The RSA Cryptosystem

R. Balasubramanian

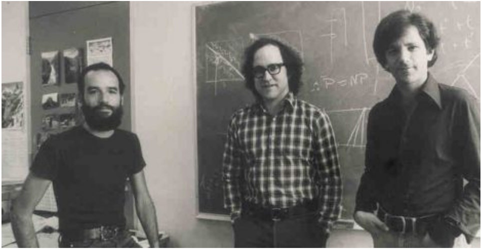
March 17, 2025

Plan

1. History and Uses
2. RSA Description
   - PKE
   - Decryption is Correct
   - Computational Issues
3. Security
   - Textbook RSA is not Semantically Secure
   - Fixes

RSA – History and Uses

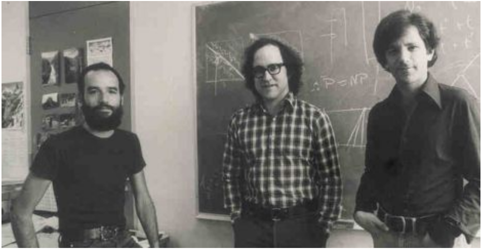
History



Invented by Ron Rivest, Adi Shamir and Len Adleman; published in *Scientific American* in 1977

Built on ‘shaky’ grounds, is unbroken till now

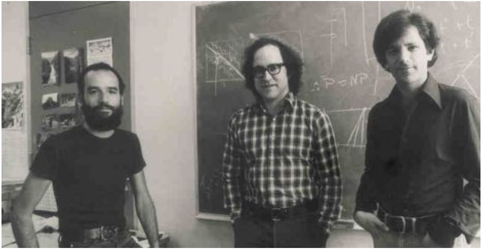
History



Invented by Ron Rivest, Adi Shamir and Len Adleman; published in *Scientific American* in 1977

Built on ‘shaky’ grounds, is unbroken till now

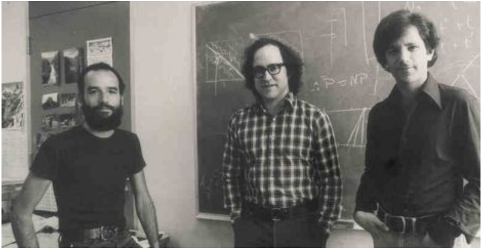
History



Invented by Ron Rivest, Adi Shamir and Len Adleman; published in *Scientific American* in 1977

Built on ‘shaky’ grounds, is unbroken till now

History



Invented by Ron Rivest, Adi Shamir and Len Adleman; published in *Scientific American* in 1977

Built on ‘shaky’ grounds, is unbroken till now

History

Diffie and Hellman in 1976 propose a new method for key exchange

Used a so-called trap-door one-way function

Easy to compute in one direction but ‘hard’ to invert

Diffie and Hellman suggested that a similar idea may be used to provide public key encryption and authentication scheme

That set R-S-A in action

A public key encryption scheme

History

Diffie and Hellman in 1976 propose a new method for key exchange

Used a so-called trap-door one-way function

Easy to compute in one direction but ‘hard’ to invert

Diffie and Hellman suggested that a similar idea may be used to provide public key encryption and authentication scheme

That set R-S-A in action

A public key encryption scheme

History

Diffie and Hellman in 1976 propose a new method for key exchange

Used a so-called trap-door one-way function

Easy to compute in one direction but ‘hard’ to invert

Diffie and Hellman suggested that a similar idea may be used to provide public key encryption and authentication scheme

That set R-S-A in action

A public key encryption scheme

History

Diffie and Hellman in 1976 propose a new method for key exchange

Used a so-called trap-door one-way function

Easy to compute in one direction but ‘hard’ to invert

Diffie and Hellman suggested that a similar idea may be used to provide public key encryption and authentication scheme

That set R-S-A in action

A public key encryption scheme

History

Diffie and Hellman in 1976 propose a new method for key exchange

Used a so-called trap-door one-way function

Easy to compute in one direction but ‘hard’ to invert

Diffie and Hellman suggested that a similar idea may be used to provide public key encryption and authentication scheme

That set R-S-A in action

A public key encryption scheme

History

Diffie and Hellman in 1976 propose a new method for key exchange

Used a so-called trap-door one-way function

Easy to compute in one direction but ‘hard’ to invert

Diffie and Hellman suggested that a similar idea may be used to provide public key encryption and authentication scheme

That set R-S-A in action

A public key encryption scheme

Where is RSA Used?

Web servers and browsers to secure web traffic

Privacy and authenticity of email

Secure remote login sessions

Electronic payment system

Can you find out more and exactly how?

Where is RSA Used?

Web servers and browsers to secure web traffic

Privacy and authenticity of email

Secure remote login sessions

Electronic payment system

Can you find out more and exactly how?

Where is RSA Used?

Web servers and browsers to secure web traffic

Privacy and authenticity of email

Secure remote login sessions

Electronic payment system

Can you find out more and exactly how?

Where is RSA Used?

Web servers and browsers to secure web traffic

Privacy and authenticity of email

Secure remote login sessions

Electronic payment system

Can you find out more and exactly how?

Where is RSA Used?

