Public Key Cryptography

Carman S. Cater

Outline

Introduction to Cryptography

- History
- Encryption
  - Symmetric Key
  - Public Key

Figure: Made using `wordart.com`

History

- **Early Days**
  - 1500 BC - Clay tablets in Mesopotamia meant to encrypt information

History

- **Early Days**
  - 1500 BC - Clay tablets in Mesopotamia meant to encrypt information
  - 500 BC - Substitution chiphers being used by Hebrew scholars

History

- **Early Days**
  - 1500 BC - Clay tablets in Mesopotamia meant to encrypt information
  - 500 BC - Substitution chiphers being used by Hebrew scholars
  - 100 BC - Caesar Cipher - Shift each letter a fixed number of positions in the alphabet

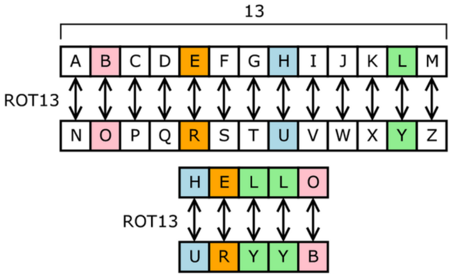


Figure: ROT13 (Type of Caesar Cipher)

History

- **Early Days**
  - 1500 BC - Clay tablets in Mesopotamia meant to encrypt information
  - 500 BC - Substitution chiphers being used by Hebrew scholars
  - 100 BC - Caesar Cipher - Shift each letter a fixed number of positions in the alphabet
  - AD 800 - Al-Kindi, an Arab mathematician used frequency analysis to break ciphers

Figure: Al-Kindi

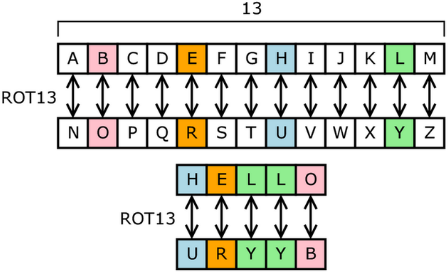


Figure: ROT13 (Type of Caesar Cipher)

History (cont.)

- **Later Advancements**
  - 1917 - M-94 developed by the US Army

Figure: M-94

History (cont.)

- **Later Advancements**
  - 1917 - M-94 developed by the US Army
  - World War II - The German Army used an electromechanical rotor machine known as the Enigma

Figure: M-94

Figure: Enigma

History (cont.)

- **Recent Developents**
  - 1970's - Now

History (cont.)

- **Recent Developents**
  - 1970's - Now
    - The Internet for commercial purposes created a need for encryption standards

History (cont.)

- **Recent Developents**
  - 1970's - Now
    - ★ The Internet for commercial purposes created a need for encryption standards
    - ★ Many new symmetric schemes as well as the discovery of public key algorithms based on one-way functions

History (cont.)

- **Recent Developents**
  - 1970's - Now
    - The Internet for commercial purposes created a need for encryption standards
    - Many new symmetric schemes as well as the discovery of public key algorithms based on one-way functions
    - Popular encryption schemes used today include AES, Diffie-Hellman, RSA, Elliptic curve, and many more

Encryption

Encryption is the process of encoding information. Start with plaintext, create ciphertext.

- **Symmetric Key Encryption (Single private key)**

Encryption

Encryption is the process of encoding information. Start with plaintext, create ciphertext.

- **Symmetric Key Encryption (Single private key)**
  - Caesar Cipher - Left/Right shift the alphabet by some amount
    - For example, suppose the word "cat" maps to "ecu". Key = right 3
    - $c = \alpha + k \ (mod\ 26)$ for some letter α and key k

Encryption

Encryption is the process of encoding information. Start with plaintext, create ciphertext.

- **Symmetric Key Encryption (Single private key)**
  - Caesar Cipher - Left/Right shift the alphabet by some amount
    - For example, suppose the word "cat" maps to "ecu". Key = right 3
    - $c = \alpha + k \ (mod\ 26)$ for some letter α and key k
  - XOR Cipher - Take message m in binary and random bit string k of equal length. Perform XOR operation
    - $c = m \oplus k = 0111\,1001\,1100 \oplus 1010\,0111\,0011 = 1101\,1110\,1111$
    - $m = c \oplus k = 1101\,1110\,1111 \oplus 1010\,0111\,0011 = 0111\,1001\,1100$

Encryption

Encryption is the process of encoding information. Start with plaintext, create ciphertext.

- **Symmetric Key Encryption (Single private key)**
  - Caesar Cipher - Left/Right shift the alphabet by some amount
    - For example, suppose the word "cat" maps to "ecu". Key = right 3
    - $c = \alpha + k \ (mod\ 26)$ for some letter α and key k
  - XOR Cipher - Take message m in binary and random bit string k of equal length. Perform XOR operation
    - $c = m \oplus k = 0111\,1001\,1100 \oplus 1010\,0111\,0011 = 1101\,1110\,1111$
    - $m = c \oplus k = 1101\,1110\,1111 \oplus 1010\,0111\,0011 = 0111\,1001\,1100$
  - The modern standard is Advanced Encryption Standard (AES) published in 1988
    - Became the U.S. federal government standard in 2002 and is approved by the NSA

Encryption (cont.)

The nontrivial issue with symmetric algorithms is the transmission of the private key over a public channel.

Encryption (cont.)

The nontrivial issue with symmetric algorithms is the transmission of the private key over a public channel.

- **Public-Key Encryption (Public/Private Key Pair)**
  - Very recent development, beginning in the 1970's
  - Popular public-key algorithms include
    - Diffie-Hellman Key Exchange
    - Elgamal Encryption Scheme
    - RSA Encryption

Number Theory Background

Definition

(The Ring of Integers Modulo n). This ring is denoted by $\mathbb{Z}_n$ and is the quotient

$$\mathbb{Z}/n\mathbb{Z} = \{0, 1, 2, 3, ..., n-1\}$$

The operations are regular addition and multiplication reduced modulo n.

Number Theory Background

Definition

(The Ring of Integers Modulo n). This ring is denoted by $\mathbb{Z}_n$ and is the quotient

$$\mathbb{Z}/n\mathbb{Z} = \{0, 1, 2, 3, ..., n-1\}$$

The operations are regular addition and multiplication reduced modulo n.

For our purposes, we will be using the multiplicative group of units

$$\mathbb{Z}_p^* = \{1, 2, ..., p-1\}$$

for p prime. Elgamal and RSA require the existence of inverses.

Number Theory Background (cont.)

Definition

(Euler's Totient Function). Given some $n \in \mathbb{N}$ how many natural numbers in the range $[1, n]$ are relatively prime to n? For $n \in \mathbb{N}$, let

$$\varphi(n) = \#\{a \in \mathbb{N} \mid 1 \leq a < n \quad and \quad gcd(a, n) = 1\}$$

Number Theory Background (cont.)

Definition

(Euler's Totient Function). Given some $n \in \mathbb{N}$ how many natural numbers in the range $[1, n]$ are relatively prime to n? For $n \in \mathbb{N}$, let

$$\varphi(n) = \#\{a \in \mathbb{N} \mid 1 \leq a < n \;\; and \;\; gcd(a, n) = 1\}$$

There is a nice formula for computing $\varphi(n)$ when the prime factorization of n is known. Suppose

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_m^{\alpha_m}$$

is the prime factorization of n, with each prime factor p_i distinct from the others. Then

$$\varphi(n) = \Pi_{i=1}^{m}(p_i^{\alpha_i} - p_i^{\alpha_i - 1})$$

Number Theory Background (cont.)

Definition

(Euler's Totient Function). Given some $n \in \mathbb{N}$ how many natural numbers in the range $[1, n]$ are relatively prime to n? For $n \in \mathbb{N}$, let

$$\varphi(n) = \#\{a \in \mathbb{N} \mid 1 \leq a < n \ \ and \ \ gcd(a, n) = 1\}$$

There is a nice formula for computing $\varphi(n)$ when the prime factorization of n is known. Suppose

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_m^{\alpha_m}$$

is the prime factorization of n, with each prime factor p_i distinct from the others. Then

$$\varphi(n) = \Pi_{i=1}^{m}(p_i^{\alpha_i} - p_i^{\alpha_i - 1})$$

Special Case

Now let's look at the special case when $n \in \mathbb{N}$ and $n = p \cdot q$ for p, q prime. This gives us

$$\phi(n) = (p-1)(q-1)$$

This will be of particular use when doing the RSA algorithm.

