

Symmetric Key

Basics

Block or Stream?

Secret Key Methods

Salting

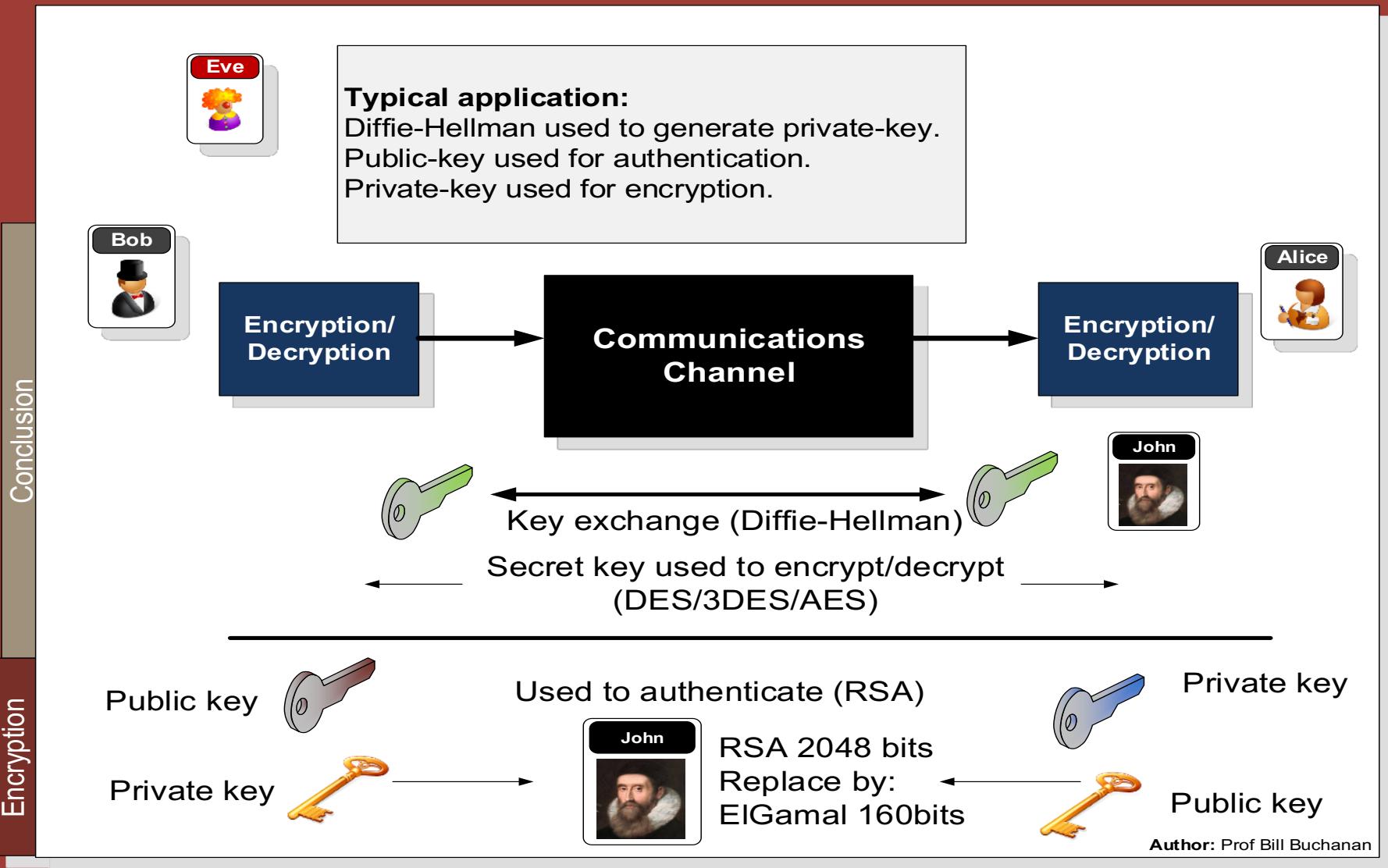
AES

Key Entropy

Prof Bill Buchanan OBE

<https://asecuritysite.com/symmetric>





Secret Key

Basics

Block or Stream?

Secret Key Methods

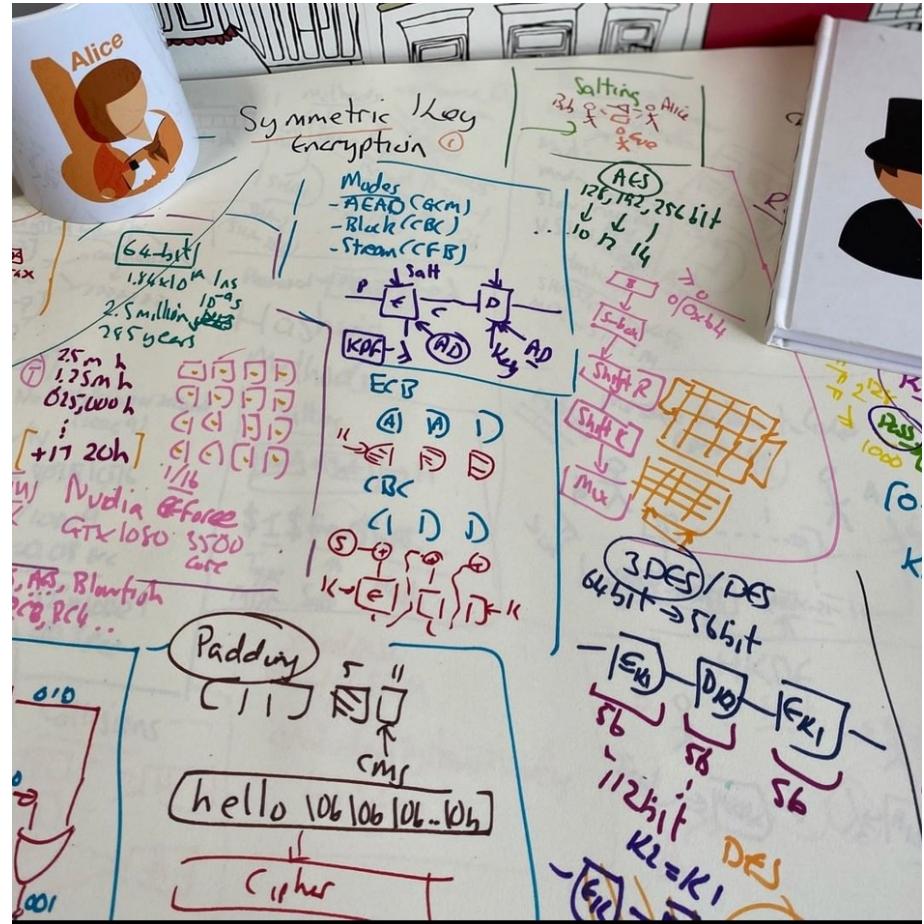
Salting

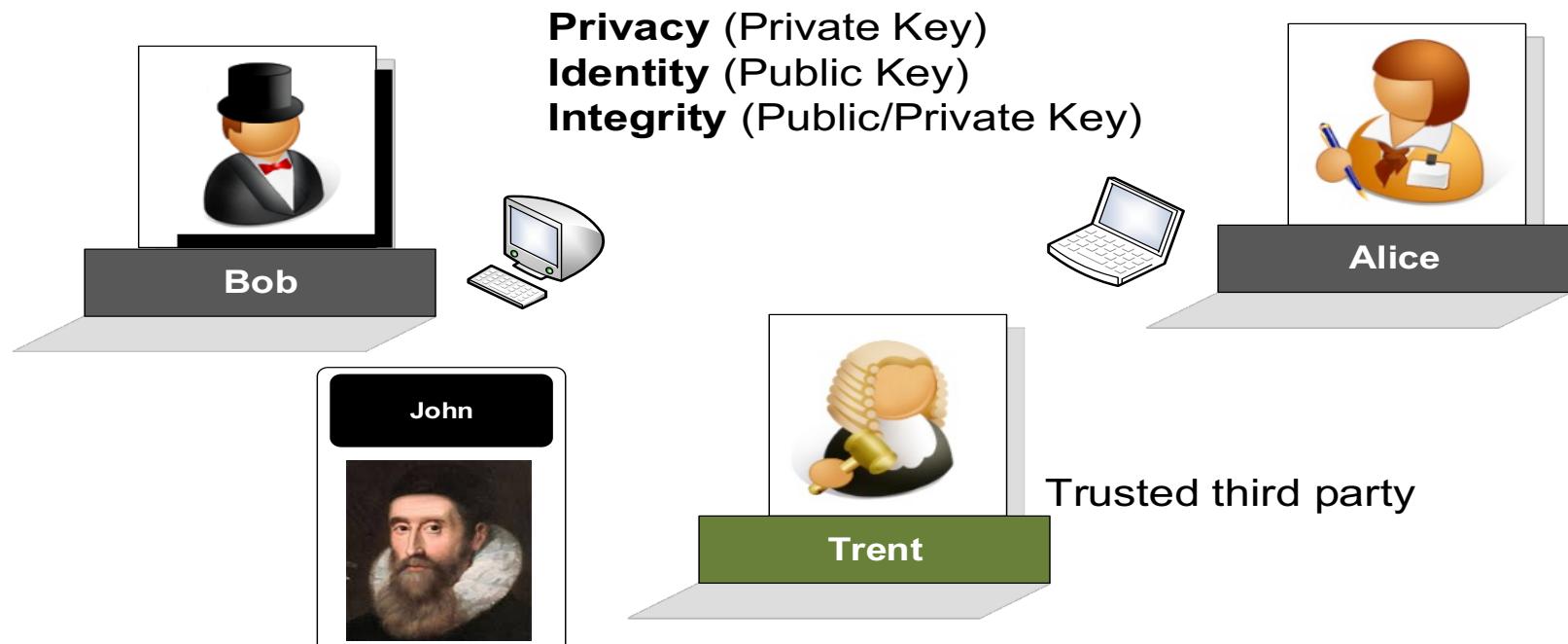
AES

Key Entropy

Prof Bill Buchanan OBE

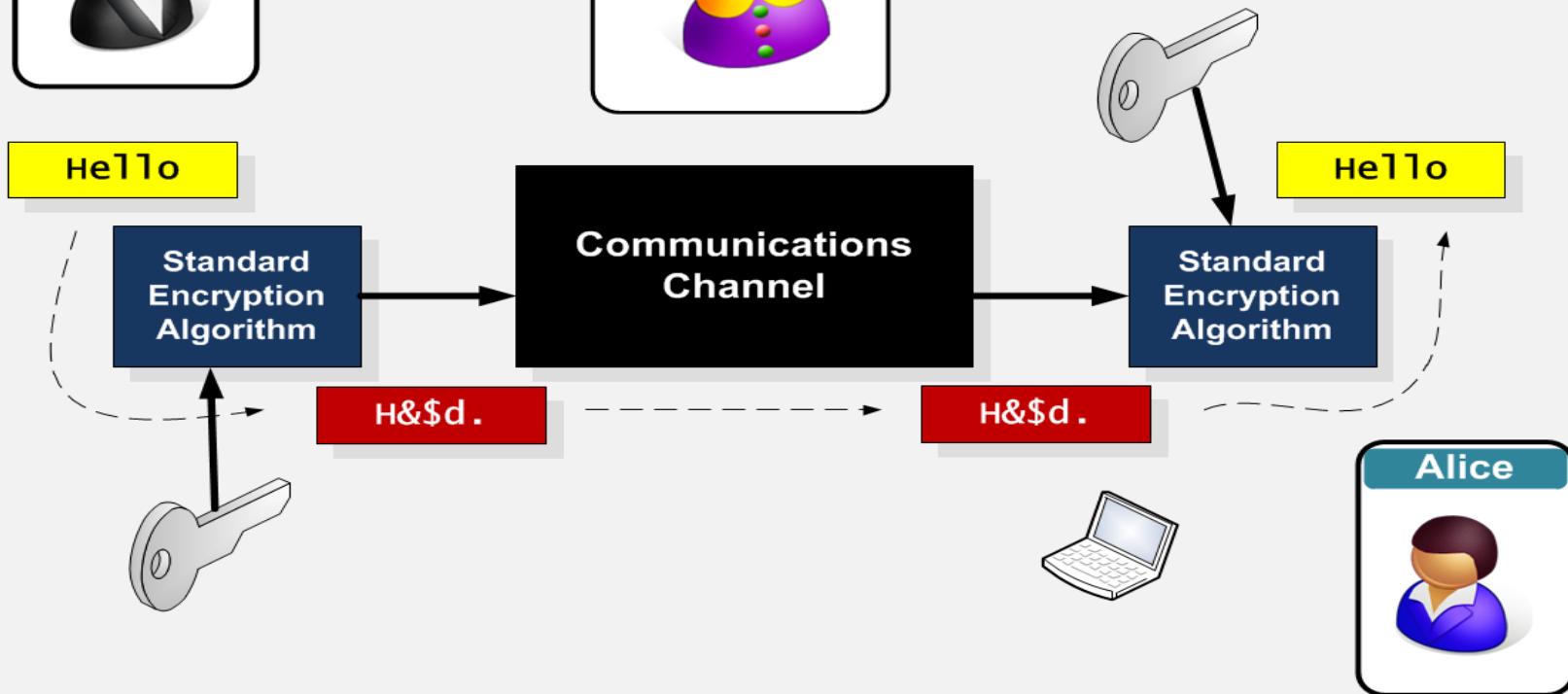
<https://asecuritysite.com/symmetric>



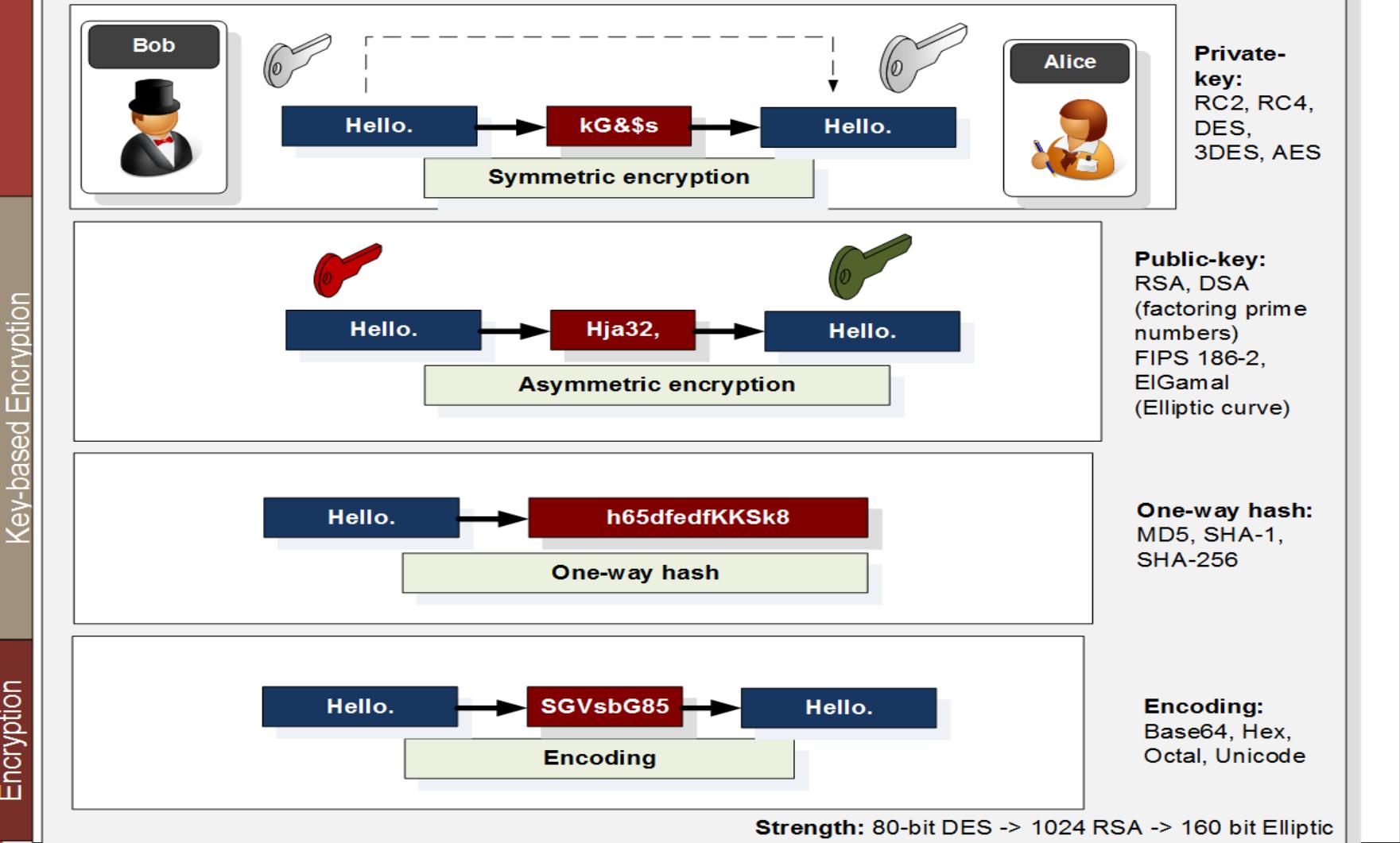


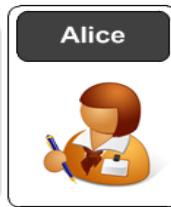


The major problem is that Eve could gain the encoding algorithm.



Encryption





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	cryptographic hash	RSA modulus
teaspoon security	0.0025 liter	35	70	242
shower security	80 liter	50	100	453
pool security	2 500 000 liter	65	130	745
rain security	0.082 km ³	80	160	1130
lake security	89 km ³	90	180	1440
sea security	3 750 000 km ³	105	210	1990
global security	1 400 000 000 km ³	114	228	2380
solar security	-	140	280	3730

9	512	44	1.76×10^{13}	84	1.93×10^{23}
10	1024	48	2.81×10^{14}	88	3.09×10^{26}





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

Time to crack

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

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

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



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

Time to crack

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

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

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



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

Time to crack

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

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

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

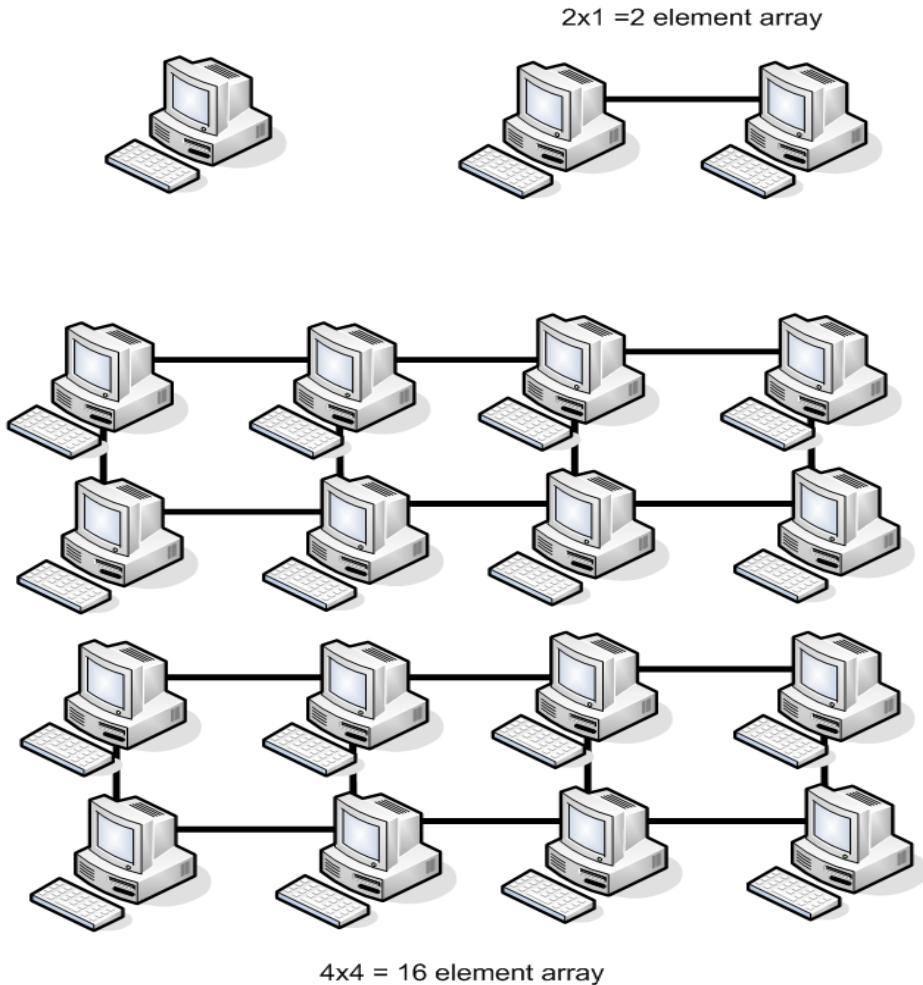


Date	Hours	Days	Years
2017	2,500,000	104,167	285
2018	1,250,000	52,083	143
2019	625,000	26,042	71
2020	312,500	13,021	36
2021	156,250	6,510	18
2022	78,125	3,255	9
2023	39,063	1,628	4
2024	19,532	814	2
+8	9,766	407	1
+9	4,883	203	1
+10	2,442	102	0.3
+11	1,221	51	0.1
+12	611	25	0.1
+13	306	13	0
+14	153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

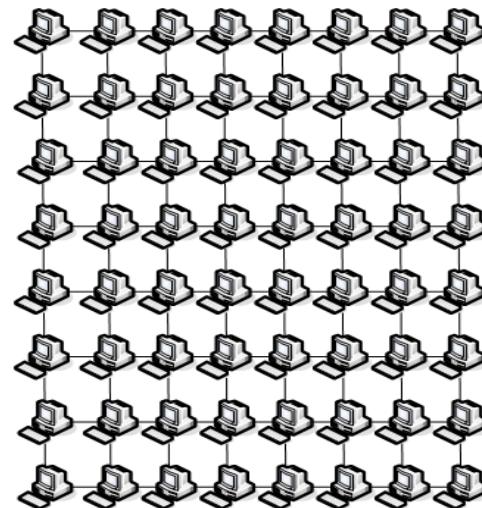
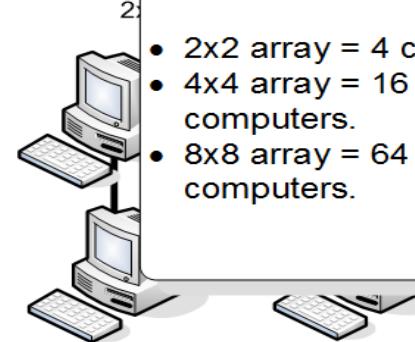
Time to crack

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

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



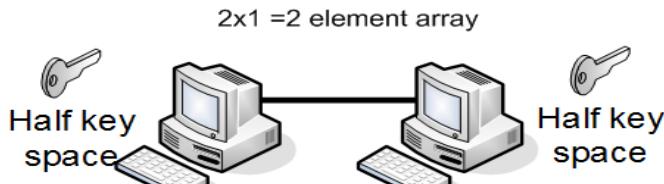
Parallel processing



16x16 = 256 element array

Parallel processing

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



Brute-force

Encryption

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				

key
ice

16x16 = 256 element array

4x4 = 16 element array

Author: Prof Bill Buchanan

Symmetric Key

Basics

Block or Stream?