Web servers and browsers to secure web traffic

Privacy and authenticity of email

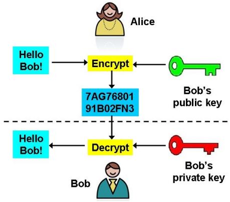
Secure remote login sessions

Electronic payment system

Can you find out more and exactly how?

RSA Encryption – Description

Public Key Encryption



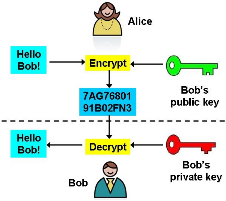
(WIKI)

Public Key Encryption Scheme

A triple of PPT algorithms (G, E, D)

1. $G(1^\lambda) \leftarrow (Pk, Sk)$
2. $E(Pk, m) \leftarrow CT$
3. $D(Sk, CT) \leftarrow m$. (E and D are consistent)

Public Key Encryption



(WIKI)

Public Key Encryption Scheme

A triple of PPT algorithms (G, E, D)

1. $G(1^\lambda) \leftarrow (Pk, Sk)$
2. $E(Pk, m) \leftarrow CT$
3. $D(Sk, CT) \leftarrow m$. (E and D are consistent)

Textbook RSA

- Setup(1^λ): Choose two large primes p, q and set $n = pq$. Let $\phi(n) := (p-1)(q-1)$.

  Choose an odd number e s.t $(e, \phi(n)) = 1$. Compute d s.t $ed \equiv 1 \mod (\phi(n))$.

  Publish $Pk = (e, n)$ and keep secret $Sk = d$.

- Encrypt(Pk, m): The sender encrypts any $m < n$ as

$$CT = m^e \mod (n).$$

- Decrypt(Sk, CT): The receiver decrypts CT as

$$m = CT^d \mod (n).$$

Textbook RSA

- Setup(1^λ): Choose two large primes p, q and set $n = pq$. Let $\phi(n) := (p-1)(q-1)$.

  Choose an odd number e s.t $(e, \phi(n)) = 1$. Compute d s.t $ed \equiv 1 \mod (\phi(n))$.

  Publish $Pk = (e, n)$ and keep secret $Sk = d$.

- Encrypt(Pk, m): The sender encrypts any $m < n$ as

$$CT = m^e \mod (n).$$

- Decrypt(Sk, CT): The receiver decrypts CT as

$$m = CT^d \mod (n).$$

Textbook RSA

- Setup(1^λ): Choose two large primes p, q and set $n = pq$. Let $\phi(n) := (p-1)(q-1)$.

  Choose an odd number e s.t $(e, \phi(n)) = 1$. Compute d s.t $ed \equiv 1 \mod (\phi(n))$.

  Publish $Pk = (e, n)$ and keep secret $Sk = d$.

- Encrypt(Pk, m): The sender encrypts any $m < n$ as

$$CT = m^e \mod (n).$$

- Decrypt(Sk, CT): The receiver decrypts CT as

$$m = CT^d \mod (n).$$

Textbook RSA

- Setup(1^λ): Choose two large primes p, q and set $n = pq$. Let $\phi(n) := (p-1)(q-1)$.

  Choose an odd number e s.t $(e, \phi(n)) = 1$. Compute d s.t $ed \equiv 1 \mod (\phi(n))$.

  Publish $Pk = (e, n)$ and keep secret $Sk = d$.

- Encrypt(Pk, m): The sender encrypts any $m < n$ as

$$CT = m^e \mod (n).$$

- Decrypt(Sk, CT): The receiver decrypts CT as

$$m = CT^d \mod (n).$$

Textbook RSA

- Setup(1^λ): Choose two large primes p, q and set $n = pq$. Let $\phi(n) := (p-1)(q-1)$.

  Choose an odd number e s.t $(e, \phi(n)) = 1$. Compute d s.t $ed \equiv 1 \mod (\phi(n))$.

  Publish $Pk = (e, n)$ and keep secret $Sk = d$.

- Encrypt(Pk, m): The sender encrypts any $m < n$ as

$$CT = m^e \mod (n).$$

- Decrypt(Sk, CT): The receiver decrypts CT as

$$m = CT^d \mod (n).$$

Euler Connection

Euler ϕ function: Is a arithmetic function equalling the number of positive integers less than n and relatively prime with n.

For p, prime $\phi(p) = p - 1$, for $n = pq$, where $p,\ q$ are prime $\phi(n) = (p-1)(q-1)$.

Euler's Theorem: If n and a are coprime then

$$a^{\phi(n)} \equiv 1 \mod (n).$$

The set of all $y < n$ with $(y, n) = 1$, say $\{a_1, \ldots, a_{\phi(n)}\}$ equals $\{a \cdot a_1, \ldots, a \cdot a_{\phi(n)}\}$. (Why?)

So $\prod_i a_i = \prod_i a \cdot a_i = a^{\phi(n)} \prod_i a_i \mod (n)$.

Euler Connection

Euler ϕ function: Is a arithmetic function equalling the number of positive integers less than n and relatively prime with n.

For p, prime $\phi(p) = p - 1$, for $n = pq$, where $p,\ q$ are prime $\phi(n) = (p-1)(q-1)$.

