Introduction to Basic Cryptography

RSA

Kalyan Chakraborty
Harish-Chandra Research Institute

CIMPA School of Number Theory in Cryptography and Its Applications
School of Science, Kathmandu University,
Dhulikhel, Nepal
July 19 - July 31, 2010

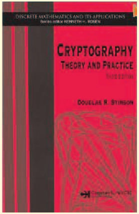
Lecture 1: July 20, 2010

http://www.hri.res.in/~jaymehta/cryptographynotesCIMPA2010.pdf

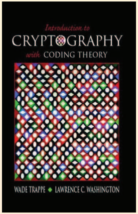
Outline

References

Cryptography - Theory and Practice
BY: **Douglas R. Stinson**



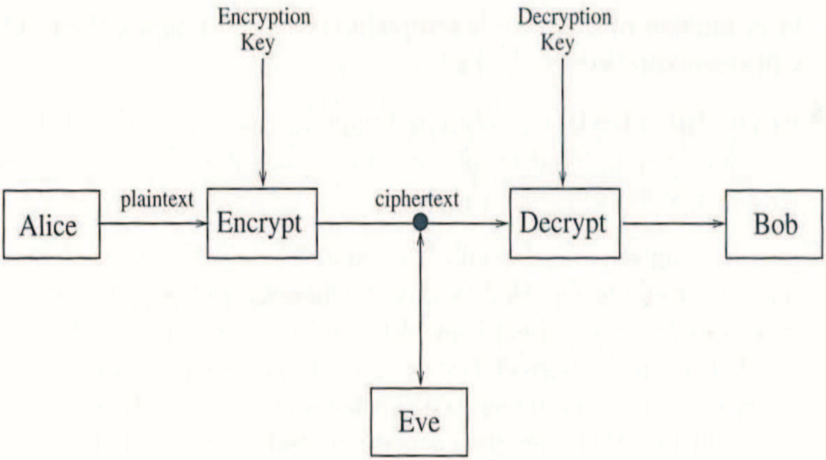
Introduction to Cryptography with Coding Theory
BY: **Wade Trappe and Lawrence Washington**



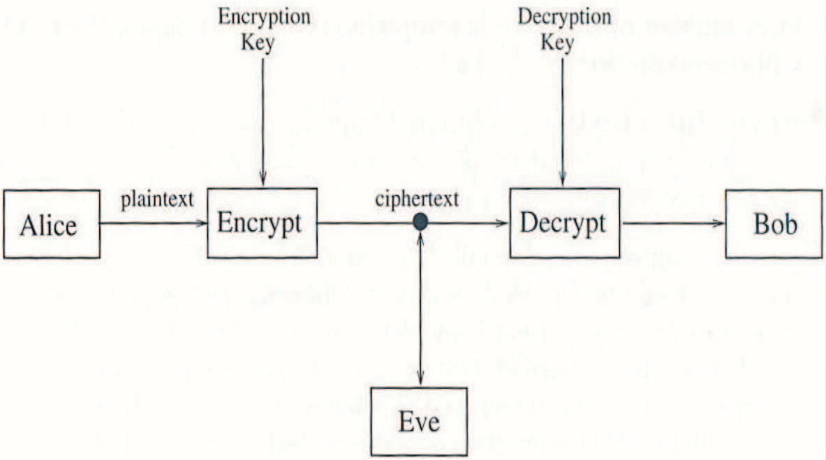
In the basic communication scenario, there are two parties, Alice and Bob, who want to communicate with each other.

A third party, Eve, is a potential eavesdropper.

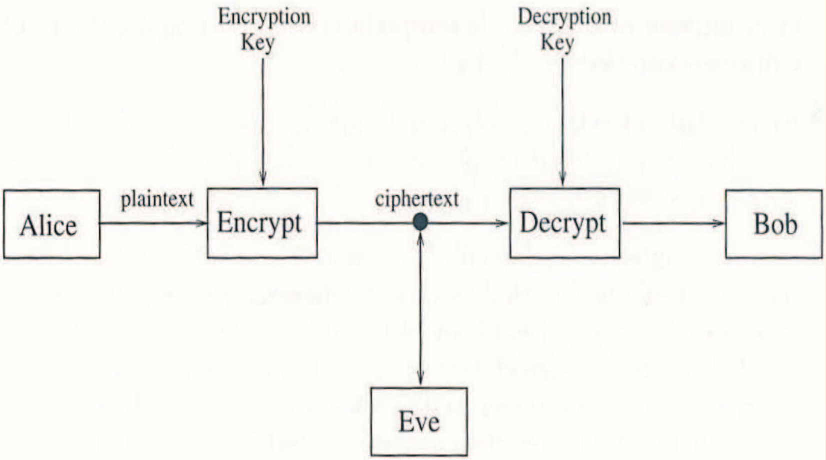
In the basic communication scenario, there are two parties, Alice and Bob, who want to communicate with each other.
A third party, Eve, is a potential eavesdropper.



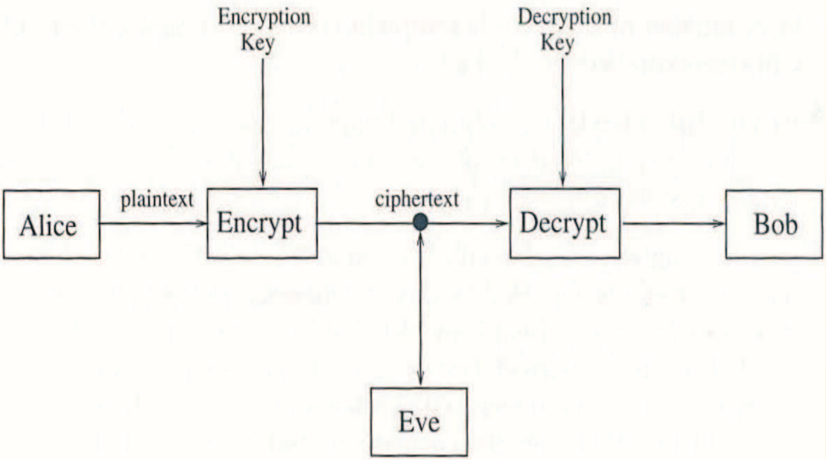
In the basic communication scenario, there are two parties, Alice and Bob, who want to communicate with each other.
A third party, Eve, is a potential eavesdropper.
Alice wants to send a message to Bob, called '*Plaintext*'.



