Basics of Cryptography - RSA

CSeC & ERC, IIT Bombay

Spring 2025

Introduction to RSA

RSA (Rivest–Shamir–Adleman) is one of the most widely used public-key cryptosystems, securing digital communications through encryption and authentication.

Introduction to RSA

RSA (Rivest–Shamir–Adleman) is one of the most widely used public-key cryptosystems, securing digital communications through encryption and authentication.

- Unlike symmetric encryption, **RSA uses a pair of keys**: a public key for encryption and a private key for decryption.
- Security relies on the **difficulty of factoring large numbers**.
- Commonly used in **secure web browsing (TLS/SSL), email encryption, digital signatures**, and more.

Introduction to RSA

RSA (Rivest–Shamir–Adleman) is one of the most widely used public-key cryptosystems, securing digital communications through encryption and authentication.

- Unlike symmetric encryption, **RSA uses a pair of keys**: a public key for encryption and a private key for decryption.
- Security relies on the **difficulty of factoring large numbers**.
- Commonly used in **secure web browsing (TLS/SSL), email encryption, digital signatures**, and more.

Alice and Bob: Alice wants to communicate securely with Bob using RSA.
First, she needs to generate a key pair...

RSA Key Generation

Alice performs the following steps:

- Generate two large random (and distinct) primes p and q, each roughly the same size

RSA Key Generation

Alice performs the following steps:

- Generate two large random (and distinct) primes p and q, each roughly the same size
- Compute $n = pq$ and $\phi(n) = (p-1)(q-1)$
- Select e such that $\gcd(e, \phi(n)) = 1$

RSA Key Generation

Alice performs the following steps:

- Generate two large random (and distinct) primes p and q, each roughly the same size
- Compute $n = pq$ and $\phi(n) = (p-1)(q-1)$
- Select e such that $\gcd(e, \phi(n)) = 1$
- Compute d such that $de \equiv 1 \mod \phi(n)$ (extended euclidean algorithm)

RSA Key Generation

Alice performs the following steps:

- Generate two large random (and distinct) primes p and q, each roughly the same size
- Compute $n = pq$ and $\phi(n) = (p-1)(q-1)$
- Select e such that $\gcd(e, \phi(n)) = 1$
- Compute d such that $de \equiv 1 \mod \phi(n)$ (extended euclidean algorithm)
- Public Key: (n, e)
- Private Key: (n, d)

KeyGen example

Let's use 4-bit primes to generate an 8-bit public key

- $p = 3, q = 11$
- $n = p \cdot q = 133$
- $\phi(n) = (3 - 1)(11 - 1) = 20$
- $e = 7$
- $d = 3$. Note: $(d \cdot e) \mod \phi(n) \equiv 1$
- Public Key: $(33, 7)$
- Private Key: $(33, 3)$

KeyGen Example

Public Key = $(n, e) = (33, 7)$ Private Key = $(n, d) = (33, 3)$

Sender

$(n, e) = (33, 7)$

The Public Key is shared with the Sender and the Private Key is kept secret with the Receiver .

Receiver

$(n, d) = (33, 3)$

General Parameters

The following parameters are usually used:

- Key size(n): typically 2048 to 4096 bits
- In practice, p and q are much larger (2048+ bits) for security.
- $e = 65537$

General Parameters

The following parameters are usually used:

- Here's a real-life RSA example for the value of n with 1024-bit values:
- $n =$
  18651807456834702322133879848281517050056378528411919705327185532613649
  53871073550376117143573911500261610663683571561239755329840551165356084
  58489749493255664310126152539134247966759383946446184291966746530306478
  86186720740788339414864636051754898024681124366333694868813220705860270
  41169681243098350490796598207277094606118087918527015019953261919721790
  83691728944551357688629991525563681748048416645330132854790645761635445
  46532539777263519245837670018004674364573208256582927730609407326871252
  27565065584243485828435006063081489493252744107233173273612263805528971
  52353148092651276049854366538956985300258324057

RSA Encryption

Bob performs the following steps:

- Obtain Alice's public key (n, e)
- Represent the message as an integer m in the interval $[0, n-1]$

RSA Encryption

Bob performs the following steps:

- Obtain Alice's public key (n, e)
- Represent the message as an integer m in the interval $[0, n-1]$
- Compute $c = m^e \mod n$
- Ciphertext: c

Represent the message as an integer?

