQi0asWPYvQ=	1234567
8.	113884	7LqYzKVeq8I=	111111
9.	83411	PMDTbP0LZxu03SwrFUvYGA==	photoshop
10.	82694	e6MPXQ5G6a8=	123123

TJ Maxx / Marshalls

47 million accounts

LinkedIn

6.5 million accounts (June 2013)

Windows Live · Facebook · Twitter · YouTube

One account hack … leads to others

1 million accounts – in plain text. 77 million compromised

Dropbox compromised 2013

citigroup

200,000 client accounts

Brute Force - How many hash codes?

- 7 digit password with [a-z] … how many?
  - Ans:
  - Time to crack - 100 billion per second:
- 7 digit with [a-zA-z] … how many?
  - Ans:
  - Time to crack – 100 billion per second:
- 7 digit with [a-zA-z!@#$%^&*()] … how many?
  - Ans:
  - Time to crack – 100 billion per second:

Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.
Secret Shares.

Prof Bill Buchanan OBE
http://asecuritysite.com/hash

LM Hash

LM Hash. LM Hash is used in many version of Windows to store user passwords that are fewer than 15 characters long.

SHA-3

SHA-3. SHA-3 was known as Keccak and is a hash function designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. MD5 and SHA-0 have been shown to be susceptible to attacks, along with theoretical attacks on SHA-1. NIST thus defined there was a need for a new hashing method which did not use the existing methods for hashing, and setup a competition for competing algorithms. In October 2012, Keccak won the NIST hash function competition, and is proposed as the SHA-3 standard.

Tiger

Bcrypt

Bcrypt. This creates a hash value which has salt.

RIPEMD

RIPEMD (RACE Integrity Primitives Evaluation Message Digest) and GOST. RIPEM160. RIPEMD is a 128-bit, 160-bit, 256-bit or 320-bit cryptographic hash function, and was created by Hans Dobbertin, Antoon Bosselaers and Bart Preneel. It is used on TrueCrypt, and is open source. The 160-bit version is seen as an alternative to SHA-1, and is part of ISO/IEC 10118

Tiger. Tiger is a 192-bit hash function, and was designed by Ross Anderson and Eli Biham in 1995. It is often used by clients within Gnutella file sharing networks, and does not suffer from known attacks on MD5 and SHA-0/SHA-1. Tiger2 is an addition, in which the message is padded with a byte of 0x80 (in a similar way to MD4, MD5 and SHA), whereas in Tiger it is 0x01. Otherwise the two methods are the same in their operation.

Author: Prof Bill Buchanan

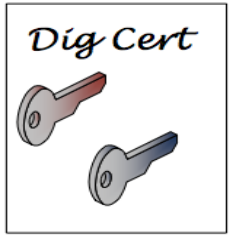
Murmur

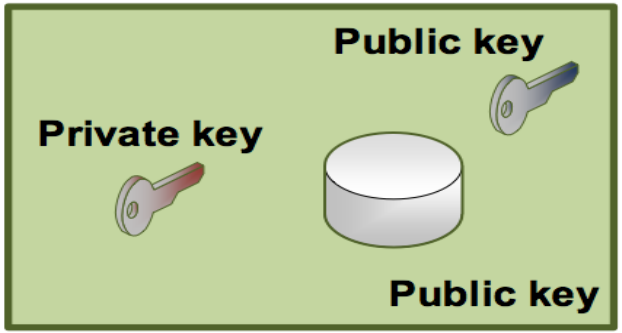
While hashing methods such as MD5 and SHA-1 use cryto methods, the Murmur and FNV hashes uses a non-cryptographic hash function. The Murmur hash, designed by Austin Appleby, uses a non-cryptographic hash function. This can be used for general hash-based lookups. It has a good performance compared with other hashing methods, and generally provide a good balance between performance and CPU utilization. Also it performs well in terms of hash collisions.

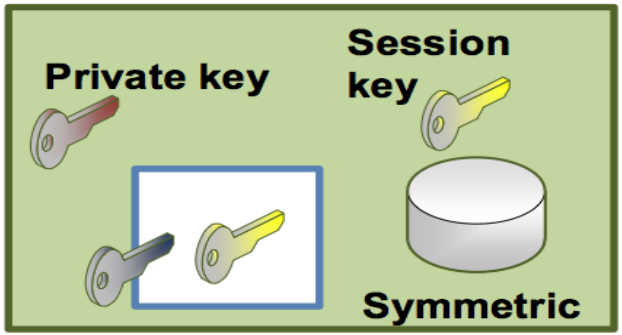
FNV

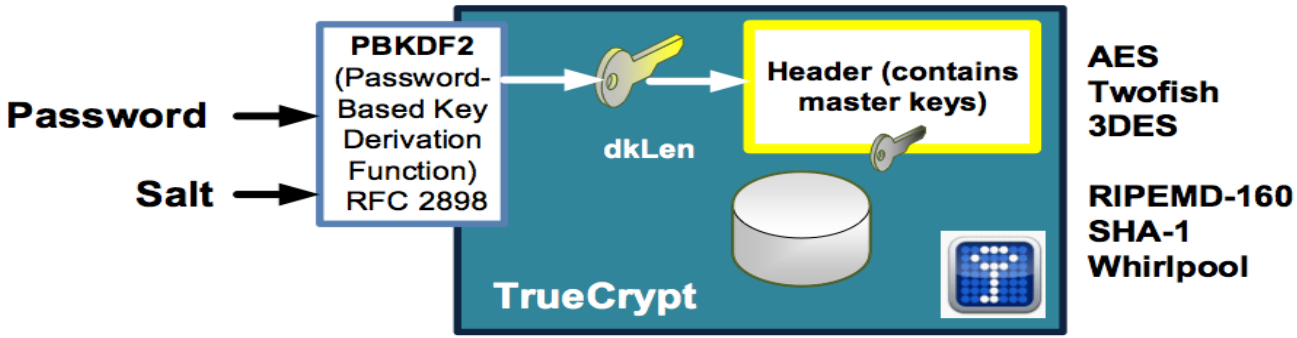
FNV (Fowler–Noll–Vo) is a 64-bit non-cryptographic hash function developed by Glenn Fowler, Landon Curt Noll, and Phong Vo. There are two main versions, of which 1a is the most up-to-date version.

Author: Prof Bill Buchanan









AES
Twofish
3DES

RIPEMD-160
SHA-1
Whirlpool

DK = PBKDF2(PRF, Password, Salt, c, dkLen)
DK = PBKDF2(HMAC−SHA1, passphrase, ssid, 4096, 256)

Encrypting disks

Chapter 3: Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.
Secret Shares.

Prof Bill Buchanan OBE
http://asecuritysite.com/hash

Adding salt

- Salt increases the range of the possible signatures

mypass → NT hash (MD4) → fa1bfa14fa13fa12fa10fa1ffa14fa12

NT-password hash for Windows NT, XP and Vista

Salt increase the range of the signatures

mypast

effahd13fa12fa10fgffa1ffa14fa144

caaahdd3fa12ccfae342345500011aff

Ddde43296945031040301 0d0ae000100

password

1fred$bATAk8UUH/IDAp9sd6IUv/

1

fred

bATAk8UUH/IDAp9sd6IUv/

password

bATAk8UUH/IDAp9sd6IUv/

fred

```
C:\openssl>openssl passwd -1 -salt fred password
$1$fred$bATAk8UUH/IDAp9sd6IUv/
```

Author: Prof Bill Buchanan


```
# cat /etc/shadow
root:$1$Etg2ExUZ$F9NTP7omafhKIlqaBMqng1:15651:0:99999:7:::
# openssl passwd -1 -salt Etg2ExUZ redhat
$1$Etg2ExUZ$F9NTP7omafhKIlqaBMqng1
```

```
$ openssl version
OpenSSL 1.0.1f 6 Jan 2014

$ openssl dgst -md5  file
MD5(file)= b1946ac92492d2347c6235b4d2611184

$ openssl genrsa -out mykey.pem 1024
Generating RSA private key, 1024 bit long modulus
.....................................................................++++++
...++++++
e is 65537 (0x10001

$ openssl rsa -in mykey.pem -pubout > mykey.pub
writing RSA key

$ cat mykey.pub
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDXv9HSFkpM+ZoOQcpdHBZiUwX8
EzIKm0nsgjc5ZTYVaF9CMLtmKoTzep7aQX9o9nKepFt1kQ73Ta9vOPd6CX61/cgY
Xy2tShw0imrtFaVDFjX+7kLmc0uWbFFCoZMtJxIaXaa9SV2kARxOCTJ2uOjRTCCe
XU09IJGHnIhSNJeIJQIDAQAB
-----END PUBLIC KEY-----

$ cat /etc/shadow
root:$1$Etg2ExUZ$F9NTP7omafhKIlqaBMqng1:15651:0:99999:7:::

$ openssl passwd -1 -salt Etg2ExUZ redhat
$1$Etg2ExUZ$F9NTP7omafhKIlqaBMqng1
```

Chapter 3: Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.
Secret Shares.

Prof Bill Buchanan OBE

http://asecuritysite.com/hash

A major factor with hash signatures is:

- **Collision.** This is where another match is found, no matter the similarity of the original message. This can be defined as a **Collision attack**.
- **Similar context**. This is where part of the message has some significance to the original, and generates the same hash signature. The can be defined as a Pre-image attack.
- **Full context**. This is where an alternative message is created with the same hash signature, and has a direct relation to the original message. This is an extension to a Pre-image attack.

In 2006 it was shown that MD5 can produce collision within less than a minute.

A 50% probability of a collision is:

$$\sqrt{N(signatures)} = \sqrt{2^n} = 2^{\frac{n}{2}}$$

where n is the number of bits in the signature. For example, for MD5 (128-bit) the number of operations that would be required for a better-than-50% chance of a collision is:

$$2^{64}$$

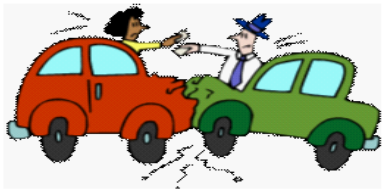
Note, in 2006, for SHA-1 the best time has been 18 hours

d131dd02c5e6eec4693d9a0698aff95c
2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325 71415a
085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6
dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1e
c69821bcb6a8839396f9652b6ff72a70

d131dd02c5e6eec4693d9a0698aff95c
2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a
085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6
dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b6708280d1e
c69821bcb6a8839396f965ab6ff72a70

The MD5 signature gives the same result

79054025255FB1A26E4BC422AEF54EB4



Nat McHugh

- 10 hours of computing on the Amazon GPU Cloud.
- Cost: 60 cents
- Used: Hashcat (on CUDA)
- Birthday attack: A group size of only 70 people results in a 99.9% chance of two people sharing the same birthday.
- M-bit output there are 2^m messages, and the same hash value would only require 2^(m/2) random messages. 18,446,744,073,709,551,616.

C:\openssl>openssl md5 hash01.jpg
MD5(hash01.jpg)= **e06723d4961a0a3f950e7786f3766338**

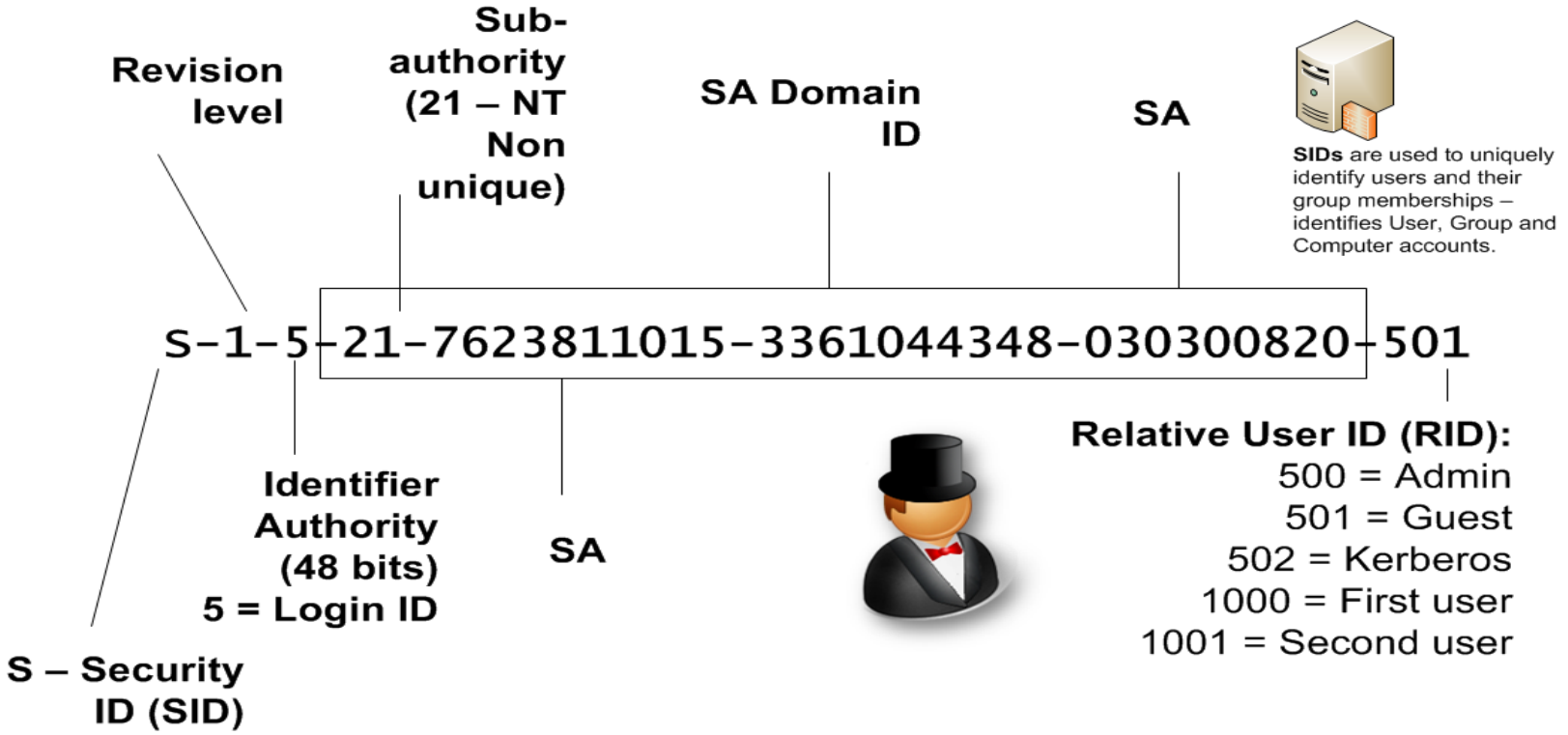
C:\openssl>openssl md5 hash02.jpg
MD5(hash02.jpg)= **e06723d4961a0a3f950e7786f3766338**

Chapter 3: Hashing

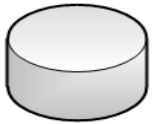
Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.
Secret Shares.

Prof Bill Buchanan OBE
http://asecuritysite.com/hash

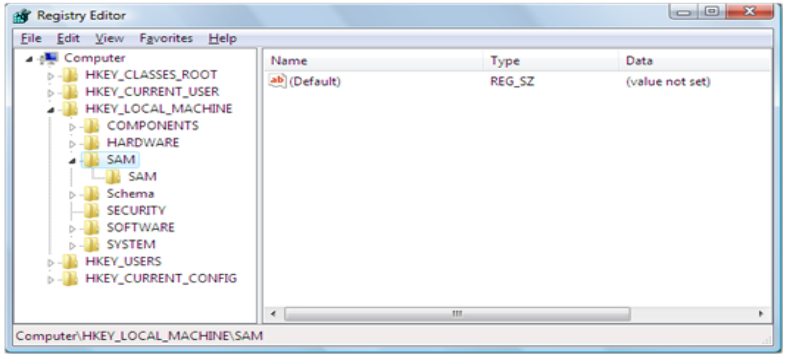
```
C:> user2sid \pluto guest
S-1-5-21-7623811015-3361044348-030300820-501
C:> sid2user 5 21 7623811015 3361044348 030300820 500
Name is Fred
Domain is PLUTO
```

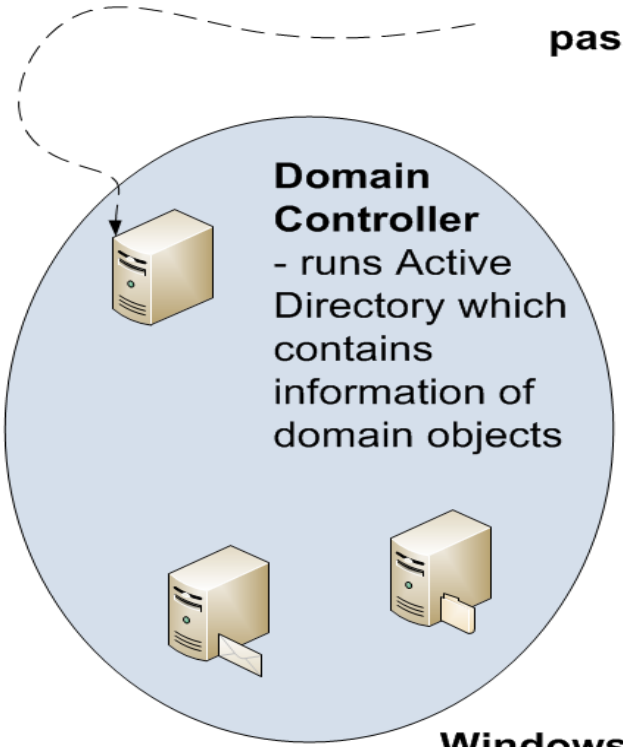


HKLM\SAM



SAM Database (stores usernames and passwords)



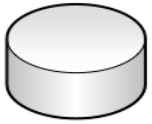


Local Authority Subsystem (Lsass) – Windows Security mechanism – Attached by Sasser Worm which exploited a buffer overflow

Responsible for local security policy

- Controls access.
- Managing password policies.
- User authentication.
- Audit messages.

SAM



Registry:
HKEY_LOCAL_MACHINE\SAM

- LM Hash (Windows XP, 2003)
- NTLMv2 (Windows 7, 8, etc) – connect to Active Directory
- NTLM (Windows 7, 8, etc) – No salt

```
C:\Windows\System32\config>dir
 Volume in drive C has no label.
 Volume Serial Number is A2B3-7C7A

 Directory of C:\Windows\System32\config
05-Oct-14  05:52 PM           262,144 SAM
05-Oct-14  05:56 PM           262,144 SECURITY
05-Oct-14  08:39 PM       149,946,368 SOFTWARE
05-Oct-14  08:40 PM        15,728,640 SYSTEM
```

- bkhive - dumps the syskey bootkey from a Windows system hive.
- samdump2 - dumps Windows 2k/NT/ XP/Vista password hashes.

hashme gives: FA-91-C4-FD-28-A2-D2-57-AA-D3-B4-35-B5-14-04-EE FF2A43841C84518A18795AB6E3C8A62E (NTLM)

napier gives: 12-B9-C5-4F-6F-E0-EC-80-AA-D3-B4-35-B5-14-04-EE 307E40814E7D4E103F6A69B04EA78F3D (NTLM)

<user>:<id>:<LM hash>:<NTLM hash>:<comment>:<homedir>:

```
Root@kali:~# cat pw
myuser:500:12B9C54F6FE0EC80AAD3B435B51404EE:307E40814E7D4E103F6A69B04EA78F3D:::
Root@kali:~# john  pw
Loaded 1 password hash (LM DES [128/128 BS SSE2])
NAPIER           (napier)
guesses: 1  time: 0:00:00:00 100% (1)  c/s: 4850  trying: NAPIER - N4PI3R
Use the "--show" option to display all of the cracked passwords reliably
```

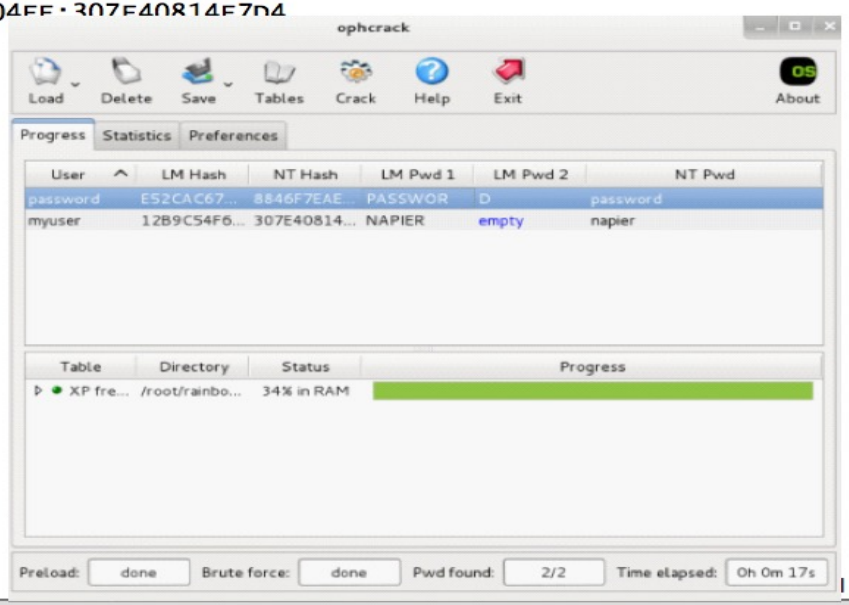
Author: Prof Bill Buchanan

SAM

Registry:
HKEY_LOCAL_MACHINE\SAM

```
Root@kali:~# cat pw
myuser:500:12B9C54F6FE0EC80AAD3B435B51404EE:307E40814E7D4E103F6A69B04EA78F3D:::
Root@kali:~# john  pw
Loaded 1 password hash (LM DES [128/128 BS SSE2])
NAPIER           (napier)
guesses: 1  time: 0:00:00:00 100% (1)  c/s: 4850  trying: NAPIER - N4PI3R
Use the "--show" option to display all of the cracked passwords reliably
```

```
<user>:<id>:<LM hash>:<NTLM hash>:<comment>:<homedir>:
password:500:E52CAC67419A9A224A3B108F3FA6CB6D:8846F7EAEE8
FB117AD06BDD830B7586C:$
myuser:500:12B9C54F6FE0EC80AAD3B435B51404EE:307E40814E7D4
E103F6A69B04EA78F3D:::
```



Hash Crackers/Bit Coin Miners

Fast Hash One

- 1.536TH/s – Cost 3-5,000 dollars.

25 GPU Hash Cracker

- An eight character NTLM password cracked in 5.5 hours. 14 character LM hash cracked in six minutes. 350 billion hashes per second.

Chapter 3: Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.

Prof Bill Buchanan OBE
http://asecuritysite.com/hasdh

Benchmark

```
Ulta fast:
Murmur:          545,716 hashes per second

Fast:
SHA-1:           134,412
SHA-256:         126,323
MD5:             125,741
SHA-512:          76,005
SHA-3 (224-bit): 72,089

Medium speed:
LDAP (SHA1):     13,718
MS DCC:           9,582
NT Hash:          7,782
MySQL:            7,724
Postgres (MD5):   7,284

Slow:
PBKDF2 (SHA-256): 5,026
Cisco PIX:        4,402
MS SQL 2000:      4,225
LDAP (MD5):       4,180
Cisco Type 7:     3,775
PBKDF2 (SHA1):    2,348

Ulta-slow:
LM Hash:            733
APR1:               234
Bcrypt:             103
DES:                 88
Oracle 10:           48
```

Hashes “The quick brown fox jumps over the lazy dog:

SHA-1: 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
SHA-256: d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592
SHA-512: 07e547d9586f6a73f73fbac0435ed76951218fb7d0c8d788a309d7
85436bbb642e93a252a954f23912547d1e8a3b5ed6e1bfd7097821233fa0538f3db854fee6

MD-5: 9e107d9d372bb6826bd81d3542a419d6
DES: ZDeS94Lcq/6zg
Bcrypt: $2a$05$2czCv5GYgkx3aobmEyewB.ejV2hePMdbvTdCyNaSzWtlGPPjB2xx6
APR1: $apr1$ZDzPE45C$3PvRanPycmNc6c2G9wT9b/
PBKDF2 (SHA1): $pbkdf2$5$WkR6UEU0NUM$0RB2bimWrMY.EPYibpaBT2q3HFg
PBKDF2 (SHA-256): $pbkdf2-sha256$5$WkR6UEU0NUM$yrJz2oJix7uBJZwZ/50vWUgdE
I/i0ffqeU4obqC0pk4

LM Hash: a7b07f9948d8cc7f97c4b0b30cae500f
NT Hash: 4e6a076ae1b04a815fa6332f69e2e231
MS DCC: efa9778bbc94a7360f664eb7d7144725
LDAP (MD5): {MD5}9e107d9d372bb6826bd81d3542a419d6
LDAP (SHA1): {SHA}2fd4e1c67a2d28fced849ee1bb76e7391b93eb12
MS SQL 2000: 0x0100BF77CE595DCD1FC87A37B3DEBC27A8C97355CB96B8BAB
63E602662BA5D5D33B913E422499BE72FF3D9BB65DE

MySQL: *A4E4D26FD0C6455E23E2187C3AABE844332AA1B3
Oracle 10: 4CDA2299FCAD0499
Postgres (MD5): md5d44c15daa11770f25c5350f7e5408dd1
Cisco PIX: kGyKN5CqdFQ1qJUs
Cisco Type 7: 15260309443B3E2D2B3875200108010D41505640135E1B0E080
5195741564015400 35E460B594D1D53020B5C

Benchmark

Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.