Euler's Theorem: If n and a are coprime then

$$a^{\phi(n)} \equiv 1 \mod (n).$$

The set of all $y < n$ with $(y, n) = 1$, say $\{a_1, \ldots, a_{\phi(n)}\}$ equals $\{a \cdot a_1, \ldots, a \cdot a_{\phi(n)}\}$. (Why?)

So $\prod_i a_i = \prod_i a \cdot a_i = a^{\phi(n)} \prod_i a_i \mod (n)$.

Euler Connection

Euler ϕ function: Is a arithmetic function equalling the number of positive integers less than n and relatively prime with n.

For p, prime $\phi(p) = p - 1$, for $n = pq$, where $p,\ q$ are prime $\phi(n) = (p-1)(q-1)$.

Euler's Theorem: If n and a are coprime then

$$a^{\phi(n)} \equiv 1 \mod (n).$$

The set of all $y < n$ with $(y, n) = 1$, say $\{a_1, \ldots, a_{\phi(n)}\}$ equals $\{a \cdot a_1, \ldots, a \cdot a_{\phi(n)}\}$. (Why?)

So $\prod_i a_i = \prod_i a \cdot a_i = a^{\phi(n)} \prod_i a_i \mod (n)$.

Euler Connection

Euler ϕ function: Is a arithmetic function equalling the number of positive integers less than n and relatively prime with n.

For p, prime $\phi(p) = p - 1$, for $n = pq$, where $p,\ q$ are prime $\phi(n) = (p-1)(q-1)$.

Euler's Theorem: If n and a are coprime then

$$a^{\phi(n)} \equiv 1 \mod (n).$$

The set of all $y < n$ with $(y, n) = 1$, say $\{a_1, \ldots, a_{\phi(n)}\}$ equals $\{a \cdot a_1, \ldots, a \cdot a_{\phi(n)}\}$. (Why?)

So $\prod_i a_i = \prod_i a \cdot a_i = a^{\phi(n)} \prod_i a_i \mod (n)$.

Euler Connection

Euler ϕ function: Is a arithmetic function equalling the number of positive integers less than n and relatively prime with n.

For p, prime $\phi(p) = p - 1$, for $n = pq$, where $p,\ q$ are prime $\phi(n) = (p-1)(q-1)$.

Euler's Theorem: If n and a are coprime then

$$a^{\phi(n)} \equiv 1 \mod (n).$$

The set of all $y < n$ with $(y, n) = 1$, say $\{a_1, \ldots, a_{\phi(n)}\}$ equals $\{a \cdot a_1, \ldots, a \cdot a_{\phi(n)}\}$. (Why?)

So $\prod_i a_i = \prod_i a \cdot a_i = a^{\phi(n)} \prod_i a_i \mod (n)$.

Decryption is Correct

RSA Decryption Theorem

For almost all m, we have $m = CT^d \mod (n)$.

Let $ed = k\phi(n) + 1$.

So $CT^d = m^{ed} = m^{k\phi(n)+1} \mod (n)$.

If $(m, n) = 1$, we can apply Euler theorem and obtain the required result.

Otherwise, decryption can not be done. But this happens with very less probability.

(When does this happen?)

Decryption is Correct

RSA Decryption Theorem

For almost all m, we have $m = CT^d \mod (n)$.

Let $ed = k\phi(n) + 1$.

So $CT^d = m^{ed} = m^{k\phi(n)+1} \mod (n)$.

If $(m, n) = 1$, we can apply Euler theorem and obtain the required result.

Otherwise, decryption can not be done. But this happens with very less probability.

(When does this happen?)

Decryption is Correct

RSA Decryption Theorem

For almost all m, we have $m = CT^d \mod (n)$.

Let $ed = k\phi(n) + 1$.

So $CT^d = m^{ed} = m^{k\phi(n)+1} \mod (n)$.

If $(m, n) = 1$, we can apply Euler theorem and obtain the required result.

Otherwise, decryption can not be done. But this happens with very less probability.

(When does this happen?)

Decryption is Correct

RSA Decryption Theorem

For almost all m, we have $m = CT^d \mod (n)$.

Let $ed = k\phi(n) + 1$.

So $CT^d = m^{ed} = m^{k\phi(n)+1} \mod (n)$.

If $(m, n) = 1$, we can apply Euler theorem and obtain the required result.

Otherwise, decryption can not be done. But this happens with very less probability.

(When does this happen?)

Decryption is Correct

RSA Decryption Theorem

For almost all m, we have $m = CT^d \mod (n)$.

Let $ed = k\phi(n) + 1$.

So $CT^d = m^{ed} = m^{k\phi(n)+1} \mod (n)$.

If $(m, n) = 1$, we can apply Euler theorem and obtain the required result.

Otherwise, decryption can not be done. But this happens with very less probability.

(When does this happen?)

Decryption is Correct

RSA Decryption Theorem

For almost all m, we have $m = CT^d \mod (n)$.

Let $ed = k\phi(n) + 1$.

So $CT^d = m^{ed} = m^{k\phi(n)+1} \mod (n)$.

If $(m, n) = 1$, we can apply Euler theorem and obtain the required result.

Otherwise, decryption can not be done. But this happens with very less probability.

(When does this happen?)

Computational Issues

- Choosing Primes: Choose a random integer and use AKS primality test algorithm.
- Testing $(e, \phi(n)) = 1$ and inverting e to obtain d: Inverse of a modulo m exists if and only if $(a, m) = 1$.

