Bachelor of Computer Science

SCS2214 - Information System Security

Handout 4 - Asymmetric Key Cryptography

Kasun de Zoysa
kasun@ucsc.cmb.ac.lk

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

Symmetric Key Cryptography

- Traditional **secret/single key** cryptography uses **one** key
- Shared by both sender and receiver
- If this key is disclosed, communications are compromised
- Also is **symmetric**, parties are equal
- Hence receiver can forge a message and claim it was sent by sender

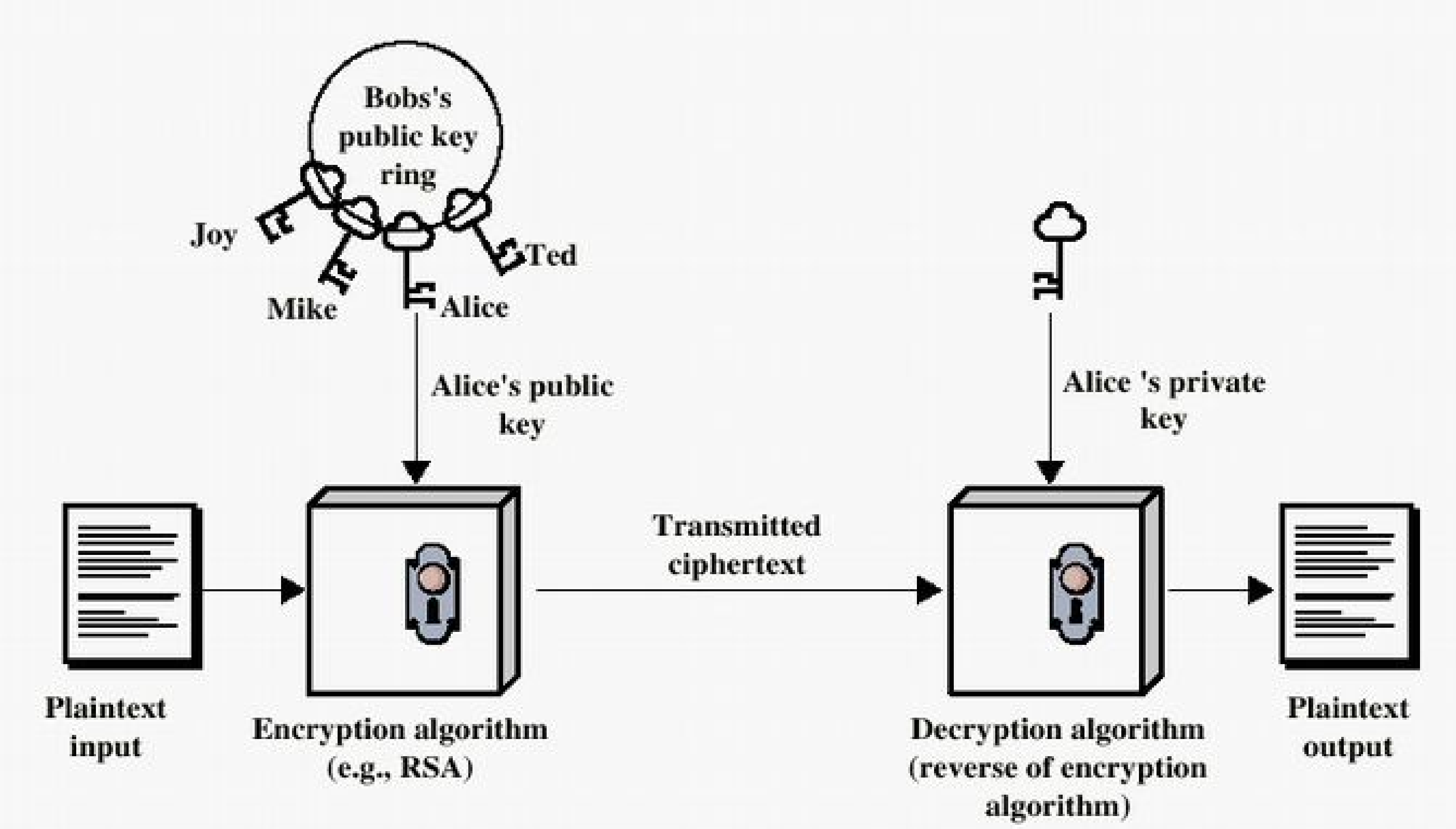
Why Public-Key Cryptography?

- Developed to address two issues:
  - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
- **digital signatures** – how to verify a message comes intact from the claimed sender
- Whitfield Diffie and Martin Hellman in 1976 known earlier in classified community