Prof Bill Buchanan OBE

http://asecuritysite.com/crypto02
http://asecuritysite.com/encryption

"testing123" and a key of "hello", and you should get:
AC2C2E614882CE7158F69B7E3B12114465945D01

HMAC-MD5
HMAC-SHA-1
HMAC-SHA-256
HMAC-SHA-384
HMAC-SHA-512

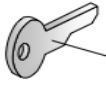
MAC 3DES

Sender

Receiver

Message

Message



HMAC

HMAC

Secret key

Secret key

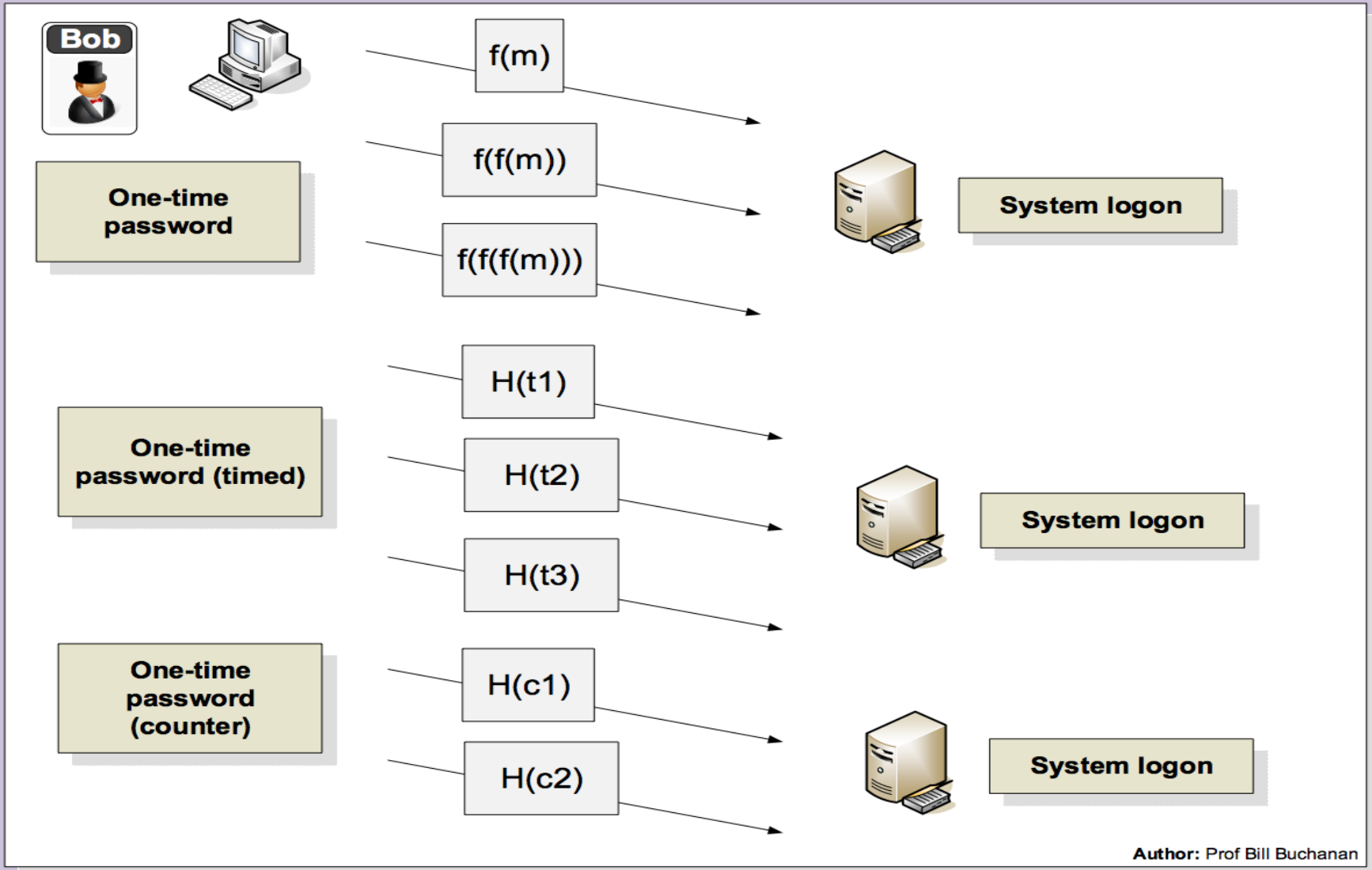
Receiver checks the HMAC code against received one – if they match the sender is validated, and the message is also confirmed

Author: Prof Bill Buchanan

Hashing

Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.

Prof Bill Buchanan OBE
http://asecuritysite.com/hash

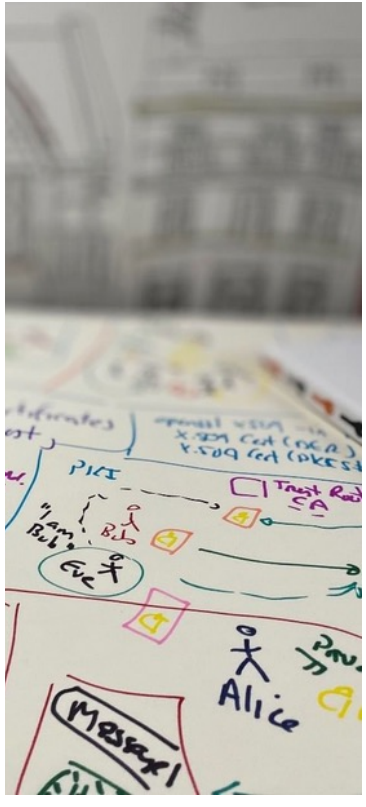

Bob
f(m)
f(f(m))
f(f(f(m)))
One-time password
System logon
H(t1)
H(t2)
H(t3)
One-time password (timed)
System logon
H(c1)
H(c2)
One-time password (counter)
System logon
Author: Prof Bill Buchanan

Hashing

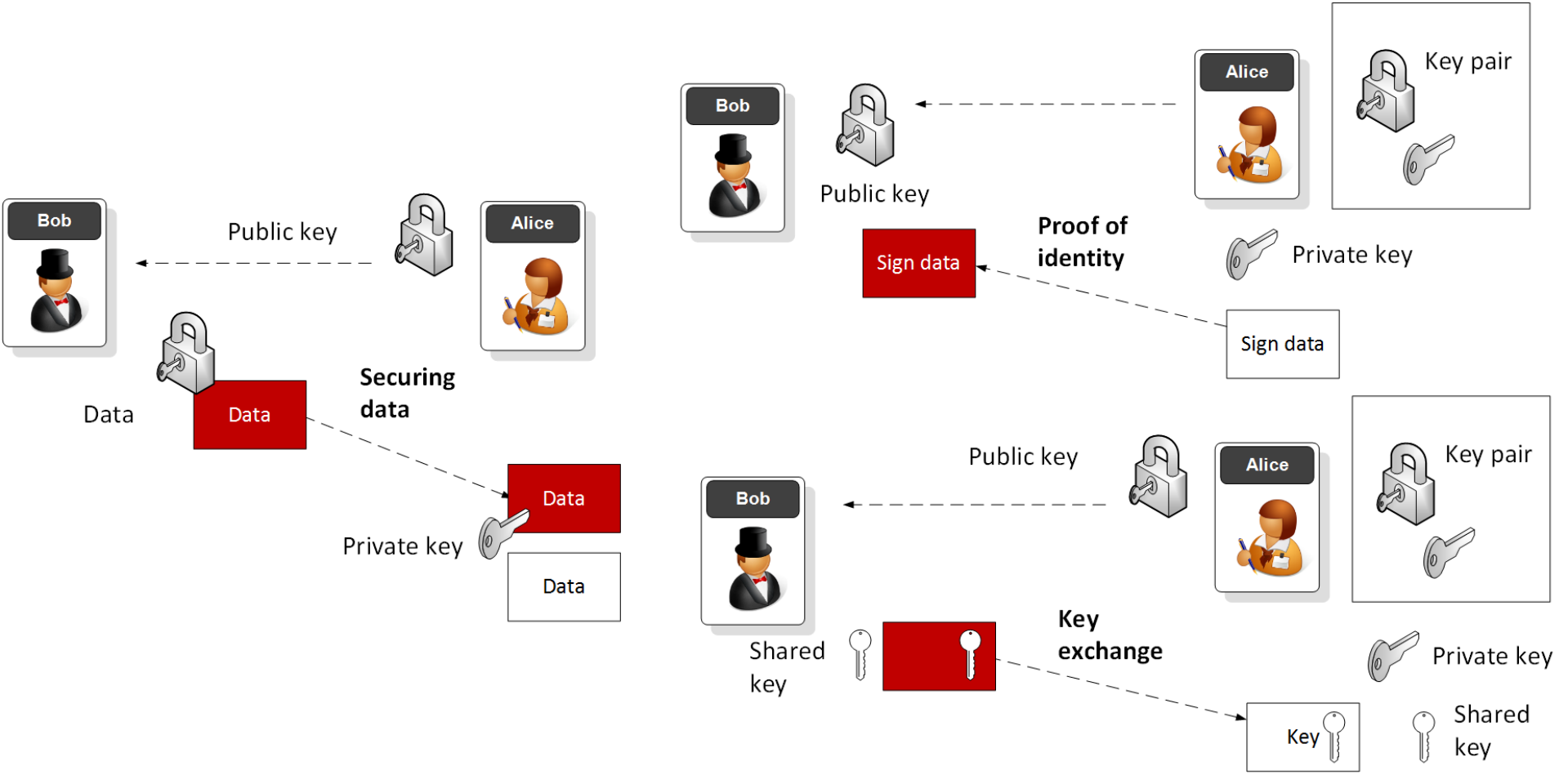
Hashing Types.
Hashing Methods.
Salting.
Collisions.
LM and NTLM Hashes (Windows).
Hash Benchmarks.
Message Authentication Codes (MACs).
OTP/HOTP.

Prof Bill Buchanan OBE
http://asecuritysite.com/**hash**

No	Date	Subject	Lab
2	15 Sept 2022	1. Introduction [Link] 2. Intrusion Detection Systems [Link]	Introduction to Vyatta Lab
3	22 Sept 2022	3. Network Security [Link]	Vyatta and Snort. [Link]
4	29 Sept 2022	4. Ciphers and Fundamentals [Link]	pfSense.
5	6 Oct 2022	5. Secret Key 6. Hashing [Link]	AWS Security and Server Infrastructures
6	13 Oct 2022	7. Public Key [Link] 8. Key Exchange [Link]	Public/Private Key and Hashing
7	20 Oct 2022	9. Digital Certificates	Certificates here
8	27 Oct 2022	10 Network Forensics here	Network Forensics lab
9	3 Nov 2022	Test 1 here	
10	10 Nov 2022	11. Splunk here	Splunk Lab
11	17 Nov 2022	12. Splunk and Machine Learning	Splunk and ML Lab
12	24 Nov 2022	13. Tunnelling here	Tunnelling
13	1 Dec 2022	14. Blockchain and Cryptocurrencies here	Blockchain Lab.
14	8 Dec 2022		
15	15 Dec 2022	Hand-in: TBC [Here]	

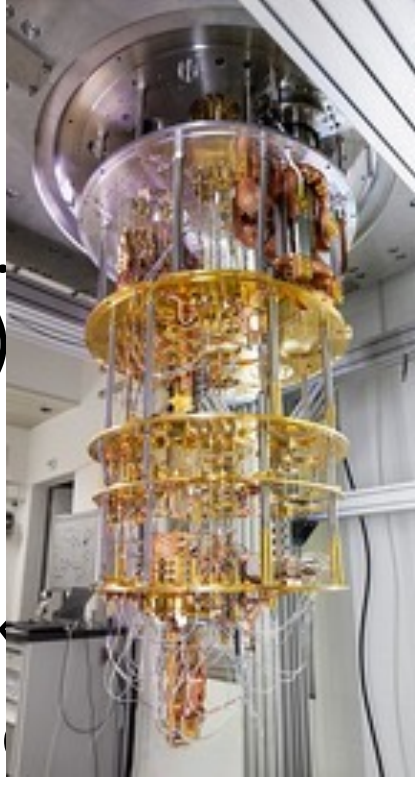


Public Key Methods



Public Key Methods

- **Integer Factorization**. Using prime numbers. Example: RSA. Key size: 2,048 bits (modulus) Digital Certificates.
- **Discrete Logarithms**. $Y = g^x \bmod P$. Example: Prime number size: 2,048 bits. Key handshak
- **Elliptic Curve Relationships**. Example: Ellipti Private key: 256 bits. Public key: 512 bits. Bitcoin, IoT, Web, etc.



Public Key

RSA

Prof Bill Buchanan OBE

http://asecuritysite.com/crypto04

http://asecuritysite.com/encryption

p

9,137,187,070,061,098,912,312,979,400,361
,251,189,847,923,809,497,258,114,688,790,
849,334,008,324,856,676,348,809,151,285,1
18,821,829,375,998,699,013,311,467,364,66
2,378,853,216,263,996,490,005,611,058,805

p

9,885,919,140,818,765,444,174,626,190,703
,294,219,553,850,295,249,705,938,896,539,
634,343,302,401,155,295,752,383,276,739,5
84,190,165,200,823,122,225,274,427,125,93
4,163,475,191,779,288,529,189,149,818,011

(p-1)*(q-1)

90,329,492,549,158,751,736,593,291,654,313,033,317,391,509,546,977,632,
830,551,342,194,781,230,803,832,847,247,315,213,556,011,813,523,182,777
,529,551,800,128,685,586,665,697,818,108,995,125,892,738,489,085,065,56
4,398,419,119,705,178,003,889,155,415,914,402,310,708,147,858,313,669,1
76,692,847,865,236,706,085,105,432,191,429,510,583,595,108,030,256,069,
207,938,161,732,170,083,525,341,774,967,620,008,260,040

With Diffie-Hellman we need the other side to be active before we send data. Can we generate a special one-way function which allows is to distribute an encryption key, while we have the decryption key?

Solved in 1977, By Ron Rivest, Adi Shamir, and Len Aldeman created the RSA algorithm for public-key encryption.

RSA

- Two primes p, q.
- Calculate N (modulus) as p x q eg 3 and 11. n=33.
- Calculate PHI as (p-1)x(q-1). PHI=20
- Select e for no common factor with PHI. e=3.
- Encryption key [e,n] or [3,33].
- (d x e) mod 20 = 1
- (d x 3) mod 20 = 1
- d= 7
- Decryption key [d,n] or [7,33] (link)

RSA

Calc

Example

- Encryption key [e,n] or [3,33].
- Decryption key [d,n] or [7,33]
- Cipher = M^e mod N

eg M=5.

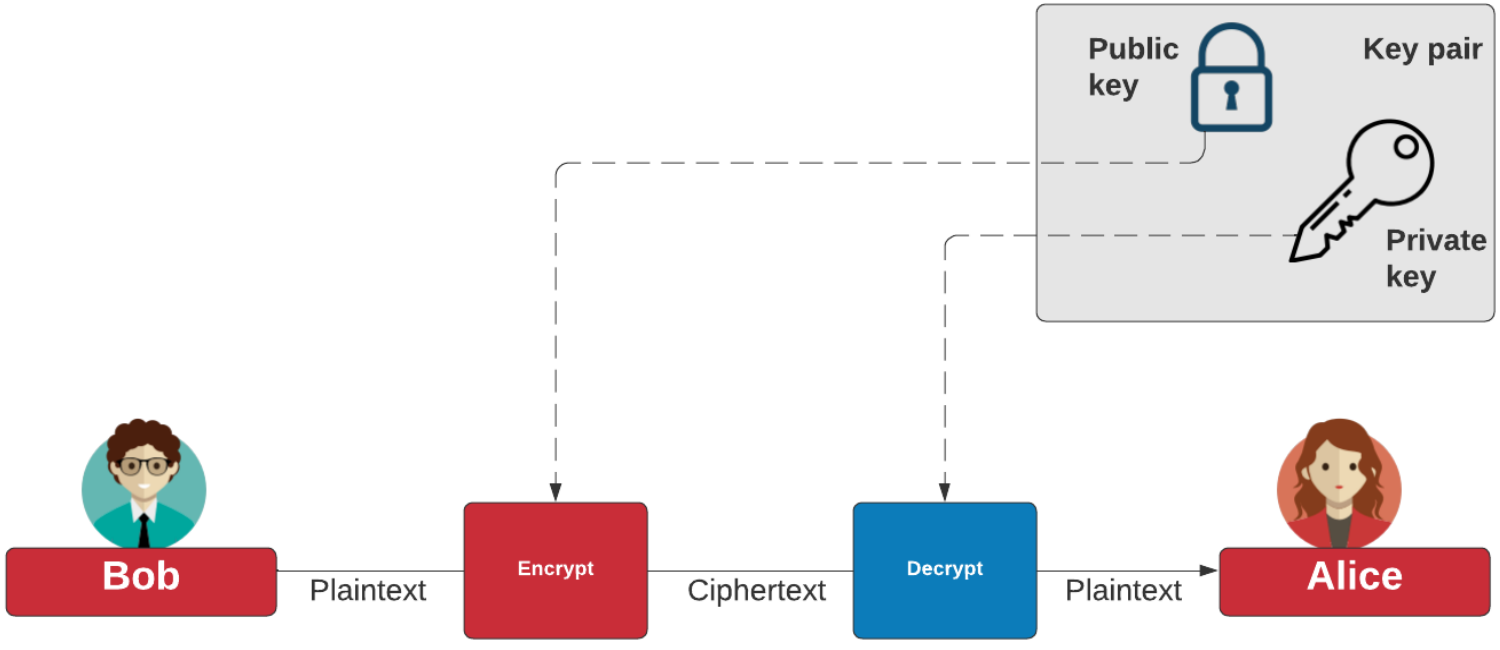
- Cipher = 5^3 mod 33 = 26
- Decipher = C^d mod N
- Decipher = $(26)^7$ mod 33 = 5

Public Key

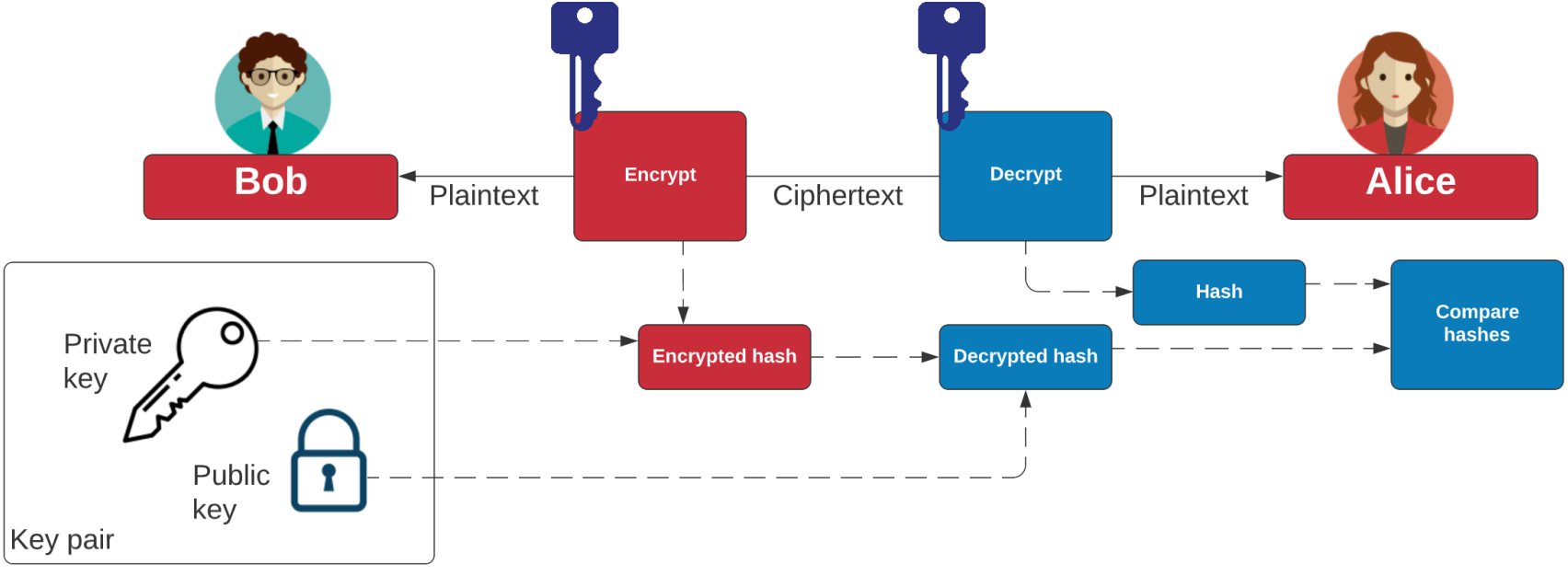
Basics
RSA
Applications (Encryption and Signing)

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

Public Key Encryption



Public Key Digital Signing



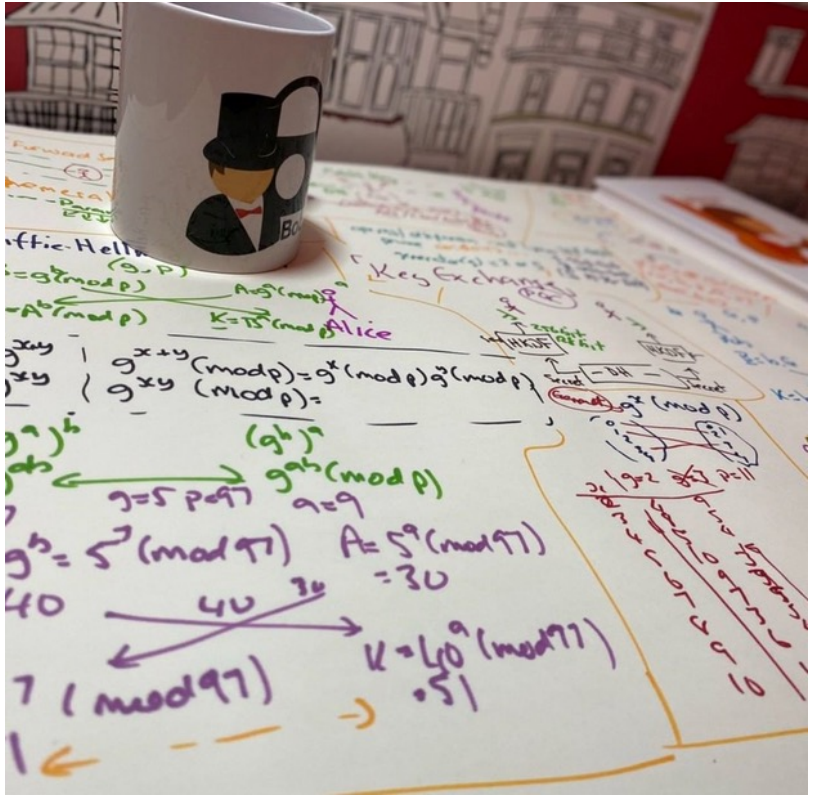
Public Key

Basics
RSA
Applications (Encryption and Signing)

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

Key Exchange

Diffie-Hellman
Diffie-Hellman Weaknesses
Passing Key Using Public Key

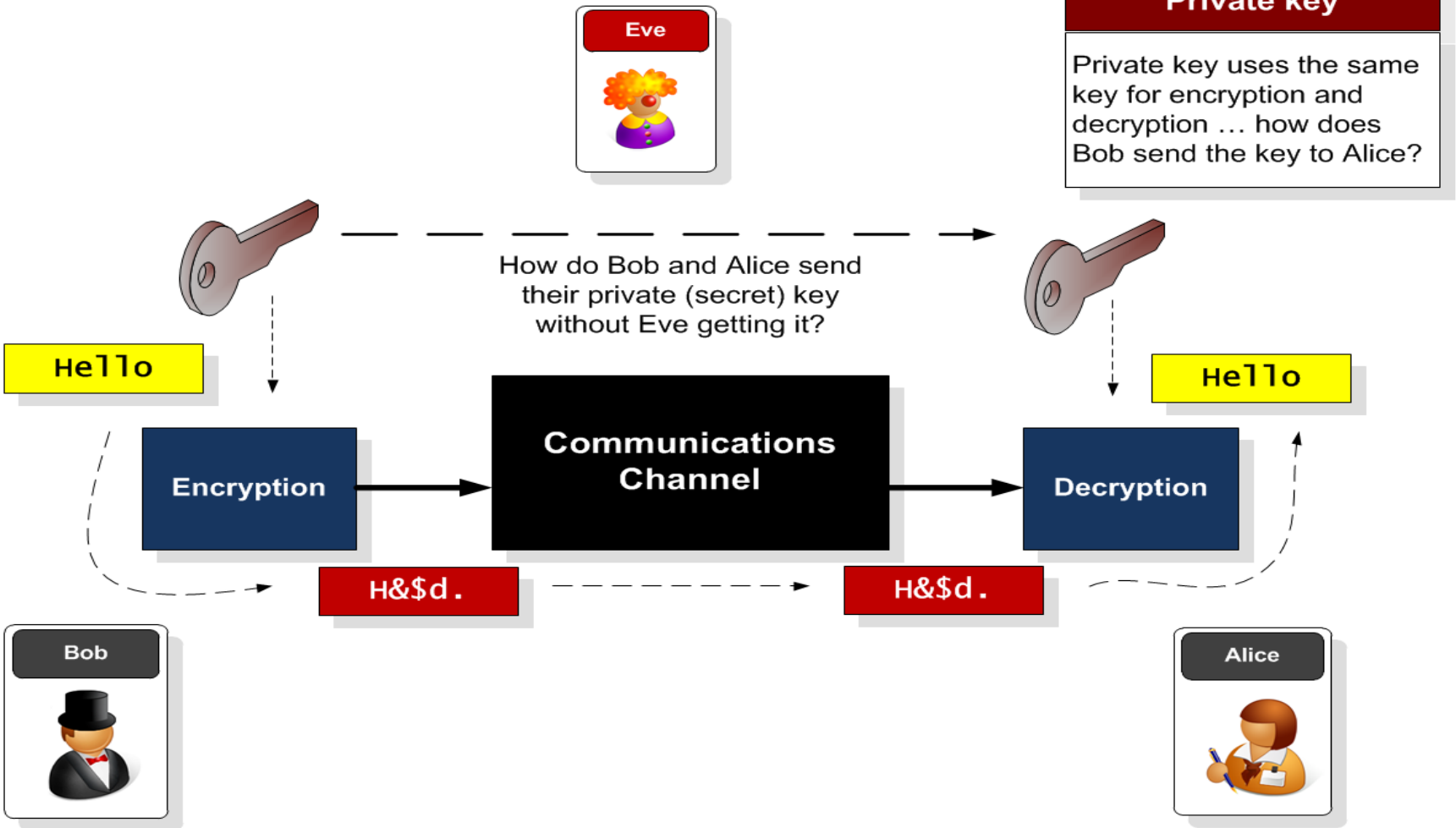


Prof Bill Buchanan OBE

http://asecuritysite.com/crypto05
http://asecuritysite.com/encryption

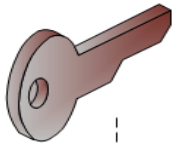
Private key

Private key uses the same key for encryption and decryption … how does Bob send the key to Alice?

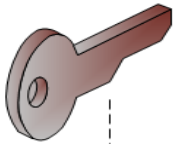


Diffie-Hellman

One of the most widely method for creating a secret key which is the same for Bob and Alice



How do Bob and Alice send their private (secret) key without Eve getting it?



Hello

Encryption

Communications Channel

Decryption

Hello

H&$d.

H&$d.

