

Unit 4: Public Key

Basics

RSA

Elliptic Curve

ElGamal

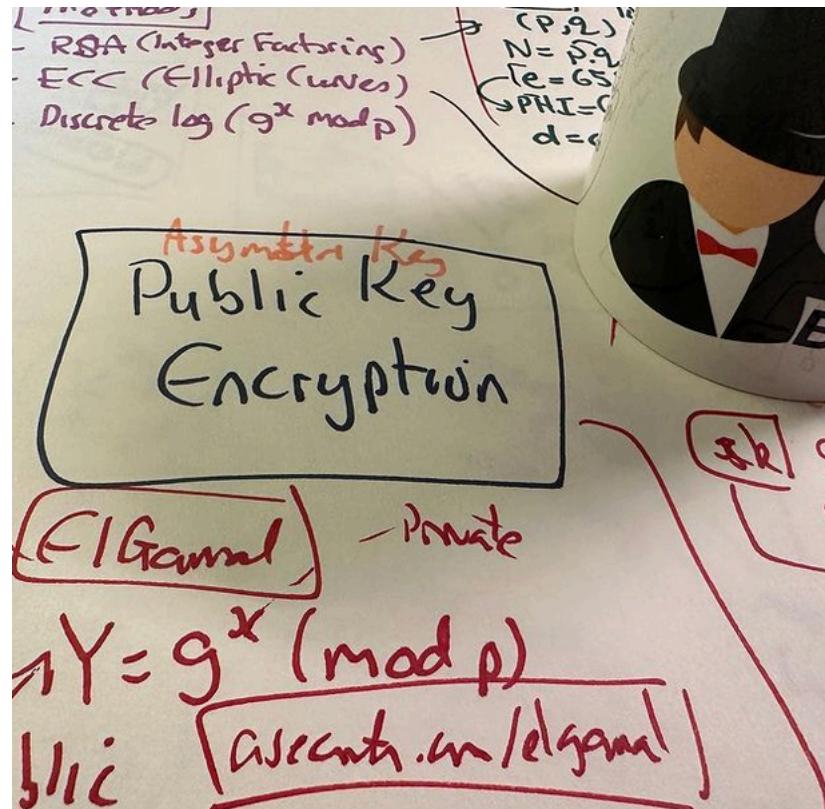
KMS

Prof Bill Buchanan OBE

<https://asecuritysite.com/rsa>

<https://asecuritysite.com/ecc>

<https://asecuritysite.com/elgaml>



Unit 4: Public Key

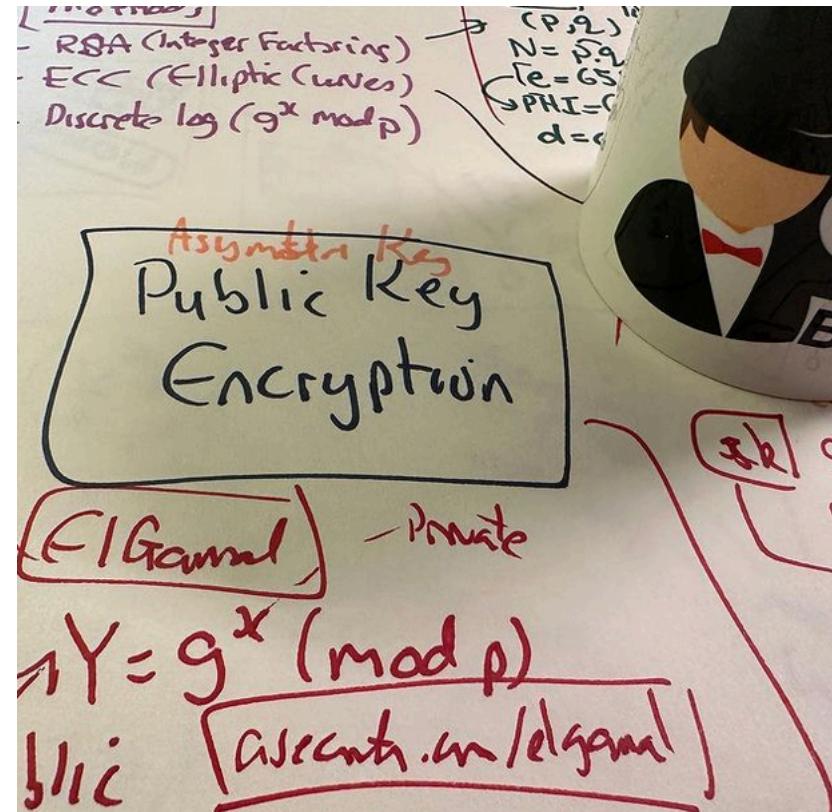
Basics

Prof Bill Buchanan OBE

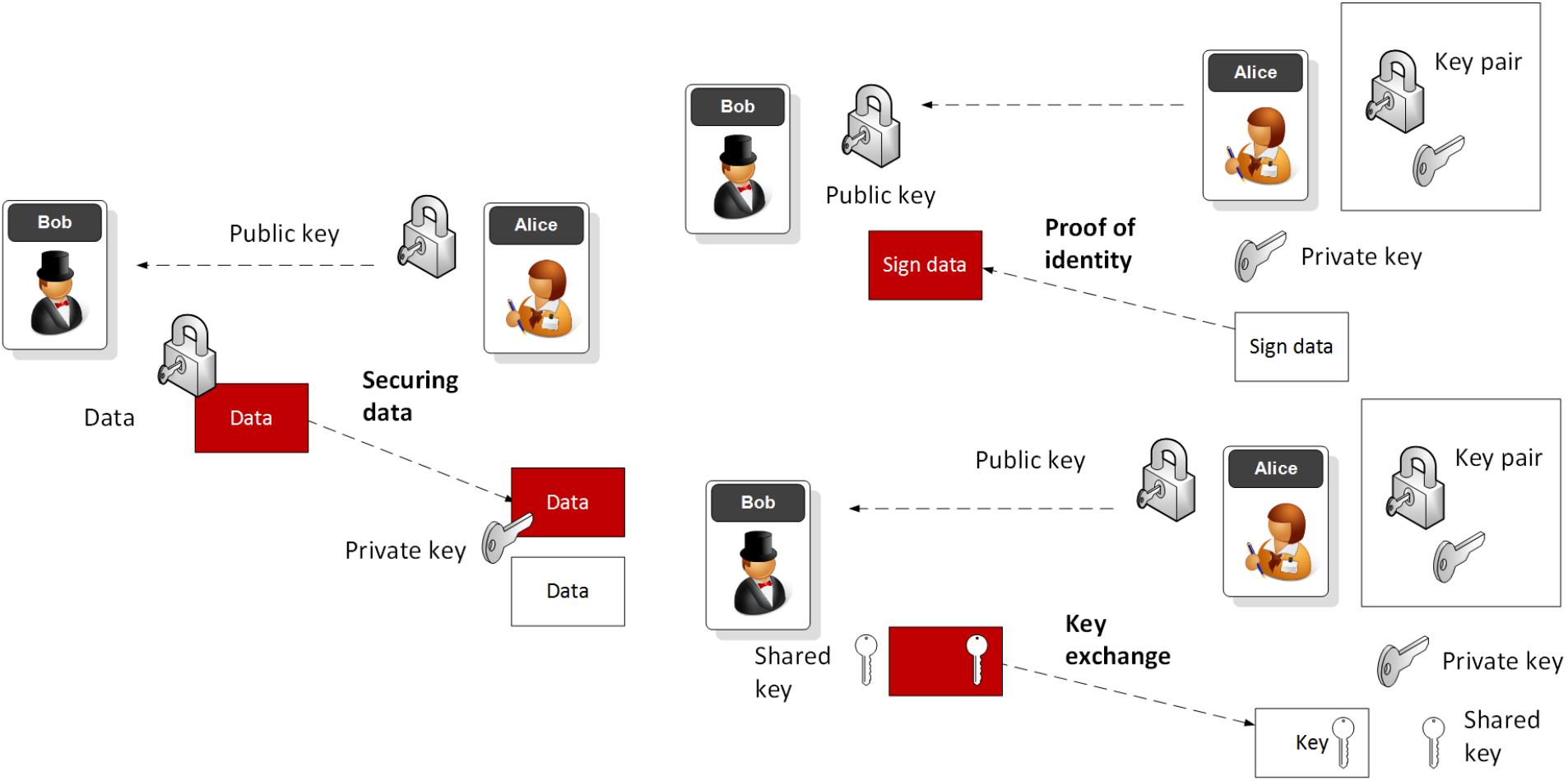
<https://asecuritysite.com/rsa>

<https://asecuritysite.com/ecc>

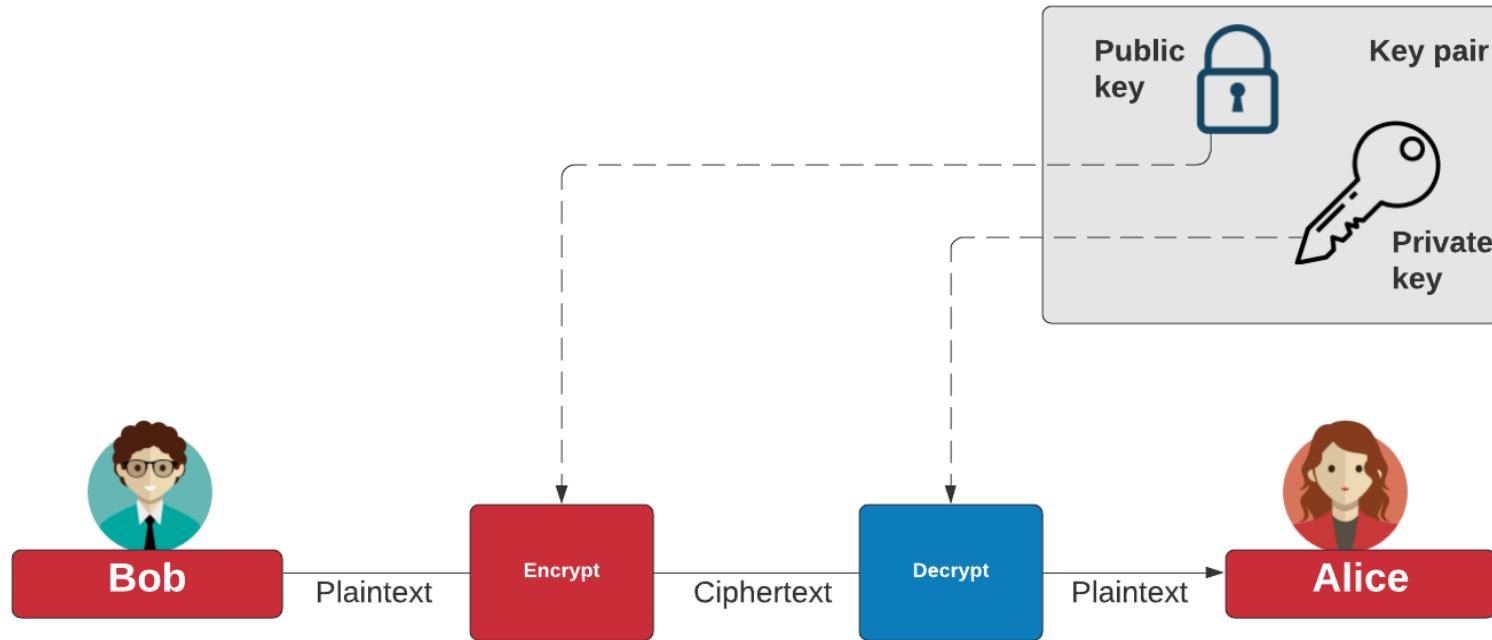
<https://asecuritysite.com/elgaml>



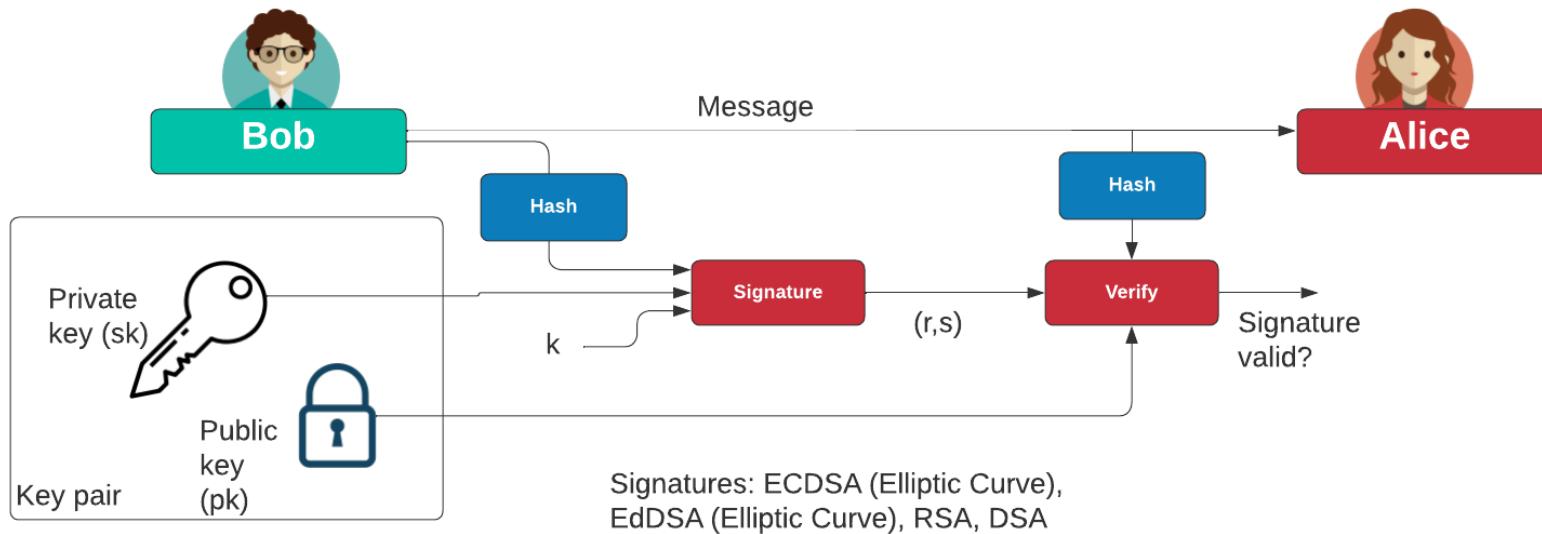
Public Key Methods



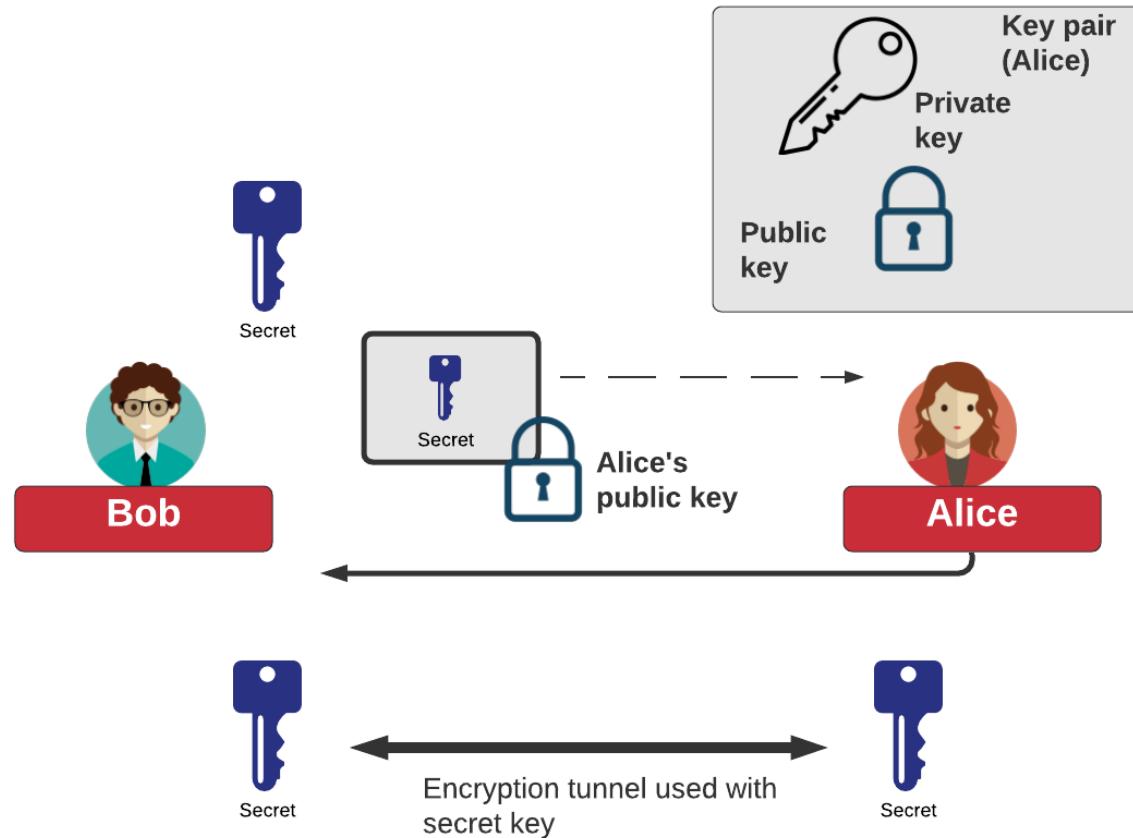
Public Key Encryption



Digital Signatures



