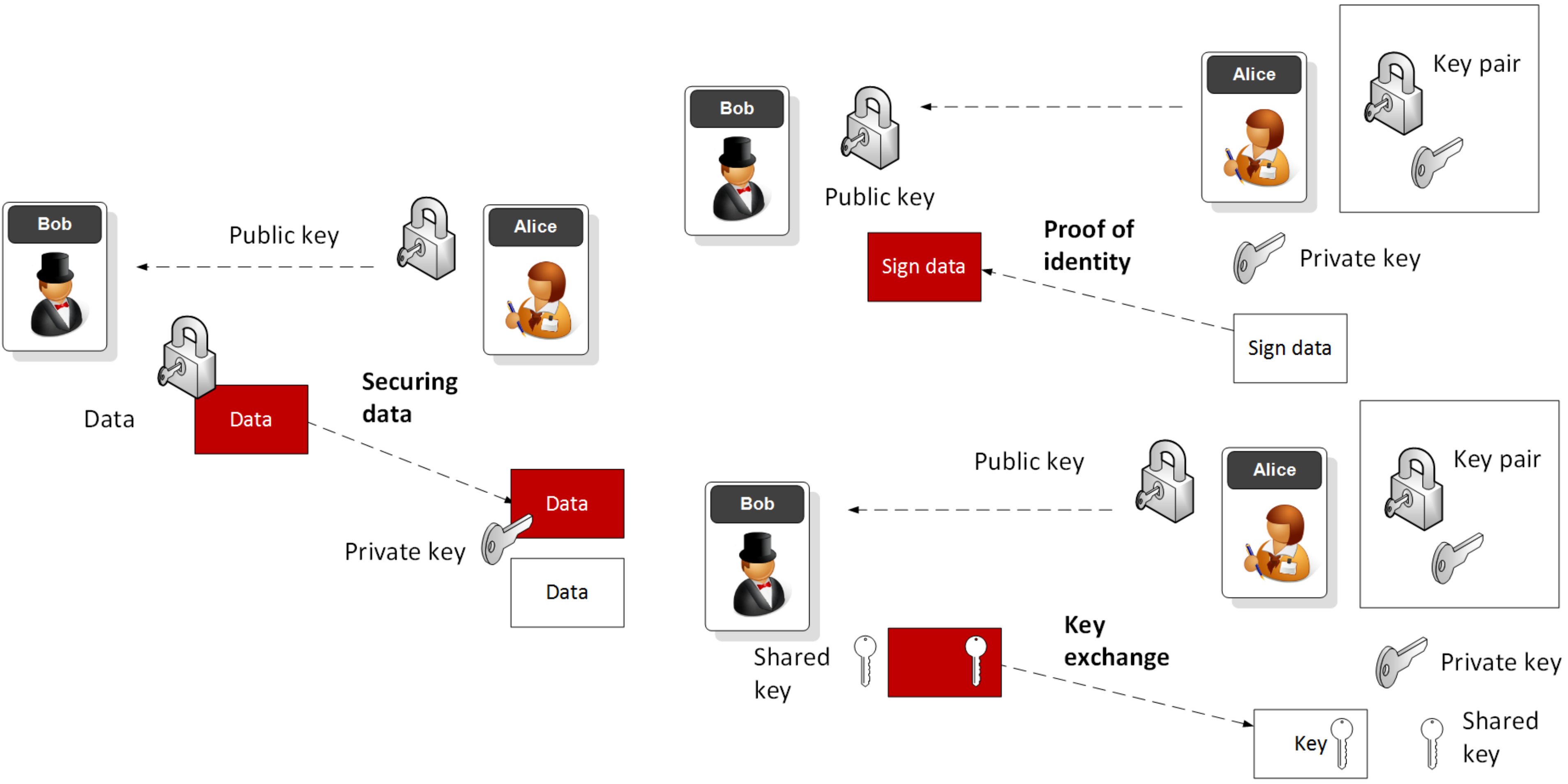
Chapter 4: Public Key

Basics
RSA (Factorizing Primes)
Elliptic Curve (Elliptic Curves)
ElGamal (Discrete Logs)

