# COM 5335 Network Security
# Lecture 6
# Public Key Cryptography & RSA

Scott CH Huang

# Outline

- One-way Trapdoor functions
- Basic Number Theory for RSA
- RSA Digital Signatures

# One-Way Trapdoor Functions

# One-Way Functions

- The most basic primitive for cryptosystem is a one-way function (OWF).
  - Informally, this is a function which is EASY to compute but HARD to invert.

# The Factorization Problem

- Factorization is a well-known candidate for OWF.
  - Randomly select two prime numbers: $p$ and $q$.
  - It is easy to compute $N=pq$.
  - However, conversely, given $N=pq$, it is *assumed* to be HARD to obtain $p$ or $q$.

# One-way Trapdoor Functions

- *A one-way trapdoor function f* is a one-way function with an extra property.
- There exists some secret information (called the *trapdoor*) that allows its possessor to *EFFICIENTLY* invert *f*.
- It is infeasible to invert *f* without knowledge of the trapdoor.

# Basic Number Theory for RSA

# Euler Totient Function


- Euler's Totient Function φ is defined by
  $\varphi(n) = \left|\{x | 1 \leq x \leq n, \gcd(x,n) = 1\}\right|$
- φ(2)=|{1}|=1
- φ(3)=|{1,2}|=2
- φ(4)={1,3}=2
- φ(5)=|{1,2,3,4}|=4
- φ(6)=|{1,5}|=2

# Calculation of Euler Totient Function

Properties

(1) For any prime $p$, and $\alpha \geq 1 \Rightarrow \varphi(p^{\alpha}) = p^{\alpha-1}(p-1)$

(2) $\forall m, n \in \mathbb{Z}$ s.t. $\gcd(m,n) = 1 \Rightarrow \varphi(mn) = \varphi(m)\varphi(n)$

Corollary: $\varphi(pq) = (p-1)(q-1)$ for *p, q* primes

# The Group $Z_n^*$

- $\mathbb{Z}_n^* = \{k | \gcd(k, n) = 1, 1 \leq k < n\}$
- For any positive integer *n*, $\mathbb{Z}_n^*$ forms a group under multiplication modulo *n*.
- Euler's Theorem:

$$\forall \alpha \in \mathbb{Z}_n^*, \text{ we have } \alpha^{\varphi(n)} \equiv 1 \pmod{n}$$

# Examples of $Z_n^*$

- $Z_{15}$={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14}
- $Z_{15}^*$={1,2,4,7,8,11,13,14}

  $\left(\varphi(15) = 8\right)\ \forall \alpha \in \mathbb{Z}_{15}^*,\ \text{we have } \alpha^8 \equiv 1 \pmod{15}$
- $Z_{12}$={0,1,2,3,4,5,6,7,8,9,10,11}
- $Z_{12}^*$={1,5,7,11}

  $\left(\varphi(12) = 4\right)\ \forall \alpha \in \mathbb{Z}_{12}^*,\ \text{we have } \alpha^4 \equiv 1 \pmod{12}$

# RSA

- In 1977 Rivest, Shamir and Adelman proposed the first candidate trapdoor function,
  - Now called the RSA. The story of modern cryptography followed.
  - The best known & widely used public-key scheme
- It is based on exponentiation in a finite group $\mathbb{Z}_n^*$ over integers modulo a number
  - exponentiation takes $O(\log^3 n)$ operations (easy)
- It uses large integers (eg. 1024 bits)
- The security relies on difficulty of factoring large numbers
  - factorization takes $O\left(e^{\sqrt{\log n \log\log n}}\right)$ operations (hard)

# RSA Key Setup

- Each user generates a public/private key pair by:
  - Selecting two large primes at random: *p, q*
  - Computing their system modulus *N=pq*
    - note $\varphi(N) = (p-1)(q-1)$
  - Selecting at random the encryption key e
    - where $1 < e < \varphi(N),\ \gcd\big(e, \varphi(N)\big) = 1$
  - Solve following equation to find decryption key d
    - $ed \equiv 1 \pmod{\varphi(N)},\ \text{and } 0 \leq d \leq N$
    - Fast to do it using Euclid's Algorithm.
  - publish their public encryption key: $P_u$ ={e,N}
  - keep secret private decryption key: $S_u$ ={d,p,q}