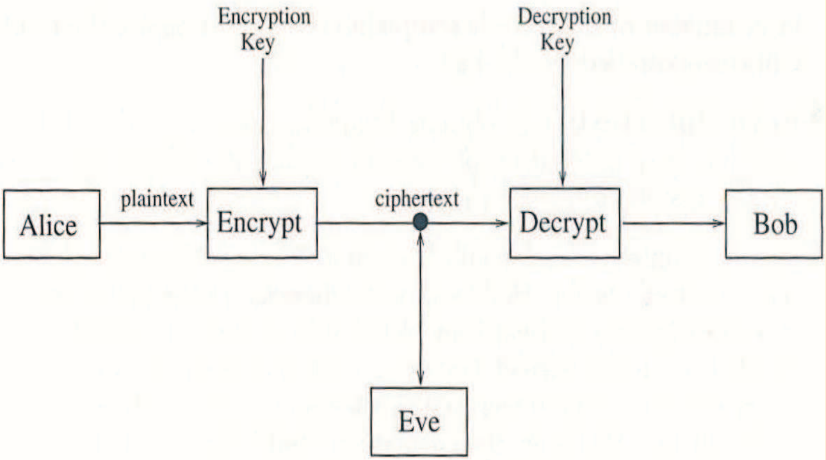
She encrypts it using a method prearranged with Bob.
Usually, the encryption method is assumed to be known to Eve. The message is kept secret because of the key.



She encrypts it using a method prearranged with Bob.
Usually, the encryption method is assumed to be known to Eve. The message is kept secret because of the key.
The encrypted message is called '*Ciphertext*'.



Bob receives the 'ciphertext' and changes it to the 'plaintext' by using a decryption key.



Eve could have one of the following goals:

Eve could have one of the following goals:

1. Read the message.

Eve could have one of the following goals:

1. Read the message.
2. Find the key and thus read all messages encrypted with that key.

Eve could have one of the following goals:

1. Read the message.
2. Find the key and thus read all messages encrypted with that key.
3. Corrupt Alice's messages into another messages in such a way that Bob will think that Alice sent the altered message.

Eve could have one of the following goals:

1. Read the message.
2. Find the key and thus read all messages encrypted with that key.
3. Corrupt Alice's messages into another messages in such a way that Bob will think that Alice sent the altered message.
4. Masquerade as Alice, and thus communicate with Bob even though Bob believes he is communicating with Alice.

Eve could have one of the following goals:

1. Read the message.
2. Find the key and thus read all messages encrypted with that key.
3. Corrupt Alice's messages into another messages in such a way that Bob will think that Alice sent the altered message.
4. Masquerade as Alice, and thus communicate with Bob even though Bob believes he is communicating with Alice.

Eve is as bad as the situation allows.

Encryption/decryption methods fall into two categories:

Symmetric key and **Public key**.

Encryption/decryption methods fall into two categories:

Symmetric key and **Public key**.

Symmetric key:

In *Symmetric key* algorithms, the encryption and decryption keys are known to both Alice and Bob.

Encryption/decryption methods fall into two categories:

Symmetric key and **Public key**.

Symmetric key:

In *Symmetric key* algorithms, the encryption and decryption keys are known to both Alice and Bob.

For example, the encryption key is shared and the decryption key is same or is easily calculated from it. All of the (pre-1970) classical cryptosystems are symmetric, as are the more recent DES (Data Encryption Standard) and AES (Advanced Encryption Standard).

Encryption/decryption methods fall into two categories:

Symmetric key and **Public key**.

Symmetric key:

In *Symmetric key* algorithms, the encryption and decryption keys are known to both Alice and Bob.

For example, the encryption key is shared and the decryption key is same or is easily calculated from it. All of the (pre-1970) classical cryptosystems are symmetric, as are the more recent DES (Data Encryption Standard) and AES (Advanced Encryption Standard).

Disadvantages :

- Needs secure channel for key exchange.
- Too many keys - Sharing a new key with every different party creates problem in managing and ensuring security.
- Origin and Authenticity of message cannot be guaranteed.

Encryption/decryption methods fall into two categories:

Symmetric key and **Public key**.

Public key:

Public key algorithms were introduced in the 1970s, which revolutionized cryptography.

Encryption/decryption methods fall into two categories:

Symmetric key and **Public key**.

Public key:

Public key algorithms were introduced in the 1970s, which revolutionized cryptography.

Suppose Alice and Bob haven't agreed on sharing a key, or using a trusted courier to carry the key. Certainly Alice cannot send a message over open channels to tell Bob the key and Ciphertext encrypted with this key.

Encryption/decryption methods fall into two categories:

Symmetric key and **Public key**.

Public key:

Public key algorithms were introduced in the 1970s, which revolutionized cryptography.

Suppose Alice and Bob haven't agreed on sharing a key, or using a trusted courier to carry the key. Certainly Alice cannot send a message over open channels to tell Bob the key and Ciphertext encrypted with this key.

This problem has a solution, called 'PKC', where encryption key is public, but it is computationally infeasible to find the decryption key without information which is known to Bob only.

The most popular implementation is RSA, based on difficulty of factoring large integers. Other versions are due to ElGamal (based on DLP), etc.

One important observation about PKC is that it never provides unconditional security.

One important observation about PKC is that it never provides unconditional security.

As an opponent, on observing a 'ciphertext' y, can encrypt each possible 'plaintext' inturn using the public encryption rule e_k until she finds the unique x such that

$$y = e_k(x).$$

One important observation about PKC is that it never provides unconditional security.

As an opponent, on observing a ‘ciphertext’ y, can encrypt each possible ‘plaintext’ inturn using the public encryption rule e_k until she finds the unique x such that

$$y = e_k(x).$$

This x is the decryption of y. So, its important to study security of PKC.
Bob’s public encryption formula e_k should be easy to compute. The decryption should be hard. Such a formula is often called a “one way formula”.

One important observation about PKC is that it never provides unconditional security.

As an opponent, on observing a 'ciphertext' y, can encrypt each possible 'plaintext' inturn using the public encryption rule e_k until she finds the unique x such that

$$y = e_k(x).$$

This x is the decryption of y. So, its important to study security of PKC.
Bob's public encryption formula e_k should be easy to compute. The decryption should be hard. Such a formula is often called a "one way formula".

In the context of encryption we want e_k to be injective one way function so that decryption can be performed. Unfortunately there aren't many functions which can be considered 'one way'.

Example: $n = pq$; b a positive integer. Then

$$f : \mathbb{Z}_n \to \mathbb{Z}_n;$$
$$f(x) \equiv x^b \pmod{n}.$$

(if $\gcd(b, \phi(n)) = 1$, this is RSA encryption function).

While construction PKC, we don't want e_k to be one way from Bob's point of view, because he should be able to decrypt messages efficiently that he receives.

Thus, it is necessary that Bob possesses a "trapdoor" which consists of secret information that permits easy inverse of e_k.

RSA

Without prior contact, Alice wants to send message to Bob and don't want to send a courier with a key, as all information will be obtained by evil observer Eve. However, its possible to send a message without being read by Eve.

RSA

Without prior contact, Alice wants to send message to Bob and don't want to send a courier with a key, as all information will be obtained by evil observer Eve. However, its possible to send a message without being read by Eve.