Prof Bill Buchanan OBE

http://asecuritysite.com/crypto04
http://asecuritysite.com/encryption

Public Key Methods



Public Key Methods

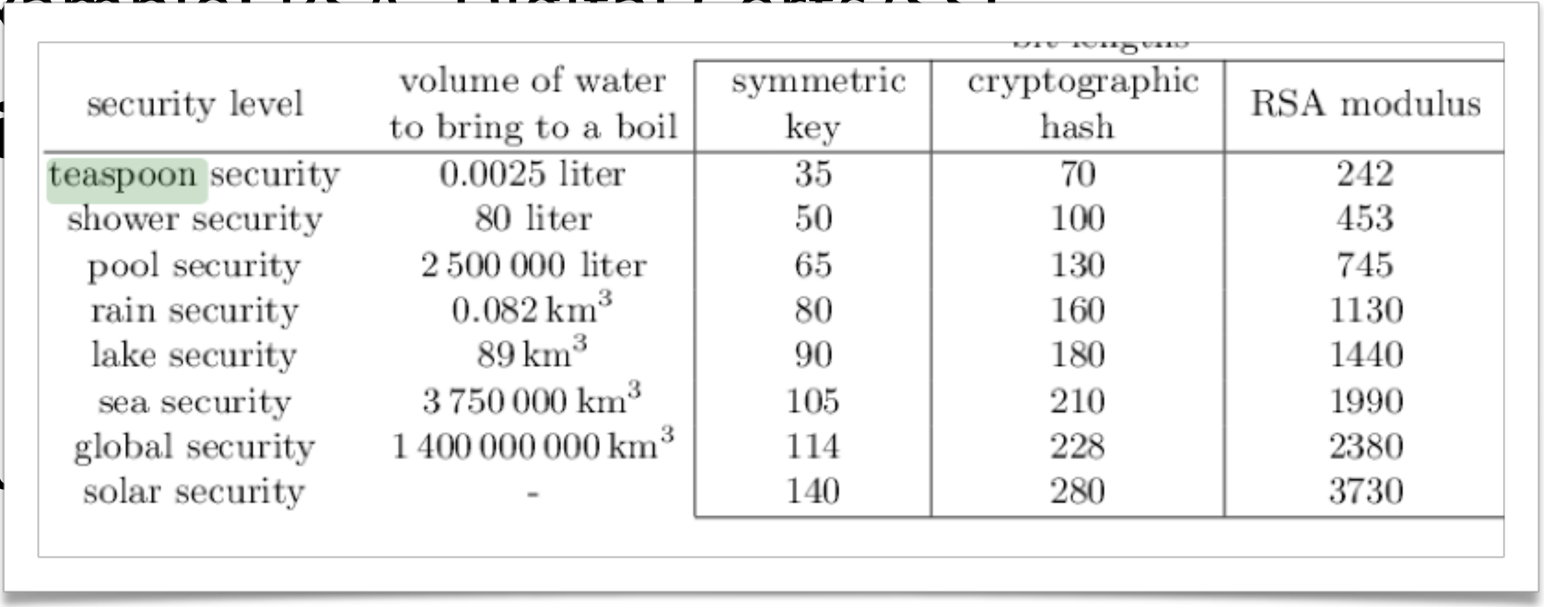
- **Integer Factorization**. Using prime numbers. Example: RSA. Digital Certs/SSL.
- **Discrete Logarithms**. $Y = G^x \bmod P$. Example: ElGamal.
- **Elliptic Curve Relationships**. Example: Elliptic Curve. Smart Cards, IoT, Tor, Bitcoin.

Public Key Methods

- **Integer Factorization**. Using prime numbers. Example: RSA. Digital Certs/SSL.
- **Discrete Logarithms**. $Y = G^x \bmod P$. 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
- **Di**
  El
- **El**
  Cu



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

Chapter 4: Public Key

RSA

Prof Bill Buchanan OBE

http://asecuritysite.com/crypto04
http://asecuritysite.com/encryption

Eve

p

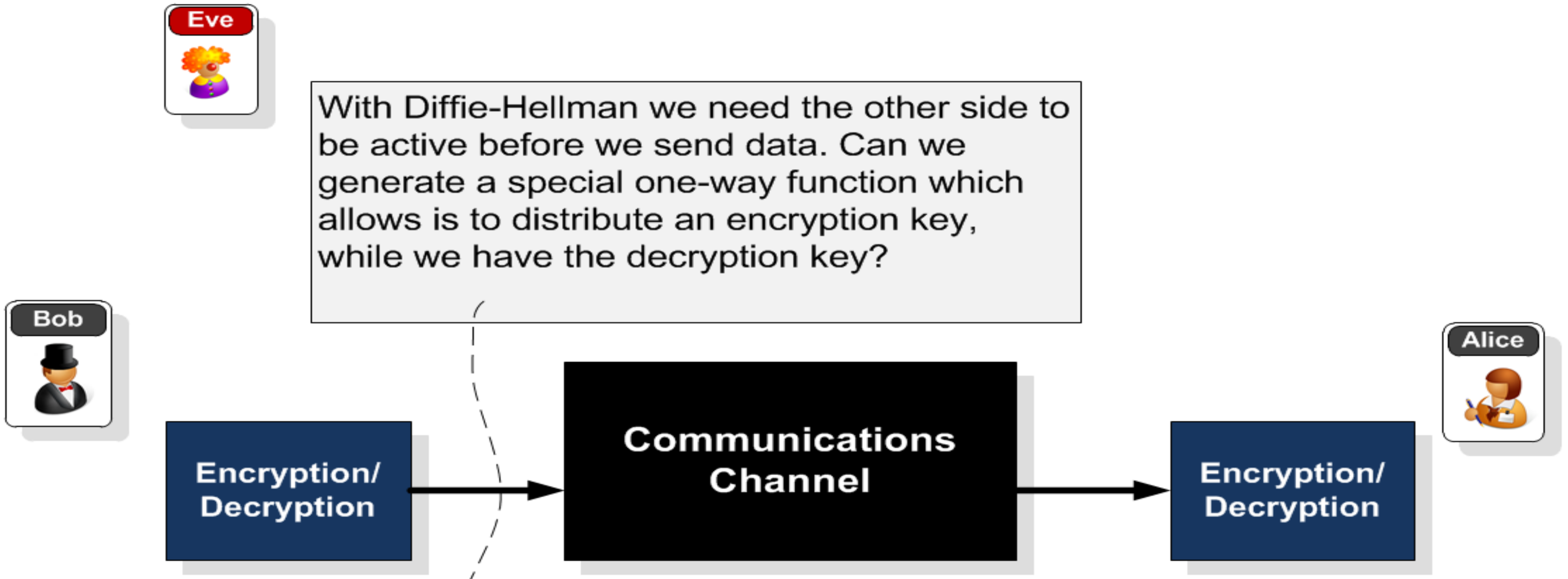
9,137,187,070,061,098,912,312,979,400,361
,251,189,847,923,809,497,258,114,688,790,
849,334,008,324,856,676,348,809,151,285,1
18,821,829,375,998,699,013,311,467,364,66
2,378,853,216,263,996,490,005,611,058,805

p

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

(p-1)*(q-1)

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



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

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

RSA

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

RSA

Calc

Example

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

eg M=5.