# RSA Encryption/Decryption

- Encrypt a message M by the sender:
  - obtains **public key** of recipient $P_u=\{e,N\}$
  - computes: $C=M^e \bmod N$, where $0 \le M < N$
- Decrypt the ciphertext C by the owner u:
  - use its private key $S_u=\{d,p,q\}$
  - compute: $M=C^d \bmod N$
- note that the message M must be smaller than the modulus N (block if needed)

# Why RSA Works

- By Euler's Theorem:
  - $\alpha^{\varphi(N)} \equiv 1 \pmod{N}$
  - where $\gcd(\alpha, N) = 1$
- In RSA, we have:
  - *N=pq*
  - *φ(N)=(p-1)(q-1)*
  - carefully chosen *e* & *d* to be inverses mod *φ(N)*
  - hence *ed=1+kφ(N)* for some *k*
- Hence (if *M* is relatively prime to *N*):

$$C^d = (M^e)^d = M^{1+k\varphi(N)} = M(M^{\varphi(N)})^k = M^1(1)^k \equiv M \pmod{N}$$

# Corollary of Euler's theorem

- Given two prime numbers p and q, and integers n = pq and m, with 0<m<n, the following relationship holds:

$$m^{\varphi(n)+1} \equiv m \pmod{n} \qquad \text{(Eq. 8.5)}$$

- Proof: When gcd(m,n)≠1, and m is a multiply of p
  → m = cp, gcd(m,q) = 1 since m < pq
  → $m^{\varphi(q)} \equiv 1 \pmod{q}$
  → $[m^{\varphi(q)}]^{\varphi(p)} \equiv 1 \pmod{q}$
  → $m^{\varphi(n)} \equiv 1 \pmod{q}$ implies that $m^{\varphi(n)} = 1 + kq$
  → $m^{\varphi(n)+1} = m + kcpq = m + kcn$ (multiply m = cp in both side)
  → $m^{\varphi(n)+1} = m \pmod{n}$

# Exponentiation

- A useful operation for PKC:
  - Given $a$, $n$, $m$, where $a \in Z_n$ and $m$ is an integer,
  - computes $a^m$ mod $n$.
- By repeated squaring, $a^m$ mod $n$ can be computed in O(log m) multiplications in mod n, hence $O(\log^3 n)$ time, if m<n.

# RSA Example

1. Select primes: $p$=17 & $q$=11
2. Compute $n = pq$ =17×11=187
3. Compute $\varphi(n)=(p-1)(q-1)=16\times10=160$
4. Select e *:* gcd(e,160)=1; choose $e$=7
5. Determine d*:* $de$=1 mod 160 and $d < 160$ Value is d=23 since 23×7=161= 10×160+1
6. Publish public key P={7,187}
7. Keep secret private key S={23,17,11}

# RSA Example cont.

- sample RSA encryption/decryption is:
- given message M = 88
- Encryption (using public key):

  $C = 88^7 \bmod 187 = 11$
- Decryption (using private key):

  $M = 11^{23} \bmod 187 = 88$

# Exponentiation

- Use the Square and Multiply Algorithm
  - a fast, efficient algorithm for exponentiation
- Concept is based on repeatedly squaring base
- and multiplying in the ones that are needed to compute the result
- look at binary representation of exponent
- only takes $O(\log_2 n)$ multiples for number n
  - eg. $7^5 = 7^4.7^1 = 3.7 = 10 \bmod 11$
  - eg. $3^{129} = 3^{128}.3^1 = 5.3 = 4 \bmod 11$

# Exponentiation

```
c ← 0; d ← 1
for i ← k downto 0
        do   c ← 2 × c
             d ← (d × d) mod n
             if    b_i = 1
                   then    c ← c + 1
                           d ← (d × a) mod n
return d
```

Equivalently, the algorithm looks at binary expansion of m. What we did is collect all the powers of two corresponding to the ones and multiply them.

For example: compute $2^{21}$ mod 22.