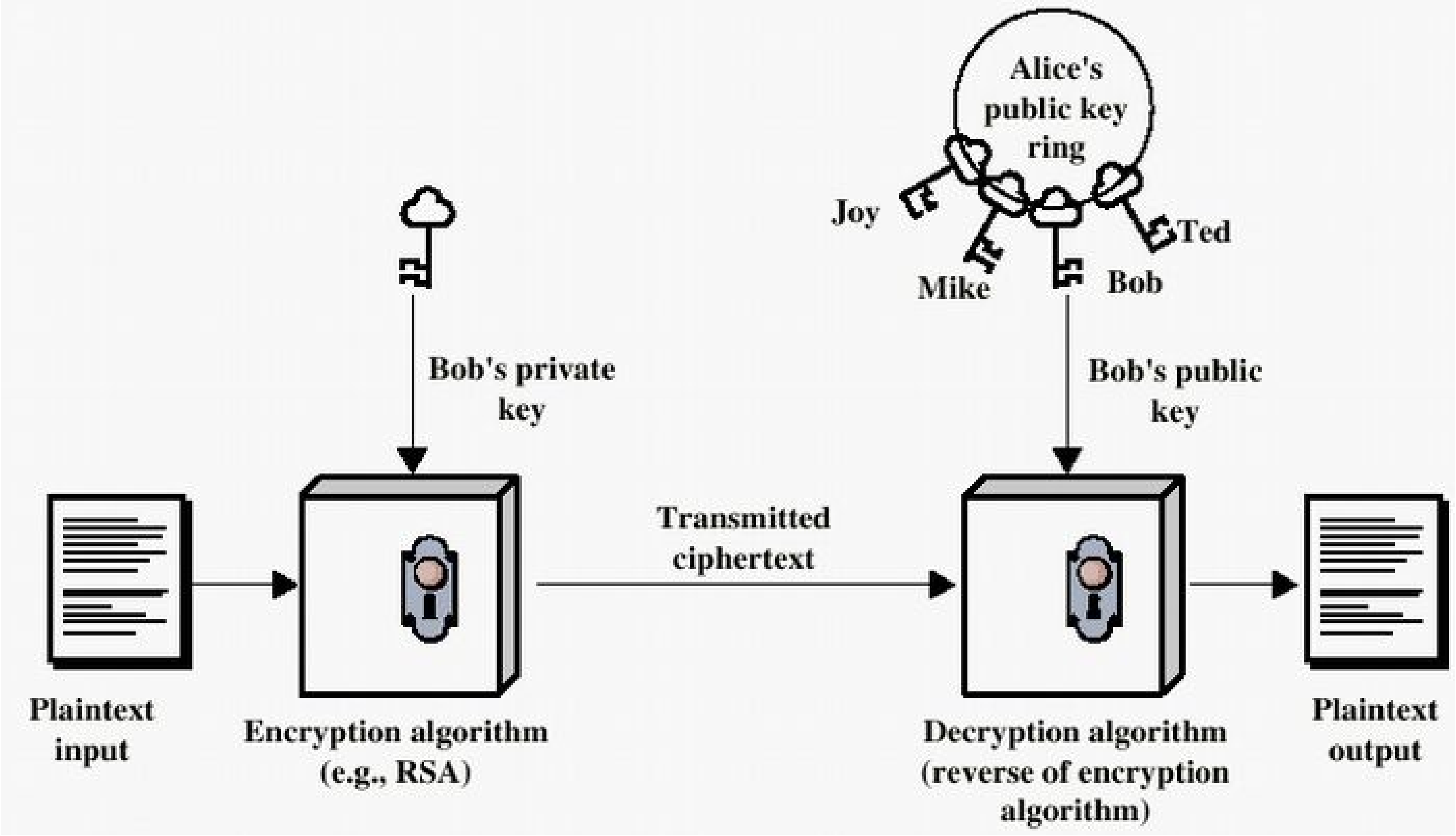
Public-Key Cryptography Principles

- The use of two keys has consequences in: key distribution, confidentiality and authentication.
- The scheme has six ingredients
  - Plaintext
  - Encryption algorithm
  - Public and private key
  - Ciphertext
  - Decryption algorithm

Encryption using Public-Key system



Authentication using Public-Key System



Applications for Public-Key Cryptosystems

- Three categories:
  - **Encryption/decryption:** The sender encrypts a message with the recipient's public key.
  - **Digital signature:** The sender ”signs” a message with its private key.
  - **Key exchange:** Two sides cooperate two exchange a session key.

Requirements for Public-Key Cryptography

- Computationally easy for a party B to generate a pair (public key KU_b, private key KR_b)
- Easy for sender to generate ciphertext:
- Easy for the receiver to decrypt ciphertect using private key:

$$C = E_{KUb}(M)$$

$$M = D_{KRb}(C) = D_{KRb}[E_{KUb}(M)]$$

Requirements for Public-Key Cryptography

- Computationally infeasible to determine private key (KR_b) knowing public key (KU_b)
- Computationally infeasible to recover message M, knowing KU_b and ciphertext C
- Either of the two keys can be used for encryption, with the other used for decryption:

$$M = D_{KRb}[E_{KUb}(M)] = D_{KUb}[E_{KRb}(M)]$$

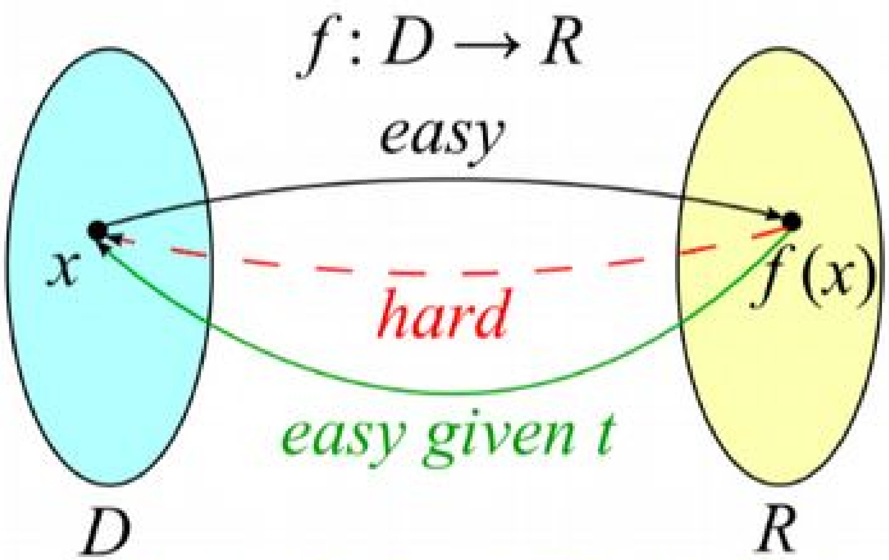
Public-Key Cryptographic Algorithms

- **RSA** - Ron Rives, Adi Shamir and Len Adleman at MIT, in 1977.
  - RSA is a block cipher
  - The most widely implemented
- **Diffie-Hellman**
  - Exchange a secret key securely
  - Compute discrete logarithms
- **Elliptic Curve Cryptography (ECC)**