  Use (extended) Euclidean algorithm. Find $x,\ y$ such that $ex + \phi(n)y = 1$.
- Modular Exponentiation: Square-and-multiply.

Computational Issues

- Choosing Primes: Choose a random integer and use AKS primality test algorithm.

- Testing $(e, \phi(n)) = 1$ and inverting e to obtain d: Inverse of a modulo m exists if and only if $(a, m) = 1$.

  Use (extended) Euclidean algorithm. Find x, y such that $ex + \phi(n)y = 1$.

- Modular Exponentiation: Square-and-multiply.

Computational Issues

- Choosing Primes: Choose a random integer and use AKS primality test algorithm.

- Testing $(e, \phi(n)) = 1$ and inverting e to obtain d: Inverse of a modulo m exists if and only if $(a, m) = 1$.

  Use (extended) Euclidean algorithm. Find x, y such that $ex + \phi(n)y = 1$.

- Modular Exponentiation: Square-and-multiply.

Computational Issues

- Choosing Primes: Choose a random integer and use AKS primality test algorithm.

- Testing $(e, \phi(n)) = 1$ and inverting e to obtain d: Inverse of a modulo m exists if and only if $(a, m) = 1$.

  Use (extended) Euclidean algorithm. Find $x,\ y$ such that $ex + \phi(n)y = 1$.

- Modular Exponentiation: Square-and-multiply.

Choosing Primes

Choose a random positive integer of requisite size

Use AKS (deterministic) polytime algorithm to test whether it is prime

There also exist some randomized algorithms which can be used

Theorems about distribution of primes guarantee that you are bound to succeed

$$\pi(N) \approx N/\log(N)$$

Choosing Primes

Choose a random positive integer of requisite size

Use AKS (deterministic) polytime algorithm to test whether it is prime

There also exist some randomized algorithms which can be used

Theorems about distribution of primes guarantee that you are bound to succeed

$$\pi(N) \approx N/\log(N)$$

Choosing Primes

Choose a random positive integer of requisite size

Use AKS (deterministic) polytime algorithm to test whether it is prime

There also exist some randomized algorithms which can be used

Theorems about distribution of primes guarantee that you are bound to succeed

$$\pi(N) \approx N/\log(N)$$

Choosing Primes

Choose a random positive integer of requisite size

Use AKS (deterministic) polytime algorithm to test whether it is prime

There also exist some randomized algorithms which can be used

Theorems about distribution of primes guarantee that you are bound to succeed

$$\pi(N) \approx N/\log(N)$$

Extended Euclidean Algorithm

Given $a > b$ GCD such that $(a, b) = ax + by$

$r_0 = a,\ r_1 = b,\ r_{i+1} = r_{i-1} - q_i r_i$

$s_1 = a,\ s_1 = 0,\ s_{i+1} = s_{i-1} - q_i s_i$ and
$t_0 = 0,\ t_1 = 1,\ t_{i+1} = t_{i-1} - q_i t_i$

End if $r_{k+1} = 0$. Then $(a, b) = r_k = s_k a + t_k b$

EEA Correctness

Given a, b EEA outputs x, y, d such that

$$d = (a, b) = ax + by.$$

$\{r_i\}$ is decreasing, so the algorithm terminates

$(r_{i-1}, r_i) = (r_i, r_{i+1})$ and $as_i + bt_i = r_i$

Extended Euclidean Algorithm

Given $a > b$ GCD such that $(a,b) = ax + by$

$r_0 = a,\ r_1 = b,\ r_{i+1} = r_{i-1} - q_i r_i$

$s_1 = a,\ s_1 = 0,\ s_{i+1} = s_{i-1} - q_i s_i$ and
$t_0 = 0,\ t_1 = 1,\ t_{i+1} = t_{i-1} - q_i t_i$

End if $r_{k+1} = 0$. Then $(a,b) = r_k = s_k a + t_k b$

EEA Correctness

Given a, b EEA outputs x, y, d such that

$$d = (a,b) = ax + by.$$

$\{r_i\}$ is decreasing, so the algorithm terminates

$(r_{i-1}, r_i) = (r_i, r_{i+1})$ and $as_i + bt_i = r_i$

Extended Euclidean Algorithm

Given $a > b$ GCD such that $(a, b) = ax + by$

$r_0 = a,\ r_1 = b,\ r_{i+1} = r_{i-1} - q_i r_i$

$s_1 = a,\ s_1 = 0,\ s_{i+1} = s_{i-1} - q_i s_i$ and
$t_0 = 0,\ t_1 = 1,\ t_{i+1} = t_{i-1} - q_i t_i$

End if $r_{k+1} = 0$. Then $(a, b) = r_k = s_k a + t_k b$

EEA Correctness

Given a, b EEA outputs x, y, d such that

$$d = (a, b) = ax + by.$$

$\{r_i\}$ is decreasing, so the algorithm terminates

$(r_{i-1}, r_i) = (r_i, r_{i+1})$ and $as_i + bt_i = r_i$

Extended Euclidean Algorithm

Given $a > b$ GCD such that $(a, b) = ax + by$

$r_0 = a,\ r_1 = b,\ r_{i+1} = r_{i-1} - q_i r_i$

$s_1 = a,\ s_1 = 0,\ s_{i+1} = s_{i-1} - q_i s_i$ and
$t_0 = 0,\ t_1 = 1,\ t_{i+1} = t_{i-1} - q_i t_i$