Key Exchange with Public Key



Public Key Methods

- **Integer Factorization.** Using prime numbers.
Example: RSA. Digital Certs/SSL.
- **Discrete Logarithms.** $Y = G^x \text{ mod } P$. Example:
ElGamal.
- **Elliptic Curve Relationships.** Example: Elliptic
Curve. Smart Cards, IoT, Tor, Bitcoin.

Public Key Methods

- **Integer Factorization.** Using prime numbers.
Example: RSA. Digital Certs/SSL.
- **Discrete Logarithms.** $Y = G^x \text{ mod } P$. Example:
ElGamal.
- **Elliptic Curve Relationships.** Example: Elliptic
Curve. Smart Cards, IoT, Tor, Bitcoin.

Public Key Methods

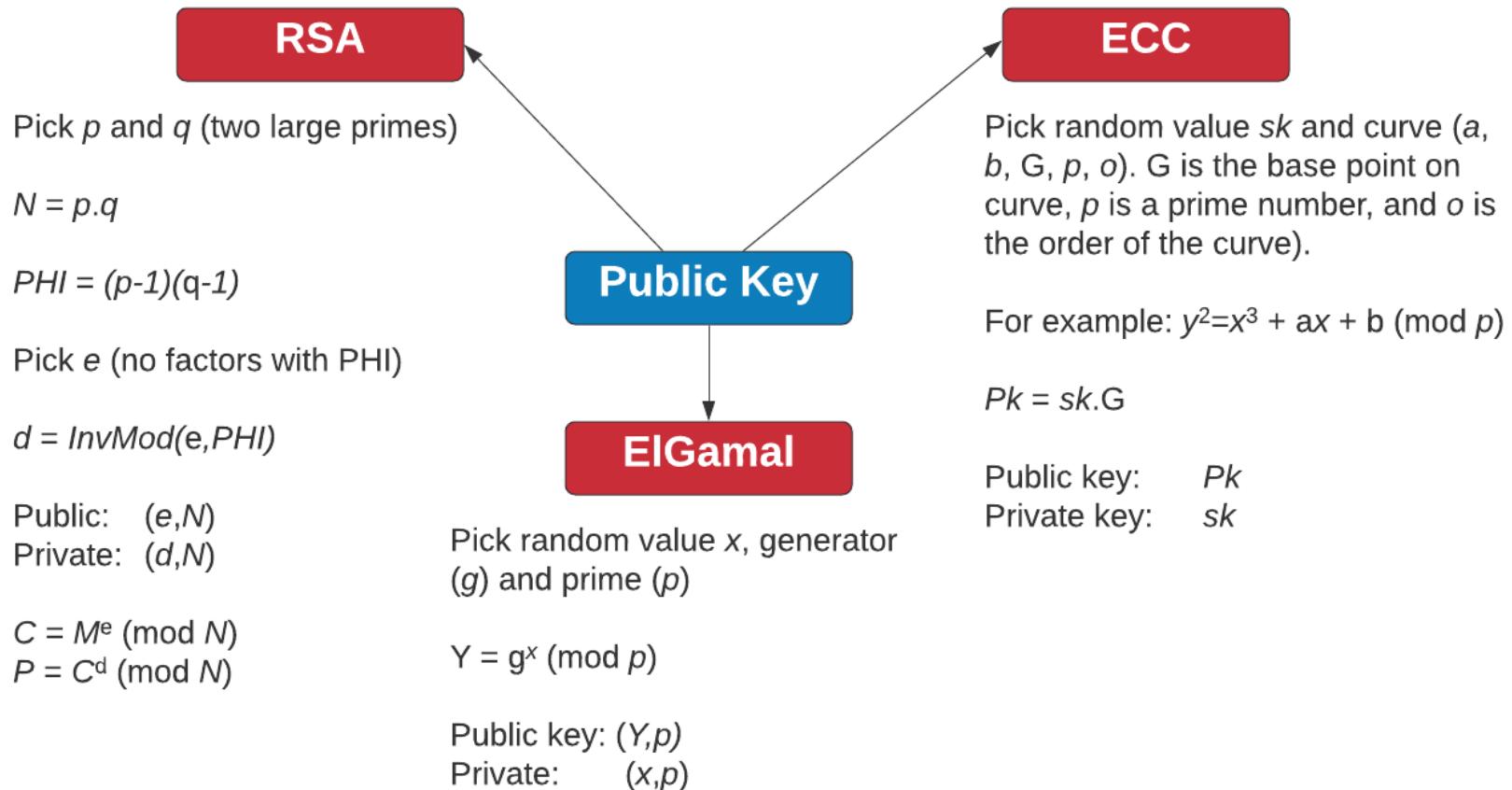
- Integer Factorization. Using prime numbers.

