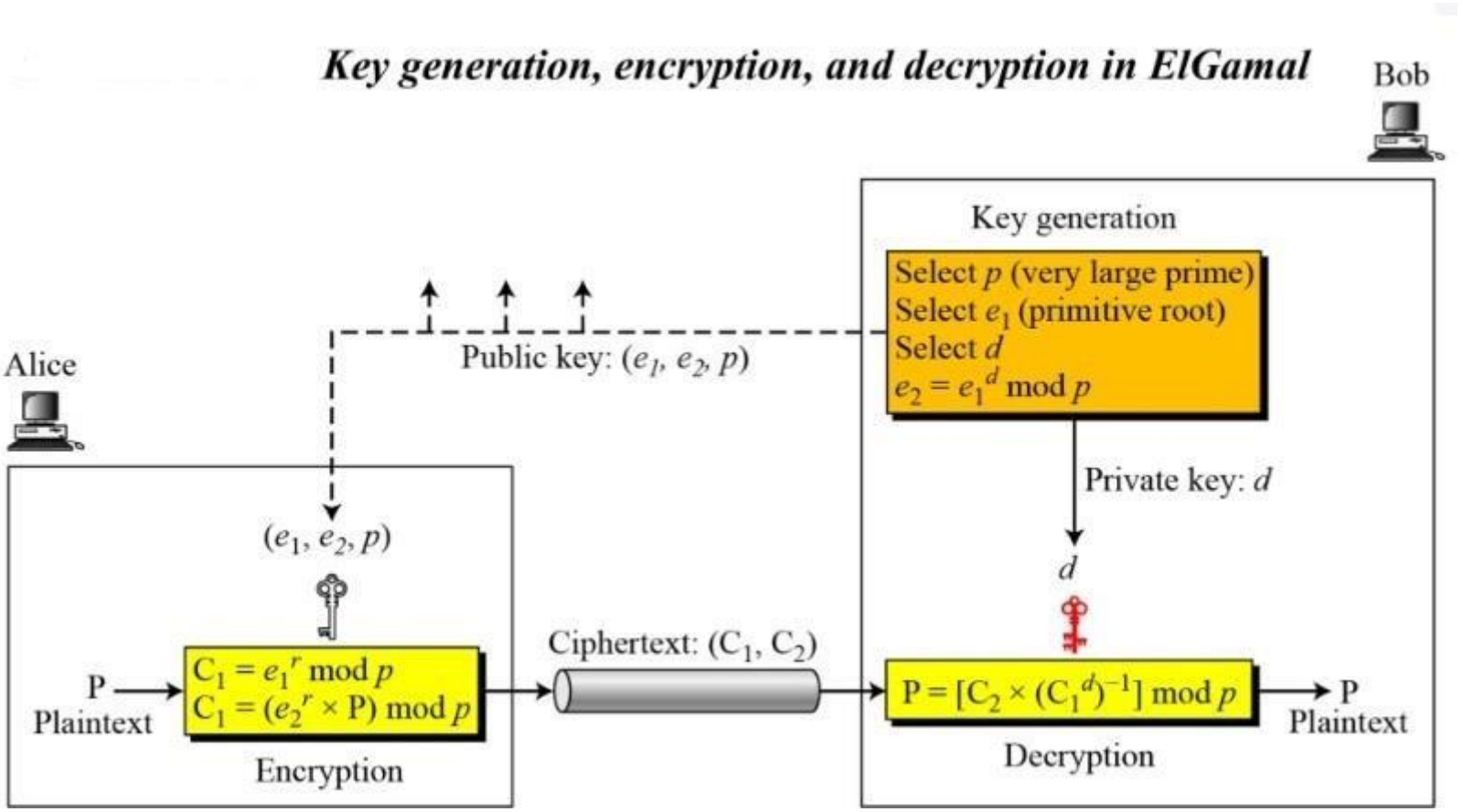
# Public Key Cryptosystem

**Sachin Tripathi**

IIT(ISM), Dhanbad

# ElGamal Cryptosystem



# Key Generation

**ElGamal_Key_Generation**

{

Select a large prime $p$

Select $d$ to be a member of the group $\mathbf{G} = < \mathbf{Z}_p^*, \times >$ such that $1 \leq d \leq p - 2$

Select $e_1$ to be a primitive root in the group $\mathbf{G} = < \mathbf{Z}_p^*, \times >$

$e_2 \leftarrow e_1^{\,d} \bmod p$

Public_key $\leftarrow (e_1, e_2, p)$ // To be announced publicly

Private_key $\leftarrow d$ // To be kept secret

return Public_key and Private_key

}

# Encryption

**ElGamal_Encryption** ($e_1$, $e_2$, $p$, P) // P is the plaintext

{

Select a random integer $r$ in the group $\mathbf{G} = < \mathbf{Z}_p^*, \times >$