End if $r_{k+1} = 0$. Then $(a, b) = r_k = s_k a + t_k b$

EEA Correctness

Given a, b EEA outputs x, y, d such that

$$d = (a, b) = ax + by.$$

$\{r_i\}$ is decreasing, so the algorithm terminates

$(r_{i-1}, r_i) = (r_i, r_{i+1})$ and $as_i + bt_i = r_i$

Extended Euclidean Algorithm

Given $a > b$ GCD such that $(a, b) = ax + by$

$r_0 = a, \ r_1 = b, \ r_{i+1} = r_{i-1} - q_i r_i$

$s_1 = a, \ s_1 = 0, \ s_{i+1} = s_{i-1} - q_i s_i$ and
$t_0 = 0, \ t_1 = 1, \ t_{i+1} = t_{i-1} - q_i t_i$

End if $r_{k+1} = 0$. Then $(a, b) = r_k = s_k a + t_k b$

EEA Correctness

Given a, b EEA outputs x, y, d such that

$$d = (a, b) = ax + by.$$

$\{r_i\}$ is decreasing, so the algorithm terminates

$(r_{i-1}, r_i) = (r_i, r_{i+1})$ and $as_i + bt_i = r_i$

Extended Euclidean Algorithm

Given $a > b$ GCD such that $(a, b) = ax + by$

$r_0 = a,\ r_1 = b,\ r_{i+1} = r_{i-1} - q_i r_i$

$s_1 = a,\ s_1 = 0,\ s_{i+1} = s_{i-1} - q_i s_i$ and
$t_0 = 0,\ t_1 = 1,\ t_{i+1} = t_{i-1} - q_i t_i$

End if $r_{k+1} = 0$. Then $(a, b) = r_k = s_k a + t_k b$

EEA Correctness

Given a, b EEA outputs x, y, d such that

$$d = (a, b) = ax + by.$$

$\{r_i\}$ is decreasing, so the algorithm terminates

$(r_{i-1}, r_i) = (r_i, r_{i+1})$ and $as_i + bt_i = r_i$

Extended Euclidean Algorithm

Given $a > b$ GCD such that $(a,b) = ax + by$

$r_0 = a,\ r_1 = b,\ r_{i+1} = r_{i-1} - q_i r_i$

$s_1 = a,\ s_1 = 0,\ s_{i+1} = s_{i-1} - q_i s_i$ and
$t_0 = 0,\ t_1 = 1,\ t_{i+1} = t_{i-1} - q_i t_i$