This scheme is called 'Public key cryptosystem', first suggested by Diffe and Hellman (without any practical implementation).

RSA

Without prior contact, Alice wants to send message to Bob and don't want to send a courier with a key, as all information will be obtained by evil observer Eve. However, its possible to send a message without being read by Eve.
This scheme is called 'Public key cryptosystem', first suggested by Diffe and Hellman (without any practical implementation).
Next successful method, based on the idea that factorization of integers into their prime factors is hard, was proposed by Rivest, Shamir and Adleman in 1977, known as RSA algorithm.

RSA

Without prior contact, Alice wants to send message to Bob and don't want to send a courier with a key, as all information will be obtained by evil observer Eve. However, its possible to send a message without being read by Eve.

This scheme is called 'Public key cryptosystem', first suggested by Diffe and Hellman (without any practical implementation).

Next successful method, based on the idea that factorization of integers into their prime factors is hard, was proposed by Rivest, Shamir and Adleman in 1977, known as RSA algorithm.

In 1997, documents released by a British cryptographic agency CESG, showed that

RSA

Without prior contact, Alice wants to send message to Bob and don't want to send a courier with a key, as all information will be obtained by evil observer Eve. However, its possible to send a message without being read by Eve.

This scheme is called 'Public key cryptosystem', first suggested by Diffe and Hellman (without any practical implementation).

Next successful method, based on the idea that factorization of integers into their prime factors is hard, was proposed by Rivest, Shamir and Adleman in 1977, known as RSA algorithm.

In 1997, documents released by a British cryptographic agency CESG, showed that

- In 1970, James Ellis discovered 'PKC'.

RSA

Without prior contact, Alice wants to send message to Bob and don't want to send a courier with a key, as all information will be obtained by evil observer Eve. However, its possible to send a message without being read by Eve.

This scheme is called 'Public key cryptosystem', first suggested by Diffe and Hellman (without any practical implementation).

Next successful method, based on the idea that factorization of integers into their prime factors is hard, was proposed by Rivest, Shamir and Adleman in 1977, known as RSA algorithm.

In 1997, documents released by a British cryptographic agency CESG, showed that

- In 1970, James Ellis discovered 'PKC'.
- In 1973, Clifford Cocks had written an internal document describing a version of RSA algorithm in which the encryption exponent e was same as the modulus n.

RSA Algorithm

Bob chooses two large primes p and q and forms

$$n = pq$$

RSA Algorithm

Bob chooses two large primes p and q and forms

$$n = pq$$

He also chooses an encryption exponent e such that

$$\big(e, (p-1)(q-1)\big) = 1$$

He makes the pair (n, e) public, keeping p and q secret.

RSA Algorithm

Bob chooses two large primes p and q and forms

$$n = pq$$

He also chooses an encryption exponent e such that

$$\big(e, (p-1)(q-1)\big) = 1$$

He makes the pair (n, e) public, keeping p and q secret.
Alice writes her message as a number m. For simplicity, assume $m < n$ (if $m > n$, she breaks the message into blocks each with length $< n$).

RSA Algorithm

Bob chooses two large primes p and q and forms

$$n = pq$$

He also chooses an encryption exponent e such that

$$\big(e, (p-1)(q-1)\big) = 1$$

He makes the pair (n, e) public, keeping p and q secret.
Alice writes her message as a number m. For simplicity, assume $m < n$ (if $m > n$, she breaks the message into blocks each with length $< n$).
Alice computes

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

and sends c to Bob.

RSA Algorithm

Bob chooses two large primes p and q and forms

$$n = pq$$

He also chooses an encryption exponent e such that

$$\big(e, (p-1)(q-1)\big) = 1$$

He makes the pair (n, e) public, keeping p and q secret.
Alice writes her message as a number m. For simplicity, assume $m < n$ (if $m > n$, she breaks the message into blocks each with length $< n$).
Alice computes

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

and sends c to Bob.
As Bob knows p and q, he can find the decryption exponent d with

$$de \equiv 1\big(\bmod (p-1)(q-1)\big)$$

and calculates

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

RSA Algorithm

The RSA Algorithm

1. Bob chooses secret primes p and q and computes $n = pq$.
2. Bob chooses e with $\big(e, (p-1)(q-1)\big) = 1$.
3. Bob computes d with $de \equiv 1\big(\mathrm{mod}(p-1)(q-1)\big)$.
4. Bob makes n and e public and keeps p, q, d secret.
5. Alice encrypts m as $c \equiv m^e(\mathrm{mod} n)$ and sends c to Bob.
6. Bob decrypts by computing $m \equiv c^d(\mathrm{mod} n)$.

RSA Algorithm

The RSA Algorithm

1. Bob chooses secret primes p and q and computes $n = pq$.
2. Bob chooses e with $\big(e, (p-1)(q-1)\big) = 1$.
3. Bob computes d with $de \equiv 1\big(\mathrm{mod}(p-1)(q-1)\big)$.
4. Bob makes n and e public and keeps p, q, d secret.
5. Alice encrypts m as $c \equiv m^e(\mathrm{mod} n)$ and sends c to Bob.
6. Bob decrypts by computing $m \equiv c^d(\mathrm{mod} n)$.

- The security of RSA is based on the belief that the encryption formula $\boxed{e_k(m) = m^e \bmod n}$ is a one-way function. The trapdoor that allows Bob to decrypt a Ciphertext is the knowledge of factorization $n = pq$.

Example Suppose Bob chooses $p = 101, q = 113$. Then

$$n = 11413$$

$$\phi(n) = 100 \times 112 = 11200$$

Example Suppose Bob chooses $p = 101, q = 113$. Then

$$n = 11413$$
$$\phi(n) = 100 \times 112 = 11200$$

Since $11200 = 2^6 5^2 7$, an integer e can be used as an encryption exponent if e is not divisible by $2, 5$ or 7.
(Bob will verify $(e, \phi(n)) = 1$ by using extended Euclidean algorithm and will compute d $(= e^{-1} \bmod \phi(n))$ at the same time.

Example Suppose Bob chooses $p = 101, q = 113$. Then

$$n = 11413$$
$$\phi(n) = 100 \times 112 = 11200$$

Since $11200 = 2^6 5^2 7$, an integer e can be used as an encryption exponent if e is not divisible by $2, 5$ or 7.
(Bob will verify $(e, \phi(n)) = 1$ by using extended Euclidean algorithm and will compute d $(= e^{-1} \bmod \phi(n))$ at the same time.
Suppose Bob chooses $e = 3533$. Then

$$e^{-1} \bmod 11200 = 6597 = d \quad \text{(which is secret)}$$

Bob publishes $n = 11413$ and $e = 3533$ in a directory.