Trapdoor Function

Trapdoor functions

- Easy to compute in one direction
- Difficult to compute in other direction (finding the inverse) but easy to compute, with some special information (trapdoor)



Source: https://en.wikipedia.org/wiki/Trapdoor_function

Discrete Logarithms

- The inverse problem to exponentiation is to find the **discrete logarithm** of a number modulo p
- That is, find x where ($a^x = b \bmod p$).
- This is also written as ($x = \log_a b \bmod p$). If a is a primitive root then the discrete logarithm always exists, otherwise it may not
  - $x = \log_3 4 \bmod 13$ (x st $3^x = 4 \bmod 13$) has no answer
  - $x = \log_2 3 \bmod 13 = 4$ by trying successive powers
- While exponentiation is relatively easy, finding discrete logarithms is generally a **hard** problem

Diffie-Hellman Key Agreement

- **Published in 1976**
- **Based on difficulty of calculating discrete logarithm in a finite field**

DH key pair generation

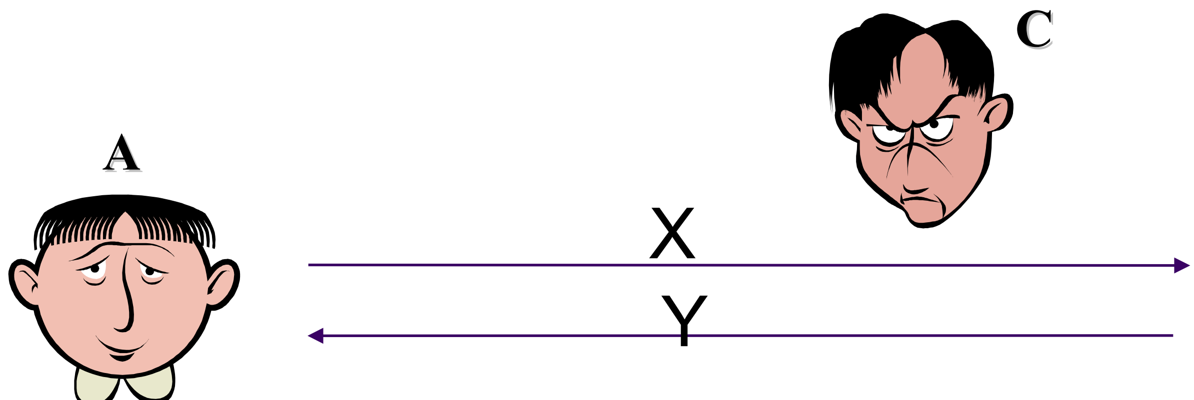
- G is finite group with generator g, p is a prime and q is a prime divisor of p-1.
- Randomly select x from [1, q-1]
- Compute $y=g^x \pmod{p}$

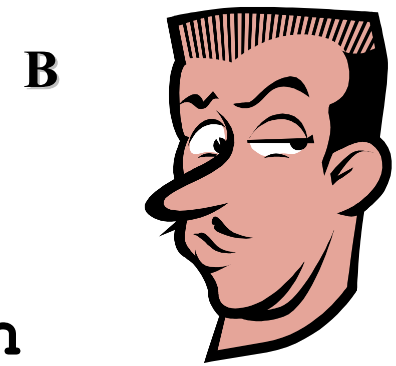
The public key is y, and private key is x.

Observation: $x=log_g\, y \pmod{p}$, x is called the discrete logarithm of y to the base g.

Given g,x, and p, it is trivial to calculate y. However, given y, g, and p it is difficult to calculate x.

Diffie-Hellman Key Agreement





$X = g^x \bmod n$

$k = Y^x \bmod n = g^{xy} \bmod n$

$Y = g^y \bmod n$

$k = X^y \bmod n = g^{xy} \bmod n$

Possible to do man in the middle attack

Prime Factorization

- An integer, *n > 1*, can be factored in a unique way as:

$$n = p_1^{a_1} . p_2^{a_2} \dots . p_t^{a_t}$$

where $p_1 < p_2 < \dots < p_t$ and a_i is a positive integer

- E.g. `91=7×13, 3600=2⁴×3²×5²`

Relatively Prime Numbers & GCD

- Two numbers *a* and *b* are **relatively prime** if they have **no common divisors** apart from 1
  - eg. 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor
- Can determine the greatest common divisor by comparing their prime factorizations and using least powers
  - E.g. $300=2^1\times3^1\times5^2$, $18=2^1\times3^2$ hence $GCD(18,300)=2^1\times3^1\times5^0=6$

Prime Numbers

- Prime numbers only have divisors of 1 and self they cannot be written as a product of other numbers
- E.g. 2,3,5,7 are prime, 4,6,8,9,10 are not
- Prime numbers are central to number theory

List of prime number less than 200 is:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191193 197 199

Greatest Common Divisor

Greatest Common Divisor - gcd(a,b)

- ***The largest integer that divides a set of numbers***
- ***If p is a prime, for any number q<p, gcd(p,q)=1***
- ***gcd(a,b)=gcd(b,a)***

Example : gcd(15,10)=5

Euclidean Algorithm

If x divides a and b, x also divides a-(k*b) for every k

Suppose x divides both a and b; then
a=x*a1; b=x*b1
a-(k*b)=x*a1 - (k*x*b1)
= x*(a1-k*b1)
= x*d
So that x divides (is a factor of) a-(k*b)

Suppose x=gcd(a,b), where a>b
a=m*b+r
a-(m*b)=r So that gcd(b,r)=x

gcd(a,b)=gcd(b,r) a>b>r>=0

Euclid's GCD Algorithm

An efficient way to find the GCD (a, b) uses theorem that:

GCD (a, b) = GCD (b, a mod b)