End if $r_{k+1} = 0$. Then $(a,b) = r_k = s_k a + t_k b$

EEA Correctness

Given a, b EEA outputs x, y, d such that

$$d = (a,b) = ax + by.$$

$\{r_i\}$ is decreasing, so the algorithm terminates

$(r_{i-1}, r_i) = (r_i, r_{i+1})$ and $as_i + bt_i = r_i$

Extended Euclidean Algorithm – Illustation

Let us compute $(240, 46)$

$240 = 5 \times 46 + 10, \; s = 1, \; t = -5$

$46 = 4 \times 10 + 6, \; s = -4, \; t = 21$

$10 = 1 \times 6 + 4, \; s = 5, \; t = -26$

$6 = 1 \times 4 + 2, \; s = -9, \; t = 47$

$4 = 2 \times 2 + 0, \; s = 23, \; t = -120$

$2 = -9 \times 240 + 47 \times 46$

Extended Euclidean Algorithm – Illustation

Let us compute $(240, 46)$

$240 = 5 \times 46 + 10, \ s = 1, \ t = -5$

$46 = 4 \times 10 + 6, \ s = -4, \ t = 21$

$10 = 1 \times 6 + 4, \ s = 5, \ t = -26$

$6 = 1 \times 4 + 2, \ s = -9, \ t = 47$

$4 = 2 \times 2 + 0, \ s = 23, \ t = -120$

$2 = -9 \times 240 + 47 \times 46$

Extended Euclidean Algorithm – Illustation

Let us compute $(240, 46)$

$240 = 5 \times 46 + 10,\ s = 1,\ t = -5$

$46 = 4 \times 10 + 6,\ s = -4,\ t = 21$

$10 = 1 \times 6 + 4,\ s = 5,\ t = -26$

$6 = 1 \times 4 + 2,\ s = -9,\ t = 47$

$4 = 2 \times 2 + 0,\ s = 23,\ t = -120$

$2 = -9 \times 240 + 47 \times 46$

Extended Euclidean Algorithm – Illustation

Let us compute $(240, 46)$

$240 = 5 \times 46 + 10,\ s = 1,\ t = -5$

$46 = 4 \times 10 + 6,\ s = -4,\ t = 21$

$10 = 1 \times 6 + 4,\ s = 5,\ t = -26$

$6 = 1 \times 4 + 2,\ s = -9,\ t = 47$

$4 = 2 \times 2 + 0,\ s = 23,\ t = -120$

$2 = -9 \times 240 + 47 \times 46$

Extended Euclidean Algorithm – Illustation

Let us compute $(240, 46)$

$240 = 5 \times 46 + 10, \; s = 1, \; t = -5$

$46 = 4 \times 10 + 6, \; s = -4, \; t = 21$

$10 = 1 \times 6 + 4, \; s = 5, \; t = -26$

$6 = 1 \times 4 + 2, \; s = -9, \; t = 47$

$4 = 2 \times 2 + 0, \; s = 23, \; t = -120$

$2 = -9 \times 240 + 47 \times 46$

Extended Euclidean Algorithm – Illustation

Let us compute $(240, 46)$

$240 = 5 \times 46 + 10,\ s = 1,\ t = -5$

$46 = 4 \times 10 + 6,\ s = -4,\ t = 21$

$10 = 1 \times 6 + 4,\ s = 5,\ t = -26$

$6 = 1 \times 4 + 2,\ s = -9,\ t = 47$

$4 = 2 \times 2 + 0,\ s = 23,\ t = -120$

$2 = -9 \times 240 + 47 \times 46$

Extended Euclidean Algorithm – Illustation

Let us compute $(240, 46)$

$240 = 5 \times 46 + 10,\ s = 1,\ t = -5$

$46 = 4 \times 10 + 6,\ s = -4,\ t = 21$

$10 = 1 \times 6 + 4,\ s = 5,\ t = -26$

$6 = 1 \times 4 + 2,\ s = -9,\ t = 47$

$4 = 2 \times 2 + 0,\ s = 23,\ t = -120$

$2 = -9 \times 240 + 47 \times 46$

Modular Exponentiation

Say you want to compute $5^{37} \mod (19)$

$37 = 2^5 + 2^2 + 2^0$

So $5^{37} = 5^{2^0} \times 5^{2^2} \times 5^{2^5} \mod (19)$

Keep squaring $5 \mod (19)$ and multiply requisite terms: 5^1, 5^2, $(5^2)^2 = 5^4$, $(5^4)^2 = 5^8$, $(5^8)^2 = 5^{16}$, $(5^{16})^2 = 5^{32}$ all modulo 19

Write down the algorithm and estimate the complexity of this method

Other methods like Motgomery laddering are also used

Modular Exponentiation

Say you want to compute $5^{37} \mod (19)$

$37 = 2^5 + 2^2 + 2^0$

So $5^{37} = 5^{2^0} \times 5^{2^2} \times 5^{2^5} \mod (19)$

Keep squaring $5 \mod (19)$ and multiply requisite terms: 5^1, 5^2, $(5^2)^2 = 5^4$, $(5^4)^2 = 5^8$, $(5^8)^2 = 5^{16}$, $(5^{16})^2 = 5^{32}$ all modulo 19

Write down the algorithm and estimate the complexity of this method

Other methods like Motgomery laddering are also used

Modular Exponentiation

Say you want to compute $5^{37} \mod (19)$

$37 = 2^5 + 2^2 + 2^0$

So $5^{37} = 5^{2^0} \times 5^{2^2} \times 5^{2^5} \mod (19)$

Keep squaring $5 \mod (19)$ and multiply requisite terms: 5^1, 5^2, $(5^2)^2 = 5^4$, $(5^4)^2 = 5^8$, $(5^8)^2 = 5^{16}$, $(5^{16})^2 = 5^{32}$ all modulo 19

Write down the algorithm and estimate the complexity of this method

Other methods like Motgomery laddering are also used

Modular Exponentiation

Say you want to compute $5^{37} \mod (19)$

$37 = 2^5 + 2^2 + 2^0$

So $5^{37} = 5^{2^0} \times 5^{2^2} \times 5^{2^5} \mod (19)$

Keep squaring 5 mod (19) and multiply requisite terms: 5^1, 5^2, $(5^2)^2 = 5^4$, $(5^4)^2 = 5^8$, $(5^8)^2 = 5^{16}$, $(5^{16})^2 = 5^{32}$ all modulo 19

Write down the algorithm and estimate the complexity of this method

Other methods like Motgomery laddering are also used

Modular Exponentiation

Say you want to compute $5^{37} \mod (19)$

$37 = 2^5 + 2^2 + 2^0$

So $5^{37} = 5^{2^0} \times 5^{2^2} \times 5^{2^5} \mod (19)$

Keep squaring $5 \mod (19)$ and multiply requisite terms: 5^1, 5^2, $(5^2)^2 = 5^4$, $(5^4)^2 = 5^8$, $(5^8)^2 = 5^{16}$, $(5^{16})^2 = 5^{32}$ all modulo 19

Write down the algorithm and estimate the complexity of this method

Other methods like Motgomery laddering are also used

Modular Exponentiation

Say you want to compute $5^{37} \mod (19)$

$37 = 2^5 + 2^2 + 2^0$

So $5^{37} = 5^{2^0} \times 5^{2^2} \times 5^{2^5} \mod (19)$

Keep squaring 5 $\mod (19)$ and multiply requisite terms: 5^1, 5^2, $(5^2)^2 = 5^4$, $(5^4)^2 = 5^8$, $(5^8)^2 = 5^{16}$, $(5^{16})^2 = 5^{32}$ all modulo 19