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

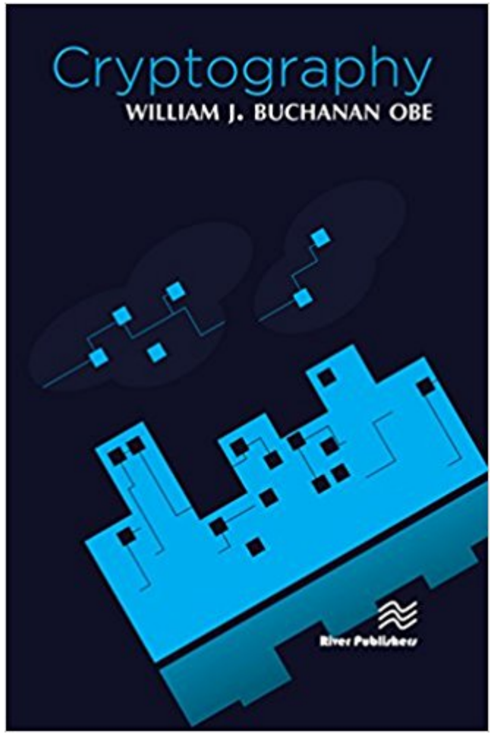
Chapter 4: Public Key

Elliptic Curve

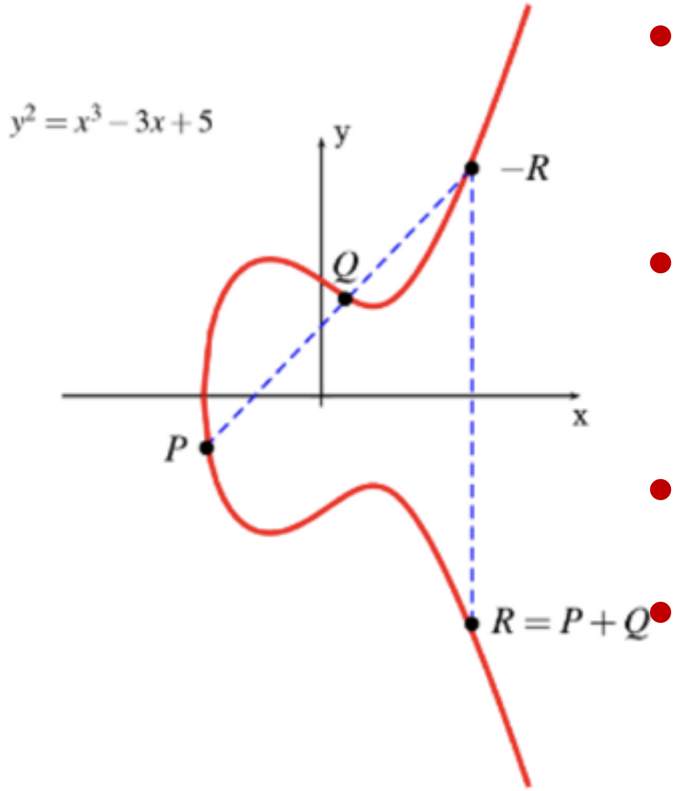
Prof Bill Buchanan OBE