$C_1 \leftarrow e_1^{\,r} \bmod p$

$C_2 \leftarrow (P \times e_2^{\,r}) \bmod p$ // $C_1$ and $C_2$ are the ciphertexts

return $C_1$ and $C_2$

}

# Decryption

```
ElGamal_Decryption (d, p, C1, C2)        // C1 and C2 are the ciphertexts
{
    P ← [C2 (C1^d)^-1] mod p             // P is the plaintext
    return P
}
```

**Note: Bit operation complexity of encryption and decryption in Elgamal Cryptosytem is polynomial**

# Proof

The ElGamal decryption expression $C_2 \times (C_1^{\,d})^{-1}$ can be verified to be P through substitution:

$$[C_2 \times (C_1^{\,d})^{-1}] \bmod p = [(e_2^{\,r} \times P) \times (e_1^{\,rd})^{-1}] \bmod p = (e_1^{\,dr}) \times P \times (e_1^{\,rd})^{-1} = P$$

# Example

***Here is a trivial example. Bob chooses p = 11 and $e_1 = 2$. and d = 3 $e_2 = e_1^d = 8$. So the public keys are (2, 8, 11) and the private key is 3. Alice chooses r = 4 and calculates C1 and C2 for the plaintext 7.***

**Plaintext: 7**

$C_1 = e_1^r \bmod 11 = 16 \bmod 11 = 5 \bmod 11$

$C_2 = (P \times e_2^r) \bmod 11 = (7 \times 4096) \bmod 11 = 6 \bmod 11$

**Ciphertext:** (5, 6)

***Bob receives the ciphertexts (5 and 6) and calculates the plaintext.***

$[C_2 \times (C_1^d)^{-1}] \bmod 11 = 6 \times (5^3)^{-1} \bmod 11 = 6 \times 3 \bmod 11 = 7 \bmod 11$

**Plaintext: 7**

Instead of using $P = [C_2 \times (C_1^d)^{-1}] \bmod p$ for decryption, we can avoid the calculation of multiplicative inverse and use $P = [C_2 \times C_1^{p-1-d}] \bmod p$ (see Fermat's little theorem

# Security of Elgamal

- Low Modulus Attack

If the value of $p$ is not large enough, Eve can use some efficient algorithms to solve the discrete logarithm problem to find $d$ or $r$. If $p$ is small, Eve can easily find $d = \log_{e1} e_2 \bmod p$ and store it to decrypt any message sent to Bob.This can be done once and used as long as Bob uses the same keys. Eve can also use the value of $C_1$ to find random number $r$ used by Alice in each transmission $r = \log_{e1} C_1 \bmod p$.