Number Theory Background (cont.)

Proposition

If $gcd(a, n) = 1$, then the equation $ax \equiv b \pmod{n}$ has a solution, and that solution is unique modulo n.

Number Theory Background (cont.)

Proposition

If $gcd(a, n) = 1$, then the equation $ax \equiv b \pmod{n}$ has a solution, and that solution is unique modulo n.

Proof.

By Bezout's Identity there exist x and y such that $ax + ny = gcd(a, n) = 1$. Looking at the equation modulo n we get $ax \equiv 1\ (mod\ n)$ Multiplying on both sides by b gives us our desired result

$$a(xb) \equiv b\ (mod\ n)$$

□

Number Theory Background (cont.)

Proposition

If $gcd(a, n) = 1$, then the equation $ax \equiv b \pmod{n}$ has a solution, and that solution is unique modulo n.

Proof.

By Bezout's Identity there exist x and y such that $ax + ny = gcd(a, n) = 1$. Looking at the equation modulo n we get $ax \equiv 1\ (mod\ n)$ Multiplying on both sides by b gives us our desired result

$$a(xb) \equiv b\ (mod\ n)$$

□

Remark: For the RSA we will need to solve the equation $ax \equiv 1 \pmod{p}$.

Number Theory Background (cont.)

Theorem

(Euler's Theorem). If $gcd(x, n) = 1$, *then*

$$x^{\phi(n)} \equiv 1 \; (mod\ n)$$

Number Theory Background (cont.)

Theorem

(Euler's Theorem). If $gcd(x, n) = 1$, then

$$x^{\phi(n)} \equiv 1 \; (mod\; n)$$

Proof.

Using machinery from abstract algebra, we can identify x as being $x \in (\mathbb{Z}/n\mathbb{Z})^*$ and since

$$\#(\mathbb{Z}/n\mathbb{Z})^* = \phi(n)$$

we can employ Lagrange's Theorem. By Lagrange's Theorem, the order of x divides $\phi(n)$. Say $\phi(n) = | x | * k$ for some $k \in \mathbb{Z}$. Then

$$x^{\phi(n)} \equiv x^{|x| * k} \equiv 1 \; (mod\; n)$$

which is the desired result. □

Note: Euler's Theorem is used in the proof of the RSA encryption algorithm.

Diffie-Hellman Key Exchange (1976)

1. About DHKE
2. How Does DHKE Work?
   - Example with Small Numbers
   - Example in SageMath
3. How to Break DHKE
4. Discrete Logarithm Problem
   - Attacks Against the DLP
5. Closing Remarks on DHKE

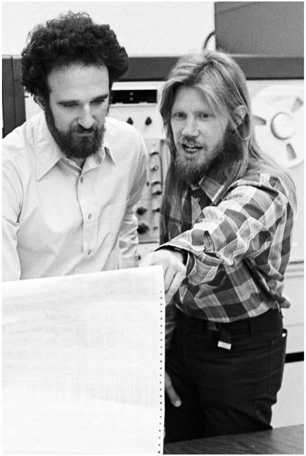


Figure: Whitfield Diffie and Martin Hellman Image: https://news.stanford.edu/news/2016/march/images/16185-turingtwo_news.jpg

About DHKE

- Discovered in 1976 by Whitfield Diffie and Martin Hellman. This marks the beginning of asymmetric cryptography

About DHKE

- Discovered in 1976 by Whitfield Diffie and Martin Hellman. This marks the beginning of asymmetric cryptography
- Allows two parties to derive a shared secret key over a public channel

About DHKE

- Discovered in 1976 by Whitfield Diffie and Martin Hellman. This marks the beginning of asymmetric cryptography
- Allows two parties to derive a shared secret key over a public channel
- Operations initially performed in $\mathbb{Z}_p^*$

About DHKE

- Discovered in 1976 by Whitfield Diffie and Martin Hellman. This marks the beginning of asymmetric cryptography
- Allows two parties to derive a shared secret key over a public channel
- Operations initially performed in $\mathbb{Z}_p^*$
- Recent Diffie-Hellman implementations use Elliptic Curves

About DHKE

- Discovered in 1976 by Whitfield Diffie and Martin Hellman. This marks the beginning of asymmetric cryptography
- Allows two parties to derive a shared secret key over a public channel
- Operations initially performed in $\mathbb{Z}_p^*$
- Recent Diffie-Hellman implementations use Elliptic Curves
- Used in cryptographic protocols such as Secure Shell (SSH), Transport Layer Security (TLS), and Internet Protocol Security (IPSec)

How Does DHKE Work?

Procedure (Public $= \{p, g, A, B\}$, Private $= \{n, m\}$)

1. Alice and bob agree publicly on a large prime number p and a primitive element g such that $1 < g < p$

How Does DHKE Work?

Procedure (Public $= \{p, g, A, B\}$, Private $= \{n, m\}$)

1. Alice and bob agree publicly on a large prime number p and a primitive element g such that $1 < g < p$
2. Alice secretly chooses an integer n
3. Bob secretly chooses an integer m

How Does DHKE Work?

Procedure (Public = $\{p, g, A, B\}$, Private = $\{n, m\}$)

1. Alice and bob agree publicly on a large prime number p and a primitive element g such that $1 < g < p$
2. Alice secretly chooses an integer n
3. Bob secretly chooses an integer m
4. Alice computes $A = g^n \pmod{p}$ and Bob computes $B = g^m \pmod{p}$. They then tell each other their results

How Does DHKE Work?

Procedure (Public $= \{p, g, A, B\}$, Private $= \{n, m\}$)

1. Alice and bob agree publicly on a large prime number p and a primitive element g such that $1 < g < p$
2. Alice secretly chooses an integer n
3. Bob secretly chooses an integer m
4. Alice computes $A = g^n \pmod{p}$ and Bob computes $B = g^m \pmod{p}$. They then tell each other their results
5. The shared secret key is

$$s \equiv (g^n)^m \equiv (g^m)^n \equiv g^{m \cdot n} \, (mod \; p)$$

How Does DHKE Work?

Procedure (Public $= \{p, g, A, B\}$, Private $= \{n, m\}$)

1. Alice and bob agree publicly on a large prime number p and a primitive element g such that $1 < g < p$
2. Alice secretly chooses an integer n
3. Bob secretly chooses an integer m
4. Alice computes $A = g^n \pmod p$ and Bob computes $B = g^m \pmod p$. They then tell each other their results
5. The shared secret key is

$$s \equiv (g^n)^m \equiv (g^m)^n \equiv g^{m \cdot n}\, (mod\ p)$$

Proof.

The correctness of the algorithm is fairly obvious. Commutativity in $\mathbb{Z}_p^*$ follows from commutativity in $\mathbb{Z}$. □

Example of DHKE with Small Numbers

Example

1. Take the prime $p = 97$, and primitive element $g = 5$

Note: We take g to be a primitive element so that when g is exponentiated, it can take on any value in the group $\mathbb{Z}_p^*$.

Example of DHKE with Small Numbers

Example

1. Take the prime $p = 97$, and primitive element $g = 5$
2. Alice randomly chooses $n = 31$
3. Bob randomly chooses $m = 95$

Note: We take g to be a primitive element so that when g is exponentiated, it can take on any value in the group $\mathbb{Z}_p^*$.

Example of DHKE with Small Numbers

Example

1. Take the prime $p = 97$, and primitive element $g = 5$
2. Alice randomly chooses $n = 31$
3. Bob randomly chooses $m = 95$
4. Alice computes $g^n \equiv 5^{31} \equiv 7 \ (mod \ 97)$
5. Bob computes $g^m \equiv 5^{95} \equiv 39 \ (mod \ 97)$

Note: We take g to be a primitive element so that when g is exponentiated, it can take on any value in the group $\mathbb{Z}_p^*$.

Example of DHKE with Small Numbers

Example

1. Take the prime $p = 97$, and primitive element $g = 5$
2. Alice randomly chooses $n = 31$
3. Bob randomly chooses $m = 95$
4. Alice computes $g^n \equiv 5^{31} \equiv 7 \, (mod\, 97)$
5. Bob computes $g^m \equiv 5^{95} \equiv 39 \, (mod\, 97)$
6. The shared secret key is $s \equiv (g^n)^m \equiv (g^m)^n \equiv 14 \, (mod\, 97)$

Note: We take g to be a primitive element so that when g is exponentiated, it can take on any value in the group $\mathbb{Z}_p^*$.