Euclid's Algorithm to compute GCD (a, b):

```
A=a; B=b;
while (B>0) {
R = A mod B;
A = B;
B = R;
}
return A;
```

Example: GCD(1970,1066)

1970 = 1 x 1066 + 904	gcd(1066, 904)
1066 = 1 x 904 + 162	gcd(904, 162)
904 = 5 x 162 + 94	gcd(162, 94)
162 = 1 x 94 + 68	gcd(94, 68)
94 = 1 x 68 + 26	gcd(68, 26)
68 = 2 x 26 + 16	gcd(26, 16)
26 = 1 x 16 + 10	gcd(16, 10)
16 = 1 x 10 + 6	gcd(10, 6)
10 = 1 x 6 + 4	gcd(6, 4)
6 = 1 x 4 + 2	gcd(4, 2)
4 = 2 x 2 + 0	gcd(2, 0)

Primality Testing

- In Cryptography, we often need to find large prime numbers
- Traditionally method using **trial division**
- i.e. divide by all numbers (primes) in turn less than the square root of the number
- only works for small numbers
- Alternatively can use statistical primality tests based on properties of primes
- for which all primes numbers satisfy property but some composite numbers, called pseudo-primes, also satisfy the property

How to find a large prime? (Solovay and Strassen)

1. If p is prime and r is any number less than p

gcd(p,r)=1 ; greatest common devisor

2. Jacobi function

$J(r,p) = 1$ *if* $r=1$

$J(r/2)*(-1)^{(p2-1)/8}$ *if r is even*

$J(p \bmod r, r)*(-1)^{(r-1)*(p-1)/4}$ *if r is odd and* $r = 1$

$J(r,p) \bmod p = r^{(p-1)/2}$

<u>Test :</u>

If test 1 and 2 passes probability(prime p) = 1/2.

Otherwise p should not be prime.

If test repeated k time probability(prime p) = $1/2^k$

RSA

The Association for Computing Machinery (ACM) has named RONALD L. RIVEST, ADI SHAMIR, and LEONARD M. ADLEMAN as winners of the 2002 A. M. Turing Award, considered the "Nobel Prize of Computing", for their contributions to public key cryptography. The Turing Award carries a $100,000 prize, with funding provided by Intel Corporation.

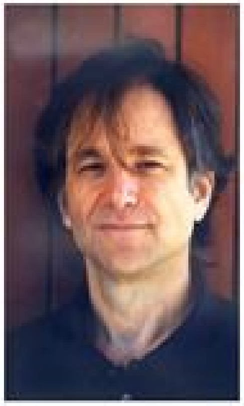
As researchers at the Massachusetts Institute of Technology in 1977, the team developed the RSA code, which has become the foundation for an entire generation of technology security products. It has also inspired important work in both theoretical computer science and mathematics. RSA is an algorithm—named for Rivest, Shamir, and Adleman—that uses number theory to provide a pragmatic approach to secure transactions. It is today's most widely used encryption method, with applications in Internet browsers and servers, electronic transactions in the credit card industry, and products providing email services.

- Excerpt from ACM news release on

2002 Turing award

Ron Rivest
born in 1947

Adi Shamir
born in 1952



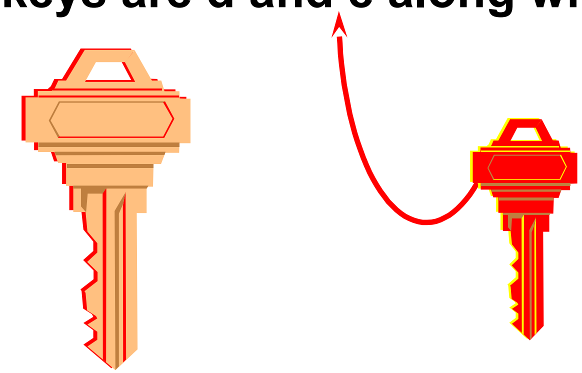
Leonard M. Adleman
born in 1945

Revest-Shamir-Adelman (RSA)

By Rivest, Shamir and Adelman in 1978

1. **Find 2 large prime numbers p and q (100 digits=512bits)**
2. **Calculate the product n=p*q (n is around 200 digits)**
3. **Select large integer e relatively prime to (p-1)(q-1)**
Relatively prime means e has no factors in common with (p-1)(q-1).
Easy way is select another prime that is larger than both(p-1) and (q-1).
4. **Select d such that e*d mod (p-1)*(q-1)=1**