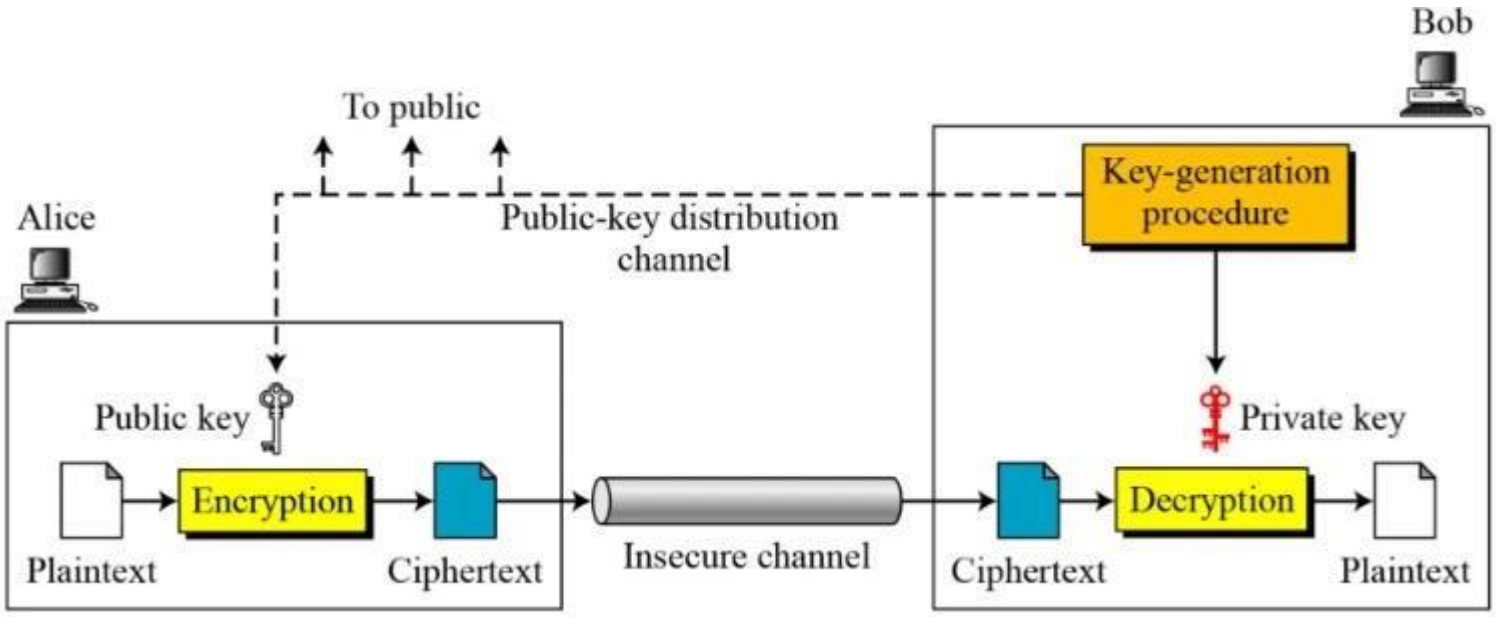
## Known Plaintext Attack

If Alice uses the same random exponent r, to encrypt two plaintexts P and P′, Eve discovers P′ if she knows P. Assume that $C_2 = P \times (e_2{}^r) \bmod p$ and $C'_2 = P' \times (e_2{}^r) \bmod p$. Eve finds P′ using the following steps:

1. $(e_2{}^r) = C_2 \times P^{-1} \bmod p$
2. $P' = C'_2 \times (e_2{}^r)^{-1} \bmod p$

It is recommended that Alice use a fresh value of $r$ to thwart the known-plaintext attacks.

