RSA Algorithm

An, Trevor, and Trinity

Situation

Alice wants to send a message to Bob

$$\text{Alice} \Rightarrow m \Rightarrow \text{Bob}$$

Situation

Alice wants to send a message to Bob

$$\text{Alice} \Rightarrow m \Rightarrow \text{Bob}$$

Alice wants to **send** this message **securely** such that **only Bob** can interpret it.

Situation

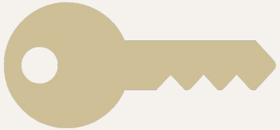
Alice wants to send a message to Bob

$$\text{Alice} \Rightarrow m \Rightarrow \text{Bob}$$

Alice wants to **send** this message **securely** such that **only Bob** can interpret it.

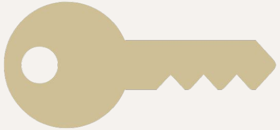
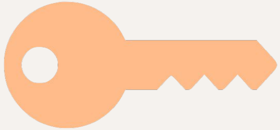
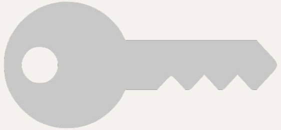
How? **Encryption!**

Types of encryption?

Symmetric	
● Each party has same key k for encryption ○ Alice ⇒ $\varepsilon(k, m) = c$ ⇒ Bob ● Bob decrypts using same k ○ Bob ⇒ $\varepsilon(k, c) = m$ **Encryption + Decryption Key** 	

Types of encryption?

Symmetric	Asymmetric

• Each party has same key k for encryption
  ○ Alice ⇒ $\varepsilon(k, m) = c$ ⇒ Bob
• Bob decrypts using same k
  ○ Bob ⇒ $\varepsilon(k, c) = m$
 	
Encryption + Decryption Key

	• Each party has public and private key
	• To send a message to Bob, Alice encrypts using Bob's public key. Only Bob's private key can decrypt the message
	
	Encryption Key  **Decryption Key** 

Rivest Shamir Adleman (RSA)

- One the **oldest** asymmetric cryptography systems
- **Versatile** and widely used today:
  - Securing Email Communication
  - E-commerce and Online Shopping
  - Digital Signature and Document Verification
  - Secure Remote Access (VPN & SSH)

Overview of RSA Encryption

- Generate Public Key and Private Key
- Encrypt Message
- Decrypt Message

Key Generation

RSA Key Generation Walkthrough

1) Choose 2 distinct random prime number *p* and *q*.

Realistically, these would be very large (~ 1024 bit to 2048)

RSA Key Generation Walkthrough

1) Choose 2 distinct random prime number *p* and *q*.

Realistically, these would be very large (~ 1024 bit to 2048)

ex. $p = 3$ & $q = 11$

RSA Key Generation Walkthrough

2) Compute $n = p \cdot q$

RSA Key Generation Walkthrough

2) Compute $n = p \cdot q$

$$n = 3 \cdot 11 = 33$$

RSA Key Generation Walkthrough

3) Calculate Euler's Totient Function **φ(n)**, the count of integers in {1, ..., n-1} that are coprime to n. If *p* and *q* are prime:

RSA Key Generation Walkthrough

3) Calculate Euler's Totient Function **φ(n)**, the count of integers in {1, ..., n-1} that are coprime to n. If *p* and *q* are prime:

$$\varphi(pq) = \varphi(p) \cdot \varphi(q)$$

RSA Key Generation Walkthrough

3) Calculate Euler's Totient Function **φ(n)**, the count of integers in {1, ..., n-1} that are coprime to n. If *p* and *q* are prime:

$$\varphi(pq) = \varphi(p) \cdot \varphi(q)$$

$$= (p - 1) \cdot (q - 1)$$

RSA Key Generation Walkthrough

3) Calculate Euler's Totient Function **φ(n)**, the count of integers in {1, ..., n-1} that are coprime to n. If *p* and *q* are prime:

$$\varphi(pq) = \varphi(p) \cdot \varphi(q)$$

$$= (p - 1) \cdot (q - 1)$$

$$= 2 \cdot 10 = 20$$

RSA Key Generation Walkthrough

4) **Choose** a **public** value **e** in [2, $\varphi(n) - 1$] that is coprime to $\varphi(n)$.

RSA Key Generation Walkthrough

4) **Choose** a **public** value **e** in [2, φ(n) - 1] that is coprime to φ(n).

With φ(n) = 20, e = 7 is valid:

- coprime to φ(n)
- 2 < 7 < 19

RSA Key Generation Walkthrough

5) Compute a private value **d** such that:

$\mathbf{(d \cdot e) \bmod \varphi(n) = 1}$.

RSA Key Generation Walkthrough

5) Compute a private value **d** such that:

$(\mathbf{d \cdot e}) \ \mathbf{mod} \ \boldsymbol{\varphi}\mathbf{(n) = 1}$.