- Let's say that the message is `Hello, world!`

Represent the message as an integer?

- Let's say that the message is `Hello, world!`
- ASCII value of `H` is $0x48$, `e` is $0x65$, `l` is $0x6c$ and so on...

Represent the message as an integer?

- Let's say that the message is `Hello, world!`
- ASCII value of `H` is $0x48$, `e` is $0x65$, `l` is $0x6c$ and so on...
- Write the entire string `Hello, World!` as concatenation of it's ASCII values in hex
- `Hello, world!`$= 0x48656c6c6f2c20776f726c6421$

Represent the message as an integer?

- Let's say that the message is `Hello, world!`
- ASCII value of `H` is $0x48$, `e` is $0x65$, `l` is $0x6c$ and so on...
- Write the entire string `Hello, World!` as concatenation of it's ASCII values in hex
- `Hello, world!`$= 0x48656c6c6f2c20776f726c6421$
- `Hello, world!`$= 5735816763073854953388147237921$

Encryption Example

Let's take $M = 13$ for simplicity

Cipher Text $C = M^e \bmod n$

$C = 13^7 \bmod 33$

$C = 62748517 \bmod 33$

$C = 7$

RSA Decryption

Alice performs the following steps:

- Obtain Alice's ciphertext c

RSA Decryption

Alice performs the following steps:

- Obtain Alice's ciphertext c
- Use the private key d to recover $m = c^d \mod n$
- Why does this work?

RSA Decryption

- Euler's theorem: $a^{\phi(n)} \equiv 1 \pmod{n}$ if $\gcd(a, n) = 1$

RSA Decryption

- Euler's theorem: $a^{\phi(n)} \equiv 1 \pmod{n}$ if $\gcd(a, n) = 1$
- Since e and d are chosen such that $e \cdot d \equiv 1 \pmod{\phi(n)}$, it follows that

$$
\begin{aligned}
e \cdot d &= k\phi(n) + 1 \\
\implies m^{ed} &= \left(m^{\phi(n)}\right)^k \cdot m \\
\implies m^{ed} &\equiv m \pmod{n} \\
\implies c^d &\equiv m \pmod{n}
\end{aligned}
$$

- Thus, raising the ciphertext c to the power of d gives back the original message m

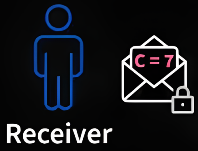
Decryption Example

Decrypted Text $M = C^d \bmod n$

$M = 7^3 \bmod 33$

$M = 343 \bmod 33$

$M = 13$



Decryption $\longrightarrow$ **$M = 13$**

The receiver uses decrypted text M = 13 to get the original message = "AC".

Is this secure?

Short answer: Yes! (For now)

- Prime factorization is computationally very expensive
- RSA-2048 would take billions of years to break with classical computers
- The largest RSA key factored to date is 829 bits (RSA-250) in 2020 (in $\sim$ 2500 core years)

Is this secure?

Short answer: Yes! (For now)

- Prime factorization is computationally very expensive
- RSA-2048 would take billions of years to break with classical computers
- The largest RSA key factored to date is 829 bits (RSA-250) in 2020 (in $\sim$ 2500 core years)
- A quantum computer with $\sim$ 20 million qubits **could** break RSA-2048 in $\sim$ 8 hours
- Post-Quantum Cryptography (PQC) is being developed as a replacement

$m^{65537} \mod n$? **Really?**

The solution: Square and Multiply Algorithm

$x \leftarrow 1$
for $i \leftarrow |e| - 1$ downto 0 **do**
    $x \leftarrow x^2 \mod n$
    if $e_i == 1$ **then**
        $x \leftarrow x \cdot m \mod n$
    end if
end for
return x

$m^{65537} \mod n$? **Really?**

The solution: Square and Multiply Algorithm

$x \leftarrow 1$
for $i \leftarrow |e| - 1$ downto 0 **do**
    $x \leftarrow x^2 \mod n$
    if $e_i == 1$ **then**
        $x \leftarrow x \cdot m \mod n$
    end if
end for
return x

Note: e_i represents the i^{th} bit of e
$m = c^d \mod n$ also uses the same algorithm, as d is generally large.

