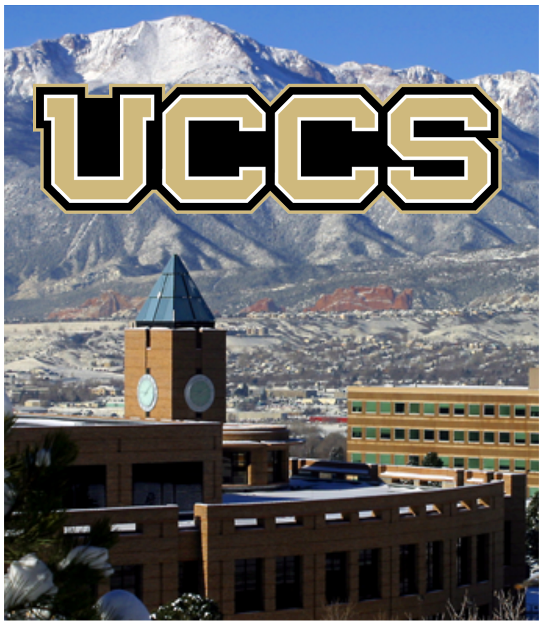

Asymmetric Cryptography
and Key Management

RSA Algorithm

Sang-Yoon Chang, Ph.D.

Module: RSA Algorithm

Prime factorization problem

RSA encryption and decryption

RSA key setup

RSA security

Primer Factorization Problem

Integer factorization:

$p \cdot q \leftarrow n$

p and *q* are prime numbers (a number that is only divisible by one and itself)

Primer Factorization Assumption

$n \leftarrow p \cdot q$ is easy for large n

$p \cdot q \leftarrow n$ is difficult for large n

Primer Factorization Assumption

n ← $p \cdot q$ is easy for large *n*
$p \cdot q$ ← *n* is difficult for large *n*

In RSA, derive public key *e* and
private key *d* from *p*, *q*

Use *e*, *d* and *n* for encryption (*m* → *c*)
and decryption (*c* → *m*)

Primer Factorization Assumption

$n \leftarrow p \cdot q$ is easy for large n

$p \cdot q \leftarrow n$ is difficult for large n

In RSA, derive public key e and
private key d from p, q

Use e, d and n for encryption ($m \rightarrow c$)
and decryption ($c \rightarrow m$)

Primer Factorization Assumption

$n \leftarrow p \cdot q$ is easy for large n
$p \cdot q \leftarrow n$ is difficult for large n

(2) In RSA, derive public key e and private key d from p, q

(1) Use e, d and n for encryption ($m \rightarrow c$) and decryption ($c \rightarrow m$)

RSA Algorithm

By Rivest, Shamir, and Adleman in 1976

Keys are typically 1024-4096 bit long

Security is based on the difficulty of finding p and q of a large n

RSA Encryption and Decryption

To encrypt a message m, the sender:

- obtains the recipient's public key $\{e,n\}$
- computes $c = m^e \bmod n$, where $0 \leq m < n$

RSA Encryption and Decryption

To encrypt a message m, the sender:

- obtains the recipient's public key $\{e,n\}$
- computes $c = m^e \bmod n$, where $0 \leq m < n$

To decrypt the c, the receiver

- uses the recipient's private key $\{d,n\}$
- computes $m = c^d \bmod n$

RSA Encryption and Decryption

To encrypt a message m, the sender:

- obtains the recipient's public key $\{e,n\}$
- computes $c = m^e \bmod n$, where $0 \leq m < n$

To decrypt the c, the receiver

- uses the recipient's private key $\{d,n\}$
- computes $m = c^d \bmod n$

RSA Decryption

The sender computes: $c = m^e \bmod n$

From c, the recipient computes:

$m \quad = c^d \bmod n$

$\quad\quad = (m^e \bmod n)^{\mathrm{d}} \bmod \mathrm{n}$

RSA Decryption

The sender computes: $c = m^e \bmod n$

From c, the recipient computes:

$$
\begin{aligned}
m &= c^d \bmod n \\
&= (m^e \bmod n)^d \bmod n \\
&= (m^e)^d \bmod n \\
&= m^{ed} \bmod n
\end{aligned}
$$