This problem was solved by Whitfield Diffie, who created the Diffie-Hellman algorithm, which is the most widely used method for passing secret keys

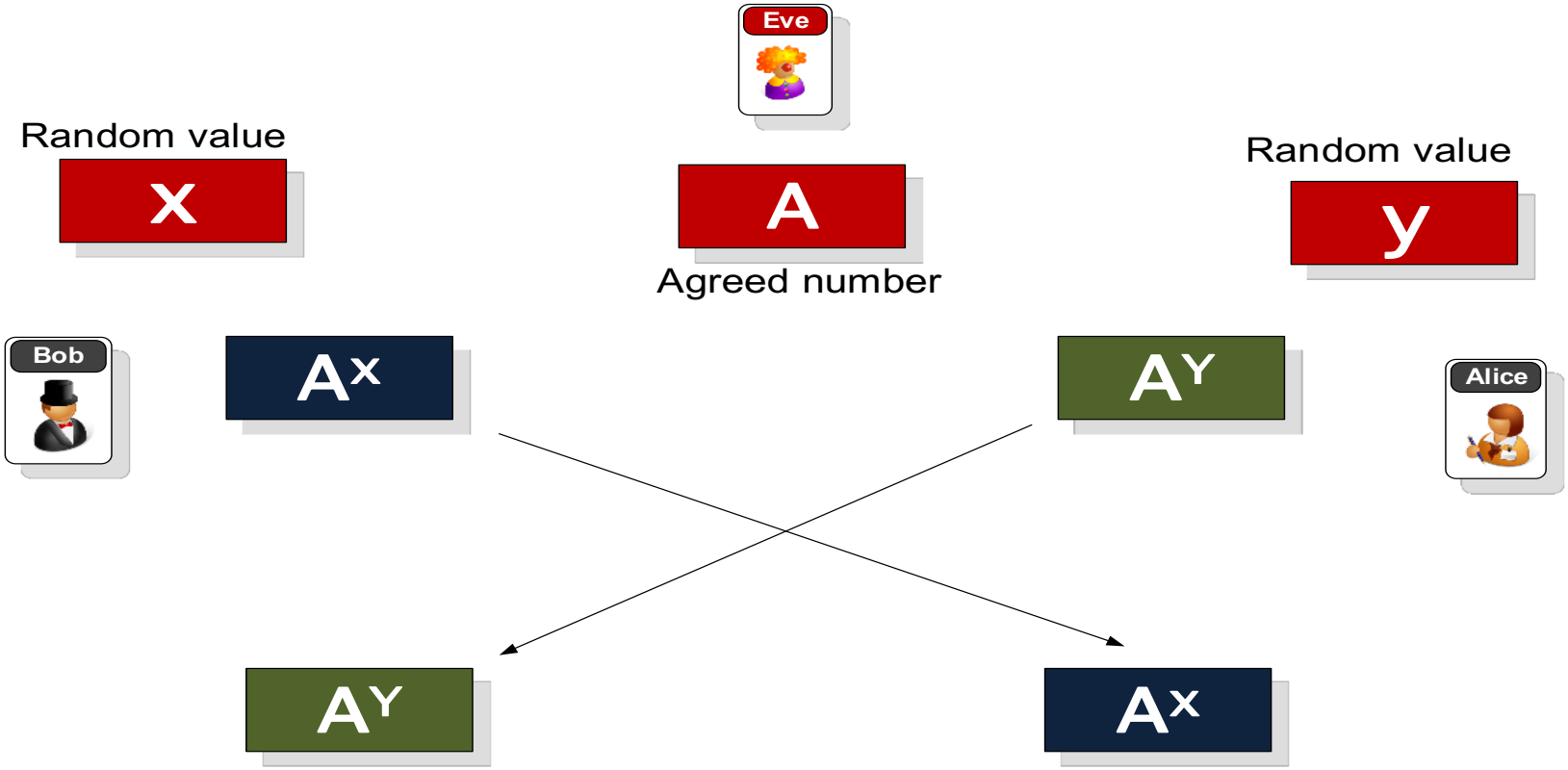
Key Exchange

- **Forward secrecy** (FS), which means that a comprise of the long-term keys will not compromise any previous session keys. A leakage of the public key of the server would cause all the sessions which used this specific public key to be compromised. FS thus aims to overcome this by making sure that all the sessions keys could not be compromised, even though the long-term key was compromised.
- **Ephemeral**. With some key exchange methods, the same key will be generated if the same parameters are used on either side. This can cause problems as an intruder could guess the key, or even where the key was static and never changed. With ephemeral methods, a different key is used for each connection, and, again, the leakage of any long-term would not cause all the associated session keys to be breached.

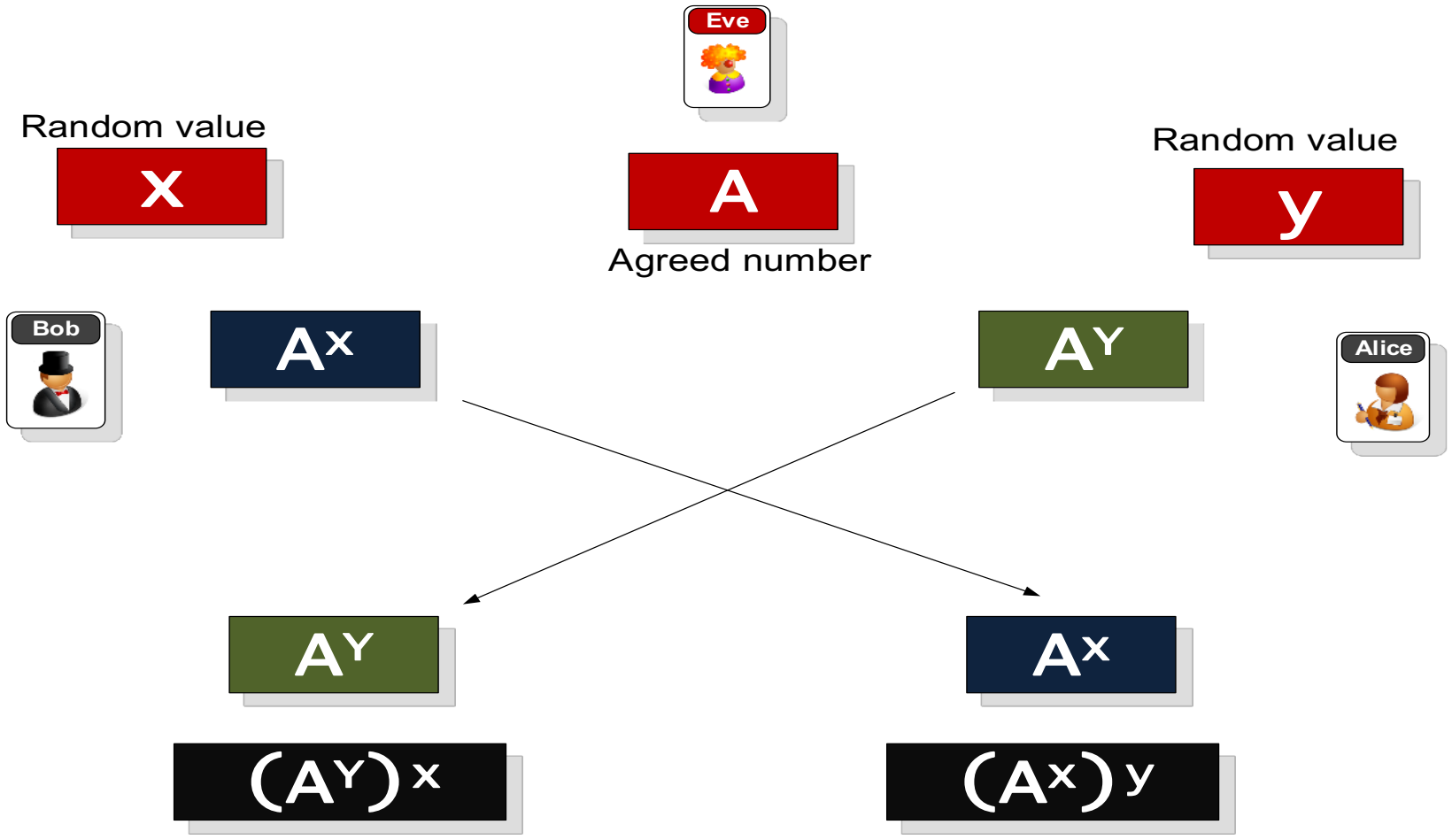
$A^x A^y \rightarrow A^{(x+y)}$

$(A^x)^y \rightarrow A^{xy}$

Author: Prof Bill Buchanan



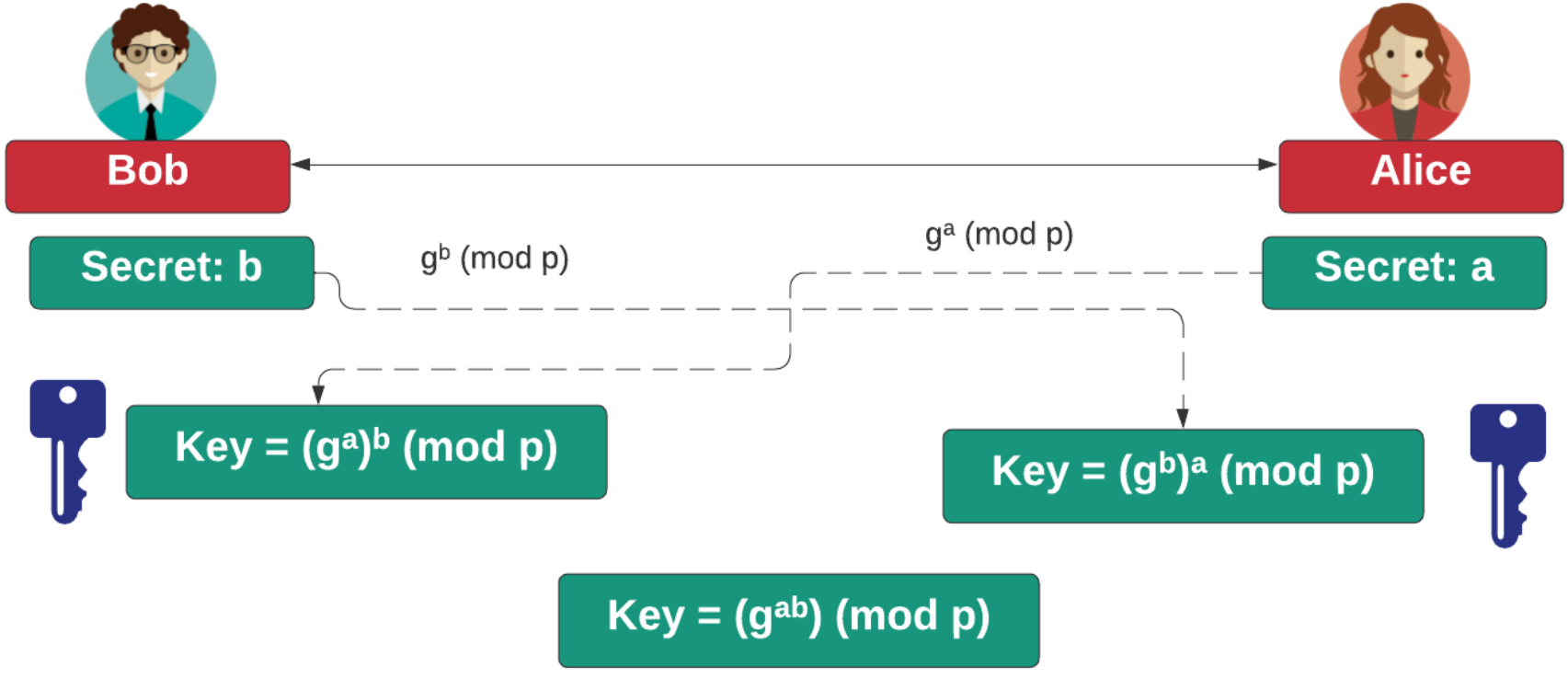
Author: Prof Bill Buchanan

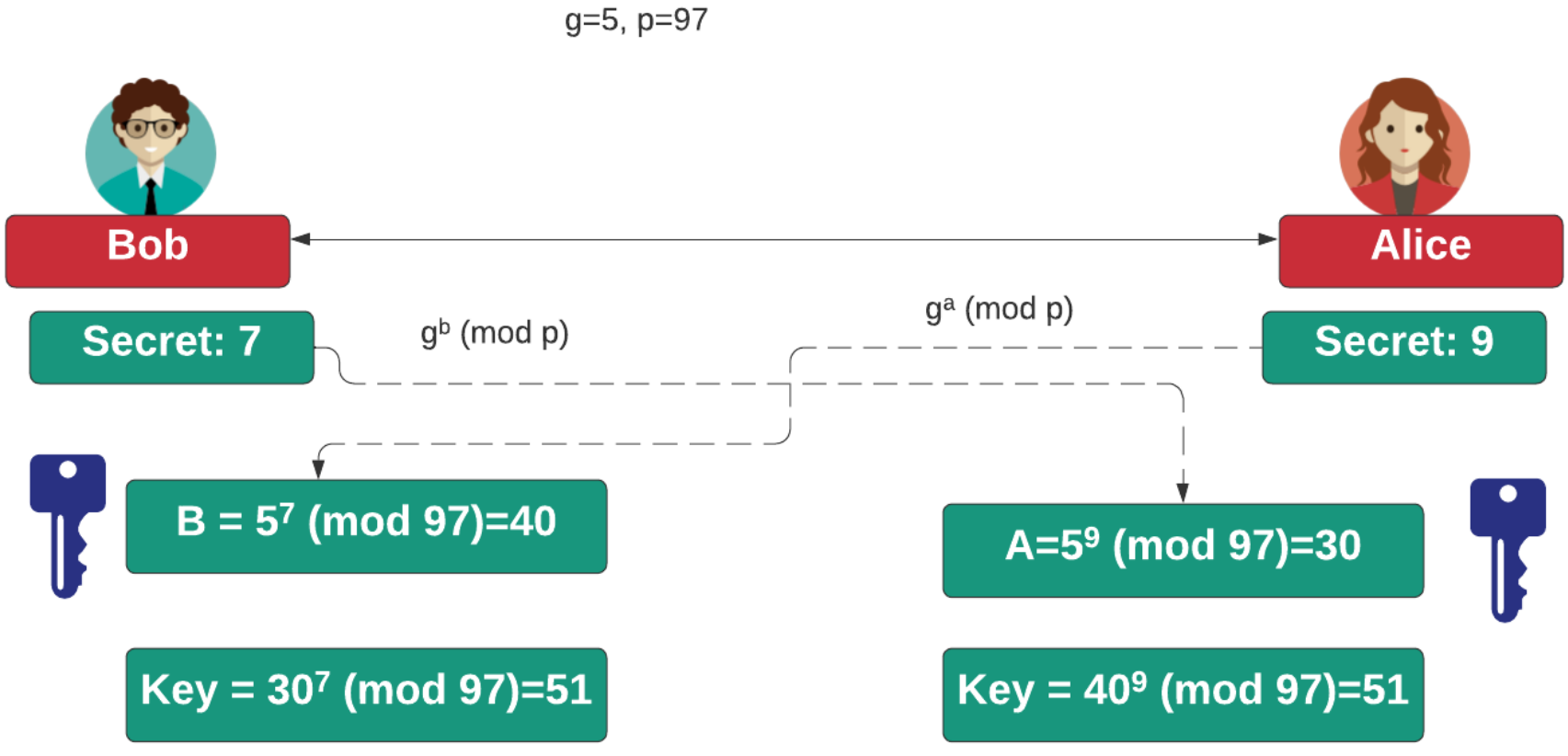


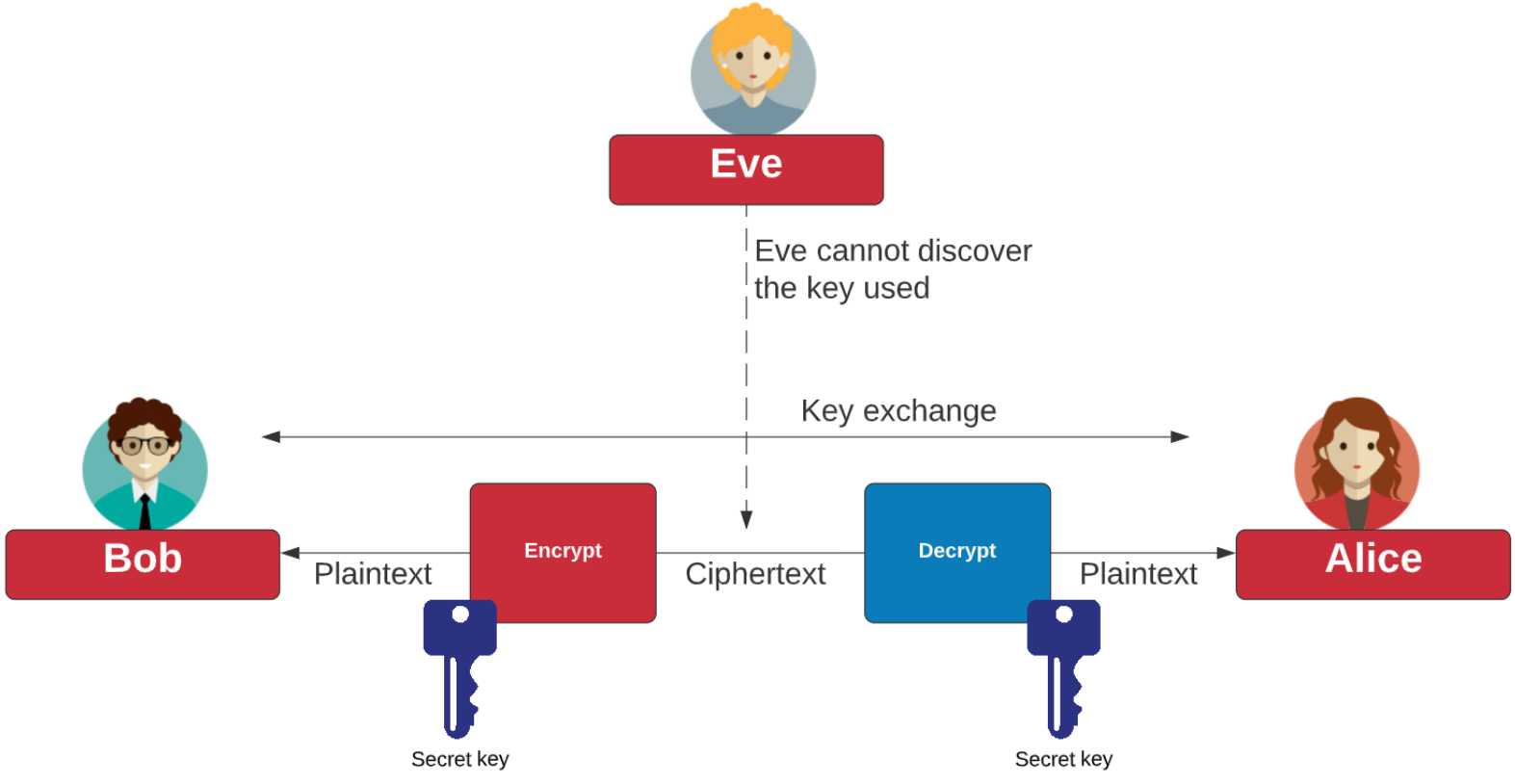
Logs

Encryption

Author: Prof Bill Buchanan


Bob
Alice
Secret: b
Secret: a
$g^b \pmod p$
$g^a \pmod p$
Key = $(g^a)^b \pmod p$
Key = $(g^b)^a \pmod p$
Key = $(g^{ab}) \pmod p$


g=5, p=97
Bob
Alice
Secret: 7
Secret: 9
$g^b \pmod p$
$g^a \pmod p$
$B = 5^7 \pmod{97}=40$
$A=5^9 \pmod{97}=30$
$Key = 30^7 \pmod{97}=51$
$Key = 40^9 \pmod{97}=51$


Eve
Eve cannot discover
the key used
Key exchange
Bob
Alice
Encrypt
Decrypt
Plaintext
Ciphertext
Plaintext
Secret key
Secret key

Diffie-Hellman Generator

$$Y = g^x \bmod p$$

p	11							
Generator	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
x	**g^x mod p**	**g^x mod p**	**g^x mod p**	**g^x mod p**	**g^x mod p**	**g^x mod p**	**g^x mod p**	**g^x mod p**
2	4	9	5	3	3	5	9	4
3	8	5	9	4	7	2	6	3
4	5	4	3	9	9	3	4	5
5	10	1	1	1	10	10	10	1
6	9	3	4	5	5	4	3	9
7	7	9	5	3	8	6	2	4
8	3	5	9	4	4	9	5	3
9	6	4	3	9	2	8	7	5
10	1	1	1	1	1	1	1	1

Picking G

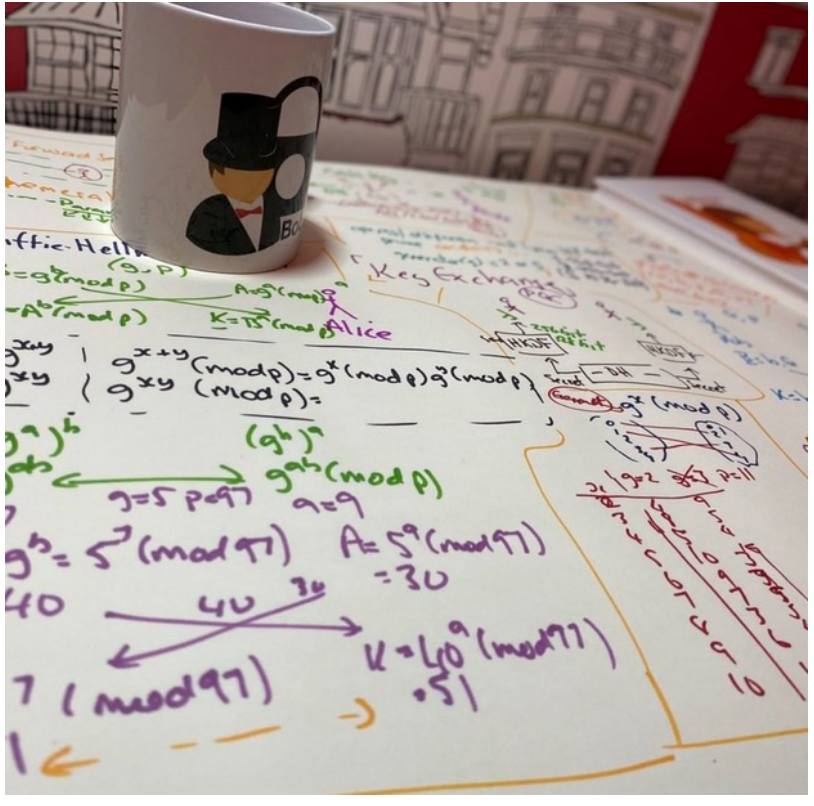
Diffie-Hellman Generation

```
C:\> openssl dhparam -out dhparams.pem 768 –text
C:\> type dhparams.pem
Diffie-Hellman-Parameters: (768 bit)
    prime:
        00:d0:37:c2:95:64:02:ea:12:2b:51:50:a2:84:6c:
        71:6a:3e:2c:a9:80:e2:65:b2:a5:ee:77:26:22:31:
        66:9e:fc:c8:09:94:e8:9d:f4:cd:bf:d2:37:b2:fb:
        b8:38:2c:87:28:38:dc:95:24:73:06:d3:d9:1f:af:
        78:01:10:6a:7e:56:4e:7b:ee:b4:8d:6b:4d:b5:9b:
        93:c6:f1:74:60:01:0d:96:7e:85:ca:b8:1f:f7:bc:
        43:b7:40:4d:4e:87:e3
    generator: 2 (0x2)
-----BEGIN DH PARAMETERS-----
MGYCYQDQN8KVZALqEitRUKKEbHFqPiypgOJlsqXudyYiMWae/MgJlOid9
M2/0jey
+7g4LIcoONyVJHMG09kfr3gBEGp+Vk577rSNa021m5PG8XRgAQ2WfoXKu
B/3vEO3
QE1Oh+MCAQI=
-----END DH PARAMETERS-----
```

- **DH Group 5**: 1,536 bit prime.
- **DH Group 2**: 1,024 bit prime.
- **DH Group 1**: 768-bit prime.

Key Exchange

Diffie-Hellman
Diffie-Hellman Weaknesses
Passing Key Using Public Key



Prof Bill Buchanan OBE

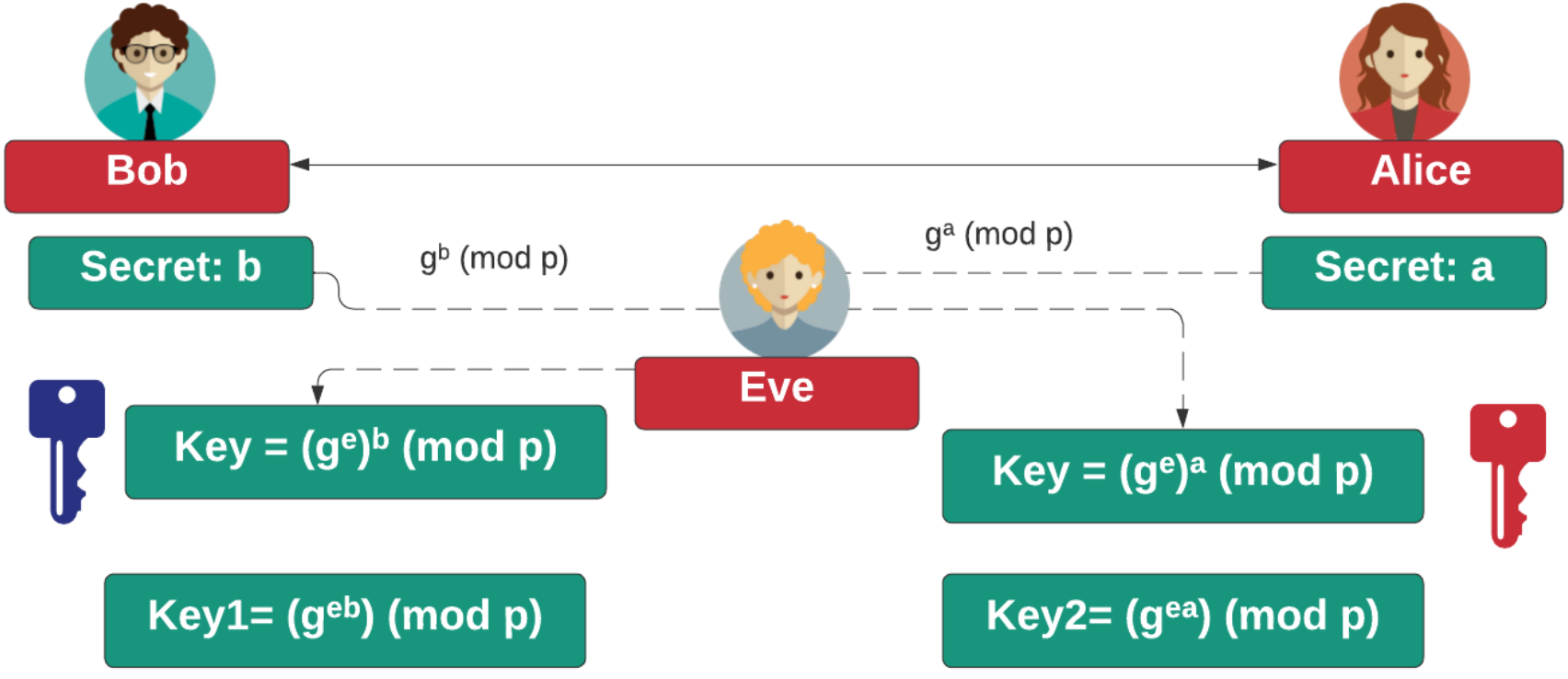
http://asecuritysite.com/crypto05
http://asecuritysite.com/encryption

Diffie-Hellman Weaknesses

- In 2015, a paper entitled *Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice* – showed that it was fairly easy to precompute on values for two popular Diffie-Hellman parameters (and which use the DHE_EXPORT cipher set).
- The research team found that one was used as a default in the around 7% of the Top 1 million web sites and was hard coded into the Apache httpd service. Overall, at the time, it was found that over 3% of Web sites were still using the default.