http://asecuritysite.com/crypto04

http://asecuritysite.com/encryption

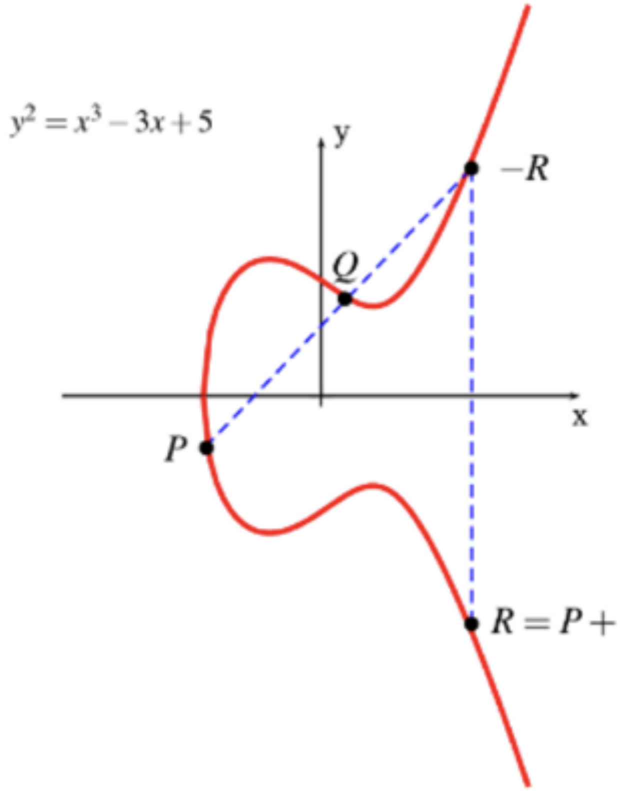


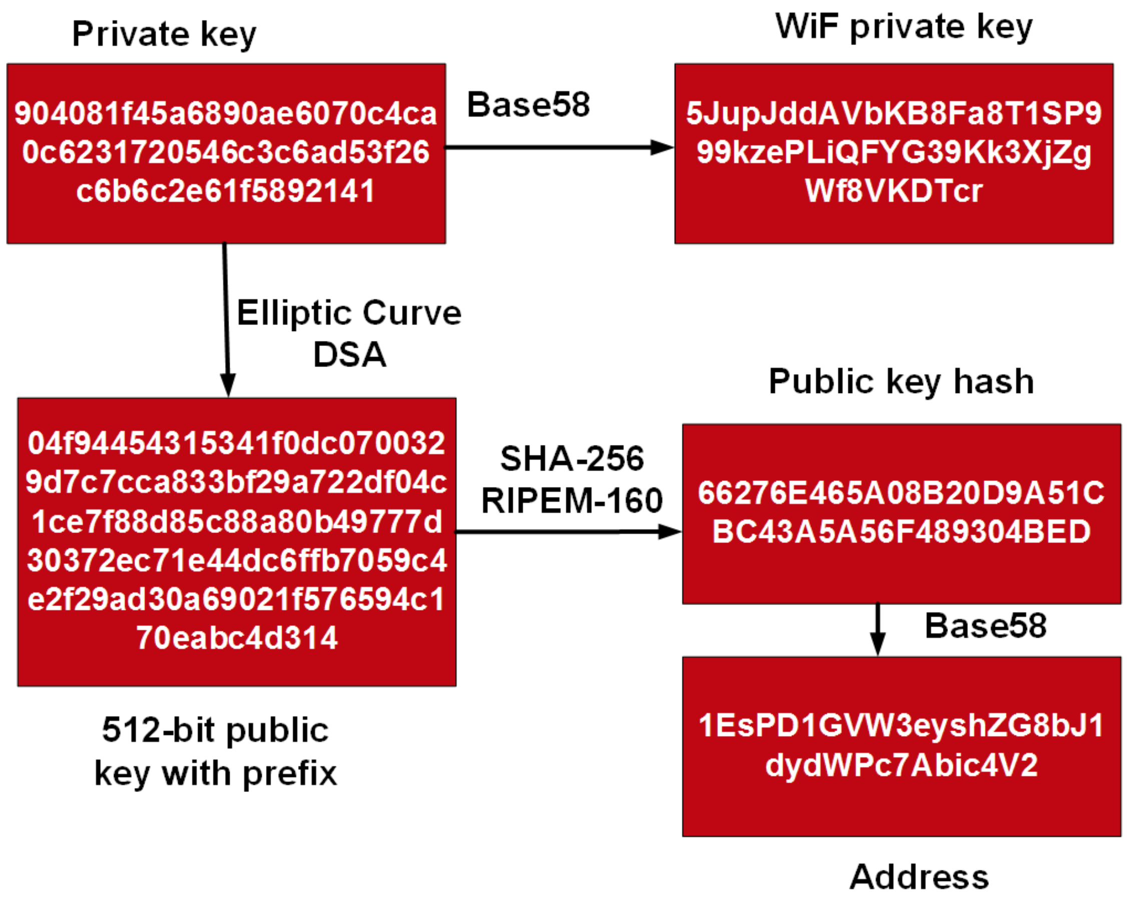
Elliptic Curve (EC)