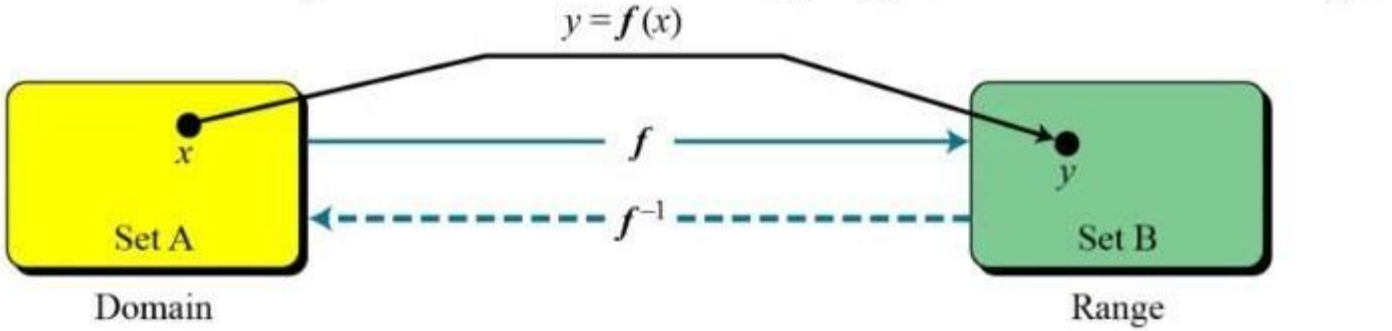
# General Idea of Public Cryptosystem



# Trapdoor One Way Function

***Functions***

***A function as rule mapping a domain to a range***



## *One-Way Function (OWF)*

***1. $f$ is easy to compute.***
***2. $f^{-1}$ is difficult to compute.***

## *Trapdoor One-Way Function (TOWF)*

***3. Given y and a trapdoor, x can be computed easily.***

# Cryptanalysis

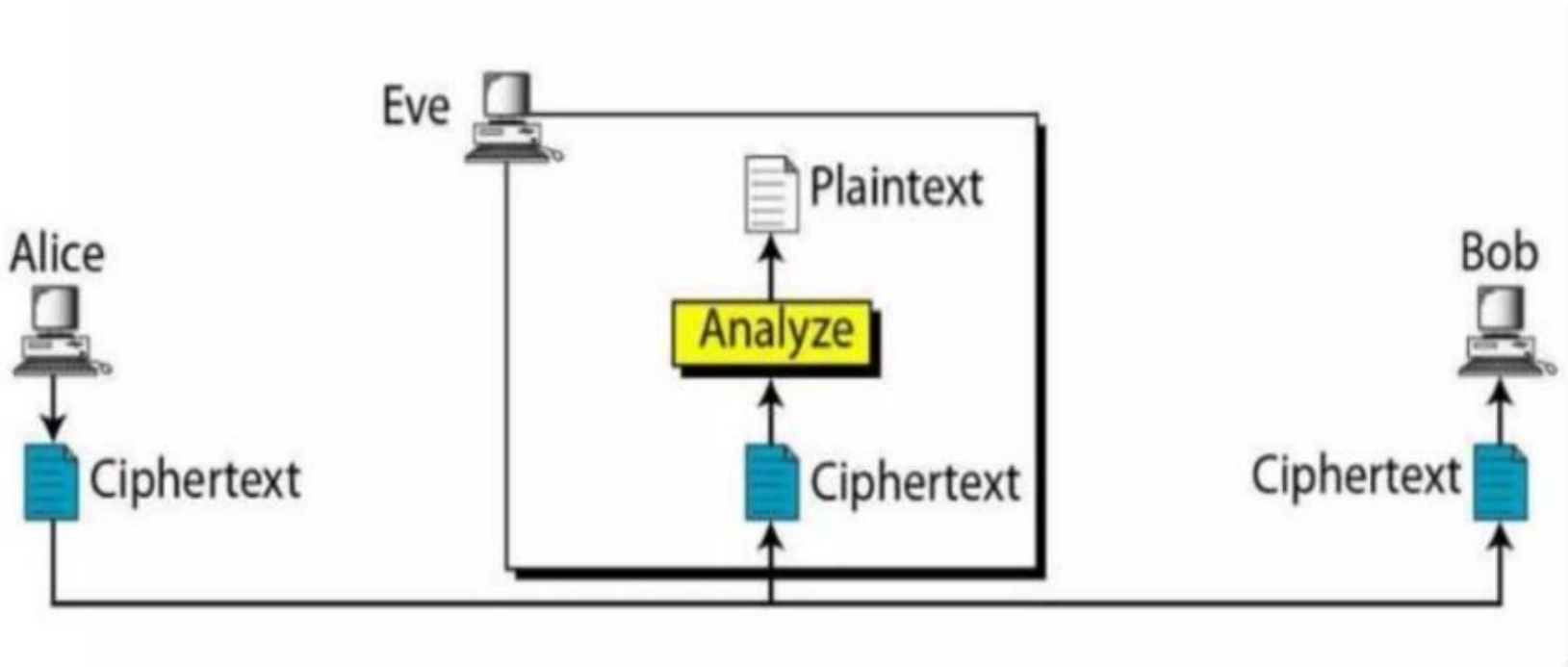
- Cryptanalysis is the art of trying to decrypt the encrypted messages without the use of the key that was used to encrypt the messages. Cryptanalysis uses mathematical analysis & algorithms to decipher the ciphers.
- The success of cryptanalysis attacks depends
  - Amount of time available
  - Computing power available
  - Storage capacity available

- **Known-Plaintext Analysis (KPA):** Attacker decrypts ciphertext with known partial plaintext.
- **Chosen-Plaintext Analysis (CPA):** Attacker uses ciphertext that matches arbitrarily selected plaintext via the same algorithm technique.
- **Ciphertext-Only Analysis (COA):** Attacker uses known ciphertext collections.
- **Man-in-the-Middle (MITM) Attack:** Attack occurs when two parties use message or key sharing for communication via a channel that appears secure but is actually compromised.