Example Suppose Bob chooses $p = 101, q = 113$. Then

$$n = 11413$$
$$\phi(n) = 100 \times 112 = 11200$$

Since $11200 = 2^6 5^2 7$, an integer e can be used as an encryption exponent if e is not divisible by $2, 5$ or 7.
(Bob will verify $(e, \phi(n)) = 1$ by using extended Euclidean algorithm and will compute d $(= e^{-1} \bmod \phi(n))$ at the same time.
Suppose Bob chooses $e = 3533$. Then

$$e^{-1} \bmod 11200 = 6597 = d \quad \text{(which is secret)}$$

Bob publishes $n = 11413$ and $e = 3533$ in a directory. Now, suppose Alice wants to send the plaintext 9726 to Bob. She will compute

$$c = 9726^{3533} \bmod 11413 = 5761$$

and sends ciphertext $c = 5761$ to Bob over the channel.

Example Suppose Bob chooses $p = 101, q = 113$. Then

$$n = 11413$$
$$\phi(n) = 100 \times 112 = 11200$$

Since $11200 = 2^6 5^2 7$, an integer e can be used as an encryption exponent if e is not divisible by $2, 5$ or 7.
(Bob will verify $(e, \phi(n)) = 1$ by using extended Euclidean algorithm and will compute d $(= e^{-1} \bmod \phi(n))$ at the same time.
Suppose Bob chooses $e = 3533$. Then

$$e^{-1} \bmod 11200 = 6597 = d \quad \text{(which is secret)}$$

Bob publishes $n = 11413$ and $e = 3533$ in a directory. Now, suppose Alice wants to send the plaintext 9726 to Bob. She will compute

$$c = 9726^{3533} \bmod 11413 = 5761$$

and sends ciphertext $c = 5761$ to Bob over the channel.
When Bob receives 5761 he uses d to compute

$$5761^{6597} \bmod 11413 = 9726$$

The encryption and decryption are inverse operations:

The encryption and decryption are inverse operations:

As, $de = 1 (\text{mod } \phi(n))$, one has

$$de = t\phi(n) + 1 \text{ for some integer } t \geq 1.$$

The encryption and decryption are inverse operations:
As, $de = 1(\text{mod } \phi(n))$, one has

$$de = t\phi(n) + 1 \text{ for some integer } t \geq 1.$$

Suppose that $x \in \mathbb{Z}_n{}^*$, then

$$\begin{aligned}
(x^d)^e &\equiv x^{t\phi(n)+1}(\text{mod } n) \\
&\equiv (x^{\phi(n)})^t x(\text{mod } n) \\
&\equiv 1^t x(\text{mod } n) && \text{(Lagrange's Theorem)} \\
&\equiv x(\text{mod } n)\blacksquare
\end{aligned}$$

The encryption and decryption are inverse operations:

As, $de = 1(\text{mod } \phi(n))$, one has

$$de = t\phi(n) + 1 \text{ for some integer } t \geq 1.$$

Suppose that $x \in \mathbb{Z}_n{}^*$, then

$$\begin{aligned}
(x^d)^e &\equiv x^{t\phi(n)+1}(\text{mod } n) \\
&\equiv (x^{\phi(n)})^t x(\text{mod } n) \\
&\equiv 1^t x(\text{mod } n) && \text{(Lagrange's Theorem)} \\
&\equiv x(\text{mod } n) \blacksquare
\end{aligned}$$

Exercise : Show that $(x^d)^e \equiv x(\text{mod } n)$ if $x \in \mathbb{Z}_n$

(**Hint:** Use the fact that $x_1 \equiv x_2(\mod pq)$ if and only if $x_1 \equiv x_2(\mod p)$ and $x_1 \equiv x_2(\mod q)$).

Implementing RSA

- There are many aspects of RSA to discuss including the setting of cryptosystem, the efficiency of encryption and decryption and the security issues.

Implementing RSA

- There are many aspects of RSA to discuss including the setting of cryptosystem, the efficiency of encryption and decryption and the security issues.
- Bob uses the RSA Parameter Generation Algorithm.

Implementing RSA

- There are many aspects of RSA to discuss including the setting of cryptosystem, the efficiency of encryption and decryption and the security issues.
- Bob uses the RSA Parameter Generation Algorithm.

RSA Parameter Generation

1. Generate two large primes p and q such that $p \neq q$

Implementing RSA

- There are many aspects of RSA to discuss including the setting of cryptosystem, the efficiency of encryption and decryption and the security issues.
- Bob uses the RSA Parameter Generation Algorithm.

RSA Parameter Generation

1. Generate two large primes p and q such that $p \neq q$
2. $n \leftarrow pq$ and $\phi(n) \leftarrow (p-1)(q-1)$

Implementing RSA

- There are many aspects of RSA to discuss including the setting of cryptosystem, the efficiency of encryption and decryption and the security issues.
- Bob uses the RSA Parameter Generation Algorithm.

RSA Parameter Generation

1. Generate two large primes p and q such that $p \neq q$
2. $n \leftarrow pq$ and $\phi(n) \leftarrow (p-1)(q-1)$
3. Choose a random e $(1 < e < \phi(n))$ such that $\gcd(e, \phi(n)) = 1$

Implementing RSA

- There are many aspects of RSA to discuss including the setting of cryptosystem, the efficiency of encryption and decryption and the security issues.
- Bob uses the RSA Parameter Generation Algorithm.

RSA Parameter Generation

1. Generate two large primes p and q such that $p \neq q$
2. $n \leftarrow pq$ and $\phi(n) \leftarrow (p-1)(q-1)$
3. Choose a random e $(1 < e < \phi(n))$ such that $\gcd(e, \phi(n)) = 1$
4. $d \leftarrow e^{-1} \mod \phi(n)$

Implementing RSA

- There are many aspects of RSA to discuss including the setting of cryptosystem, the efficiency of encryption and decryption and the security issues.
- Bob uses the RSA Parameter Generation Algorithm.

RSA Parameter Generation

1. Generate two large primes p and q such that $p \neq q$
2. $n \leftarrow pq$ and $\phi(n) \leftarrow (p-1)(q-1)$
3. Choose a random e $(1 < e < \phi(n))$ such that $\gcd(e, \phi(n)) = 1$
4. $d \leftarrow e^{-1} \mod \phi(n)$
5. Public key is (n, e) and private key is (p, q, d)

Implementing RSA

- There are many aspects of RSA to discuss including the setting of cryptosystem, the efficiency of encryption and decryption and the security issues.
- Bob uses the RSA Parameter Generation Algorithm.

RSA Parameter Generation

1. Generate two large primes p and q such that $p \neq q$
2. $n \leftarrow pq$ and $\phi(n) \leftarrow (p-1)(q-1)$
3. Choose a random e $(1 < e < \phi(n))$ such that $\gcd(e, \phi(n)) = 1$
4. $d \leftarrow e^{-1} \mod \phi(n)$
5. Public key is (n, e) and private key is (p, q, d)

- Step 1 will be discussed next.
- Step 2, 3, 4 can be done in time $O((\log n)^2)$.