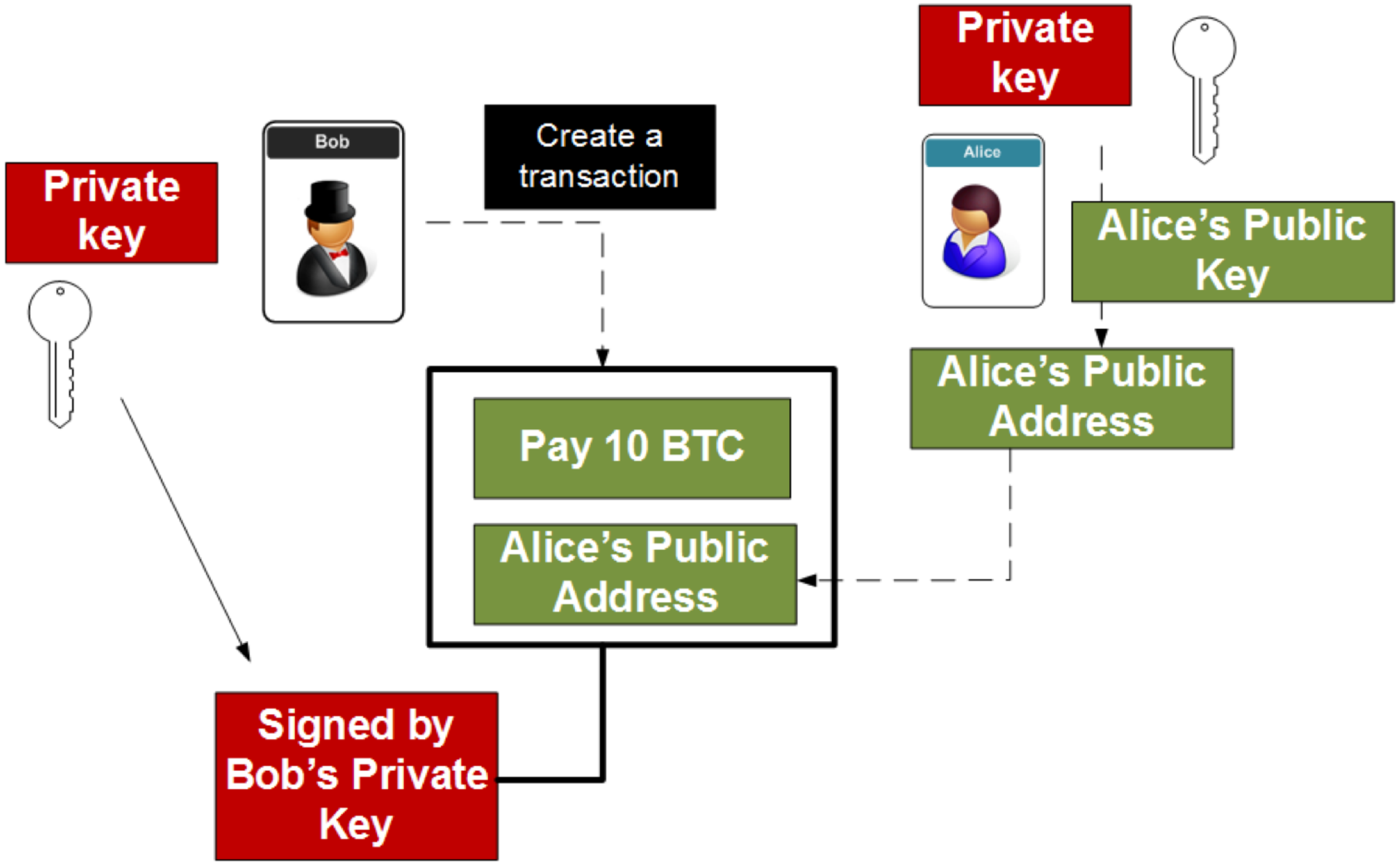
- Pick a point on the elliptic curve (G).
- Generate a random number (n) – this will be the private key.
- Public key is P = n x G
- Bitcoin, IoT and Tor use Curve 55219 [here].

Bitcoin Key Generation





Bitcoin Transaction



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-----
MHQCAQEEIEa56GG2PTUJylt4FydaMNItYsjNj6ZIbd7jXvDY4ElfoAcGBSuBBAAK
oUQDQgAEJQDn8/vd8oQpA/VE3ch0lM6VAprOTiV9VLp38rwfOog3qUYcTxxX/sxJ
l1M4HncqEopYIKkkovoFFi62Yph6nw==
-----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-----
MHQCAQEEIEa56GG2PTUJyIt4FydaMNItYsjNj6ZIbd7jXvDY4ElfoAcGBSuBBAAK
oUQDQgAEJQDn8/vd8oQpA/VE3ch0lM6VAprOTiV9VLp38rwfOog3qUYcTxxX/sxJ
l1M4HncqEopYIKkkovoFFi62Yph6nw==
-----END EC PRIVATE KEY-----
```

Example

Elliptic Curve (EC)

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

C \ >
-----B
BgUr
-----E
-----B
MHC
oUQ
l1M4
-----E
```

```
C \> openssl ec -in priv.pem -text -noout
read EC key
Private-Key  (256 bit)
priv
    46 b9 e8 61 b6 3d 35 09 c8 8b 78 17 27 5a 30
    d2 2d 62 c8 cd 8f a6 48 6d de e3 5e f0 d8 e0
    49 5f
pub
    04 25 00 e7 f3 fb dd f2 84 29 03 f5 44 dd c8
    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-----
MHQCAQEEIEa56GG2PTUJyIt4FydaMNItYsjNj6ZIbd7jXvDY4ElfoAcGBSuBBAAK
oUQDQgAEJQDn8/vd8oQpA/VE3ch0lM6VAprOTiV9VLp38rwfOog3qUYcTxxX/sxJ
l1M4HncqEopYIKkkovoFFi62Yph6nw==
-----END EC PRIVATE KEY-----
```

Example

Elliptic Curve (EC)

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

C \ >
-----B
BgUr
-----E
-----B
MHC
oUQ
l1M4
-----E
```

```
C \> openssl ec -in priv.pem -text -noout
read EC key
Private-Key  (256 bit)
priv
    46 b9 e8 61 b6 3d 35 09 c8 8b 78 17 27 5a 30
    d2 2d 62 c8 cd 8f a6 48 6d de e3 5e f0 d8 e0
    49 5f
pub
    04 25 00 e7 f3 fb dd f2 84 29 03 f5 44 dd c8
    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