Example: RSA, Digital Certs/SSL

- Diffie-Hellman
- Elliptic Curve

| security level | volume of water to bring to a boil | 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 |

Public Key Encryption Methods



Unit 4: Public Key

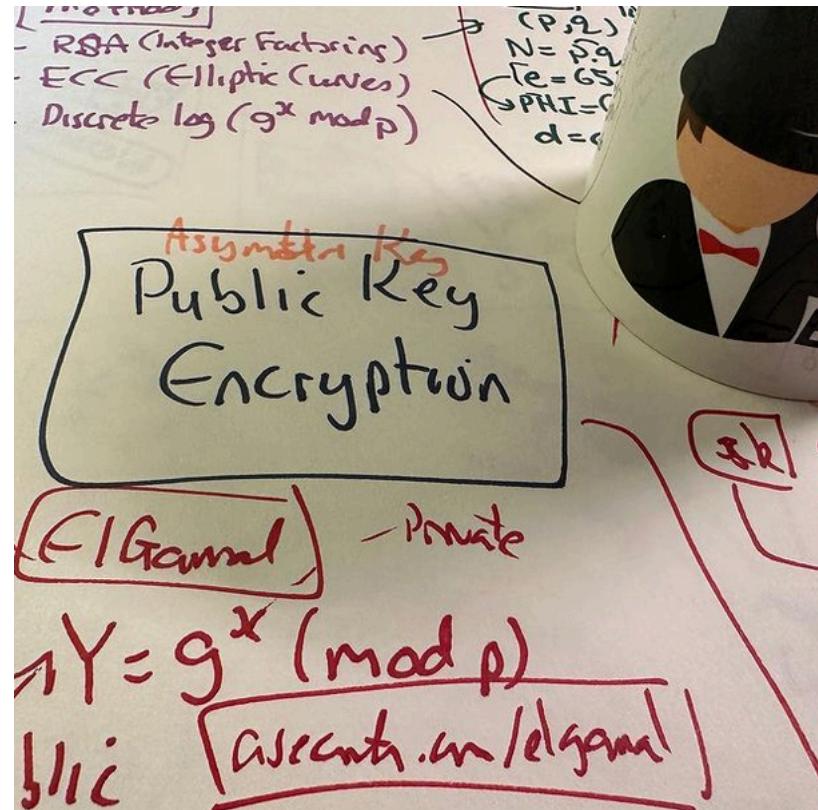
RSA

Prof Bill Buchanan OBE

<https://asecuritysite.com/rsa>

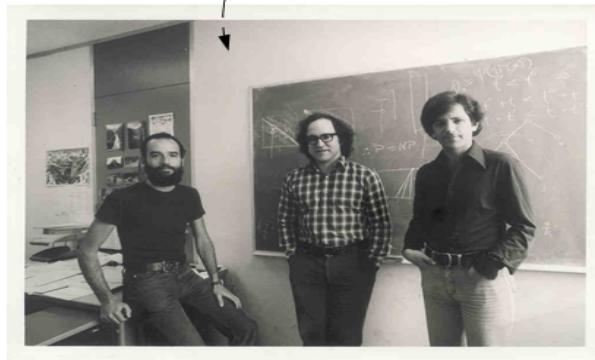
<https://asecuritysite.com/ecc>

<https://asecuritysite.com/elgamal>





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 $(p-1) \times (q-1)$. $\text{PHI}=20$
- 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]**

RSA

Calc

Example

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

eg $M=5$.



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

RSA

openssl genrsa -out private.pem 1024

$$C = M^e \bmod N$$

$$D = C^d \bmod N$$

cat private.pem

-----BEGIN RSA PRIVATE KEY-----

```
MIICXQIBAAKBgQC3qXK4kCxN3BNk87vJUMwIznU8pTjr10kma9+Jkj4zEy/fiZtY
xvdmn1rKNq/8fEUDCcRVC8hQBpevqxFiJ3dbA7ZM6VjUAmzt0fRfxSezgvkjswVS
F1/cgBM32AB4nx1dkCV/Wgedn3KFIFU+b8LH1ZLoyRMyLnwWmAkT/mBC/QIDAQAB
AoGAE8Yao+Rh44y+SdA0F6irTwdrd+wSBNJYSrKyjo1ARR97uAWIxDYnzNS7Yaoh
qH14sKsMiFuMZZFQI4m3hWnaX70FjhJvxKjP6+BdXKsnwLxpwecc7RS6n9ptA7q1E
aIFFVARyiWjG+q+8Bg8CTaHjGgtYPnfLzJM0Vef6gKg5vgECQQDZSKGxtbdbpXwXw
VAC78Syf00YmlWKL1HiZs0nyTOnZmhMSkE4+S38zhDTjITh0cuKTksTFeUku/sRij
4T4Y9iz5AkEA2GMpeeRT3IQntmzQgTc7Rgez73Y/UWFynuErg++9gzI758T03AoV
1Fs4NOUAqhZ5fdwizs6sa0bjYm+BC1mbJQJBAMQVts4QItVSSqK6vDrfh/xctd4v
KUh5oAWe4otfPBCCio7j1DLgxzp+K9TRxRvUWeMvNe4/uEMKgdiss6GAskCQQCF
MpVZMDriifgNppDgABqDsZcWfhCnduI1McQqFT+APn0ETy9Bg8nM1DAN+k061b4c
ctDJBhSj+EtiKFbwWsRhAkAnEPn+6m3djTwJMw82DxK1q2fcIjTR0ng8pyrF2iIR
P7oBP8I4hGix/FOrV8M8virK6iCsslEcZBo39FkEqc0N
```

-----END RSA PRIVATE KEY-----

openssl rsa -in private.pem -text

Private-Key: (1024 bit)

modulus:

```
00:b7:a9:72:b8:90:2c:67:dc:13:64:f3:bb:c9:50:
cc:08:ce:75:3c:a5:38:eb:d7:42:a6:6b:df:89:92:
3e:33:13:2f:df:89:9b:58:c6:f7:66:9f:5a:ca:36:
af:fc:7c:45:03:09:c4:55:0b:c8:50:06:97:af:ab:
11:62:27:77:5b:03:b6:4c:e9:58:d4:02:6c:ed:39:
f4:5f:c5:27:b3:82:f9:23:b3:05:52:17:5f:dc:80:
13:37:d8:00:78:9f:1d:5d:90:25:7f:5a:07:9d:9f:
72:85:20:55:3e:6f:c2:c7:d5:92:e8:c9:13:32:2e:
7c:16:98:09:13:fe:60:42:fd
```

publicExponent: 65537 (0x10001)

privateExponent:

```
13:c6:1a:a3:e4:61:e3:8c:be:49:d0:34:17:a8:ab:
4f:07:6b:77:ec:12:04:d2:58:4a:b2:b2:8e:8d:40:
45:1f:7b:b8:05:88:c4:36:27:cc:d4:bb:61:aa:21:
a8:7d:78:b0:ab:0c:88:5b:8c:65:91:50:23:89:b7:
85:69:da:5f:b3:85:8e:12:6f:c4:a8:cf:eb:e0:5d:
5c:ab:27:c1:6c:69:c1:e7:3b:45:2e:a7:f6:9b:40:
ee:a9:44:68:81:5f:54:04:72:89:68:c6:fa:af:bc:
06:0f:02:4d:a1:e3:1a:0b:58:3e:77:cb:cc:93:34:
55:e7:fa:80:a8:39:be:01
```

prime1:

```
00:d9:48:a1:b1:b5:d6:e9:5f:05:f0:54:00:bb:f1:
2c:9f:38:e6:26:58:a2:f5:1e:26:6c:d2:7c:93:3a:
76:66:84:c4:a4:13:8f:92:df:cc:e1:0d:38:c8:4e:
1d:1c:b8:a4:e4:b1:31:5e:52:4b:bf:b1:18:a3:e1:
3e:18:f6:2c:f9
```

prime2:

```
00:d8:63:29:e4:53:dc:84:27:b6:6c:d0:81:37:
3b:46:07:b3:ef:76:3f:51:61:72:9e:e1:2b:83:ef:
bd:83:32:3b:e7:c4:ce:dc:0a:15:94:5b:38:34:e5:
00:aa:16:79:7d:dc:22:ce:ce:ac:6b:46:e3:62:6f:
81:0b:59:9b:25
```

RSA

openssl genrsa -out private.pem 1024

$$C = M^e \text{ mod } N$$

openssl rsa -in private.pem -text

Private-Key: (1024 bit)

modulus:

00:b7:a9:72:b8:90:2c:67:dc:13:64:f3:bb:c9:50:
cc:08:ce:75:3c:a5:38:eb:d7:42:a6:6b:df:89:92:
3e:33:13:2f:df:89:9b:58:c6:f7:66:9f:5a:ca:36:

-----BEGIN PGP MESSAGE-----

Version: GnuPG v1

hQEMA8anVEMIIe/JAQf/cUmIvTbhQQhr70vPY817xRld7NNUrFIqWoz0S7BfpXDi
kvNbw/tIR5yS8gIbm25QFl5kUCukZh3zBq1vZ4pSq35e0ReH4RZQRmDe6Wtn244D
0PJ6W0e4c4y+87shZdJhAwgpLZl5gqZ3YnySoX7kH2CbqDYJUr+4giq/TWGYGb+F
7ztIBwnTZEyijFpWrYhtBVz2DM1HfMDgH3wNWLH0LbE+s7XwqBP/3FHp4Holaqrt
BMU9+MzlM5rqq/AnGXW80/VR8eELJs500qRZmHcI8D06p8sgNBTeuchadSkKZLgP
i0+l/m2/9n0Fg++JSCRpul+JVQU+IngP9pgG13NvktJUAa2/McEaBRYeIr1X4v6g
Syr5jcHBqCR3zyMV06rg2+r0VK3Z0a9DV4QG0yKewJhPEwPDfLo4SoWZa5n9zwNP
JWm6iiSYz2wLiYd5Pg6Zr/DpGbDF
=50st

-----END PGP MESSAGE-----

-----BEGIN RSA PRIVATE KEY-----

MpVZMDriifgNppDgABqDszcWfhCnduI1McQqFT+APn0ETy9Bg8nM1DAN+k061b4c
ctDJBhSj+EtiKFbwWsRhAkAnEPn+6m3djTwJMw82DxK1q2fcIjTR0ng8pyrF2iIR
P7oBP8I4hGix/F0rV8M8virK6iCss1EcZBo39FkEqc0N

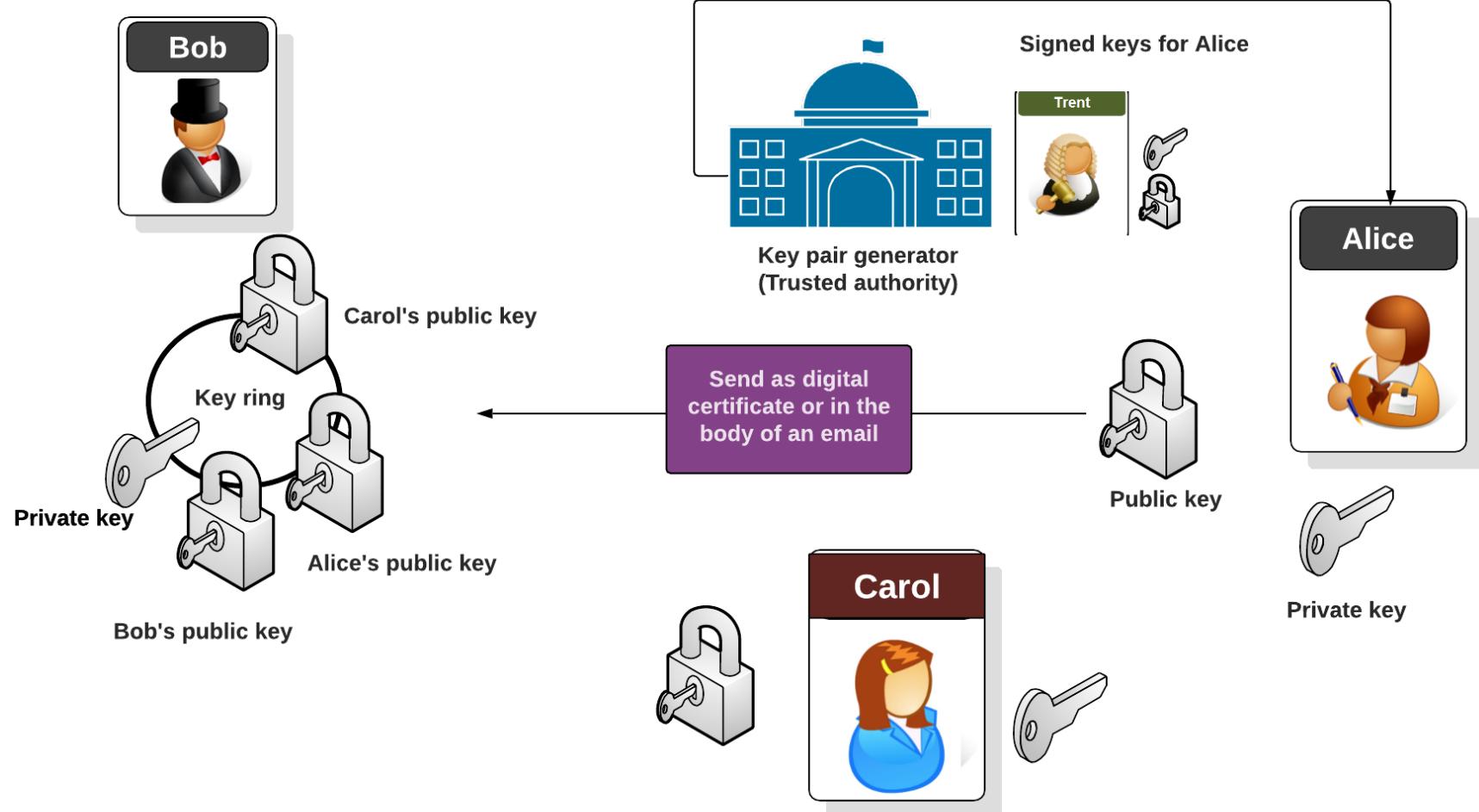
-----END RSA PRIVATE KEY-----

3e:18:f6:2c:f9

prime2:

00:d8:63:29:79:e4:53:dc:84:27:b6:6c:d0:81:37:
3b:46:07:b3:ef:76:3f:51:61:72:9e:e1:2b:83:ef:
bd:83:32:3b:e7:c4:ce:dc:0a:15:94:5b:38:34:e5:
00:aa:16:79:7d:dc:22:ce:ce:ac:6b:46:e3:62:6f:
81:0b:59:9b:25

Key ring



Unit 4: Public Key

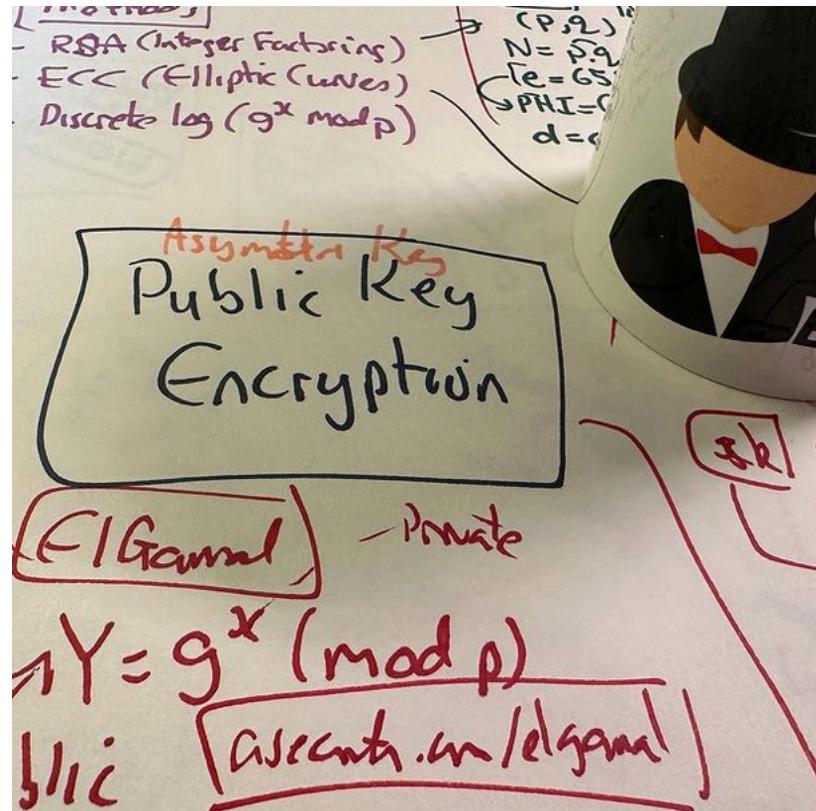
Elliptic Curve

Prof Bill Buchanan OBE

<https://asecuritysite.com/rsa>

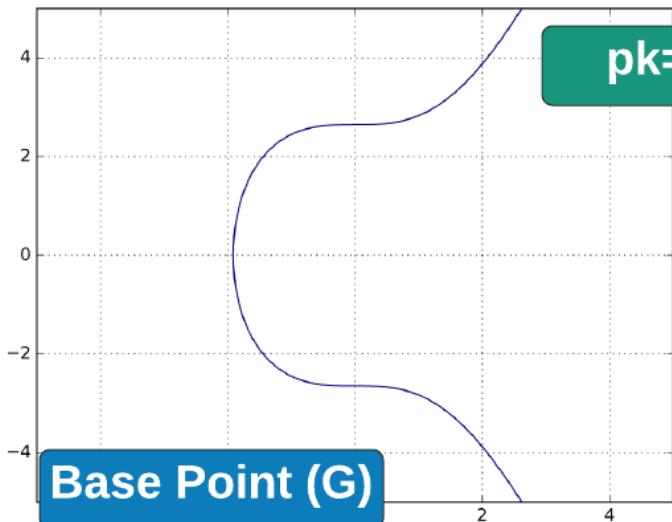
<https://asecuritysite.com/ecc>

<https://asecuritysite.com/elgaml>



Elliptic Curve Methods

$$y^2=x^3+ax+b \pmod{p}$$



$pk=sk.G$



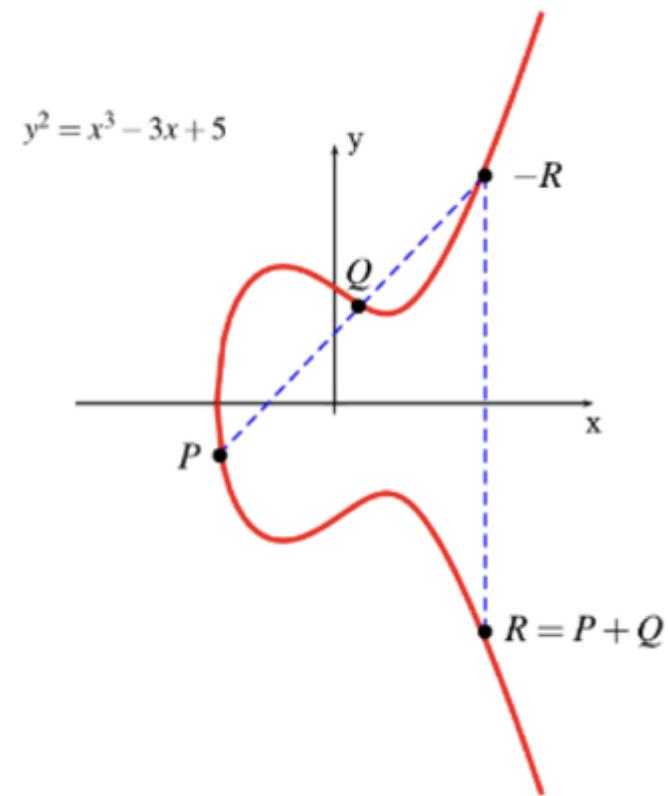
Bob

sk (private key)

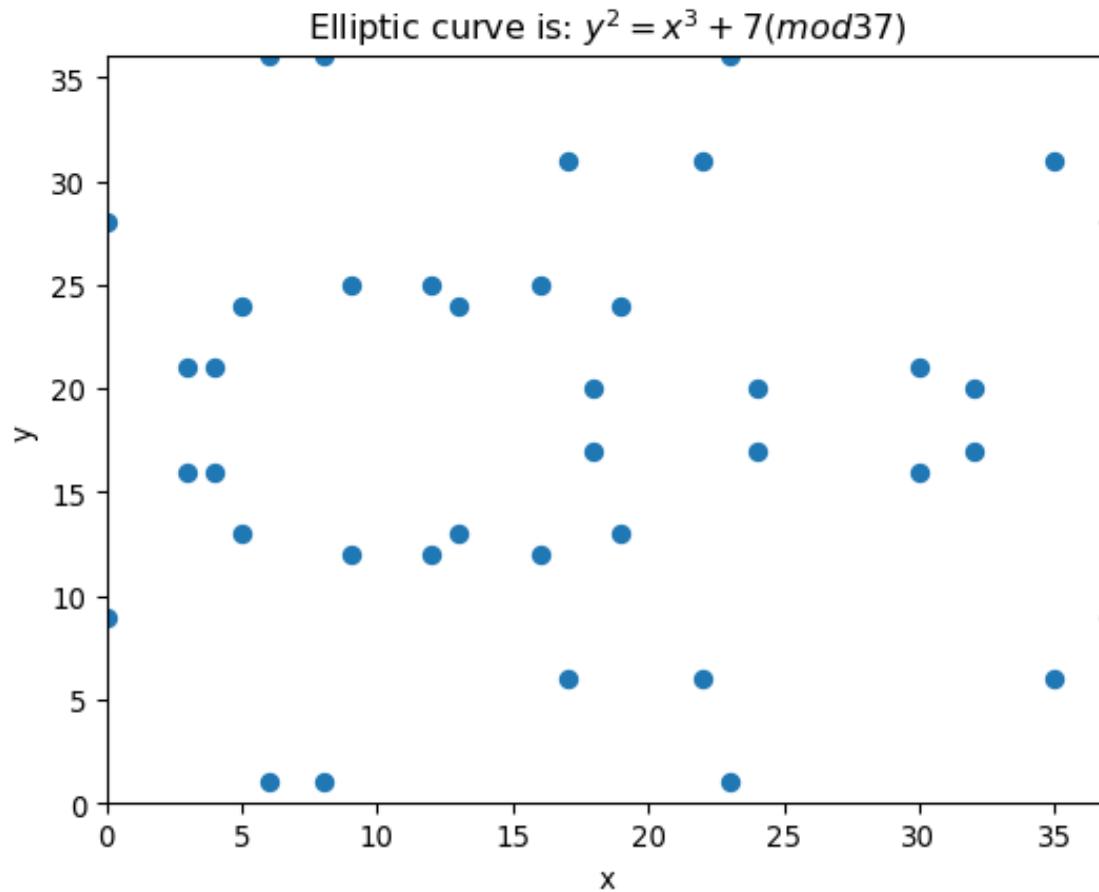
Basic operations:
• Point add.
• Point double.

Elliptic Curve (EC)