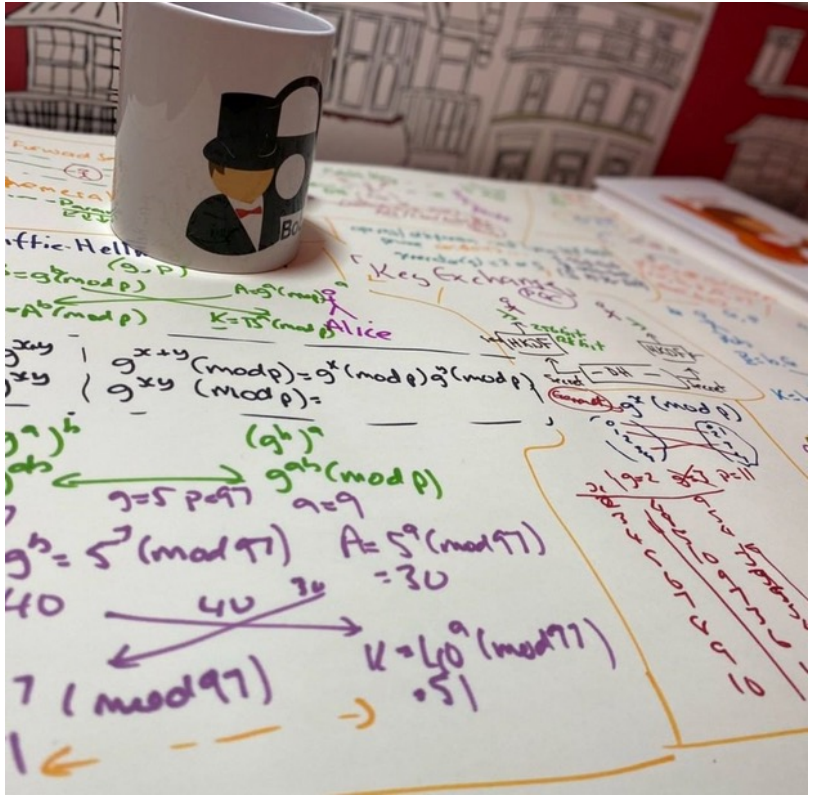
- Diffie-Hellman-Parameters: (512 bit)
- prime:
- 00:9f:db:8b:8a:00:45:44:f0:04:5f:17:37:d0:ba:
- 2e:0b:27:4c:df:1a:9f:58:82:18:fb:43:53:16:a1:
- 6e:37:41:71:fd:19:d8:d8:f3:7c:39:bf:86:3f:d6:
- 0e:3e:30:06:80:a3:03:0c:6e:4c:37:57:d0:8f:70:
- e6:aa:87:10:33
- generator: 2 (0x2)

Eve-in-the-middle



Key Exchange

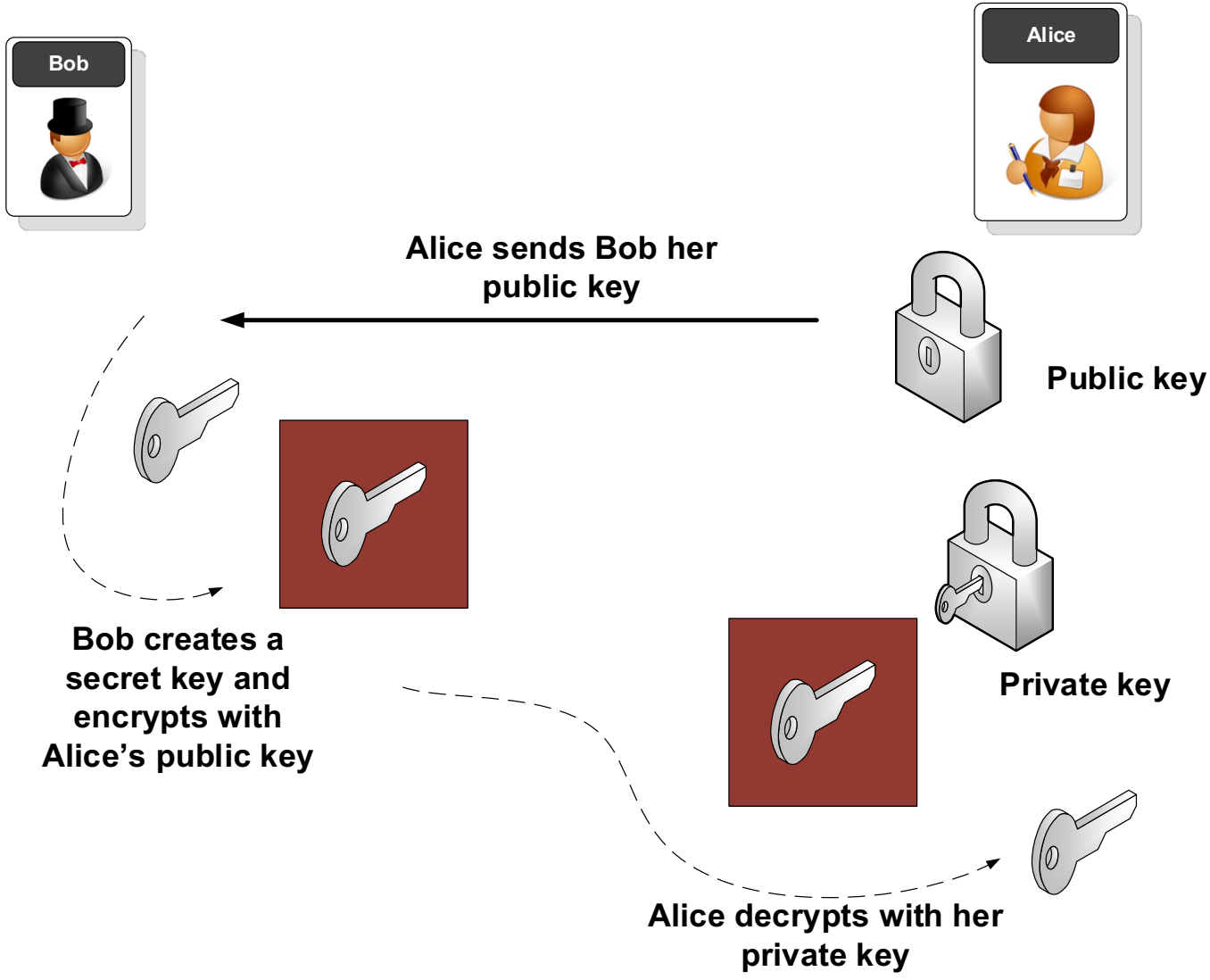
Diffie-Hellman
Diffie-Hellman Weaknesses
Passing Key Using Public Key



Prof Bill Buchanan OBE

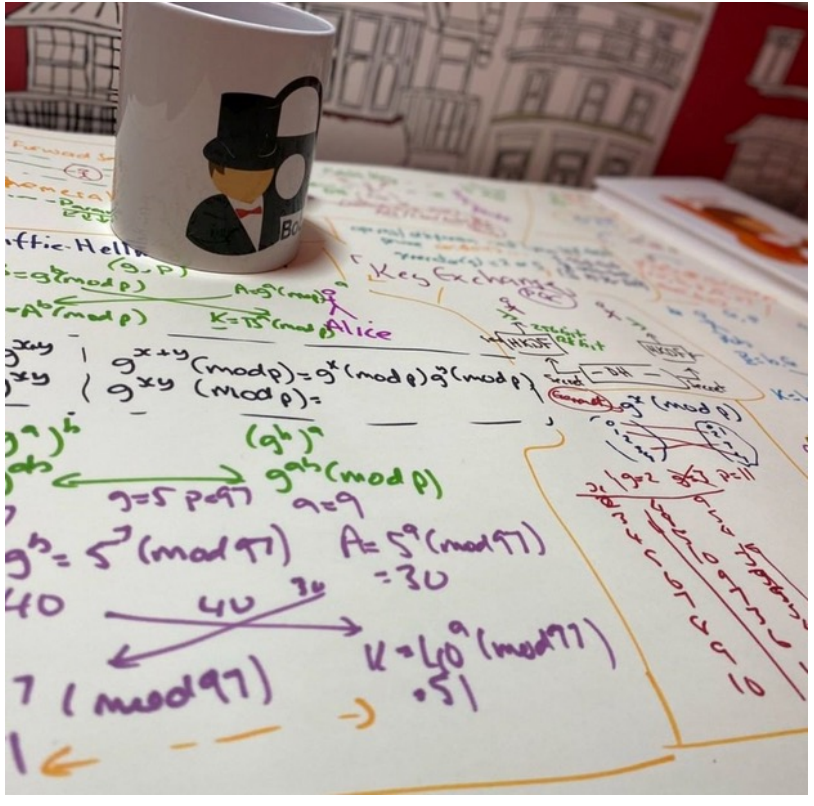
http://asecuritysite.com/crypto05
http://asecuritysite.com/encryption

Key Exchange with Public Key



Key Exchange

Diffie-Hellman
Diffie-Hellman Weaknesses
Passing Key Using Public Key



Prof Bill Buchanan OBE

http://asecuritysite.com/crypto05
http://asecuritysite.com/encryption

Dig

Introd

Authe

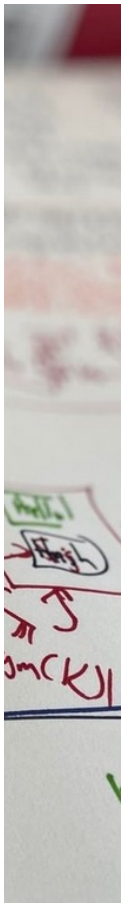
PKI

Digita

Prof

https:

2	15 Sept 2022	1. Introduction [Link] 2. Intrusion Detection Systems [Link]	Introduction to Vyatta Lab
3	22 Sept 2022	3. Network Security [Link]	Vyatta and Snort. [Link]
4	29 Sept 2022	4. Ciphers and Fundamentals [Link]	pfSense.
5	6 Oct 2022	5. Secret Key 6. Hashing [Link]	AWS Security and Server Infrastructures
6	13 Oct 2022	7. Public Key [Link] 8. Key Exchange [Link]	Public/Private Key and Hashing
7	20 Oct 2022	9. Digital Certificates	Certificates here
8	27 Oct 2022	10 Network Forensics here	Network Forensics lab
9	3 Nov 2022	Test 1 here	
10	10 Nov 2022	11. Splunk here	Splunk Lab
11	17 Nov 2022	12. Splunk and Machine Learning	Splunk and ML Lab
12	24 Nov 2022	13. Tunnelling here	Tunnelling
13	1 Dec 2022	14. Blockchain and Cryptocurrencies here	Blockchain Lab.
14	8 Dec 2022		
15	15 Dec 2022	Hand-in: TBC [Here]	



Identity on the Internet

Identifies it is trusted (Digital Certificate)

Keeps communications secure (encryption)

Fundamental principles

Authentication.
Confidence/Assurance.
Privacy/Confidentiality.

Authentication
(Device,
User, Servers,
Connections, etc)

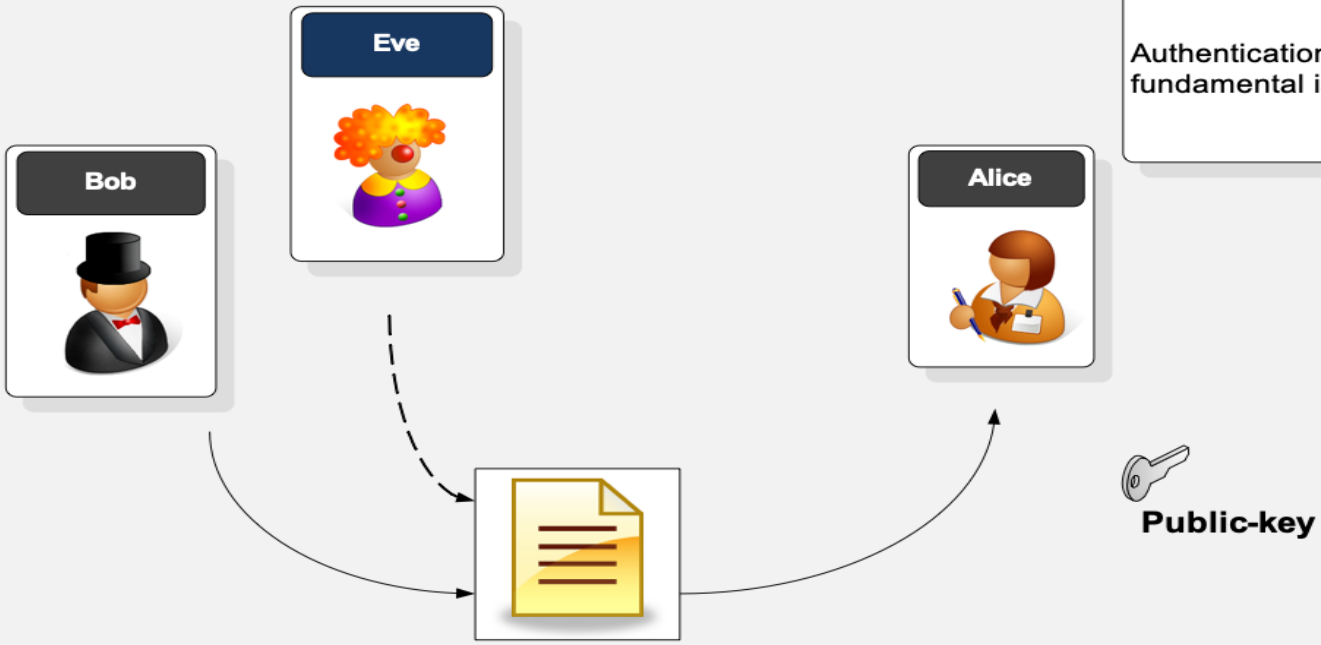
Confidentiality
(Encryption)

Assurance
(Integrity)

Author: Prof Bill Buchanan

Authentication

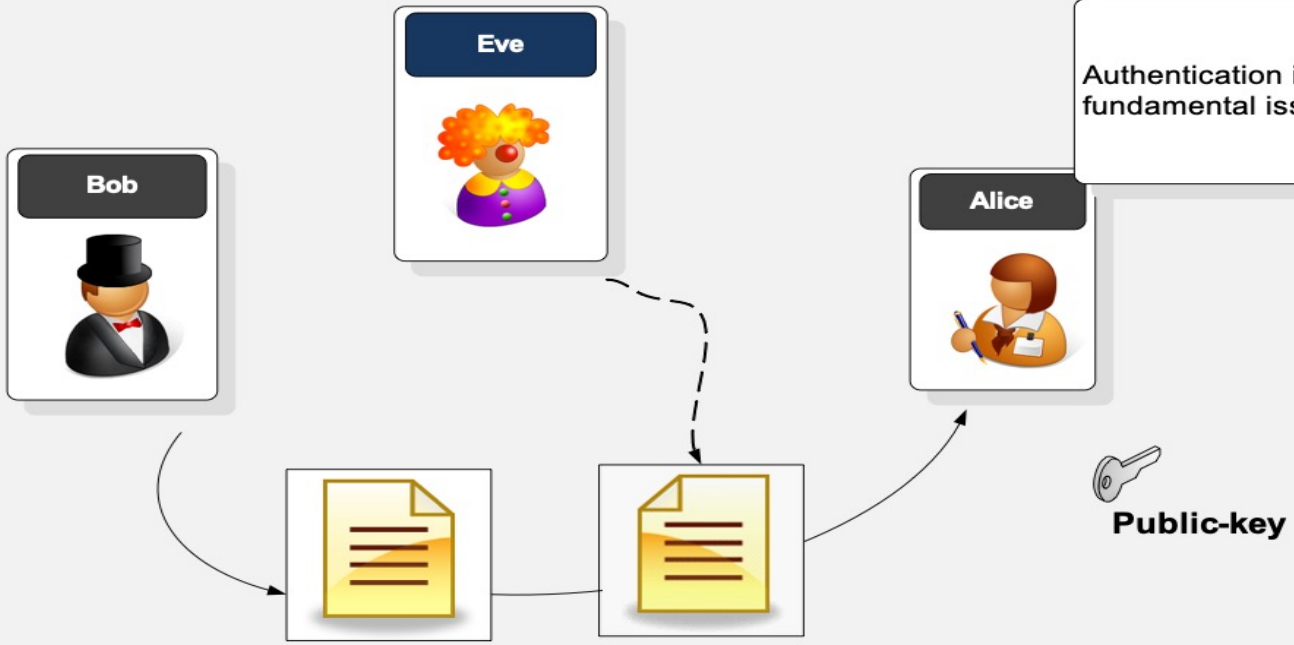
Authentication is a fundamental issue in security.



How do we know that it was really Bob who sent the data, as anyone can get Alice's public key, and thus pretend to be Bob?

Authentication

Authentication is a fundamental issue in security.

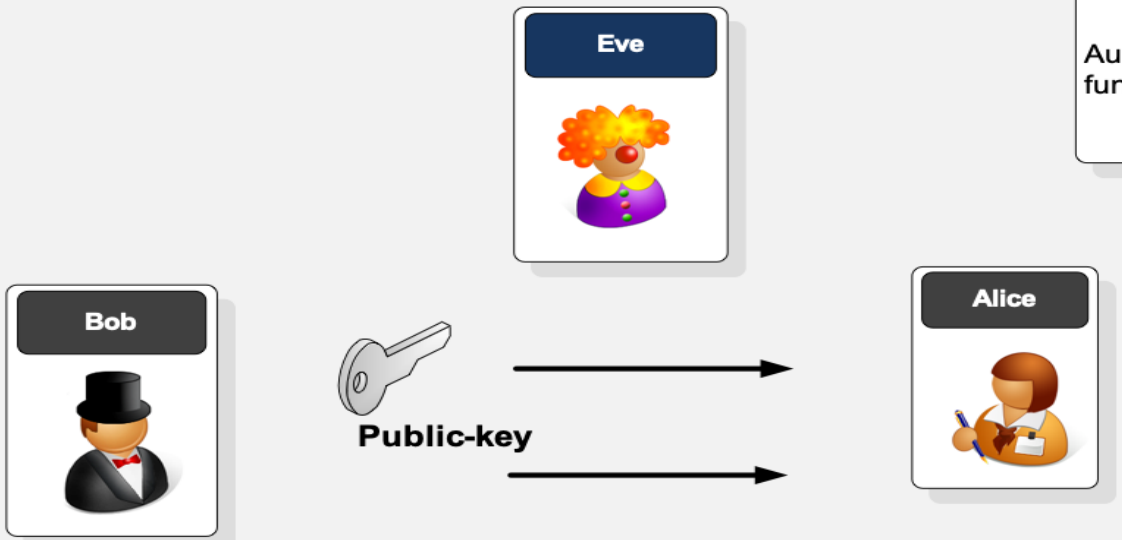


How can we tell that the message has not been tampered with ?

Author: Prof Bill Buchanan

Authentication

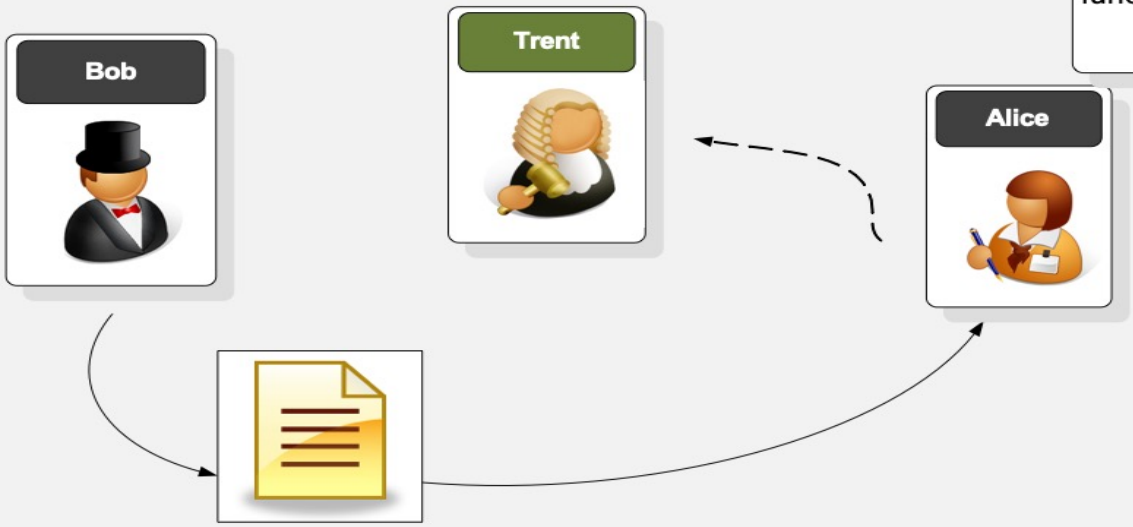
Authentication is a fundamental issue in security.



How does Bob distribute his public key to Alice, without having to post it onto a Web site or for Bob to be on-line when Alice reads the message?

Authentication

Authentication is a fundamental issue in security.



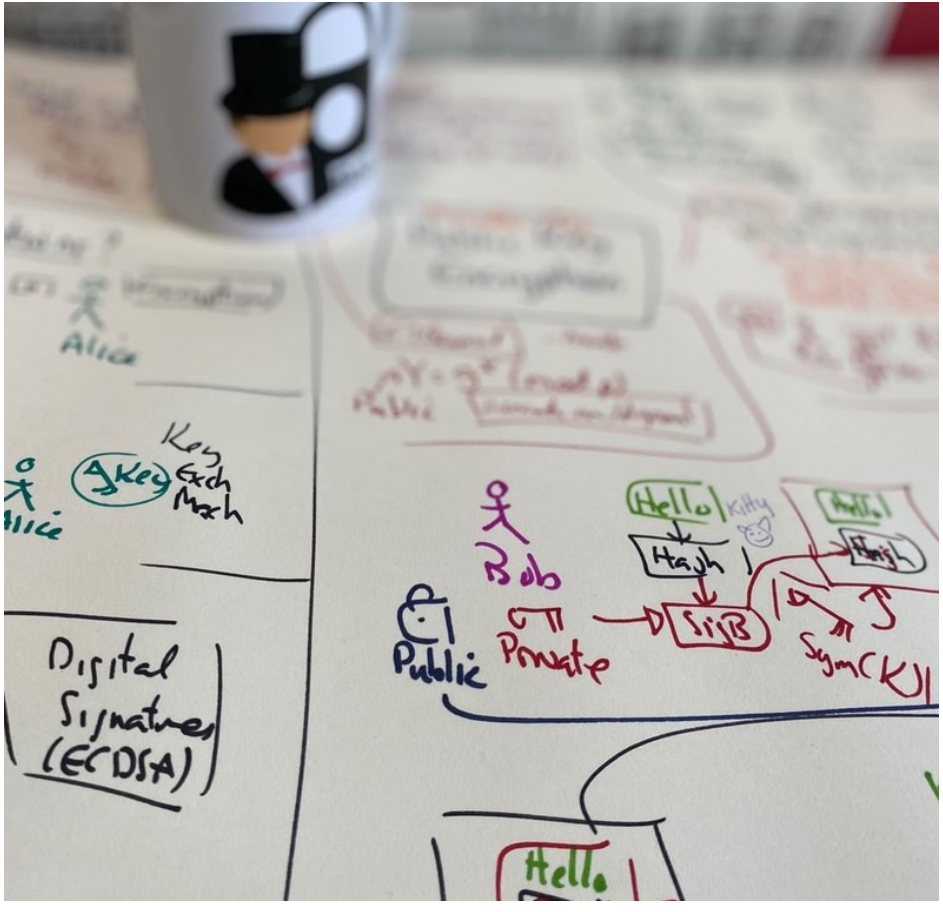
Who can we *really* trust to properly authenticate Bob? Obviously we can't trust Bob to authenticate that he really is Bob.

Author: Prof Bill Buchanan

Unit 6: Digital Certificates

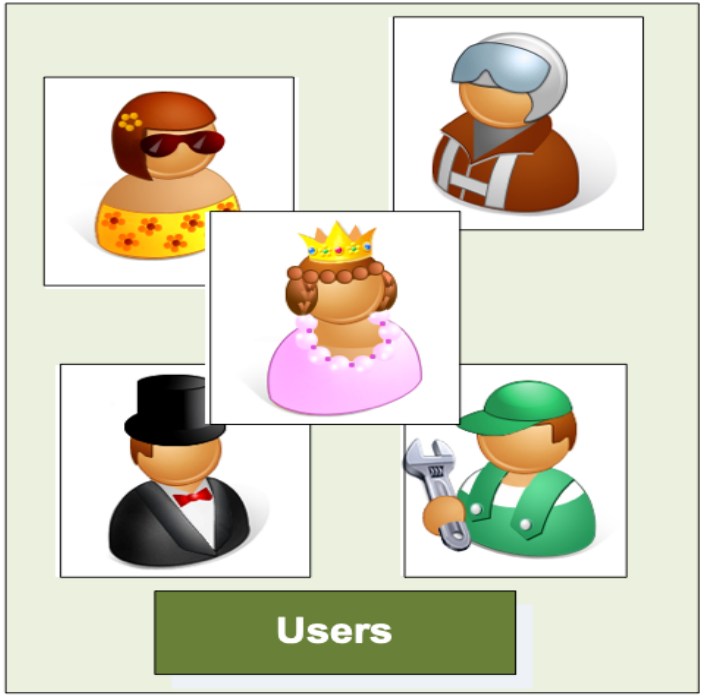
Introduction
Authentication Methods

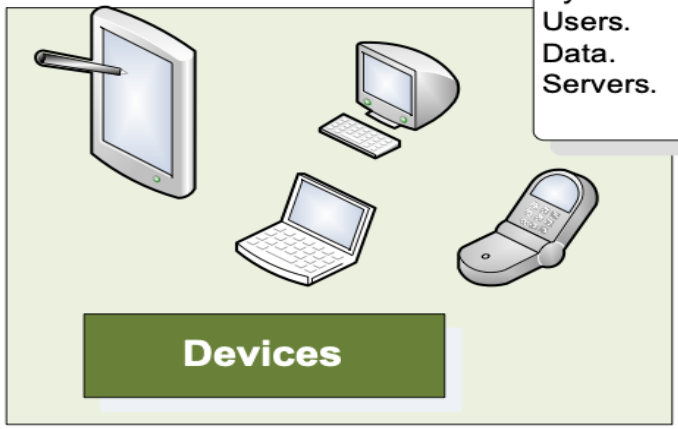
Prof Bill Buchanan OBE
https://asecuritysite.com/tunnelling/

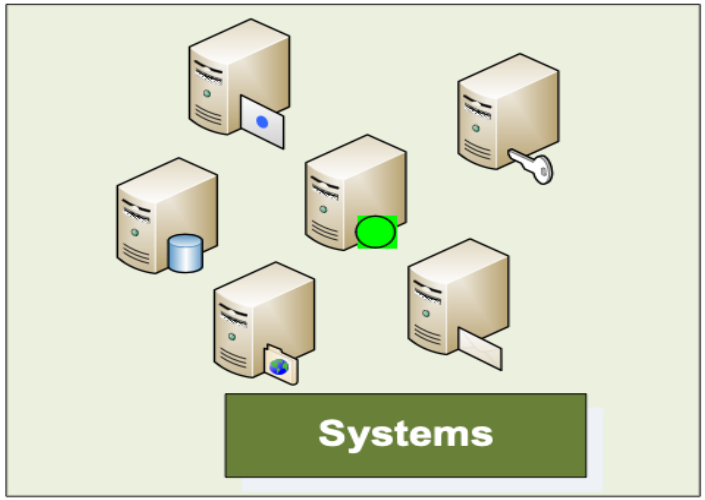


What to authenticate?