Secret Key Methods

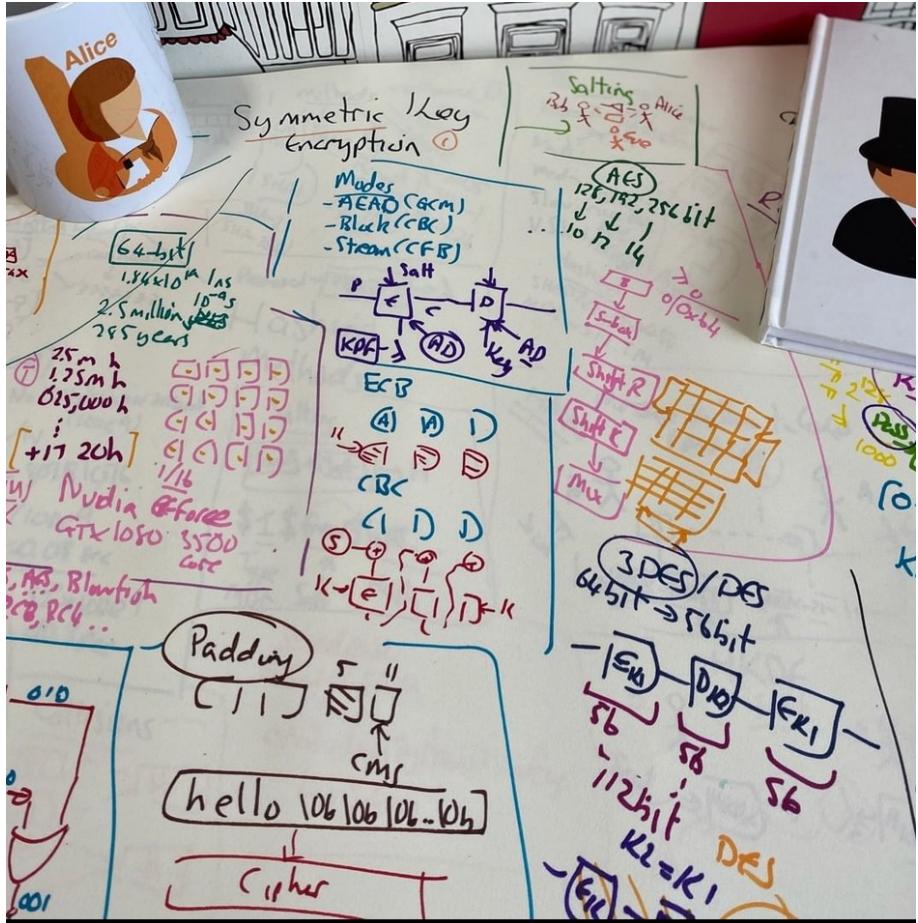
Salting

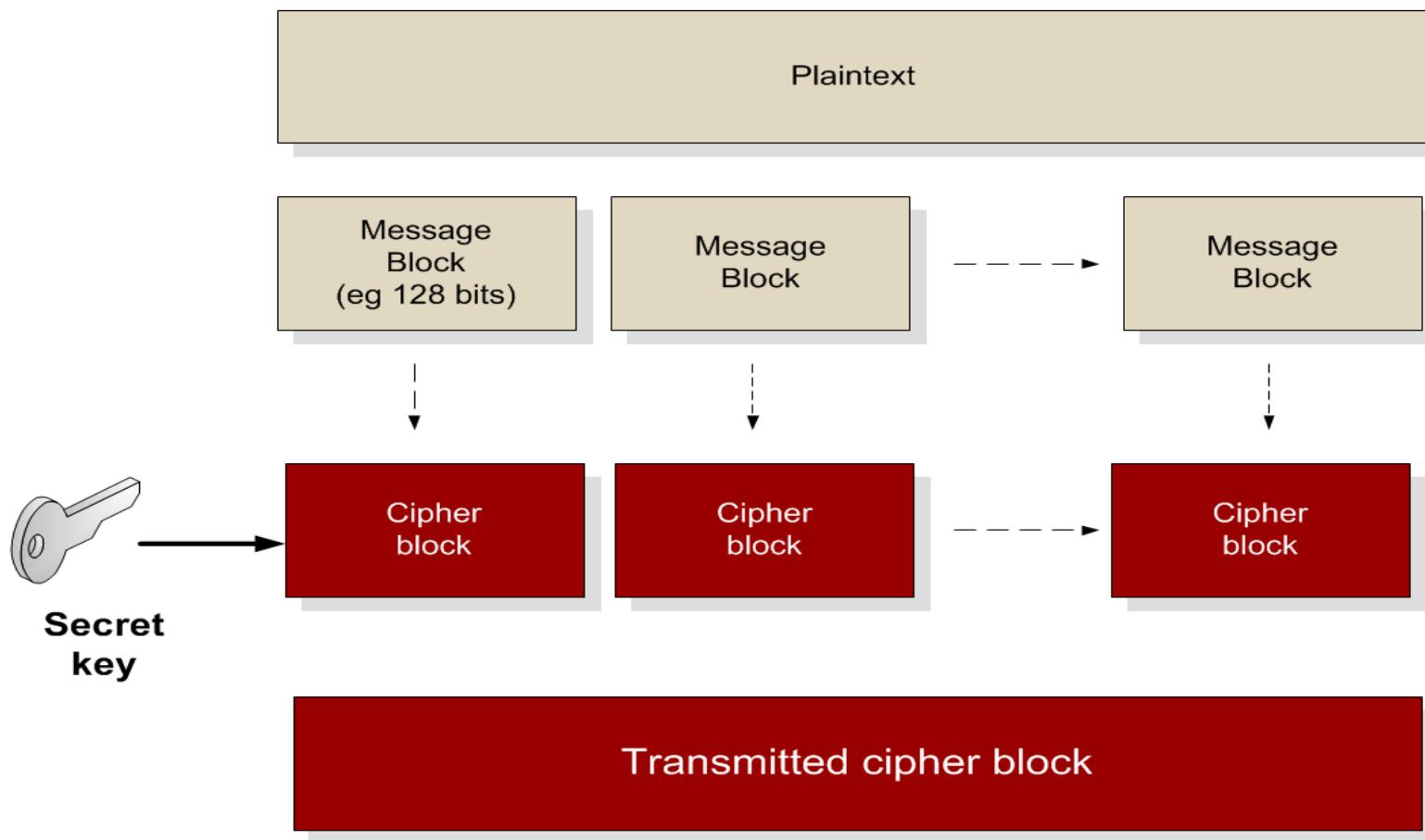
AES

Key Entropy

Prof Bill Buchanan OBE

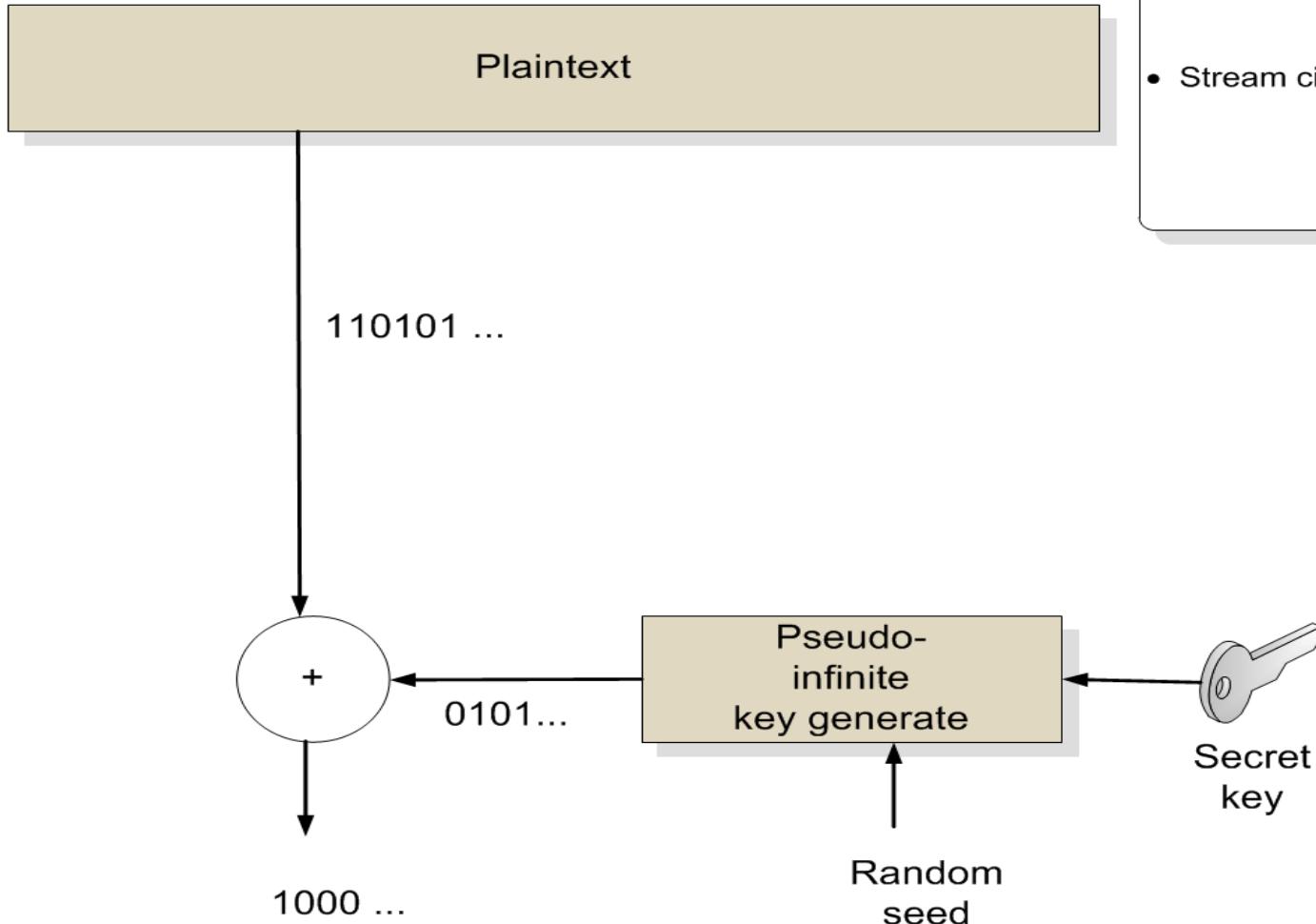
<https://asecuritysite.com/symmetric>



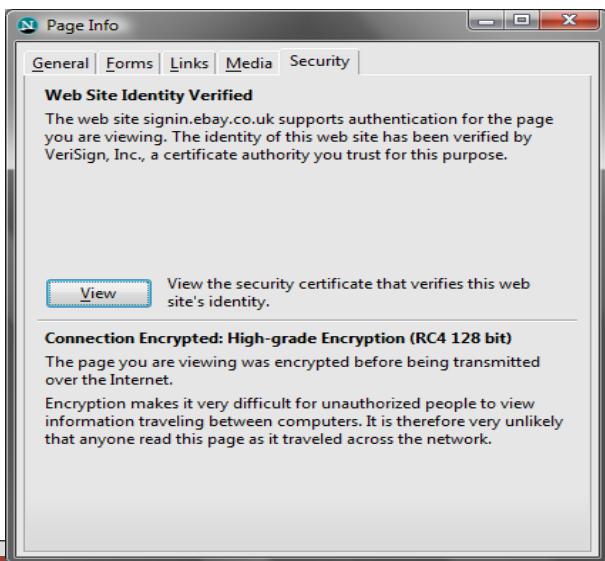
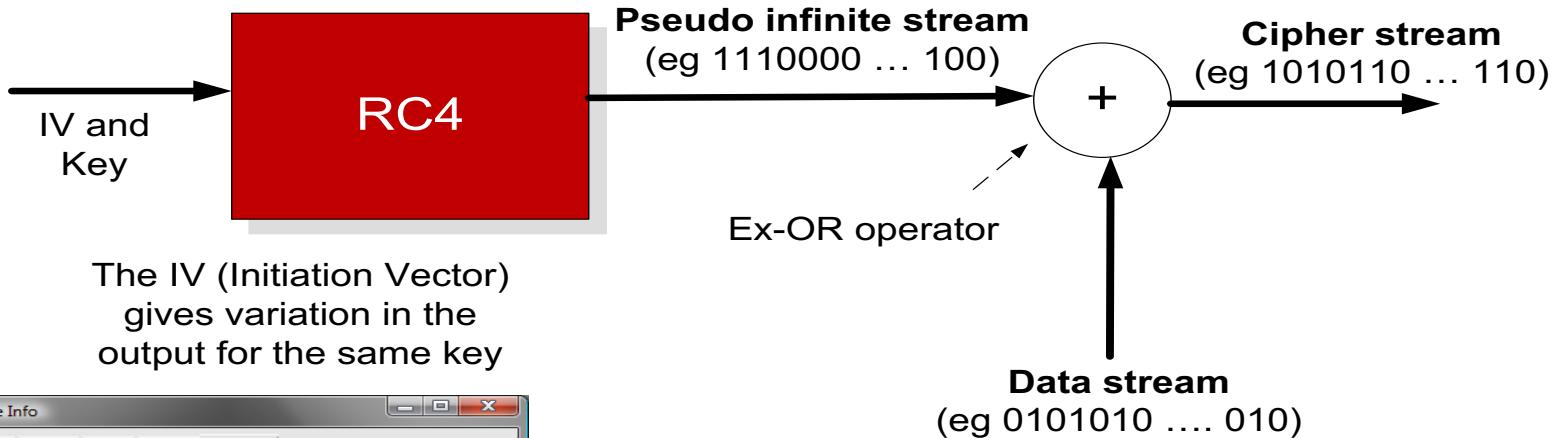


Stream cipher

- Stream cipher (RC4)



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



Data stream 0101010 ... 010
Pseudo infinite stream 1110000 ... 100 +
Cipher stream 1010110 ... 110

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): 68656c6c6f0b0b0b0b0b0b0b0b0b0b0b0b0b
Cipher (ECB): 0a7ec77951291795bac6690c9e7f4c0d
- After padding (Bit): 68656c6f80000000000000000000000000000000
Cipher (ECB): 731abffc2e3b2c2b5caa9ca2339344f9
- After padding (ZeroLen): 68656c6f000000000000000000000000a
Cipher (ECB): d28e2f7e8e44e068732b292bde444245
- After padding (Null): 68656c6f00000000000000000000000000000000
Cipher (ECB): 444797422460453d95856eb2a1520ece
- After padding (Space): 68656c6f00000000000000000000000000000000
Cipher (ECB): 444797422460453d95856eb2a1520ece
- After padding (Random): 68656c6fffc6ecfd884a38798d62a**0a**
Cipher (ECB): c2c88b4364d2c2dc6f2cac9ab73c995d

Symmetric Key

Basics

Block or Stream?

Secret Key Methods

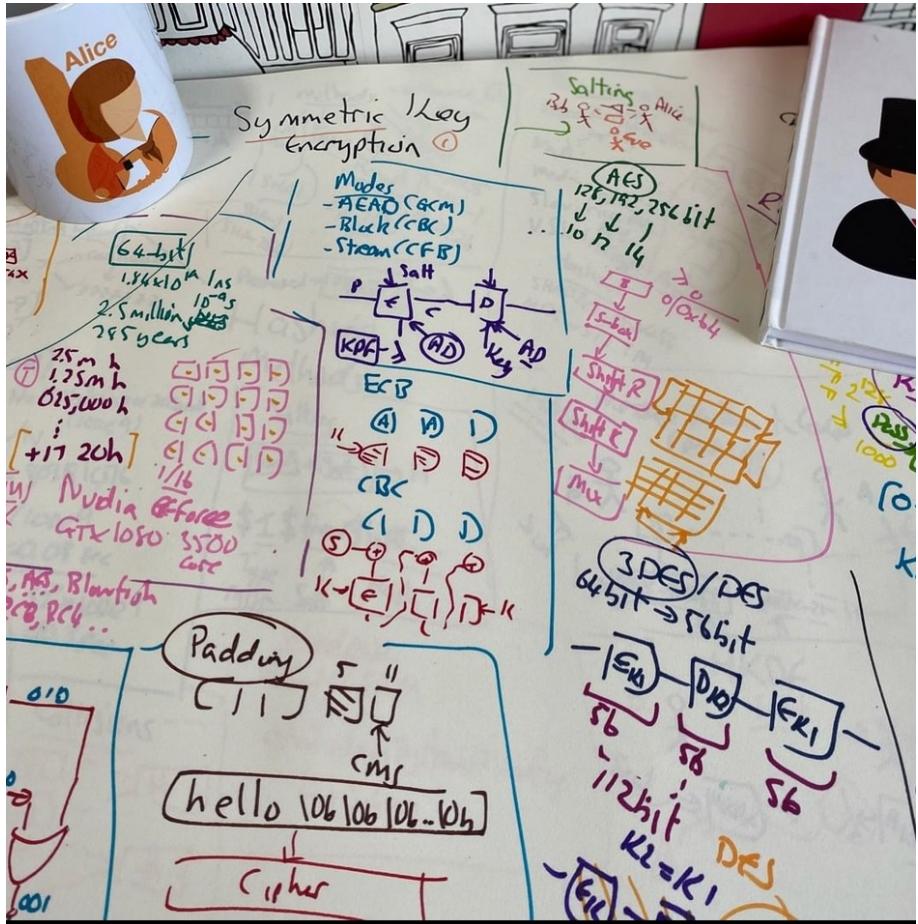
Salting

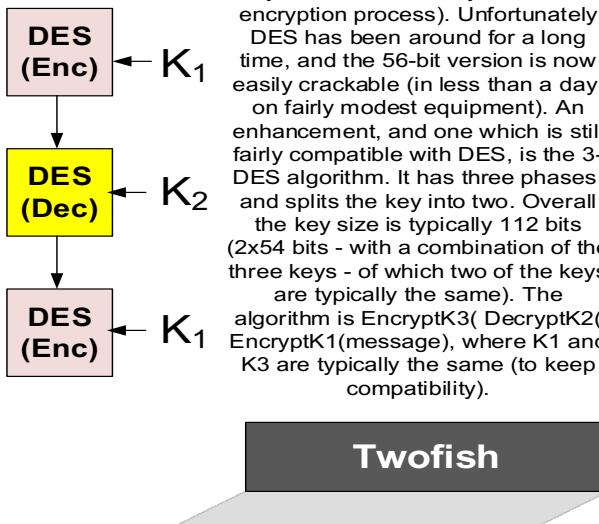
AES

Key Entropy

Prof Bill Buchanan OBE

<https://asecuritysite.com/symmetric>





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

DES

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

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

AES

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

Blowfish

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

RC2

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

Others

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

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

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

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

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



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

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



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

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



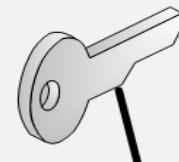


The major problem is that Eve could gain the encoding algorithm.

Hello

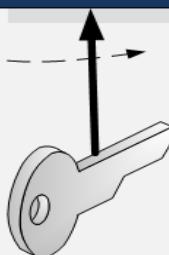
Standard
Encryption
Algorithm

Communications
Channel



Hello

Standard
Encryption
Algorithm



H&\$d.

H&\$d.



Alice

Symmetric Key

Basics

Block or Stream?

Secret Key Methods

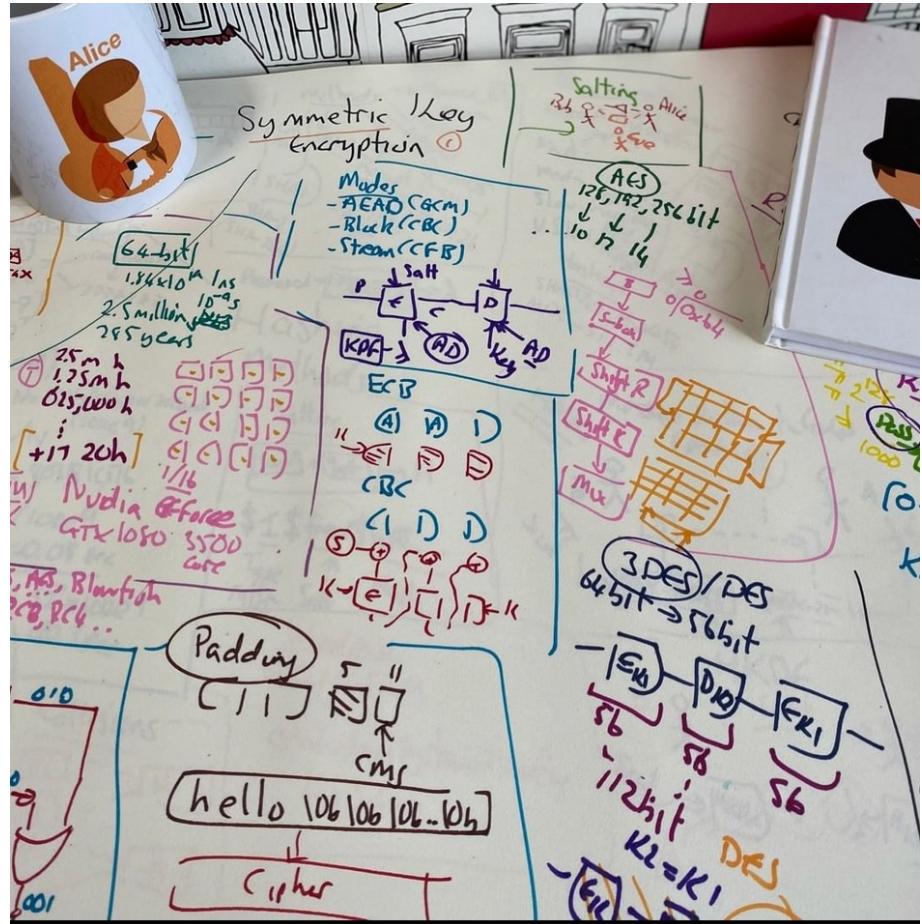
Salting

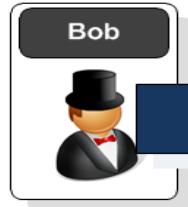
AES

Key Entropy

Prof Bill Buchanan

<https://asecuritysite.com/symmetric>





Hello. How are you?



kG&\$s &FDsaf *fd\$

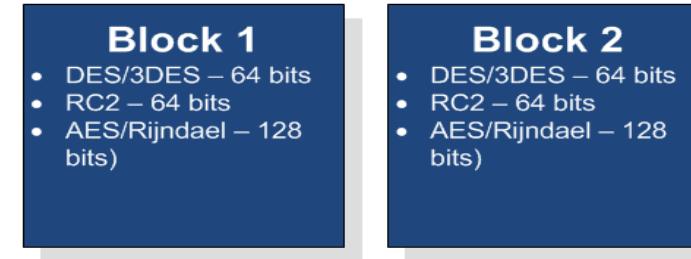


kG&\$s &FDsaf *fd\$



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

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

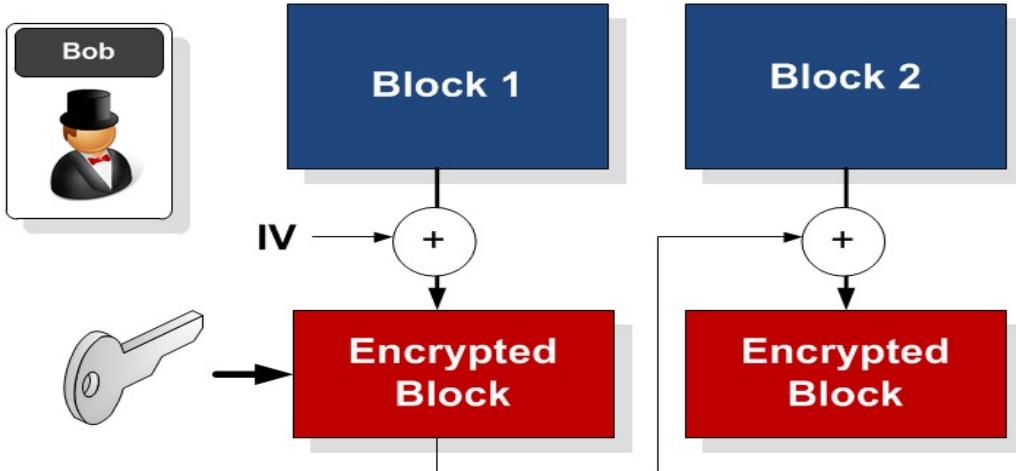


Electronic Code Book (ECB) method. This is weak, as the same cipher text appears for the same blocks.

Hello → 5ghd%43f=

Hello → 5ghd%43f=

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



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



Original image



Image with AES using ECB



Image with AES using CBC

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

eeeeeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeee

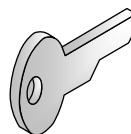
[eeeeeeee] [eeeeeeee] [eeeeeeee][eeeeeeee] [eeeeeeee] [eeeeeeee][eeeeee <PADDING>]

eeeeeeee

eeeeeeee

Block 1
DES (64-bit)

Block 2
DES (64-bit)



Encrypted Block



Encrypted Block

“bill12345”

ED291A7588D871B1

ED291A7588D871B1

ED291A7588D871B1ED291A7588D871B1ED291A7588D
871B1ED291A7588D871B1ED291A7588D871B1ED291A
7588D871B18D6DF6795DDEDACD

Symmetric Key

Basics

Block or Stream?

Secret Key Methods

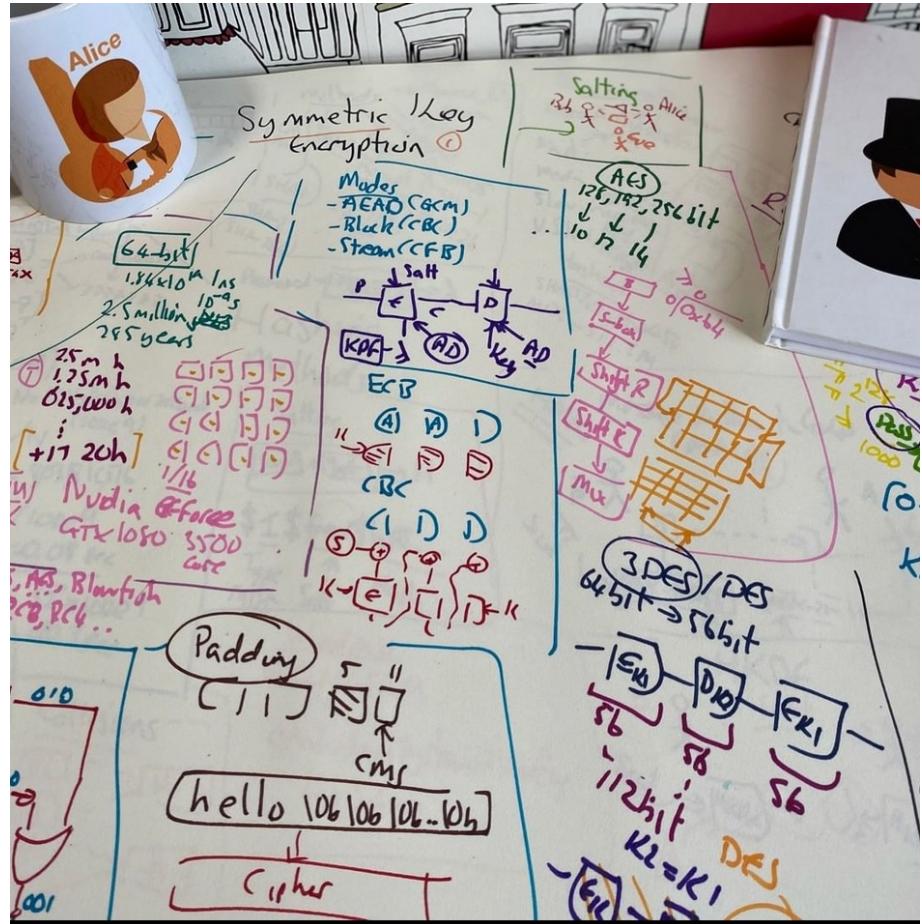
Salting

AES

Key Entropy

Prof Bill Buchanan

<https://asecuritysite.com/symmetric>



Key Entropy

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

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

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

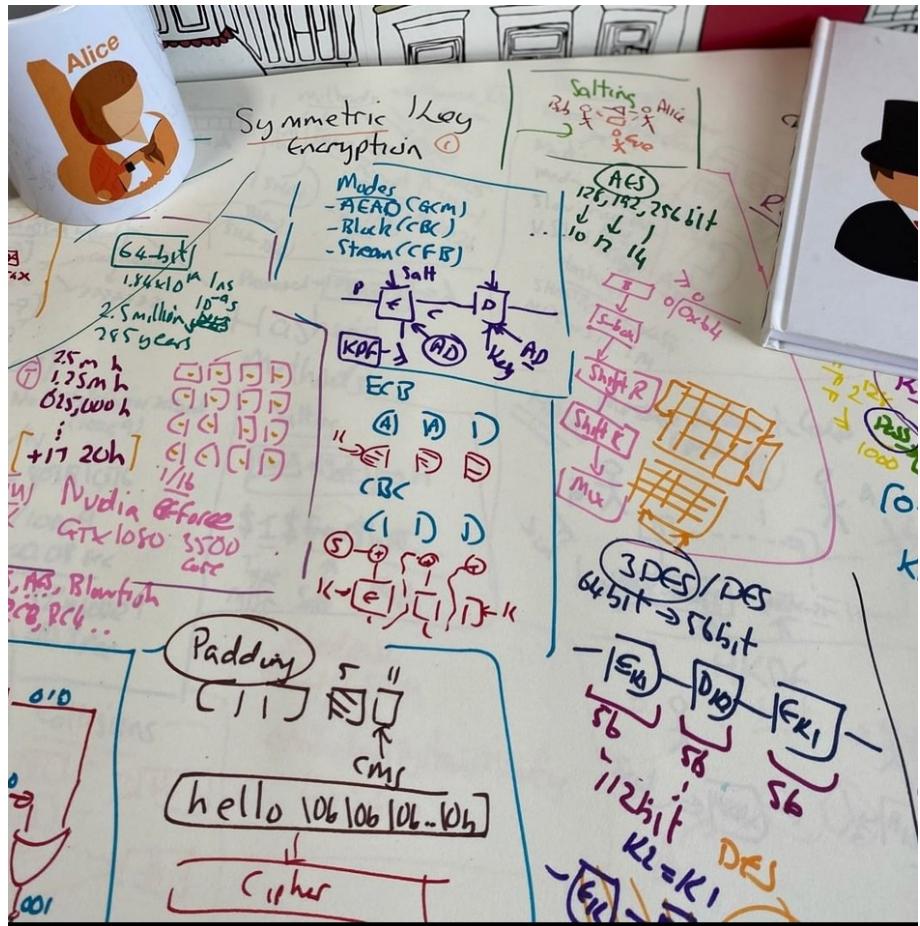
Salting

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.

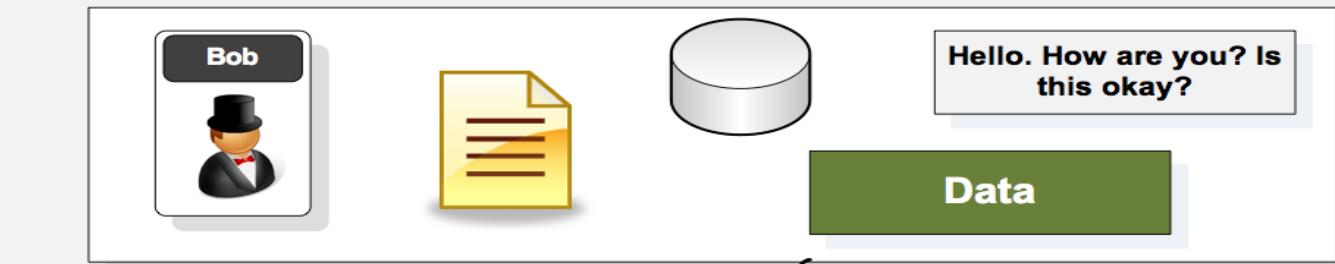
Secret Shares.

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.**



Author: Prof Bill Buchanan



**Hashing
Algorithm (MD5)
- 128 bit 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



**Hashing
Algorithm (SHA-1)**
- 160 bit signature



hello

qvTGHdzF6KLavt4P00gs2a6pQ00=

Hello

9/+ei3uy4Jtwk1pdeF4MxdnQq/A=

Hello. How are you?

Puh2Am76bhjqE51btTwtsqbdFC8=

Napier

v4GxNavod2b09GR2Tqw4yopOuro=

Base-64

hello

AAF4C61DDCC5E8A2DABEDE0F3B482CD9AEA9434D

Hello

F7FF9E8B7BB2E09B70935A5D785E0CC5D9D0ABF0

Hello. How are you?

3EE876026EFA6E18EA13995B4D6B70B2A6DD142F

Napier

BF81B135A5687766F4F464764EAC38CA8A4EBABA

Hex



**Hashing
Algorithm (MD5)
- 128 bit signature**



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

8A8BDC3FF80A01917D0432800201CFBF

iovcP/gKAZF9BDKAAgHPVW==

Hex

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
```



Authentication

Message Hash

[Path] / filename

[C:\windows\System32\
12520437.cpx
12520850.cpx
8point1.wav
aaclient.dll
AC3ACM.acm
Ac3audio.ax
ac3filter.cpl
accessibilitycpl.dll
ACCTRES.dll
acledit.dll
.
.
ZSHP1020.CHM
ZSHP1020.EXE
ZSHP1020.HLP
ZSPOOL.DLL
ZTAG.DLL
ZTAG32.DLL

MD 5 sum

0a0feb9eb28bde8cd835716343 b03b14
d 69ae057cd82d04ee7d311809 abefb2a
beab 165fa58ec5253185 f32e124685d5
ad 45dedfdcf69a28cbaf6a2ca84b5f1e
59683d1e4cd0b1ad6ae32e1d627ae25f
4b87d889edf278e5fa223734 a9bbe79a
10b27174d46094984 e7a05f3c36acd2a
ac 4cecc86eeb8e1cc2e9fe022cff3ac1
58f57f2f2133a2a77607 c8ccc9a30f73
0bcee3f36752213d1b09d18e69383898

c 671ed [Path] / filename

96e45a
[C:\windows\system32\
12520437.cpx
12520850.cpx
8point1.wav
aaclient.dll
AC3ACM.acm
Ac3audio.ax

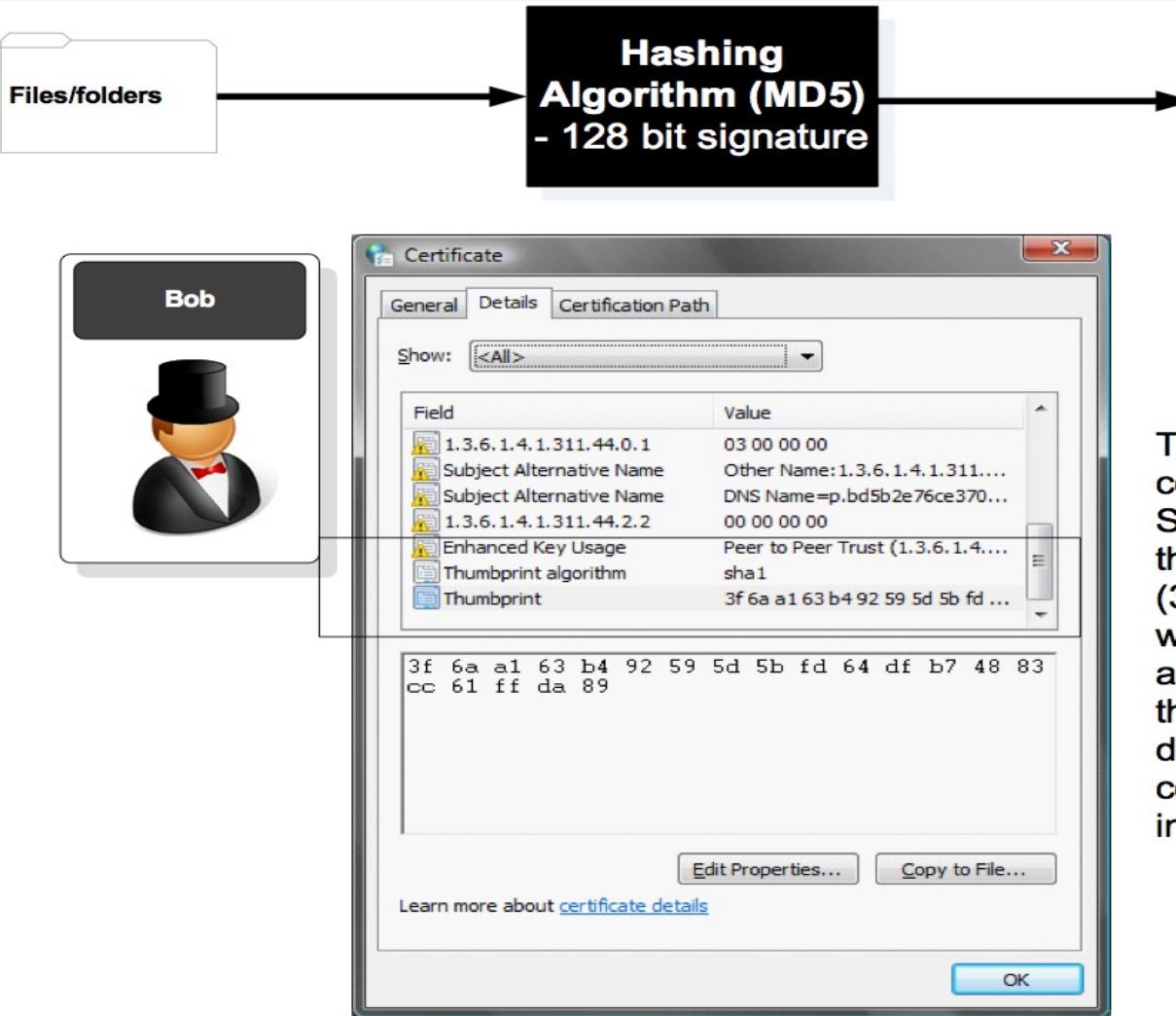
MD 5 sum

Cg /rnrKL3ozYNXFjQ7A7FA==
1prgV82C0E7n0xGAmr77Kg==
vqswX 6w0xSUxhfMuEkaF 1Q==
rUXe 39z2mijLr2osqEtFhg==
wg 9HkzQsa1q4y4dYnriXw==
S 4fYie3ye0X6Ijc0qbvnmg==

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.



Hash signature

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

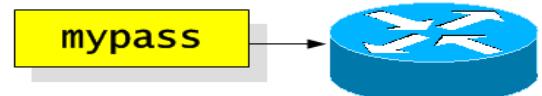
The digital certificate has an SHA-1 hash thumbprint (`3f6a...89`) which will be checked, and if the thumbprint is different, the certificate will be invalid.

Windows login/ authentication



NT-password
hash for Windows
NT, XP and Vista

Cisco password storage (MD5)



MD5 encoded
password