Example of DHKE in SageMath

Remember, $\{p, g, A, B\}$ below are public, while $\{n, m\}$ are private.

```
In [96]: bitSize = 2^20 # The bit size of p
In [97]: p = next_prime(ZZ.random_element(1, bitSize)) ; p # This is the modulus (public)
Out[97]: 136811
In [98]: g = Integers(p).multiplicative_generator(); g # Primitive element (public)
Out[98]: 2
In [99]: g.multiplicative_order()
Out[99]: 136810
In [100]: n = ZZ.random_element(1, p) ; n # This is Alice's Key (private)
Out[100]: 82089
In [101]: m = ZZ.random_element(1, p) ; m # This is Bob's key (private)
Out[101]: 90529
In [102]: A = Mod(g^n, p); A # Alice sends this to Bob (public)
Out[102]: 132163
In [103]: B = Mod(g^m, p); B # Bob sends this to Alice (public)
Out[103]: 135216
In [104]: Mod((g^n)^m, p) # This is the shared secret key
Out[104]: 14437
In [105]: Mod((g^m)^n, p) # This is the shared secret key
Out[105]: 14437
```

Let's looks at a live demo with a larger bit length.

How to Break DHKE

Recall the DHKE procedure

Procedure (Public = $\{p, g, A, B\}$, Private = $\{n, m\}$)

1. Alice and bob agree publicly on a large prime number p and a primitive element g such that $1 < g < p$
2. Alice secretly chooses an integer n
3. Bob secretly chooses an integer m
4. Alice computes $A = g^n \pmod p$ and Bob computes $B = g^m \pmod p$. They then tell each other their results
5. The shared secret key is $s \equiv (g^n)^m \equiv (g^m)^n \equiv g^{m \cdot n}\ (mod\ p)$

If we can find a way to compute either Alice's or Bob's secret keys n or m, then we can simply compute the secret key by exponentiating B or A respectively.

Discrete Logarithm Problem

Discrete Log Problem (DLP)

Let G be a finite group such as $\mathbb{Z}_p^*$. Given $b \in G$ and a power a of b, find a positive integer n such that $b^n \equiv a \,(mod\ p)$

Discrete Logarithm Problem

Discrete Log Problem (DLP)

Let G be a finite group such as $\mathbb{Z}_p^*$. Given $b \in G$ and a power a of b, find a positive integer n such that $b^n \equiv a \, (mod\ p)$

Example

Take $\mathbb{Z}_{47}^*$ and $\alpha = 5$ such that $\mid \alpha \mid = 46$ so that α is a primitive root. Find the positive integer x such that $5^x \equiv 41 \, (mod\ 47)$

Discrete Logarithm Problem

Discrete Log Problem (DLP)

Let G be a finite group such as $\mathbb{Z}_p^*$. Given $b \in G$ and a power a of b, find a positive integer n such that $b^n \equiv a \, (mod \; p)$

Example

Take $\mathbb{Z}_{47}^*$ and $\alpha = 5$ such that $\mid \alpha \mid = 46$ so that α is a primitive root. Find the positive integer x such that $5^x \equiv 41 \, (mod \; 47)$

Note: If we can solve the DLP, then we can easily break the DHKE. There are several classical (non quantum) algorithms we mention below.

Attacks Against the DLP

- **Brute-Force Search**
  - Simply compute powers and hope for a match.
  - $| G | \geq 2^{80}$ to be infeasible. Greater than 24 decimal digits

Attacks Against the DLP

- **Brute-Force Search**
  - Simply compute powers and hope for a match.
  - $\mid G \mid \geq 2^{80}$ to be infeasible. Greater than 24 decimal digits
- **Shanks' Baby-Step Giant-Step Method**
  - Time-memory tradeoff. Reduces the time at the cost of extra storage.
  - $\mid G \mid \geq 2^{160}$ to be infeasible. Greater than 48 decimal digits

Attacks Against the DLP

- **Brute-Force Search**
  - Simply compute powers and hope for a match.
  - $\mid G \mid \geq 2^{80}$ to be infeasible. Greater than 24 decimal digits
- **Shanks' Baby-Step Giant-Step Method**
  - Time-memory tradeoff. Reduces the time at the cost of extra storage.
  - $\mid G \mid \geq 2^{160}$ to be infeasible. Greater than 48 decimal digits
- **Pollard's Rho Method for Logarithms**
  - Currently the best known classical algorithm for computing discrete log in elliptic curve groups
  - $\mid G \mid \geq 2^{160}$ to be infeasible

Attacks Against the DLP cont.

- **Pohlig-Hellman Algorithm**
  - Based on the Chinese Remainder Theorem, it relies on the prime factorization of $\mid G \mid$
  - Group order must have prime factor $\geq 2^{160}$

Attacks Against the DLP cont.

- **Pohlig-Hellman Algorithm**
  - Based on the Chinese Remainder Theorem, it relies on the prime factorization of $\mid G \mid$
  - Group order must have prime factor $\geq 2^{160}$
- **Index Calculus Method**
  - Exploits properties of $\mathbb{Z}_p^*$, while the previous methods were independent of the underlying group
  - $\mid G \mid \geq 2^{1024}$ to be infeasible. Greater than 308 decimal digits

Attacks Against the DLP cont.

- **Pohlig-Hellman Algorithm**
  - Based on the Chinese Remainder Theorem, it relies on the prime factorization of $\mid G \mid$
  - Group order must have prime factor $\geq 2^{160}$
- **Index Calculus Method**
  - Exploits properties of $\mathbb{Z}_p^*$, while the previous methods were independent of the underlying group
  - $\mid G \mid \geq 2^{1024}$ to be infeasible. Greater than 308 decimal digits
- **Man in the Middle Attack**
  - Eve intercepts Alice's g^n and Bob's g^m
  - Eve sends Alice and Bob g^t and now acts as the middleman
  - The two private keys are now $g^{n \cdot t}$ and $g^{m \cdot t}$ using Eve's integer t
  - From now on, Eve is able to intercept, decrypt, and change the messages in subtle ways

List of Records for solving the DLP

`https://en.wikipedia.org/wiki/Discrete_logarithm_records`

Closing Remarks on DHKE

- Diffie-Hellman is a widely used protocol for key exchange, often then used in conjunction with symmetric algorithms

Closing Remarks on DHKE

- Diffie-Hellman is a widely used protocol for key exchange, often then used in conjunction with symmetric algorithms
- It relies on the "difficulty" of the discrete logarithm problem. Quantum computers may pose a threat

Closing Remarks on DHKE

- Diffie-Hellman is a widely used protocol for key exchange, often then used in conjunction with symmetric algorithms
- It relies on the "difficulty" of the discrete logarithm problem. Quantum computers may pose a threat
- The best known attacks against RSA are
  - General number field sieve for classical computers
  - Shor's algorithm for quantum computers

Elgamal Encryption Scheme (1985)

- About Elgamal
- How Does Elgamal Work?
  - Example by Hand
  - Example using SageMath
- Security of Elgamal
- Closing Remarks on Elgamal

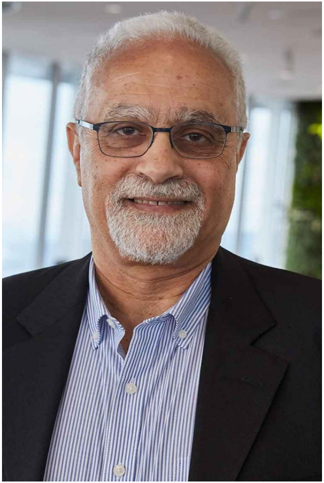


Figure: Taher Elgamal Image:
https://evolutionequity.com/team/taher-elgamal

About Elgamal

- An extension of Diffie-Hellman proposed in 1985 by Taher Elgamal

About Elgamal

- An extension of Diffie-Hellman proposed in 1985 by Taher Elgamal
- Is used to encrypt a message m as ciphertext

About Elgamal

- An extension of Diffie-Hellman proposed in 1985 by Taher Elgamal
- Is used to encrypt a message m as ciphertext
- Security relies on the Discrete Log Problem

How does Elgamal Work?