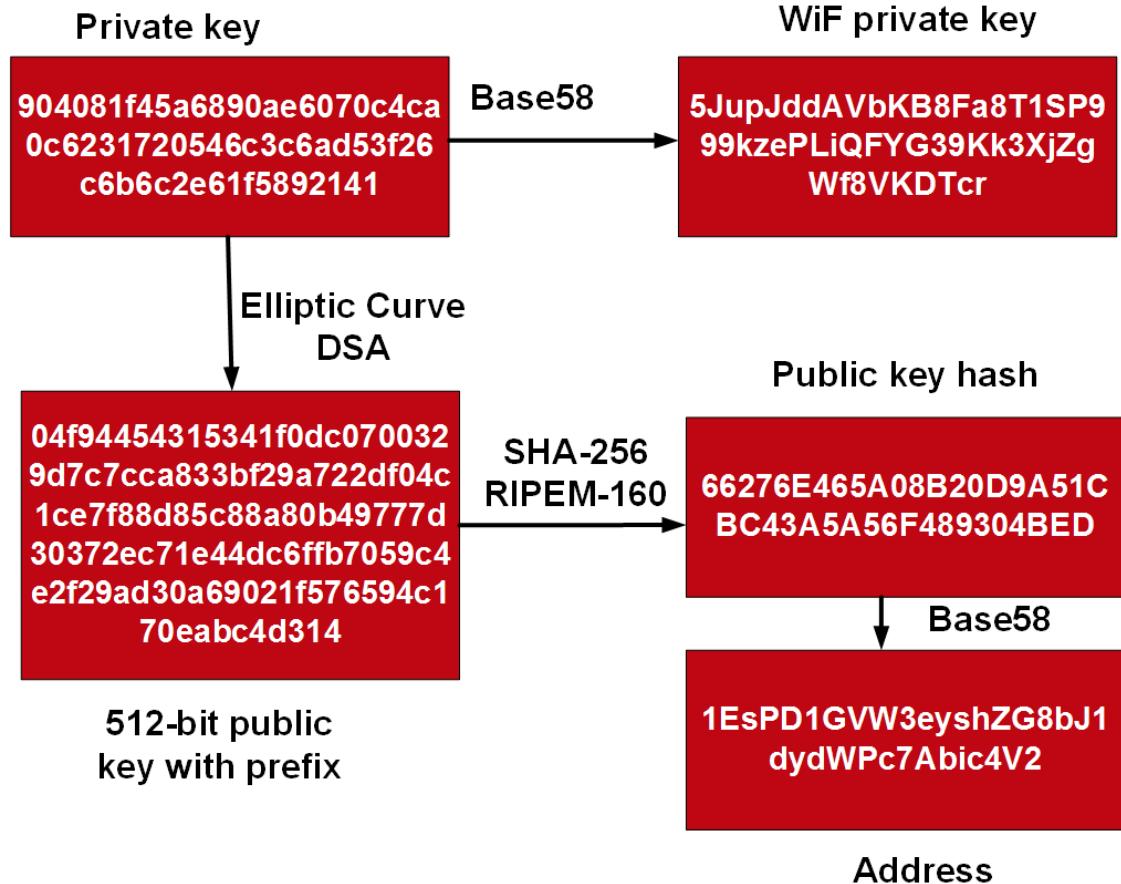
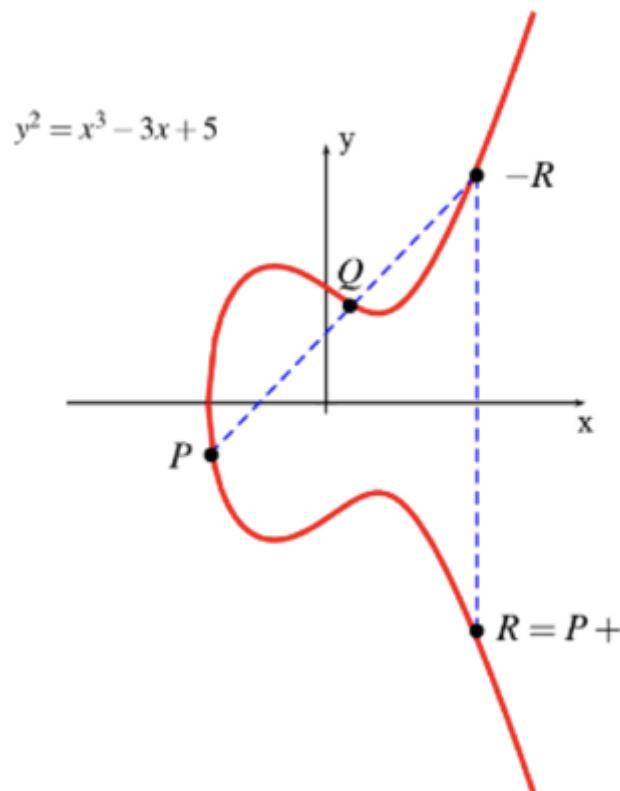
- Pick a point on the elliptic curve (G).
- Generate a random number (n) – this will be the private key.
- Public key is $P = n \times G \pmod{p}$, where p is a prime number (eg 256-bit prime for Curve 25519).
- n is a scalar value which multiples with G to give P (public key)
- Bitcoin uses secp256k1 and Tor uses Curve 25519 [[here](#)].



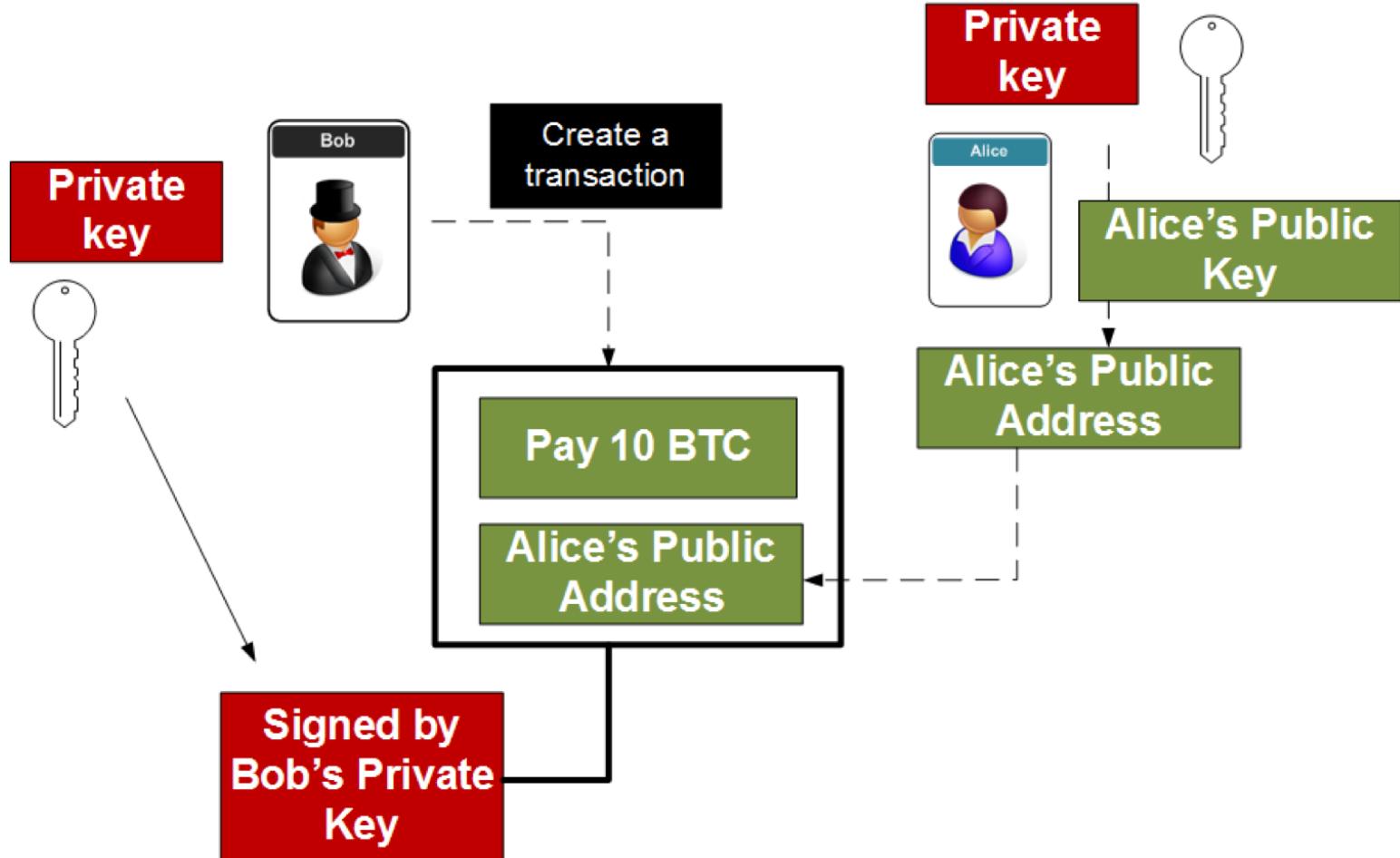
Elliptic Curve (EC) - (mod p)