21='10101'

| 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| $a^{16}$ | $a^{8}$ | $a^{4}$ | $a^{2}$ | $a^{1}$ |
| 1 | 0 | 1 | 0 | 1 |

$2^1=2 \pmod{22}$ $2^2=4 \pmod{22}$ $2^4=16 \pmod{22}$

$2^8=16*16=256=220+36=36 \pmod{22}=14 \pmod{22}$

$2^{16}=14*14=196=22*8+20=20 \pmod{22}$

Therefore,

$2^{21}=2^{16}*2^4*2^1=20*16*2=20*32=$

$=20*10 \pmod{22}=200 \pmod{22}=22*9+2=2 \pmod{22}$.

# Some Remarks on RSA

# The Hardness to Invert RSA

- Thus far, the best way known to invert RSA is to first factor *n*.
- The best running time for a fully proved algorithm is Dixon's random squares algorithms which runs in time: $O\left(e^{\sqrt{\log n \log\log n}}\right)$
- But, in practice we may consider others.

- Let $l = |p|$ where $p$ is the smallest prime divisor of $n$.

The Elliptic Curve algorithm takes *expected time*

$$O\left(e^{\sqrt{2 \log l \log \log l}}\right)$$

- The Quadratic Sieve algorithm runs in *expected time*:

$$O\left(e^{\sqrt{\log n \log \log n}}\right)$$

- The recommended size for $n$ these days is 1024 bits.

# Knowledge of $\varphi(n)$ is equivalent to knowledge of the factorization

$\varphi(n)$ ⟵ factorization

To compute $\varphi(n)$ from p and q:

$\varphi(n) = (p-1)(q-1) = n+1-(p+q)$.

$\varphi(n)$ ⟶ factorization

To compute out $p$ and $q$ from $\varphi(n)$.

*Since* $pq=n$ and $p+q=n+1-\varphi(n)$.

Define $2b = n+1-\varphi(n)$ since $\varphi(n)$ is even.

p and q must be the root of equation $b \pm \sqrt{b^2 - n}$

$x^2-2bx+n=0$. Thus $p$ and $q$ equal to

# RSA Key Generation Remarks

- Users of RSA must:
  - determine two primes at random  p, q
  - select either e or d and compute the other
- Primes p,q must not be easily derived from modulus N=p.q
  - means must be sufficiently large
  - typically guess and use probabilistic test
- Exponents e, d  are inverses, so use Inverse algorithm to compute the other

# RSA Security

- three approaches to attacking RSA:
  - brute force key search (infeasible given size of numbers)
  - mathematical attacks (based on difficulty of computing $\varphi(N)$, by factoring modulus N)
  - timing attacks (on running of decryption)

# Factoring Problem

- To attack RSA, we can do either of the followings.
  1. factor N=p.q, hence find $\varphi(N)$ and then d
  2. determine $\varphi(N)$ directly and find d
  3. find d directly
- If we can crack factoring => we can crack RSA, but not vice versa (i.e. if we crack RSA we may not be able to do factoring).
- Currently we believed RSA is equivalent to factoring
  - have seen slow improvements over the years
    - as of Aug-99 best is 130 decimal digits (512) bit with GNFS
  - biggest improvement comes from improved algorithm
    - cf "Quadratic Sieve" to "Generalized Number Field Sieve"
  - barring dramatic breakthrough 1024+ bit RSA secure
    - ensure p, q of similar size and matching other constraints

# How to choose *p* and *q*

(1). The two primes should not be too close to each other (e. g. one should be a few decimal digits longer than the other).

Also, any one of *p* and *q* should not be too small due to the **Elliptic Curve** algorithm

Reason: $n = pq \Rightarrow n = \left((p+q)/2\right)^2 - \left((p-q)/2\right)^2 = t^2 - s^2$

Since *p* and *q* are close together we get: *s* is small and *t* is an integer only slightly larger than $\sqrt{n}$ . If you test the successive integers $t > \sqrt{n}$ you will *soon* find one such that $n = t^2 - s^2$, at which point you have $p = t + s$ and $q = t - s$.

(2). $p$-1 and $q$-1 should have a fairly small g.c.d. and both have at least one large prime factor.

(3). Of course, if someone discovers a factorization method that works quickly under certain other conditions on $p$ and $q$, then further users of RSA would have to take care to avoid those conditions as well.

# Summary

We have covered:

- The principles of public-key cryptography
- RSA algorithm, implementation, security