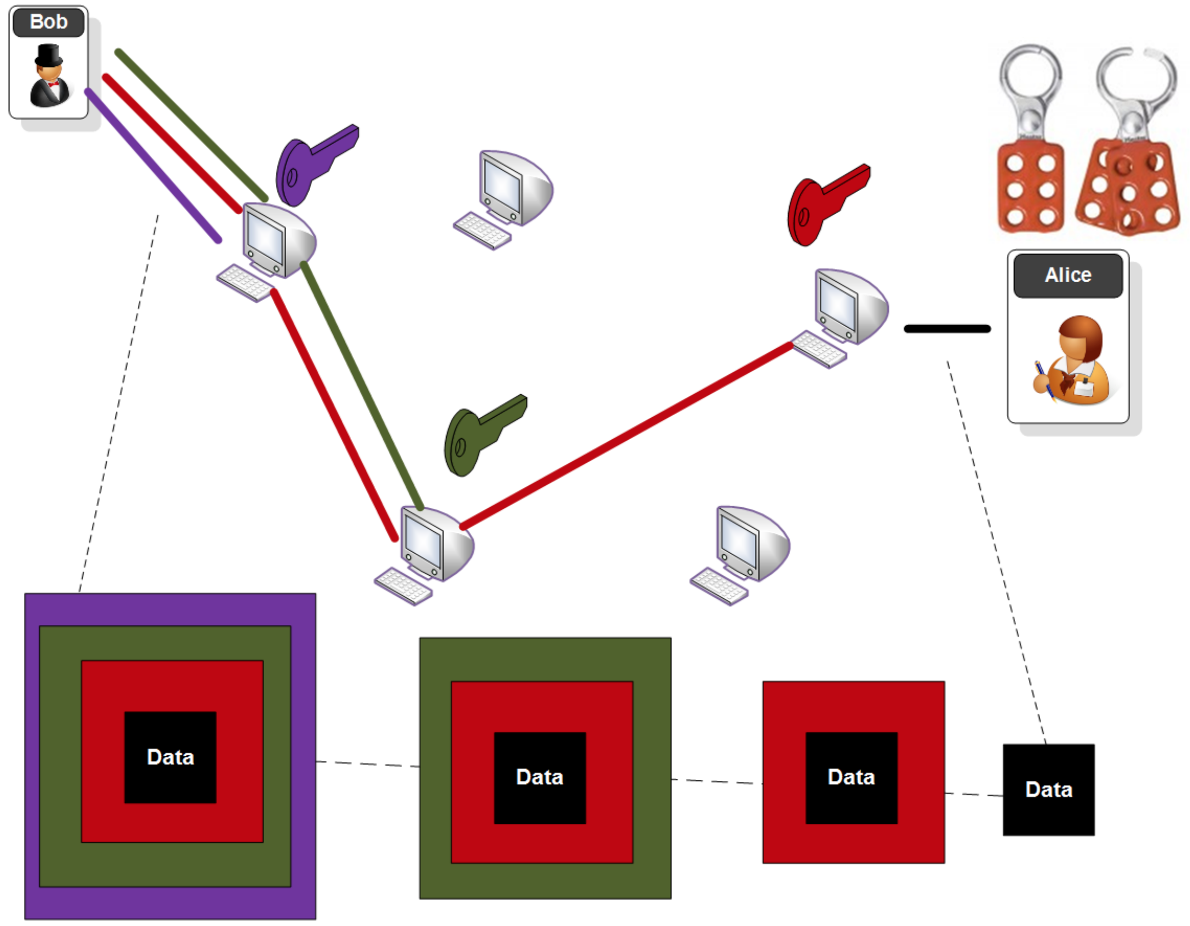
```
C \ > open

C \ >
-----B
BgUr
-----E
-----B
MHC
oUQ
l1M4
-----E
```

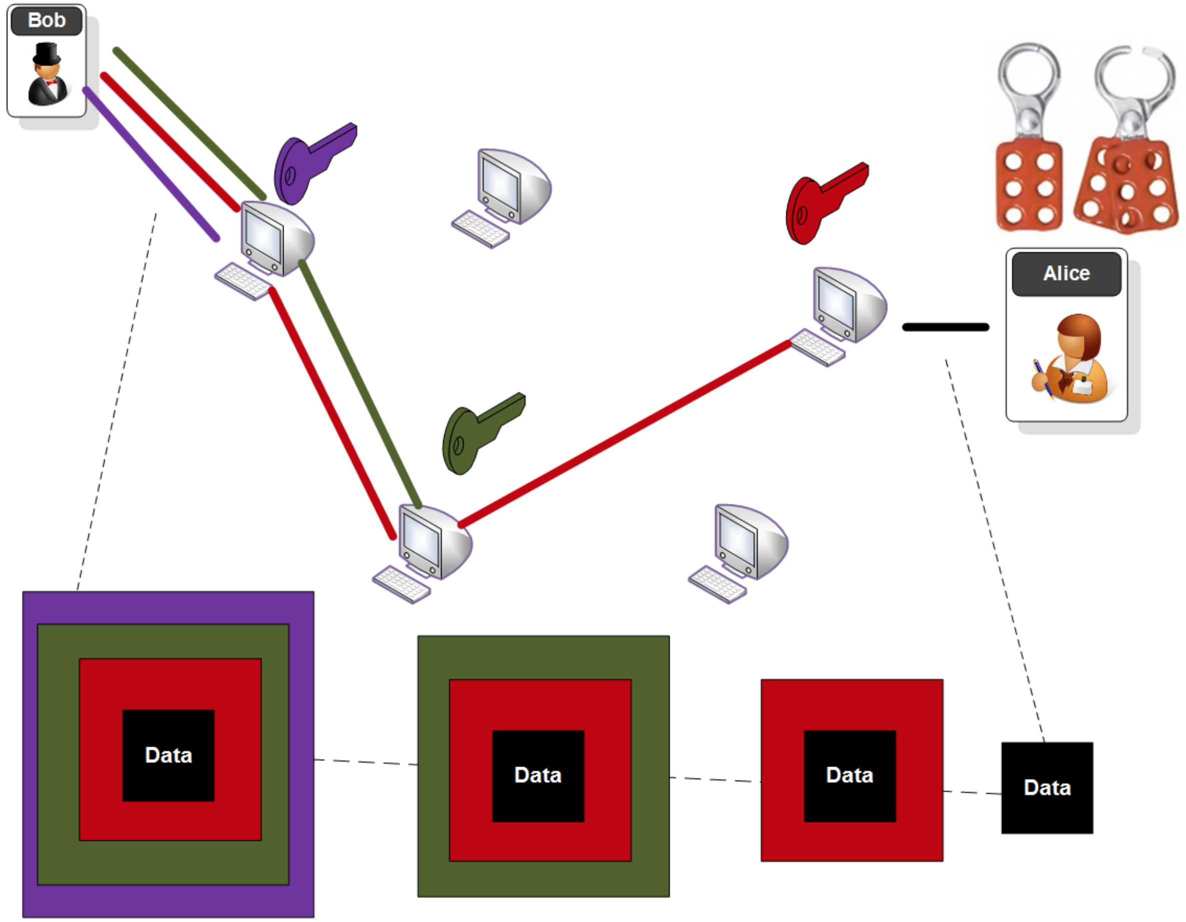
```
C \> (
read
Priva
priv
  46
  d2
  49
pub
  04
  74
  1f
  38
  b6
ASN1
```

```
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: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:
    00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
    ff:fe:ba:ae:dc:e6:af:48:a0:3b:bf:d2:5e:8c:d0:
    36:41:41
Cofactor:  1 (0x1)
```