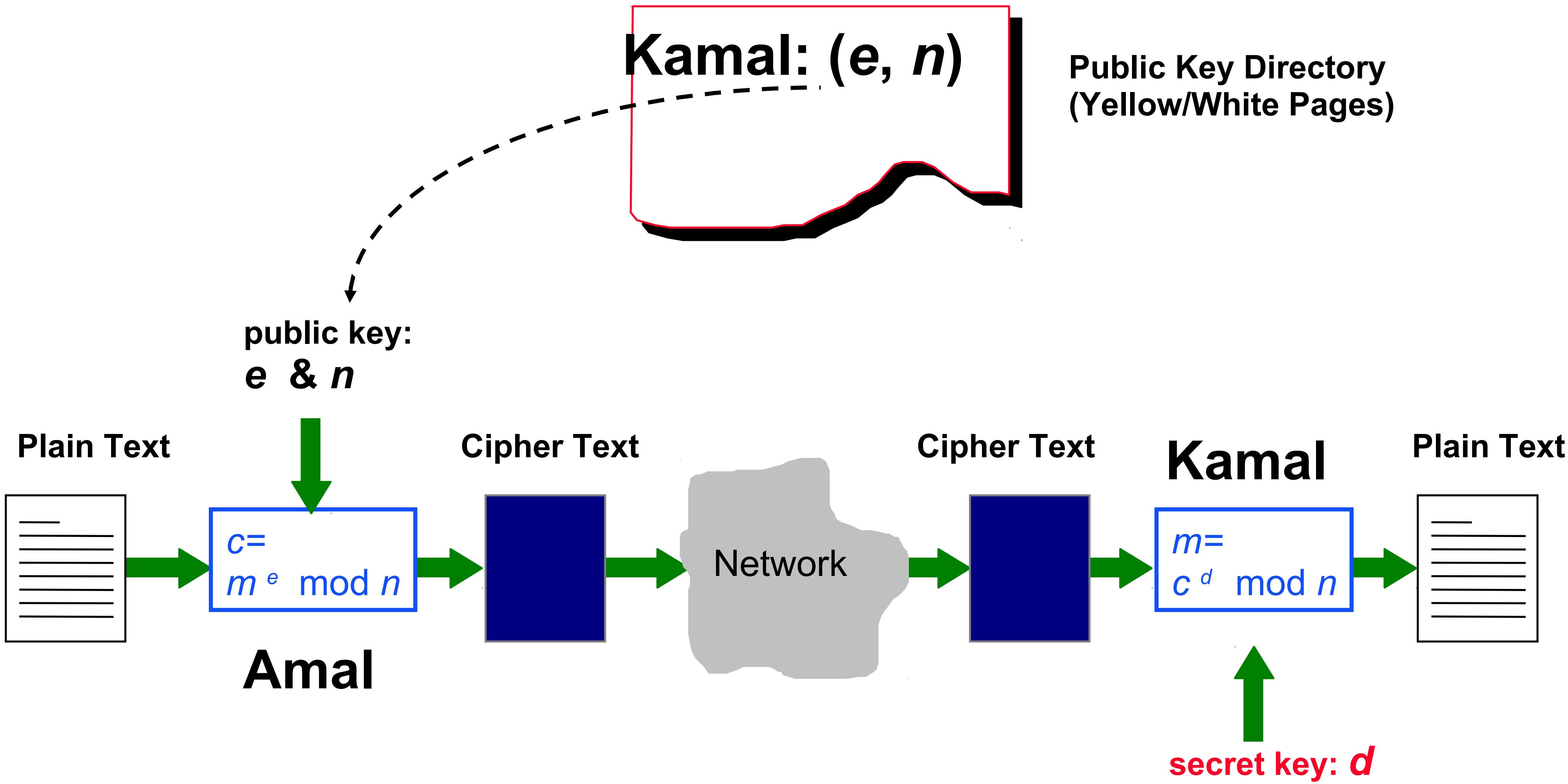
Two keys are d and e along with n



Encryption
$C=P^e \bmod n$

Decryption
$P=C^d \bmod n$

RSA Public Key Cryptosystem



RSA - Simple Example

1. Find 2 prime numbers p and q

Let p=11 and q=17

2. Calculate the product n=p*q

*n = 11*17=187*

3. Select large integer e relatively prime to (p-1)(q-1)

*E=7 ; 7 IS Relatively prime to (p-1)(q-1) =10*16=160*

4. Select d such that e*d mod (p-1)*(q-1)=1

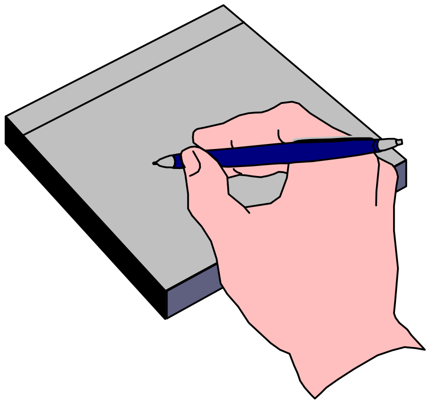
*d=23 because, 23*7 mod 10*16=161 mod 160 =1*

Encryption

$C=P^e$ mod n

Decryption

$P=C^d$ mod n



RSA - Simple Example

recipient knows:

- $PR=\{23,187\}$ // d=23, n=187
- $187=17\times11$ // p=17, q=11
- $\phi(n)=(p-1)(q-1)=160$ // check: $7\times23 \bmod 160=1$

sender knows:

- $PU=\{7,187\}$ // e=7, n=187
- plaintext to encrypt: $M=88$ // $88 < 187$

RSA - Simple Example

sender knows:

- $PU=\{7,187\}$
- plaintext to encrypt: $M=88$ // $88 < 187$

ciphertext

Encryption

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

RSA - Simple Example

recipient knows:

- PR={23,187}
- $187=17\times 11$ // p=17, q=11
- $\phi(n)=(p-1)(q-1)=160$ // check: $7\times 23 \bmod 160=1$
- receives cipher text: 11

Decryption

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$

$$11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$

$$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$$

$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$$

plaintext

RSA --- 2nd small example (1)

- Kamal:
  - chooses 2 primes: *p=5, q=11*
    multiplies p and q: *n = p*q = 55*
  - finds out two numbers *e=3* & *d=27* which satisfy
    (*3 * 27*) *mod 40 = 1*
  - Kamal's public key
    - 2 numbers: (*3, 55*)
    - encryption alg: modular exponentiation
  - secret key: (*27,55*)

RSA --- 2nd small example (2)

- Amal has a message *m=13* to be sent to Kamal:
  - finds out Kamal's public encryption key (*3, 55*)
  - calculates c:

$$\begin{aligned} c &= m^e \ (mod\ n) \\ &= 13^3 \ (mod\ 55) \\ &= 2197 \ (mod\ 55) \\ &= 52 \end{aligned}$$

  - sends the ciphertext *c=52* to Kamal

RSA --- 2nd small example (3)