```
# 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
```

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.

One-way hash

Types

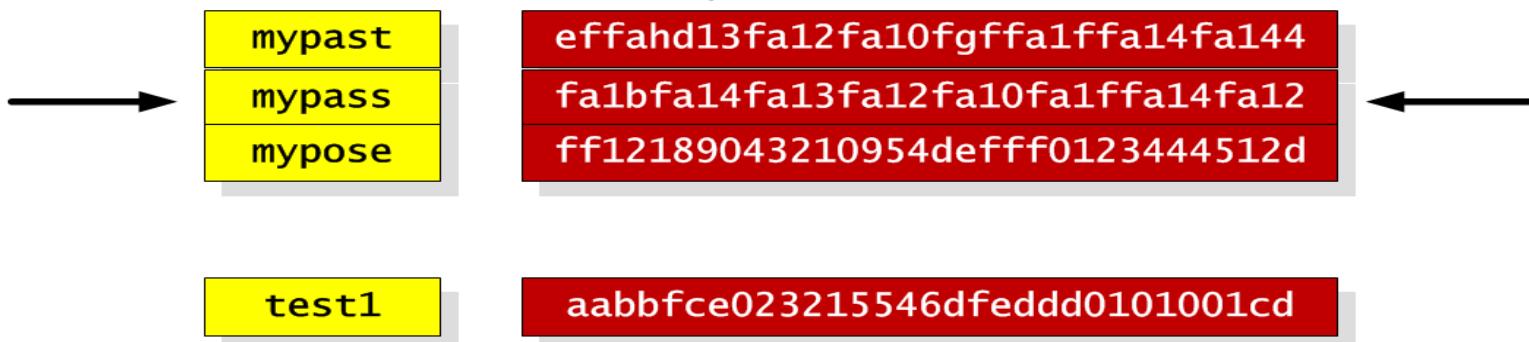
One-way hash

Windows login/ authentication



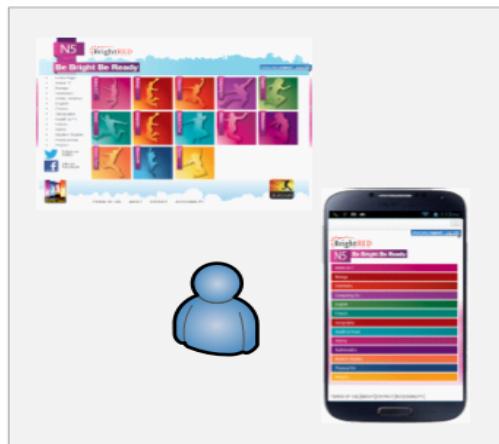
Hashing suffers from **dictionary attacks** where the signatures of well known words are stored in a table, and the intruders does a lookup on this

NT-password
hash for Windows
NT, XP and Vista





Risk 4: One Password Fits All



TJ-maxx
Marshalls.

47 million accounts



1 million accounts – in plain text. 77 million compromised

LinkedIn

6.5 million accounts
(June 2013)



150 million accounts compromised

#	Count	Ciphertext	Plaintext
1.	1911938	EQ7fIpT7i/Q=	123456
2.	446162	j9p+HwtWWT86aMjgZFLzYg==	123456789
3.	345834	L8qbAD3j13jioxG6CatHBw==	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



One account hack ... leads to others



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 versions 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. RIPEMD160. 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

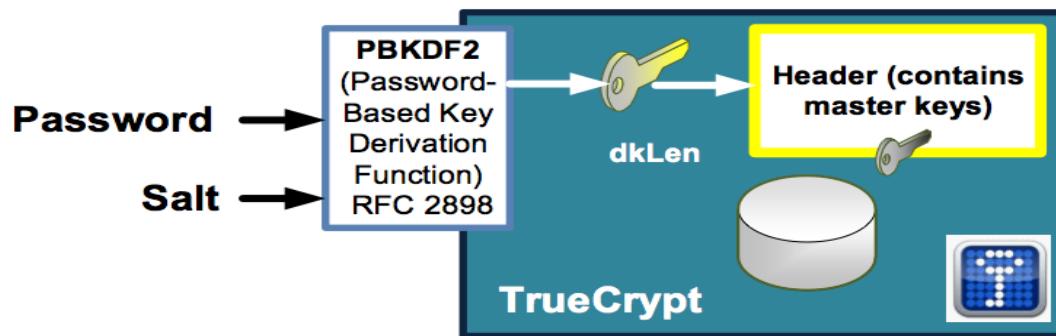
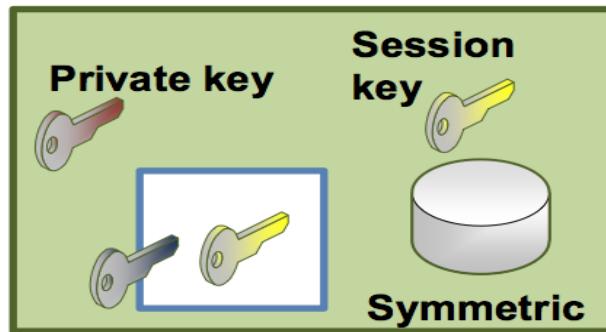
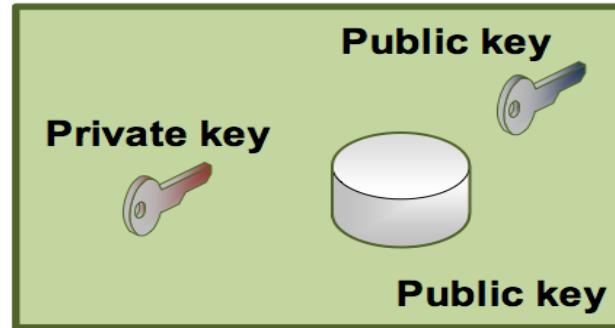
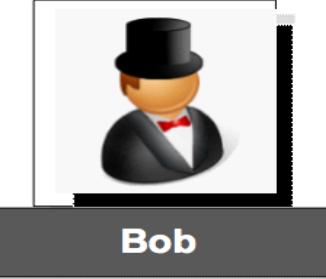
Tiger. Tiger is a 192-bit hash function, and was designed by Ross Anderson and Eli Biham in 1995. It is often used by clients within Gnutella file sharing networks, and does not suffer from known attacks on MD5 and SHA-0/SHA-1. Tiger2 is an addition, in which the message is padded with a byte of 0x80 (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.

Murmur

While hashing methods such as MD5 and SHA-1 use crypto 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.



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



NT-password
hash for Windows
NT, XP and Vista

Salt increase the range of the signatures



password

\$1\$fred\$bATAk8UUH/IDAp9sd6IUv/

1

fred



bATAk8UUH/IDAp9sd6IUv/

password

bATAk8UUH/IDAp9sd6IUv/

fred

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



```
# cat /etc/shadow
root:$1$Etg2ExUZ$F9NTP7omafhK1lqaBMqng1:15651:0:99999:7:::
# openssl passwd -1 -salt Etg2ExUZ redhat
$1$Etg2ExUZ$F9NTP7omafhK1lqaBMqng1
```

```
$ 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-----
MIIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDXv9HSFkpM+ZoOQcpdHBZiuwX8
EzIKm0nsgjc5ZTYVaF9CMLtmKoTzep7aQX9o9nKepFt1kq73Ta9v0Pd6Cx61/cgY
xy2tShw0imrtFaVDFjX+7kLmc0uwbFFCoZMtJxIaXaa9SV2kARxOCTJ2u0jRTCCe
XU09IJGHnIhSNJeIJQIDAQAB
-----END PUBLIC KEY-----
```

```
$ cat /etc/shadow
root:$1$Etg2ExUZ$F9NTP7omafhK1lqaBMqng1:15651:0:99999:7:::
```

```
$ openssl passwd -1 -salt Etg2ExUZ redhat
$1$Etg2ExUZ$F9NTP7omafhK1lqaBMqng1
```

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. This 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:

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



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

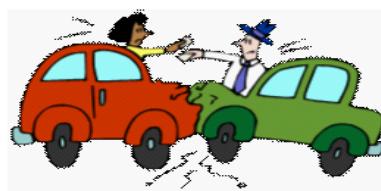
```
d131dd02c5e6eec4693d9a0698aff95c  
2fcab58712467eab4004583eb8fb7f89  
55ad340609f4b30283e488832571415a  
085125e8f7cdc99fd91dbdf280373c5b  
d8823e3156348f5bae6dacd436c919c6  
dd53e2b487da03fd02396306d248cda0  
e99f33420f577ee8ce54b67080a80d1e  
c69821bcb6a8839396f9652b6ff72a70
```

```
d131dd02c5e6eec4693d9a0698aff95c  
2fcab50712467eab4004583eb8fb7f89  
55ad340609f4b30283e4888325f1415a  
085125e8f7cdc99fd91dbd7280373c5b  
d8823e3156348f5bae6dacd436c919c6  
dd53e23487da03fd02396306d248cda0  
e99f33420f577ee8ce54b67080280d1e  
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.

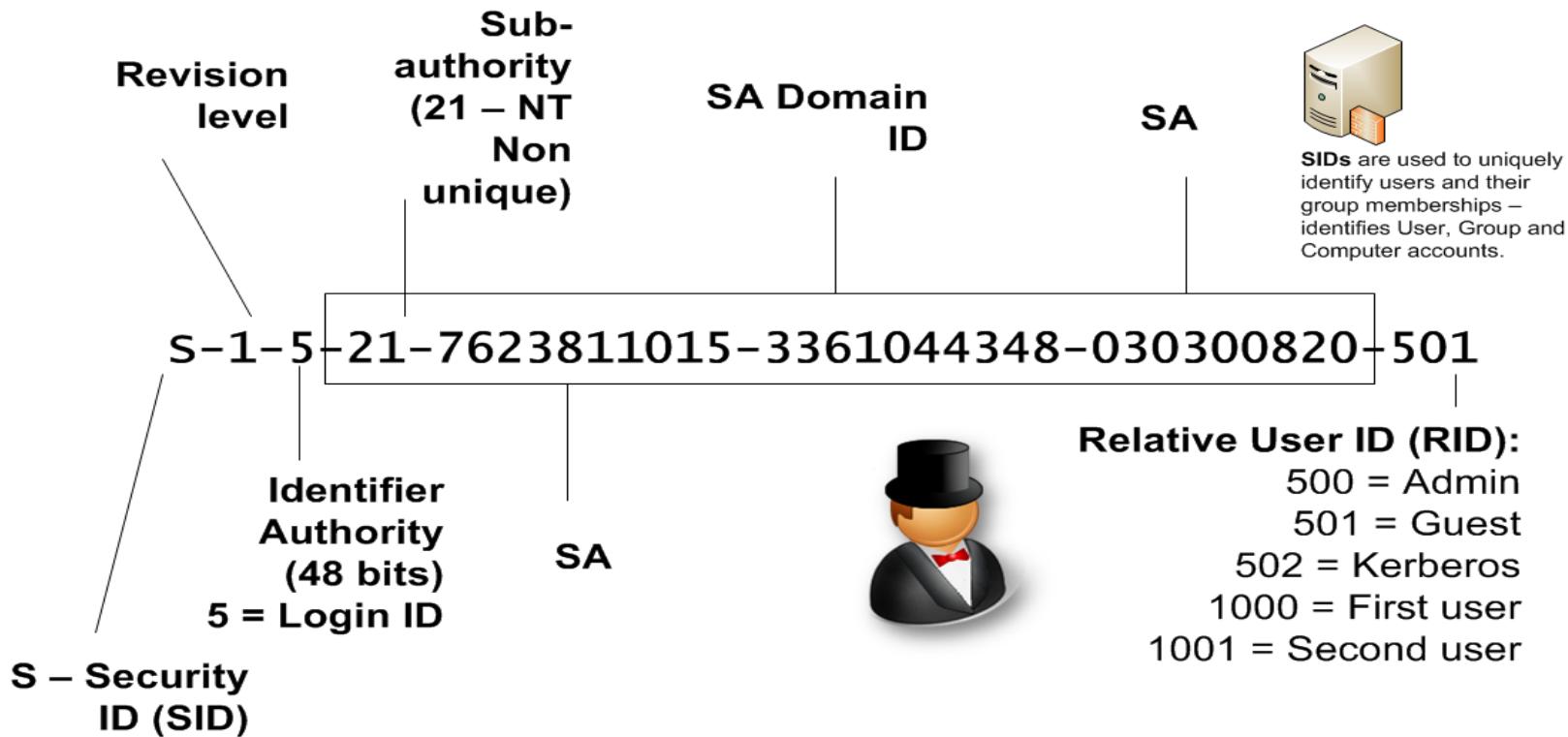
Prof Bill Buchanan OBE

<http://asecuritysite.com/hash>



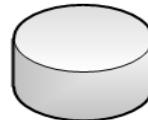
```
C:> user2sid \pluto guest  
S-1-5-21-7623811015-3361044348-030300820-501  
C:> sid2user 5 21 7623811015 3361044348 030300820 500  
Name is Fred  
Domain is PLUTO
```

Windows
CEH

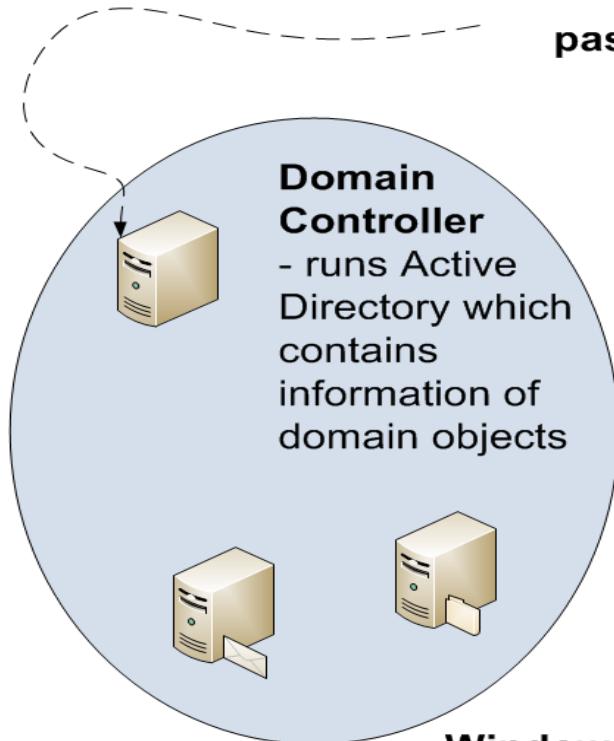
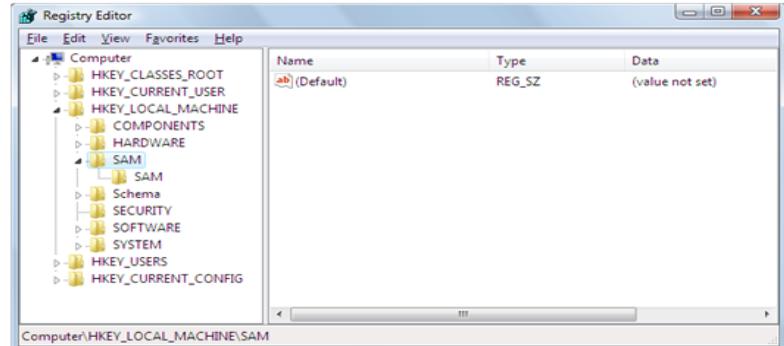




HKLM\SAM



SAM Database
(stores
usernames
and
passwords)



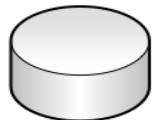
Subsystem (Lsass) – Windows Security mechanism – Attached by Sasser Worm which exploited a buffer overflow



Local Authority

- 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
```



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:500:12B9C54F6FE0EC80AAD3B435B51404FF·307E40814F7d4
E103F6A69B04EA78F3D:::
```



The ophcrack interface shows the following data in its main table:

User	LM Hash	NT Hash	LM Pwd 1	LM Pwd 2	NT Pwd
password	E52CAC67...	8846F7EA...	PASSWOR	D	password
myuser	12B9C54F6...	307E40814...	NAPIER	empty	napier

Below the table, the progress bar indicates 34% completion for an XP forensic task.

Preload: done Brute force: done Pwd found: 2/2 Time elapsed: 0h 0m 17s

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

Ultra 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
Ultra-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:	2fd4e1c67a2d28fcfd849ee1bb76e7391b93eb12
SHA-256:	d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592
SHA-512:	07e547d9586f6a73f73fbac0435ed76951218fb7d0c8d788a309d7 85436bbb642e93a252a954f23912547d1e8a3b5ed6e1bfd7097821233fa0538f3db854fee6
MD-5:	9e107d9d372bb6826bd81d3542a419d6
DES:	ZDeS94Lcq/6zg
Bcrypt:	\$2a\$05\$2czCv5GYgkx3aobmEyewB.ejV2hePMdbvTdCyNaSzWtIGPPjB2xx6
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}2fd4e1c67a2d28fcfd849ee1bb76e7391b93eb12
MS SQL 2000:	0x0100BF77CE595DCD1FC87A37B3DEBC27A8C97355CB96B8BAB 63E602662BA5D5D33B913E422499BE72FF3D9BB65DE
MySQL:	*A4E4D26FD0C6455E23E2187C3AABE844332AA1B3
Oracle 10:	4CDA2299FCAD0499
Postgres (MD5):	md5d44c15daa11770f25c5350f7e5408dd1
Cisco PIX:	kGyKN5CqdFQ1qJUs
Cisco Type 7:	15260309443B3E2D2B3875200108010D41505640135E1B0E080 519574156401540035E460B594D1D53020B5C

Benchmark

Hashing

Hashing Types.

Hashing Methods.

Salting.

Collisions.

LM and NTLM Hashes (Windows).

Hash Benchmarks.

Message Authentication Codes (MACs).

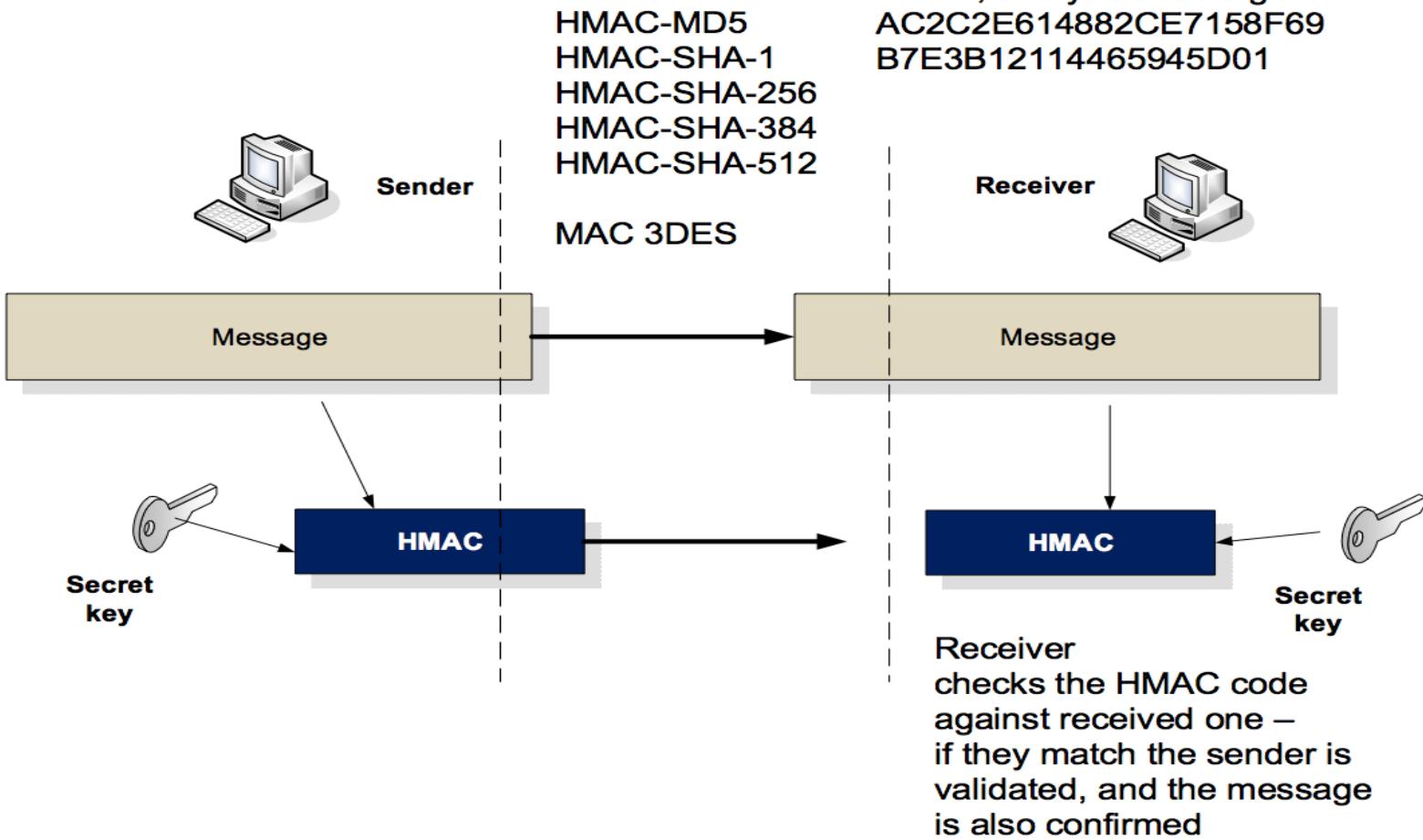
OTP/HOTP.

Prof Bill Buchanan OBE

<http://asecuritysite.com/crypto02>

<http://asecuritysite.com/encryption>





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>





One-time password

$f(m)$

$f(f(m))$

$f(f(f(m)))$



System logon

One-time password (timed)

$H(t_1)$

$H(t_2)$

$H(t_3)$



System logon

One-time password (counter)

$H(c_1)$

$H(c_2)$



System logon

Hashing

Hashing Types.

Hashing Methods.

Salting.

Collisions.

LM and NTLM Hashes (Windows).

Hash Benchmarks.

Message Authentication Codes (MACs).

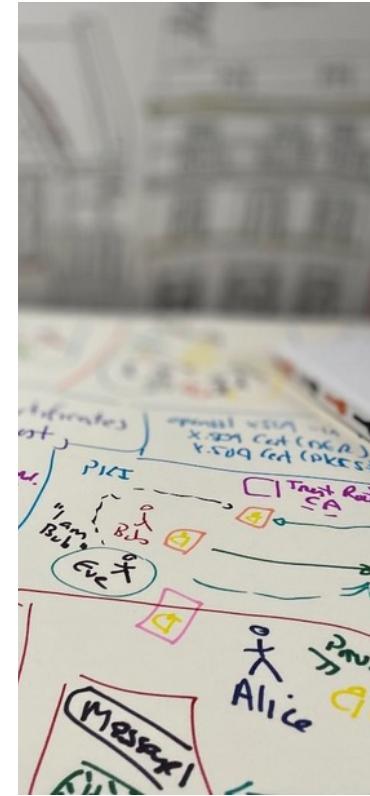
OTP/HOTP.

Prof Bill Buchanan OBE

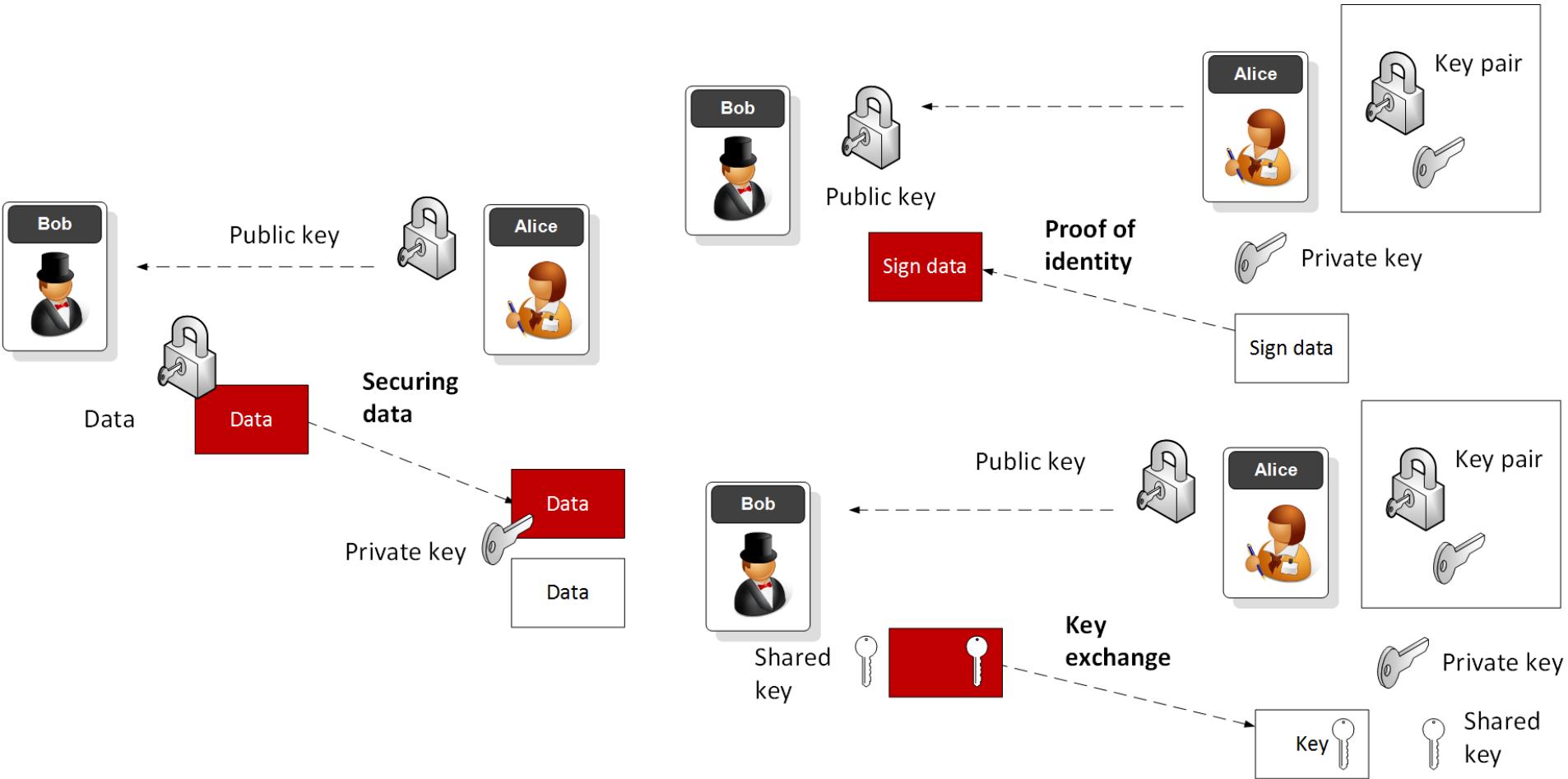
<http://asecuritysite.com/hash>



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]	

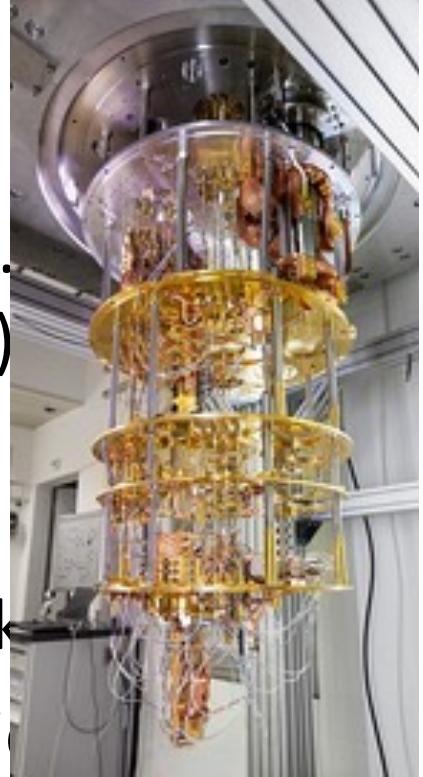


Public Key Methods



Public Key Methods

- **Integer Factorization.** Using prime numbers.
Example: RSA. Key size: 2,048 bits (modulus)
Digital Certificates.
- **Discrete Logarithms.** $Y = g^x \text{ mod } P$. Example:
Prime number size: 2,048 bits. Key handshake.
- **Elliptic Curve Relationships.** Example: Elliptic curves.
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

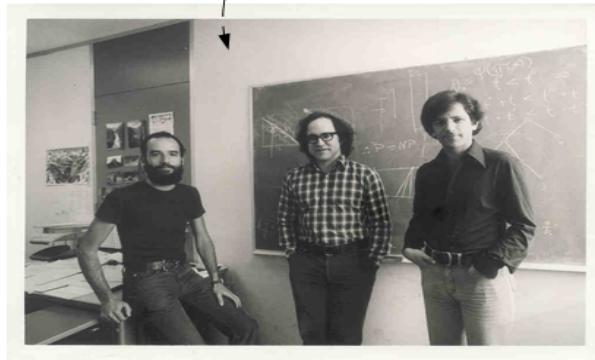
(p-1)*(q-1)

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 us to distribute an encryption key, while we have the decryption key?