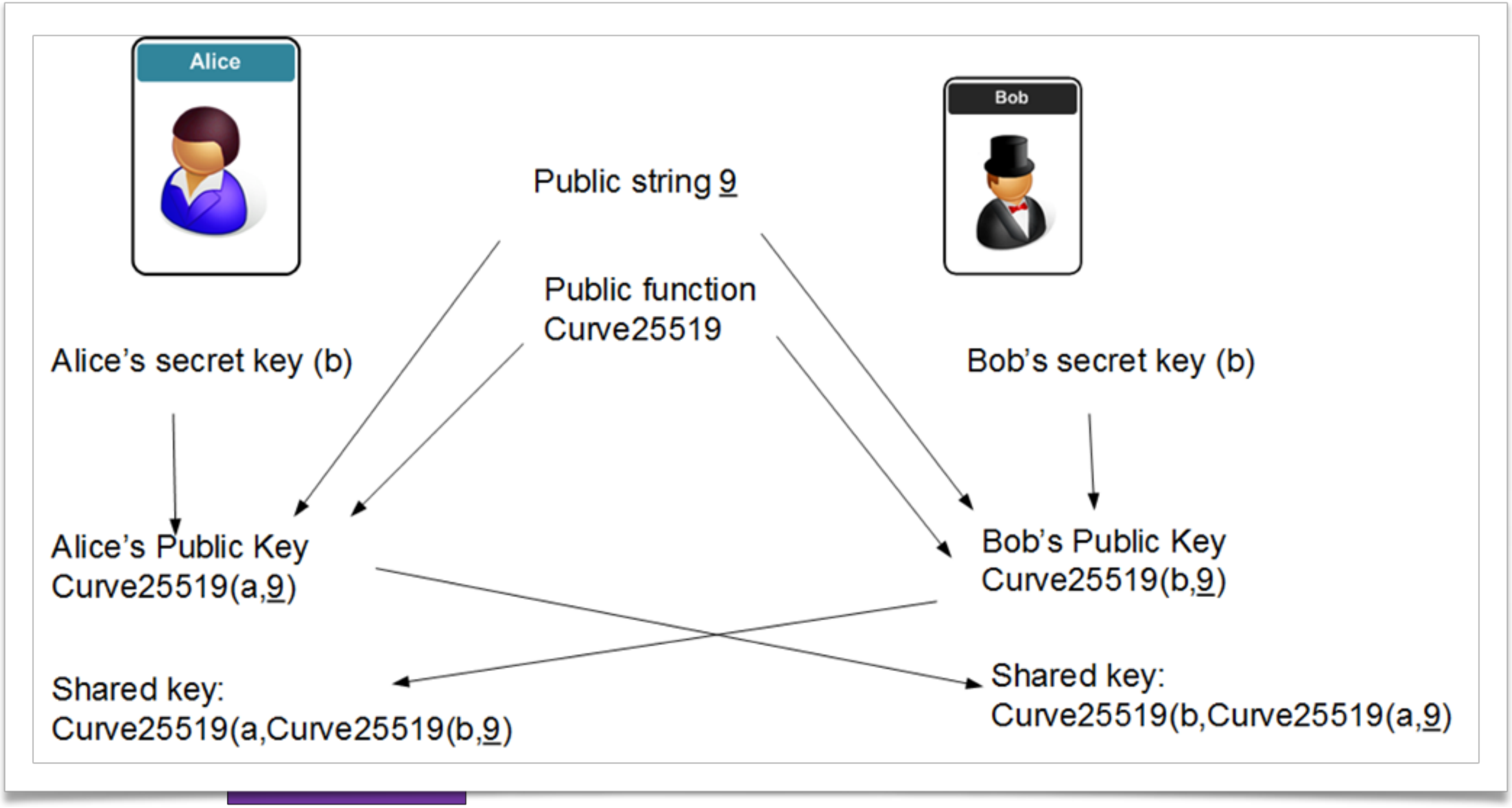
Elliptic Curve Diffie Hellman (ECDH)