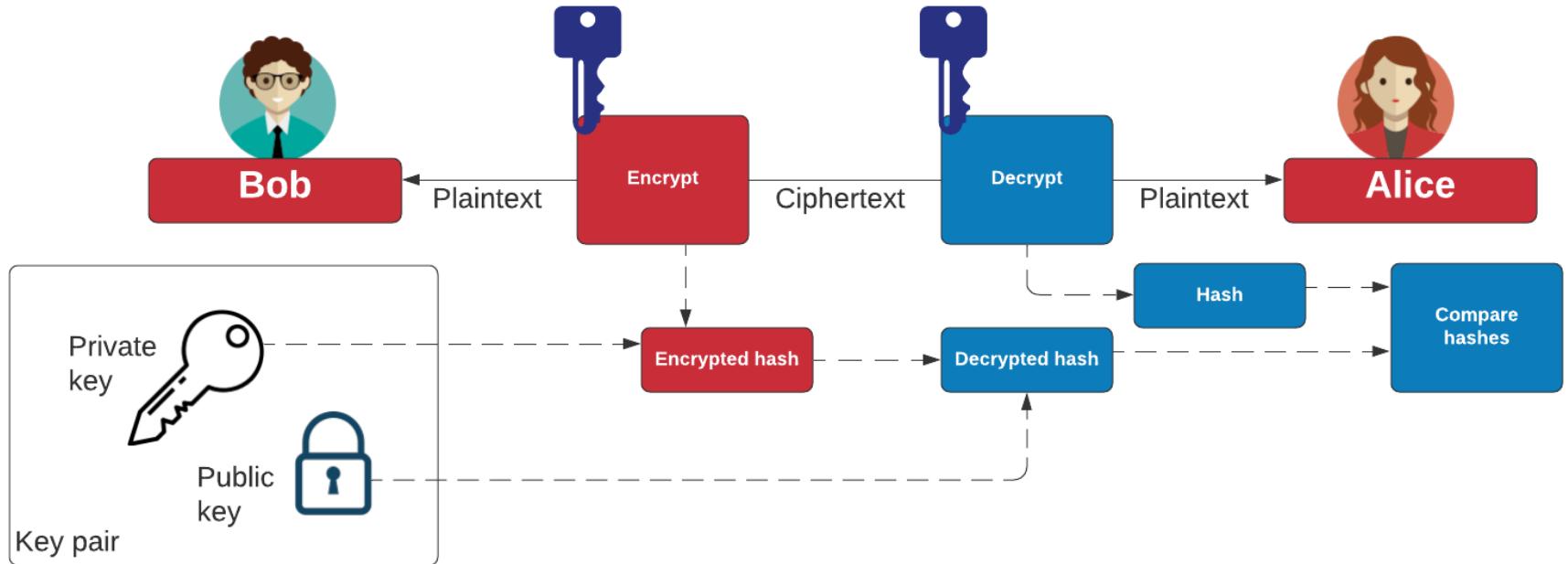
Bitcoin Key Generation



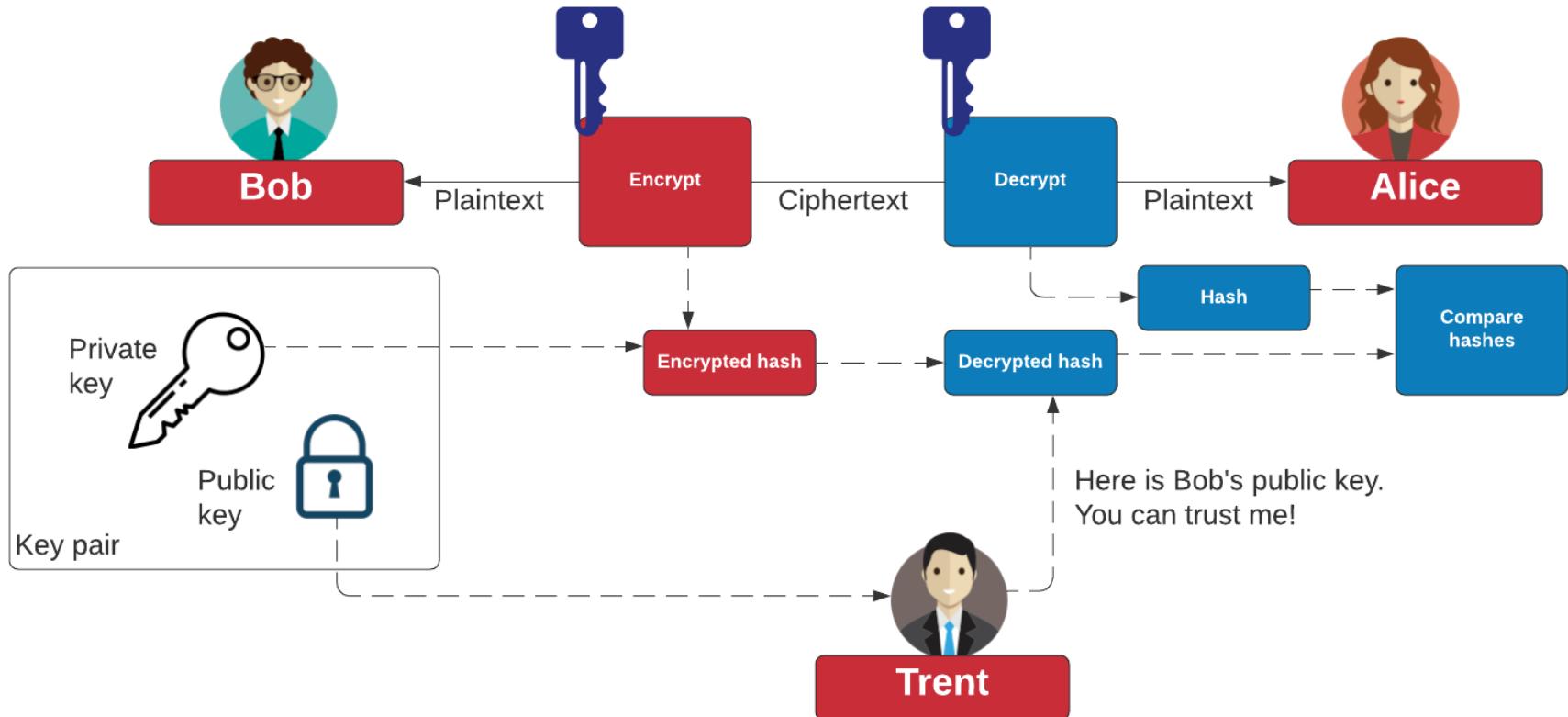
Bitcoin Transaction



Signing



Signing



Example

Elliptic Curve (EC)

```
C \ > openssl ecparam -name secp256k1 -genkey -out priv.pem
```

```
C \ > type ec-priv.pem
```

```
-----BEGIN EC PARAMETERS-----
```

```
BgUrgQQACg==
```

```
-----END EC PARAMETERS-----
```

```
-----BEGIN EC PRIVATE KEY-----
```

```
MHQCAQEEIEa56GG2PTUJylt4FydaMNltYsjNj6Zlbd7jXvDY4ElfoAcGBSuBBAK  
oUQDQgAEJQDn8/vd8oQpA/VE3ch0IM6VAprOTiV9VLp38rwfOog3qUYcTxxX/sxJ  
I1M4HncqEopYIKkkovoFFi62Yph6nw==
```

```
-----END EC PRIVATE KEY-----
```

Example

Elliptic Curve (EC)

```
C \ > openssl ecparam -name secp256k1 -genkey -out priv.pem
```

```
C \ > type ec-priv.pem
```

```
-----BEGIN EC PARAMETERS-----
```

```
BgUrgQQACg==
```

```
-----END EC PARAMETERS-----
```

```
-----BEGIN EC PRIVATE KEY-----
```

```
MHQCAQEEIEa56GG2PTUJylt4FydaMNltYsjNj6Zlbd7jXvDY4ElfoAcGBSuBBAK  
oUQDQgAEJQDn8/vd8oQpA/VE3ch0IM6VAprOTiV9VLp38rwfOog3qUYcTxxX/sxJ  
I1M4HncqEopYIKkkovoFFi62Yph6nw==
```

```
-----END EC PRIVATE KEY-----
```

Example

Elliptic Curve (EC)

```
C \ > openssl ecparam -name secp256k1 -genkey -out priv.pem
C \ > openssl ec -in priv.pem -text -noout
C \ > read EC key
-----E Private-Key (256 bit)
BgUr priv
-----E 46 b9 e8 61 b6 3d 35 09 c8 8b 78 17 27 5a 30
-----E d2 2d 62 c8 cd 8f a6 48 6d de e3 5e f0 d8 e0
MHG 49 5f
oUQ pub
l1M4 04 25 00 e7 f3 fb dd f2 84 29 03 f5 44 dd c8
-----E 74 94 ce 95 02 9a ce 4e 25 7d 54 ba 77 f2 bc
1f 3a 88 37 a9 46 1c 4f 1c 57 fe cc 49 97 53
38 1e 77 2a 12 8a 58 20 a9 24 a2 fa 05 16 2e
b6 62 98 7a 9f
ASN1 OID secp256k1
```

Example

Elliptic Curve (EC)

```
C \ > openssl ecparam -name secp256k1 -genkey -out priv.pem
```

```
C \ > type ec-priv.pem
```

```
-----BEGIN EC PARAMETERS-----
```

```
BgUrgQQACg==
```

```
-----END EC PARAMETERS-----
```

```
-----BEGIN EC PRIVATE KEY-----
```

```
MHQCAQEEIEa56GG2PTUJylt4FydaMNltYsjNj6Zlbd7jXvDY4ElfoAcGBSuBBAK  
oUQDQgAEJQDn8/vd8oQpA/VE3ch0IM6VAprOTiV9VLp38rwfOog3qUYcTxxX/sxJ  
I1M4HncqEopYIKkkovoFFi62Yph6nw==
```

```
-----END EC PRIVATE KEY-----
```

Example

Elliptic Curve (EC)

```
C \> openssl ecparam -name secp256k1 -genkey -out priv.pem
C \> openssl ec -in priv.pem -text -noout
C \> read EC key
-----BEGIN PRIVATE KEY----- (256 bit)
BgUrpriv
-----END PRIVATE KEY----- (256 bit)
-----BEGIN PUBLIC KEY----- (256 bit)
MHC495f
-----END PUBLIC KEY----- (256 bit)
I1M4042500e7f3fbddf2842903f544ddc8
-----END PUBLIC KEY----- (256 bit)
ASN1 OID: secp256k1
```

Example

```
C:> openssl ecparam -in priv.pem -text -param_enc explicit -noout
```

```
Field Type: prime-field
```

```
Prime:
```

```
00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:  
ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:fe:ff:
```

```
ff:fc:2f
```

```
A: 0
```

```
B: 7 (0x7)
```

```
Generator (uncompressed):
```

```
04:79:be:66:7e:f9:dc:bb:ac:55:a0:62:95:ce:87:
```

```
0b:07:02:9b:fc:db:2d:ce:28:d9:59:f2:81:5b:16:
```

```
f8:17:98:48:3a:da:77:26:a3:c4:65:5d:a4:fb:fc:
```

```
0e:11:08:a8:fd:17:b4:48:a6:85:54:19:9c:47:d0:
```

```
8f:fb:10:d4:b8
```

```
Order:
```

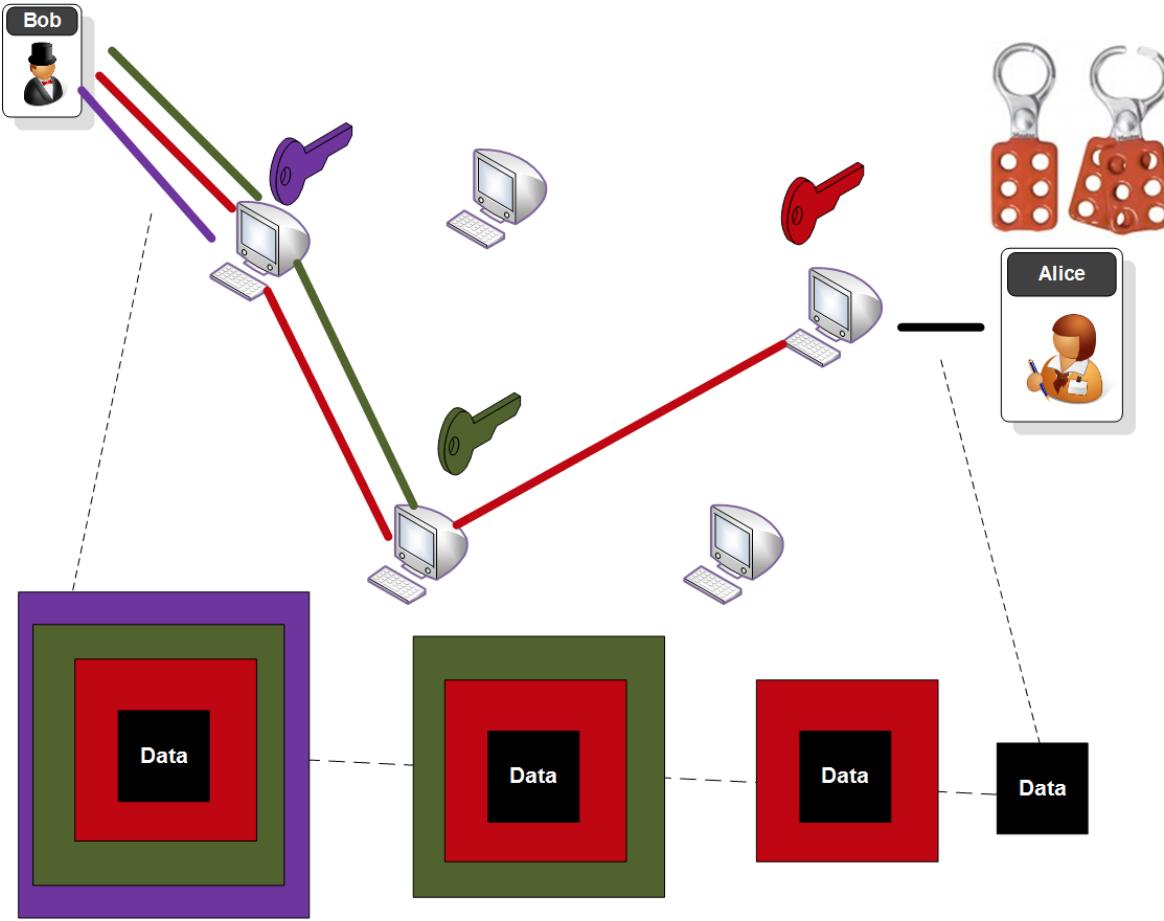
```
38: 00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
```

```
b6: ff:fe:ba:ae:dc:e6:af:48:a0:3b:bf:d2:5e:8c:d0:
```

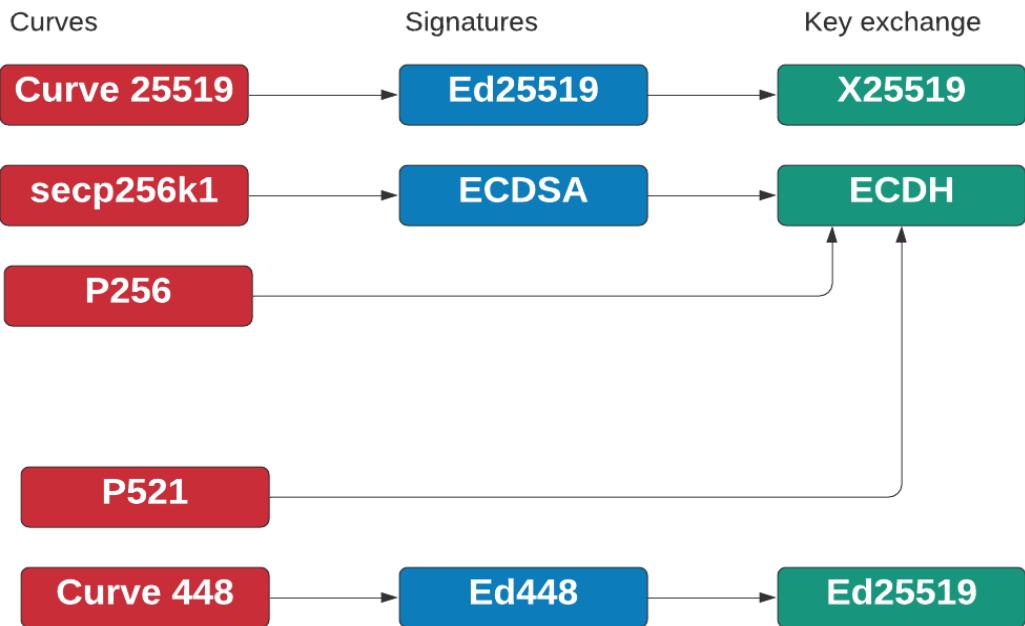
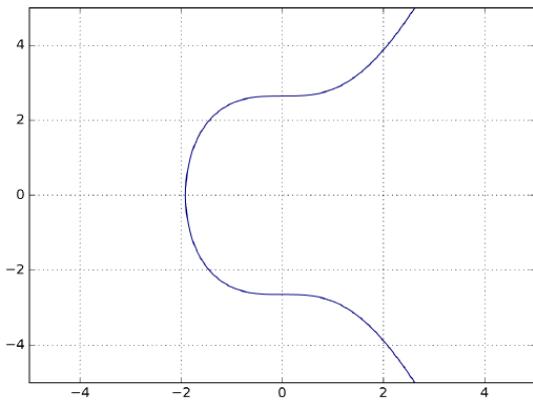
```
ASN1: 36:41:41
```

```
Cofactor: 1 (0x1)
```