Procedure (Public $= \{p, g, g^n, g^m\}$, Private $= \{n, m\}$)

1. Perform the Diffie-Hellman Key Exchange and arrive at the shared secret key

$$s \equiv (g^n)^m \equiv (g^m)^n \equiv g^{m \cdot n} \; (mod \; p)$$

How does Elgamal Work?

Procedure (Public $= \{p, g, g^n, g^m\}$, Private $= \{n, m\}$)

1. Perform the Diffie-Hellman Key Exchange and arrive at the shared secret key

$$s \equiv (g^n)^m \equiv (g^m)^n \equiv g^{m \cdot n} \ (mod\ p)$$

2. Alice converts her plaintext message m into an element of $\mathbb{Z}_p^*$ and computes

$$c \equiv m \cdot s \ (mod\ p)$$

She sends the ciphertext c to Bob

How does Elgamal Work?

Procedure (Public $= \{p, g, g^n, g^m\}$, Private $= \{n, m\}$)

1. Perform the Diffie-Hellman Key Exchange and arrive at the shared secret key

$$s \equiv (g^n)^m \equiv (g^m)^n \equiv g^{m \cdot n} \, (mod \; p)$$

2. Alice converts her plaintext message m into an element of $\mathbb{Z}_p^*$ and computes

$$c \equiv m \cdot s \, (mod \; p)$$

   She sends the ciphertext c to Bob

3. Bob computes the inverse of s

How does Elgamal Work?

Procedure (Public $= \{p, g, g^n, g^m\}$, Private $= \{n, m\}$)

1. Perform the Diffie-Hellman Key Exchange and arrive at the shared secret key

$$s \equiv (g^n)^m \equiv (g^m)^n \equiv g^{m \cdot n} \ (mod\ p)$$

2. Alice converts her plaintext message m into an element of $\mathbb{Z}_p^*$ and computes

$$c \equiv m \cdot s \ (mod\ p)$$

   She sends the ciphertext c to Bob

3. Bob computes the inverse of s
4. Bob recovers the plaintext message by computing

$$c \cdot s^{-1} \equiv (m \cdot s) \cdot s^{-1} \equiv m \ (mod\ p)$$

How does Elgamal Work?

Procedure (Public $= \{p, g, g^n, g^m\}$, Private $= \{n, m\}$)

1. Perform the Diffie-Hellman Key Exchange and arrive at the shared secret key

$$s \equiv (g^n)^m \equiv (g^m)^n \equiv g^{m \cdot n} \, (mod \; p)$$

2. Alice converts her plaintext message m into an element of $\mathbb{Z}_p^*$ and computes

$$c \equiv m \cdot s \, (mod \; p)$$

   She sends the ciphertext c to Bob
3. Bob computes the inverse of s
4. Bob recovers the plaintext message by computing

$$c \cdot s^{-1} \equiv (m \cdot s) \cdot s^{-1} \equiv m \, (mod \; p)$$

Proof.

The proof is essentially baked into the procedure. We rely on the existence of s^{-1}, which exists because s is a power of g and g is primitive. □

Example of Elgamal with Small Numbers

Example

1. Take the prime $p = 29$, and primitive element $g = 2$

Example of Elgamal with Small Numbers

Example

1. Take the prime $p = 29$, and primitive element $g = 2$
2. Alice randomly chooses $n = 5$, Bob randomly chooses $m = 12$

Example of Elgamal with Small Numbers

Example

1. Take the prime $p = 29$, and primitive element $g = 2$
2. Alice randomly chooses $n = 5$, Bob randomly chooses $m = 12$
3. Alice computes $g^n \equiv 2^5 \equiv 3 \ (mod\ 29)$
4. Bob computes $g^m \equiv 2^{12} \equiv 7 \ (mod\ 29)$

Example of Elgamal with Small Numbers

Example

1. Take the prime $p = 29$, and primitive element $g = 2$
2. Alice randomly chooses $n = 5$, Bob randomly chooses $m = 12$
3. Alice computes $g^n \equiv 2^5 \equiv 3 \, (mod\, 29)$
4. Bob computes $g^m \equiv 2^{12} \equiv 7 \, (mod\, 29)$
5. The shared secret key is $s \equiv (g^n)^m \equiv (g^m)^n \equiv 16 \, (mod\, 29)$

Example of Elgamal with Small Numbers

Example

1. Take the prime $p = 29$, and primitive element $g = 2$
2. Alice randomly chooses $n = 5$, Bob randomly chooses $m = 12$
3. Alice computes $g^n \equiv 2^5 \equiv 3 \ (mod\ 29)$
4. Bob computes $g^m \equiv 2^{12} \equiv 7 \ (mod\ 29)$
5. The shared secret key is $s \equiv (g^n)^m \equiv (g^m)^n \equiv 16 \ (mod\ 29)$
6. Alice takes message $m = 26$ and sends Bob the ciphertext

$$c \equiv m \cdot s \equiv 26 \cdot 16 \equiv 10 \ (mod\ 29)$$

Example of Elgamal with Small Numbers

Example

1. Take the prime $p = 29$, and primitive element $g = 2$
2. Alice randomly chooses $n = 5$, Bob randomly chooses $m = 12$
3. Alice computes $g^n \equiv 2^5 \equiv 3 \ (mod\ 29)$
4. Bob computes $g^m \equiv 2^{12} \equiv 7 \ (mod\ 29)$
5. The shared secret key is $s \equiv (g^n)^m \equiv (g^m)^n \equiv 16 \ (mod\ 29)$
6. Alice takes message $m = 26$ and sends Bob the ciphertext

$$c \equiv m \cdot s \equiv 26 \cdot 16 \equiv 10 \ (mod\ 29)$$

7. Bob computes $s^{-1} = 20$ and retrieves the plaintext by computing

$$m \equiv c \cdot s^{-1} \equiv (m \cdot s) \cdot s^{-1} \equiv 10 \cdot 20 \equiv 26 \ (mod\ 29)$$

Example of Elgamal in SageMath

Here are the functions we will use in SageMath to demo the Elgamal Encryption Scheme.

```
In [297]: def encode(s):
              s = str(s) # make input a string
              return sum(ord(s[i])*256^i for i in range(len(s)))
In [298]: def decode(n):
              n = Integer(n) # make input an integer
              v = []
              while n != 0:
                  v.append(chr(n % 256))
                  n = n//256 # this replaces n by floor(n/256)
              return ''.join(v)
In [299]: def encryptElgamal(m, k):
              return lift(Mod(m*k, p)) # Multiply message by the private key
In [264]: def decryptElgamal(c, k):
              kInverse = inverse_mod(Integer(k), Integer(p)) # Compute the inverse of the private key
              return lift(Mod(c*kInverse, p))
```

Let's looks at a live demo with a large bit length.

Closing Remarks on Elgamal

- **Security**
  - Same as Diffie-Hellman, if the discrete logarithm can be solved, it can be broken.

Closing Remarks on Elgamal

- **Security**
  - Same as Diffie-Hellman, if the discrete logarithm can be solved, it can be broken.
  - A more subtle attack can be used which changes the message that Alice is trying to send

Closing Remarks on Elgamal

- **Security**
  - Same as Diffie-Hellman, if the discrete logarithm can be solved, it can be broken.
  - A more subtle attack can be used which changes the message that Alice is trying to send
    1. If an eavesdropper is able to intercept $c \equiv m \cdot s \, (mod \; p)$, she can replace it with $r \cdot c \equiv r \cdot m \cdot s \, (mod \; p)$ for some element r

Closing Remarks on Elgamal

- **Security**
  - Same as Diffie-Hellman, if the discrete logarithm can be solved, it can be broken.
  - A more subtle attack can be used which changes the message that Alice is trying to send
    1. If an eavesdropper is able to intercept $c \equiv m \cdot s \, (mod \; p)$, she can replace it with $r \cdot c \equiv r \cdot m \cdot s \, (mod \; p)$ for some element r
    2. When Bob decrypts, he gets
    $$r \cdot c \cdot s^{-1} \equiv r \cdot (m \cdot s) \cdot s^{-1} \equiv r \cdot m \, (mod \; p)$$
    3. In the event this is a bank transaction, we could manipulate the value being sent

RSA Encryption - Rivest, Shamir, Adleman (1977)

- About RSA
- How Does RSA Work?
  - Example with Small Numbers
  - Example in SageMath
- How to Break RSA
- Factorization Problem
  - Attacks Against the FP in RSA
- Closing Remarks on RSA