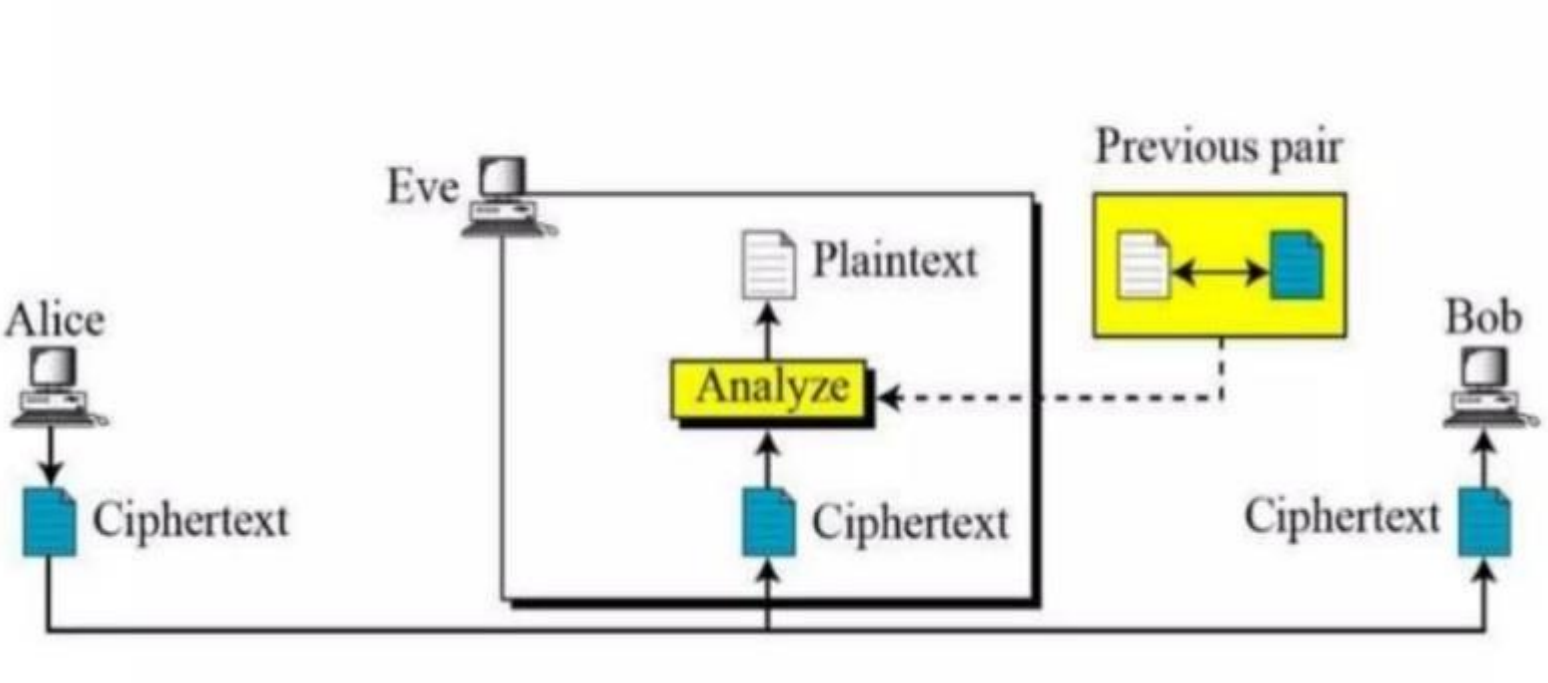
# Types

- **Brute force attack**– this type of attack uses algorithms that try to guess all the possible logical combinations of the plaintext which are then ciphered and compared against the original cipher.
- **Dictionary attack–** this type of attack uses a wordlist in order to find a match of either the plaintext or key. It is mostly used when trying to crack encrypted passwords.
- **Rainbow table attack–** this type of attack compares the cipher text against pre- computed hashes to find matches.