Systems.
Users.
Data.
Servers.



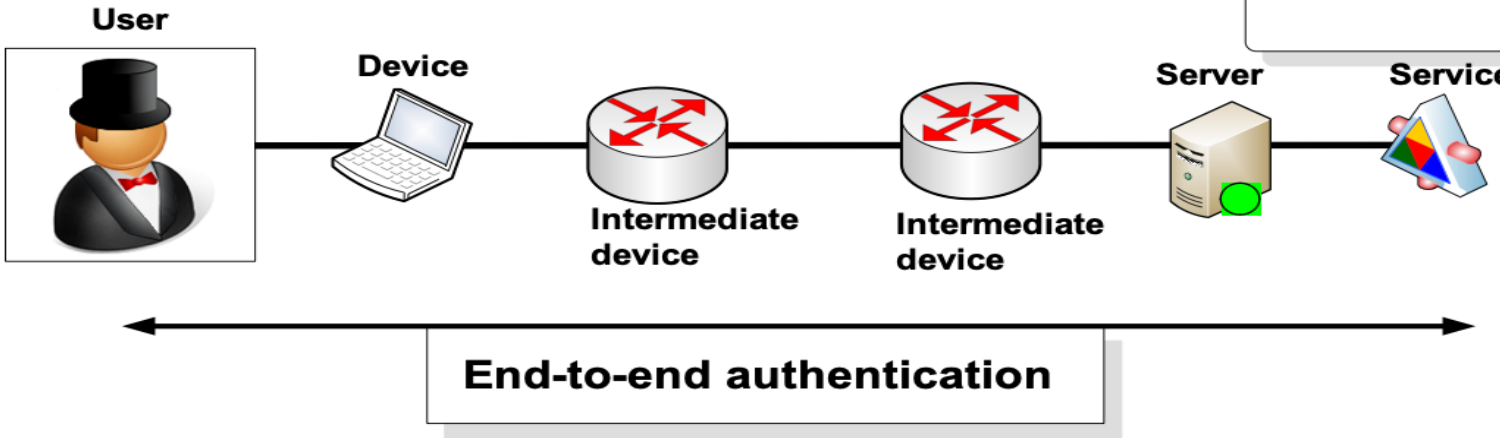


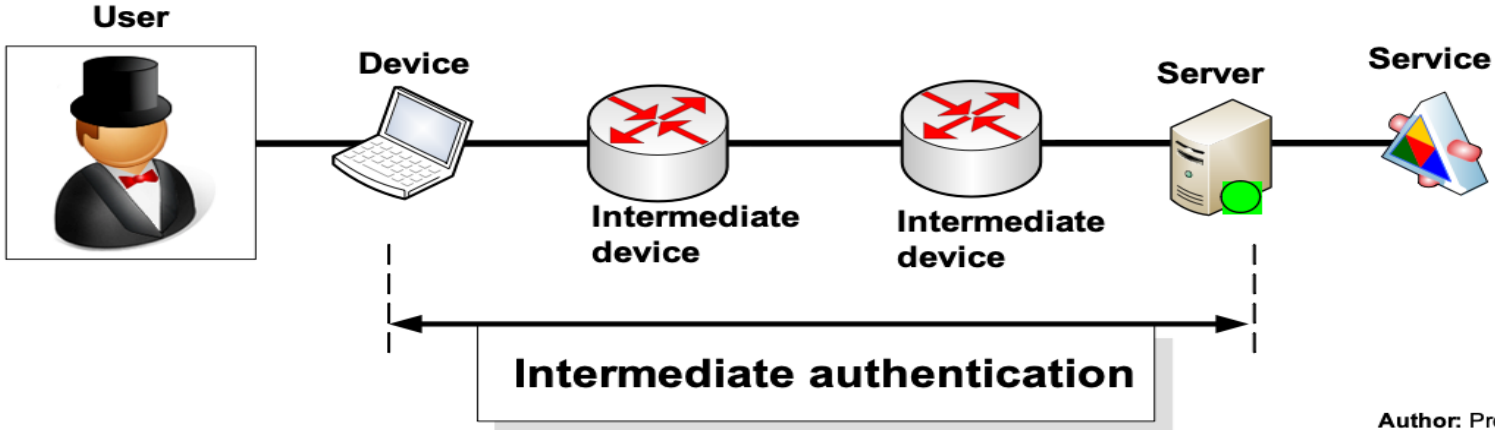


Author: Prof Bill Buchanan

Where authenticated?

End-to-end. User to service. Intermediate. Part of the authentication process.

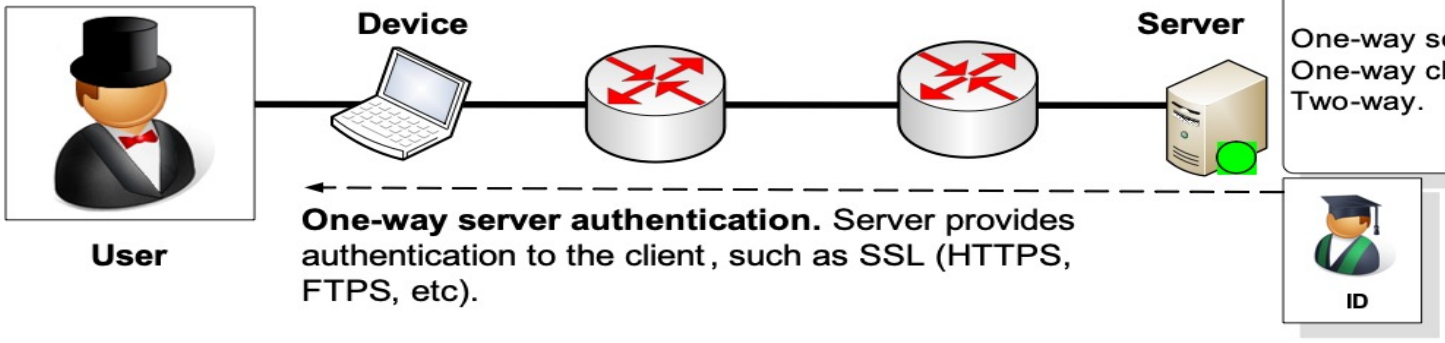




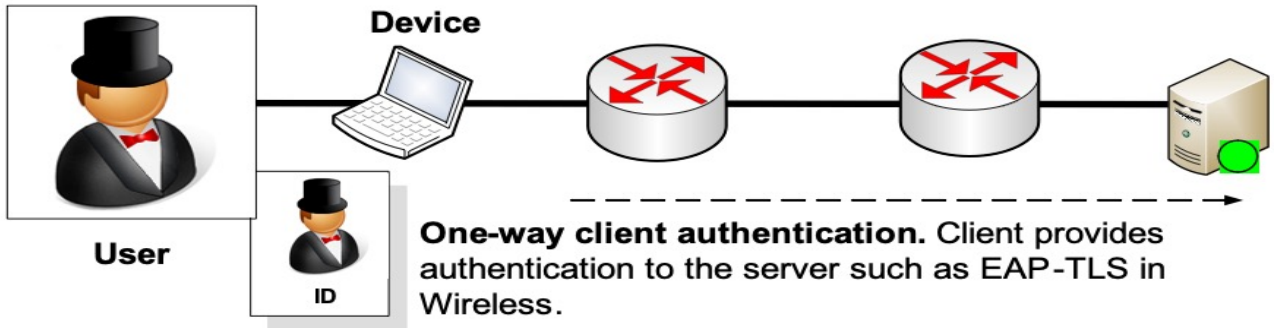
Author: Prof Bill Buchanan

Authentication type

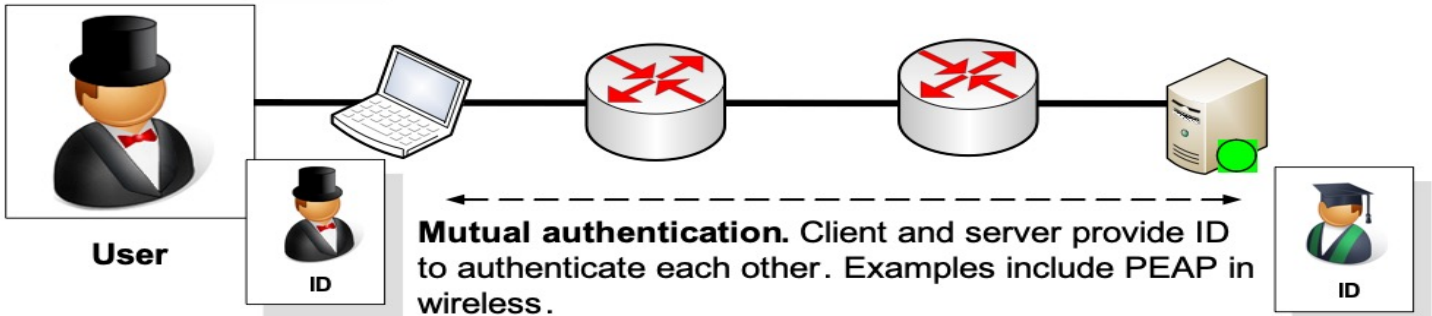
One-way server.
One-way client.
Two-way.



One-way server authentication. Server provides authentication to the client, such as SSL (HTTPS, FTPS, etc).



One-way client authentication. Client provides authentication to the server such as EAP-TLS in Wireless.



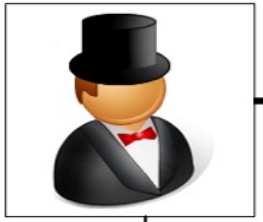
Mutual authentication. Client and server provide ID to authenticate each other. Examples include PEAP in wireless.

Author: Prof Bill Buchanan

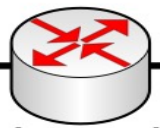
Authentication type

One-way server.
One-way client.
Two-way.

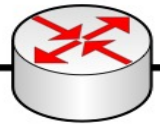
User



Device



Intermediate device



Intermediate device

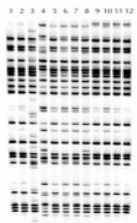
Server

Service

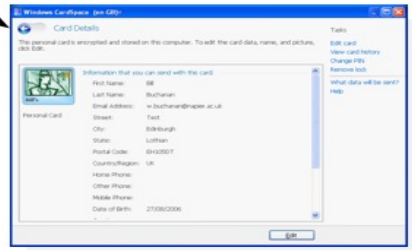
Authentication

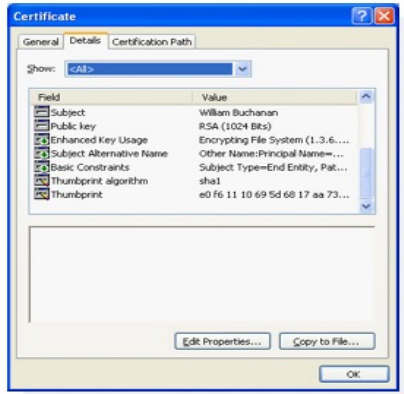
Methods

Username/password
Digital Certificate
Token Card
Soft Tokens
Session key
Pass phrase
Biometrics



Device name
Digital Certificate
Pass phrase
MAC address
Encryption key





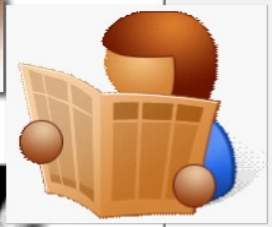
Author: Prof Bill Buchanan

Authentication methods

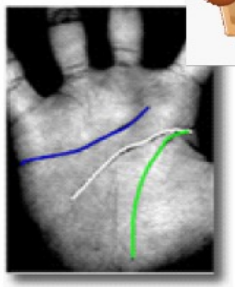
Something you have
Something you know
Something you are

Authentication Methods

Iris scans



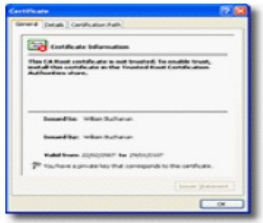
Retina scan



Palm prints

Finger prints

Something you are



Digital certificate

Network/physical address

Smart card

Something you have

Username/ password

Mother's maiden name

Something you know

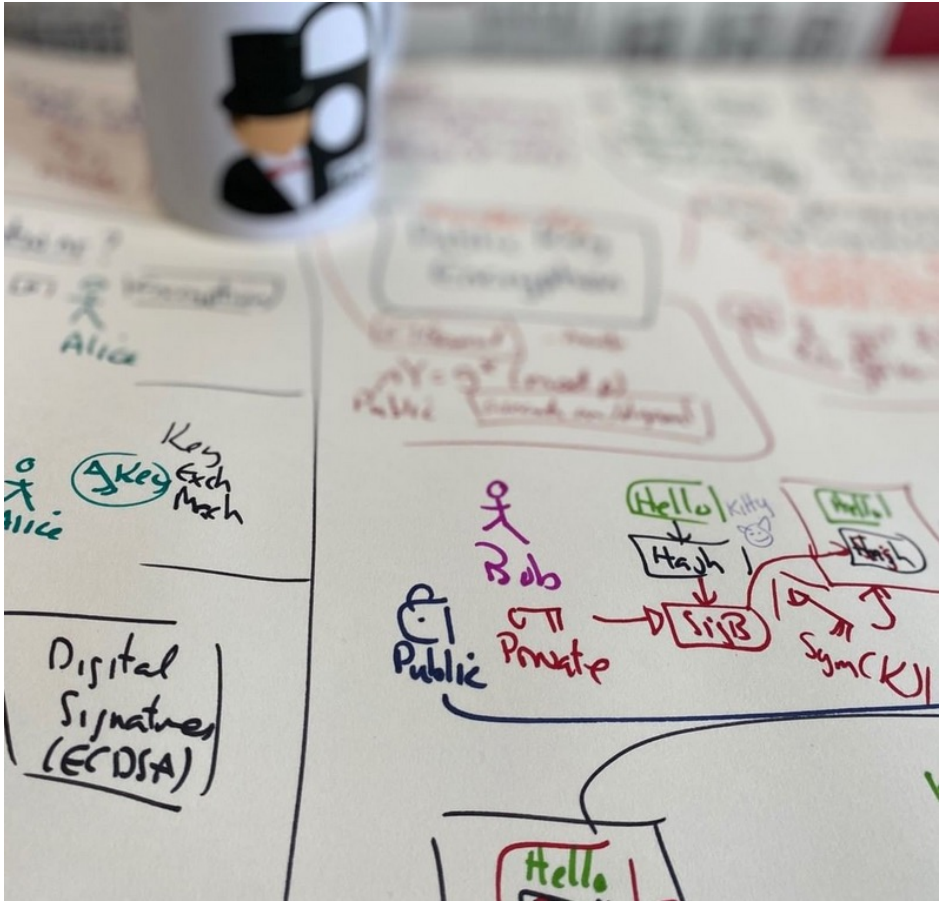
Author: Prof Bill Buchanan

Digital Certificates

Introduction
Authentication Methods
PKI
Digital Certificate Passing

Prof Bill Buchanan OBE

https://asecuritysite.com/tunnelling



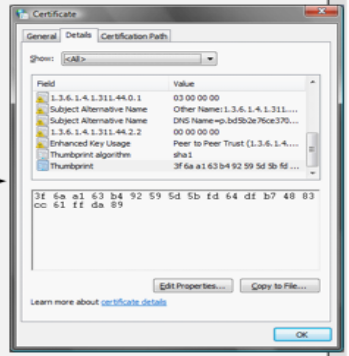
Digital Certificates

Digital certificates are a soft token of authentication, and require a trust mechanism.

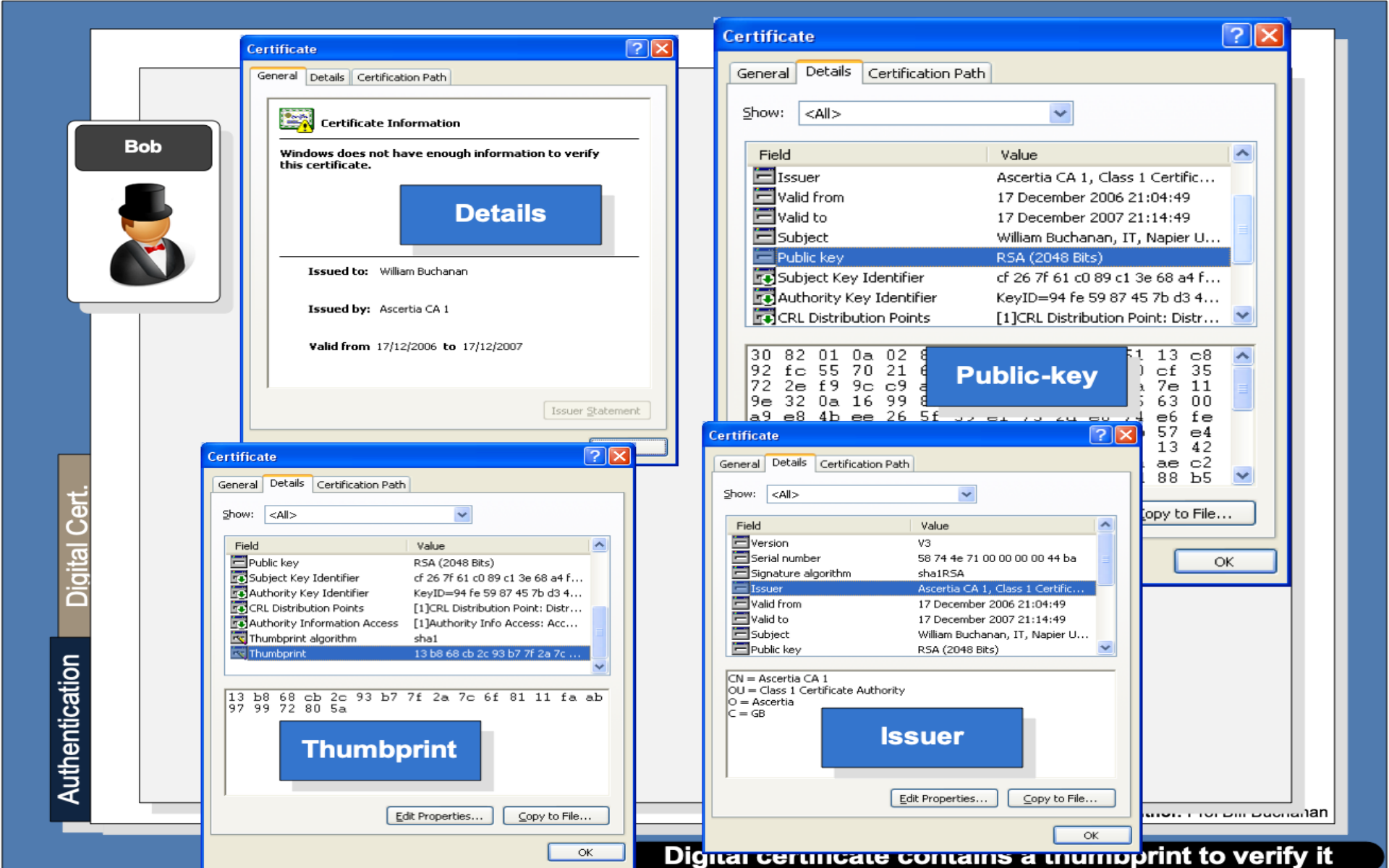
One method is the digital certificate which can carry the public key (and also the private key, if nesc.)

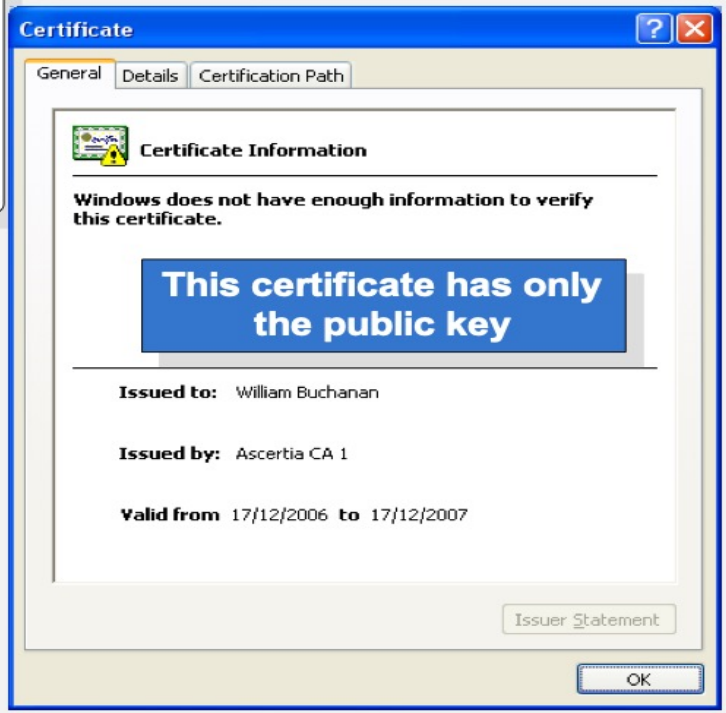
Now that we need the public key to either encrypt data for a receipiant, or to authenticate a sender...

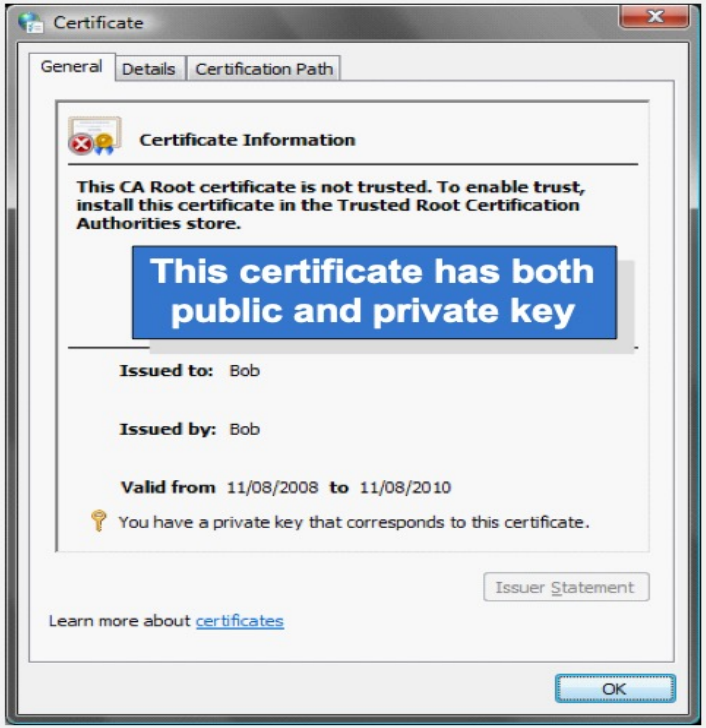
How does Bob distribute his public key to Alice, without having to post it onto a Web site or for Bob to be on -line when Alice reads the message?



How to store the private key and pass the public key ?


Bob
Certificate
General
Details
Certification Path
Certificate Information
Windows does not have enough information to verify this certificate.
Details
Issued to: William Buchanan
Issued by: Ascertia CA 1
Valid from 17/12/2006 to 17/12/2007
Issuer Statement
Certificate
General
Details
Certification Path
Show: <All>
Field
Value
Issuer
Ascertia CA 1, Class 1 Certific...
Valid from
17 December 2006 21:04:49
Valid to
17 December 2007 21:14:49
Subject
William Buchanan, IT, Napier U...
Public key
RSA (2048 Bits)
Subject Key Identifier
cf 26 7f 61 c0 89 c1 3e 68 a4 f...
Authority Key Identifier
KeyID=94 fe 59 87 45 7b d3 4...
CRL Distribution Points
[1]CRL Distribution Point: Distr...
30 82 01 0a 02
92 fc 55 70 21
72 2e f9 9c c9
9e 32 0a 16 99
a9 e8 4b ee 26 5f
Public-key
Copy to File...
OK
Certificate
General
Details
Certification Path
Show: <All>
Field
Value
Public key
RSA (2048 Bits)
Subject Key Identifier
cf 26 7f 61 c0 89 c1 3e 68 a4 f...
Authority Key Identifier
KeyID=94 fe 59 87 45 7b d3 4...
CRL Distribution Points
[1]CRL Distribution Point: Distr...
Authority Information Access
[1]Authority Info Access: Acc...
Thumbprint algorithm
sha1
Thumbprint
13 b8 68 cb 2c 93 b7 7f 2a 7c ...
13 b8 68 cb 2c 93 b7 7f 2a 7c 6f 81 11 fa ab
97 99 72 80 5a
Thumbprint
Edit Properties...
Copy to File...
OK
Certificate
General
Details
Certification Path
Show: <All>
Field
Value
Version
V3
Serial number
58 74 4e 71 00 00 00 00 44 ba
Signature algorithm
sha1RSA
Issuer
Ascertia CA 1, Class 1 Certific...
Valid from
17 December 2006 21:04:49
Valid to
17 December 2007 21:14:49
Subject
William Buchanan, IT, Napier U...
Public key
RSA (2048 Bits)
CN = Ascertia CA 1
OU = Class 1 Certificate Authority
O = Ascertia
C = GB
Issuer
Edit Properties...
Copy to File...
OK
Digital Cert.
Authentication
Digital certificate contains a thumbprint to verify it





Digital certificates should only be distributed with the public key

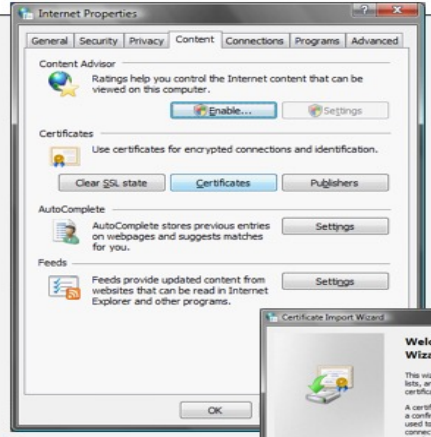
Digital Cert.