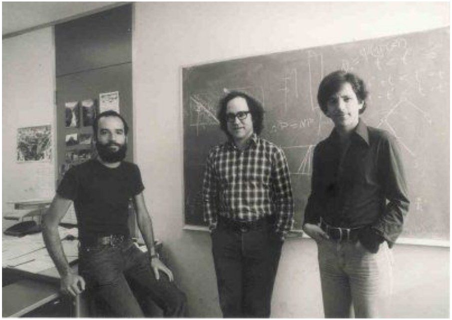


Figure: Ron Rivest, Adi Shamir, and Leonard Adleman Image: https://cdn.firespring.com/images/bf650823-bb00-4999-ad53-30b967fe948d.jpg

About RSA

- First described publicly in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman

About RSA

- First described publicly in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman
- A similar scheme was discovered four years earlier, in 1973 by Clifford Cocks of the British signals intelligence agency. It was not declassified until 1997

About RSA

- First described publicly in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman
- A similar scheme was discovered four years earlier, in 1973 by Clifford Cocks of the British signals intelligence agency. It was not declassified until 1997
- The security of the algorithm relies on the difficulty of factoring the product of large primes

About RSA

- First described publicly in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman
- A similar scheme was discovered four years earlier, in 1973 by Clifford Cocks of the British signals intelligence agency. It was not declassified until 1997
- The security of the algorithm relies on the difficulty of factoring the product of large primes
- Generally used to transmit secret keys to then be used with a symmetric key algorithm

How does RSA Work?

Procedure (Public $= \{n, e\}$, Private $= \{p, q, d\}$)

1. Alice picks two large prime numbers p and q, and computes $n = p \cdot q$

How does RSA Work?

Procedure (Public = $\{n, e\}$, Private = $\{p, q, d\}$)

1. Alice picks two large prime numbers p and q, and computes $n = p \cdot q$
2. Alice computes Euler's Totient function $\varphi(n) = (p - 1) \cdot (q - 1)$

How does RSA Work?

Procedure (Public $= \{n, e\}$, Private $= \{p, q, d\}$)

1. Alice picks two large prime numbers p and q, and computes $n = p \cdot q$
2. Alice computes Euler's Totient function $\varphi(n) = (p-1) \cdot (q-1)$
3. Alice chooses a random integer e such that $1 < e < \varphi(n)$ and $gcd(e, \varphi(n)) = 1$

How does RSA Work?

Procedure (Public $= \{n, e\}$, Private $= \{p, q, d\}$)

1. Alice picks two large prime numbers p and q, and computes $n = p \cdot q$
2. Alice computes Euler's Totient function $\varphi(n) = (p-1) \cdot (q-1)$
3. Alice chooses a random integer e such that $1 < e < \varphi(n)$ and $gcd(e, \varphi(n)) = 1$
4. Alice finds a solution d to the equation $e \cdot d \equiv 1 \; (mod \; \varphi(n))$

How does RSA Work?

Procedure (Public $= \{n, e\}$, Private $= \{p, q, d\}$)

1. Alice picks two large prime numbers p and q, and computes $n = p \cdot q$
2. Alice computes Euler's Totient function $\varphi(n) = (p-1) \cdot (q-1)$
3. Alice chooses a random integer e such that $1 < e < \varphi(n)$ and $gcd(e, \varphi(n)) = 1$
4. Alice finds a solution d to the equation $e \cdot d \equiv 1 \ (mod \ \varphi(n))$
5. Now by publishing the pair (n, e), anyone can encrypt an encoded message x to Alice by computing and sending

$$E(x) \equiv x^e \ (mod \ n)$$

How does RSA Work?

Procedure (Public $= \{n, e\}$, Private $= \{p, q, d\}$)

1. Alice picks two large prime numbers p and q, and computes $n = p \cdot q$
2. Alice computes Euler's Totient function $\varphi(n) = (p-1)\cdot(q-1)$
3. Alice chooses a random integer e such that $1 < e < \varphi(n)$ and $gcd(e, \varphi(n)) = 1$
4. Alice finds a solution d to the equation $e \cdot d \equiv 1 \, (mod \; \varphi(n))$
5. Now by publishing the pair (n, e), anyone can encrypt an encoded message x to Alice by computing and sending

$$E(x) \equiv x^e \, (mod \; n)$$

6. To decrypt, Alice computes

$$D(x^e) \equiv (x^e)^d \equiv x \, (mod \; n)$$

How does RSA Work? (cont.)

Borrowed from [Paar and Pelzl 2010]

Proof.

Given a ciphertext x^e, we need to show that $(x^e)^d \equiv x \, (mod \; n)$.

How does RSA Work? (cont.)

Borrowed from [Paar and Pelzl 2010]

Proof.

Given a ciphertext x^e, we need to show that $(x^e)^d \equiv x \, (mod \; n)$.

First recall that by hypothesis we have e, d such that $e \cdot d \equiv 1 \, (mod \; \varphi(n))$. This gives us $e \cdot d = 1 + t \cdot \varphi(n)$ for some integer t. Plugging this in to the congruence above we get

$$x^{e \cdot d} \equiv x^{1 + t \cdot \varphi(n)} \equiv (x^{\varphi(n)})^t \cdot x \, (mod \; n)$$

How does RSA Work? (cont.)

Borrowed from [Paar and Pelzl 2010]

Proof.

Given a ciphertext x^e, we need to show that $(x^e)^d \equiv x \,(mod\ n)$.

First recall that by hypothesis we have e, d such that $e \cdot d \equiv 1 \,(mod\ \varphi(n))$. This gives us $e \cdot d = 1 + t \cdot \varphi(n)$ for some integer t. Plugging this in to the congruence above we get

$$x^{e \cdot d} \equiv x^{1+t \cdot \varphi(n)} \equiv (x^{\varphi(n)})^t \cdot x \,(mod\ n)$$

We now consider two cases.

- Case 1: $gcd(x, n) = 1$

  By Euler's Theorem we have that $x^{\varphi(n)} \equiv 1 \,(mod\ n)$. This immediately gives us our result

$$(x^{\varphi(n)})^t \cdot x \equiv 1^t \cdot x \equiv x \,(mod\ n)$$

Continued on next page...

How does RSA Work? (cont.)

Proof.

- Case 2: $gcd(x, n) = gcd(x, p \cdot q) \neq 1$

  Given that p and q are primes, it follows that x must have one of them as a factor. Without loss of generality, assume that $x = r \cdot p$ for some integer r such that $r < q$. Since $gcd(x, q) = 1$ we have by Euler's Theorem

$$(x^{\varphi(q)})^t \equiv 1^t \equiv 1 \; (mod \; q)$$

How does RSA Work? (cont.)

Proof.

- Case 2: $gcd(x, n) = gcd(x, p \cdot q) \neq 1$

  Given that p and q are primes, it follows that x must have one of them as a factor. Without loss of generality, assume that $x = r \cdot p$ for some integer r such that $r < q$. Since $gcd(x, q) = 1$ we have by Euler's Theorem

  $$(x^{\varphi(q)})^t \equiv 1^t \equiv 1 \; (mod \; q)$$

  We now look at $(x^{\varphi(n)})^t$ again, giving us

  $$(x^{\varphi(n)})^t \equiv (x^{(q-1)(p-1)})^t \equiv ((x^{\varphi(q)})^t)^{(p-1)} \equiv 1^{(p-1)} \equiv 1 \; (mod \; q)$$

How does RSA Work? (cont.)

Proof.

- Case 2: $gcd(x, n) = gcd(x, p \cdot q) \neq 1$

  Given that p and q are primes, it follows that x must have one of them as a factor. Without loss of generality, assume that $x = r \cdot p$ for some integer r such that $r < q$. Since $gcd(x, q) = 1$ we have by Euler's Theorem

$$(x^{\varphi(q)})^t \equiv 1^t \equiv 1 \; (mod\ q)$$

  We now look at $(x^{\varphi(n)})^t$ again, giving us

$$(x^{\varphi(n)})^t \equiv (x^{(q-1)(p-1)})^t \equiv ((x^{\varphi(q)})^t)^{(p-1)} \equiv 1^{(p-1)} \equiv 1 \; (mod\ q)$$

  Now for some integer u this gives us

$$(x^{\varphi(n)})^t = 1 + u \cdot q$$

Continued on next page...

How does RSA Work? (cont.)

Proof.