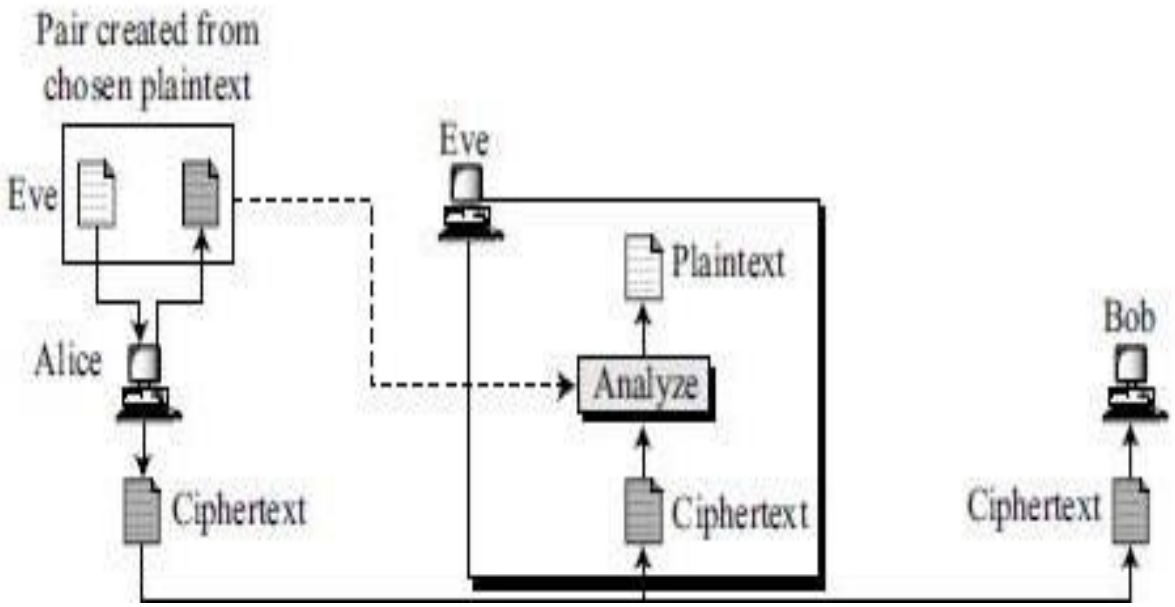
# Cipher text Only Attack



# Known Plaintext Attack

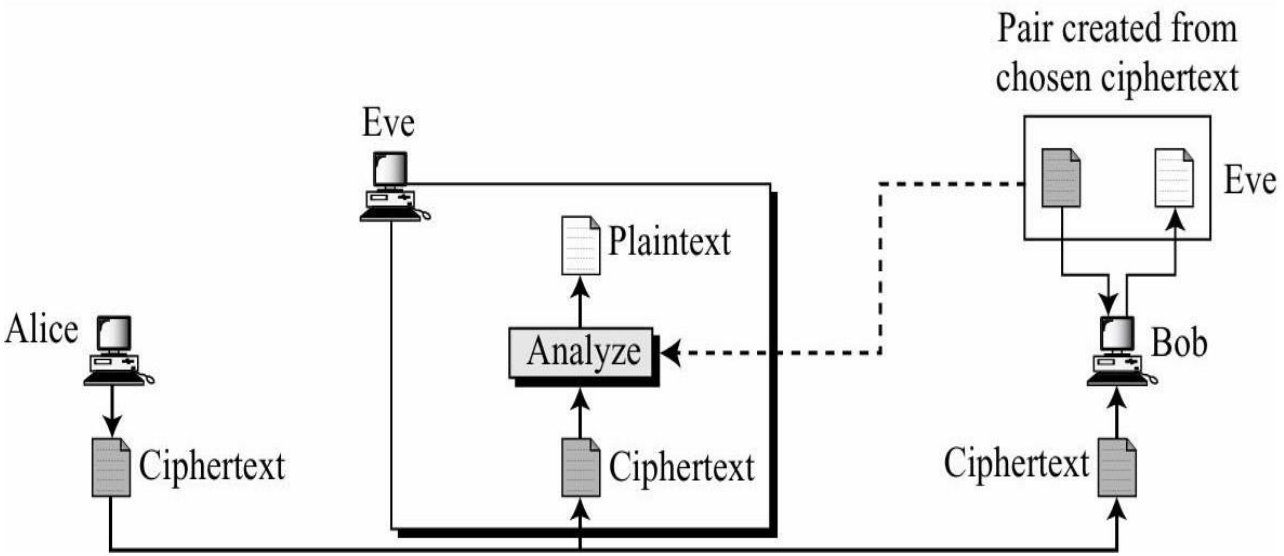


# Chosen Plaintext Attack



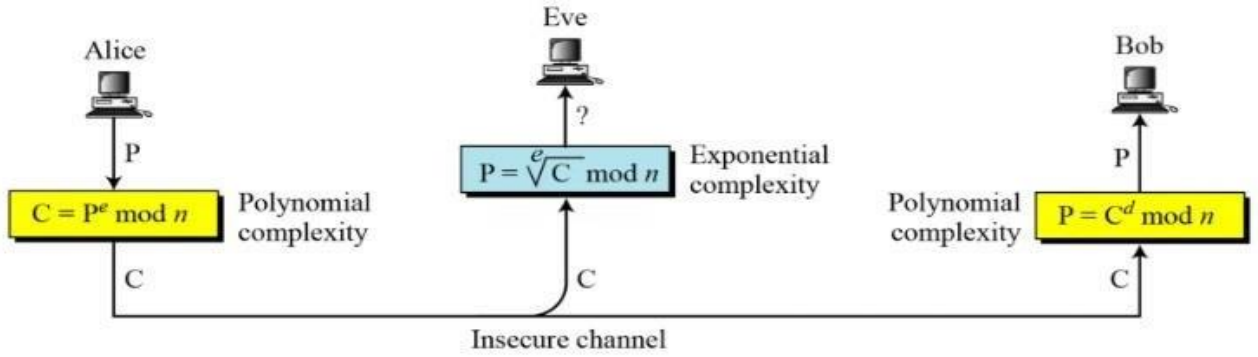
# Chosen Cipher text Attack

- Similar to the chosen plaintext attack, except that Eve chooses some cipher text and decrypt it to form a cipher text/plaintext pair