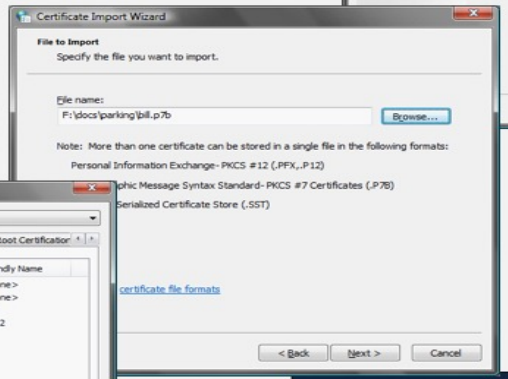
Authentication

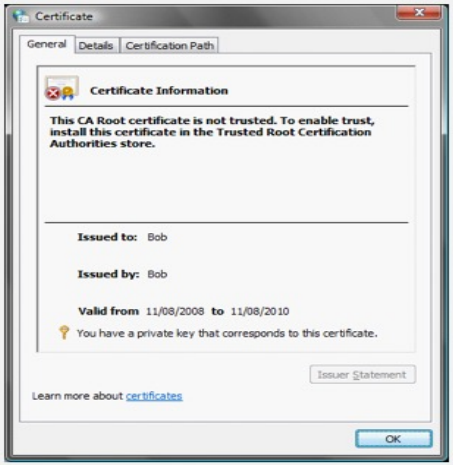
P7b format

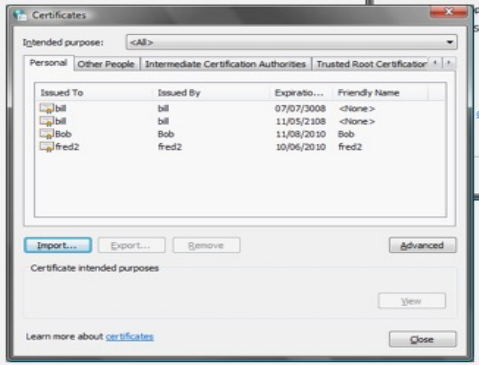
Bob

```
-----BEGIN CERTIFICATE-----
MIID2zCCA4WgAwIBAgIKWHROcQAAAABEujANBgkqhkiG 9w0BAQUFADBgMQswCQYD
VQQGEwJHQjERMA 8GA1UEChMIQXNjZXJ 0aWExJjAkBgNVBAsTHUNsYXNzIDEgQ 2Vy
dGlmaWNhdGUgQXV 0aG9yaXR 5MRYwFAYDVQQDEw 1Bc2NlcnRpYSBDQSAxMB 4XDTA2
MTIxNzIxMDQ 0OVoXDTA3MTIxNzIxMTQ 0OVowgZ 8xJjAkBgkqhkiG 9w0BCQEWF 3cu
YnVjaGFuYW 5AbmFwaWVyLmFjLnVrMQswCQYDVQQGEwJVSzEQMA 4GA1UECBMHTG 90
aGlhbjESMBAGA 1UEBxMJRWRpbmJ 1cmdoMRowGAYDVQQKExFOYXBpZXIgVW 5pdmVy
c2l0eTELMAkGA 1UECxMCSVQxGTAXBgNVBAMTEFdpbGxpYW 0gQnVjaGFuYW 4wggEi
MA0GCSqGSIb 3DQEBAQUAA 4IBDwAwggEKAoIBAQCvCFETyJL 8VXAhbEMRzQI 0gM81
ci75nMmsOamjZcb 6fhGeMgoWmYcoscmQkrVjAKnoS +4mXznhcy 3md0b+sZbwOVaX
M5FOxhSrV +Q86hsK 8cDc+1sqyJ 8TQtufuDNs 0NfNY 6tR6q 7CgGqQ 8/VjSxNqzK 39
iLUF1ahhyCet /ab60/qWzL 4iVsz 2nmL 4dyAuyilhLPlVbppHGdE 6sDQXwYd0CpfV
ZN7pauD 5fqBESfO 6bUkCieI 47AzRMQj 3kHuDt 7MexVW 7aoX+nXLP 4wn 7IamaxasF
QvhdOKyCZhYs 82JQDGatXRCqkklztmZW 5i 6GkPsE 7xVuX 265WJQ 5afhp 2hYlAgMB
AAGjggEXMIIBEzAdBgNVHQ 4EFgQUzyZ /YcCJwT 5opPHLPlcQKkOlkJwwYWYDVR 0j
BFwwWoAUlP 5Zh0V700k 6CorvRMWB 9ifVkBmhP 6Q9MDsxCzAJBgNVBAYTAkdCMREw
DwYDVQQKEwhBc 2NlcnRpYTEZMBcGA 1UEAxMQQXNjZXJ 0aWEgUm 9vdCBDQYIBDTBN
BgNVHR 8ERjBEMEKgQKA +hjxodHRwOi 8vd3d3LmFzY 2VydGlhLmNvbS 9PbmxpbmVD
QS 9jcmxzL 0FzY 2VydGlhQ 0ExL 2NsYXNzMS 5jcmwwPgYIKwYBBQUHAQEEMjAwMC 4G
CCsGAQUFBzAChiJodHRwOi 8vb2NzcC 5nbG 9iYWx 0cnVzdGZpbmRlci 5jb20vMA0G
CSqGSIb 3DQEBBQUAA 0EATOCwGJ 1tS0kTlupmpjkMl 8IdxMmD 5WuhszjBlGsMhPxI
H+vXhL 9yaOw+Prpzy 7ajS4/3xXU 8vRANhyU 9yU4qDA==
-----END CERTIFICATE-----
```









The main certificate formats include:

- P7b. Text format
- PFX/P12. Binary.
- SST. Binary.

Digital certificates should only be distributed with the public key

A. Bob creates the message.
B. Bob encrypts with Alice's public key and sends Alice the encrypted message
C. Alice decrypts with her private key
D. Alice receives the message

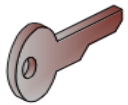
A

Encryption

Communications Channel

Decryption

Alice sends her digital certificate with her public key on it



Hello

B

C

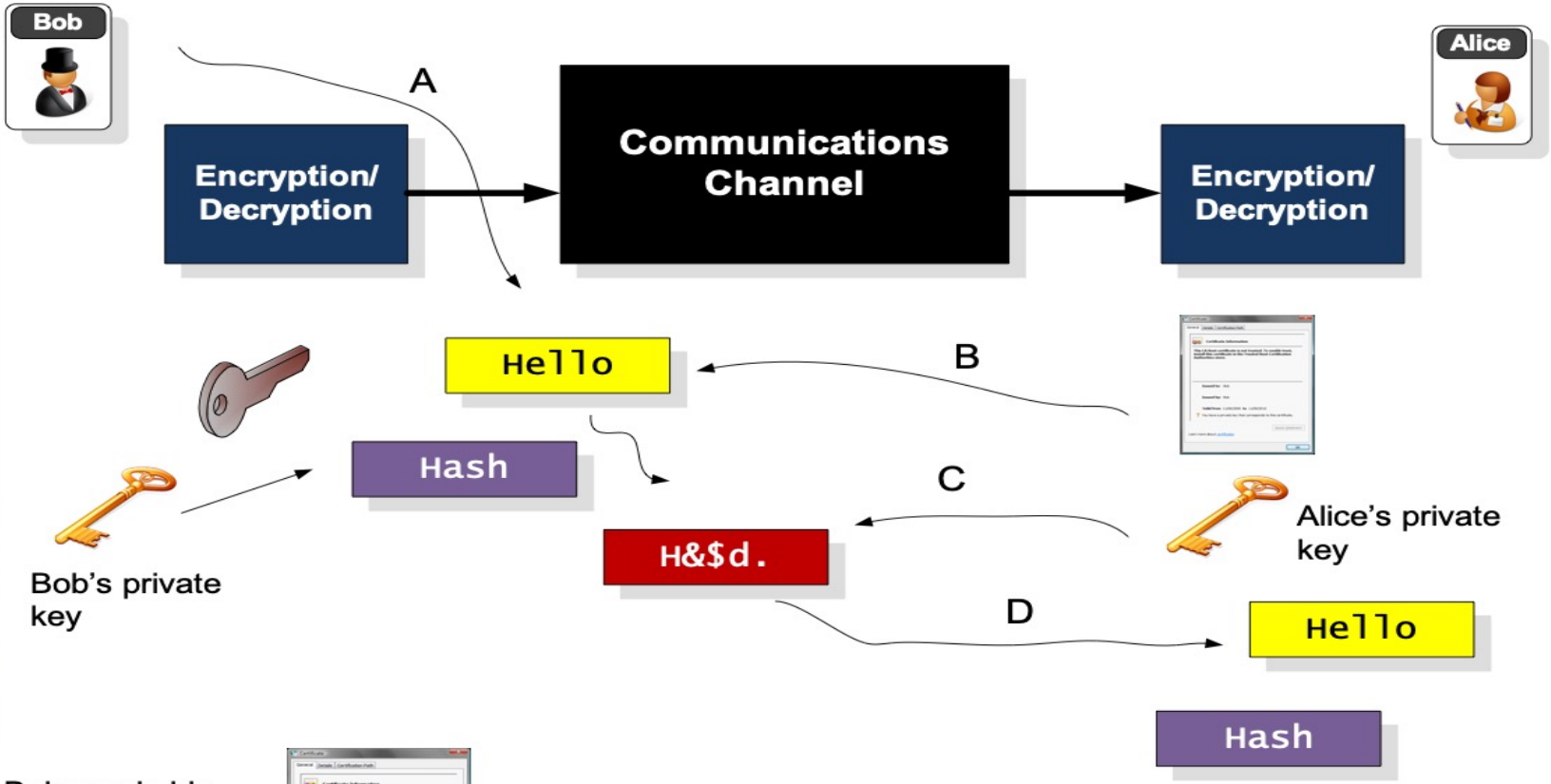
H&$d.

Alice's private key

D

Hello

Author: Prof Bill Buchanan



Bob sends his Digital certificate to authenticate himself

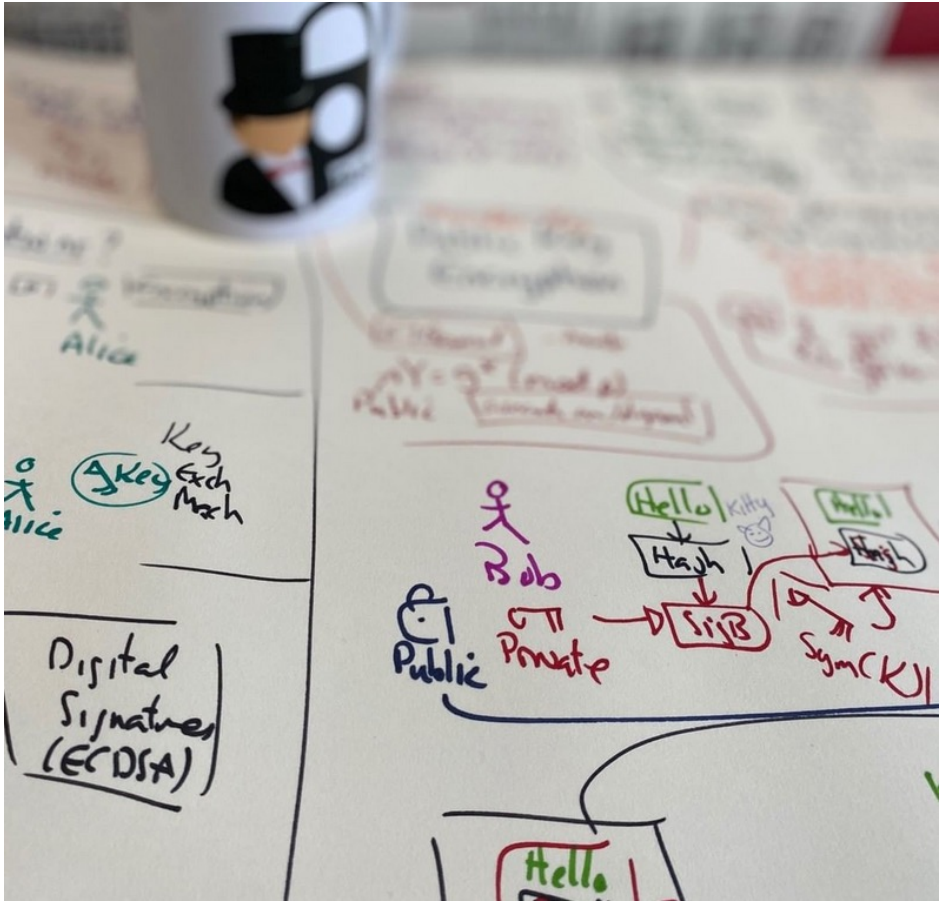
Alice checks the hash using Bob's public key from his certificate

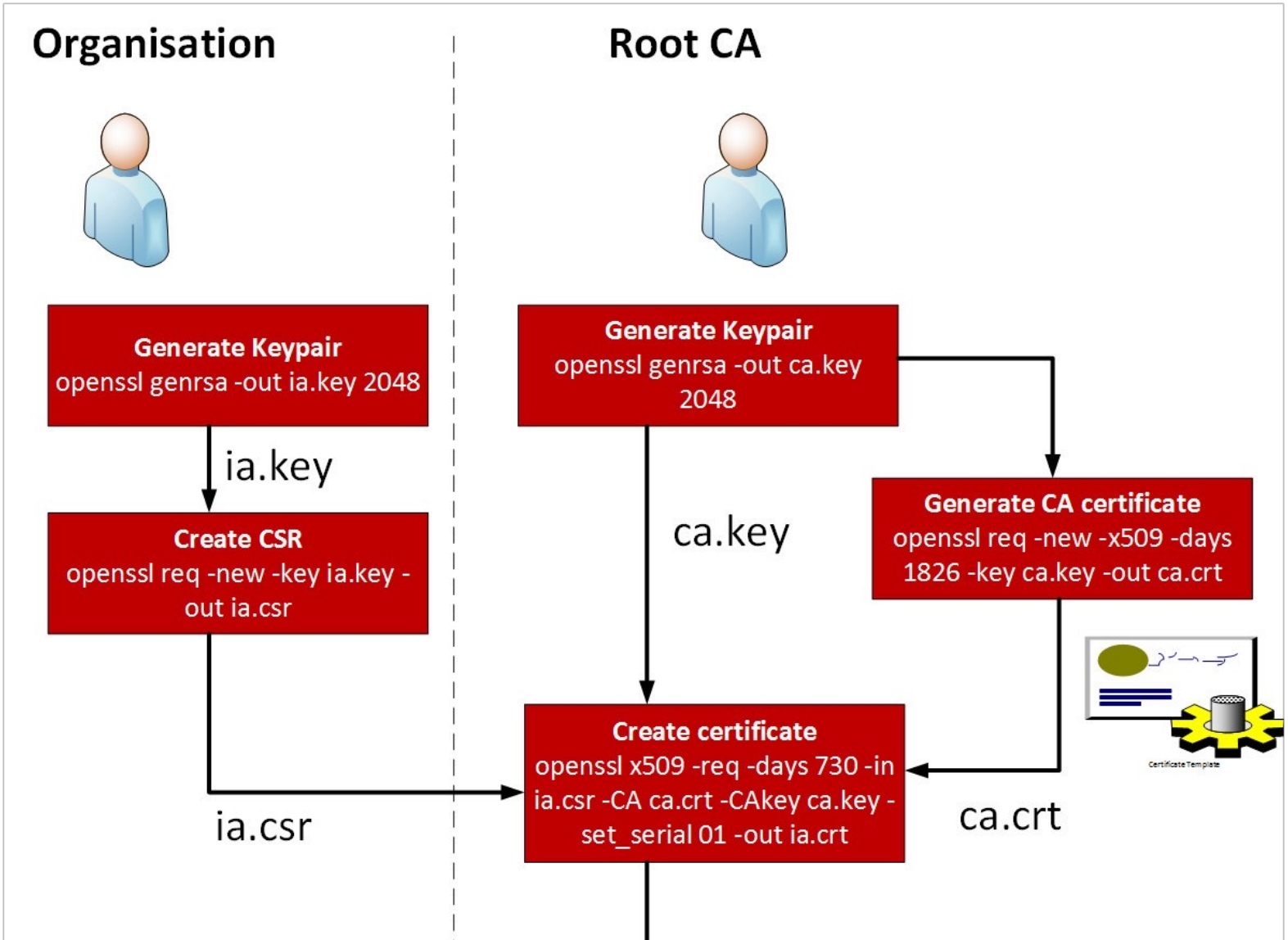
Author: Prof Bill Buchanan

Digital Certificates

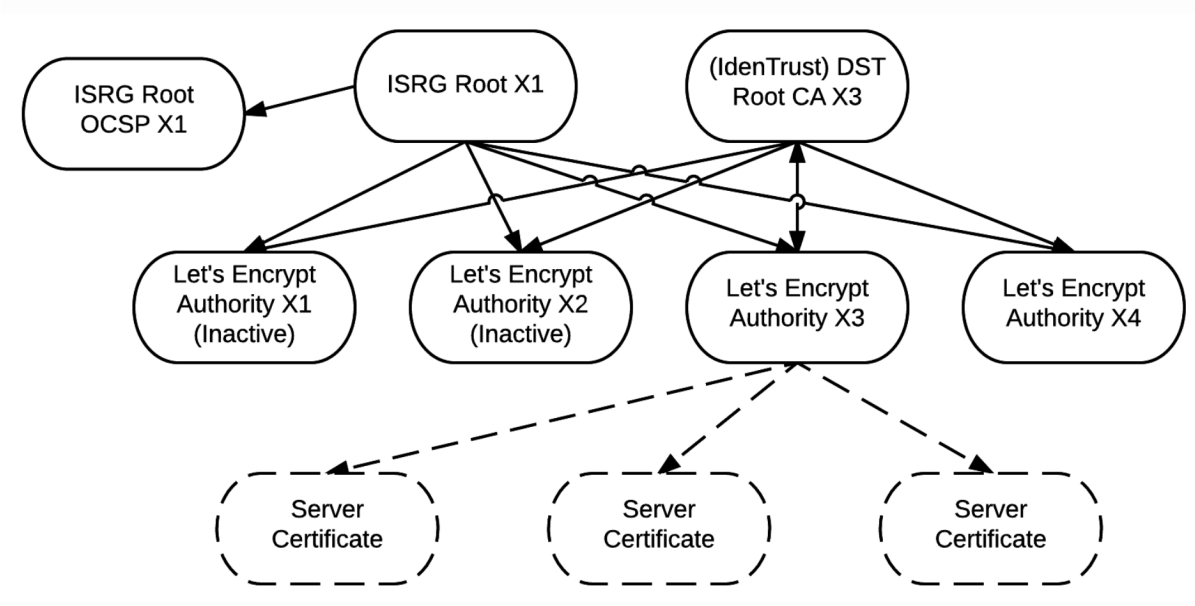
Introduction
Authentication Methods
PKI
Certificate Creation
Digital Certificate Passing

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




Organisation
Root CA
Generate Keypair
openssl genrsa -out ia.key 2048
ia.key
Create CSR
openssl req -new -key ia.key -out ia.csr
ia.csr
Generate Keypair
openssl genrsa -out ca.key 2048
ca.key
Generate CA certificate
openssl req -new -x509 -days 1826 -key ca.key -out ca.crt
ca.crt
Create certificate
openssl x509 -req -days 730 -in ia.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out ia.crt


SECURE THE WEB
Let's Encrypt


ISRG Root OCSP X1
ISRG Root X1
(IdenTrust) DST Root CA X3
Let's Encrypt Authority X1 (Inactive)
Let's Encrypt Authority X2 (Inactive)
Let's Encrypt Authority X3
Let's Encrypt Authority X4
Server Certificate
Server Certificate
Server Certificate

PEM

```
-----BEGIN CERTIFICATE-----
MIIEwTCCA6mgAwIBAgIRAO93GGFLf                    YJKoZIhvcNAQELBQAw
QjELMAkGA1UEBhMCVVMxHjAcBgNVBAoTFUdvb2dsZSBUcnVzdCBTZXJ2aWNlczET
MBEGA1UEAxMKR1RTIENBIDFPMTAeFw0yMDAyMTIxMTQ3NDFaFw0yMDA1MDYxMTQ3
NDFaMGgxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRYwFAYDVQQH
Ew1Nb3VudGFpbiBWaWV3MRMwEQYDVQQKEwpHb29nbGUgTExDMRcwFQYDVQQDEw53
d3cuZ29vZ2xlLmNvbTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABCzzLJYFJb8W
TpQxWL0TySDvfEWCkMe7l81CFspT60knl13YILNdTM22sUwPycyogKjBSaQZ9Axi
```

- X.509 Certificate (DER)
- X.509 Certificate (PKCS#7)

PKCS#1 v2 - Padding for Public Key

PKCS #7 v1.5 - Cryptography Message Syntax

PKCS 10 v1.7 - Certificate Request Standard.

```
tlHtqVqITZPCAI/ANvbHdZiMEePB8eA+oTiw2ucChqLlsop5Mio8ckg7aXG4/QfC
AIDbvkoffK44rs4UEpsaqGe9QqMJjTuO7ZCZrFd1m3geq2ARKPHgPoPM6UbDdqgO
TNQo9C0F5ZlOkOgV/qshvpTO4YzE7TB2U571eYEqNXH48syVx8XSk3P7FjM7FIZ2
IzbJSEipZJm8DsP10fFXpToIn+zK
-----END CERTIFICATE-----
```

PKCS#7

```
-----BEGIN PKCS7-----
MIIE8gYJKoZIhvcNAQcCoIIE4zC                    9w0BBwGgggTF
wTCCA6mgAwIBAgIRAO93GGFLfHwOCAAAAAAucZgwDQYJKoZIhvcNAQELBQAw
MAkGA1UEBhMCVVMxHjAcBgNVBAoTFUdvb2dsZSBUcnVzdCBTZXJ2aWNlczET
A1UEAxMKR1RTIENBIDFPMTAeFw0yMDAyMTIxMTQ3NDFaFw0yMDA1MDYxMTQ3
MGgxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRYwFAYDVQQH
b3VudGFpbiBWaWV3MRMwEQYDVQQKEwpHb29nbGUgTExDMRcwFQYDVQQDEw53
Z29vZ2xlLmNvbTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABCzzLJYFJb8W
WL0TySDvfEWCkMe7l81CFspT60knl13YILNdTM22sUwPycyogKjBSaQZ9Axi
EAiJkbejggJVMIICUTAOBgNVHQ8BAf8EBAMCB4AwEwYDVR0lBAwwCgYIKwYB
AwEwDAYDVR0TAQH/BAIwADAdBgNVHQ4EFgQU9Ty3t90o7UW9+Hc6kv/j9511
HwYDVR0jBBgwFoAUmNH4bhDrz5vsYJ8YkBug630J/SswZAYIKwYBBQUHAQEE
MCcGCCsGAQUFBzABhhtodHRwOi8vb2NzcC5wa2kuZ29vZy9ndHMxbzEwKwYI
BQUHMAKGH2h0dHA6Ly9wa2kuZ29vZy9nc3IyL0dUUzFPMS5jcnQwGQYDVR0R
EIIOd3d3Lmdvb2dsZS5jb20wIQYDVR0gBBowGDAIBgZngQwBAgIwDAYKKwYB
eQIFAzAvBgNVHR8EKDAmMCSgIqAghh5odHRwOi8vY3JsLnBraS5nb29nL0dU
MS5jcmwwggEFBgorBgEEAdZ5AgQCBIH2BIHzAPEAdgCyHgXMi6LNiiBOh2b5
JSBna9r6cOeySVMt74uQXgAAAXA5cNqwAAAEAwBHMEUCIQCojKtzOe8l1JYK
bt1puqT4AE3peMVAVk/WeWGp1QIgBvRMbNkwF6i8+JVv3CfTHKvFd8e00+GY
Zb1drKEAdwBep3P531bA57U2SH3QSeAyepGaDIShEhKEGHWWgXFFWAAAAXA5
AAAEAwBIMEYCIQCQSONppELPODh65oFT5ZAGsSQz4FsjNNZedgS7WqzLnQIh
u0QUuoE3RWngfhI6W7n1kxkEmd0vSZTe6+Ql+JVuMA0GCSqGSIb3DQEBCwUA
AQCDU8CVusGstVk1lAmrtg3DyYhNV1UnKyLk3RrHKumwYz5mC25bWGoLVKvv
N3zA329/9JZq0mnn0vmPMxjTDvh9sVQ7g4znwha/KJfzL8AZKUdldD9O+hDC
qVqITZPCAI/ANvbHdZiMEePB8eA+oTiw2ucChqLlsop5Mio8ckg7aXG4/QfC
vkoffK44rs4UEpsaqGe9QqMJjTuO7ZCZrFd1m3geq2ARKPHgPoPM6UbDdqgO
9C0F5ZlOkOgV/qshvpTO4YzE7TB2U571eYEqNXH48syVx8XSk3P7FjM7FIZ2
SEipZJm8DsP10fFXpToIn+zKoQAxAA==
-----END PKCS7-----
```

Base64

```
openssl x509 -outform der -in www-google-com.pem -out google.crt
openssl pkcs12 -export -in server.pem -out keystore.pkcs12
```

Binary

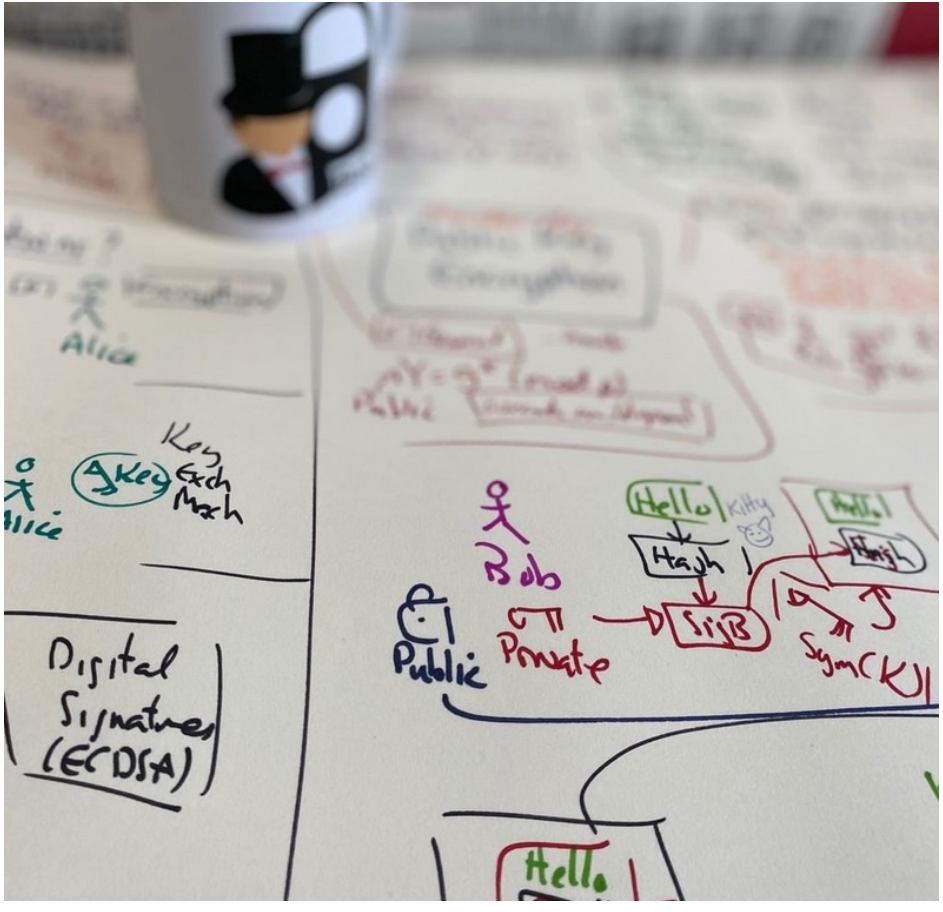
DER
CER

Digital Certificates

Introduction
Authentication Methods
PKI
Digital Certificate Passing

Prof Bill Buchanan OBE

http://asecuritysite.com/crypto06
http://asecuritysite.com/encryption



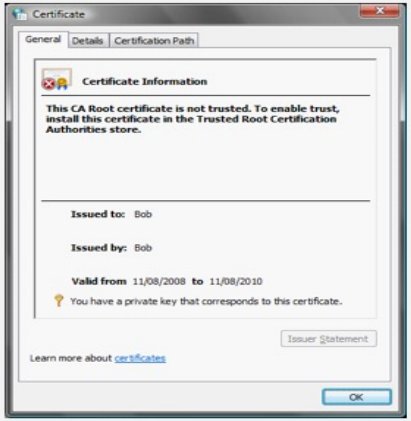
Digital Certificates

Digital certificates are a soft token of authentication, and require a trust mechanism.