- Case 2: $gcd(x, n) = gcd(x, p \cdot q) \neq 1$ (cont.)

  Multiplying both sides of the equality above by x gives us

$$
\begin{aligned}
(x^{\varphi(n)})^t \cdot x &= (1 + u \cdot q) \cdot x \\
&= x + x \cdot u \cdot q \\
&= x + (r \cdot p) \cdot u \cdot q \\
&= x + r \cdot u \cdot (p \cdot q) \\
&= x + r \cdot u \cdot n \\
(x^{\varphi(n)})^t \cdot x &\equiv x \, (mod\ n)
\end{aligned}
$$

How does RSA Work? (cont.)

Proof.

- Case 2: $gcd(x, n) = gcd(x, p \cdot q) \neq 1$ (cont.)

  Multiplying both sides of the equality above by x gives us

$$\begin{aligned}
(x^{\varphi(n)})^t \cdot x &= (1 + u \cdot q) \cdot x \\
&= x + x \cdot u \cdot q \\
&= x + (r \cdot p) \cdot u \cdot q \\
&= x + r \cdot u \cdot (p \cdot q) \\
&= x + r \cdot u \cdot n \\
(x^{\varphi(n)})^t \cdot x &\equiv x \,(mod\ n)
\end{aligned}$$

  Which was the desired result. Thus in either case we see that we have successfully decrypted our ciphertext x^e by exponentiating

$$(x^e)^d \equiv x^{1+t\cdot\varphi(n)} \equiv (x^{\varphi(n)})^t \cdot x \equiv x \,(mod\ n)$$

$\square$

Example of RSA with Small Numbers

Example

Public $= \{n, e\}$, Private $= \{p, q, d\}$)

1. Alice randomly chooses $p = 17$ and $q = 19$, so that $n = p \cdot q = 17 \cdot 19 = 323$

Example of RSA with Small Numbers

Example

Public $= \{n, e\}$, Private $= \{p, q, d\}$)

1. Alice randomly chooses $p = 17$ and $q = 19$, so that $n = p \cdot q = 17 \cdot 19 = 323$
2. Alice computes $\varphi(n) = \varphi(p \cdot q) = (p-1) \cdot (q-1) = 16 \cdot 18 = 288$

Example of RSA with Small Numbers

Example

Public = $\{n, e\}$, Private = $\{p, q, d\}$)

1. Alice randomly chooses $p = 17$ and $q = 19$, so that $n = p \cdot q = 17 \cdot 19 = 323$
2. Alice computes $\varphi(n) = \varphi(p \cdot q) = (p-1) \cdot (q-1) = 16 \cdot 18 = 288$
3. Alice randomly chooses an $1 < e < 288$ such that $gcd(e, 288) = 1$. We take $e = 95$

Example of RSA with Small Numbers

Example

Public $= \{n, e\}$, Private $= \{p, q, d\})$

1. Alice randomly chooses $p = 17$ and $q = 19$, so that $n = p \cdot q = 17 \cdot 19 = 323$
2. Alice computes $\varphi(n) = \varphi(p \cdot q) = (p-1) \cdot (q-1) = 16 \cdot 18 = 288$
3. Alice randomly chooses an $1 < e < 288$ such that $gcd(e, 288) = 1$. We take $e = 95$
4. Alice solves the linear congruence

$$95 \cdot d \equiv 1 \ (mod\ 288)$$

Using the Extended Euclidean Algorithm, we find that $d = 191$ solves the equation

Example of RSA with Small Numbers

Example

Public $= \{n, e\}$, Private $= \{p, q, d\}$)

1. Alice randomly chooses $p = 17$ and $q = 19$, so that $n = p \cdot q = 17 \cdot 19 = 323$
2. Alice computes $\varphi(n) = \varphi(p \cdot q) = (p-1) \cdot (q-1) = 16 \cdot 18 = 288$
3. Alice randomly chooses an $1 < e < 288$ such that $gcd(e, 288) = 1$. We take $e = 95$
4. Alice solves the linear congruence

$$95 \cdot d \equiv 1 \; (mod\; 288)$$

   Using the Extended Euclidean Algorithm, we find that $d = 191$ solves the equation

5. Alice shares the public key $(n, e) = (323, 95)$ with Bob
6. Bob can take an encoded message $x = 123$, encrypt it and send it to Alice

$$c \equiv x^e \equiv 123^{95} \equiv 149 \; (mod\; 323)$$

Example of RSA with Small Numbers

Example

Public $= \{n, e\}$, Private $= \{p, q, d\})$

1. Alice randomly chooses $p = 17$ and $q = 19$, so that $n = p \cdot q = 17 \cdot 19 = 323$
2. Alice computes $\varphi(n) = \varphi(p \cdot q) = (p-1) \cdot (q-1) = 16 \cdot 18 = 288$
3. Alice randomly chooses an $1 < e < 288$ such that $gcd(e, 288) = 1$. We take $e = 95$
4. Alice solves the linear congruence

$$95 \cdot d \equiv 1 \ (mod\ 288)$$

   Using the Extended Euclidean Algorithm, we find that $d = 191$ solves the equation
5. Alice shares the public key $(n, e) = (323, 95)$ with Bob
6. Bob can take an encoded message $x = 123$, encrypt it and send it to Alice

$$c \equiv x^e \equiv 123^{95} \equiv 149 \ (mod\ 323)$$

7. Alice decrypts by computing

$$x \equiv c^d \equiv (x^e)^d \equiv 149^{191} \equiv 123 \ (mod\ 323)$$

Example of RSA in SageMath

Implementation of the RSA Encryption Scheme.

```
In [1]: def encode(s):
            s = str(s) # make input a string
            return sum(ord(s[i])*256^i for i in range(len(s))) # Base 256 for ASCII
In [2]: def decode(n):
            n = Integer(n) # make input an integer
            v = []
            while n != 0:
                v.append(chr(n % 256))
                n = n//256 # this replaces n by floor(n/256)
            return ''.join(v)
In [34]: def rsa(bits):
            p = next_prime(ZZ.random_element(2^(bits)))
            q = next_prime(ZZ.random_element(2^(bits)))
            n = p*q
            phi_n = (p-1)*(q-1)

            while True:
                e = ZZ.random_element(1, phi_n)
                if gcd(e, phi_n) == 1: break
            d = lift(Mod(e, phi_n)^(-1))
            return e, d, n, p, q, phi_n
In [35]: def encrypt(m, e, n):
            return lift(Mod(m, n)^e)
In [36]: def decrypt(c, d, n):
            return lift(Mod(c, n)^d)
```

Let's looks at a live demo encrypting text using a large bit size

How to Break RSA

Currently the most promising approach to solving the RSA problem is to factor the modulus n. This is in general not an easy task.

How to Break RSA

Currently the most promising approach to solving the RSA problem is to factor the modulus n. This is in general not an easy task.

Procedure

1. Factor $n = p \cdot q$
2. Compute Euler's Totient function $\varphi(n) = (p-1) \cdot (q-1)$
3. Find a solution d to the equation $e \cdot d \equiv 1 \; (mod \; \varphi(n))$

- With the decryption key d in hand, any message captured can be decrypted

How to Break RSA

Currently the most promising approach to solving the RSA problem is to factor the modulus n. This is in general not an easy task.

Procedure

1. Factor $n = p \cdot q$
2. Compute Euler's Totient function $\varphi(n) = (p-1) \cdot (q-1)$
3. Find a solution d to the equation $e \cdot d \equiv 1 \; (mod \; \varphi(n))$

- With the decryption key d in hand, any message captured can be decrypted

Before we talk about factorization in general, let's explore a particular case when $\varphi(n)$ is known.

How to Break RSA (cont.)

Factoring $n = p \cdot q$ given n and $\varphi(n)$

1. Expand

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

How to Break RSA (cont.)

Factoring $n = p \cdot q$ given n and $\varphi(n)$

1. Expand

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

2. Rearrange

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

How to Break RSA (cont.)

Factoring $n = p \cdot q$ given n and $\varphi(n)$

1. Expand
$$\varphi(n) = \varphi(p \cdot q) = (p-1) \cdot (q-1) = p \cdot q - (p+q) + 1$$
2. Rearrange
$$p + q = p \cdot q + 1 - \varphi(n) = n + 1 - \varphi(n)$$
3. We now have a polynomial whose roots are precisely p and q
$$\begin{aligned} x^2 - (p+q)x + p \cdot q &= x^2 - (n + 1 - \varphi(n)) \cdot x + n \\ &= (x-p) \cdot (x-q) \end{aligned}$$

How to Break RSA (cont.)