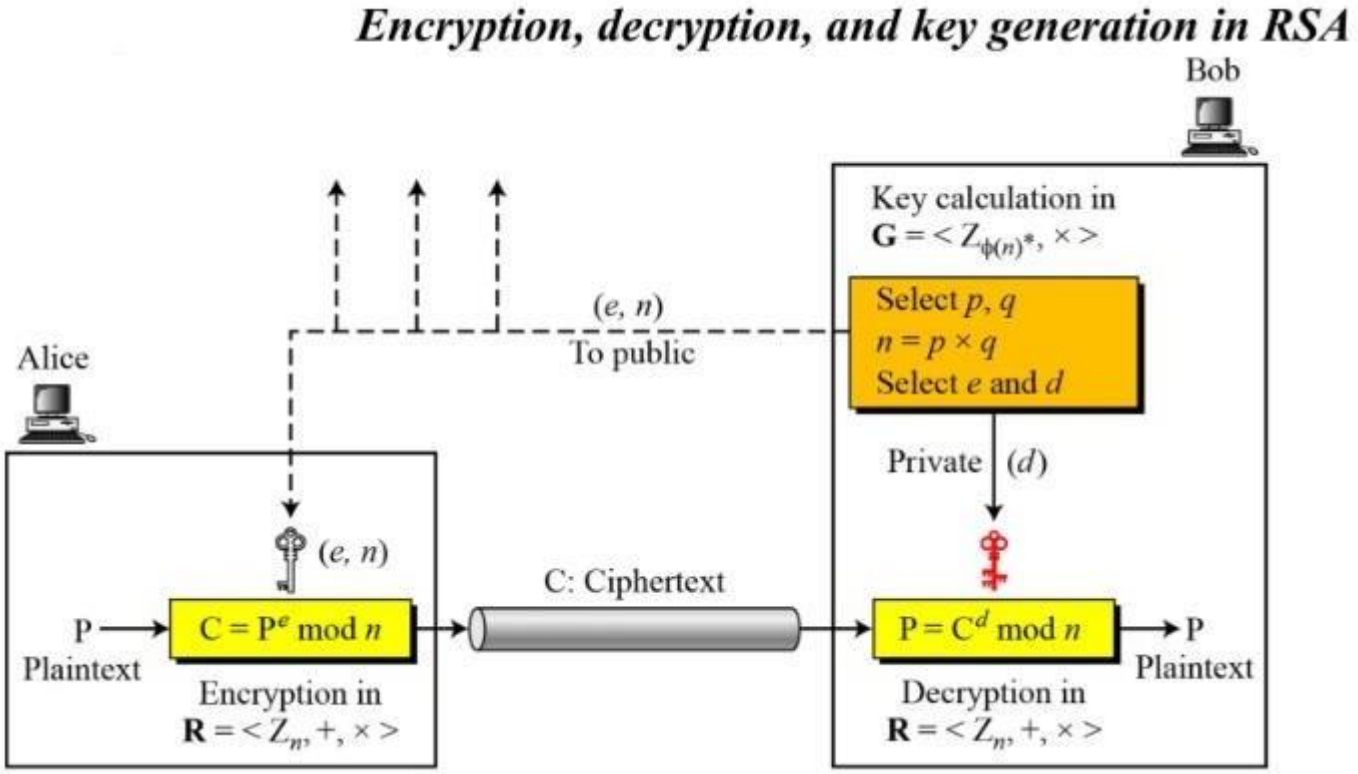
# RSA Cryptosystem

- The most common public key algorithm (Rivest, Shamir and Adleman)
- Complexity of operations in RSA




**RSA uses modular exponentiation for encryption/decryption; To attack it, Eve needs to calculate $\sqrt[e]{C} \bmod n$.**

# RSA Algorithm



## *Two Algebraic Structures*

***Encryption/Decryption Ring:*** $R = \langle Z_n, +, \times \rangle$

***Key-Generation Group:*** $G = \langle Z_{\phi(n)}^{*}, \times \rangle$

**RSA uses two algebraic structures:**
**a public ring $R = \langle Z_n, +, \times \rangle$ and a private group $G = \langle Z_{\phi(n)}^{*}, \times \rangle$.**

**In RSA, the tuple $(e, n)$ is the public key; the integer $d$ is the private key.**

# RSA Key Generation

**RSA_Key_Generation**

{

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

$n \leftarrow p \times q$

$\phi(n) \leftarrow (p-1) \times (q-1)$

Select $e$ such that $1 < e < \phi(n)$ and $e$ is coprime to $\phi(n)$

$d \leftarrow e^{-1} \bmod \phi(n)$ // $d$ is inverse of $e$ modulo $\phi(n)$

Public_key $\leftarrow$ (e, $n$) // To be announced publicly

Private_key $\leftarrow d$ // To be kept secret

return Public_key and Private_key

}

# Encryption

```
RSA_Encryption (P, e, n)                    // P is the plaintext in Z_n and P < n
{
    C ← Fast_Exponentiation (P, e, n)       // Calculation of (P^e mod n)
    return C
}
```