RSA Decryption

The sender computes: $c = m^e \bmod n$

From c, the recipient computes:

$$
\begin{aligned}
m \quad &= c^d \bmod n \\
&= (m^e \bmod n)^{\mathrm{d}} \bmod \mathrm{n} \\
&= (m^e)^d \bmod n \\
&= m^{ed} \bmod n \qquad\qquad = m
\end{aligned}
$$

RSA Decryption

The sender computes: $c = m^e \bmod n$

From c, the recipient computes:

$$
\begin{aligned}
m \quad &= c^d \bmod n \\
&= (m^e \bmod n)^{\mathrm{d}} \bmod \mathrm{n} \\
&= (m^e)^d \bmod n \\
&= m^{ed} \bmod n \qquad\qquad = m
\end{aligned}
$$

This holds for carefully chosen e and d!

RSA Public Key (*e*) and Private Key (*d*)

Each user selects two large primes (p, q)

Compute $n = p \cdot q$ ➔ $ø(n)=(p-1)(q-1)$

// Euler Totient Function

RSA Public Key (*e*) and Private Key (*d*)

Each user selects two large primes (p, q)

Compute $n = p \cdot q$ ➔ $\phi(n)=(p-1)(q-1)$
// Euler Totient Function

Select random e where
$1<e<\phi(n)$, $\gcd(e \cdot \phi(n))=1$

RSA Public Key (e) and Private Key (d)

Each user selects two large primes (p, q)

Compute $n = p \cdot q$ ➔ $\phi(n)=(p-1)(q-1)$

// Euler Totient Function

Select random e where
$1<e<\phi(n)$, $\gcd(e \cdot \phi(n))=1$

Solve d where $e \cdot d \equiv 1 \bmod \phi(n)$, $0 \leq d \leq n$

// Extended Euclidean algorithm

RSA Public Key (*e*) and Private Key (*d*)

Each user selects two large primes (p, q)

Compute $n = p \cdot q$ ➔ $\phi(n)=(p-1)(q-1)$

// Euler Totient Function

Select random e where
$1<e<\phi(n)$, $\gcd(e \cdot \phi(n))=1$

Solve d where $e \cdot d \equiv 1 \bmod \phi(n)$, $0 \le d \le n$

// Extended Euclidean algorithm

RSA Public Key (*e*) and Private Key (*d*)

Select p=11, q=13

Each user selects two large primes (p, q)

Compute $n = p \cdot q$ ➔ $\phi(n)=(p-1)(q-1)$

// Euler Totient Function

Select random e where

$1<e<\phi(n)$, $\gcd(e \cdot \phi(n))=1$

Solve d where $e \cdot d \equiv 1 \bmod \phi(n)$, $0 \le d \le n$

// Extended Euclidean algorithm

RSA Public Key (*e*) and Private Key (*d*)

Select p=11, q=13

Each user selects two large primes (p, q)

Compute $n = p \cdot q$ ➔ $\phi(n)=(p-1)(q-1)$

Compute n=143 ➔ $\phi(n)=10 \cdot 12=120$

Select random e where

$1<e<\phi(n)$, $\gcd(e \cdot \phi(n))=1$

Solve d where $e \cdot d \equiv 1 \bmod \phi(n)$, $0 \le d \le n$

// Extended Euclidean algorithm

RSA Public Key (*e*) and Private Key (*d*)

Select p=11, q=13

Each user selects two large primes (p, q)

Compute $n = p \cdot q$ ➔ $ø(n)=(p-1)(q-1)$

Compute n=143 ➔ $ø(\mathrm{n})=10 \cdot 12=120$

Select random e where

$1<e<ø(n)$, $\gcd(e \cdot ø(n))=1$ Select e=11

Solve d where $e \cdot d \equiv 1 \bmod ø(n)$, $0 \le d \le n$

// Extended Euclidean algorithm

RSA Public Key (*e*) and Private Key (*d*)

Select p=11, q=13

Each user selects two large primes (p, q)