Factoring $n = p \cdot q$ given n and $\varphi(n)$

1. Expand
$$\varphi(n) = \varphi(p \cdot q) = (p-1)\cdot(q-1) = p \cdot q - (p+q) + 1$$
2. Rearrange
$$p + q = p \cdot q + 1 - \varphi(n) = n + 1 - \varphi(n)$$
3. We now have a polynomial whose roots are precisely p and q
$$\begin{aligned} x^2 - (p+q)x + p \cdot q &= x^2 - (n + 1 - \varphi(n)) \cdot x + n \\ &= (x-p)\cdot(x-q) \end{aligned}$$
4. Plugging in the known values n and $\varphi(n)$ we find the roots using the quadratic formula

Let's do a live example of the implementation in SageMath

```
In [52]: def crack_rsa(n, phi_n):
             R.<x> = PolynomialRing(QQ)
             f = x^2 - (n+1 - phi_n)*x + n
             return [b for b, _ in f.roots()]
```

Factorization Problem

Factorization Problem (FP)

Given a positive integer n, find primes p_i and nonnegative e_i such that

$$n = \prod_{i=1}^{k} p_i^{e_i}$$

In the case of RSA, we are only concerned with integers of the form

$$n = p \cdot q$$

Example

- Factor $n = 4820624953600272 23$

Factorization Problem

Factorization Problem (FP)

Given a positive integer n, find primes p_i and nonnegative e_i such that

$$n = \prod_{i=1}^{k} p_i^{e_i}$$

In the case of RSA, we are only concerned with integers of the form

$$n = p \cdot q$$

Example

- Factor $n = 482062495360027223$
- Solution $p = 899910527$ and $q = 535678249$

Attacks Against the FP in RSA

Note: It can be shown that for $n = p \cdot q$ where $p \leq q$ we have $p \leq \sqrt{n}$

Brute Force Attack on RSA (Very slow)

Checking all integers

```
def factorBruteForce(n):
    s = Integer(int(round(sqrt(n))))
    r = 2
    while r < s:
        if n % r == 0:
            return r, n/r
        r += 1
```

Attacks Against the FP in RSA

Note: It can be shown that for $n = p \cdot q$ where $p \leq q$ we have $p \leq \sqrt{n}$

Brute Force Attack on RSA (Very slow)

Checking all integers

```
def factorBruteForce(n):
    s = Integer(int(round(sqrt(n))))
    r = 2
    while r < s:
        if n % r == 0:
            return r, n/r
        r += 1
```

Checking only primes

```
def factorBruteForcev2(n):
    s = Integer(int(round(sqrt(n))))

    for r in prime_range(2, s + 1):
        if n % r == 0:
            return r, n/r
```

Attacks Against the FP in RSA

Note: It can be shown that for $n = p \cdot q$ where $p \leq q$ we have $p \leq \sqrt{n}$

Brute Force Attack on RSA (Very slow)

Checking all integers

```
def factorBruteForce(n):
    s = Integer(int(round(sqrt(n))))
    r = 2
    while r < s:
        if n % r == 0:
            return r, n/r
        r += 1
```

Checking only primes

```
def factorBruteForcev2(n):
    s = Integer(int(round(sqrt(n))))

    for r in prime_range(2, s + 1):
        if n % r == 0:
            return r, n/r
```

It may appear that the implementation on right would be better. However, even SageMath throws the error "ValueError: Cannot compute primes beyond 436273290".

Attacks Against the FP in RSA (cont.)

When p and q are Close (Fermat Factorization Method)

1. It is widely known that any odd number can be written as a difference of two squares $n = a^2 - b^2 = (a + b) \cdot (a - b)$

Attacks Against the FP in RSA (cont.)

When p and q are Close (Fermat Factorization Method)

1. It is widely known that any odd number can be written as a difference of two squares $n = a^2 - b^2 = (a+b) \cdot (a-b)$
2. If $n = p \cdot q$ is a factorization then we have

$$n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

Attacks Against the FP in RSA (cont.)

When p and q are Close (Fermat Factorization Method)

1. It is widely known that any odd number can be written as a difference of two squares $n = a^2 - b^2 = (a+b) \cdot (a-b)$
2. If $n = p \cdot q$ is a factorization then we have

$$n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

3. Given that p and q are close, $s = \frac{p-q}{2}$ is small, and $t = \frac{p+q}{2}$ is slightly larger than $\sqrt{n}$

Attacks Against the FP in RSA (cont.)

When p and q are Close (Fermat Factorization Method)

1. It is widely known that any odd number can be written as a difference of two squares $n = a^2 - b^2 = (a+b) \cdot (a-b)$
2. If $n = p \cdot q$ is a factorization then we have

$$n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

3. Given that p and q are close, $s = \frac{p-q}{2}$ is small, and $t = \frac{p+q}{2}$ is slightly larger than $\sqrt{n}$
4. Substituting s and t into the above equation and rearranging we see that

$$t^2 - n = s^2$$

Attacks Against the FP in RSA (cont.)

When p and q are Close (Fermat Factorization Method)

1. It is widely known that any odd number can be written as a difference of two squares $n = a^2 - b^2 = (a+b) \cdot (a-b)$
2. If $n = p \cdot q$ is a factorization then we have

$$n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

3. Given that p and q are close, $s = \frac{p-q}{2}$ is small, and $t = \frac{p+q}{2}$ is slightly larger than $\sqrt{n}$
4. Substituting s and t into the above equation and rearranging we see that

$$t^2 - n = s^2$$

5. So we just try $t = \lceil \sqrt{n} \rceil, \ \lceil \sqrt{n} \rceil + 1, \ \lceil \sqrt{n} \rceil + 2, \ \ldots$ until $t^2 - n$ is a perfect square s^2

Attacks Against the FP in RSA (cont.)

When p and q are Close (Fermat Factorization Method)

1. It is widely known that any odd number can be written as a difference of two squares $n = a^2 - b^2 = (a+b)\cdot(a-b)$
2. If $n = p \cdot q$ is a factorization then we have

$$n = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

3. Given that p and q are close, $s = \frac{p-q}{2}$ is small, and $t = \frac{p+q}{2}$ is slightly larger than $\sqrt{n}$
4. Substituting s and t into the above equation and rearranging we see that

$$t^2 - n = s^2$$

5. So we just try $t = \lceil\sqrt{n}\rceil,\ \lceil\sqrt{n}\rceil + 1,\ \lceil\sqrt{n}\rceil + 2,\ \ldots$ until $t^2 - n$ is a perfect square s^2
6. By adding and subtracting $s = \frac{p-q}{2}$ and $t = \frac{p+q}{2}$ we get our formulas

$$p = t + s, \qquad q = t - s$$

Attacks Against the FP (cont.)

Let's look at a small example when p and q are close

Example

1. Take $n = 23360947609$, then $\lceil \sqrt{n} \approx 152842.88 \rceil = 152843$

Attacks Against the FP (cont.)

Let's look at a small example when p and q are close

Example

1. Take $n = 23360947609$, then $\lceil \sqrt{n} \approx 152842.88 \rceil = 152843$
2. Start checking
   - If $t = 152843$, then $s = \sqrt{t^2 - n} \approx 187.18$
   - If $t = 152844$, then $s = \sqrt{t^2 - n} \approx 583.71$
   - If $t = 152845$, then $s = \sqrt{t^2 - n} = 804$

Attacks Against the FP (cont.)

Let's look at a small example when p and q are close

Example

1. Take $n = 23360947609$, then $\lceil \sqrt{n} \approx 152842.88 \rceil = 152843$
2. Start checking
   - If $t = 152843$, then $s = \sqrt{t^2 - n} \approx 187.18$
   - If $t = 152844$, then $s = \sqrt{t^2 - n} \approx 583.71$
   - If $t = 152845$, then $s = \sqrt{t^2 - n} = 804$
3. Thus $p = t + s = 153649$ and $q = t - s = 152041$

Attacks Against the FP (cont.)