Square and Multiply

- How many multiplications will you perform to compute $5^{13} \mod 33$?

Square and Multiply

- How many multiplications will you perform to compute $5^{13} \mod 33$?
- I can perform it using 7 multiplications! (even 5 if I start with 5)

Square and Multiply

- How many multiplications will you perform to compute $5^{13} \mod 33$?
- I can perform it using 7 multiplications! (even 5 if I start with 5)
- 1. Compute $1^2 \mod 33$ (Square 1)

Square and Multiply

- How many multiplications will you perform to compute $5^{13} \mod 33$?
- I can perform it using 7 multiplications! (even 5 if I start with 5)
- 1. Compute $1^2 \mod 33$ (Square 1)
  2. Compute $5 \mod 33$ (Multiply 1^2 and 5)

Square and Multiply

- How many multiplications will you perform to compute $5^{13} \mod 33$?
- I can perform it using 7 multiplications! (even 5 if I start with 5)
- 1. Compute $1^2 \mod 33$ (Square 1)
  2. Compute $5 \mod 33$ (Multiply 1^2 and 5)
  3. Compute $5^2 \mod 33$ (Square 5)

Square and Multiply

- How many multiplications will you perform to compute $5^{13} \mod 33$?
- I can perform it using 7 multiplications! (even 5 if I start with 5)
- 1. Compute $1^2 \mod 33$ (Square 1)
  2. Compute $5 \mod 33$ (Multiply 1^2 and 5)
  3. Compute $5^2 \mod 33$ (Square 5)
  4. Compute $5^3 \mod 33$ (Multiply 5^2 and 5)

Square and Multiply

- How many multiplications will you perform to compute $5^{13} \mod 33$?
- I can perform it using 7 multiplications! (even 5 if I start with 5)
- 1. Compute $1^2 \mod 33$ (Square 1)
  2. Compute $5 \mod 33$ (Multiply 1^2 and 5)
  3. Compute $5^2 \mod 33$ (Square 5)
  4. Compute $5^3 \mod 33$ (Multiply 5^2 and 5)
  5. Compute $5^6 \mod 33$ (Square 5^3)

Square and Multiply

- How many multiplications will you perform to compute $5^{13} \mod 33$?
- I can perform it using 7 multiplications! (even 5 if I start with 5)
- 1. Compute $1^2 \mod 33$ (Square 1)
  2. Compute $5 \mod 33$ (Multiply 1^2 and 5)
  3. Compute $5^2 \mod 33$ (Square 5)
  4. Compute $5^3 \mod 33$ (Multiply 5^2 and 5)
  5. Compute $5^6 \mod 33$ (Square 5^3)
  6. Compute $5^{12} \mod 33$ (Square 5^6)

Square and Multiply

- How many multiplications will you perform to compute $5^{13} \mod 33$?
- I can perform it using 7 multiplications! (even 5 if I start with 5)
-
  1. Compute $1^2 \mod 33$ (Square 1)
  2. Compute $5 \mod 33$ (Multiply 1^2 and 5)
  3. Compute $5^2 \mod 33$ (Square 5)
  4. Compute $5^3 \mod 33$ (Multiply 5^2 and 5)
  5. Compute $5^6 \mod 33$ (Square 5^3)
  6. Compute $5^{12} \mod 33$ (Square 5^6)
  7. Compute $5^{13} \mod 33$ (Multiply $5^1$2 and 5)