Elliptic Curve Diffie Hellman (ECDH)



ECC Methods



Unit 4: Public Key

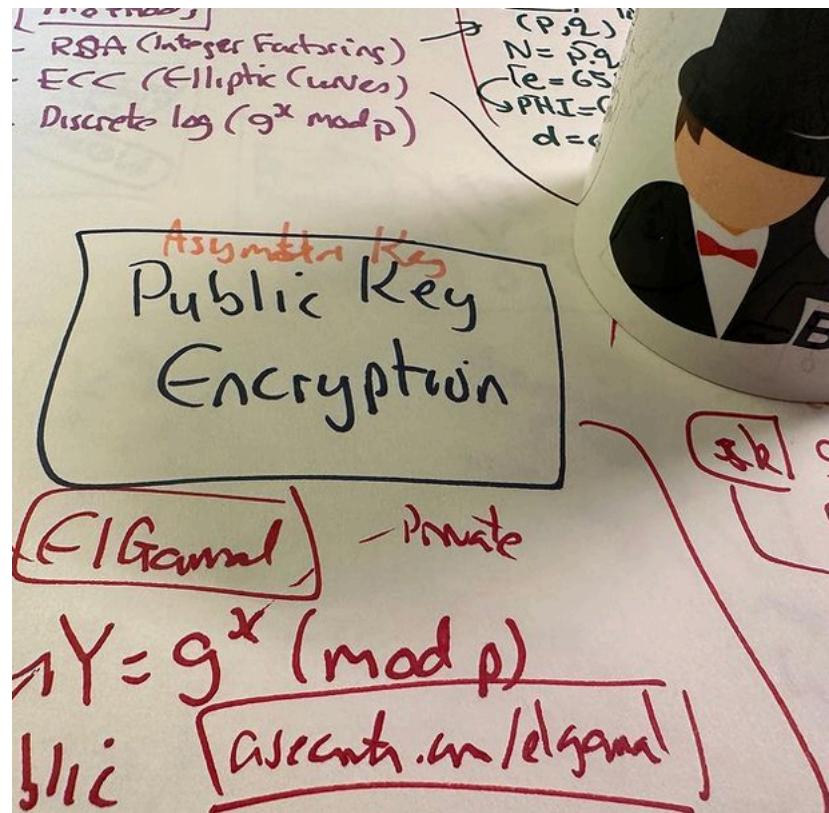
ElGamal

Prof Bill Buchanan OBE

<https://asecuritysite.com/rsa>

<https://asecuritysite.com/ecc>

<https://asecuritysite.com/elgamal>



ElGamal



- $Y = g^x \text{ mod } p$
- g is picked from cyclic group
(Explained in Key Handshaking section). [Here](#).
- p is a prime number.
- Example [here](#).