One such d- "**modular multiplicative inverse**" of e

RSA Key Generation Walkthrough

5) Compute a private value **d** such that:

$(d \cdot e) \bmod \varphi(n) = 1$.

One such d- "**modular multiplicative inverse**" of e

e = 7, then **d = 3**:

3 * 7 = 21   &   21 mod 20 = 1

Final Values

6) **Public key (e, n)**

    (e: 7, n: 33)

    Private key (d, n, p, q)

    (d: 3, n: 33, p: 3, q: 11)

Encryption

How Does Encryption / Decryption Work?

- Encryption and decryption involves **exponentiation** to make a message huge and **modding** it to bring it down to a reasonably sized but secure cipher.

RSA Encryption and Decryption

To encrypt the message m:

$$c = m^e \pmod{n}$$

RSA Encryption and Decryption

To encrypt the message m:

$$c = m^e \pmod n$$

To decrypt the cipher c:

$$m = c^d \pmod n$$

RSA Encryption and Decryption

To encrypt the message m:

$$\mathbf{c = m^e \pmod{n}}$$

To decrypt the cipher c:

$$\mathbf{m = c^d \pmod{n}}$$

(with **ed mod φ(n) = 1**)

RSA Encryption Walkthrough

With **Public key** ⇒ (7, 33) and **Private key**⇒(3, 33, 3, 11):

RSA Encryption Walkthrough

With **Public key** ⇒ (7, 33) and **Private key**⇒(3, 33, 3, 11):

Encrypt message m = 2:

$$c = m^e \pmod{n} = \mathbf{2^7} \bmod \mathbf{33 = 29}$$

RSA Encryption Walkthrough

With **Public key** ⇒ (7, 33) and **Private key**⇒(3, 33, 3, 11)):

Encrypt message m = 2:

$$c = m^e \pmod{n} = \mathbf{2^7} \text{ mod } \mathbf{33 = 29}$$

Decrypt cipher c:

$$m = c^d \pmod{n} = \mathbf{29^3} \text{ mod } \mathbf{33 = 2}$$

Why does this work?

- ***e*** and ***d*** are modular multiplicative inverses, which are related through totient function **φ(n)**.

RSA encryption – How is it secure?

- **Trapdoor Function:** Easy to calculate ϕ(n) given p & q, but very **difficult** to calculate **p or q** from **ϕ(n)**

RSA encryption – How is it secure?

- **Trapdoor Function:** Easy to calculate $\phi(n)$ given p & q, but very **difficult** to calculate **p or q** from **$\phi(n)$**
- Means that without knowing p & q, it's very difficult to **brute force** what d is, even though its value is calculated using e and $\phi(n)$

Why is it interesting?

- Relatively straightforward / **simple calculations** with a pretty **secure** outcome

Why is it interesting?

- Relatively straightforward / **simple calculations** with a pretty **secure** outcome
- **Proofs** of statements include a lot of math **we've learned** in discrete
  - proof of **Euler's Totient Function** via **inclusion-exclusion**
  - **Fermat's Little Theorem** proof via induction

Efficiency

Increasing Efficiency

- Carmichael's Totient function vs Euler's Totient Function

Increasing Efficiency

- Carmichael's Totient function vs Euler's Totient Function
- Decryption using Chinese remainder theorem

Increasing Efficiency

- Carmichael’s Totient function vs Euler’s Totient Function
- Decryption using Chinese remainder theorem
- Optimized Code

Euler's Vs. Carmichael’s Key Generation

Euler’s $\varphi(n)$	
$\varphi(n)$ is the total sum of positive integers that are lesser than n and are co-prime numbers to n $\varphi(pq) = (p-1)(q-1)$	