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

RSA



- Two primes p, q.
- Calculate N (modulus) as $p \times q$
eg 3 and 11. n=33.
- Calculate PHI as $(p-1) \times (q-1)$.
PHI=20
- Select e for no common factor with PHI. e=3.
- **Encryption key [e,n] or [3,33].**
- $(d \times e) \bmod 20 = 1$
- $(d \times 3) \bmod 20 = 1$
- 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 = $M^e \text{ mod } N$
eg $M=5$.
 - Cipher = $5^3 \text{ mod } 33 = 26$
 - Decipher = $C^d \text{ mod } N$
 - Decipher = $(26)^7 \text{ mod } 33 = 5$

Public Key

Basics
RSA

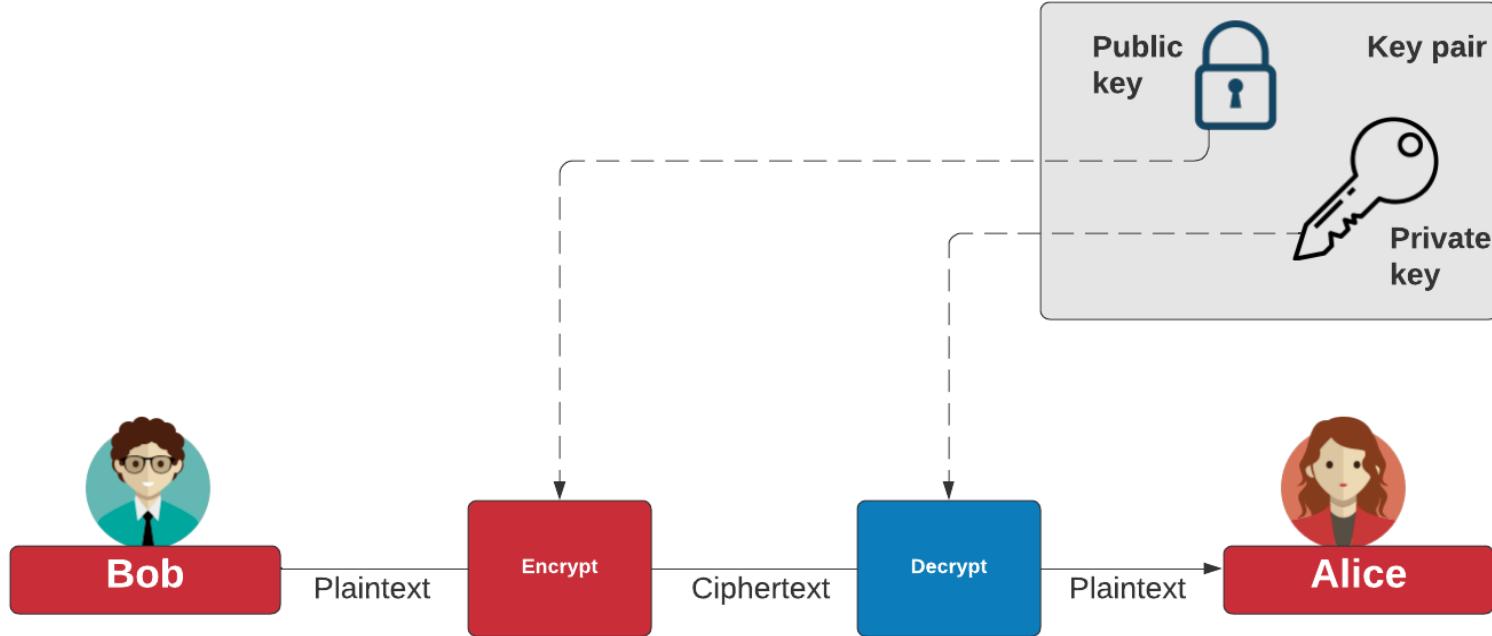
Applications (Encryption and Signing)

Prof Bill Buchanan OBE

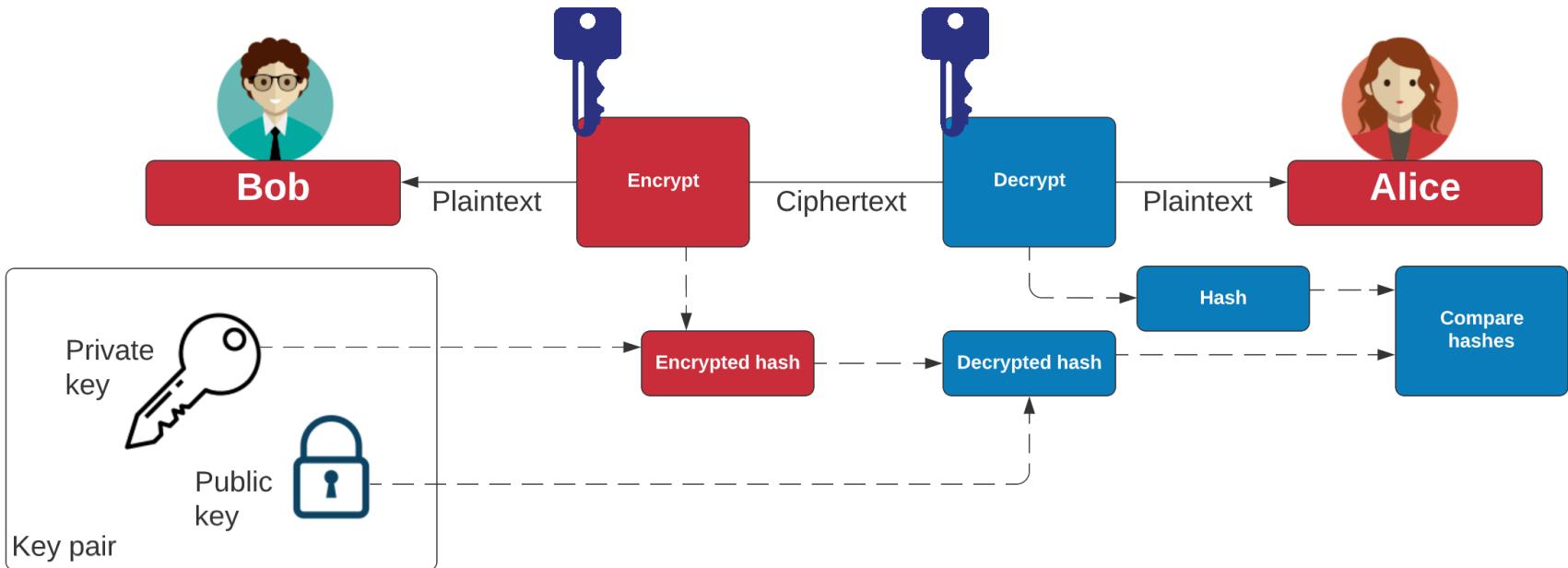
<https://asecuritysite.com/rsa/>



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

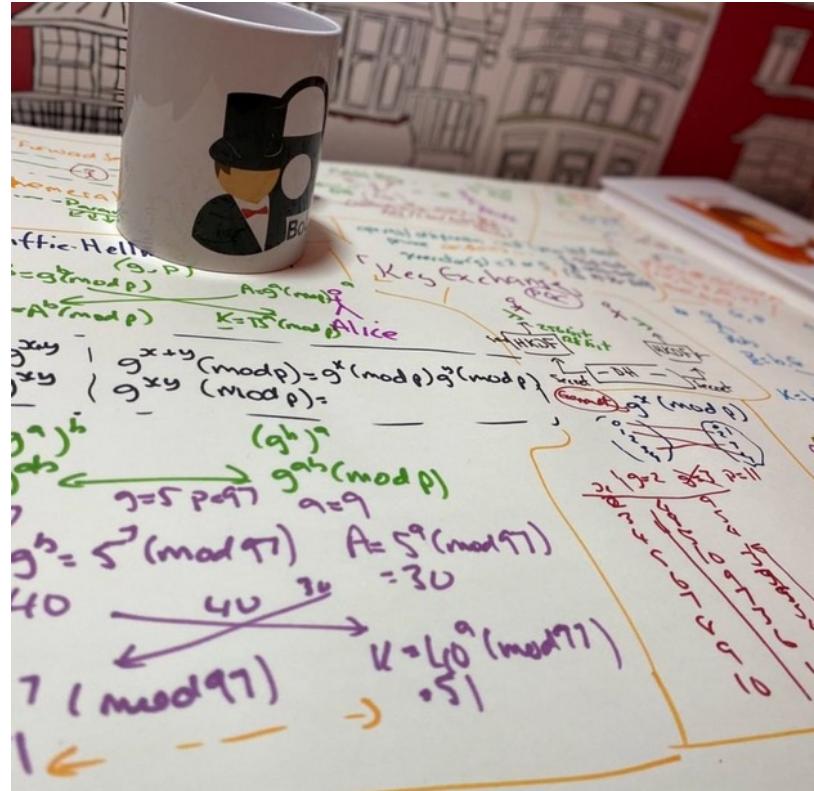
Diffie-Hellman Weaknesses

Passing Key Using Public Key

Prof Bill Buchanan OBE

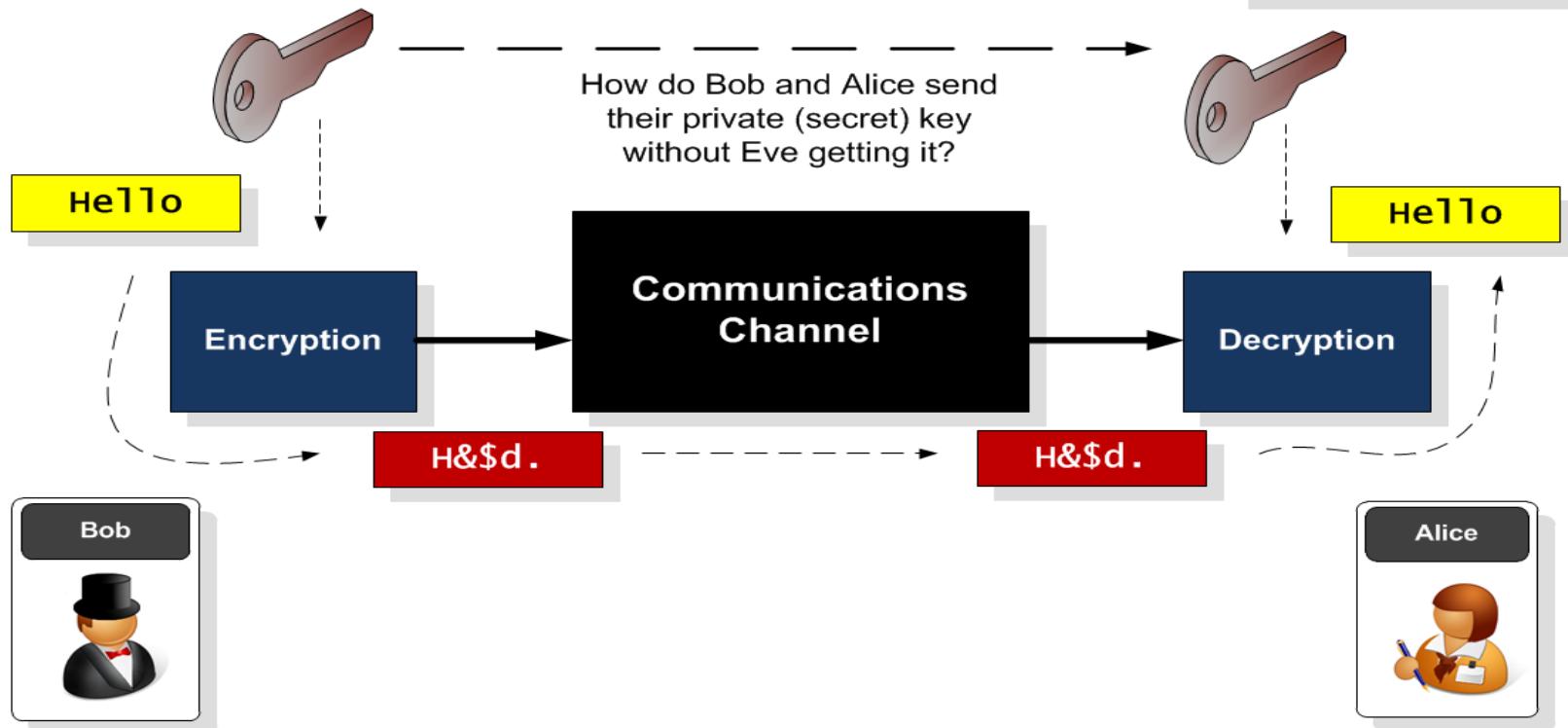
<http://asecuritysite.com/crypto05>

<http://asecuritysite.com/encryption>



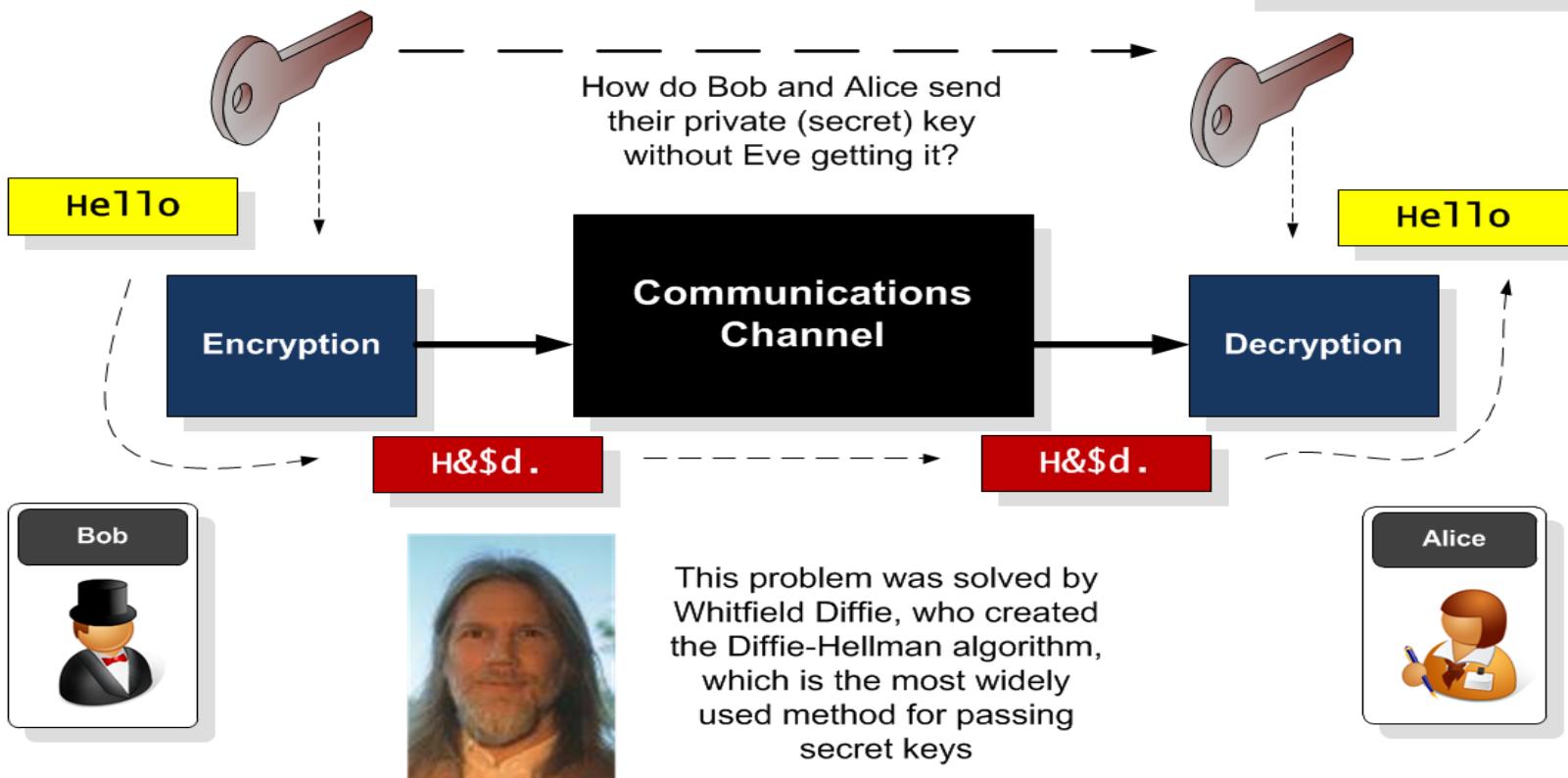
Private key

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



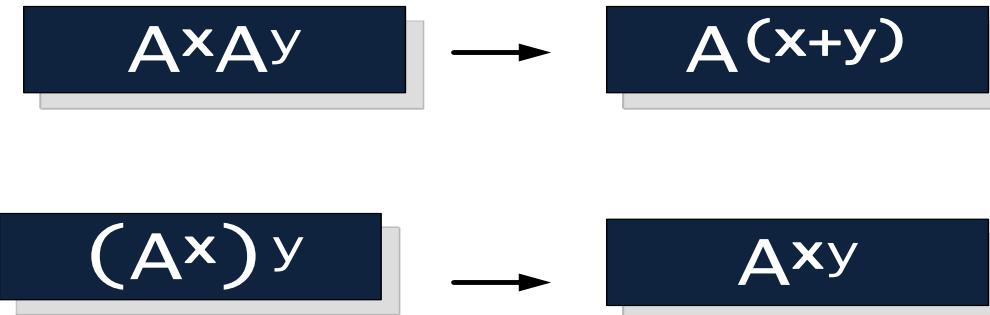
Diffie-Hellman

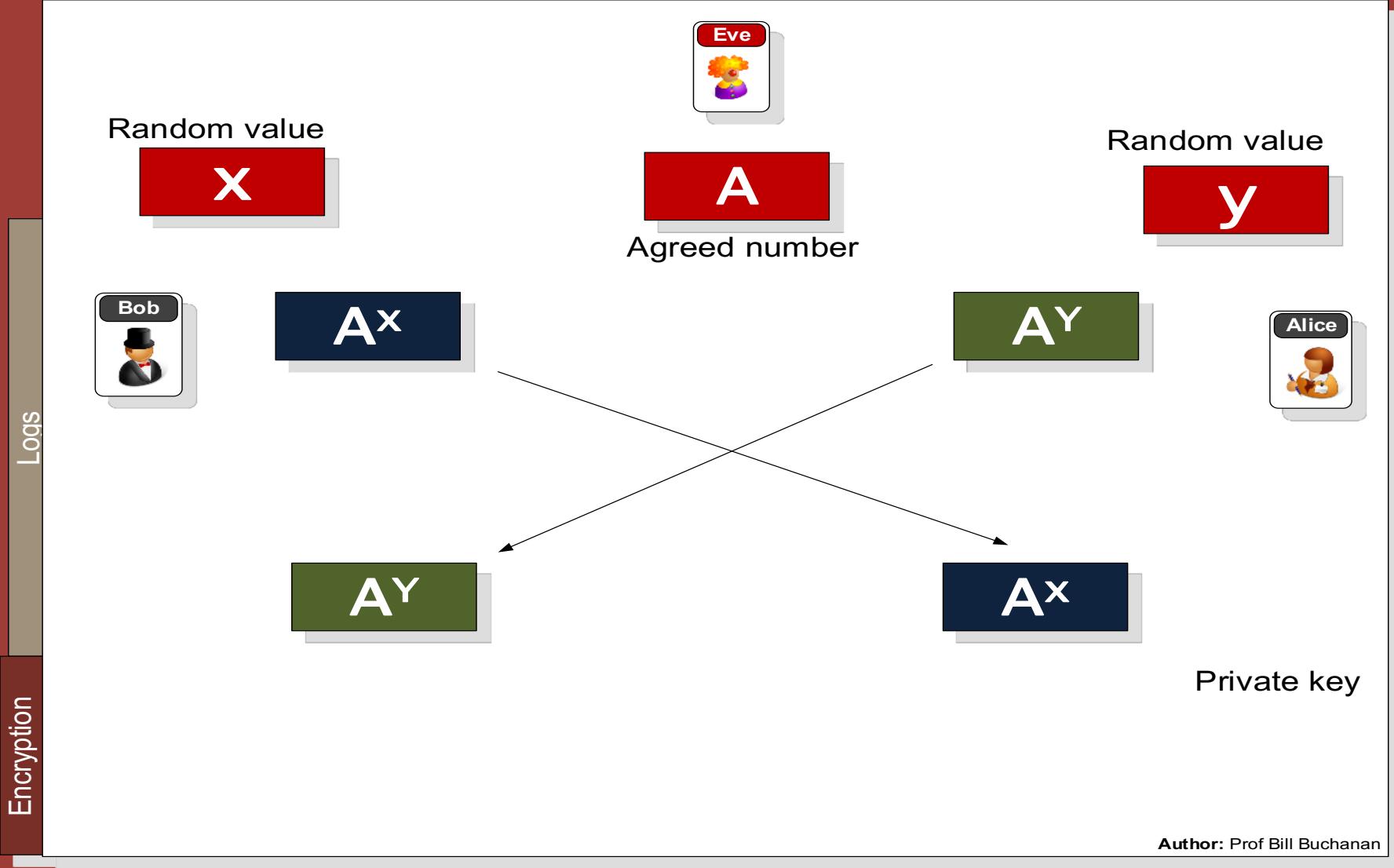
One of the most widely used methods for creating a secret key which is the same for Bob and Alice

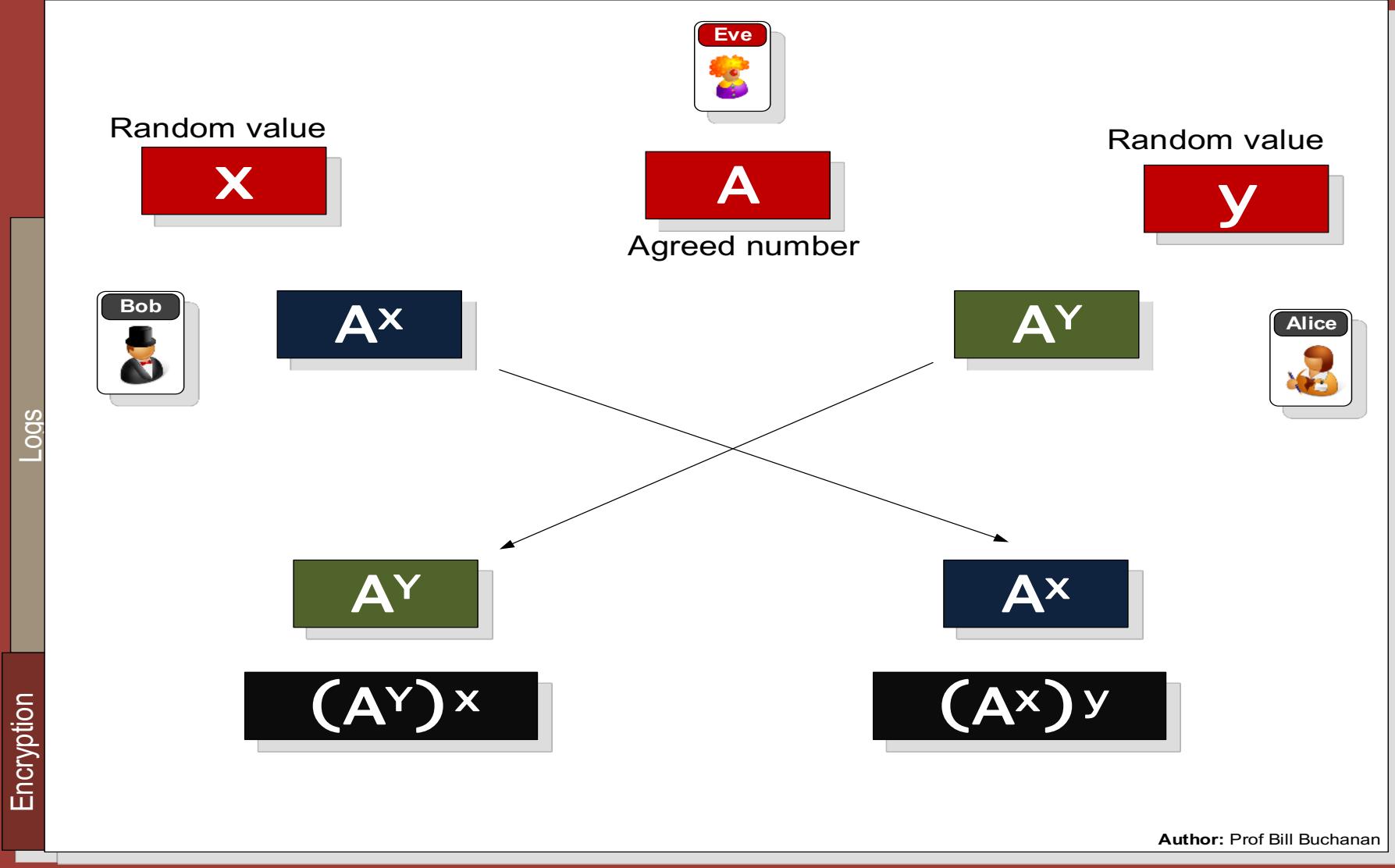


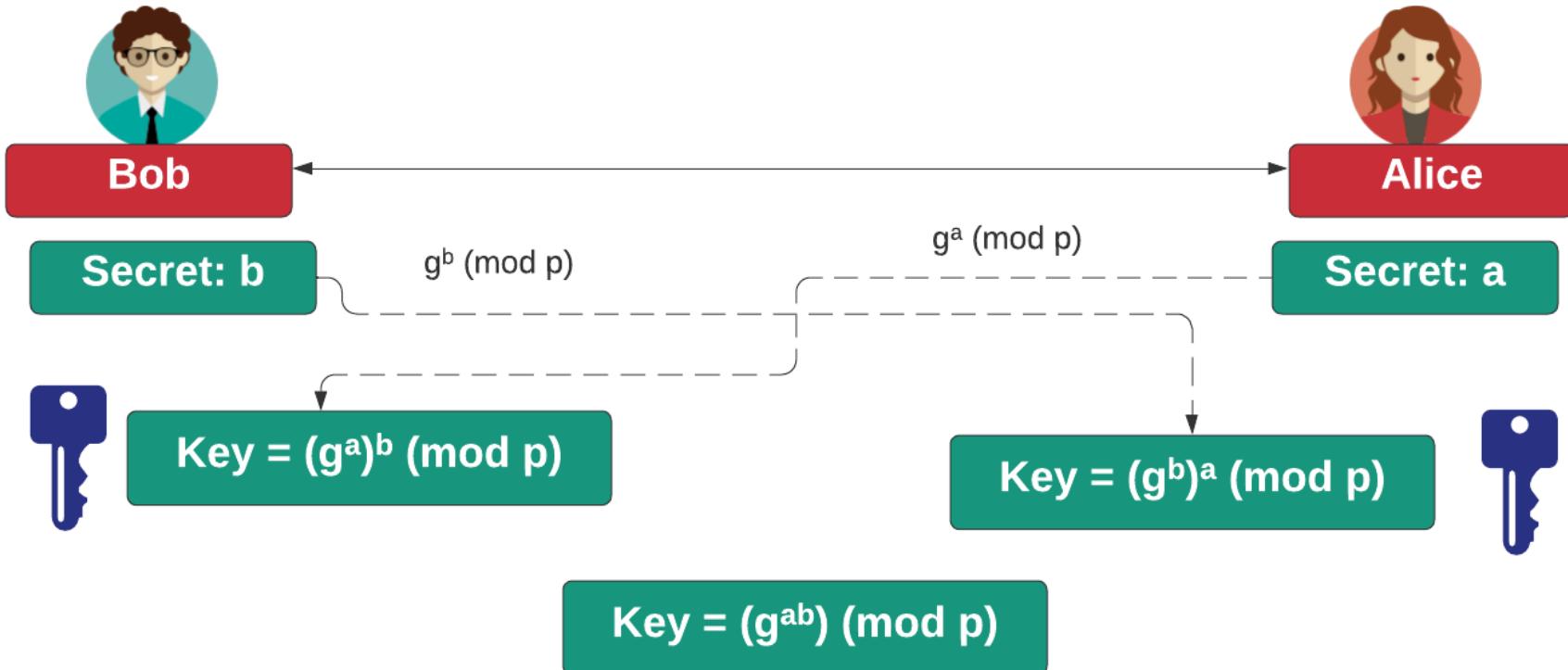
Key Exchange

- **Forward secrecy (FS)**, which means that a compromise 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.

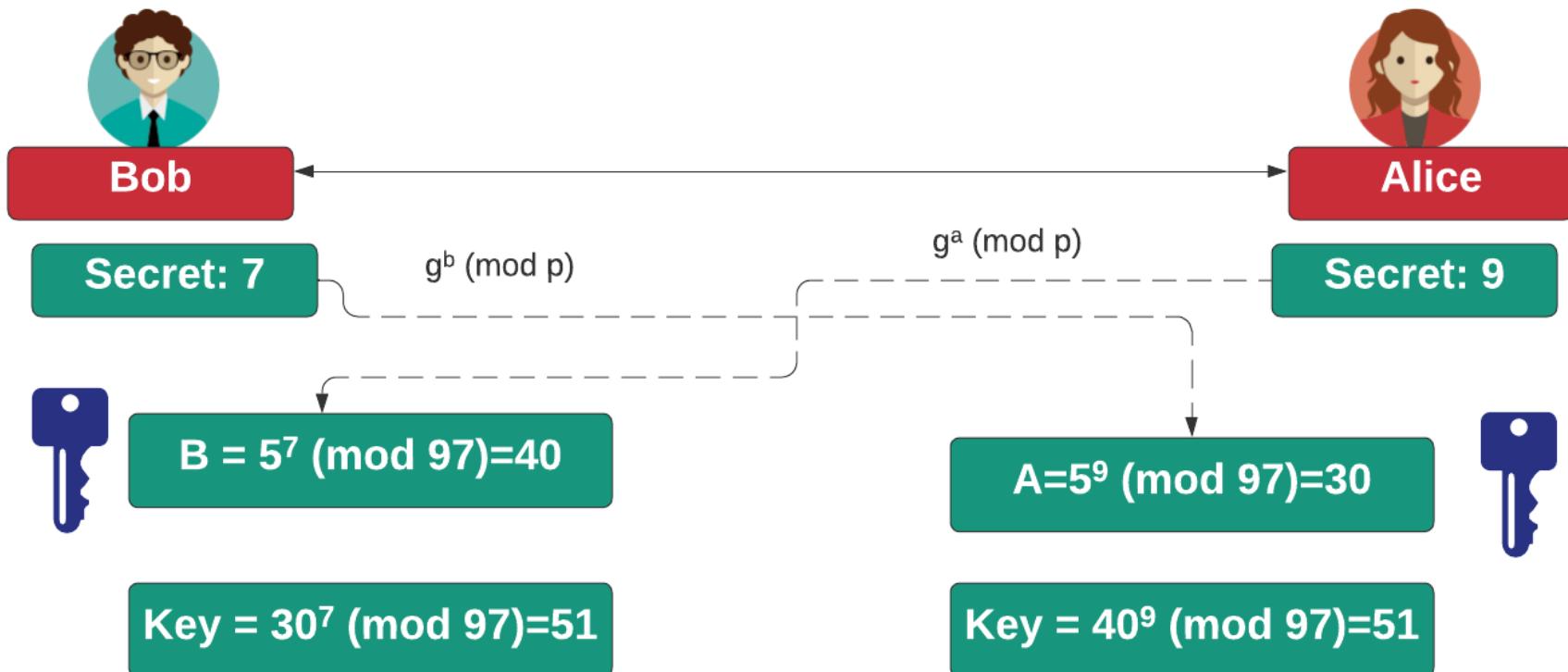


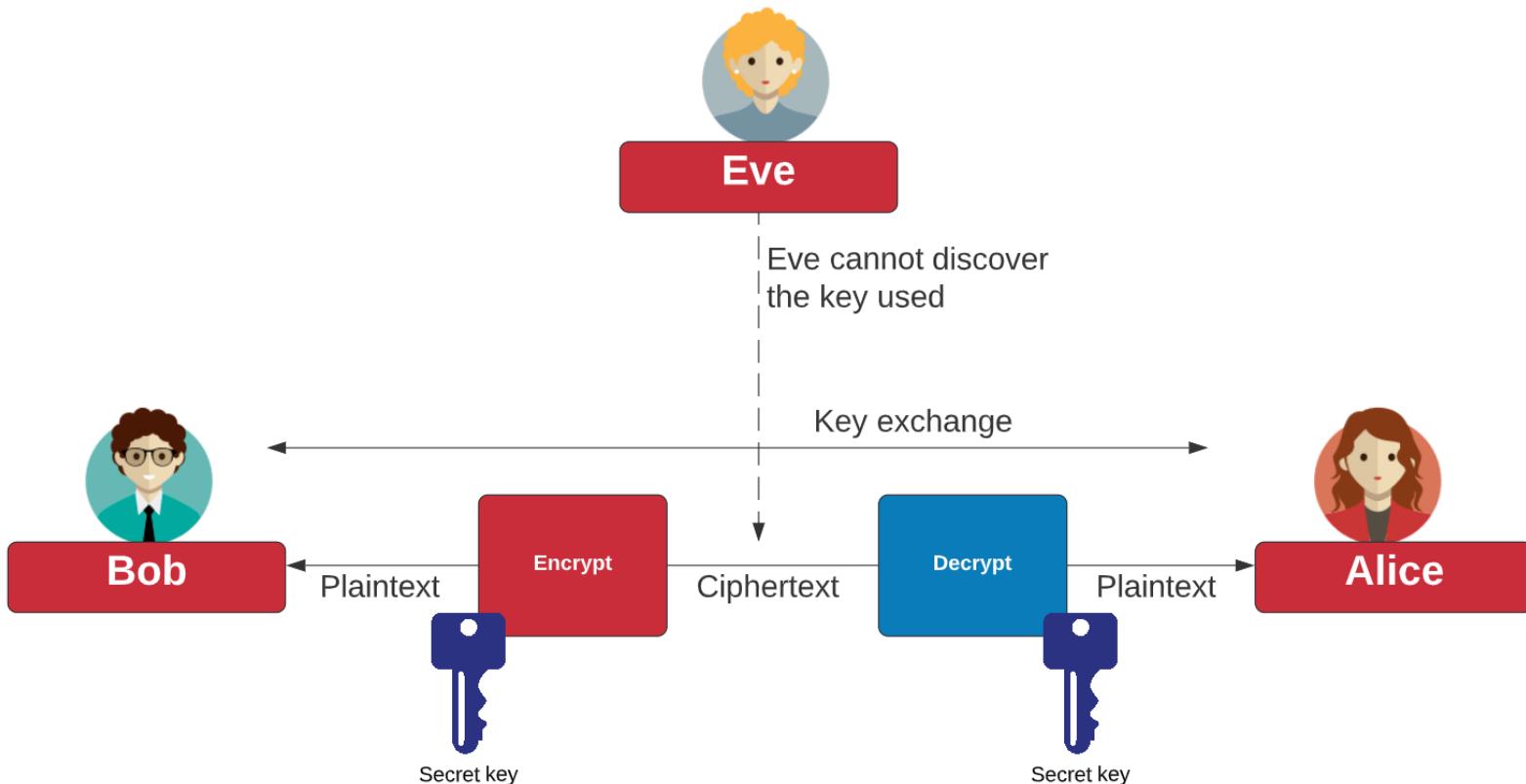






$$g=5, p=97$$





Diffie-Hellman Generator

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

p	11								
Generator	2	3	4	5	6	7	8	9	
x	$g^x \bmod p$								
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
```

```
+7g4LlcoONyVJHMG09kfr3gBEGp+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