- Kamal:
  - receives the ciphertext *c=52* from Amal
  - uses his matching secret decryption key *27* to calculate m:

$$m = 52^{27} \pmod{55}$$
$$= 13 \text{ (Amal's message)}$$

RSA --- 3rd small example

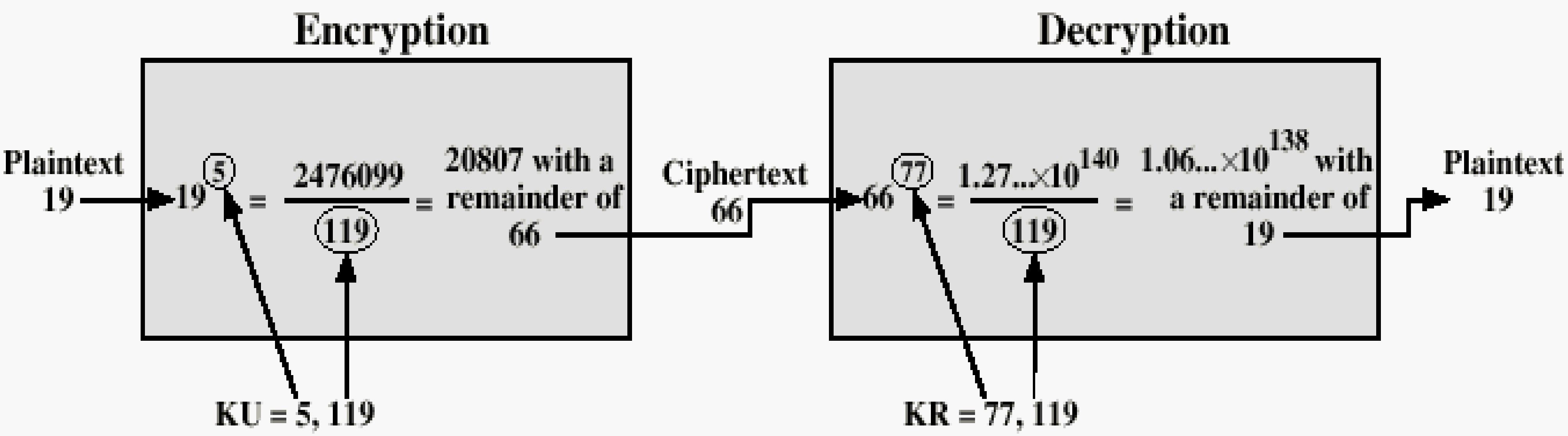


Figure 3.9 Example of RSA Algorithm

RSA Signature --- an eg (1)

- Kamal:
  - chooses 2 primes: $p=5, q=11$
    multiplies p and q: $n = p*q = 55$
  - finds out two numbers $e=3$ & $d=27$ which satisfy
    $(3 * 27) \bmod 40 = 1$
  - Kamal's public key
    - 2 numbers: $(3, 55)$
    - encryption algo: modular exponentiation
  - secret key: $(27,55)$

RSA Signature --- an eg (2)

- Kamal has a document $m=19$ to sign:
  - uses his secret key $d=27$ to calculate the digital signature of $m=19$:

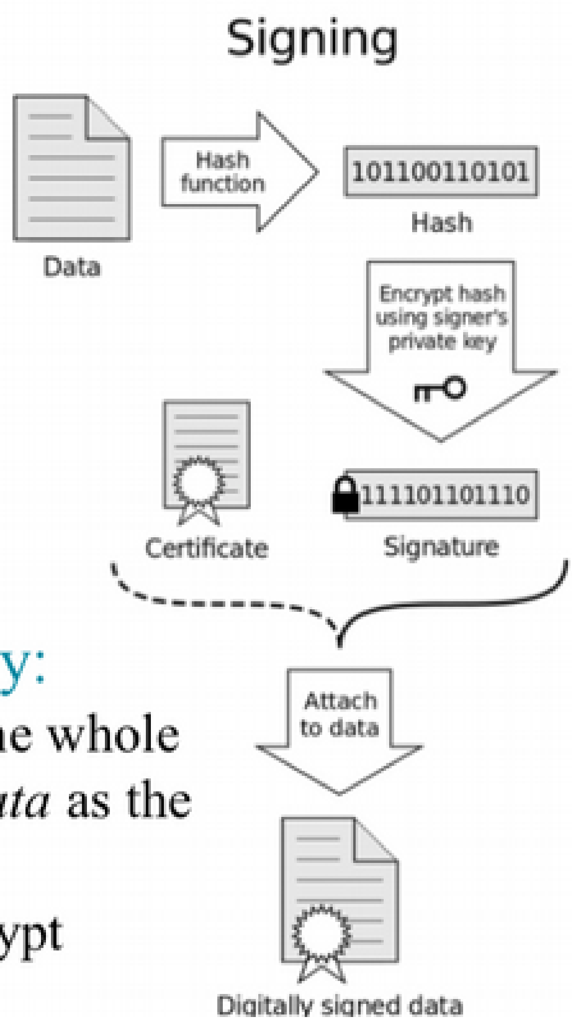
    $$\begin{aligned} s &= m^d \pmod n \\ &= 19^{27} \pmod{55} \\ &= 24 \end{aligned}$$