Eve

Bob

Alice



Trust

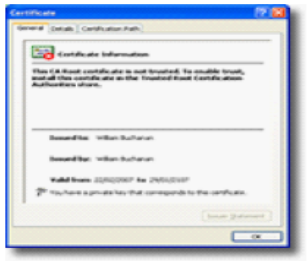
Who do we trust to get Bob's certificate … we can't trust Bob, as he may be Eve… meet Trent.

Who can we trust to get the digital certificate from ?

Trusted Root CA

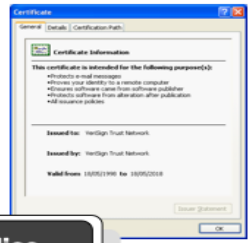
Certificate Authority (CA) - Able to grant certificates Examples; Verisign, Entrust, Microsoft Trust.

The Trusted Root CE (Trent) checks Bob's identity and creates a certificate which he signs



Trusted root certificates are installed as a default on the machine (or installed with the user's permission)

Trusted root certificate



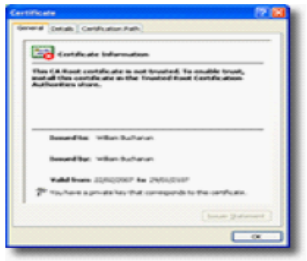
Alice checks the signature of the certificate to validate Bob. Both Alice and Bob trust the CA (Trent) as a third party.

Author: Prof Bill Buchanan

Trusted Root CA

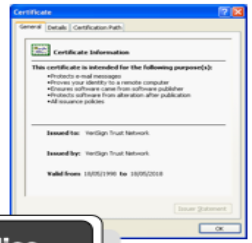
Certificate Authority (CA)
- Able to grant certificates
Examples; Verisign, Entrust, Microsoft Trust.

Eve tricks the CA to get a certificate with Bob's name



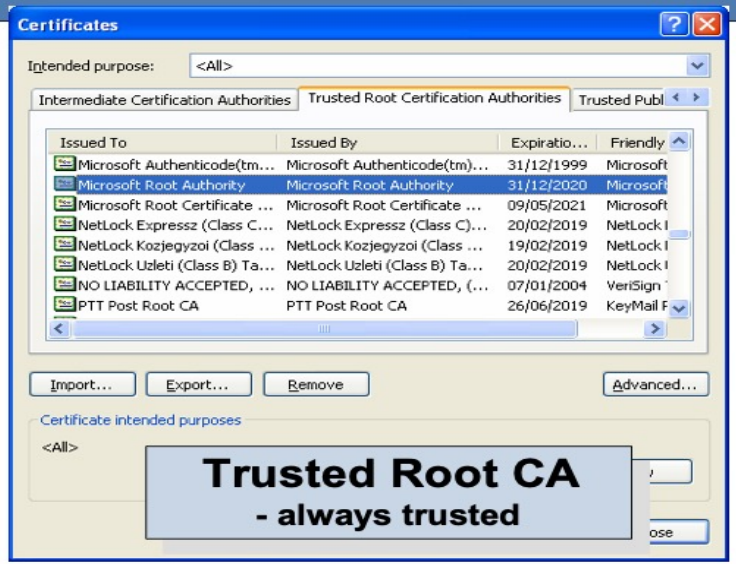
Trusted root certificates are installed as a default on the machine (or installed with the user's permission)

Trusted root certificate



Alice checks the signature of the certificate to validate Bob. Both Alice and Bob trust the CA (Trent) as a third party.

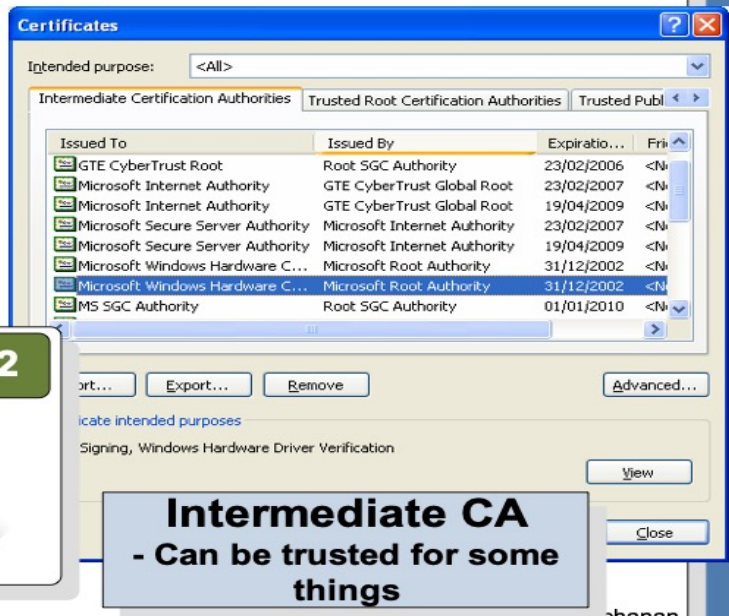
Author: Prof Bill Buchanan



Trusted Root CA

Certificate purposes:

- Secure email.
- Server authentication.
- Code signing.
- Driver authentication.
- Time stamping.
- Client authentication.
- IP tunnelling.
- EFS (Encrypted File System).

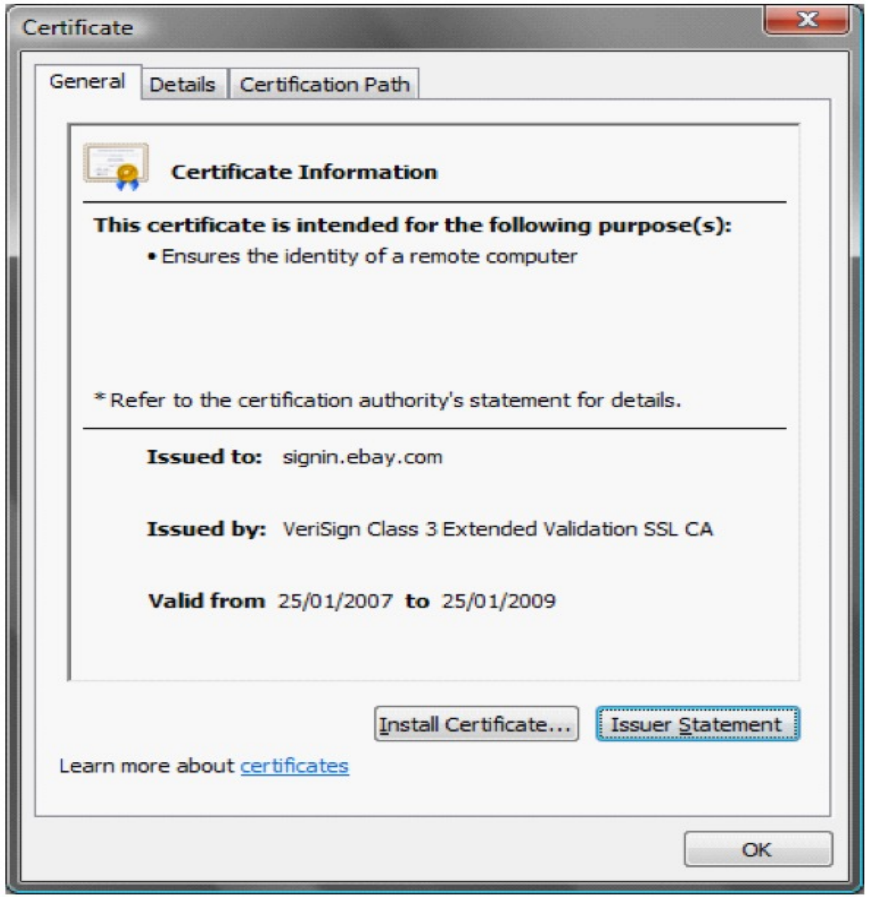


The two main problems with digital certificates are:

- Lack of understanding of how they work.
- They can be spoofed.

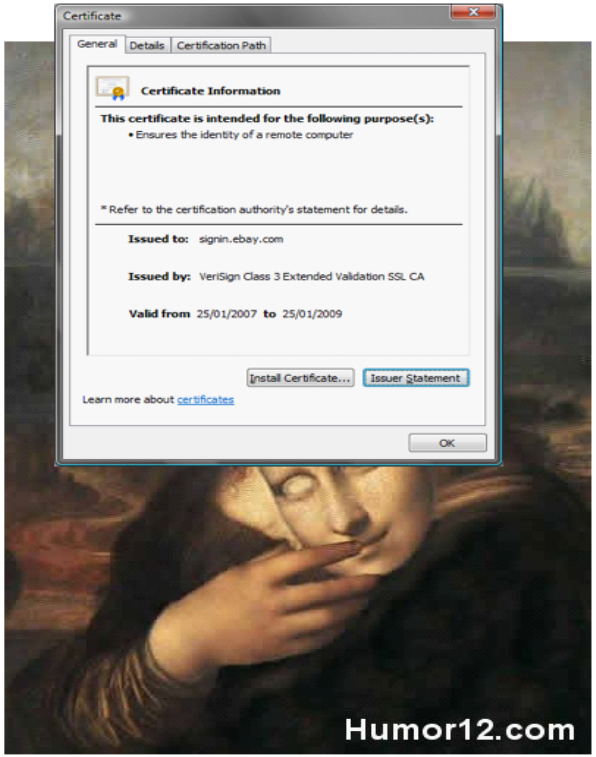
So let's look at a few … are they real or fake?

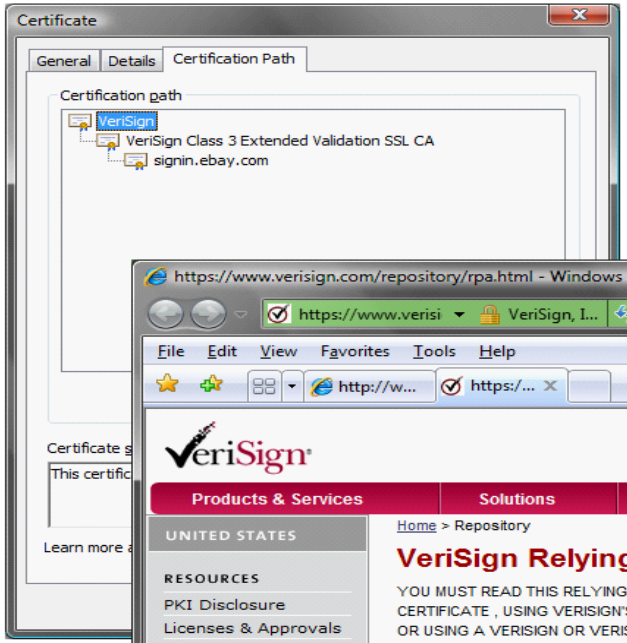
Author: Prof Bill Buchanan



Real or fake?

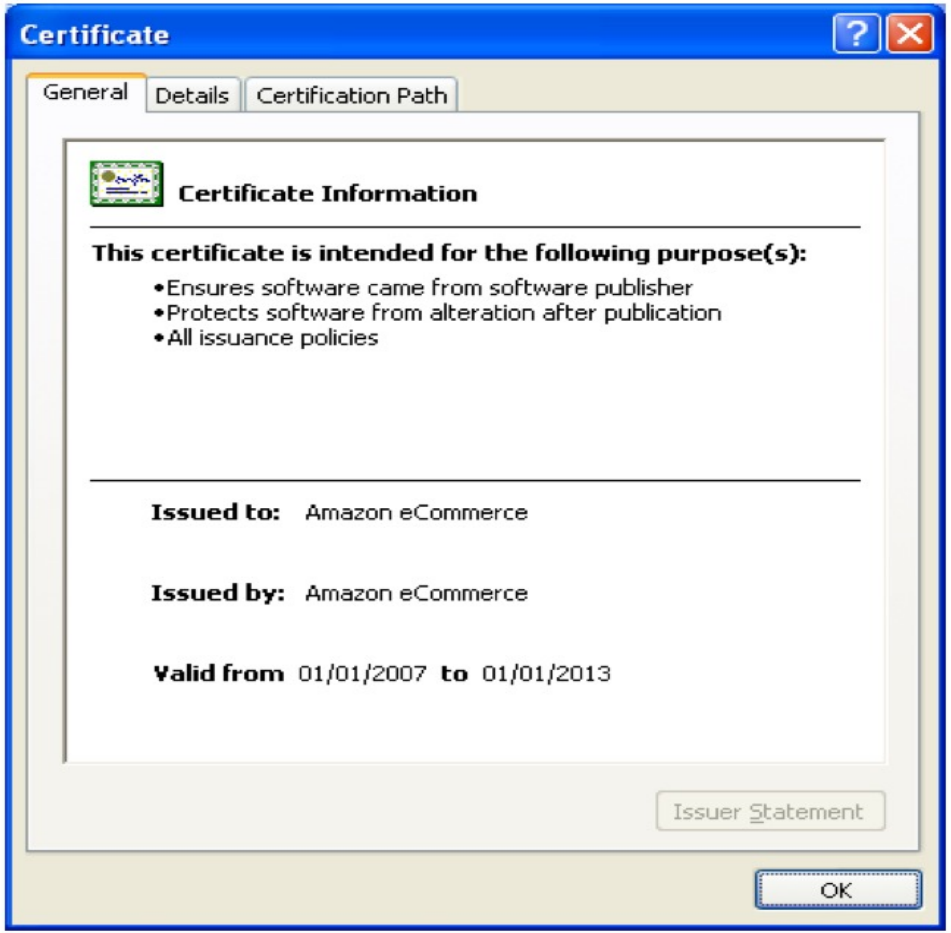
Author: Prof Bill Buchanan





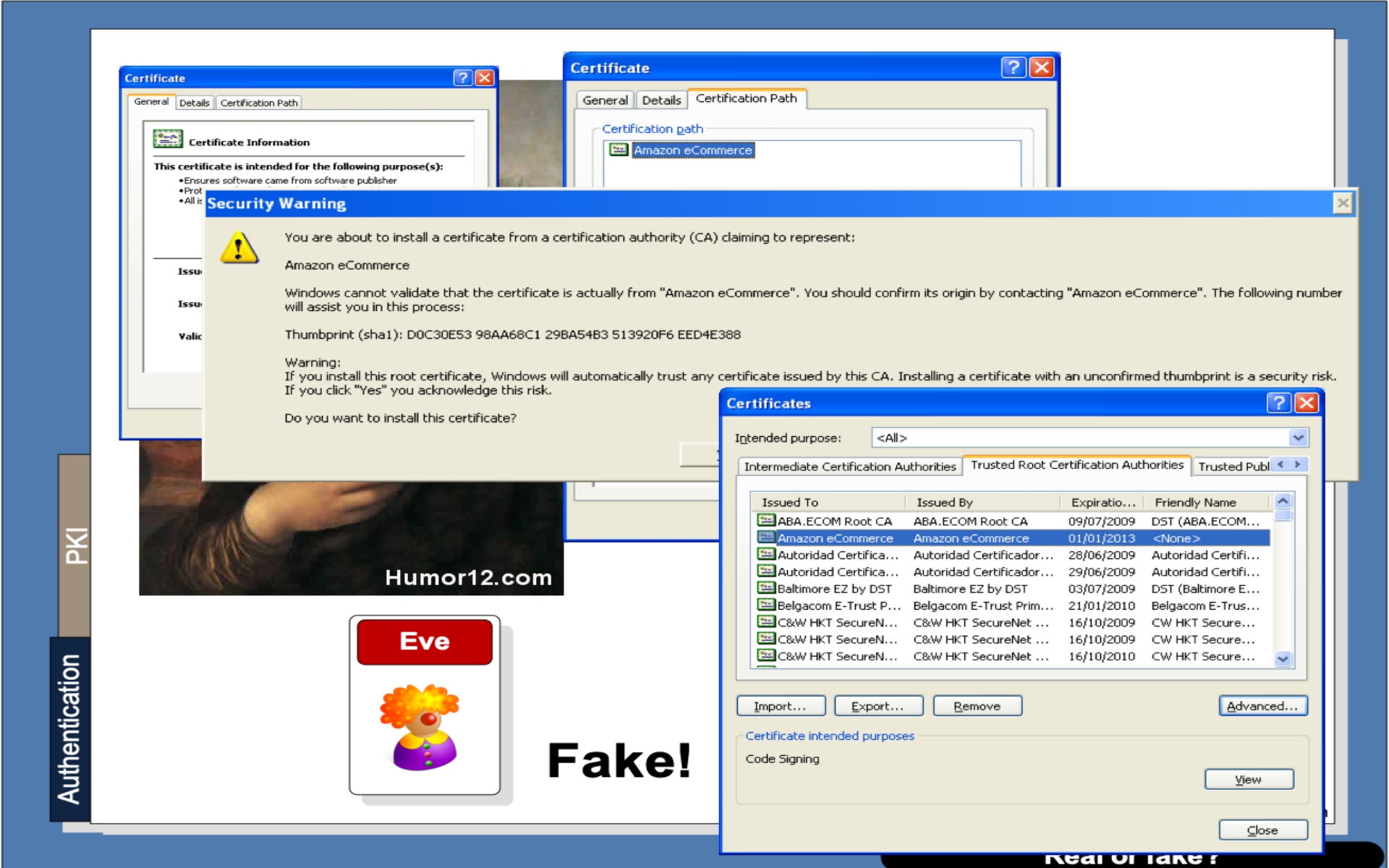
Real!

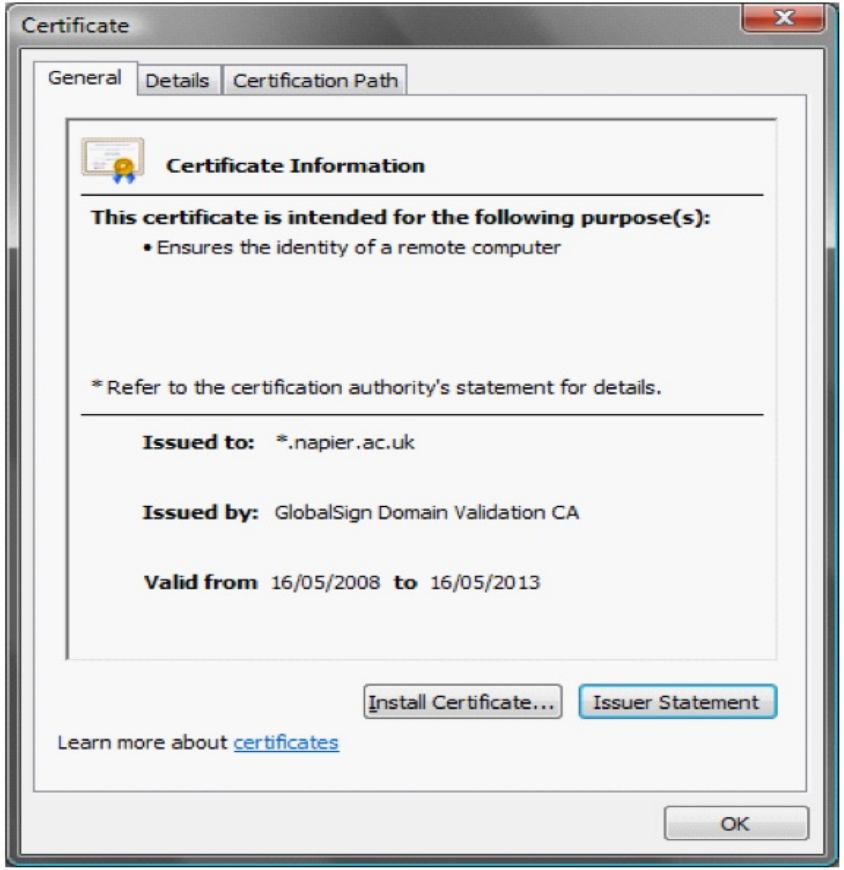
Author: Prof Bill Buchanan



Real or fake?

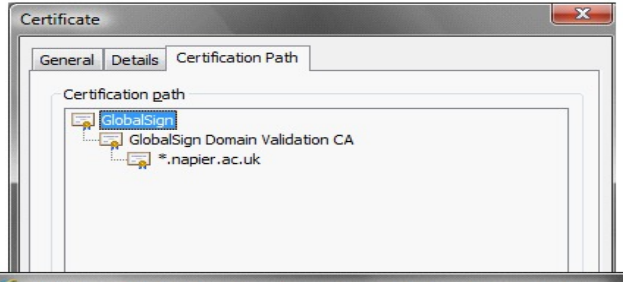
Author: Prof Bill Buchanan

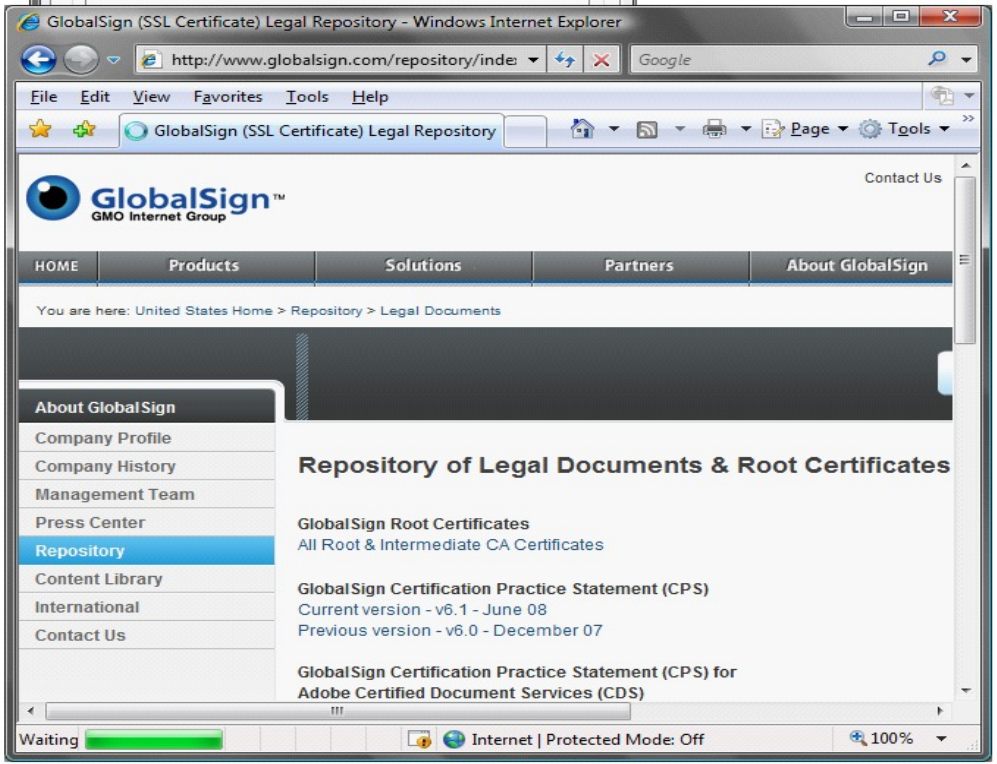

Certificate
General
Details
Certification Path
Certificate Information
This certificate is intended for the following purpose(s):
•Ensures software came from software publisher
Certificate
General
Details
Certification Path
Certification path
Amazon eCommerce
Security Warning
You are about to install a certificate from a certification authority (CA) claiming to represent:
Amazon eCommerce
Windows cannot validate that the certificate is actually from "Amazon eCommerce". You should confirm its origin by contacting "Amazon eCommerce". The following number will assist you in this process:
Thumbprint (sha1): D0C30E53 98AA68C1 29BA54B3 513920F6 EED4E388
Warning:
If you install this root certificate, Windows will automatically trust any certificate issued by this CA. Installing a certificate with an unconfirmed thumbprint is a security risk. If you click "Yes" you acknowledge this risk.
Do you want to install this certificate?
Humor12.com
Certificates
Intended purpose:
<All>
Intermediate Certification Authorities
Trusted Root Certification Authorities
Trusted Publ
Issued To
Issued By
Expiratio...
Friendly Name
ABA.ECOM Root CA
ABA.ECOM Root CA
09/07/2009
DST (ABA.ECOM...
Amazon eCommerce
Amazon eCommerce
01/01/2013
<None>
Autoridad Certifica...
Autoridad Certificador...
28/06/2009
Autoridad Certifi...
Autoridad Certifica...
Autoridad Certificador...
29/06/2009
Autoridad Certifi...
Baltimore EZ by DST
Baltimore EZ by DST
03/07/2009
DST (Baltimore E...
Belgacom E-Trust P...
Belgacom E-Trust Prim...
21/01/2010
Belgacom E-Trus...
C&W HKT SecureN...
C&W HKT SecureNet ...
16/10/2009
CW HKT Secure...
C&W HKT SecureN...
C&W HKT SecureNet ...
16/10/2009
CW HKT Secure...
C&W HKT SecureN...
C&W HKT SecureNet ...
16/10/2010
CW HKT Secure...
Import...
Export...
Remove
Advanced...
Certificate intended purposes
Code Signing
View
Close
PKI
Authentication
Eve
Fake!