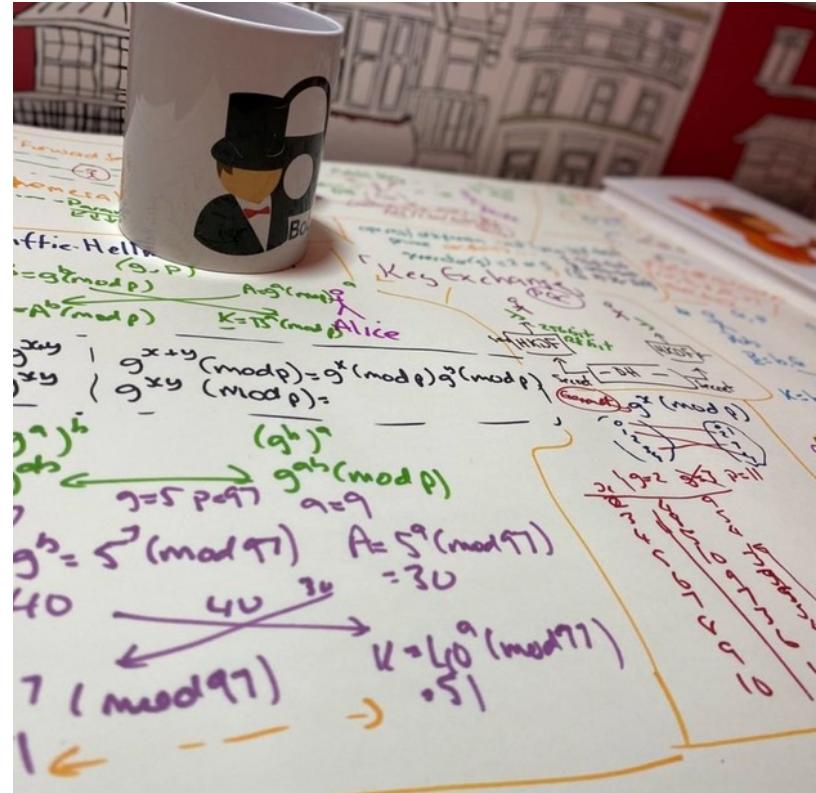
Diffie-Hellman Weaknesses

Passing Key Using Public Key

Prof Bill Buchanan OBE

<http://asecuritysite.com/crypto05>

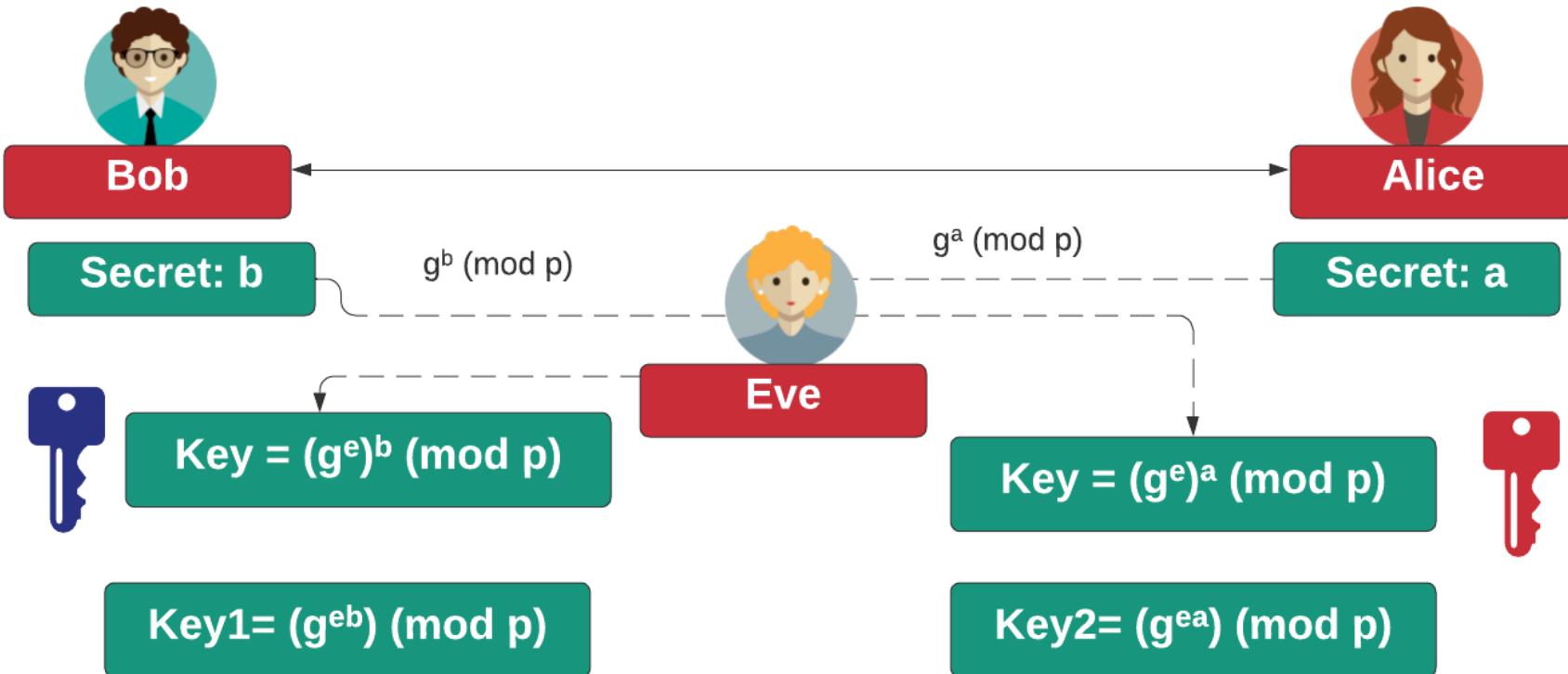
<http://asecuritysite.com/encryption>



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

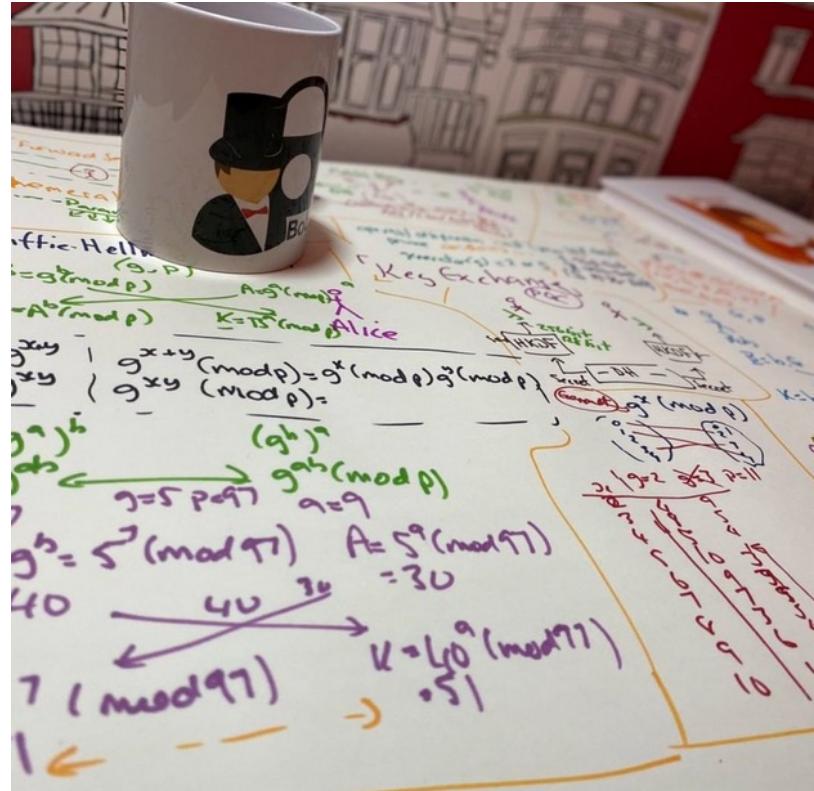
Diffie-Hellman Weaknesses

Passing Key Using Public Key

Prof Bill Buchanan OBE

<http://asecuritysite.com/crypto05>

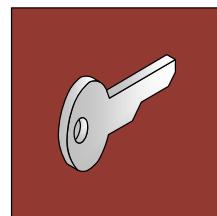
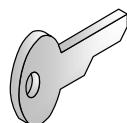
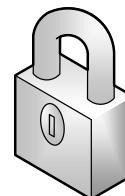
<http://asecuritysite.com/encryption>



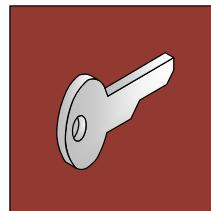
Key Exchange with Public Key



Alice sends Bob her public key



Bob creates a secret key and encrypts with Alice's public key



Alice decrypts with her private key



Key Exchange

Diffie-Hellman

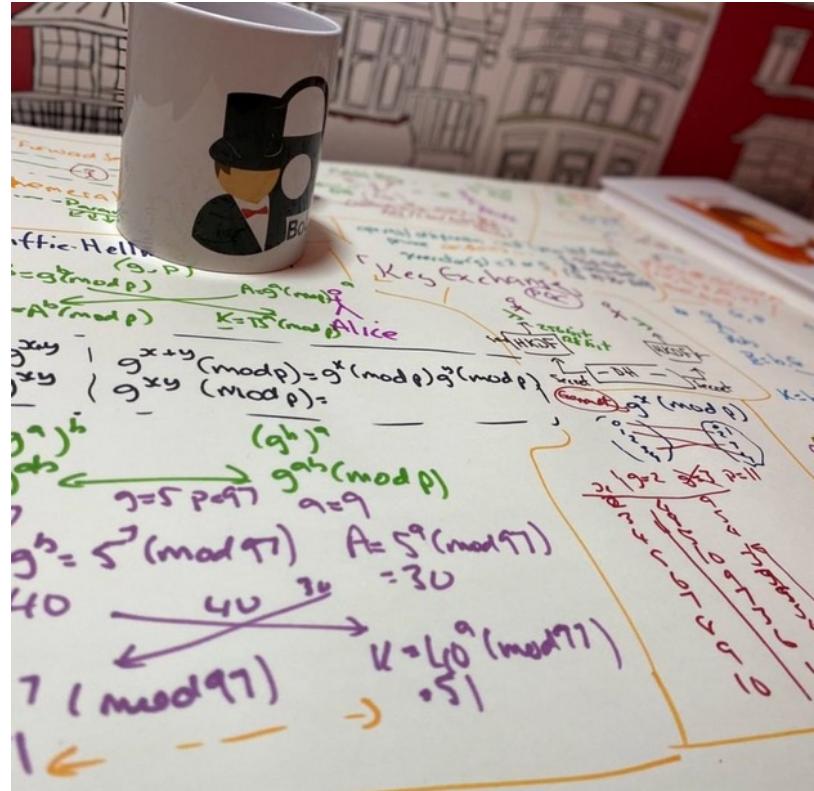
Diffie-Hellman Weaknesses

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)

Firefox

P Accept Online Payments And Mobile Pa...

Paypal, Inc. (US) | https://www.paypal.com/uk/webapps/mpp/home-merchant

You are connected to
paypal.com
which is run by
PayPal, Inc.
San Jose
California, US
Verified by: VeriSign, Inc.

The connection to this website is secure.

More Information...

However you do business, PayPal gets you paid.
Choose your payment solution, you can switch any time.

Accept card payments anywhere with PayPal Here™ [Learn More](#)

Eve

Trent

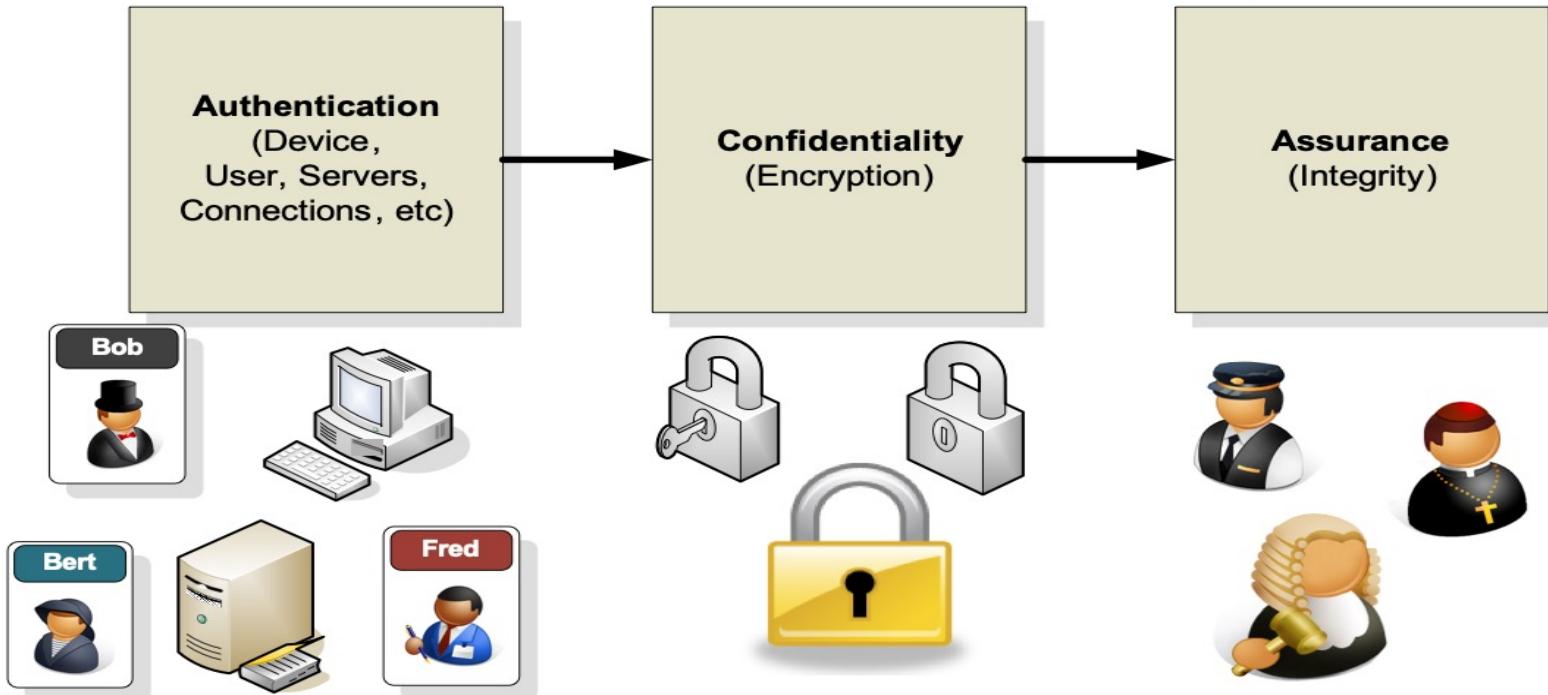


Bob

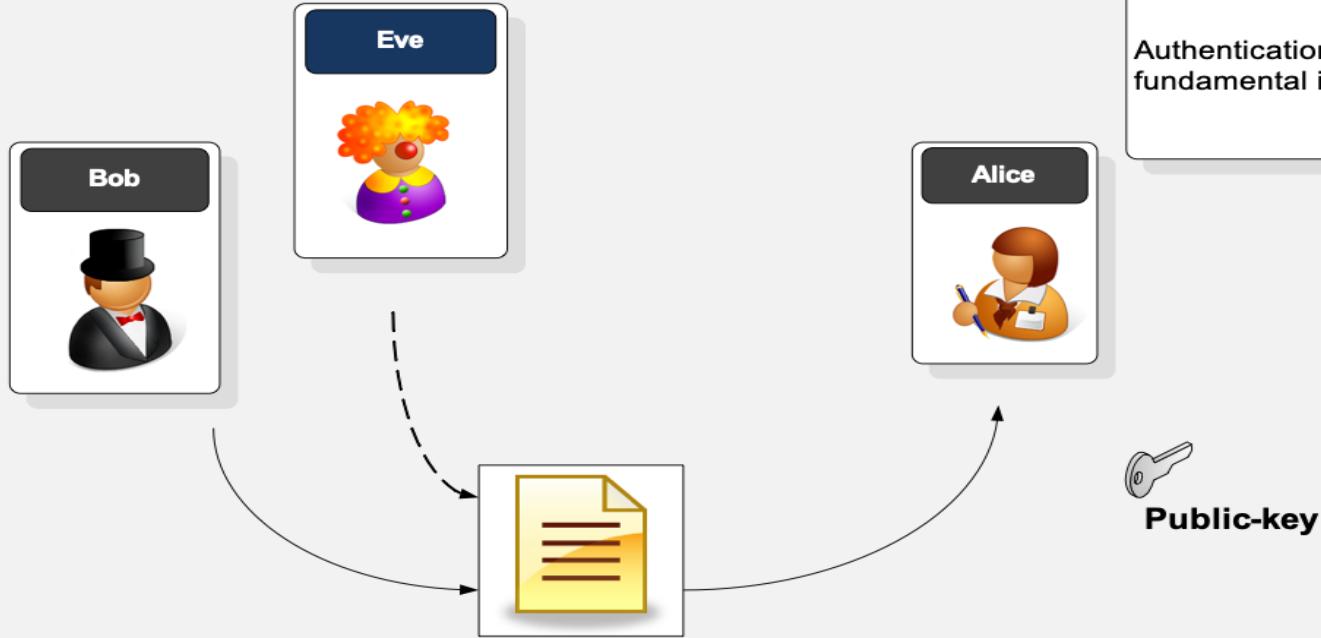


Fundamental principles

Authentication.
Confidence/Assurance.
Privacy/Confidentiality.

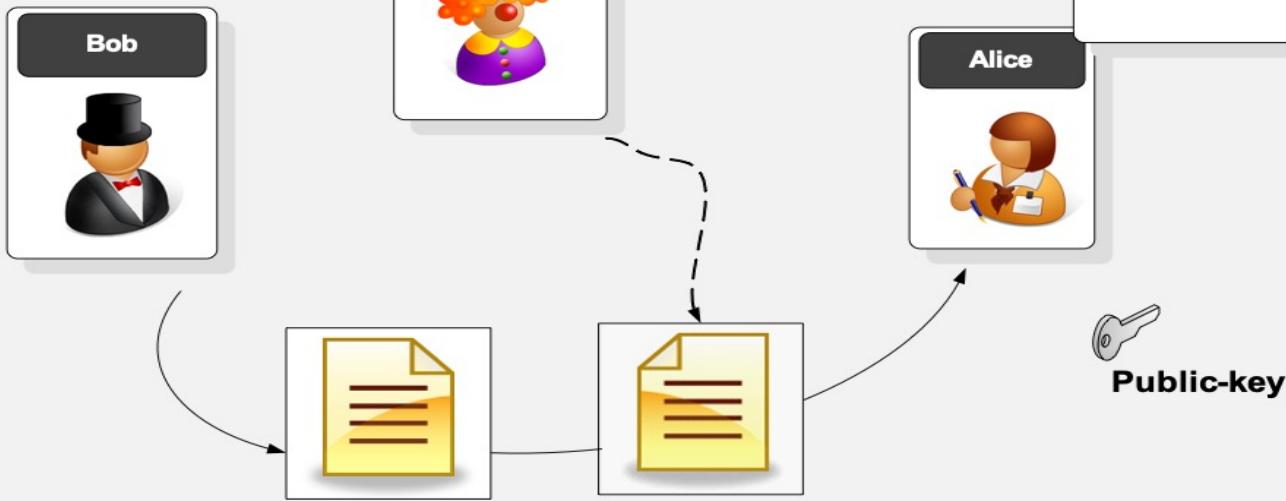


Authentication is a fundamental issue in security.



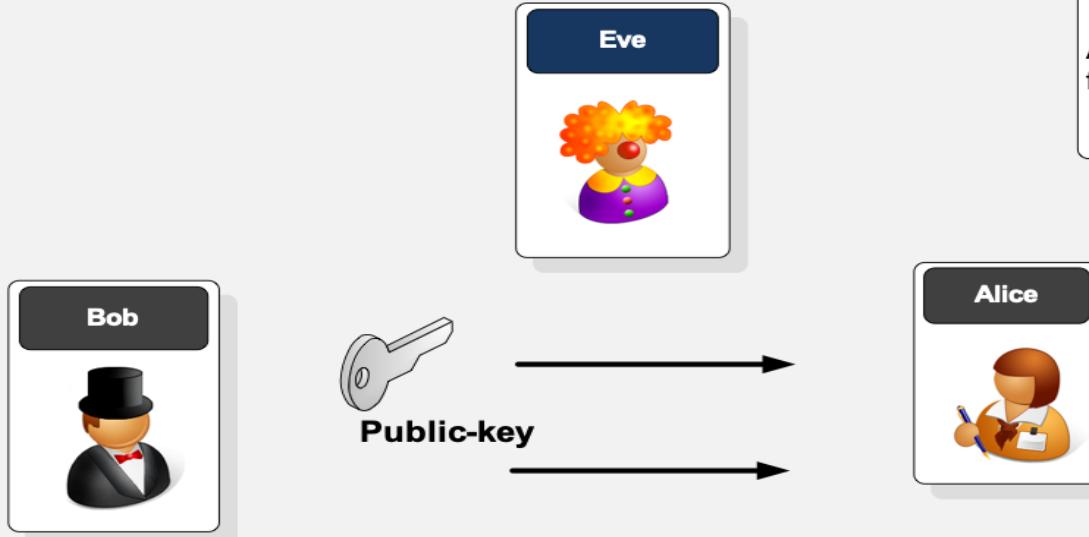
How do we know that it was really Bob who sent the data , as anyone can get Alice's public key , and thus pretend to be Bob?

Authentication is a fundamental issue in security

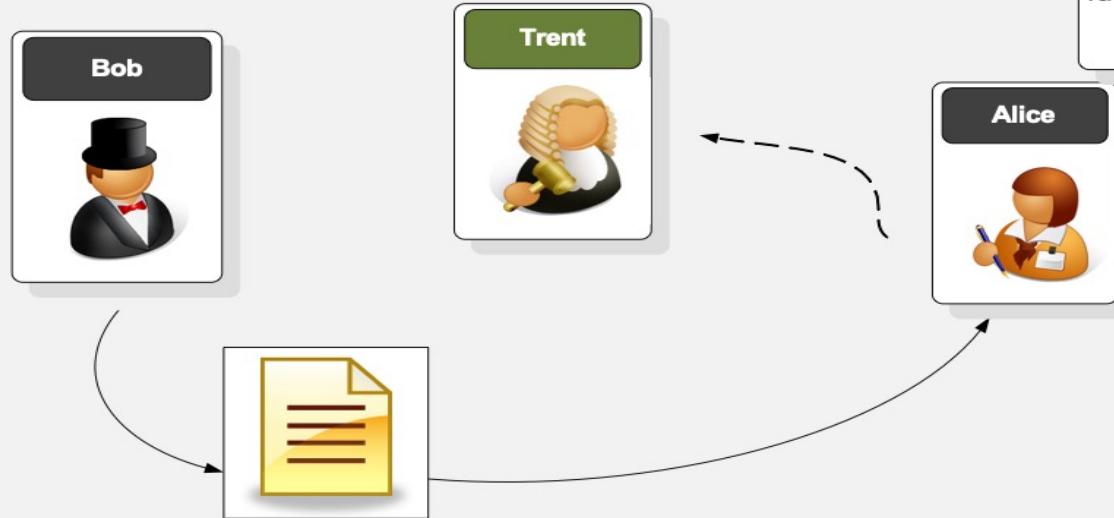


How can we tell that the message has not been tampered with ?

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 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.



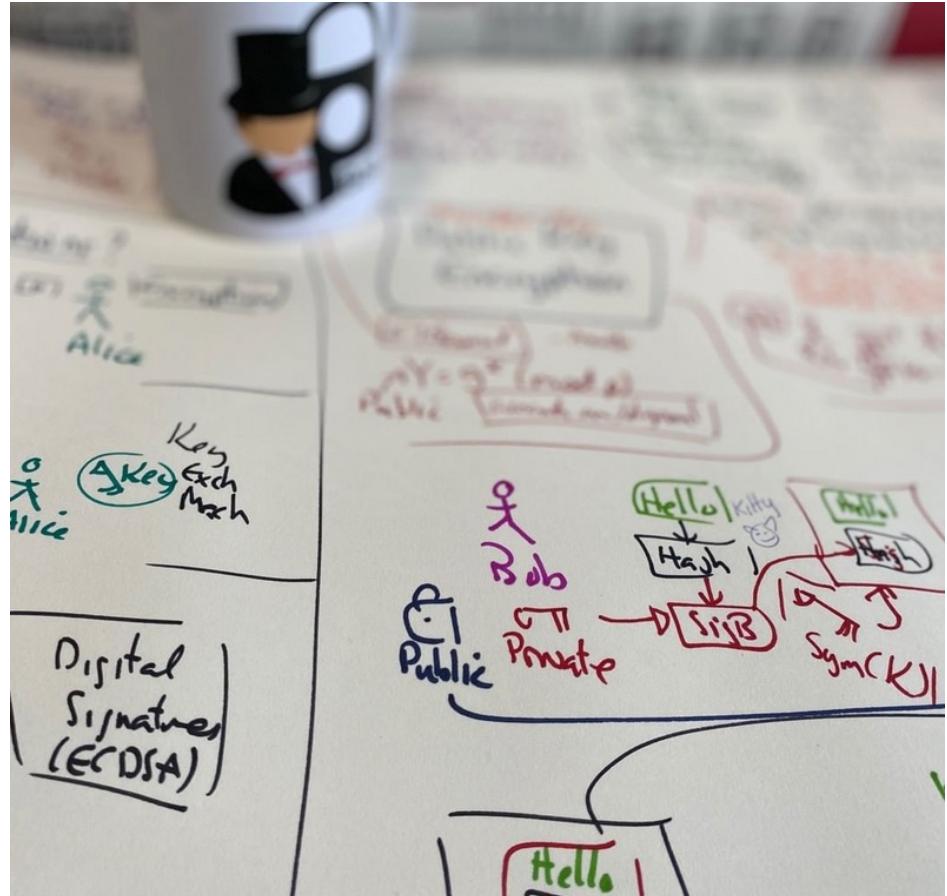
Unit 6: Digital Certificates

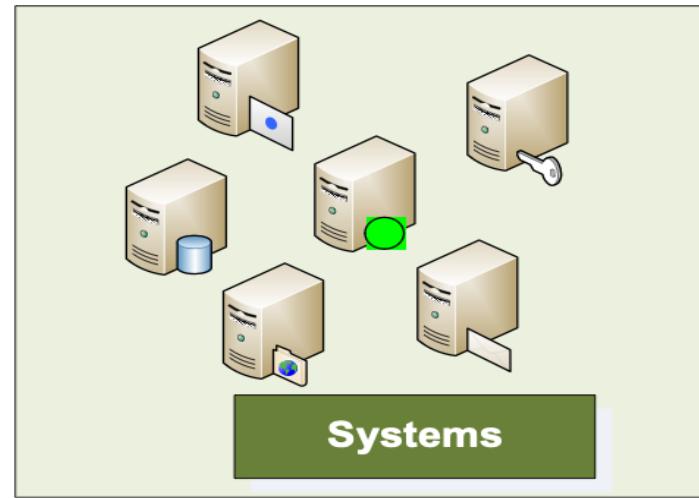
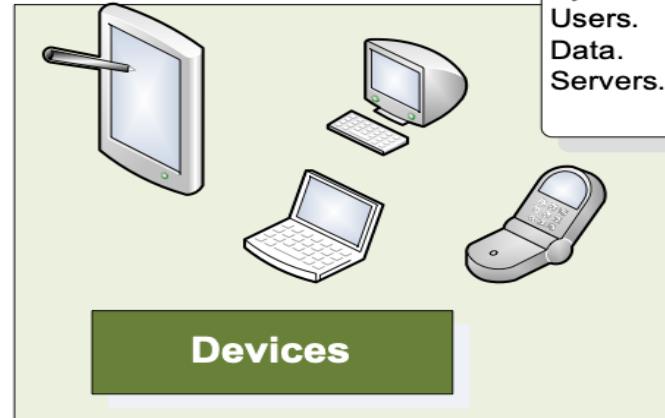
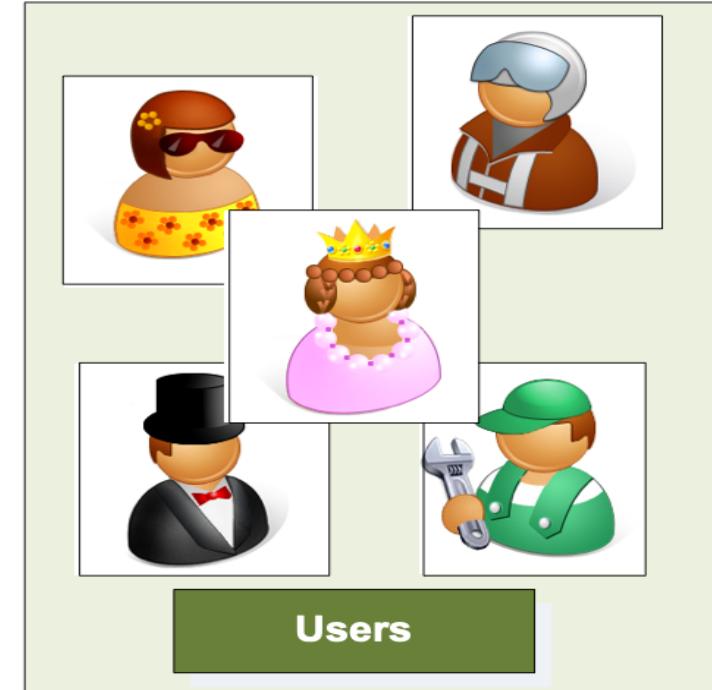
Introduction

Authentication Methods

Prof Bill Buchanan OBE

<https://asecuritysite.com/tunnelling/>





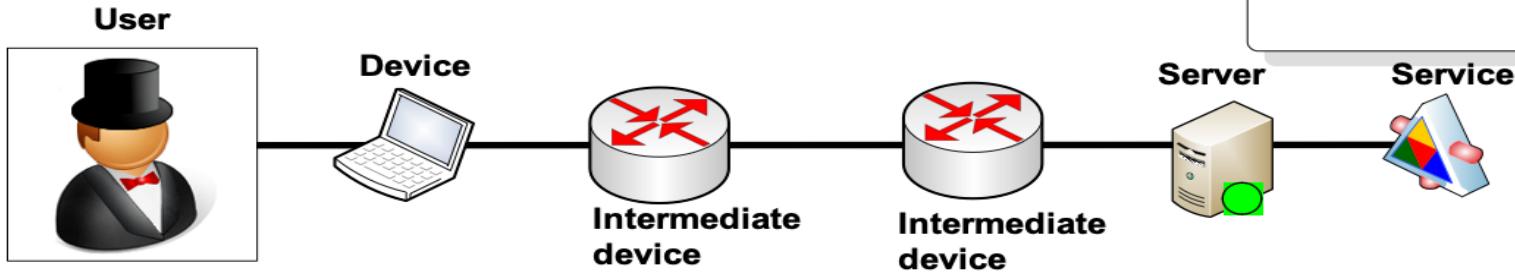
What to authenticate?

Systems.
Users.
Data.
Servers.