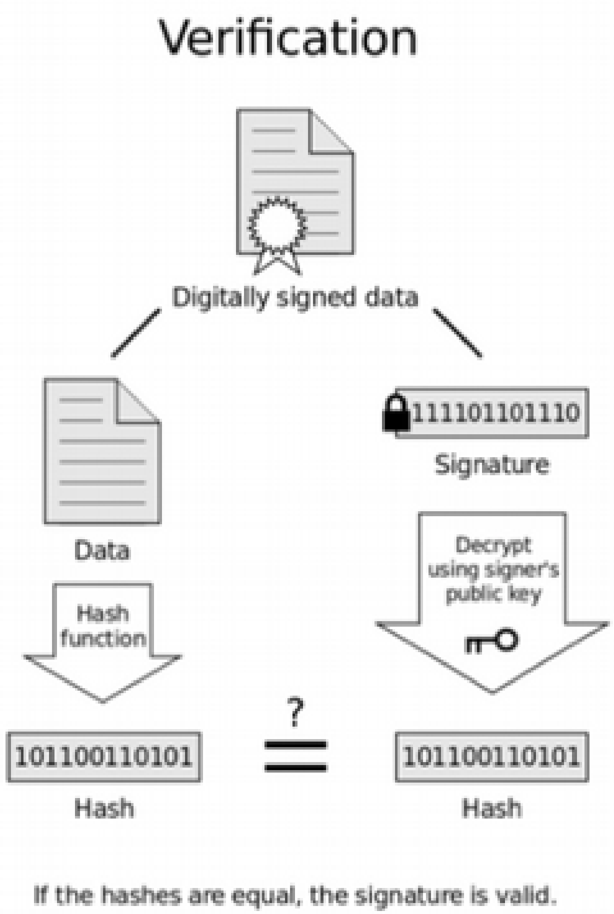
  - appends 24 to 19. Now $(m, s) = (19, 24)$ indicates that the doc is 19, and Kamal's signature on the doc is 24.

RSA Signature --- an eg. (3)

- Nimal, a verifier:
  - receives a pair *(m,s)=(19, 24)*
  - looks up the phone book and finds out Kamal's public key *(e, n)=(3, 55)*
  - calculates $t = s^e \pmod{n}$
    $= 24^3 \pmod{55}$
    $= 19$
  - checks whether $t=m$
  - confirms that (*19,24*) is a genuinely signed document of Kamal if $t=m$.

Typical Digital Signature





For confidentiality:

- Need to encrypt the whole *digitally signed data* as the plaintext.
- Four encrypt/decrypt operations!

Signature Creation

- **Generate Public/Private key pair**

```
  KeyPairGenerator keyGen=
KeyPairGenerator.getInstance(“DSA");
  keyGen.initialize(1024,new SecureRandom());
  KeyPair keyPair = keyGen.generateKeyPair();
```

- **Initialize the Signature object**

```
 Signature signature= Signature.getInstance("SHA1withDSA");
  signature.initSign(keyPair.getPrivate(),new SecureRandom());
```

- **Create the signature**

```
  signature.update(msg.getBytes());
  byte[]  sigBytes = signature.sign();
```

Signature Verification

- **Retrieves the Public key**

```
Let's say KeyPair object is keyPair
```

- **Initialize the Signature object**

```
Signature signature= Signature.getInstance("SHA1withDSA");
signature.initVerify(keyPair.getPublic());
```

- **Verify the signature**

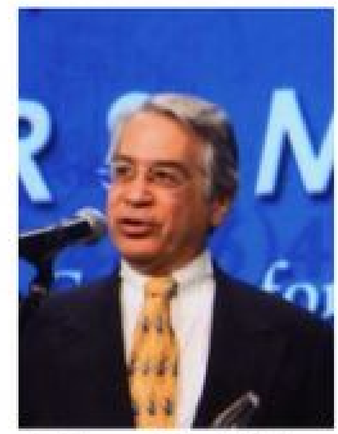
```
Let's say sigBytes contains the original signature

signature.update(msg.getBytes());
signature.verify(sigBytes)
```

Elliptic curve cryptography (ECC)

ECC invented (independently):

- 1985
- wide-scale adoption circa 2005
  barrier to adoption: patent/license protections

Neal Koblitz
born in 1948



Victor S. Miller
born in 1947

Elliptic curve cryptography (ECC)

Elliptic curves have been studied by mathematicians for over a hundred years. They have been deployed in diverse areas

- Number theory: proving Fermat`s Last Theorem in 1995 [4]
  - The equation $x^n + y^n = z^n$ has no nonzero integer solutions for x,y,z when the integer n is grater than 2.
- Modern physics: String theory
  - The notion of a point-like particle is replaced by a curve-like string.
- Elliptic Curve Cryptography
  - An efficient public key cryptographic system.

Elliptic curve cryptography (ECC)

Elliptic curves over real numbers

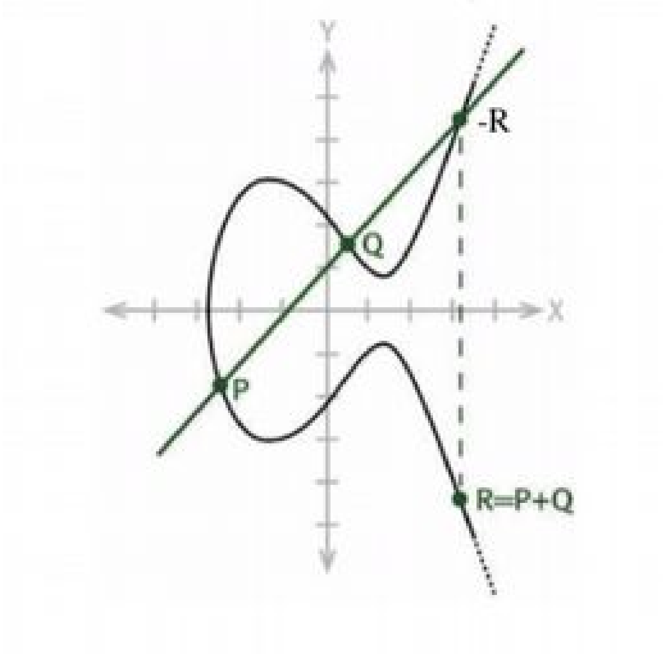
- Calculations prove to be slow
- Inaccurate due to rounding error
- Infinite field

Cryptographic schemes need fast and accurate arithmetic

- In the cryptographic schemes, elliptic curves over two finite fields are mostly used.
  - Prime field $\mathbb{F}_p$, where p is a prime.
  - Binary field $\mathbb{F}_{2^m}$, where m is a positive integer.