- Both encryption and decryption in RSA are modular exponentiation.

- Both encryption and decryption in RSA are modular exponentiation.
- Direct modular multiplications are very inefficient if the exponent is very large.

- Both encryption and decryption in RSA are modular exponentiation.
- Direct modular multiplications are very inefficient if the exponent is very large.
- One uses the well- known "Square-And-Multiply Algorithm" to reduce the number of modular multiplications.

- Both encryption and decryption in RSA are modular exponentiation.
- Direct modular multiplications are very inefficient if the exponent is very large.
- One uses the well- known "Square-And-Multiply Algorithm" to reduce the number of modular multiplications.
- By using this method one can verify that RSA encryption and decryption in the time $O((\log n)^3)$, which is a polynomial function of the number of bits in one plaintext.

- Both encryption and decryption in RSA are modular exponentiation.
- Direct modular multiplications are very inefficient if the exponent is very large.
- One uses the well- known "Square-And-Multiply Algorithm" to reduce the number of modular multiplications.
- By using this method one can verify that RSA encryption and decryption in the time $O((\log n)^3)$, which is a polynomial function of the number of bits in one plaintext.

  Exercise : The ciphertext 5859 was obtained from the RSA algorithm using $n = 11413$ and $e = 7467$. Using the factorization $11413 = 101 \times 113$, find the plaintext.

Connection with Factoring

One obvious attack on the RSA Cryptosystem is to attempt to factor n. If this can be done, it is simple to compute $\phi(n)$ and then compute the decryption exponent d from e as Bob did.

Connection with Factoring

One obvious attack on the RSA Cryptosystem is to attempt to factor n. If this can be done, it is simple to compute $\phi(n)$ and then compute the decryption exponent d from e as Bob did.

NOTE : *It has been conjectured that breaking the **RSA** is polynomially equivalent to factoring n, but this remains unproved.*

Connection with Factoring

One obvious attack on the RSA Cryptosystem is to attempt to factor n. If this can be done, it is simple to compute $\phi(n)$ and then compute the decryption exponent d from e as Bob did.

NOTE : *It has been conjectured that breaking the **RSA** is polynomially equivalent to factoring n, but this remains unproved.*

To secure RSA it is necessary that $n = pq$ must be large enough such that factoring it will be computationally infeasible.

Connection with Factoring

One obvious attack on the RSA Cryptosystem is to attempt to factor n. If this can be done, it is simple to compute $\phi(n)$ and then compute the decryption exponent d from e as Bob did.

NOTE : *It has been conjectured that breaking the* ***RSA*** *is polynomially equivalent to factoring* n*, but this remains unproved.*

To secure RSA it is necessary that $n = pq$ must be large enough such that factoring it will be computationally infeasible.

Current factoring algorithm are able to factor numbers having upto 512 bits in their binary representation. It is generally recommended, one should choose each of p and q to be 512-bit prime, then n would be a 1024-bit modulus.
Factoring a number of this size is well beyond the capacity of the best current algorithm.

Primality Testing

In setting up the RSA Cryptosystem, it is necessary to generate large 'random primes' and then test them for primality.

Primality Testing

In setting up the RSA Cryptosystem, it is necessary to generate large ‘random primes’ and then test them for primality.

In 2002, it was shown by Agrawal, Kayal and Saxena that there is a polynomial-time deterministic algorithm for primality testing. In practice, primality testing is still done mainly by using a randomized polynomial-time Monte Carlo Algorithm such as Solovay-Strassen Algorithm or Miller-Rabin Algorithm.

Primality Testing

In setting up the RSA Cryptosystem, it is necessary to generate large 'random primes' and then test them for primality.

In 2002, it was shown by Agrawal, Kayal and Saxena that there is a polynomial-time deterministic algorithm for primality testing. In practice, primality testing is still done mainly by using a randomized polynomial-time Monte Carlo Algorithm such as Solovay-Strassen Algorithm or Miller-Rabin Algorithm.

These algorithms are fast, but there is a possibility that the algorithm may claim that n is prime, when it is not. However, by running algorithm enough times the error probability can be reduced.

Primality Testing

In setting up the RSA Cryptosystem, it is necessary to generate large 'random primes' and then test them for primality.

In 2002, it was shown by Agrawal, Kayal and Saxena that there is a polynomial-time deterministic algorithm for primality testing. In practice, primality testing is still done mainly by using a randomized polynomial-time Monte Carlo Algorithm such as Solovay-Strassen Algorithm or Miller-Rabin Algorithm.

These algorithms are fast, but there is a possibility that the algorithm may claim that n is prime, when it is not. However, by running algorithm enough times the error probability can be reduced.

The other pertinent question is how many random integers (of a specified size) will need to be tested until we find one that is prime.

$$\text{Set } \pi(N) \to \# \text{ of primes} \leq N.$$
$$\text{PNT} \to \pi(N) \sim \frac{N}{\log N}.$$

$$\text{Set } \pi(N) \to \# \text{ of primes} \leq N.$$
$$\text{PNT} \to \pi(N) \sim \frac{N}{\log\ N}.$$

Hence, if an integer p is chosen at random between 1 and N, then probability that it is prime is about $\frac{1}{\log\ N}$.

$$\text{Set } \pi(N) \to \# \text{ of primes} \leq N.$$
$$\text{PNT} \to \pi(N) \sim \frac{N}{\log N}.$$

Hence, if an integer p is chosen at random between 1 and N, then probability that it is prime is about $\frac{1}{\log N}$.

For a 1024 bit modulus $n = pq$; p and q will be 512 bit primes. A random 512 bit integer will be prime with probability approx.

$$\frac{1}{\ln 2^{512}} \approx \frac{1}{355}$$

i.e. on average, given 355 random 512 bit integers p, one of them will be prime (restricting to odd integers, probability doubles to about $\frac{2}{355}$).

$$\text{Set } \pi(N) \rightarrow \# \text{ of primes} \leq N.$$
$$\text{PNT} \rightarrow \pi(N) \sim \frac{N}{\log\ N}.$$

Hence, if an integer p is chosen at random between 1 and N, then probability that it is prime is about $\frac{1}{\log\ N}$.

For a 1024 bit modulus $n = pq$; p and q will be 512 bit primes. A random 512 bit integer will be prime with probability approx.

$$\frac{1}{\ln\ 2^{512}} \approx \frac{1}{355}$$

i.e. on average, given 355 random 512 bit integers p, one of them will be prime (restricting to odd integers, probability doubles to about $\frac{2}{355}$).

So, we can generate sufficiently large random numbers that are "probably prime" and hence parameter generation for the RSA Cryptosystem is indeed practical.