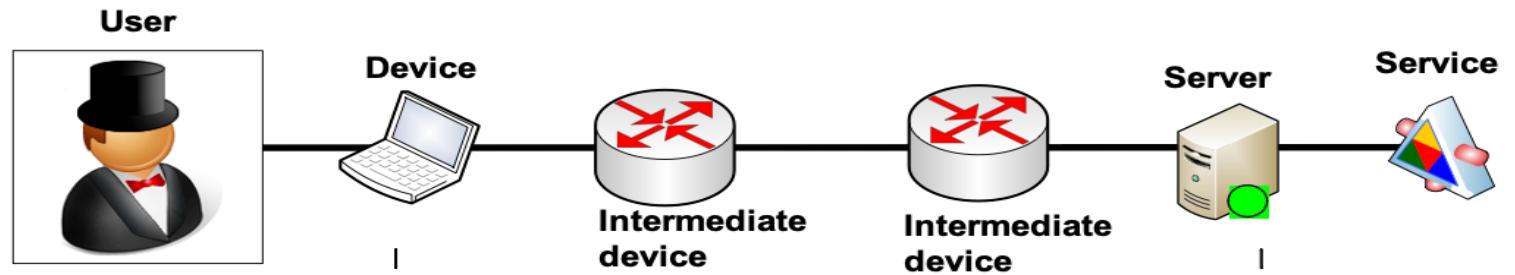
Where authenticated?

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

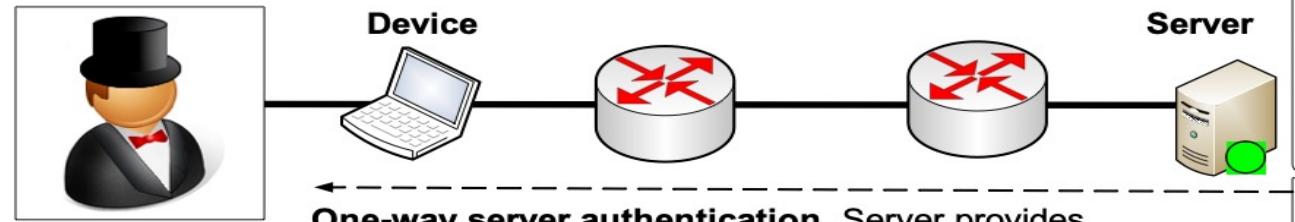
Authentication Methods



End-to-end authentication



Intermediate authentication

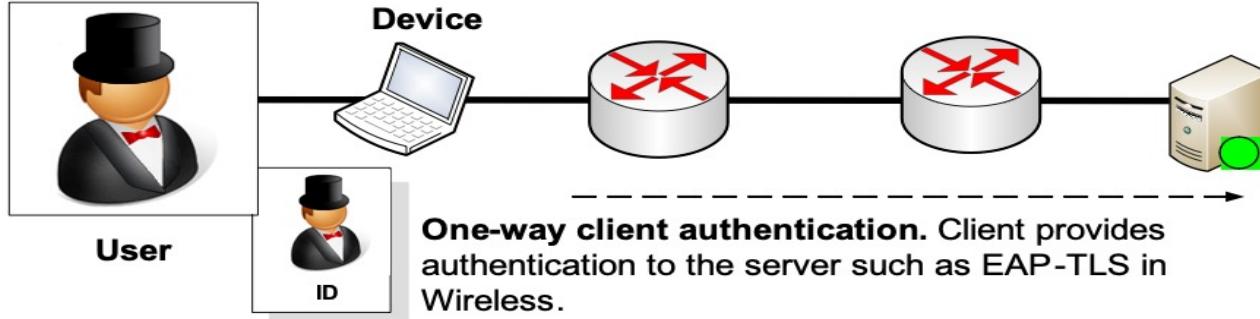


Authentication type

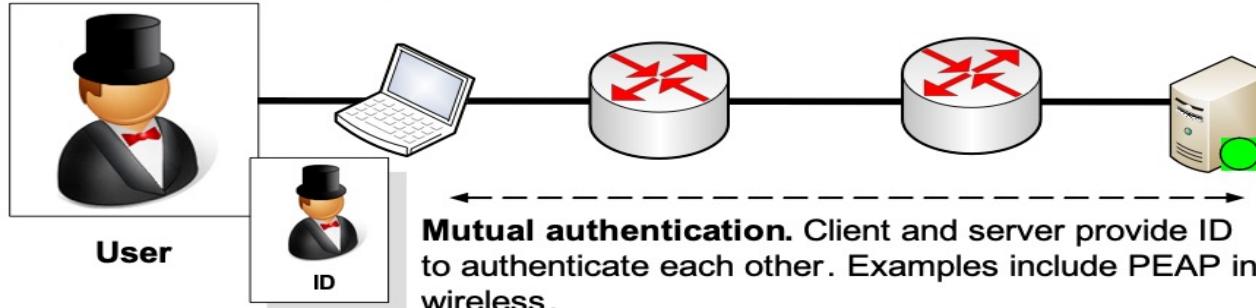
One-way server.
One-way client.
Two-way.



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



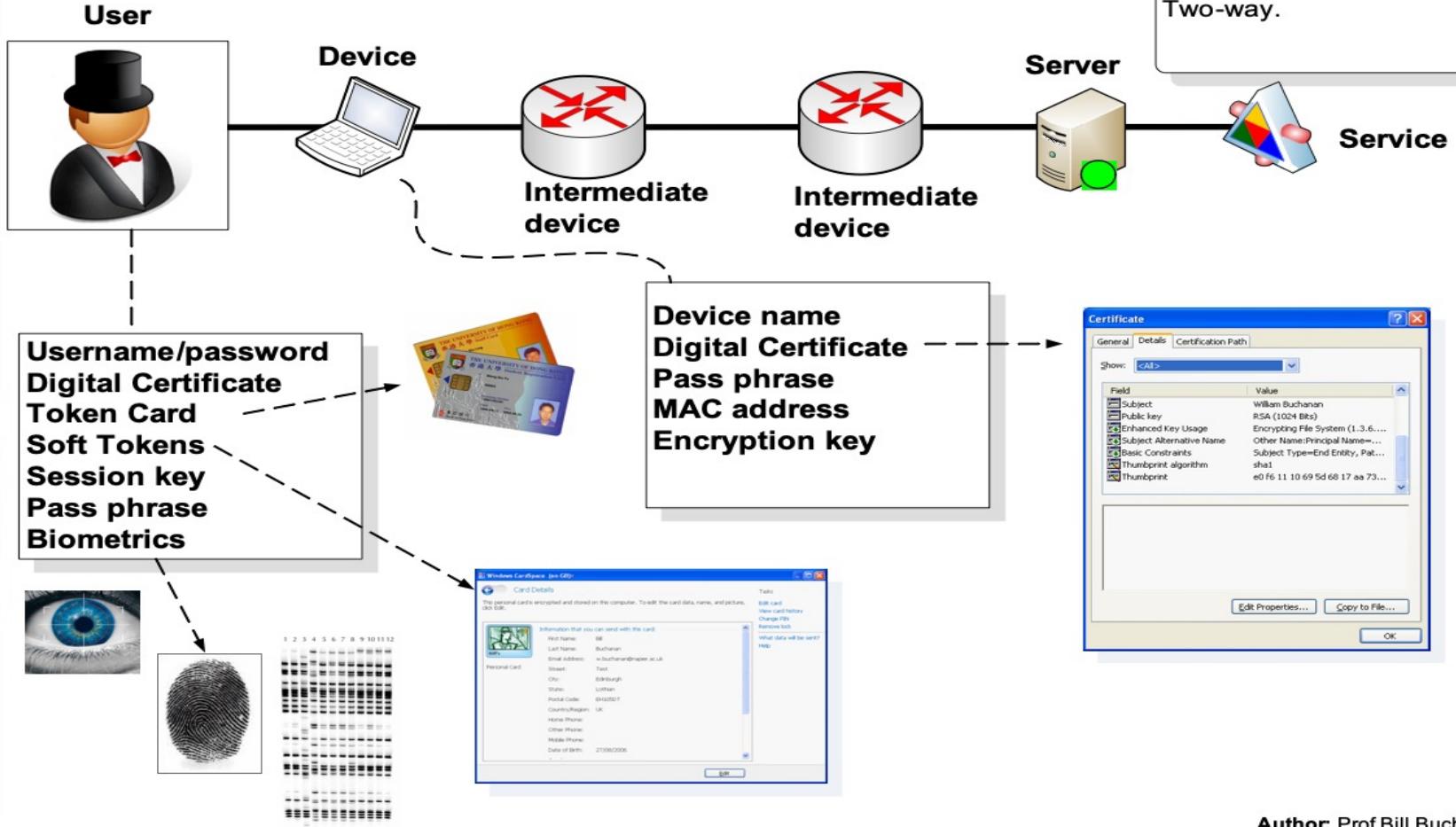
One-way client authentication. Client provides authentication to the server such as EAP-TLS in Wireless.



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



Authentication



Authentication type

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



Digital Certificates

Introduction

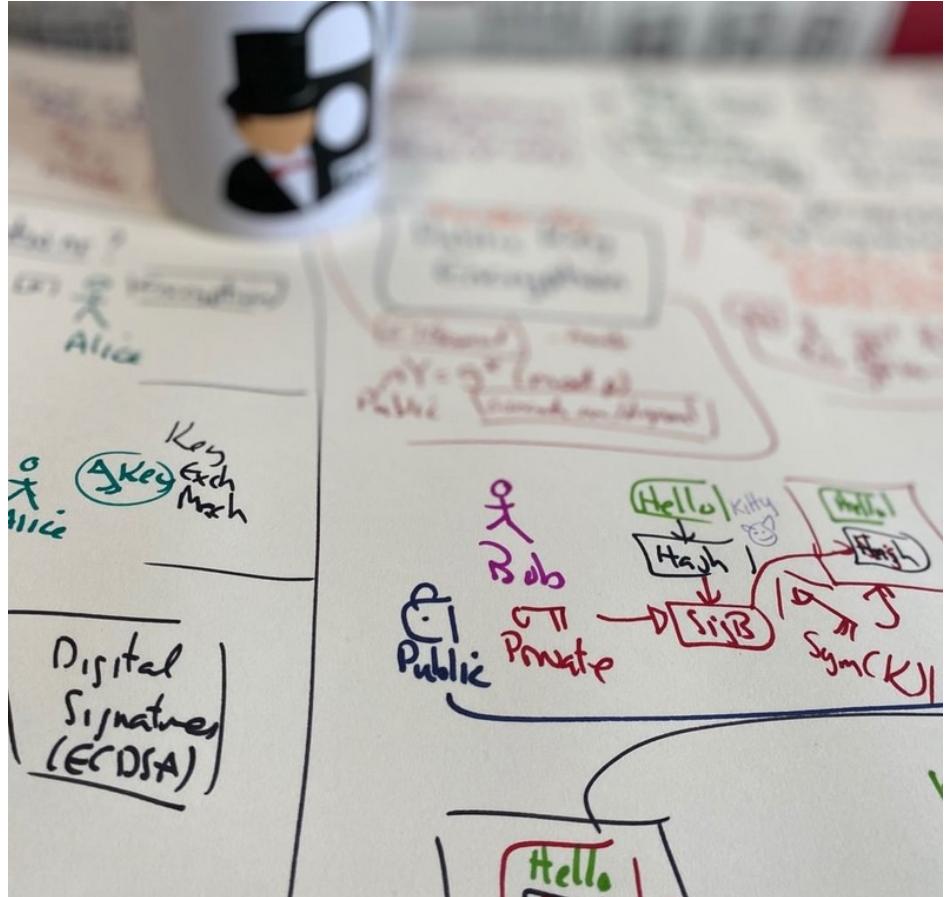
Authentication Methods

PKI

Digital Certificate Passing

Prof Bill Buchanan OBE

<https://asecuritysite.com/tunnelling>



Now that we need the public key to either encrypt data for a recipient, 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?



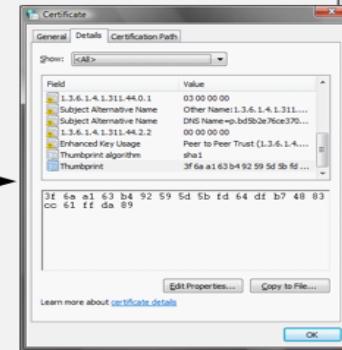
Public-key



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.)



Authentication

Digital Cert.

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
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 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
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...

Public-key

Certificate

General Details Certification Path

Show: <All>

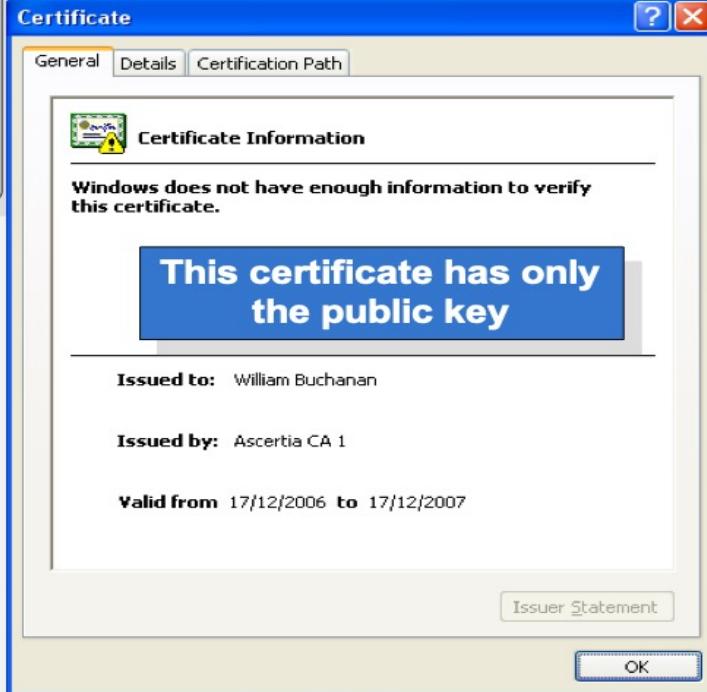
Field	Value
Version	V3
Serial number	58 74 4e 71 00 00 00 00 44 ba
Signature algorithm	sha1RSA
Issuer	Ascertia CA 1, Class 1 Certific...
Valid from	17 December 2006 21:04:49
Valid to	17 December 2007 21:14:49
Subject	William Buchanan, IT, Napier U...
Public key	RSA (2048 Bits)

CN = Ascertia CA 1
OU = Class 1 Certificate Authority
O = Ascertia
C = GB

Issuer

Edit Properties... Copy to File... OK

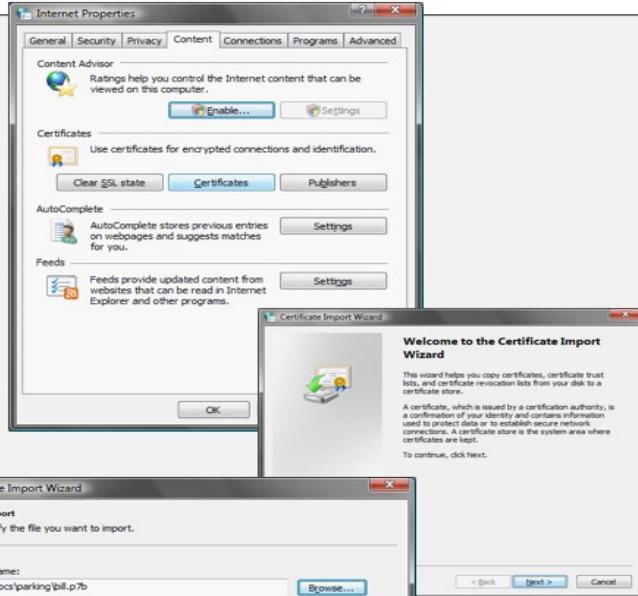
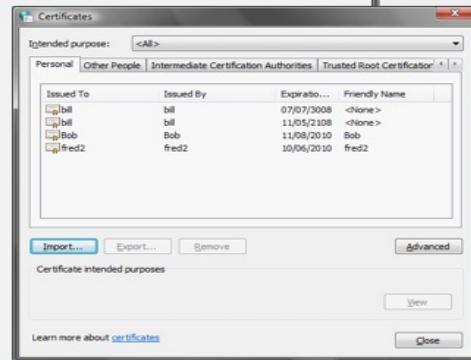
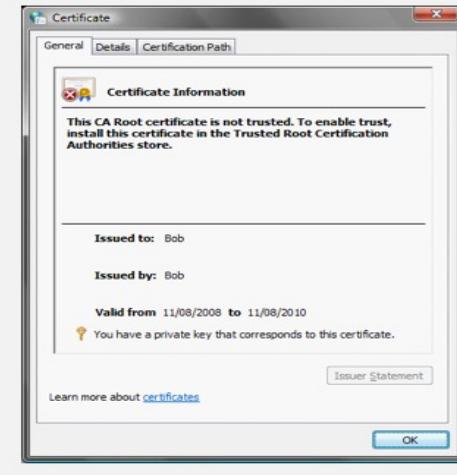
Digital certificate contains a thumbprint to verify it

Bob

Bob

P7b format

```
-----BEGIN CERTIFICATE-----
MIIDZCCA4wgAwIBAgIKWHR0cQAAAABeujANBgkqhkiG 9w0BAQUFADBgMQSwCQYD
VQQGEWJHqjERMA 8GA1UEChMIQXnjZXJ 0awExJjAkBgNvBAS THUnsYXNzIDEgQ_2Vv
dglmawNhgdUgox 0 ac9yaXR 5MRYwFAyDVQDew 1bc2n1cnRpysBDQSAxMB 4XDTA2
MTIxNzIxMDQ 0 0voxDTA3MTIxNzIxMTQ 0 0VowgZ 8xJjAkBgkqhkiG 9w0BCQEWf3cu
YnvjagFuYw 5AbmFwawVylMfjLnVmRQSwCQYDVQGEWJSZEOMA 4GA1UECBMHTG 90
ag1hbjEsmBAGA 1UEBxMjRWrbmJ 1cmdoMrRowGAYDVQKEXFOYKbpZXigVw 5pdmy
c210eTELMAKGA 1UECMCSVQGTXAxBgNVBAMTEfdpbGxpVw 0 qgnVjaGFuYw 4wgEi
MA0GCSqGSIb3DQEBAQAA4IBDwAwggEKAoIBAQCVCFETyJL 8VxAhEMRzQ10gM81
ci75nmMs0amjzcB 6fhGmGownycoscmQkrVjAknoS +4mxnhcy3mdob+szbwVaX
M5FoXhsrV+Q86hsk8Cdc+1sqy3TQtqfuDns0nfNY6tR6q7CgGqQ8/VjSxnzK 39
iLUFlahycet/ab60/gwzL4ivsz2nml4dyauyt1hLP1VbppHGde 6sDQXWyd0cpfv
ZN7paud5fqBESf06bukcieI47AzRMQj 3kHuDt7MexVw7aoX+nXLP4wn7iamaxasF
QvhodkyCzhYs B2JQDGatXRCqkk1tzmz 5i6GkPse7XvuX265Wjq5fhp2hY1AgMB
AAGjggEXMIEBEzAdbgNVHQ 4EFgQUzy/YccJwT5opPHLP1CQkkolkjwwyyvDR 0j
BFwwloAUlP 5Zh0V700k6CorvRMwB9ifvkBmhP6Q9MDsxZ2AjBgNVBAYTAkdCMREW
DwYDVQKEwhBc 2N1cnRpYTCBcgA 1UEAxM0QXnjZXJ 0awEgum9vdCBQYIBDTBN
BgNvHR8ERjBEMEkQKA+hjxodhrwo1 8vd3dLmfzY2VydG1hlmNbVs 5PbxpbmVD
QS9jcmxzL0Fz2Vdg1hQ0ExL2NsYXNzMS 5jcmrwPgYKwYBQHUQAQEMJAWMC 4G
CCsGAQFBzAChiJodhrwo1 8vb2Nzcc5nbG9iYwx0cnvzdgZpbmr1ci 5j2b20vMA0G
CSqGSib 3DQEBBQAA0EATOCwgJ 1t50kt1upmpjkM1 8idxMmD5wuhszjB1GsMhPxI
H+vXhL9yaOw+Prpz7aJS4/3xxu8vRAnhyu 9yu4qDA==
-----END CERTIFICATE-----
```



- The main certificate formats include:**
- P7b. Text format
 - PFX/P12. Binary.
 - SST. Binary.

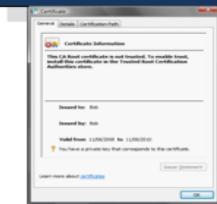


- A. Bob creates the message.
- B. Bob encrypts with Alice's public key and sends Alice the encrypted message
- C. Alice decrypts with her private key
- D. Alice receives the message



Hello

H&\$d .

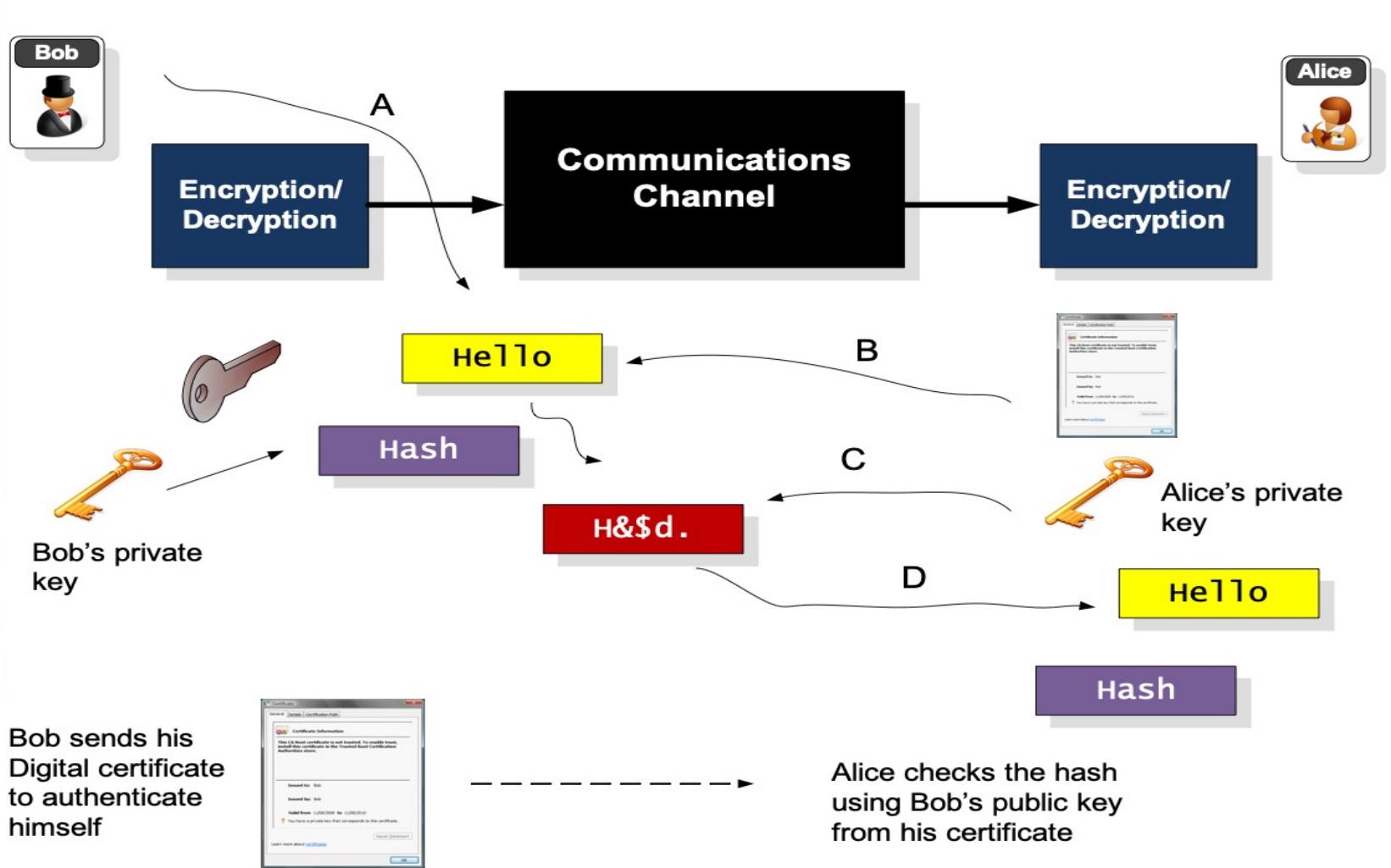


Alice sends her digital certificate with her public key on it



Alice's private key

Hello



Bob sends his Digital certificate to authenticate himself



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

Digital Certificates

Introduction

Authentication Methods

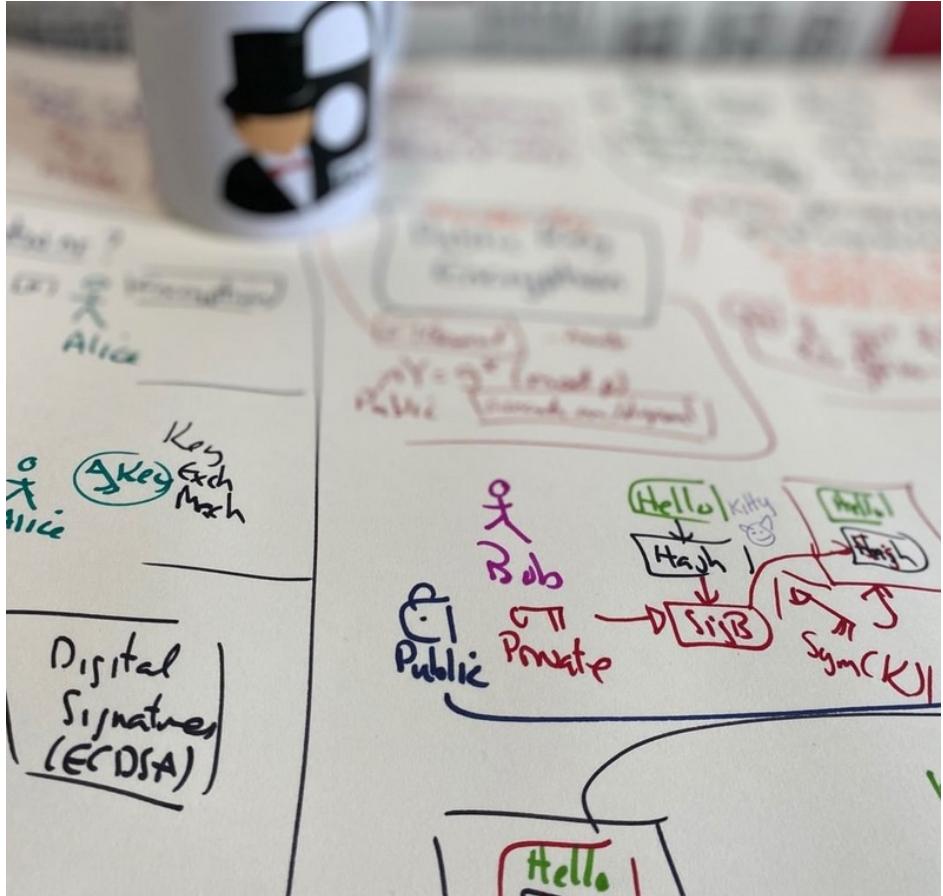
PKI

Certificate Creation

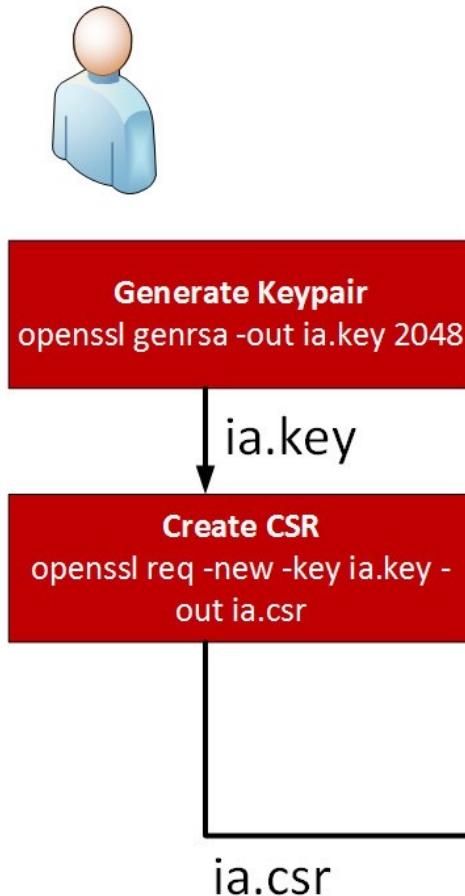
Digital Certificate Passing

Prof Bill Buchanan OBE

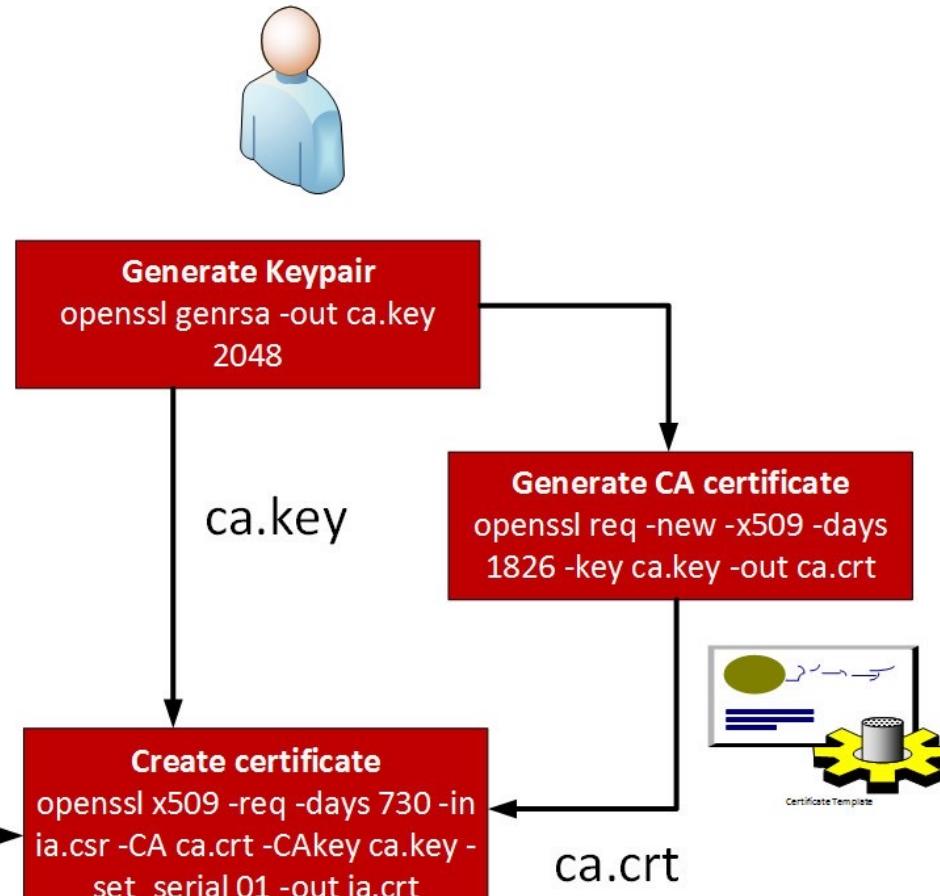
<https://asecuritysite.com/tunnelling>

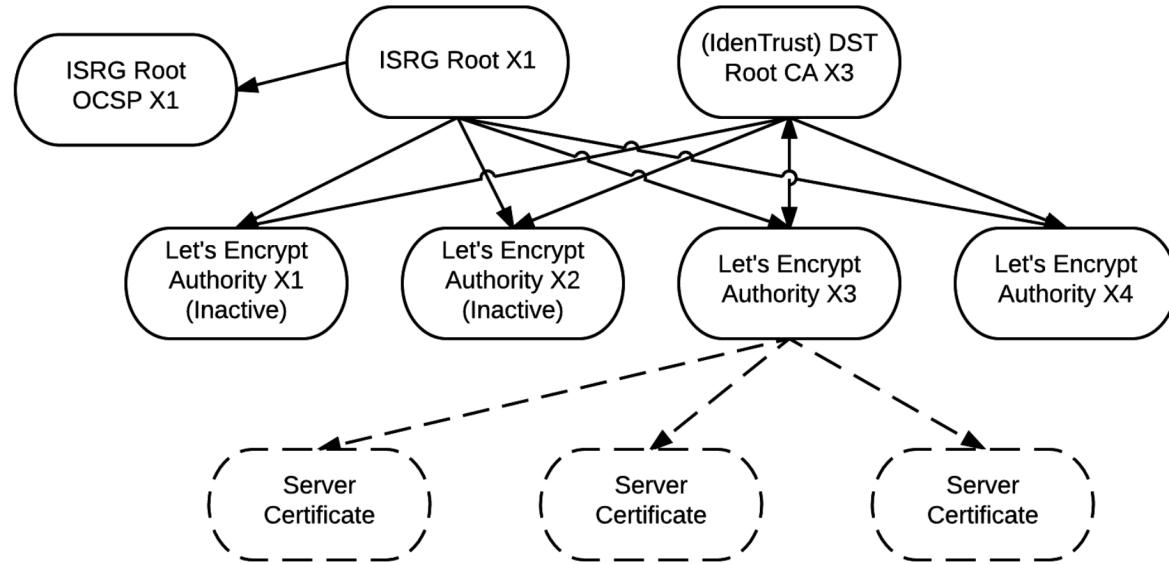


Organisation



Root CA





-----BEGIN CERTIFICATE-----

PEM

MIIEwTCCA6mgAwIBAgIRA093GGFLf
 QjELMAkGA1UEBhMCVVMxHjAcBgNVBAoTFUdvb2dsZSBucnVzdCBTZJ2aWNLczET
 MBEGA1UEAxMKR1RTIENBIDFPMTAeFw0yMDAyMTIxMTQ3NDFaFw0yMDA1MDYxMTQ3
 NDfaMGgxCzAJBgNVBAYTA1VTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRYwFAYDVQHQH
 Ew1Nb3VudGFpbIBWaWV3MRMwEQYDVQQKEwpHb29nbGUgTExDMRcwFQYDVQQDEw53
 d3cuZ29vZ2x1LmNvbTBZMBMGBYqGSM49AgEGCCqGSM49AwEHA0IABCzzLJYFJb8W
 TpQxWL01vSDvEWCKed7181CFspT60kn13YILNdTM22sUwPcyogKjBSaQZ9Axi