Real or fake?

Author: Prof Bill Buchanan





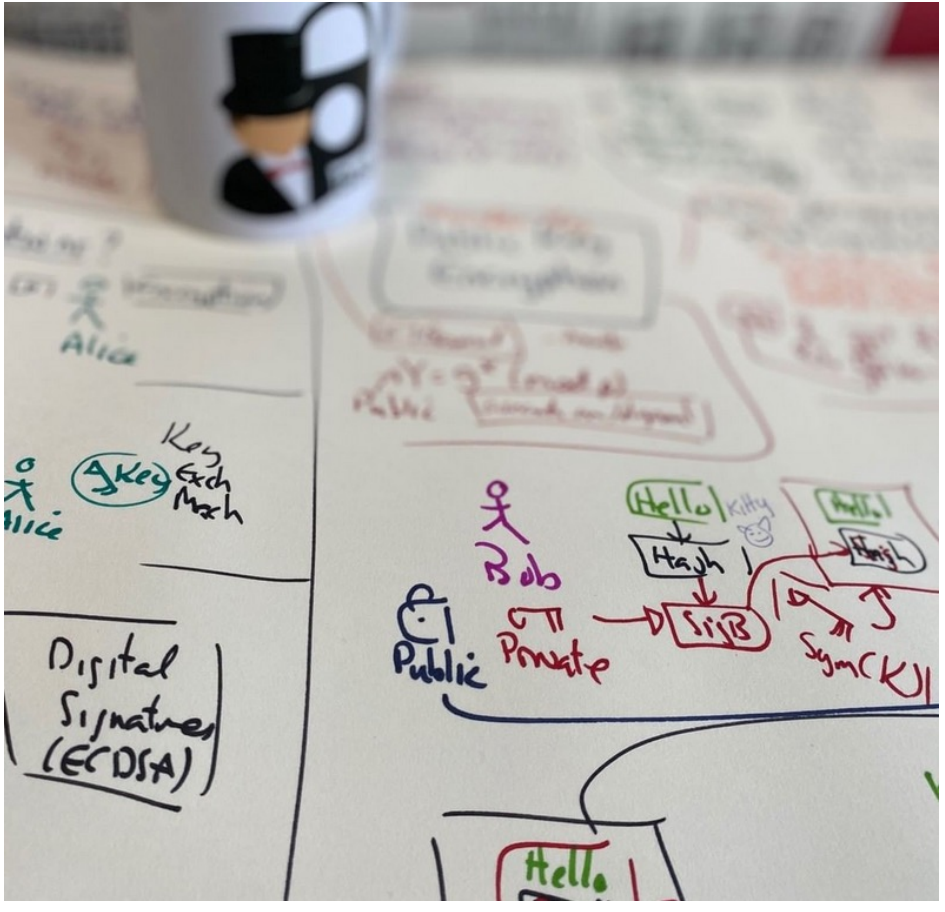
Real

Author: Prof Bill Buchanan

Chapter 6: Digital Certificates

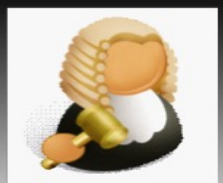
Introduction
Authentication Methods
PKI
Digital Certificate Passing

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



Public key encryption … secret … identity … trust

Eve

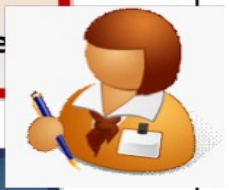


Trent

MegaCorp

ob's Private Key

Alice's Public Ke

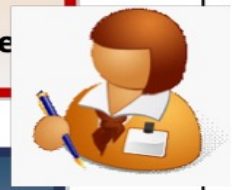


Bob's Public Key

Alice's Private Key

Public key encryption … secret … identity … trust

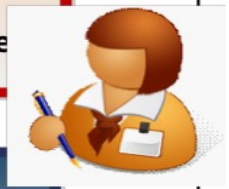
Public key encryption … secret … identity … trust



Public key encryption … secret … identity … trust

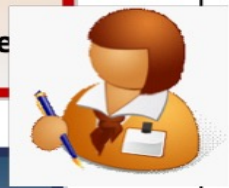
Hello Alice,
Wish you were here!
- Bob

Bob.



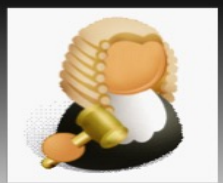
Public key encryption … secret … identity … trust

Hello Alice,
Wish you were here!
- Bob



Public key encryption … secret … identity … trust

Eve



Trent

MegaCorp

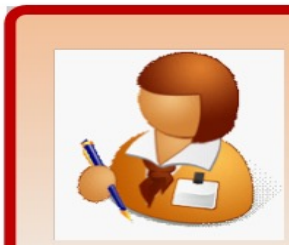
Alice's Public Key

ob's Private Key

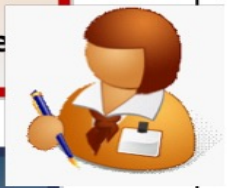
Alice's Public Key

Hello Alice,
Wish you were here!
- Bob

Bob.



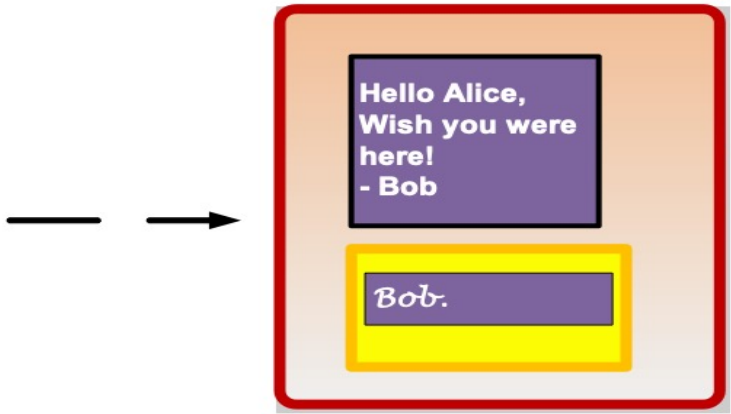
Alice's Public Ke



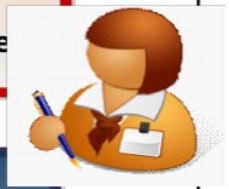
Bob's Public Key

Alice's Private Key

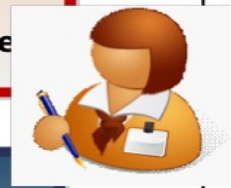
Public key encryption … secret … identity … trust

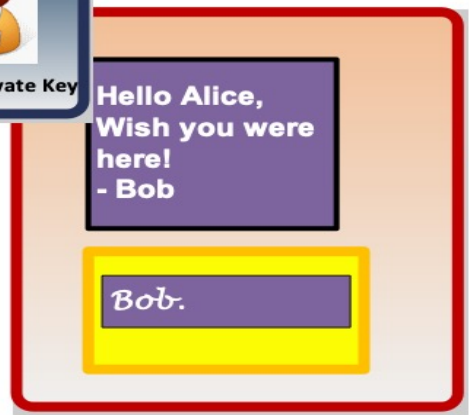


Which key to open the message?



Public key encryption … secret … identity … trust



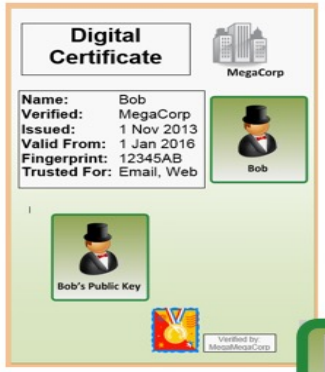


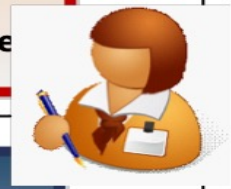
Which key to open the message?

Public key encryption … secret … identity … trust

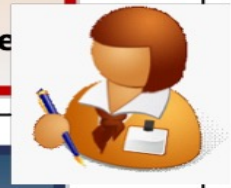
Which key to we open the signature with?

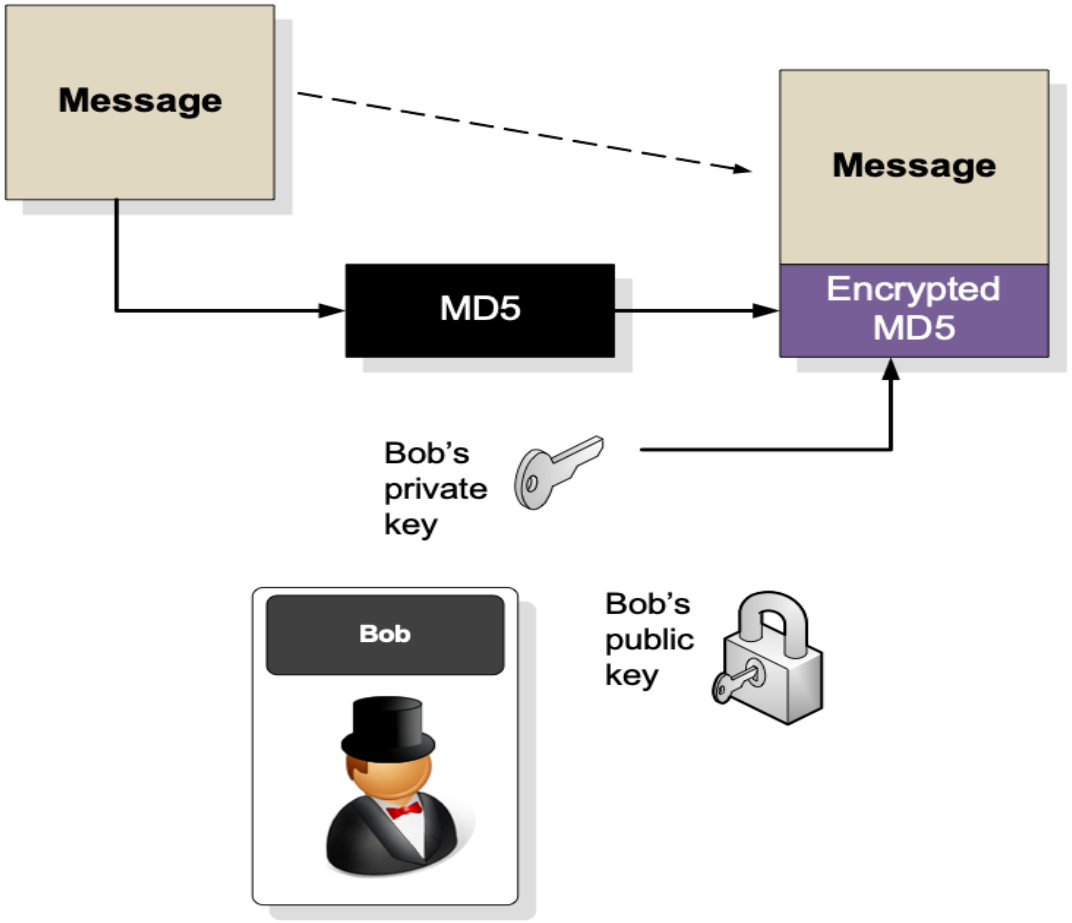
Public key encryption … secret … identity … trust



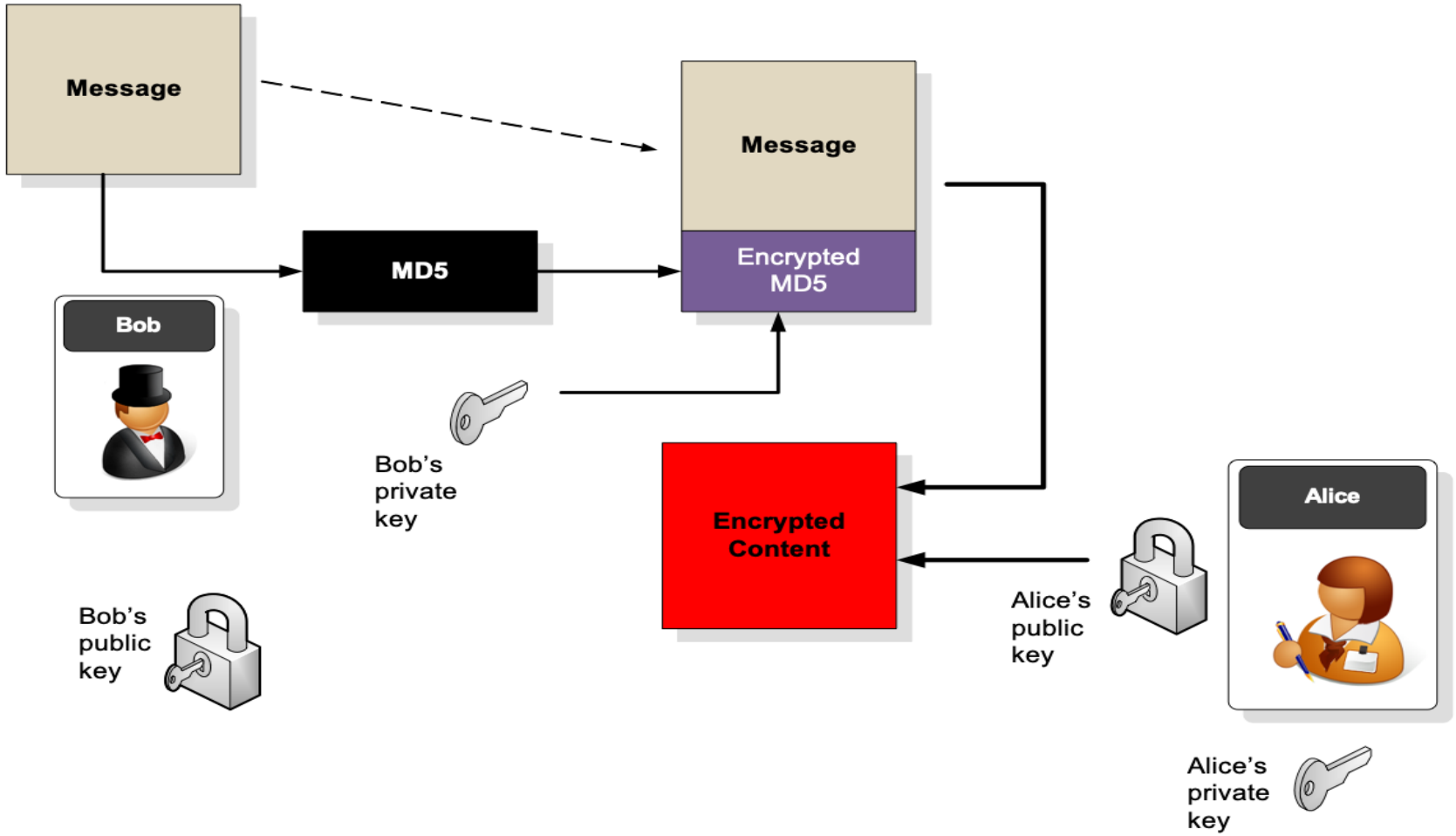


Public key encryption … secret … identity … trust



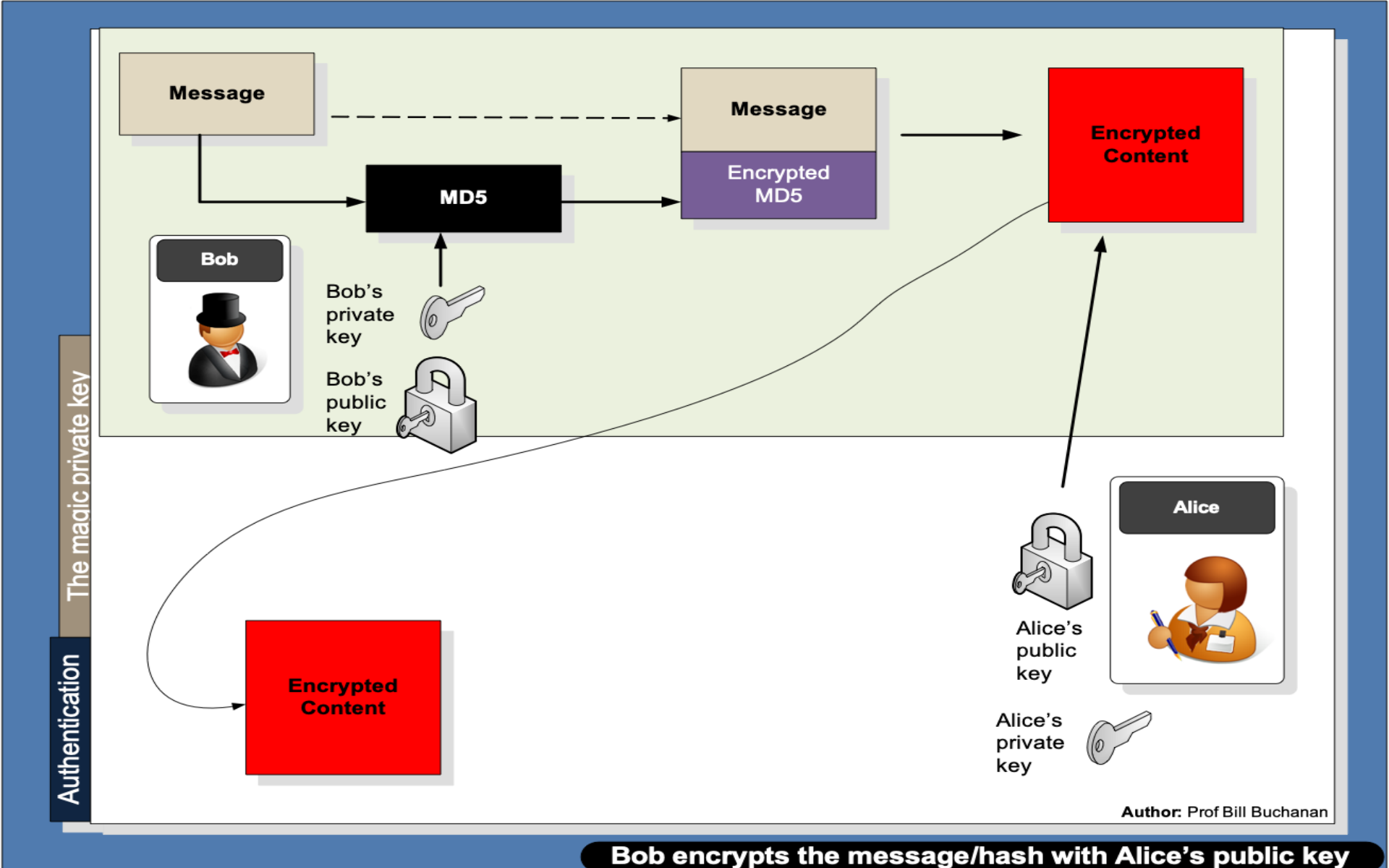


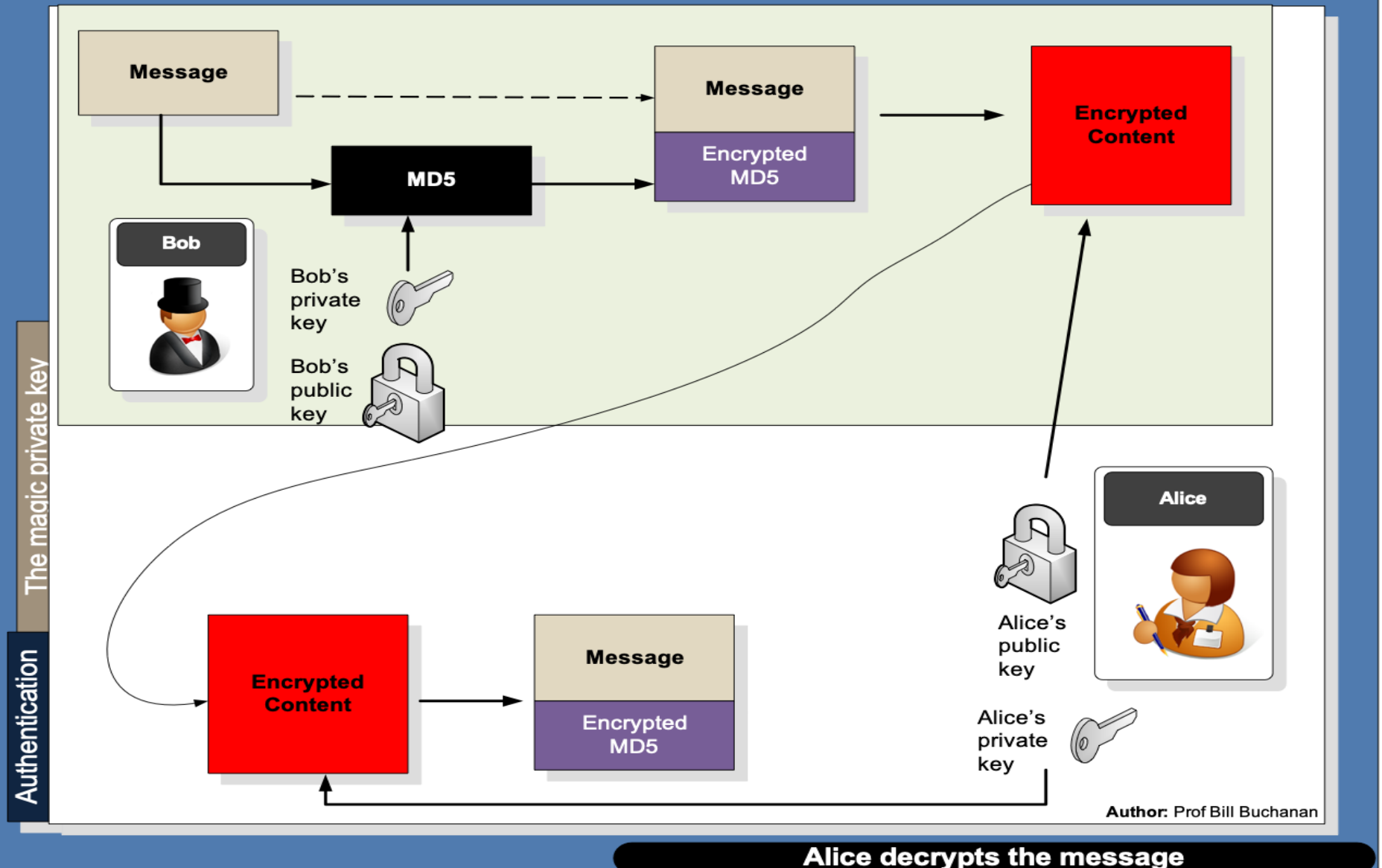
Author: Prof Bill Buchanan



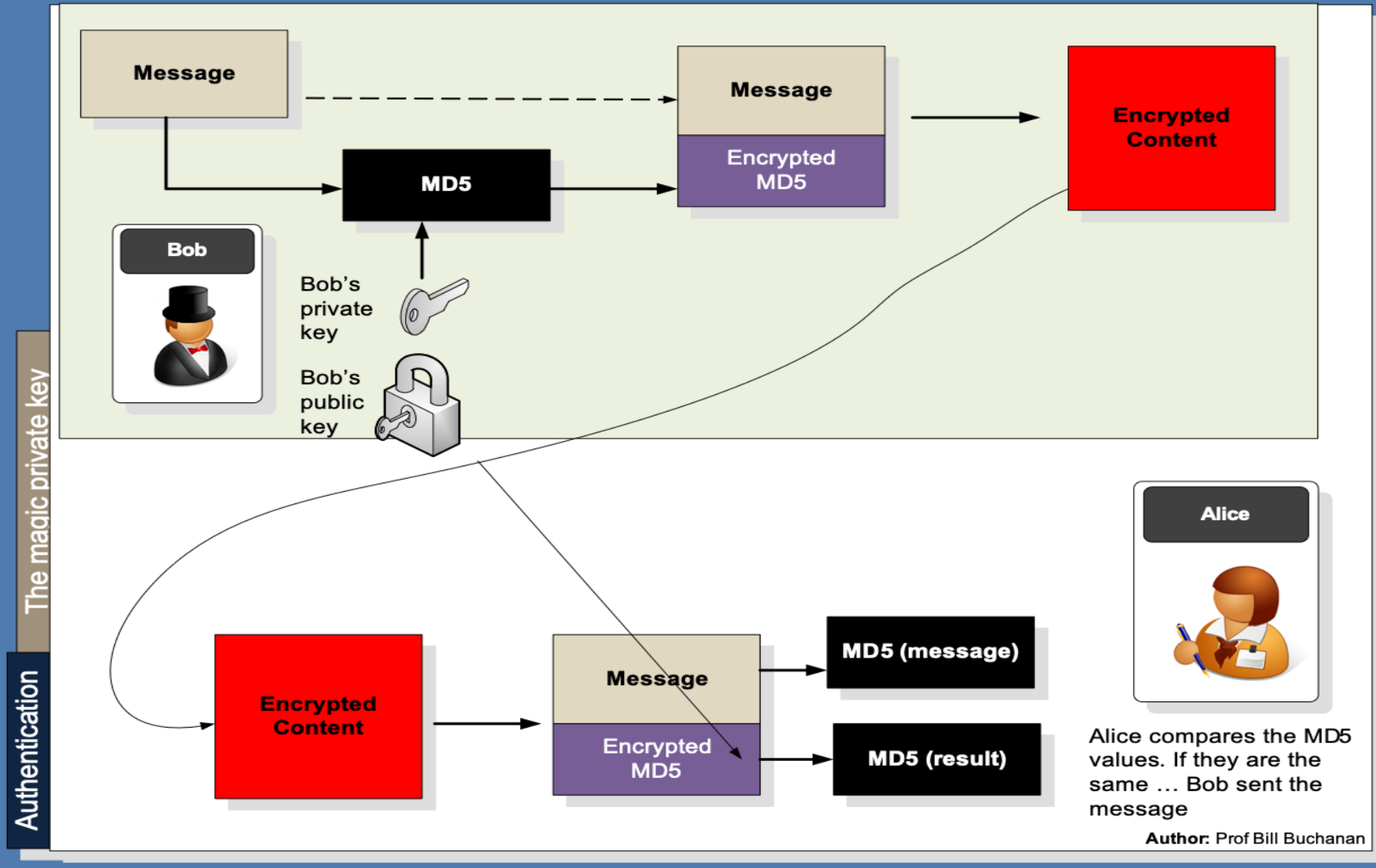
Author: Prof Bill Buchanan

Bob encrypts the message/hash with Alice's public key


Message
Message
Encrypted MD5
MD5
Encrypted Content
Bob
Bob's private key
Bob's public key
The magic private key
Authentication
Encrypted Content
Alice
Alice's public key
Alice's private key
Author: Prof Bill Buchanan
Bob encrypts the message/hash with Alice's public key



Alice decrypts the message



Alice decrypts the message

Digital Certificates

Introduction
Authentication Methods
PKI
Digital Certificate Passing

Prof Bill Buchanan OBE

https://asecuritysite.com/tunnelling