Definition:

Let p be an odd prime and $a \in \mathbb{Z}$;

- a is said to be quadratic residue modulo p if $a \not\equiv 0 \pmod{p}$ and the congruence $y^2 \equiv a \pmod{p}$ has a solution $y \in \mathbb{Z}_p$.
- a is quadratic non-residue mod p; otherwise.

Definition:

Let p be an odd prime and $a \in \mathbb{Z}$;

- a is said to be quadratic residue modulo p if $a \not\equiv O \pmod{p}$ and the congruence $y^2 \equiv a \pmod{p}$ has a solution $y \in \mathbb{Z}_p$.
- a is quadratic non-residue mod p; otherwise.

Example:
In $\mathbb{Z}_{11}$,
$1, 3, 4, 5, 9$ are quadratic residue modulo 11.
$2, 6, 7, 8, 10$ are quadratic non-residue modulo 11.

Definition:

Let p be an odd prime and $a \in \mathbb{Z}$;

- a is said to be quadratic residue modulo p if $a \not\equiv 0 \pmod{p}$ and the congruence $y^2 \equiv a \pmod{p}$ has a solution $y \in \mathbb{Z}_p$.
- a is quadratic non-residue mod p; otherwise.

Example:
In $\mathbb{Z}_{11}$,
$1, 3, 4, 5, 9$ are quadratic residue modulo 11.
$2, 6, 7, 8, 10$ are quadratic non-residue modulo 11.

Euler's Criterion :

Let p be an odd prime. Then a is a quadratic residue mod p if and only if

$$a^{\frac{(p-1)}{2}} \equiv 1 \pmod{p}.$$

Legendre and Jacobi Symbols :

Legendre Symbol $\left(\frac{a}{p}\right)$:

Suppose p is an odd prime. For any integer a, define symbol $\left(\frac{a}{p}\right)$ as:

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } a \equiv 0 \pmod{p} \\ 1 & \text{if } a \text{ is a quadratic residue modulo } p \\ -1 & \text{if } a \text{ is a quadratic non-residue modulo } p \end{cases}$$

Therefore, $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$.

Legendre and Jacobi Symbols :

Legendre Symbol $\left(\frac{a}{p}\right)$:

Suppose p is an odd prime. For any integer a, define symbol $\left(\frac{a}{p}\right)$ as:

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } a \equiv 0 \pmod{p} \\ 1 & \text{if } a \text{ is a quadratic residue modulo } p \\ -1 & \text{if } a \text{ is a quadratic non-residue modulo } p \end{cases}$$

Therefore, $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$.

Jacobi Symbol $\left(\frac{a}{n}\right)$:

Suppose n is an odd positive integer, and $n = \prod_{i=1}^{k} {p_i}^{e_i}$.

Let a be an integer, then

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{e_i}.$$

The Solovay-Strassen Algorithm (n) :

Choose a random integer a such that $1 \leq a \leq n-1$

$x \leftarrow \left(\frac{a}{n}\right)$

if $x = 0$ **then**

  return ("n is composite")

$y \leftarrow a^{\frac{(n-1)}{2}} \pmod{n}$

if $x = y \pmod{n}$ **then**

  return ("n is prime")

else

  return ("n is composite")

It is a yes - biased Monte Carlo algorithm with error probability at the most $\frac{1}{2}$.

REMARKS on Solovay-Strassen Algorithm:

REMARKS on Solovay-Strassen Algorithm:

Suppose $n > 1$ is odd. If n is prime then $\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n}$ for any a. But, if n is composite, it may or may not be the case. If this is the case then a is called *Euler pseudo-prime* to the base n.

REMARKS on Solovay-Strassen Algorithm:

Suppose $n > 1$ is odd. If n is prime then $\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n}$ for any a. But, if n is composite, it may or may not be the case. If this is the case then a is called *Euler pseudo-prime* to the base n.

Example: 10 is an Euler pseudo-prime to the base 91, since

$$\left(\frac{10}{91}\right) = -1 = 10^{45} \pmod{91}.$$

REMARKS on Solovay-Strassen Algorithm:

Suppose $n > 1$ is odd. If n is prime then $\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n}$ for any a. But, if n is composite, it may or may not be the case. If this is the case then a is called *Euler pseudo-prime* to the base n.

Example: 10 is an Euler pseudo-prime to the base 91, since

$$\left(\frac{10}{91}\right) = -1 = 10^{45} \pmod{91}.$$

However, it can be shown that, for any odd composite n, at most half of the integers a such that $1 \leq a \leq n-1$ are *Euler pseudo - primes* to the base n.

REMARKS on Solovay-Strassen Algorithm:

Suppose $n > 1$ is odd. If n is prime then $\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \pmod{n}$ for any a. But, if n is composite, it may or may not be the case. If this is the case then a is called *Euler pseudo-prime* to the base n.

Example: 10 is an Euler pseudo-prime to the base 91, since

$$\left(\frac{10}{91}\right) = -1 = 10^{45} \pmod{91}.$$

However, it can be shown that, for any odd composite n, at most half of the integers a such that $1 \leq a \leq n-1$ are *Euler pseudo - primes* to the base n.
Hence, error probability of Solovay-Strassen Algorithm is atmost $\frac{1}{2}$.
(The next exercise will prove this error probability).

Exercise

Define $G(n) = \{a \ : \ a \in \mathbb{Z}_n^*, \ \left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \mod n\}$.

- Show that $G(n)$ is a subgroup of $\mathbb{Z}_n^*$. Thus, if $G(n) \neq \mathbb{Z}_n^*$,
$$|G(n)| \leq \frac{|\mathbb{Z}_n^*|}{2} \leq \frac{n-1}{2}.$$
- If $n = p^k q$ where p and q are odd, p is prime, $k \geq 2$ and $\gcd(p,q) = 1$. Let $a = 1 + p^{(k-1)q}$. Show that $\left(\frac{a}{n}\right) \not\equiv a^{(n-1)/2} \mod n\}$.
- If $n = p_1 p_2, \dots p_s$ where p_i's are distinct odd primes. Suppose $a \equiv u \mod p_1$ and $a \equiv 1 \mod p_2 \dots p_s$ where u is a quadratic non-residue $\mod p_1$.
Then show that $\left(\frac{a}{n}\right) \equiv -1 \mod n$ but $a^{(n-1)/2} \equiv 1 \mod p_2 \dots p_s$.
So, $a^{(n-1)/2} \not\equiv 1 \mod n$
- If n is odd and composite $|G(n)| \leq \frac{n-1}{2}$.
- Conclude : The error probability of the Solovay-Strassen Primality test is atmost $\frac{1}{2}$.

Check whether it is a polynomial-time algorithm:

Check whether it is a polynomial-time algorithm: We can evaluate $a^{\frac{(n-1)}{2}} \bmod n$ in $O\big((\log n)^3\big)$ time. One can evaluate $\left(\frac{a}{n}\right)$ without factoring n.