- 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.

t1HtqVqITZPCAI/ANvbHdZiMEePB8eA+oTiw2ucChqLlsop5Mio8ckg7aXG4/QfC
 AIDbvkoFFK44rs4UEpsaqGe90qmMjjTu07ZCzrFd1m3geq2ARKPHgPoPM6UbDdqg0
 TNQo9C0F5Zl0k0gV/qshvpT04YzE7TB2U571eYEeqNXH48syVx8XSk3P7FjM7FIZ2
 IzbJSEipZJm8DsP10fFXpToIn+zK

-----END CERTIFICATE-----

PKCS#7

-----BEGIN PKCS7-----

MIIE8gYJKoZIhvCNQcCoII4zC0...9w0BBwGgggTR
 wTCCA6mgAwIBAgIRA093GGFLfHw0CAAAAAAcZgwDQYJKoZIhvCNQELBQA...
 MAKGA1UEBhMCVVMxHjAcBgNVBAoTFUdvb2dsZSBucnVzdCBTZJ2aWNLczET
 A1UEAxMKR1RTIENBIDFPMTAeFw0yMDAyMTIxMTQ3NDFaFw0yMDA1MDYxMTQ3
 MGgxCzAJBgNVBAYTA1VTMRMwEQYDVQQIEwpHb29nbGUgTExDMRcwFQYDVQQDEw53
 d3cuZ29vZ2x1LmNvbTBZMBMGBYqGSM49AgEGCCqGSM49AwEHA0IABCzzLJYFJb8W
 WL0TySDvEWCKed7181CFspT60kn13YILNdTM22sUwPcyogKjBSaQZ9Axi
 EAiJkbejggJVMICUTAOBgNHQ8BaF8EBAMCB4AwEwDYDR01LBawwCgYIKwYE
 AwEwDAYDR0TAQH/BAiWAdAdBgNHQ4EFgQU9Ty3t9007UW9+Hc6kv/j9511
 HwYDVR0jBBgwFoAUmNH4bhDrz5vsYJ8YKBUG630J/SswZAYIKwYBBQUHAQE
 MCcGCCsGAQUFBzAhhhtodHRw0i8vb2NzcC5wa2kuZ29vZy9ndHmxzbEwKwYI
 BQUHMAKGh2h0dHAGLy9wa2kuZ29vZy9nc3IyL0dUzFPMS5jcncQwGQYDVR0
 EII0d3d3Lmdvb2dsZs5jb20wIQYDVR0gBBowGDAIBgZngQwBAgIwDAYKKwYE
 eQIFAzAvBgNVHR8EKDAmMCsgIqAghh5odHRw0i8vY3JsLnBras5nb29nL0d
 MS5jcmwggEFBgorBgEEAdZ5AgQCBH2BIHzAPEAdgCyHgXMi6LNiiB0h2b
 JSBna9r6c0eySvMt74uQXgAAAXA5cNqwAAAEAwBHMEUCIQCojKtz0e8l1JYK
 bt1puqt4AE3peMVAVk/WewGp1QIgBvRmbNkwF6i8+JVv3CfTHKvFd8e00+G
 Zb1drKEAdwBep3P531bA57U2SH3QSeAyeGaDIShEhKEGHwWgXFFWAAAAXA5
 AAAEAwBIMEYCIQCOS0NppELPOdH65oFT5ZAGsSQz4FsJNNZedgS7WqzLnQI
 u0QuuoE3RWngfhI6W7n1kxkEmd0vSZTe6+0l+JVuMA0GCSqGSIB3DQEBCwUA
 AQCDU8CVusGstVkk1Amrtg3DyYhNV1UnKyLk3RrHKumwYz5mC25bWGoLVKv
 N3za329/9JZq0mn0vpmPxjTDvh9sVQ7g4znwha/KJfzL8AZKudldD90+hD0
 qVqITZPCAI/ANvbHdZiMEePB8eA+oTw2ucChqLlsop5Mio8ckg7aXG4/QfC
 vkoffK44rs4UEpsaqGe90qmMjjTu07ZCzrFd1m3geq2ARKPHgPoPM6UbDdqg0
 9C0F5Zl0k0gV/qshvpT04YzE7TB2U571eYEeqNXH48syVx8XSk3P7FjM7FIZ2
 SEipZJm8DsP10fFXpToIn+zKoQAxAA==
 -----END PKCS7-----

Base64

Binary

DER
CER

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

Digital Certificates

Introduction

Authentication Methods

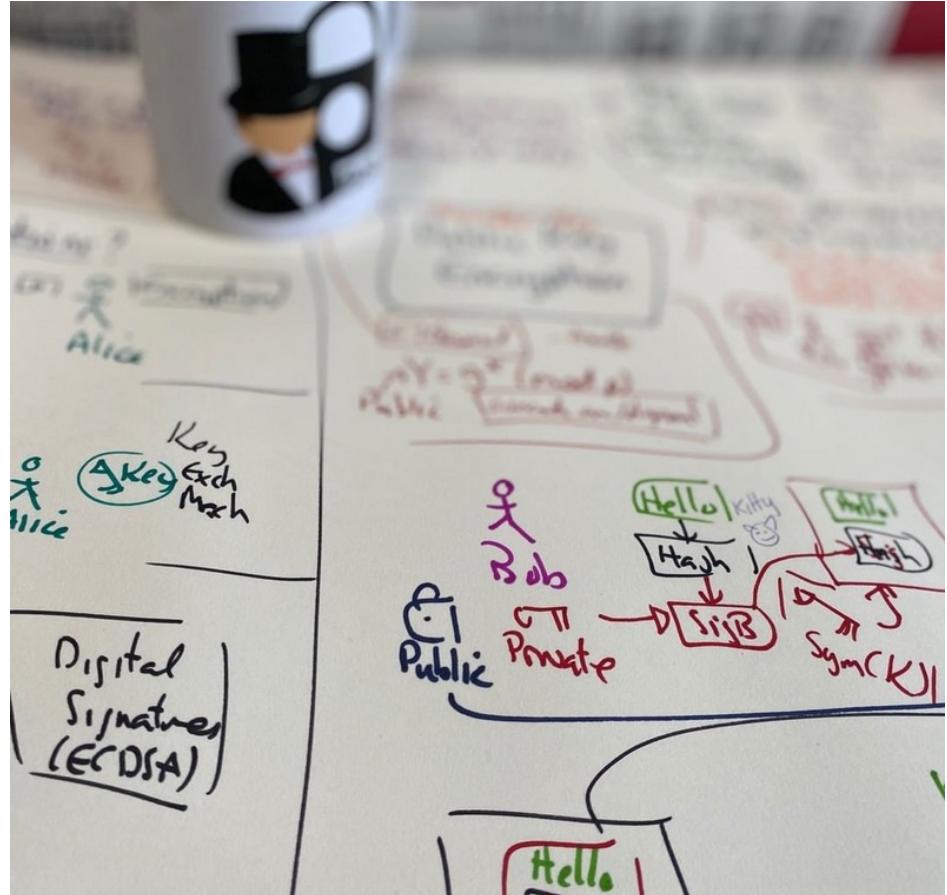
PKI

Digital Certificate Passing

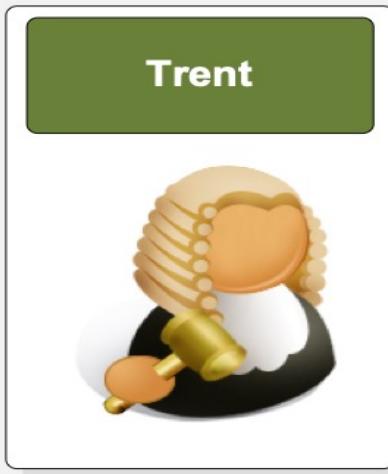
Prof Bill Buchanan OBE

<http://asecuritysite.com/crypto06>

<http://asecuritysite.com/encryption>



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



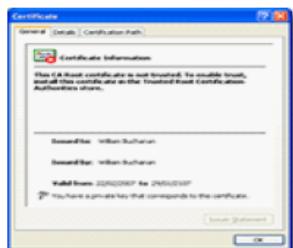
Digital Certificates

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

Trusted Root CA



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



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

Trent



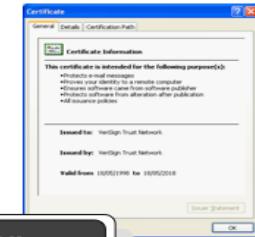
Trusted root certificates are installed as a default on the machine (or installed with the user's permission)

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.



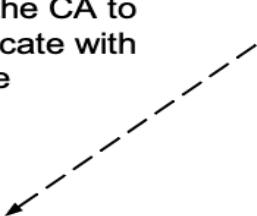
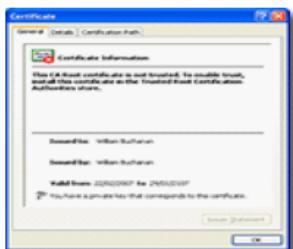
Author: Prof Bill Buchanan



Trusted Root CA



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



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

Trent



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

Certificates

Intended purpose: <All>

Intermediate Certification Authorities Trusted Root Certification Authorities Trusted Publ

Issued To	Issued By	Expiration...	Friendly
Microsoft Authenticode(tm)...	Microsoft Authenticode(tm)...	31/12/1999	Microsoft
Microsoft Root Authority	Microsoft Root Authority	31/12/2020	Microsoft
Microsoft Root Certificate ...	Microsoft Root Certificate ...	09/05/2021	Microsoft
NetLock Expressz (Class C...)	NetLock Expressz (Class C...)	20/02/2019	NetLock I
NetLock Kozjegyzoj (Class ...)	NetLock Kozjegyzoj (Class ...)	19/02/2019	NetLock I
NetLock Uzleti (Class B) Ta...	NetLock Uzleti (Class B) Ta...	20/02/2019	NetLock I
NO LIABILITY ACCEPTED, (...)	NO LIABILITY ACCEPTED, (...)	07/01/2004	VeriSign
PTT Post Root CA	PTT Post Root CA	26/06/2019	KeyMail F

Import... Export... Remove Advanced...

Certificate intended purposes <All>

Trusted Root CA
- always trusted

Trusted Root CA



Certificate purposes:

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

Certificate

General Details Certification Path

Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Self signed
- Can never be trusted

Issued to: William Buchanan
Issued by: William Buchanan
Valid from 22/02/2007 to 29/01/2107
You have a private key that corresponds to this certificate.

Issuer Statement OK



Certificates

Intended purpose: <All>

Intermediate Certification Authorities Trusted Root Certification Authorities Trusted Publ

Issued To	Issued By	Expiration...	Friendly
GTE CyberTrust Root	Root SGC Authority	23/02/2006	<N
Microsoft Internet Authority	GTE CyberTrust Global Root	23/02/2007	<N
Microsoft Internet Authority	GTE CyberTrust Global Root	19/04/2009	<N
Microsoft Secure Server Authority	Microsoft Internet Authority	23/02/2007	<N
Microsoft Secure Server Authority	Microsoft Internet Authority	19/04/2009	<N
Microsoft Windows Hardware C...	Microsoft Root Authority	31/12/2002	<N
Microsoft Windows Hardware C...	Microsoft Root Authority	31/12/2002	<N
MS SGC Authority	Root SGC Authority	01/01/2010	<N

Import... Export... Remove Advanced...

Certificate intended purposes

Signing, Windows Hardware Driver Verification

Intermediate CA
- Can be trusted for some things

Levels of trust



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?



Humor12.com

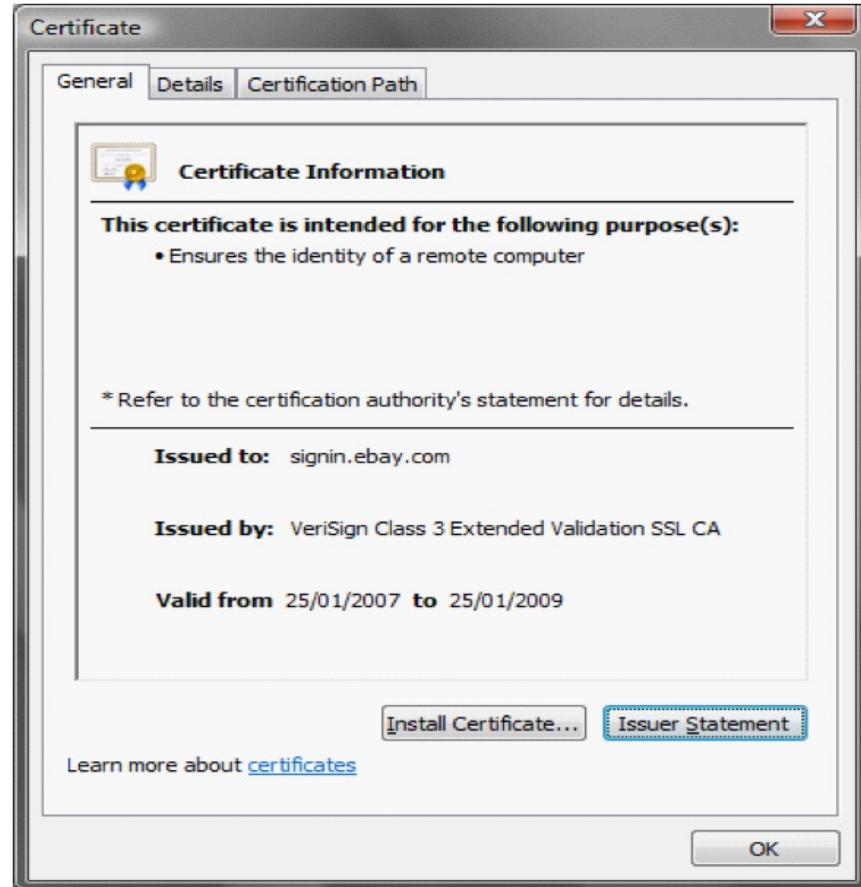


Bob



Eve

Real or fake?



Author: Prof Bill Buchanan

Real or fake?



Certificate

General Details Certification Path

Certification path

VeriSign
VeriSign Class 3 Extended Validation SSL CA
signin.ebay.com

https://www.verisign.com/repository/rpa.html - Windows Internet Explorer

File Edit View Favorites Tools Help

Products & Services Solutions Support About VeriSign

UNITED STATES

RESOURCES

PKI Disclosure
Licenses & Approvals
E-Sign
Publications

Home > Repository

VeriSign Relying Party Agreement

YOU MUST READ THIS RELYING PARTY AGREEMENT ("AGREEMENT") BEFORE VALIDATING A CERTIFICATE, USING VERISIGN'S ONLINE CERTIFICATE STATUS PROTOCOL ("OCSP") OR USING A VERISIGN OR VERISIGN AFFILIATE DATABASE OF CERTIFICATE REVOCATION LIST ("CRL"). IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT SUBMIT A QUERY AND DO NOT ACCESS, OR RELY ON ANY VERISIGN INFORMATION. IN CONSIDERATION OF YOUR AGREEMENT, YOU ARE ENTITLED TO USE VERISIGN INFORMATION AS SET FORTH HEREIN.

1. Term of Agreement. This Agreement becomes effective when you submit a query to validate a Certificate, or rely on any VeriSign Information in the manner set forth in the preamble and shall be applicable for as long as you use and/or rely on such VeriSign Information.

2. Definitions.
"Certificate" or "Digital Certificate" means a message that, at least, states a name or identifier for the Subscriber, contains the Subscriber's public key, identifies the Certificate's serial number, and contains a digital signature of the issuing CA.

Done

Internet | Protected Mode: Off

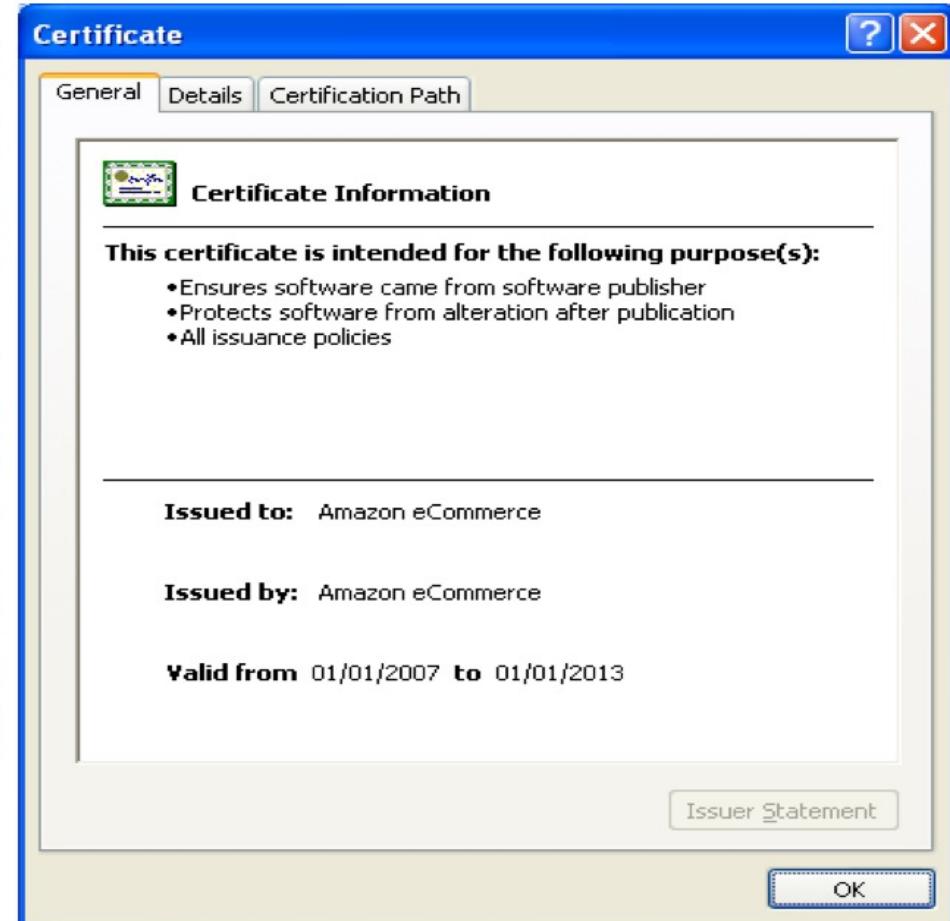
100%



Real!

Author: Prof Bill Buchanan

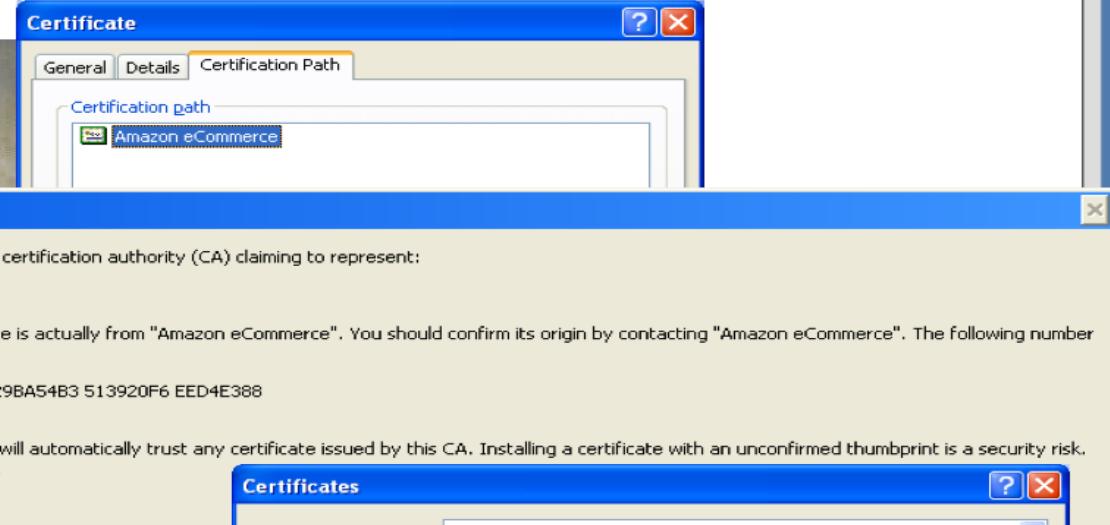
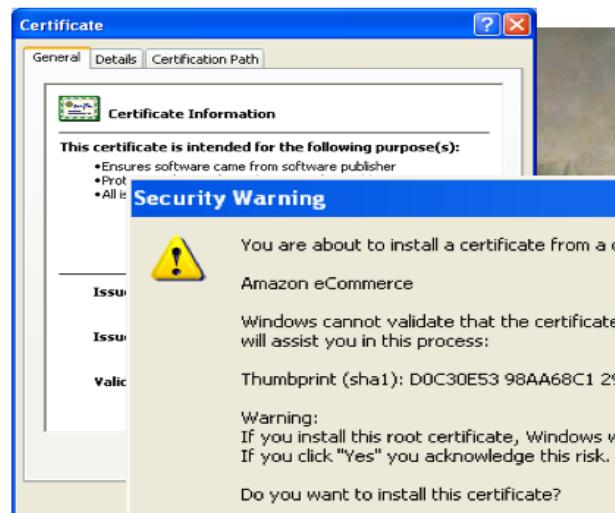
Real or fake?



Real or fake?

Author: Prof Bill Buchanan

Real or fake?



Certificates

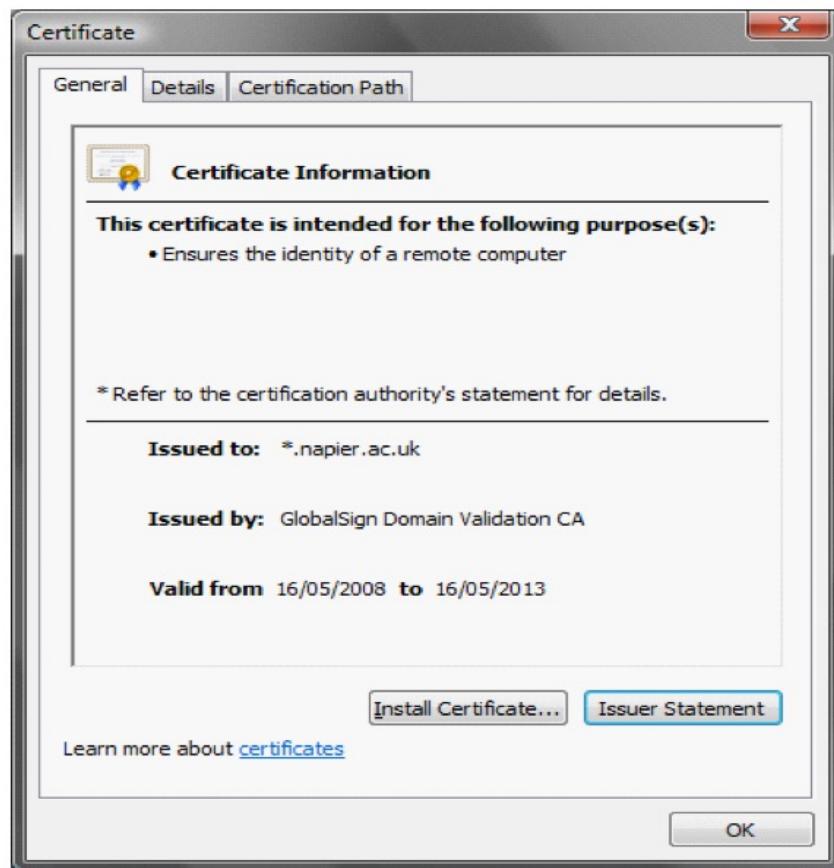
Intended purpose: <All>

Issued To	Issued By	Expiration Date	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



Real or fake?

Author: Prof Bill Buchanan

Real or fake?



Real



Certificate

General Details Certification Path

Certification path

- GlobalSign
- GlobalSign Domain Validation CA
- *.napier.ac.uk

GlobalSign (SSL Certificate) Legal Repository - Windows Internet Explorer

File Edit View Favorites Tools Help

GlobalSign (SSL Certificate) Legal Repository

Contact Us

HOME Products Solutions Partners About GlobalSign

You are here: United States Home > Repository > Legal Documents

About GlobalSign

- Company Profile
- Company History
- Management Team
- Press Center
- Repository
- Content Library
- International
- Contact Us

Repository of Legal Documents & Root Certificates

GlobalSign Root Certificates
All Root & Intermediate CA Certificates

GlobalSign Certification Practice Statement (CPS)
Current version - v6.1 - June 08
Previous version - v6.0 - December 07

GlobalSign Certification Practice Statement (CPS) for
Adobe Certified Document Services (CDS)

Waiting Internet | Protected Mode: Off 100%

Author: Prof Bill Buchanan

Chapter 6: Digital Certificates

Introduction

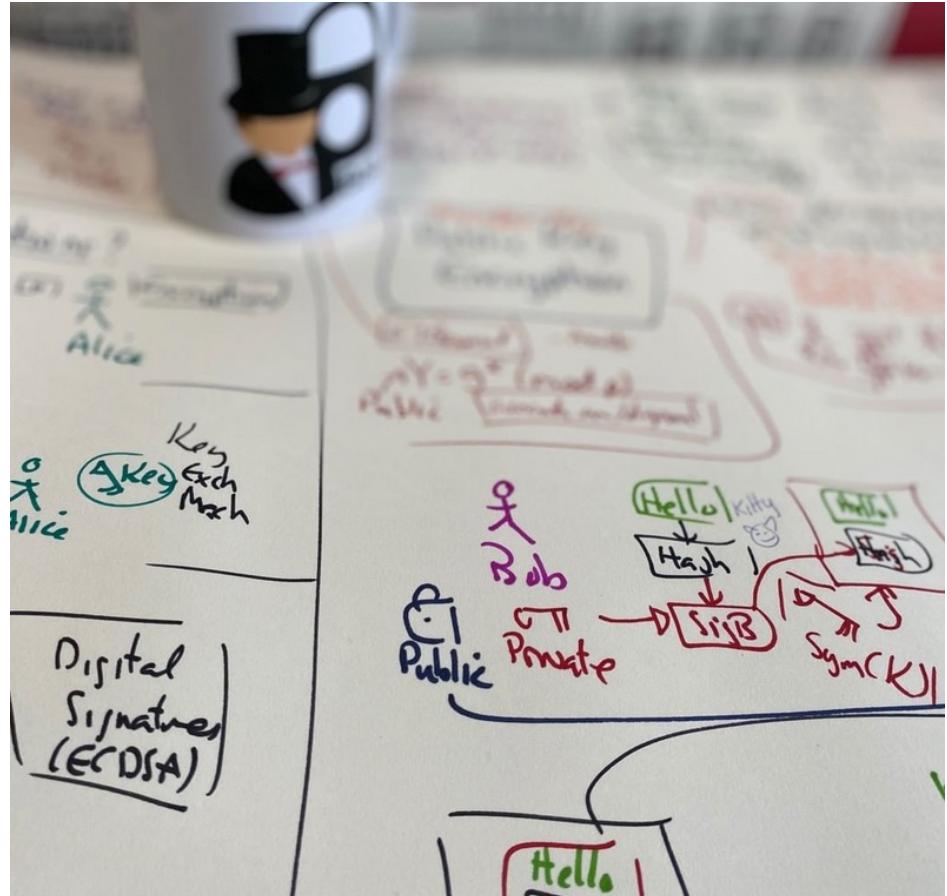
Authentication Methods

PKI

Digital Certificate Passing

Prof Bill Buchanan OBE

<https://asecuritysite.com/tunnelling>





Public key encryption ... secret ... identity ... trust



Eve



Trent



MegaCorp



Bob's Private Key



Bob's Public Key



Alice's Public Key



Alice's Private Key



Public key encryption ... secret ... identity ... trust



Eve



Trent



MegaCorp



Bob's Private Key



Alice's Public Key



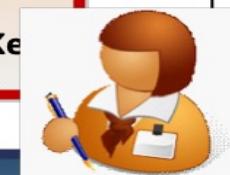
Alice's Public Key



Alice's Public Key

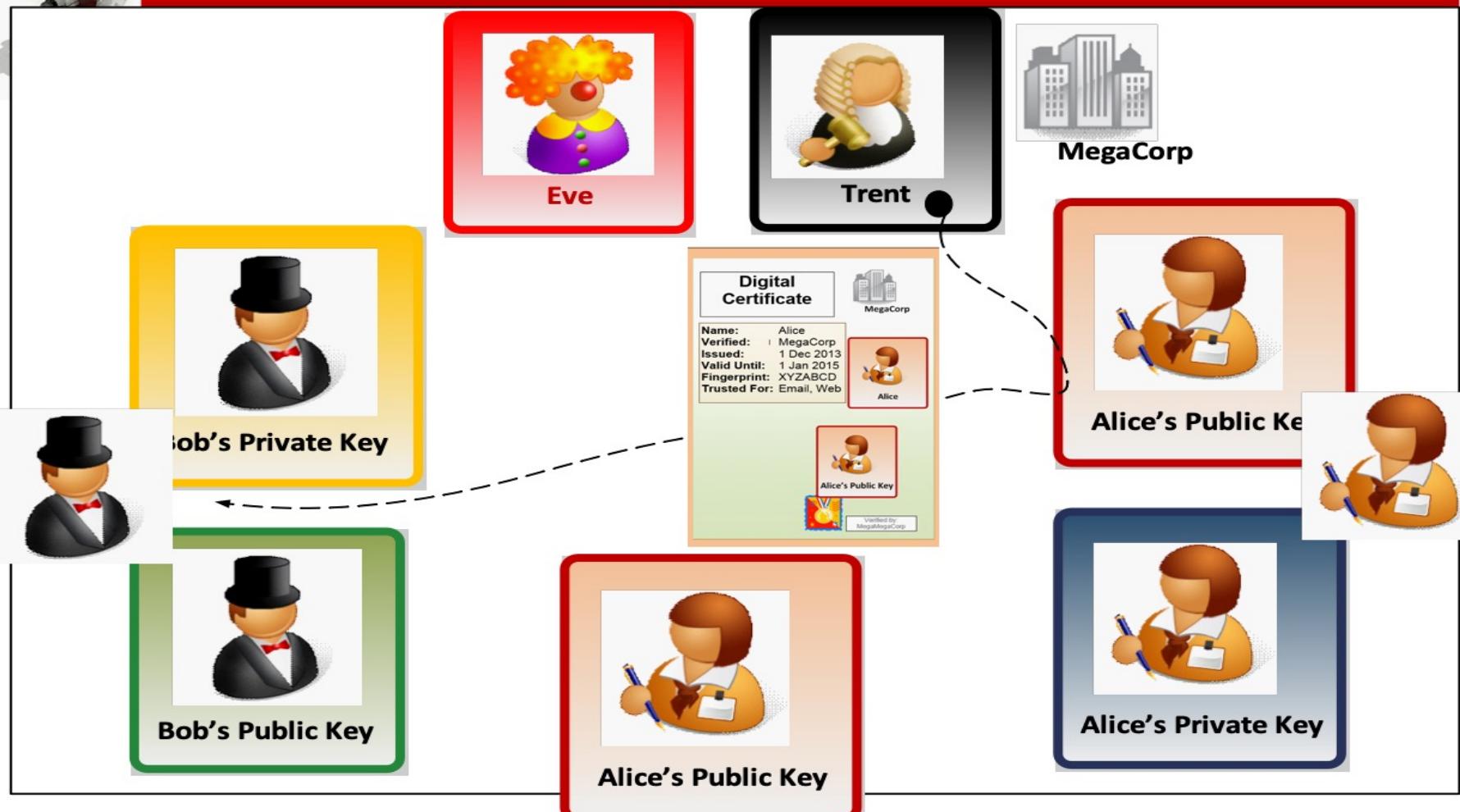


Alice's Private Key





Public key encryption ... secret ... identity ... trust





Public key encryption ... secret ... identity ... trust



Eve



Trent



MegaCorp



Alice's Public Key



Bob's Private Key



Hello Alice,
Wish you were
here!
- Bob

Bob.