Compute $n = p \cdot q$ ➔ $\phi(n)=(p-1)(q-1)$

Compute n=143 ➔ $\phi(\mathrm{n})=10 \cdot 12=120$

Select random e where
$1<e<\phi(n)$, $\gcd(e \cdot \phi(n))=1$ Select e=11

Solve d where $e \cdot d \equiv 1 \bmod \phi(n)$, $0 \leq d \leq n$

Compute d=11 ➔ $11 \cdot 11 \equiv 1 \bmod 120$

RSA Public Key (*e*) and Private Key (*d*)

Select p=11, q=13

Each user selects two large primes (p, q)

Compute $n = p \cdot q$ ➔ $ø(n)=(p-1)(q-1)$

Compute n=143 ➔ ø(n)=10·12=120

Select random e where
$1<e<ø(n)$, $\gcd(e \cdot ø(n))=1$ Select e=11

Solve d where $e \cdot d \equiv 1 \bmod ø(n)$, $0 \le d \le n$

Compute d=11 ➔ $11 \cdot 11 \equiv 1 \bmod 120$

Encryption: $c = m^e \bmod n$

RSA Public Key (*e*) and Private Key (*d*)

Select p=11, q=13

Each user selects two large primes (p, q)

Compute $n = p \cdot q$ ➔ $ø(n)=(p-1)(q-1)$

Compute n=143 ➔ ø(n)=10·12=120

Select random e where
$1<e<ø(n)$, $\gcd(e \cdot ø(n))=1$ Select e=11

Solve d where $e \cdot d \equiv 1 \bmod ø(n)$, $0 \leq d \leq n$

Compute d=11 ➔ 11·11 ≡ 1 mod 120

Encryption: $c = 7^e \bmod n = 106$

RSA Public Key (*e*) and Private Key (*d*)

Select $p=11$, $q=13$

Each user selects two large primes (p, q)

Compute $n = p \cdot q$ ➔ $ø(n)=(p-1)(q-1)$

Compute $n=143$ ➔ $ø(\mathrm{n})=10 \cdot 12=120$

Select random e where
$1<e<ø(n)$, $\gcd(e \cdot ø(n))=1$ Select $e=11$

Solve d where $e \cdot d \equiv 1 \bmod ø(n)$, $0 \leq d \leq n$

Compute $d=11$ ➔ $11 \cdot 11 \equiv 1 \bmod 120$

Encryption: $c = 7^e \bmod n = 106$

Decryption: $m = c^d \bmod n = 7$

RSA Key Setup and Encryption

p, q are used for e, d generation

p, q must not be easily derived from n

Select either e or d and compute the other (mod $\phi(n)$)

$\gcd(e, \phi(n))=1$ and $e \cdot d \equiv 1 \bmod \phi(n)$

The encryption/decryption computes exponentiation over mod n

RSA Security

Brute force key search

Prime factorization assumption

Timing-based side channel attack

Chosen ciphertext attack

Prime Factorization Problem

“Factoring could turn out to be easy”

- Rivest

RSA factoring challenge, 1991-2007

Timing Side-Channel Attacks

Paul Kocher in mid 1990’s

Infer operand size based on operation duration (higher exponent takes longer)

Countermeasures based on obfuscating operation duration

Chosen Ciphertext Attacks

Attackers choose ciphertexts and get the decrypted plaintext back

Vulnerability from being multiplicative:
$\mathrm{Enc}(m_1)\cdot\mathrm{Enc}(m_2) = \mathrm{Enc}(m_1\cdot m_2)$

Attacker wants to know m from c

Chooses $c' = c\cdot r^e \pmod{n}$ for some r

➔ $m' = m\cdot r \pmod{n}$

Chosen Ciphertext Attacks

Attackers choose ciphertexts and get the decrypted plaintext back

Vulnerability from being multiplicative:
$\mathrm{Enc}(m_1) \cdot \mathrm{Enc}(m_2) = \mathrm{Enc}(m_1 \cdot m_2)$

Attacker wants to know m from c

Chooses $c' = c \cdot r^e \pmod{n}$ for some r

➔ $m' = m \cdot r \pmod{n}$

Counter with random pad of plaintext, e.g., OAEP