Unit 4: Public Key

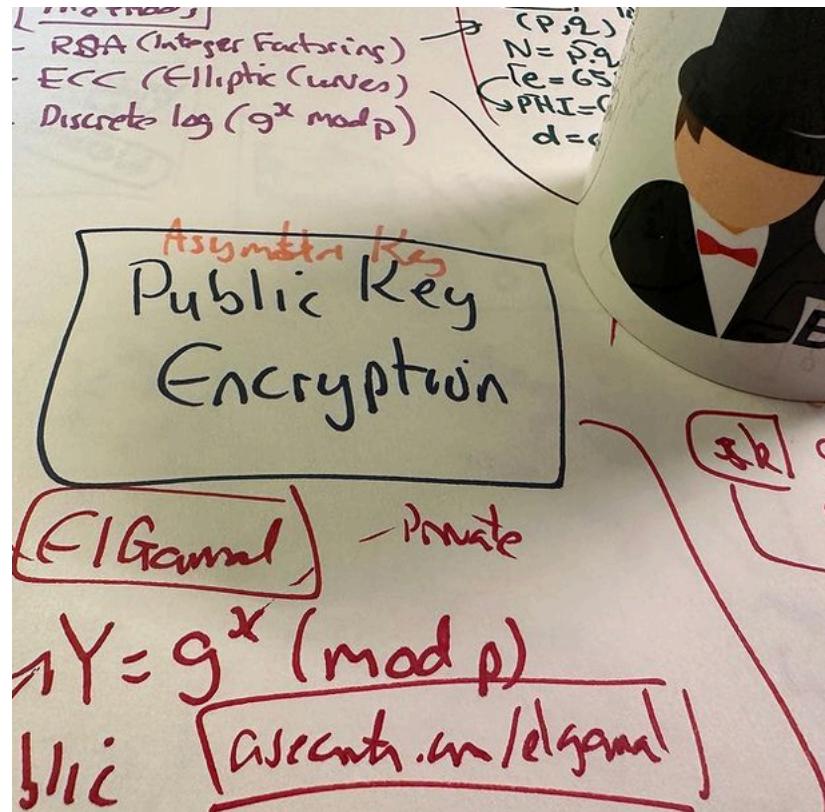
PGP/Box

Prof Bill Buchanan OBE

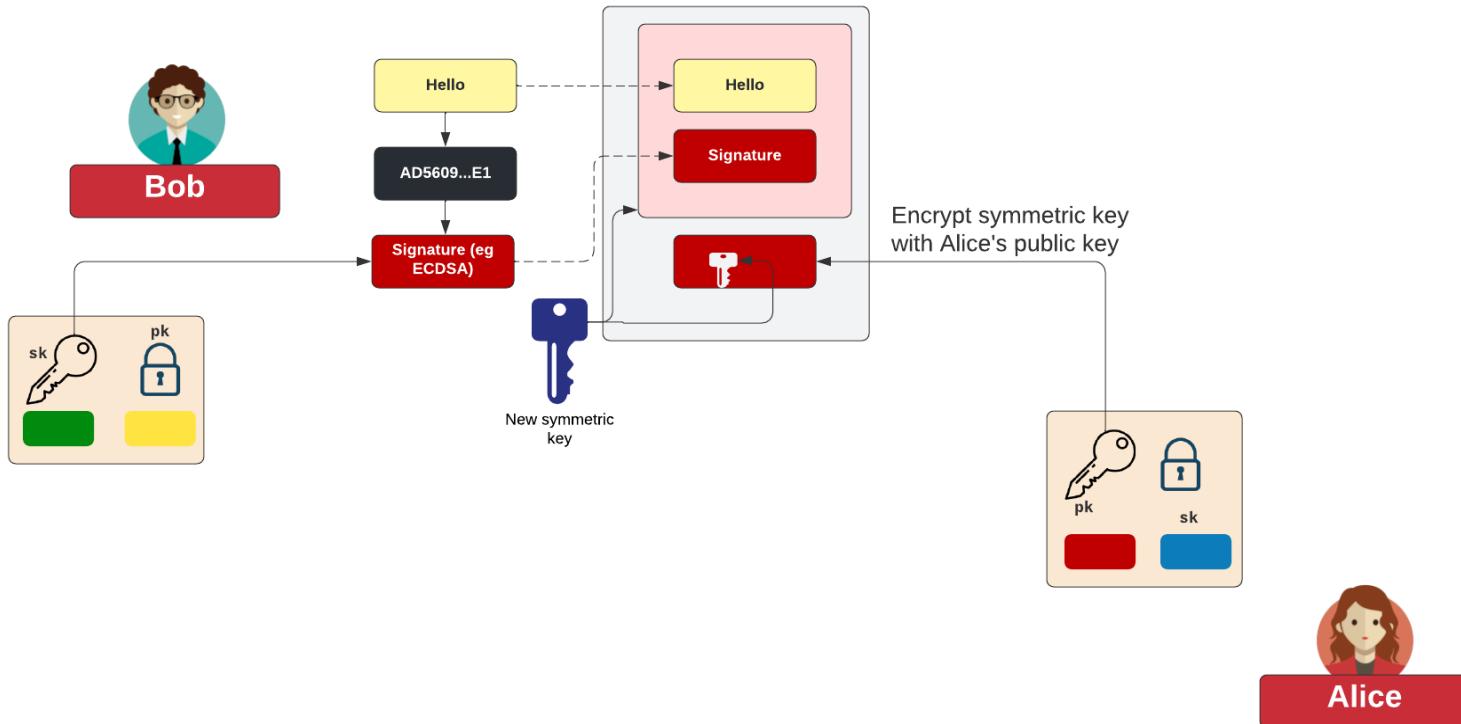
<https://asecuritysite.com/rsa>

<https://asecuritysite.com/ecc>

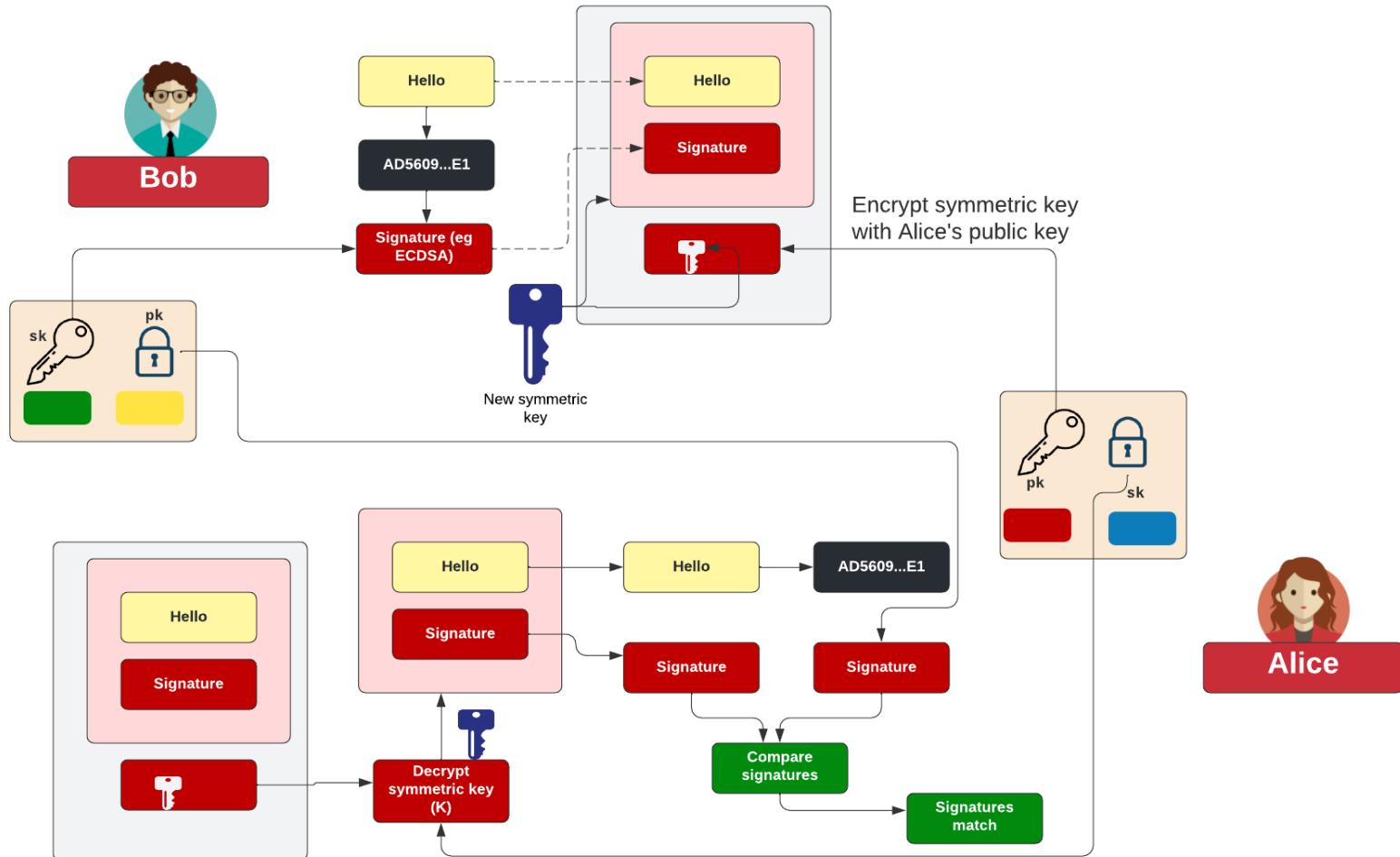
<https://asecuritysite.com/elgamal>



PGP



PGP



Unit 4: Public Key

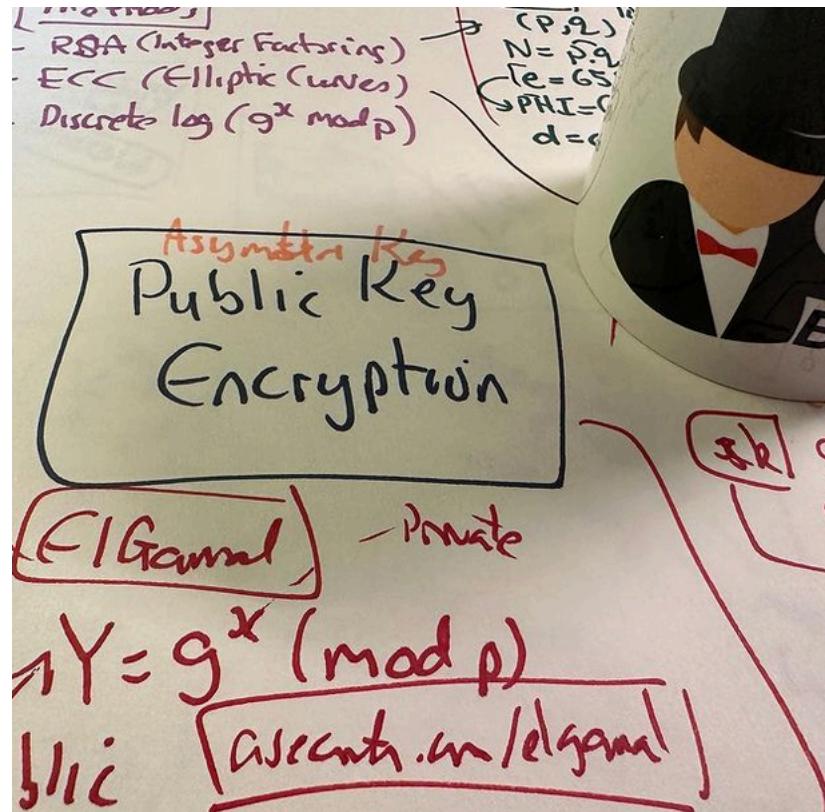
AWS Key Management System

Prof Bill Buchanan OBE

<https://asecuritysite.com/rsa>

<https://asecuritysite.com/ecc>

<https://asecuritysite.com/elgamal>



KMS

Key Management Service (KMS)

AWS managed keys
Customer managed keys
Custom key stores

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

KMS > Customer managed keys

Customer managed keys (3)

| Aliases | Key ID | Status | Key spec | Key usage |
|---------------------------------------|--------------------------------------|---------|-------------------|---------------------|
| <input type="checkbox"/> MyPublicKey | 68ded69b-6c19-4b34-9f91-f8c2628ee612 | Enabled | RSA_2048 | Encrypt and decrypt |
| <input type="checkbox"/> BillsNewKey | 98a90e1f-2cb5-4564-a3aa-d0c060cf1ea | Enabled | SYMMETRIC_DEFAULT | Encrypt and decrypt |
| <input type="checkbox"/> MyPublicKey2 | mrk-563b89d2385b4e70899e0dfd5158ef7b | Enabled | RSA_2048 | Encrypt and decrypt |

Demo

Configure key



Key type Help me choose

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

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

Key usage Help me choose

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

Sign and verify
Key pairs for digital signing
Uses the private key for signing and the public key for verification.

Key spec Help me choose

RSA_2048
 RSA_3072
 RSA_4096

Viewing public key

Key policy | Cryptographic configuration | Tags | **Public key** | Aliases

Copy Download

Public key

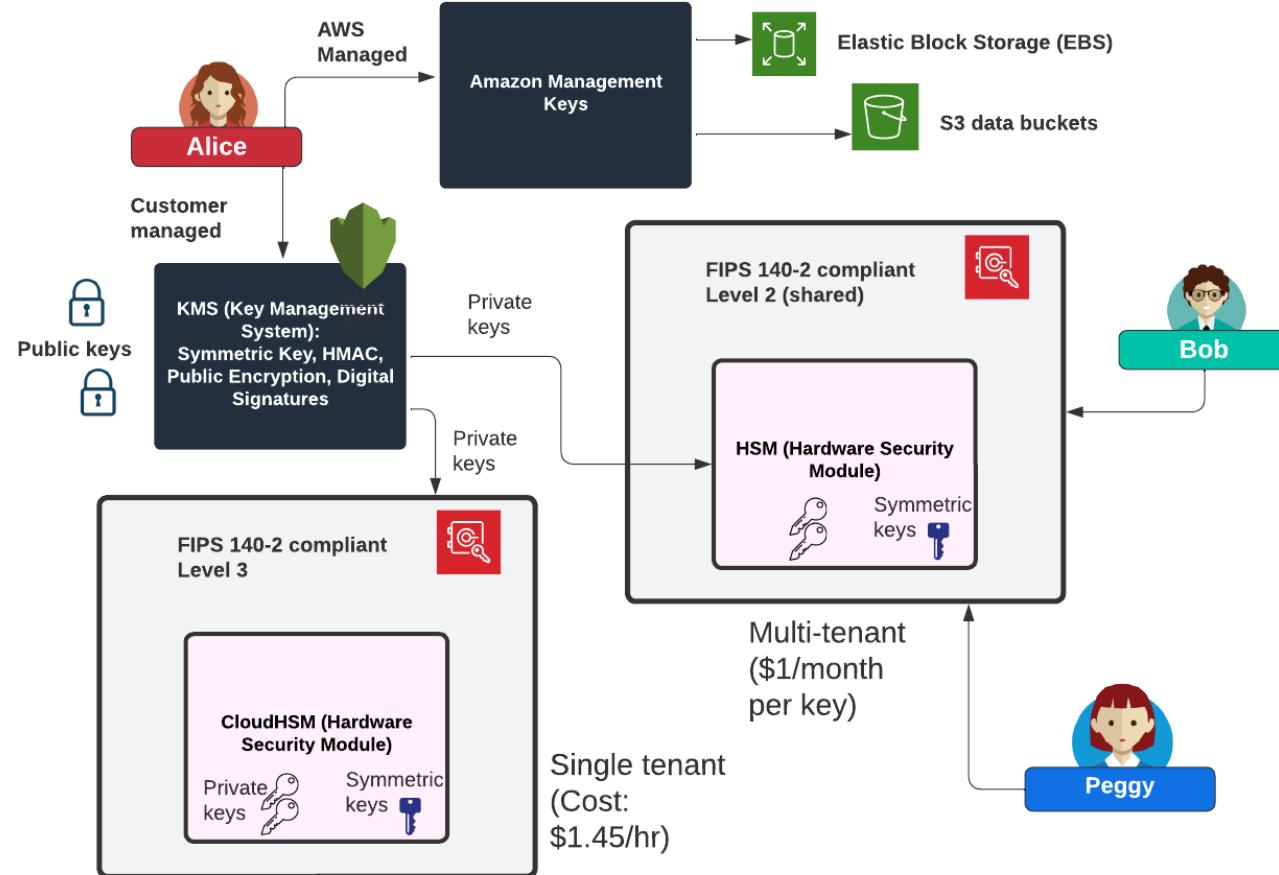
Public key for encryption operations only. Learn more

```
-----BEGIN PUBLIC KEY-----  
MIIBIjANBgkqhkiG9wBAQEFAOCAQ8AMII�gKCAQEAsXdtH0dCete0bzugPF6E  
Njeft6CDGjbbaR9t40++q4jqtSd5lsdYe1Rn3mL+oXqKJz9o+aoXcCMfhku6w  
qqDvbIOP7nsXl03p+4G7uUS93g3cCSRodEAn3jb2ybjhjvfs90BSBM7bh6Kw21Yu/  
Yn+onUjGal/d4o7+Nyu0mDEambNh+1Q61rpfbu1Y231gpbld78xG1v1dz2nq  
yBG8vaZ#W90fr05jDcpDrNm109QX10pEhwNGcvcsxchods1AzrlzKUre/nZ5MTNL  
3uigw8wSlzQURFibpLHKpcNBaxu3zoSMK2Dvj+1+L2PejlydAPfqB5N8ds0  
AQIDAQAB  
-----END PUBLIC KEY-----
```

Download public key

```
% aws kms get-public-key --key-id alias/PublicKeyForDemo  
{  
    "KeyId": "arn:aws:kms:us-east-1:103269750866:key/de30e8e6-c753-4a2c-881a-53c761242644",  
    "PublicKey": "MIIBIjANBgkqhkiG9wBAQEFAOCAQ8AMII�gKCAQEAsXdtH0dCete0bzugPF6E  
Njeft6CDGjbbaR9t40++q4jqtSd5lsdYe1Rn3mL+oXqKJz9o+aoXcCMfhku6w  
qqDvbIOP7nsXl03p+4G7uUS93g3cCSRodEAn3jb2ybjhjvfs90BSBM7bh6Kw21Yu/  
Yn+onUjGal/d4o7+Nyu0mDEambNh+1Q61rpfbu1Y231gpbld78xG1v1dz2nq  
yBG8vaZ#W90fr05jDcpDrNm109QX10pEhwNGcvcsxchods1AzrlzKUre/nZ5MTNL  
3uigw8wSlzQURFibpLHKpcNBaxu3zoSMK2Dvj+1+L2PejlydAPfqB5N8ds0AQIDAQAB",  
    "CustomMasterKeySpec": "RSA_2048",  
    "KeySpec": "RSA_2048",  
    "KeyUsage": "ENCRYPT_DECRYPT",  
    "EncryptionAlgorithms": [  
        "RSAES_OAEP_SHA_1",  
        "RSAES_OAEP_SHA_256"  
    ]  
}
```

HSM or CloudHSM



Unit 4: Public Key

Basics

RSA

Elliptic Curve

ElGamal

KMS

Prof Bill Buchanan OBE

<https://asecuritysite.com/rsa>

<https://asecuritysite.com/ecc>

<https://asecuritysite.com/elgaml>