Square and Multiply

- How many multiplications will you perform to compute $5^{13} \mod 33$?
- I can perform it using 7 multiplications! (even 5 if I start with 5)
-
  1. Compute $1^2 \mod 33$ (Square 1)
  2. Compute $5 \mod 33$ (Multiply 1^2 and 5)
  3. Compute $5^2 \mod 33$ (Square 5)
  4. Compute $5^3 \mod 33$ (Multiply 5^2 and 5)
  5. Compute $5^6 \mod 33$ (Square 5^3)
  6. Compute $5^{12} \mod 33$ (Square 5^6)
  7. Compute $5^{13} \mod 33$ (Multiply 5^12 and 5)
- Square and Multiply algorithm does exactly this

Another look at Square and Multiply

$x \leftarrow 1$
for $i \leftarrow |e| - 1$ downto 0 **do**
    $x \leftarrow x^2 \mod n$
    if $e_i == 1$ **then**
        $x \leftarrow x \cdot m \mod n$
    end if
end for
return x

- All this says is that if the i^{th} bit is 0, then square. If it is 1, then square and multiply

Another look at Square and Multiply

- We check the binary representation of the exponent to decide whether to square and multiply, or just square
- $13 = (1101)_2$
- The first two bits are 1 so we square and multiply twice, the third bit is 0 so we square once, and finally the last bit is 1 so we square and multiply once
- $\underbrace{\text{square and multiply}}_{1} \rightarrow \underbrace{\text{square and multiply}}_{1} \rightarrow \underbrace{\text{square}}_{0} \rightarrow \underbrace{\text{square and multiply}}_{1}$

Is Square and Multiply Secure?

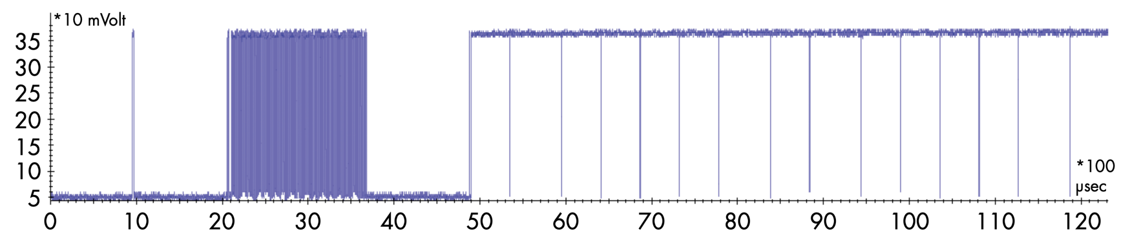


Figure: Power consumption trace of a square-and-multiply execution

Is Square and Multiply Secure?

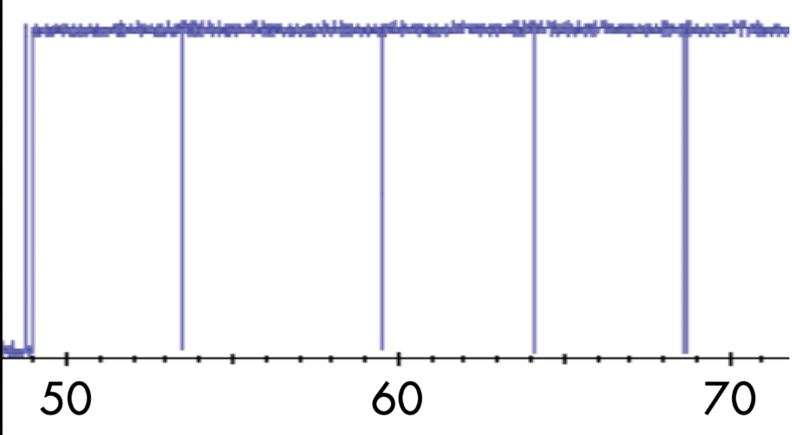


Figure: Same Power consumption trace zoomed in

Is Square and Multiply Secure?