Write down the algorithm and estimate the complexity of this method

Other methods like Motgomery laddering are also used

RSA Security

Semantic Security

Textbook RSA is not even semantically secure

In fact, semantic security is a weak notion

It is *deterministic*

The adversary can distinguish ciphertext for 1 and 0, as he can himself encrypt

Ciphertext is malleable

Particular attacks for $e = 3$

Semantic Security

Textbook RSA is not even semantically secure

In fact, semantic security is a weak notion

It is *deterministic*

The adversary can distinguish ciphertext for 1 and 0, as he can himself encrypt

Ciphertext is malleable

Particular attacks for $e = 3$

Semantic Security

Textbook RSA is not even semantically secure

In fact, semantic security is a weak notion

It is *deterministic*

The adversary can distinguish ciphertext for 1 and 0, as he can himself encrypt

Ciphertext is malleable

Particular attacks for $e = 3$

Semantic Security

Textbook RSA is not even semantically secure

In fact, semantic security is a weak notion

It is *deterministic*

The adversary can distinguish ciphertext for 1 and 0, as he can himself encrypt

Ciphertext is malleable

Particular attacks for $e = 3$

Semantic Security

Textbook RSA is not even semantically secure

In fact, semantic security is a weak notion

It is *deterministic*

The adversary can distinguish ciphertext for 1 and 0, as he can himself encrypt

Ciphertext is malleable

Particular attacks for $e = 3$

Semantic Security

Textbook RSA is not even semantically secure

In fact, semantic security is a weak notion

It is *deterministic*

The adversary can distinguish ciphertext for 1 and 0, as he can himself encrypt

Ciphertext is malleable

Particular attacks for $e = 3$

Some Fixes

Must make the encryption function non-deterministic

Even if the message space is small non-deterministic encryption will ensure different ciphertexts for same message

Padding: Encrypt padded message and remove padding after decryption

PKCS (Public Key Cryptography Standard), OAEP (Optimal Asymmetric Encryption Padding) provide padding schemes

Making RSA secure is a huge research area

Some Fixes

Must make the encryption function non-deterministic

Even if the message space is small non-deterministic encryption will ensure different ciphertexts for same message

Padding: Encrypt padded message and remove padding after decryption

PKCS (Public Key Cryptography Standard), OAEP (Optimal Asymmetric Encryption Padding) provide padding schemes

Making RSA secure is a huge research area

Some Fixes

Must make the encryption function non-deterministic

Even if the message space is small non-deterministic encryption will ensure different ciphertexts for same message

Padding: Encrypt padded message and remove padding after decryption

PKCS (Public Key Cryptography Standard), OAEP (Optimal Asymmetric Encryption Padding) provide padding schemes

Making RSA secure is a huge research area

Some Fixes

Must make the encryption function non-deterministic

Even if the message space is small non-deterministic encryption will ensure different ciphertexts for same message

Padding: Encrypt padded message and remove padding after decryption

PKCS (Public Key Cryptography Standard), OAEP (Optimal Asymmetric Encryption Padding) provide padding schemes

Making RSA secure is a huge research area

Some Fixes

Must make the encryption function non-deterministic

Even if the message space is small non-deterministic encryption will ensure different ciphertexts for same message

Padding: Encrypt padded message and remove padding after decryption

PKCS (Public Key Cryptography Standard), OAEP (Optimal Asymmetric Encryption Padding) provide padding schemes

Making RSA secure is a huge research area

RSA vs. Factoring

RSA Problem

Given (n, e) and a $CT = m^e \mod (n)$ from the RSA encryption scheme, determine m.

Factoring

Given $n = pq$, find p.

Integer factorization is hard

If an algorithm for factoring is known, then $\phi(n)$ can be computed, the d can be found and RSA problem solved

Is factoring as hard as RSA problem? – Not known

RSA vs. Factoring

RSA Problem

Given (n, e) and a $CT = m^e \mod (n)$ from the RSA encryption scheme, determine m.

Factoring

Given $n = pq$, find p.

Integer factorization is hard

If an algorithm for factoring is known, then $\phi(n)$ can be computed, the d can be found and RSA problem solved

Is factoring as hard as RSA problem? – Not known

RSA vs. Factoring

RSA Problem

Given (n, e) and a $CT = m^e \mod (n)$ from the RSA encryption scheme, determine m.

Factoring

Given $n = pq$, find p.

Integer factorization is hard

If an algorithm for factoring is known, then $\phi(n)$ can be computed, the d can be found and RSA problem solved

Is factoring as hard as RSA problem? – Not known

RSA vs. Factoring

RSA Problem

Given (n, e) and a $CT = m^e \mod (n)$ from the RSA encryption scheme, determine m.

Factoring

Given $n = pq$, find p.

Integer factorization is hard

If an algorithm for factoring is known, then $\phi(n)$ can be computed, the d can be found and RSA problem solved

Is factoring as hard as RSA problem? – Not known

RSA vs. Factoring

RSA Problem

Given (n, e) and a $CT = m^{\mathrm{e}} \mod (n)$ from the RSA encryption scheme, determine m.

Factoring

Given $n = pq$, find p.