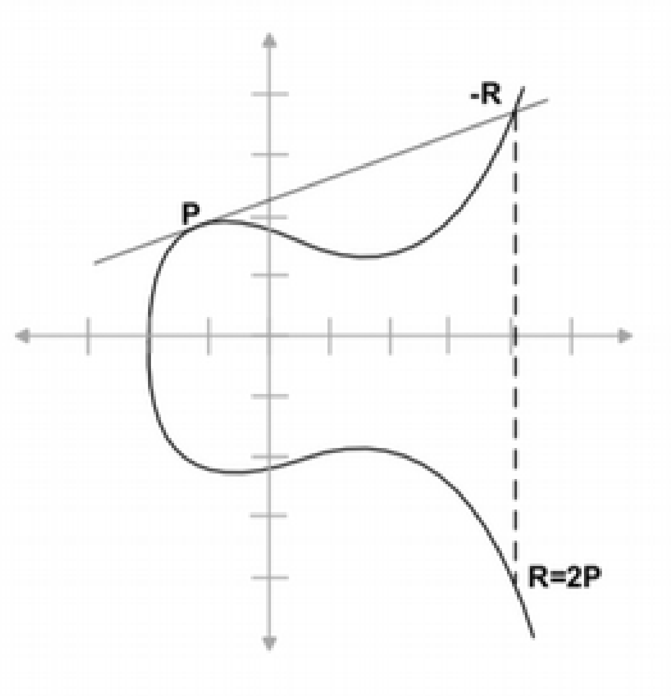
Point Addition

To add two distinct points P and Q on an elliptic curve, draw a straight line between them. The line will intersect the elliptic cure at exactly one more point –R. The reflection of the point –R with respect to x-axis gives the point R, which is the results of addition of points P and Q.



Point Doubling

To the point P on elliptic curve, draw the tangent line to the elliptic curve at P. The line intersects the elliptic cure at the point –R. The reflection of the point –R with respect to x-axis gives the point R, which is the results of doubling of point P.



ECC Cryptography

Elliptic curves are used to construct the public key cryptography system

The private key d is randomly selected from [1,n-1], where n is integer.

Then the public key Q is computed by dP, where P,Q are points on the elliptic curve.

Like the conventional cryptosystems, once the key pair (d, Q) is generated, a variety of cryptosystems such as signature, encryption/decryption, key management system can be set up.

Computing dP is denoted as scalar multiplication. It is not only used for the computation of the public key but also for the signature, encryption, and key agreement in the ECC system.

Scalar Multiplication

Intuitive approach:

$$dP = \underbrace{P+P+\ldots+P}_{d \text{ times}}$$

It requires d-1 times point addition over the elliptic curve.

Observation: To compute 17 P, we could start with 2P, double that, and that two more times, finally add P, i.e. 17P=2(2(2(2P)))+P. This needs only 4 point doublings and one point addition instead of 16 point additions in the intuitive approach. This is called Double-and-Add algorithm.

Elliptic Curve Digital Signature Algorithm (ECDSA)

Alice

Private key d_A, Public key $Q_A = d_A P$.

Signature generation

1.Select a random k from $[1, n-1]$

2.Compute $kP = (x_1, y_1)$ and $r = x_1 \bmod n$. if $r = 0$ goto step 1

3.Compute $e = H(m)$, where H is a hash functon, m is the message.

4.Compute $s = k^{-1}(e + d_A r) \bmod n$. If $s = 0$ go to step 1.

(r, s) is Alice's signature of message m

$\xrightarrow{m, (r, s)}$

Bob

Signature verification

1.Verify that r, s are in the interval $[1, n-1]$

2.Compute $e = H(m)$, where H is a hash functon, m is the message.

3.Compute $w = s^{-1} \bmod n$

4.Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.

5.Compute $X = u_1 P + u_2 Q_A = (x_1, y_1)$

6.Compute $v = x_1 \bmod n$

7.Accept the signature if and only if $v = r$

Elliptic Curve Deffie-Hellmen (ECDH)

Alice		**Bob**
Ephemeral key pair generation Select a private key $n_A \in [1, n\text{-}1]$ Calculate public key $Q_A = n_A P$	$\xrightarrow{Q_A}$ $\xleftarrow{Q_B}$	*Ephemeral key pair generation* Select a private key $n_B \in [1, n\text{-}1]$ Calculate public key $Q_B = n_B P$
Shared key computation $K = n_A Q_B$		*Shared key computation* $K = n_B Q_A$

- Consistency: $K = n_A Q_B = n_A n_B P = n_B Q_A$
- ECDH is vulnerable to the man-in-the-middle attack

http://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/

Key measure: Encryption strength

The mathematic background of ECC is more complex than other cryptographic systems

- Geometry, abstract algebra, number theory

ECC provides greater security and more efficient performance than the first generation public key techniques (RSA and Diffie-Hellman)

- Mobile systems
- Systems required high security level (such as 256 bit AES)

Bits of Security	Symmetric Key Algorithm	Corresponding RSA Key Size	Corresponding ECC Key Size
80	Triple DES (2 keys)	1024	160
112	Triple DES (3 keys)	2048	224
128	AES-128	3072	256
192	AES-192	7680	384
256	AES-256	15360	512

Factoring a product of two large primes

- **The best known conventional algorithm requires the solution time proportional to:**

$$T(n) = \exp\left[c(\ln n)^{1/3}(\ln\ln n)^{2/3}\right]$$

For p & q 65 digits long T(n) is approximately one month using cluster of workstations.

For p&q 200 digits long T(n) is astronomical.

Quantum Computing algorithm for factoring.

- **In 1994 Peter Shor from the AT&T Bell Laboratory showed that in principle a quantum computer could factor a very long product of primes in seconds.**
- **Shor's algorithm time computational complexity is**

$$T(n)=O\left[(\ln n)^3\right]$$

Once a quantum computer is built the RSA method would not be safe.

Discussion