Check whether it is a polynomial-time algorithm: We can evaluate $a^{\frac{(n-1)}{2}}$ mod n in $O\big((\log n)^3\big)$ time. One can evaluate $\left(\frac{a}{n}\right)$ without factoring n.

1. If n is positive and $m_1 \equiv m_2 (\text{mod } n)$ then
$$\left(\frac{m_1}{n}\right) = \left(\frac{m_2}{n}\right)$$

Check whether it is a polynomial-time algorithm: We can evaluate $a^{\frac{(n-1)}{2}}$ mod n in $O\big((\log n)^3\big)$ time. One can evaluate $\left(\frac{a}{n}\right)$ without factoring n.

1. If n is positive and $m_1 \equiv m_2 (\text{mod } n)$ then
$$\left(\frac{m_1}{n}\right) = \left(\frac{m_2}{n}\right)$$

2. If n is positive,
$$\left(\frac{2}{n}\right) = \begin{cases} 1 & \text{if } n \equiv \pm 1 (\text{mod } 8) \\ -1 & \text{if } n \equiv \pm 3 (\text{mod } 8) \end{cases}$$

Check whether it is a polynomial-time algorithm: We can evaluate $a^{\frac{(n-1)}{2}}$ mod n in $O\big((\log n)^3\big)$ time. One can evaluate $\left(\frac{a}{n}\right)$ without factoring n.

1. If n is positive and $m_1 \equiv m_2 (\text{mod } n)$ then
$$\left(\frac{m_1}{n}\right) = \left(\frac{m_2}{n}\right)$$
2. If n is positive,
$$\left(\frac{2}{n}\right) = \begin{cases} 1 & \text{if } n \equiv \pm 1 (\text{mod } 8) \\ -1 & \text{if } n \equiv \pm 3 (\text{mod } 8) \end{cases}$$
3. $\left(\frac{m_1 m_2}{n}\right) = \left(\frac{m_1}{n}\right)\left(\frac{m_2}{n}\right)$. In particular, if $m = 2^k t$; t odd
$$\left(\frac{m}{n}\right) = \left(\frac{2}{n}\right)^k \left(\frac{t}{n}\right)$$

Check whether it is a polynomial-time algorithm: We can evaluate $a^{\frac{(n-1)}{2}} \bmod n$ in $O\big((\log n)^3\big)$ time. One can evaluate $\left(\frac{a}{n}\right)$ without factoring n.

1. If n is positive and $m_1 \equiv m_2 (\bmod\ n)$ then
$$\left(\frac{m_1}{n}\right) = \left(\frac{m_2}{n}\right)$$

2. If n is positive,
$$\left(\frac{2}{n}\right) = \begin{cases} 1 & \text{if } n \equiv \pm 1 (\bmod\ 8) \\ -1 & \text{if } n \equiv \pm 3 (\bmod\ 8) \end{cases}$$

3. $\left(\frac{m_1 m_2}{n}\right) = \left(\frac{m_1}{n}\right)\left(\frac{m_2}{n}\right)$. In particular, if $m = 2^k t$; t odd
$$\left(\frac{m}{n}\right) = \left(\frac{2}{n}\right)^k \left(\frac{t}{n}\right)$$

4. Suppose m and n are two positive, odd integers.
$$\left(\frac{m}{n}\right) = \begin{cases} -\left(\frac{n}{m}\right) & \text{if } m \equiv n \equiv 3 (\bmod\ 4) \\ \left(\frac{n}{m}\right) & \text{otherwise} \end{cases}$$

- In general, by applying these four properties, it is possible to compute $\left(\frac{a}{n}\right)$ in polynomial time. The only arithmetic operations that are required are modular reductions and factoring out power of 2.

- In general, by applying these four properties, it is possible to compute $\left(\frac{a}{n}\right)$ in polynomial time. The only arithmetic operations that are required are modular reductions and factoring out power of 2.

- It is not difficult to show that at most $O(\log n)$ modular reductions are performed, each of which can be done in time $O\big((\log n)^2\big)$. This shows that the complexity is $O\big((\log n)^3\big)$, which is polynomial in $\log n$.
  $\big($In fact, it can be shown that it is $O((\log n)^2)\big)$.

- In general, by applying these four properties, it is possible to compute $\left(\frac{a}{n}\right)$ in polynomial time. The only arithmetic operations that are required are modular reductions and factoring out power of 2.

- It is not difficult to show that at most $O(\log n)$ modular reductions are performed, each of which can be done in time $O\big((\log n)^2\big)$. This shows that the complexity is $O\big((\log n)^3\big)$, which is polynomial in $\log n$.
  $\big($In fact, it can be shown that it is $O((\log n)^2)\big)$.

- Suppose that we have generated a number n and tested it for primality using the Solovay-Stressan algorithm.
  If we have run the algorithm m times, what is our confidence that n is prime?

  It is $1 - 2^{-m}$.

Miler-Rabin Primality Test

We have an integer of 200 digits to be tested for primality. Why not divide by all the primes less than its square root?

Miler-Rabin Primality Test

We have an integer of 200 digits to be tested for primality. Why not divide by all the primes less than its square root?

There are around, 4×10^{97} primes $< 10^{100}$, which is more than the number of particles in the universe.
Moreover, if a computer can handle 10^9 primes/ sec., the calculation would take 10^{81} years.

Clearly better methods are needed.

Miler-Rabin Primality Test

We have an integer of 200 digits to be tested for primality. Why not divide by all the primes less than its square root?

There are around, 4×10^{97} primes $< 10^{100}$, which is more than the number of particles in the universe.
Moreover, if a computer can handle 10^9 primes/ sec., the calculation would take 10^{81} years.

Clearly better methods are needed.

Basic Principle: Let n be an integer and suppose there exists integers x and y with $x^2 \equiv y^2 \pmod{n}$, but $x \not\equiv \pm y (\text{mod } n)$. Then n is composite. Moreover, $(x - y, n)$ gives a non-trivial factor of n.

Miler-Rabin Primality Test

We have an integer of 200 digits to be tested for primality. Why not divide by all the primes less than its square root?

There are around, 4×10^{97} primes $< 10^{100}$, which is more than the number of particles in the universe.
Moreover, if a computer can handle 10^9 primes/ sec., the calculation would take 10^{81} years.

Clearly better methods are needed.

Basic Principle: Let n be an integer and suppose there exists integers x and y with $x^2 \equiv y^2 \pmod{n}$, but $x \not\equiv \pm y \pmod{n}$. Then n is composite. Moreover, $(x-y, n)$ gives a non-trivial factor of n.

Example: Since $12^2 \equiv 2^2 \pmod{35}$, but $12 \not\equiv 2 \pmod{35}$. 35 is composite and $(12-2, 35) = 5$ is a non-trivial factor of 35.

Factorization and primality testing are not the same. It is much easier to prove that a number is composite than to factor it.

Factorization and primality testing are not the same. It is much easier to prove that a number is composite than to factor it.