Integer factorization is hard

If an algorithm for factoring is known, then $\phi(n)$ can be computed, the d can be found and RSA problem solved

Is factoring as hard as RSA problem? – Not known

Algorithms for Factoring

No poly-time algorithm known for factoring

Many recent advances in so-called index calculus methods for factoring. Number Field Sieve (NFS) and its generalizations yield subexponential methods

CADO-NFS and MSIEVE projects give very efficient implementations

Prof. Veni Madhavan and his team at IISc are researching on this issue

Thorsten Kleinjung, Joppe W Bos, Arjen K Lenstra. "Mersenne Factorization Factory"

Algorithms for Factoring

No poly-time algorithm known for factoring

Many recent advances in so-called index calculus methods for factoring. Number Field Sieve (NFS) and its generalizations yield subexponential methods

CADO-NFS and MSIEVE projects give very efficient implementations

Prof. Veni Madhavan and his team at IISc are researching on this issue

Thorsten Kleinjung, Joppe W Bos, Arjen K Lenstra. "Mersenne Factorization Factory"

Algorithms for Factoring

No poly-time algorithm known for factoring

Many recent advances in so-called index calculus methods for factoring. Number Field Sieve (NFS) and its generalizations yield subexponential methods

CADO-NFS and MSIEVE projects give very efficient implementations

Prof. Veni Madhavan and his team at IISc are researching on this issue

Thorsten Kleinjung, Joppe W Bos, Arjen K Lenstra. "Mersenne Factorization Factory"

Algorithms for Factoring

No poly-time algorithm known for factoring

Many recent advances in so-called index calculus methods for factoring. Number Field Sieve (NFS) and its generalizations yield subexponential methods

CADO-NFS and MSIEVE projects give very efficient implementations

Prof. Veni Madhavan and his team at IISc are researching on this issue

Thorsten Kleinjung, Joppe W Bos, Arjen K Lenstra. "Mersenne Factorization Factory"

Algorithms for Factoring

No poly-time algorithm known for factoring

Many recent advances in so-called index calculus methods for factoring. Number Field Sieve (NFS) and its generalizations yield subexponential methods

CADO-NFS and MSIEVE projects give very efficient implementations

Prof. Veni Madhavan and his team at IISc are researching on this issue

Thorsten Kleinjung, Joppe W Bos, Arjen K Lenstra. "Mersenne Factorization Factory"

Recent Attack on RSA Public keys

https://factorable.net/index.html

Look at RSA public keys (N, e) available on the internet

If not sufficient randomness is used while generating n, results in weakness

They could succeed in finding private keys of 0.5% of TLS hosts and 0.03% SSH hosts because the modulii shared a non-trivial factor

Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices Nadia Heninger, Zakir Durumeric, Eric Wustrow, J. Alex Halderman 21st USENIX Security Symposium, August 2012

Recent Attack on RSA Public keys

https://factorable.net/index.html

Look at RSA public keys (N, e) available on the internet

If not sufficient randomness is used while generating n, results in weakness

They could succeed in finding private keys of 0.5% of TLS hosts and 0.03% SSH hosts because the modulii shared a non-trivial factor

Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices Nadia Heninger, Zakir Durumeric, Eric Wustrow, J. Alex Halderman 21st USENIX Security Symposium, August 2012

Recent Attack on RSA Public keys

https://factorable.net/index.html

Look at RSA public keys (N, e) available on the internet

If not sufficient randomness is used while generating n, results in weakness

They could succeed in finding private keys of 0.5% of TLS hosts and 0.03% SSH hosts because the modulii shared a non-trivial factor

Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices Nadia Heninger, Zakir Durumeric, Eric Wustrow, J. Alex Halderman 21st USENIX Security Symposium, August 2012

Recent Attack on RSA Public keys

https://factorable.net/index.html

Look at RSA public keys (N, e) available on the internet

If not sufficient randomness is used while generating n, results in weakness

They could succeed in finding private keys of 0.5% of TLS hosts and 0.03% SSH hosts because the modulii shared a non-trivial factor

Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices Nadia Heninger, Zakir Durumeric, Eric Wustrow, J. Alex Halderman 21st USENIX Security Symposium, August 2012

Recent Attack on RSA Public keys

https://factorable.net/index.html

Look at RSA public keys (N, e) available on the internet

If not sufficient randomness is used while generating n, results in weakness

They could succeed in finding private keys of 0.5% of TLS hosts and 0.03% SSH hosts because the modulii shared a non-trivial factor

Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices Nadia Heninger, Zakir Durumeric, Eric Wustrow, J. Alex Halderman 21st USENIX Security Symposium, August 2012

References

Rivest, R.; Shamir, A.; Adleman, L. (February 1978). "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Communications of the ACM 21 (2)

Menezes, Alfred; van Oorschot, Paul C.; Vanstone, Scott A. (October 1996). Handbook of Applied Cryptography. CRC Press

A Course in Number Theory and Cryptography, Graduate Texts in Math. No. 114, Springer-Verlag, New York, 1987. Neal Koblitz, Second edition, 1994

References

Douglas Stinson. Cryptography Theory and Practice, Third Edition, CRC Press, November 2005

Antoine Joux. Algorithmic Cryptanalysis, CRC Press, 2009

Ron was wrong, Whit is right. Arjen K. Lenstra and James P. Hughes and Maxime Augier and Joppe W. Bos and Thorsten Kleinjung and Christophe Wachter, http://eprint.iacr.org/2012/064

Thank You!