Euler's Vs. Carmichael’s Key Generation

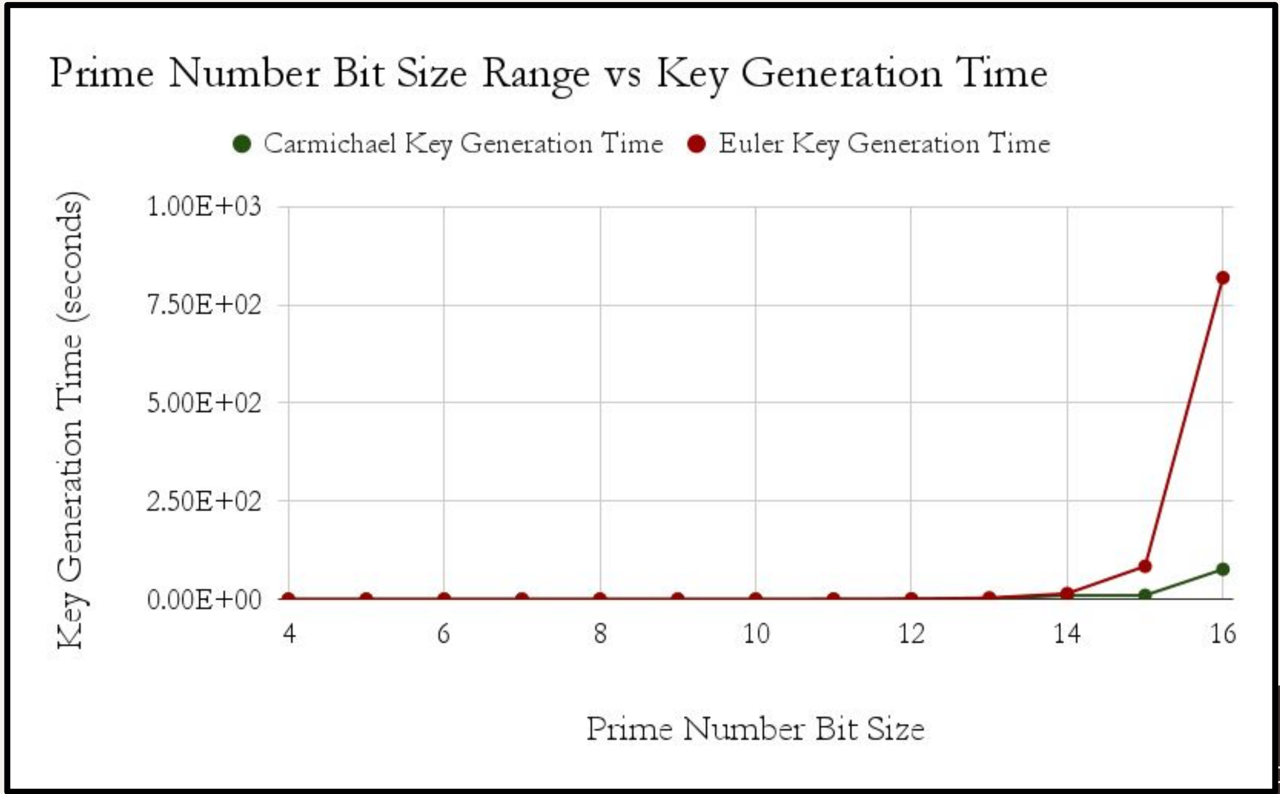
Euler’s *φ(n)*	Carmichael’s λ(n)
φ(n) is the total sum of positive integers that are lesser than n and are co-prime numbers to n $\varphi(pq) = (p-1)(q-1)$	λ(n) is the smallest integer *m* where $a^m \equiv 1 \pmod{n}$ for all a co-prime to n and < n. $\lambda(pq) = \mathrm{lcm}(p-1, q-1)$

Euler's Vs. Carmichael’s Key Generation

Euler’s $\varphi(n)$	Carmichael’s $\lambda(n)$
$\varphi(n)$ is the total sum of positive integers that are lesser than n and are co-prime numbers to n $\varphi(pq) = (p-1)(q-1)$	$\lambda(n)$ is the smallest integer *m* where $a^m \equiv 1 \pmod{n}$ for all a co-prime to n and < n. $\lambda(pq) = \mathrm{lcm}(p-1, q-1)$

n	1	2	3	4	8	12	15	16	20	21
$\lambda(n)$	1	1	2	2	2	2	4	4	4	6
$\phi(n)$	1	1	2	2	4	4	8	8	8	12

Time Complexity of Generating Public/Private Key Pairs



Attacks

Attacks

- RSA Algorithm currently has no major vulnerabilities
- Attacks usually require specific conditions to be fulfilled
- Most attacks involve figuring out p, q, or d
  - If you know p and q, you can calculate φ(n) which allows you to calculate the private key (d)

Attacks - Implementation

- **Wiener's Attack:** If the p & q values are close, and the private key (d) size is small, we can calculate what d is from the convergents of the continued fraction e/n

- **Hastad's Broadcast Attack:** You can bypass needing to know the private key if you have enough ciphertexts using the same public key and same plaintext

Future of RSA

- Currently a widely used encryption algorithm
- Increasing computational efficiency and quantum computing
  - Currently no widely accepted method of cracking securely implemented RSA encryption

Future of RSA

- Currently a widely used encryption algorithm
- Increasing computational efficiency and quantum computing
  - Currently no widely accepted method of cracking securely implemented RSA encryption
    - **Shor's Algorithm:** *theoretically* allows factorization of prime numbers in polynomial time rather than exponential time in quantum computing

Future of RSA

- Currently a widely used encryption algorithm
- Increasing computational efficiency and quantum computing
    - Currently no widely accepted method of cracking securely implemented RSA encryption
        - **Shor's Algorithm:** *theoretically* allows factorization of prime numbers in polynomial time rather than exponential time in quantum computing
        - Chinese Researchers have "claimed" to have created another algorithm that could easily break a 2048-bit key RSA (2023)

Thanks For Listening

Citations