Fermat's little Theorem: If p is a prime, then

$$2^{p-1} \equiv 1 (\mathrm{mod}\ p).$$

Factorization and primality testing are not the same. It is much easier to prove that a number is composite than to factor it.

Fermat's little Theorem: If p is a prime, then

$$2^{p-1} \equiv 1 (\text{mod } p).$$

We show that 35 is not prime. By successive squaring, we find

$$\begin{array}{lll} 2^4 & \equiv & 16 \\ 2^8 & \equiv & 256 \quad \equiv 11 \\ 2^{16} & \equiv & 121 \quad \equiv 16 \\ 2^{32} & \equiv & 256 \quad \equiv 11 \end{array}$$

Therefore, $2^{34} \equiv 2^{32}.2^2 \equiv 11.4 \equiv 9 \not\equiv 1 (\text{mod } 35)$. So, it is not a prime.

Factorization and primality testing are not the same. It is much easier to prove that a number is composite than to factor it.

Fermat's little Theorem: If p is a prime, then

$$2^{p-1} \equiv 1 (\text{mod } p).$$

We show that 35 is not prime. By successive squaring, we find

$$\begin{aligned} 2^4 &\equiv 16 \\ 2^8 &\equiv 256 \equiv 11 \\ 2^{16} &\equiv 121 \equiv 16 \\ 2^{32} &\equiv 256 \equiv 11 \end{aligned}$$

Therefore, $2^{34} \equiv 2^{32}.2^2 \equiv 11.4 \equiv 9 \not\equiv 1(\text{mod } 35)$. So, it is not a prime.

So, we have proved that 35 is composite without finding its factors. This method generalizes as : Miller-Rabin Primality Test.

Miler-Rabin Primality Test

Let $n > 1$ be an odd integer. Write

$$n - 1 \equiv 2^k m \text{ with } m \text{ odd.}$$

Choose a random integer a with $1 < a < n - 1$.
Compute $b_0 \equiv a^m (\text{mod } n)$.

if $b_0 \equiv \pm 1 (\text{mod } n)$, **then**

stop and declare that n is probably prime.

otherwise

let $b_1 \equiv {b_0}^2 (\text{mod } n)$.

if $b_1 \equiv 1 (\text{mod } n)$, **then**

n is composite and $(b_0 - 1, n)$ is a factor of n.

if $b_1 \equiv -1 (\text{mod } n)$, **then**

stop and declare that n is probably a prime.

Miler-Rabin Primality Test

otherwise

let $b_2 \equiv {b_1}^2 (\text{mod } n)$.

if $b_2 \equiv 1(\text{mod } n)$, **then**

n is composite and $(b_1 - 1, n)$ is a factor of n.

if $b_2 \equiv -1(\text{mod } n)$, **then**

stop and declare that n is probably a prime.

otherwise

let $b_3 \equiv {b_2}^2 (\text{mod } n)$.

Continue in this way until stopping or reaching b_{k-1}.

If $b_{k-1} \not\equiv -1(\text{mod } n)$, then n is composite.

The Miller-Rabin Algorithm (mod n) :

Write $n-1 = 2^k m$, where m is odd.

Choose a random integer a, $1 \leq a \leq n-1$.

Compute $b = a^m \pmod{n}$.

if $b \equiv 1 \pmod{n}$ **then**

  return ("n is prime") and **quit**

for $i = 0$ **to** $k-1$

$$\textbf{do} \begin{cases} \textbf{if} & b \equiv -1 \pmod{n} \\ \textbf{then} & \text{return ("}n\text{ is prime")} \\ \textbf{else} & b \equiv b^2 \pmod{n} \end{cases}$$

return ("n is composite")

Example

Let $n = 561$. Then $n - 1 = 560 = 16.35$; so $2^k = 2^4$ and $m = 35$. Let $a = 2$. Then

$$\begin{aligned} b_0 &\equiv 2^{35} &\equiv 263 \pmod{561} \\ b_1 &\equiv {b_0}^2 &\equiv 166 \pmod{561} \\ b_2 &\equiv {b_1}^2 &\equiv 67 \pmod{561} \\ b_3 &\equiv {b_2}^2 &\equiv 1 \pmod{561} \end{aligned}$$

as $b_3 \equiv 1 \pmod{561}$, we conclude that 561 is composite. Moreover $(b_2 - 1, 561) = 33$, is a non-trivial factor of 561.

Why does the test work?

Why does the test work?

Suppose, for example, that $b_3 \equiv 1 (\text{mod } n)$. This means that

$$b_2^2 \equiv 1^2 (\text{mod } n).$$

Why does the test work?

Suppose, for example, that $b_3 \equiv 1 (\text{mod } n)$. This means that

$$b_2^2 \equiv 1^2 (\text{mod } n).$$

Apply the 'basic principle':

Either $b_2 \equiv \pm 1 (\text{mod } n)$ or $b_2 \not\equiv \pm 1 (\text{mod } n)$ and n is composite. In the former case, the algorithm would have stopped by the previous step.

Why does the test work?

Suppose, for example, that $b_3 \equiv 1 (\text{mod } n)$. This means that

$$b_2^2 \equiv 1^2 (\text{mod } n).$$

Apply the 'basic principle':

Either $b_2 \equiv \pm 1 (\text{mod } n)$ or $b_2 \not\equiv \pm 1 (\text{mod } n)$ and n is composite. In the former case, the algorithm would have stopped by the previous step.
If we reach b_{k-1}, we have computed

$$b_{k-1} \equiv a^{\frac{(n-1)}{2}} (\text{mod } n)$$

The square is a^{n-1}, which must be $1 (\text{mod } n)$ if n is prime.

Why does the test work?

Suppose, for example, that $b_3 \equiv 1 (\mathrm{mod}\ n)$. This means that

$$b_2^2 \equiv 1^2 (\mathrm{mod}\ n).$$

Apply the 'basic principle':

Either $b_2 \equiv \pm 1 (\mathrm{mod}\ n)$ or $b_2 \not\equiv \pm 1 (\mathrm{mod}\ n)$ and n is composite. In the former case, the algorithm would have stopped by the previous step.

If we reach b_{k-1}, we have computed

$$b_{k-1} \equiv a^{\frac{(n-1)}{2}} (\mathrm{mod}\ n)$$

The square is a^{n-1}, which must be $1 (\mathrm{mod}\ n)$ if n is prime.

So, if n is prime, $b_{k-1} \equiv \pm 1 (\mathrm{mod}\ n)$, all other choices means n is composite.

Moreover, if $b_{k-1} = 1$, then, if we didn't stop at an earlier step,

$b_{k-2}^2 \equiv 1^2 (\mathrm{mod}\ n)$ with $b_{k-2} \not\equiv \pm 1 (\mathrm{mod}\ n)$.

$\Rightarrow n$ is composite.