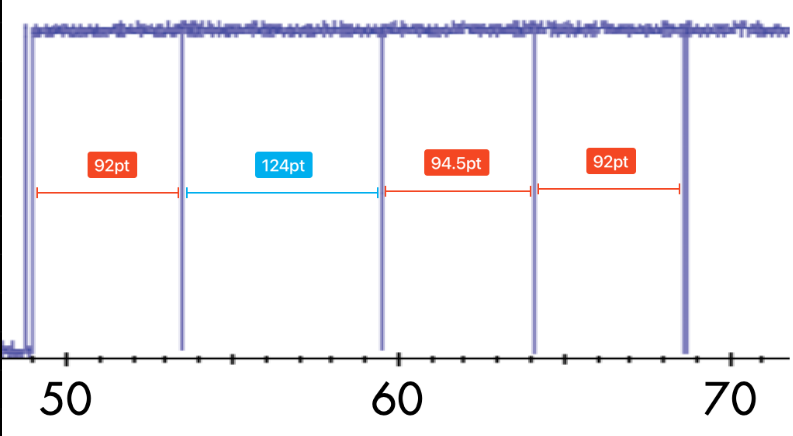


Figure: Same Power consumption trace with widths marked

Is Square and Multiply Secure?

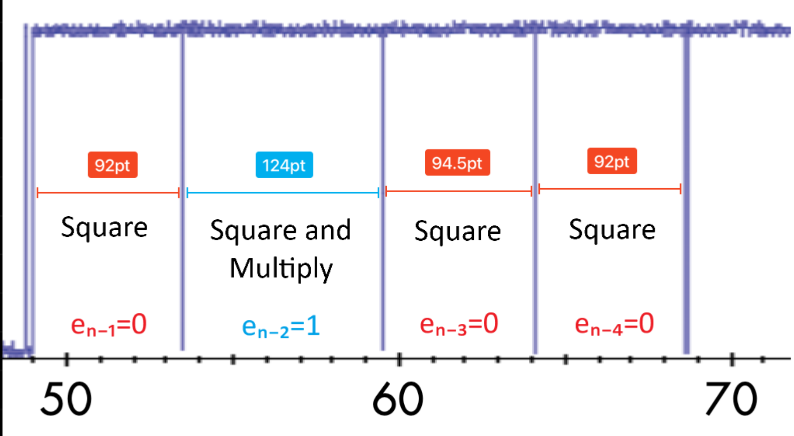


Figure: Same Power consumption trace with widths and exponent bits marked

Breaking RSA Cipher

- Since $m = c^d \mod n$ uses the Square and Multiply algorithm, we can get a Power trace similar to the example shown in the previous slide, during decryption
- When the exponent bit is 1, the time taken to compute will be higher compared to when the exponent bit is 0

Breaking RSA Cipher

- Since $m = c^d \mod n$ uses the Square and Multiply algorithm, we can get a Power trace similar to the example shown in the previous slide, during decryption
- When the exponent bit is 1, the time taken to compute will be higher compared to when the exponent bit is 0
- The effect? We retrieve the private key, d!

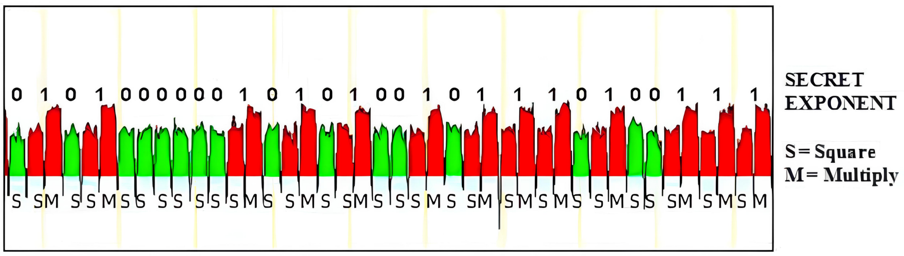


Figure: Square and Multiply Power consumption trace for an FPGA https://bit.ly/hard-hack