Let's look at a small example when p and q are close

Example

1. Take $n = 23360947609$, then $\lceil \sqrt{n} \approx 152842.88 \rceil = 152843$
2. Start checking
   - If $t = 152843$, then $s = \sqrt{t^2 - n} \approx 187.18$
   - If $t = 152844$, then $s = \sqrt{t^2 - n} \approx 583.71$
   - If $t = 152845$, then $s = \sqrt{t^2 - n} = 804$
3. Thus $p = t + s = 153649$ and $q = t - s = 152041$

Here is the implementation in SageMath, let's look at a live demo

```
def crack_when_pq_close(n):
    t = Integer(ceil(sqrt(n)))
    while True:
        k = t^2 - n
        if k <= 0: break
        if k > 0:
            s = Integer(int(round(sqrt(t^2 - n))))
            if s^2 + n == t^2:
                return t + s, t - s
        t += 1
```

Closing Remarks on RSA

- The best known attacks against RSA are
  - General number field sieve for classical computers
  - Shor's algorithm for quantum computers

Closing Remarks on RSA

- The best known attacks against RSA are
  - General number field sieve for classical computers
  - Shor's algorithm for quantum computers
- It is recommended to use 617 decimal digit numbers, or 2048 bits for the modulus

Closing Remarks on RSA

- The best known attacks against RSA are
  - General number field sieve for classical computers
  - Shor's algorithm for quantum computers
- It is recommended to use 617 decimal digit numbers, or 2048 bits for the modulus
- The largest RSA number factored to date is RSA-250, has 250 decimal digits (829 bits), and was factored in February 2020

Closing Remarks on RSA

- The best known attacks against RSA are
  - General number field sieve for classical computers
  - Shor's algorithm for quantum computers
- It is recommended to use 617 decimal digit numbers, or 2048 bits for the modulus
- The largest RSA number factored to date is RSA-250, has 250 decimal digits (829 bits), and was factored in February 2020
- The next largest RSA number is RSA-260 (862 bits)

  2211282552952966643528108525502623092761208950247001539441374831912882
  2941402001986512729726569746599085900330031400051170742204560859276357
  9537571859542988389587092292384910067030341246205457845664136645406842
  14361293017694020846391065875914794251435144458199

List of Records for factoring RSA numbers
`https://en.wikipedia.org/wiki/RSA_numbers#RSA-2048`

Closing Remarks

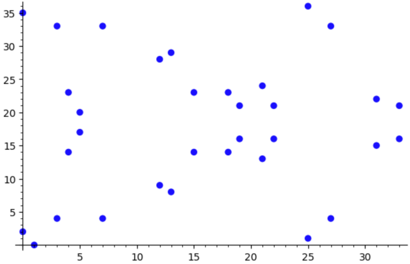
- Elliptic Curve Cryptography
  - Gives us higher security with smaller keys for Diffie-Hellman

Closing Remarks

- Elliptic Curve Cryptography
  - Gives us higher security with smaller keys for Diffie-Hellman
  - Factorization algorithms use Elliptic Curves
    - ★ Pollards (p - 1) Method
    - ★ Lenstra's Method

Closing Remarks

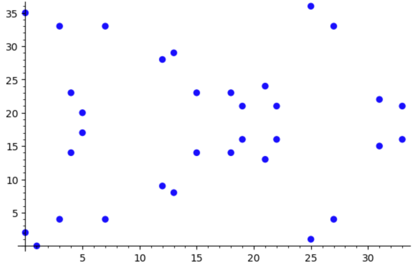
- Elliptic Curve Cryptography
  - Gives us higher security with smaller keys for Diffie-Hellman
  - Factorization algorithms use Elliptic Curves
    - ★ Pollards (p - 1) Method
    - ★ Lenstra's Method
  - $E(K) = \{(x, y) \in K \times K : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$
    - ★ K a finite field, E an elliptic curve, $\mathcal{O}$ identity element



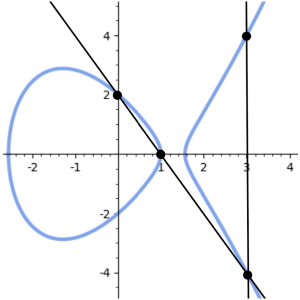
E: $y^2 = x^3 + -5x + 4$ $\quad K = \mathbb{Z}_{37}$

Closing Remarks

- Elliptic Curve Cryptography
  - Gives us higher security with smaller keys for Diffie-Hellman
  - Factorization algorithms use Elliptic Curves
    - ★ Pollards (p - 1) Method
    - ★ Lenstra's Method
  - $E(K) = \{(x, y) \in K \times K : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$
    - ★ K a finite field, E an elliptic curve, $\mathcal{O}$ identity element
  - Operation is Point Addition



E: $y^2 = x^3 + -5x + 4$ $\quad K = \mathbb{Z}_{37}$



$(1, 0) + (0, 2) = (3, 4)$

Closing Remarks (cont.)

- The Future of Computer Security
  - Cryptography is a fast paced field with many current developments. The security of our technology depends on it

Closing Remarks (cont.)

- The Future of Computer Security
  - Cryptography is a fast paced field with many current developments. The security of our technology depends on it
  - As quantum computers are further developed, certain security schemes in use may become vulnerable to attack

Closing Remarks (cont.)

- The Future of Computer Security
  - Cryptography is a fast paced field with many current developments. The security of our technology depends on it
  - As quantum computers are further developed, certain security schemes in use may become vulnerable to attack
  - Shor's algorithm which is shown to work on quantum computers may pose serious threats to RSA and Diffie-Hellman

Closing Remarks (cont.)

- The Future of Computer Security
  - Cryptography is a fast paced field with many current developments. The security of our technology depends on it
  - As quantum computers are further developed, certain security schemes in use may become vulnerable to attack
  - Shor's algorithm which is shown to work on quantum computers may pose serious threats to RSA and Diffie-Hellman
- Further Areas to Explore
  - Classical algorithms used to speed up the factorization problem and discrete log problem
    - Quadratic Sieve (under 100 digits)
    - General Number Field Sieve (integers larger than 10^{100})

Closing Remarks (cont.)

- The Future of Computer Security
  - Cryptography is a fast paced field with many current developments. The security of our technology depends on it
  - As quantum computers are further developed, certain security schemes in use may become vulnerable to attack
  - Shor's algorithm which is shown to work on quantum computers may pose serious threats to RSA and Diffie-Hellman
- Further Areas to Explore
  - Classical algorithms used to speed up the factorization problem and discrete log problem
    - Quadratic Sieve (under 100 digits)
    - General Number Field Sieve (integers larger than 10^{100})
  - Elliptic Curve Algorithms

Closing Remarks (cont.)

- The Future of Computer Security
  - Cryptography is a fast paced field with many current developments. The security of our technology depends on it
  - As quantum computers are further developed, certain security schemes in use may become vulnerable to attack
  - Shor's algorithm which is shown to work on quantum computers may pose serious threats to RSA and Diffie-Hellman
- Further Areas to Explore
  - Classical algorithms used to speed up the factorization problem and discrete log problem
    - Quadratic Sieve (under 100 digits)
    - General Number Field Sieve (integers larger than 10^{100})
  - Elliptic Curve Algorithms
  - Quantum Computers as it relates to cryptography
    - Shor's Algorithm

Thank you, I hope you all stay safe and well!

Carman S. Cater

References

Buchanan, William J., *Cryptography* , Rivers Publishers, 2017. Available electronically through CCSU library.

Stein, William, *Elementary Number Theory: Primes, Congruences, and Secrets* , Springer, 2009. Available free from author https://wstein.org/ent/.

Paar, Christof; Pelzl, Jan, *Understanding Cryptography: A Textbook for Students and Practioners* , Springer, 2010.

History of Cryptography; https://en.wikipedia.org/wiki/History_of_cryptography

Image Links

Diffie and Hellman:
https://news.stanford.edu/news/2016/march/images/16185-turingtwo_news.jpg

Elgamal: https://evolutionequity.com/team/taher-elgamal

Rivest, Shamir, Adleman:
https://cdn.firespring.com/images/bf650823-bb00-4999-ad53-30b967fe948d.jpg

Al-Kindi: https://en.wikipedia.org/wiki/File:Al-kindi_cryptographic.gif

ROT13: https://en.wikipedia.org/wiki/Substitution_cipher#/media/File:ROT13.png

M-94: https://en.wikipedia.org/wiki/M-94#/media/File:Ytm94_1b.jpg

Enigma: https://en.wikipedia.org/wiki/Enigma_machine#/media/File:
Enigma_(crittografia)_-_Museo_scienza_e_tecnologia_Milano.jpg