Bob's Public Key



Alice's Public Key



Alice's Private Key



Public key encryption ... secret ... identity ... trust



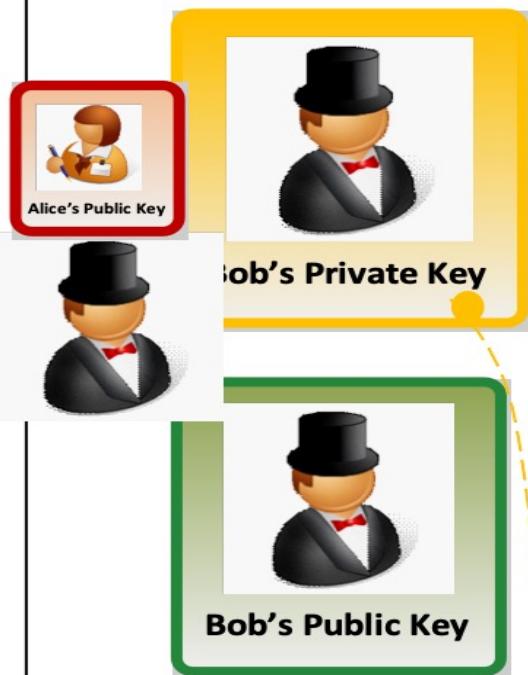
Eve



Trent



MegaCorp



Hello Alice,
Wish you were
here!
- Bob

Bob:



Bob's Private Key



Alice's Public Key



Alice's Private Key



Public key encryption ... secret ... identity ... trust



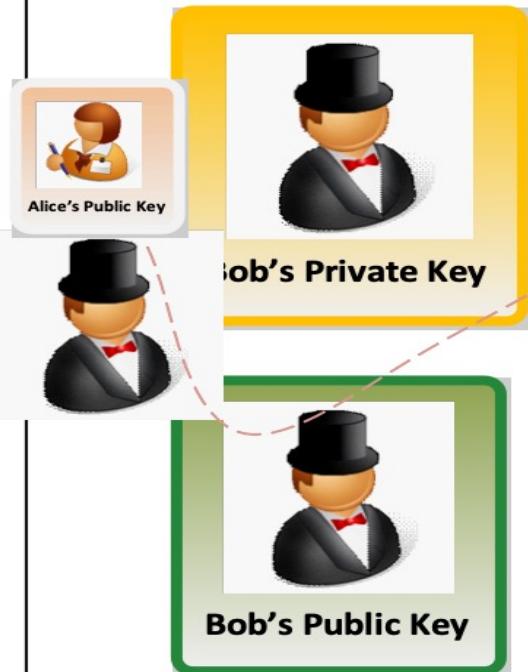
Eve



Trent



MegaCorp





Public key encryption ... secret ... identity ... trust



Eve



Trent



MegaCorp



Bob's Private Key



Bob's Public Key



Which key to open
the message?



Alice's Public Key



Alice's Private Key

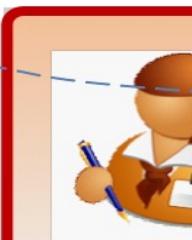


Public key encryption ... secret ... identity ... trust



Hello Alice,
Wish you were
here!
- Bob

Bob.



Which key to open
the message?



Public key encryption ... secret ... identity ... trust



Eve



Trent



MegaCorp



Bob's Private Key



Bob's Public Key

Hello Alice,
Wish you were
here!
- Bob

Bob:

Which key to we
open the signature
with?



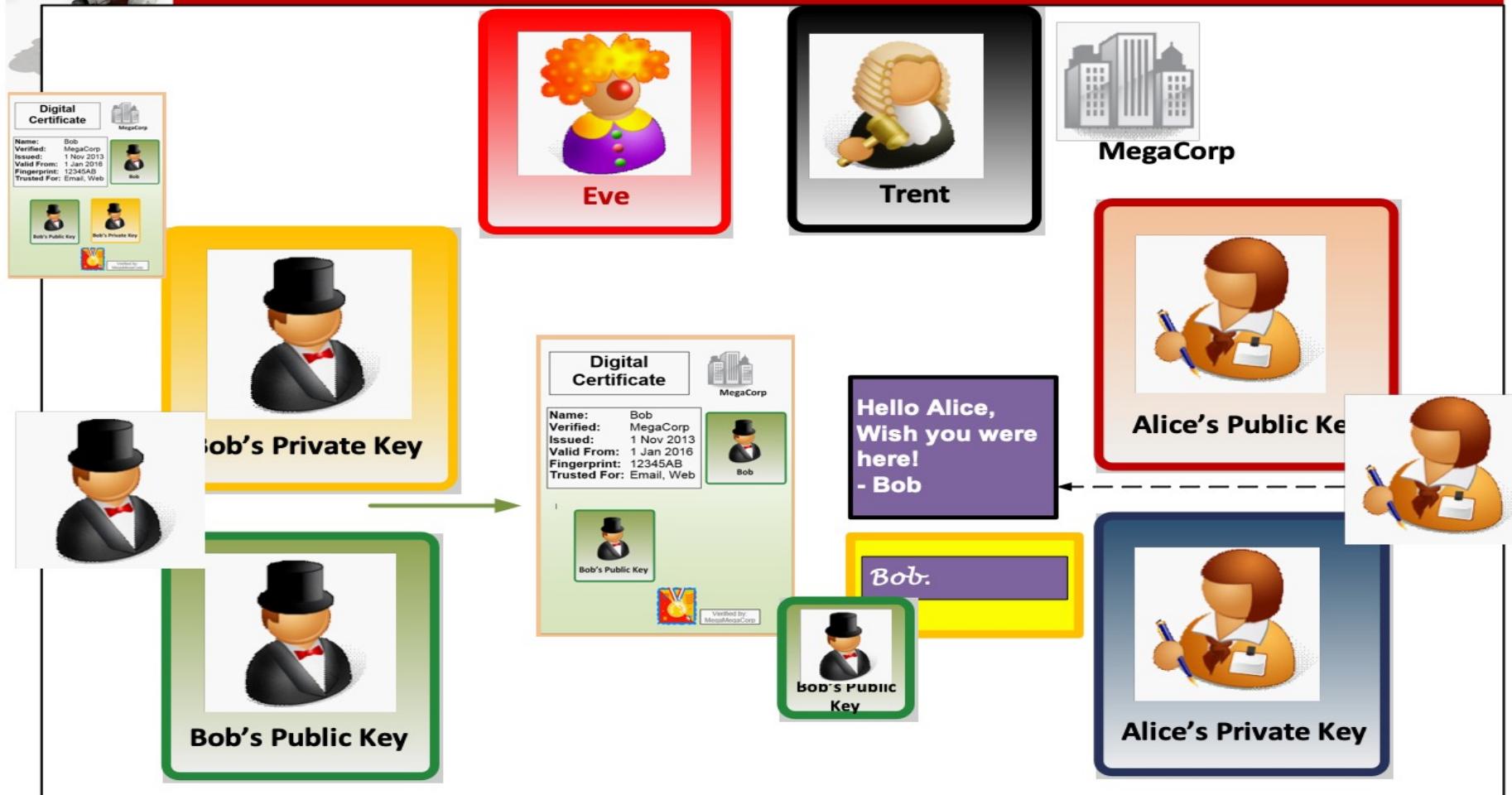
Alice's Public Key



Alice's Private Key

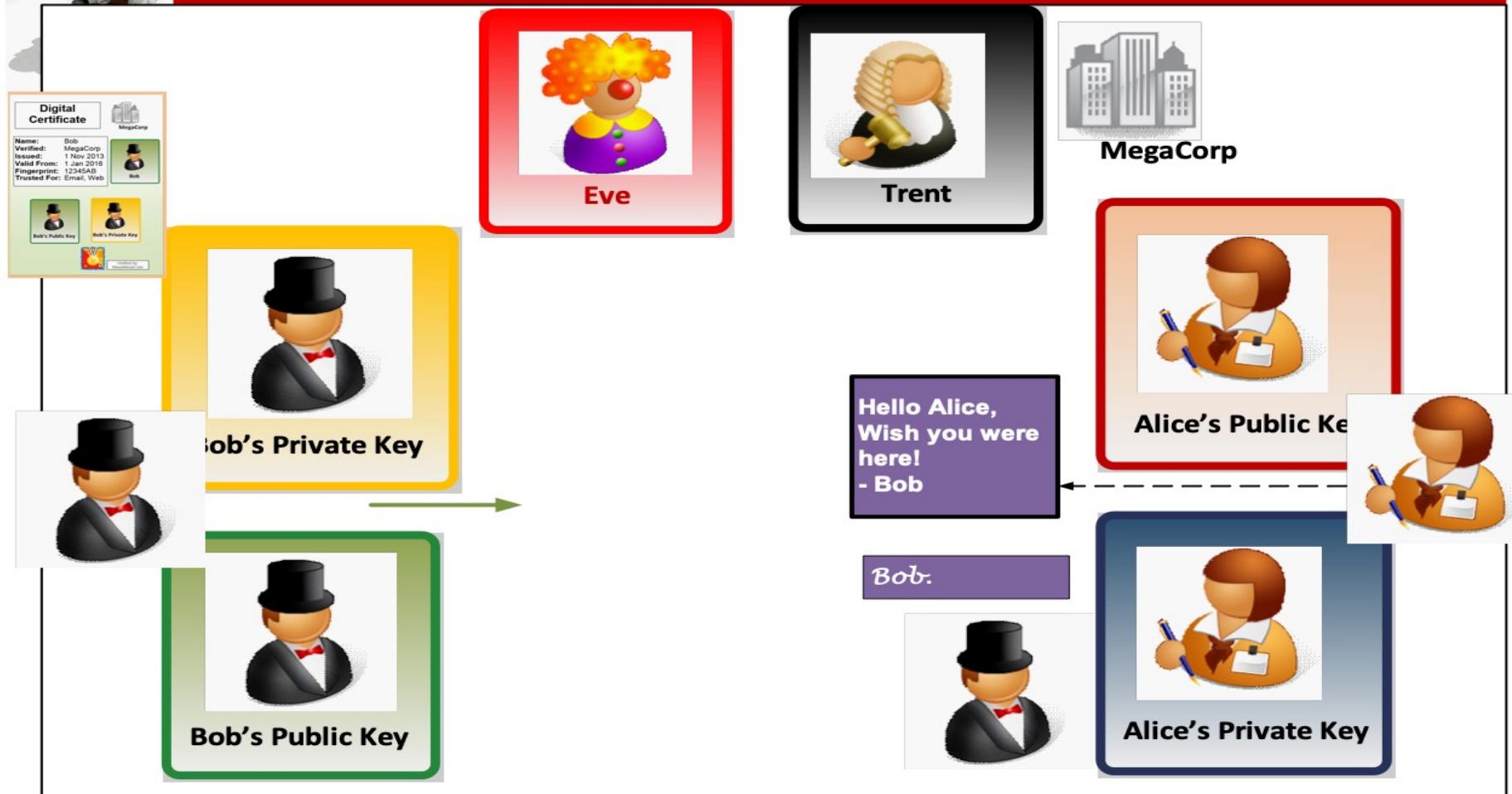


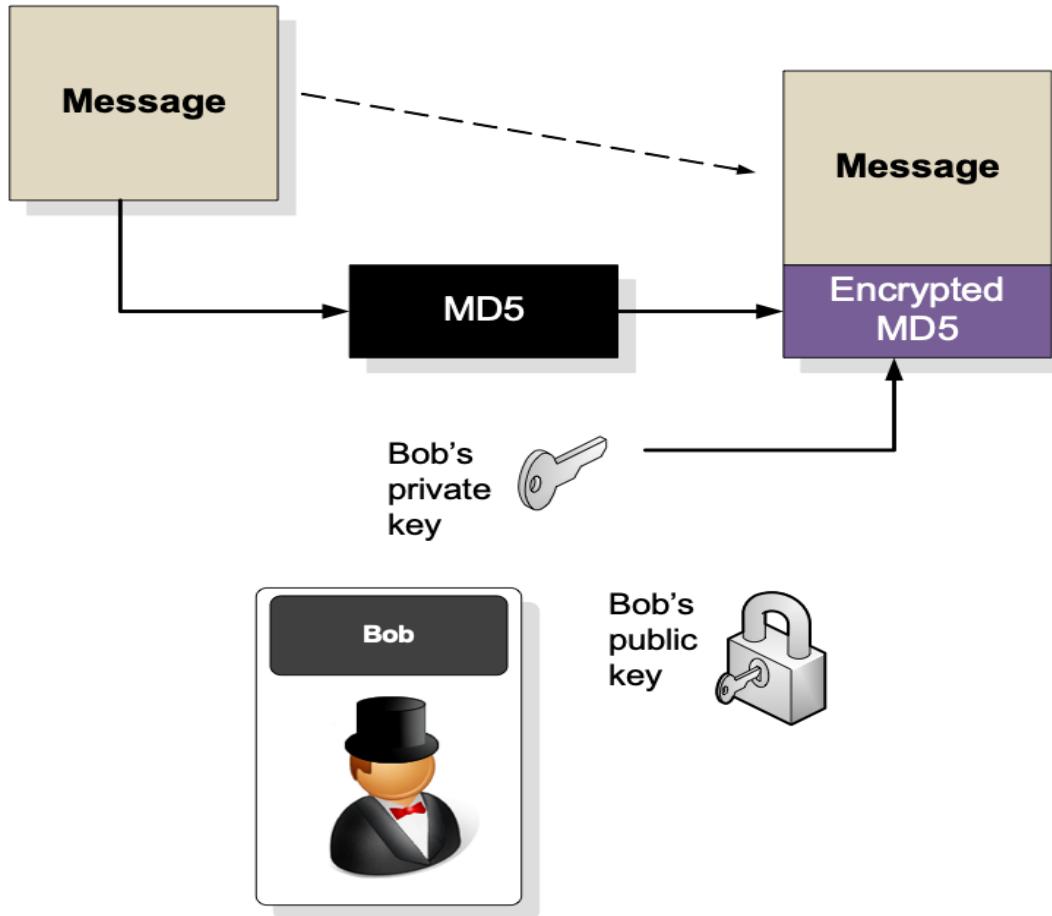
Public key encryption ... secret ... identity ... trust

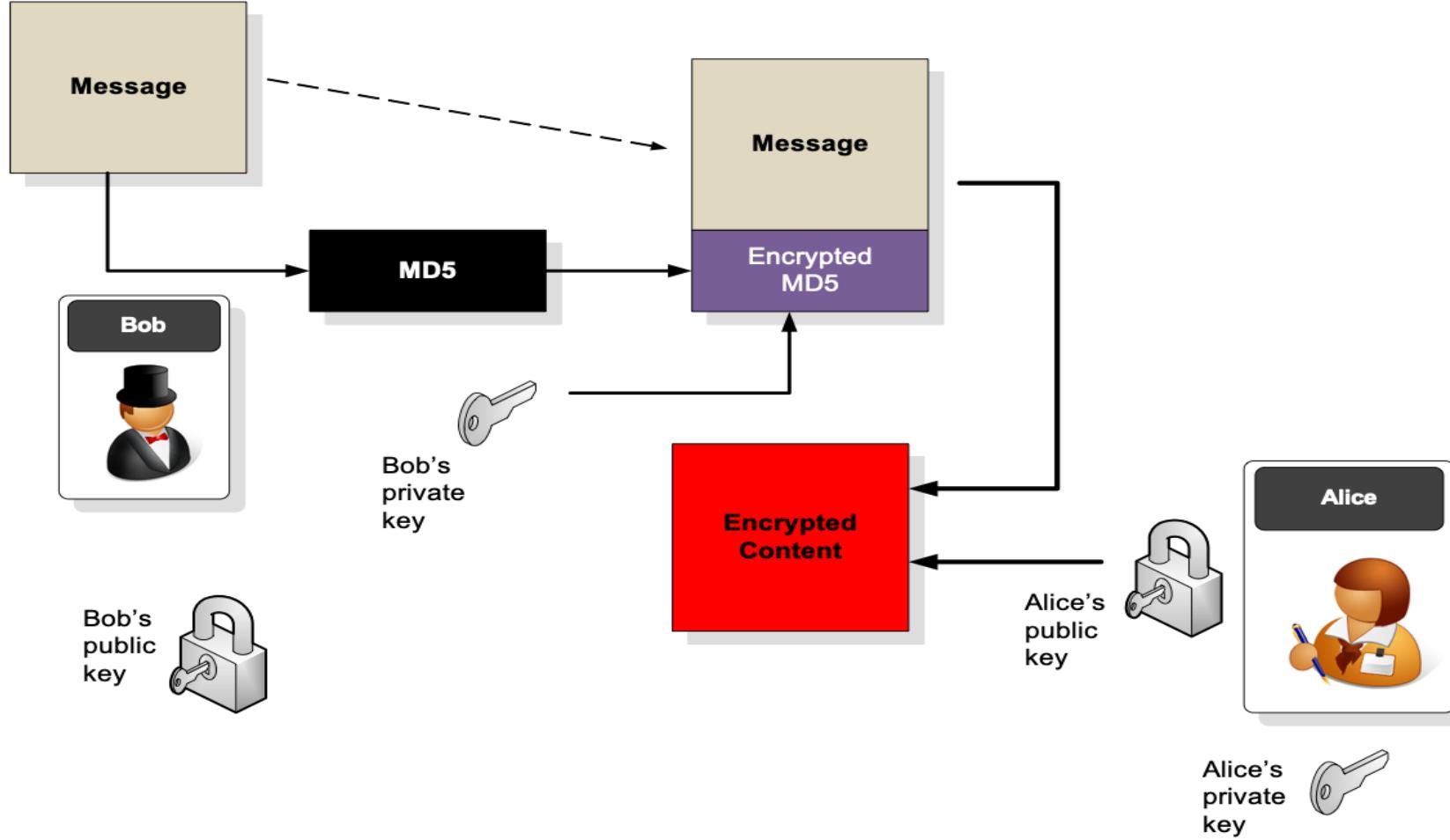




Public key encryption ... secret ... identity ... trust

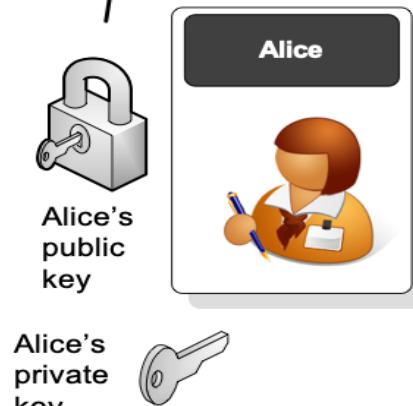
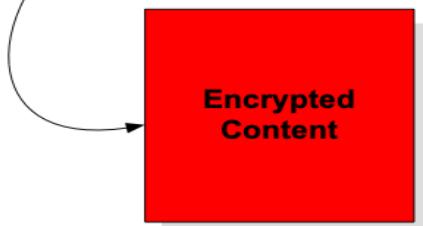
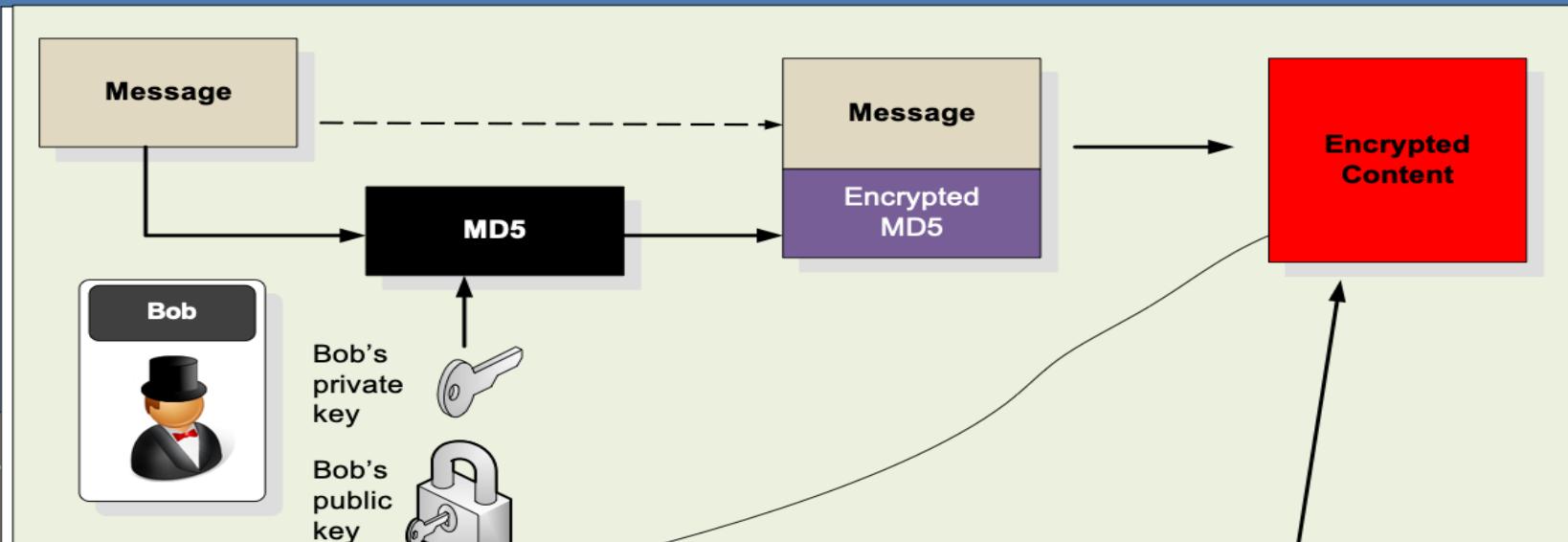






Authentication

The magic private key

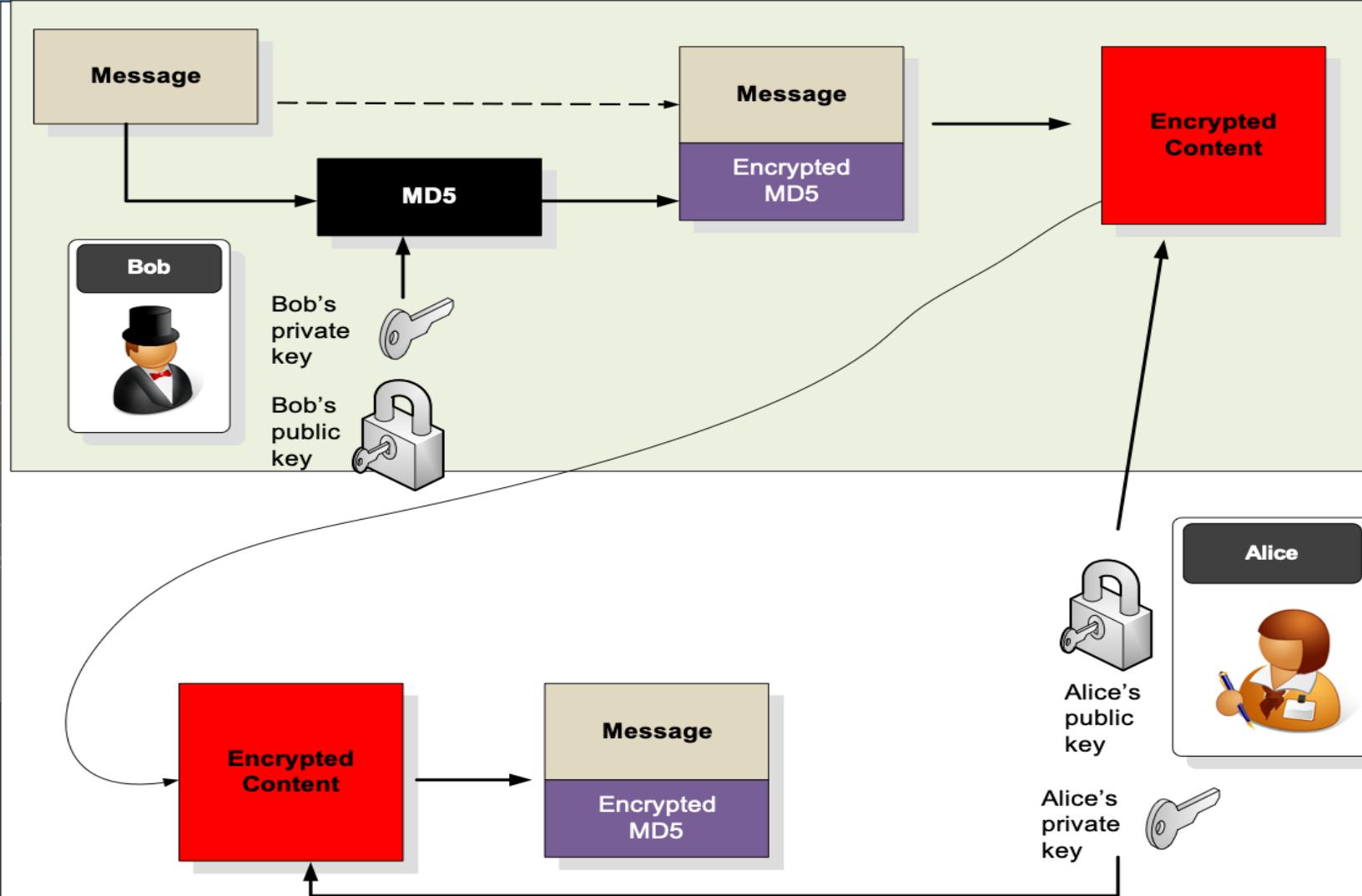


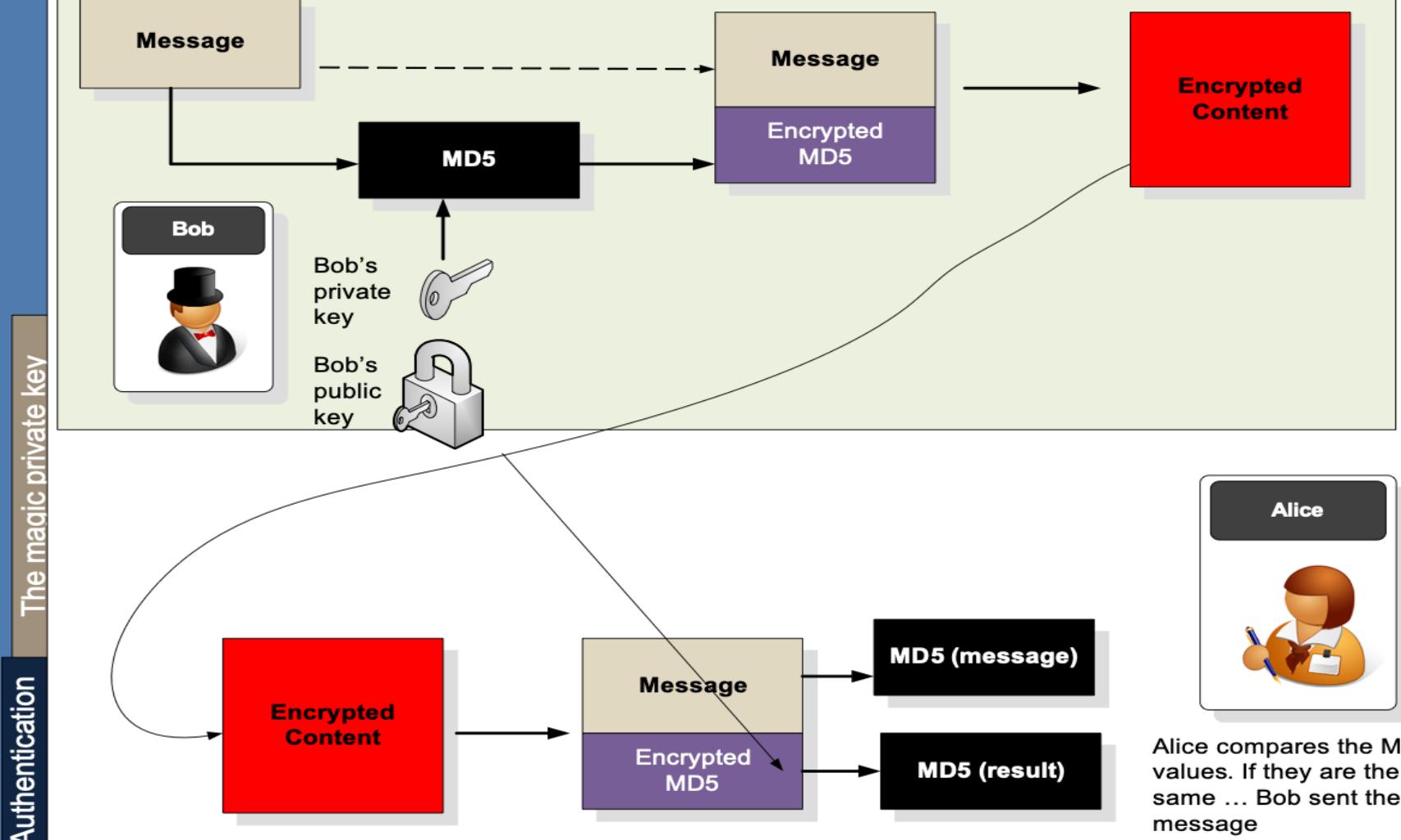
Bob encrypts the message/hash with Alice's public key

Author: Prof Bill Buchanan

Authentication

The magic private key





Authentication

The magic private key

Alice compares the MD5 values. If they are the same ... Bob sent the message

Author: Prof Bill Buchanan

Alice decrypts the message

Digital Certificates

Introduction

Authentication Methods

PKI

Digital Certificate Passing

Prof Bill Buchanan OBE

<https://asecuritysite.com/tunnelling>