Elliptic Curve Diffie Hellman (ECDH)



Elliptic Curve Diffie Hellman (ECDH)


Alice
Bob
Public string 9
Public function Curve25519
Alice's secret key (b)
Bob's secret key (b)
Alice's Public Key Curve25519(a,9)
Bob's Public Key Curve25519(b,9)
Shared key: Curve25519(a,Curve25519(b,9)
Shared key: Curve25519(b,Curve25519(a,9)

Chapter 4: Public Key

ElGamal

Prof Bill Buchanan OBE

http://asecuritysite.com/crypto04
http://asecuritysite.com/encryption

ElGamal

- $Y = G^x \mod p$
- G is picked from cyclic group (Explained in Key Handshaking section). Here.
- p is a prime number.
- Example here.

Chapter 4: Public Key

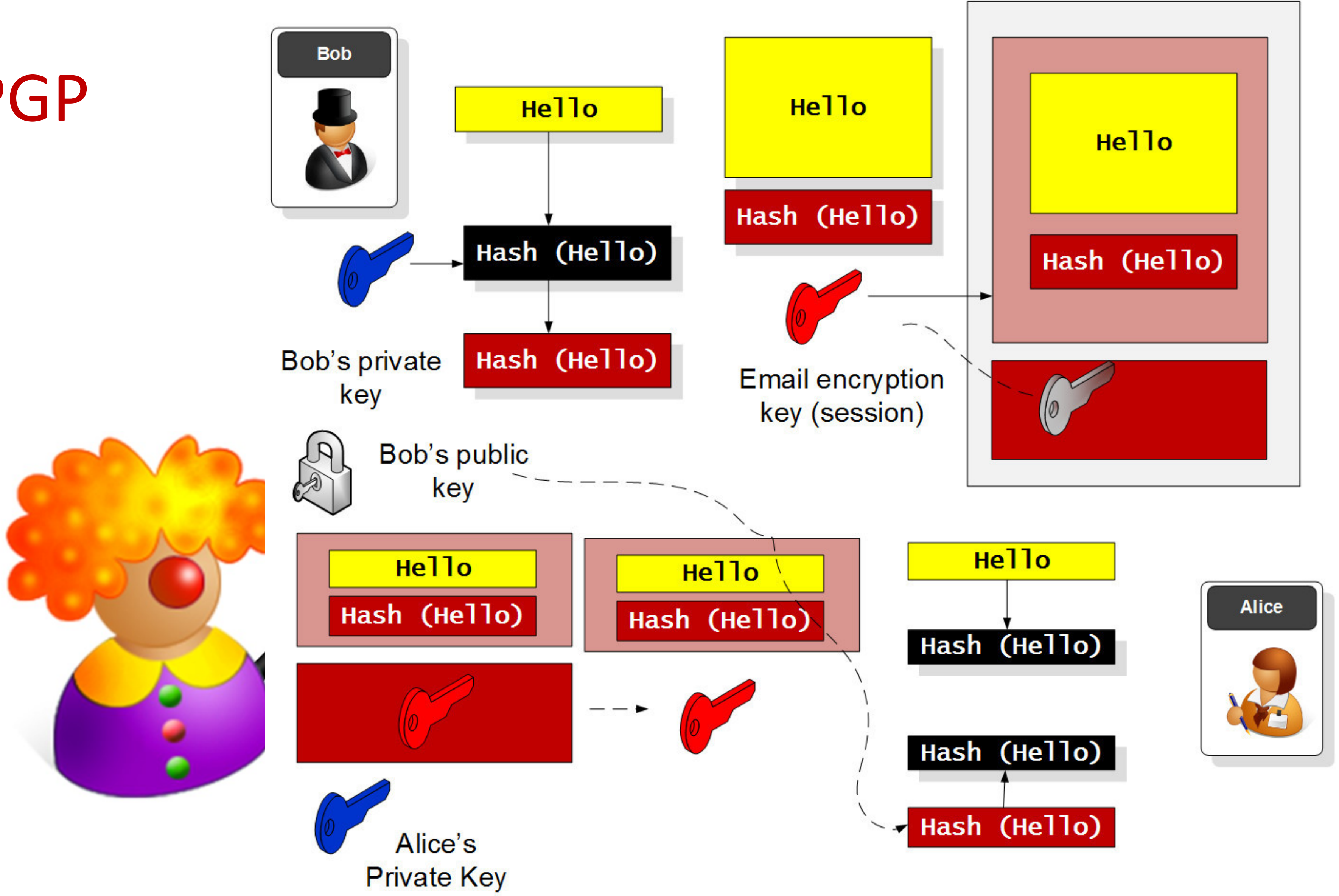
PGP

Prof Bill Buchanan OBE

http://asecuritysite.com/crypto04

http://asecuritysite.com/encryption

PGP


Bob
Hello
Hash (Hello)
Bob's private key
Hash (Hello)
Hello
Hash (Hello)
Email encryption key (session)
Hello
Hash (Hello)
Bob's public key
Hello
Hash (Hello)
Hello
Hash (Hello)
Hello
Hash (Hello)
Alice
Hash (Hello)
Hash (Hello)
Alice's Private Key

Chapter 4: Public Key

Basics
RSA
Elliptic Curve
ElGamal

Prof Bill Buchanan OBE

http://asecuritysite.com/crypto04
http://asecuritysite.com/encryption