**In RSA, $p$ and $q$ must be at least 512 bits; $n$ must be at least 1024 bits.**

# Decryption

```
RSA_Decryption (C, d, n)                    //C is the ciphertext in Z_n
{
    P ← Fast_Exponentiation (C, d, n)       // Calculation of (C^d mod n)
    return P
}
```

# Proof of RSA

If $n = p \times q$, $a < n$, and $k$ is an integer, then $a^{k \times \phi(n)+1} \equiv a \pmod{n}$.

$$P_1 = C^d \bmod n = (P^e \bmod n)^d \bmod n = P^{ed} \bmod n$$

$ed = k\phi(n) + 1$ // $d$ and $e$ are inverses modulo $\phi(n)$

$P_1 = P^{ed} \bmod n \rightarrow P_1 = P^{k\phi(n)+1} \bmod n$

$P_1 = P^{k\phi(n)+1} \bmod n = P \bmod n$ // Euler's theorem (second version)

# Examples

Bob chooses 7 and 11 as *p* and *q* and calculates *n* = 77. The value of $\phi(n) = (7 - 1)(11 - 1)$ or 60. Now he chooses two exponents, *e* and *d*, from $Z_{60}^*$. If he chooses *e* to be 13, then d is 37. Note that $e \times d \bmod 60 = 1$ (they are inverses of each Now imagine that Alice wants to send the plaintext 5 to Bob. She uses the public exponent 13 to encrypt 5.

| | | |
|---|---|---|
| Plaintext: 5 | $C = 5^{13} = 26 \bmod 77$ | Ciphertext: 26 |

Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:

| | | |
|---|---|---|
| Ciphertext: 26 | $P = 26^{37} = 5 \bmod 77$ | Plaintext: 5 |

Now assume that another person, John, wants to send a message to Bob. John can use the same public key announced by Bob (probably on his website), 13; John's plaintext is 63. John calculates the following:

| | | |
|---|---|---|
| Plaintext: 63 | $C = 63^{13} = 28 \bmod 77$ | Ciphertext: 28 |

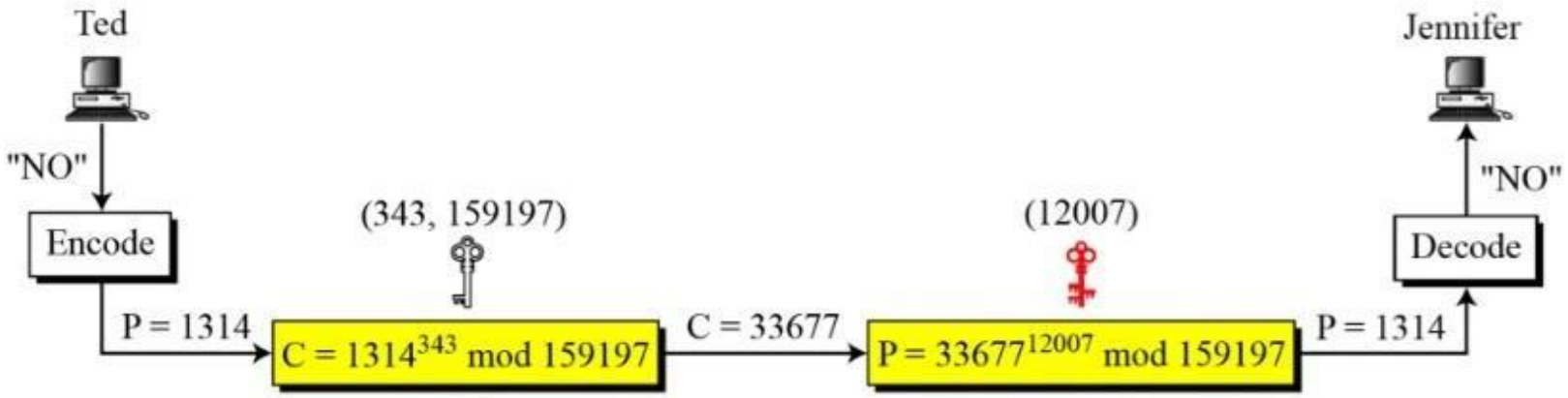
Bob receives the ciphertext 28 and uses his private key 37 to decipher the ciphertext:

| | | |
|---|---|---|
| Ciphertext: 28 | $P = 28^{37} = 63 \bmod 77$ | Plaintext: 63 |

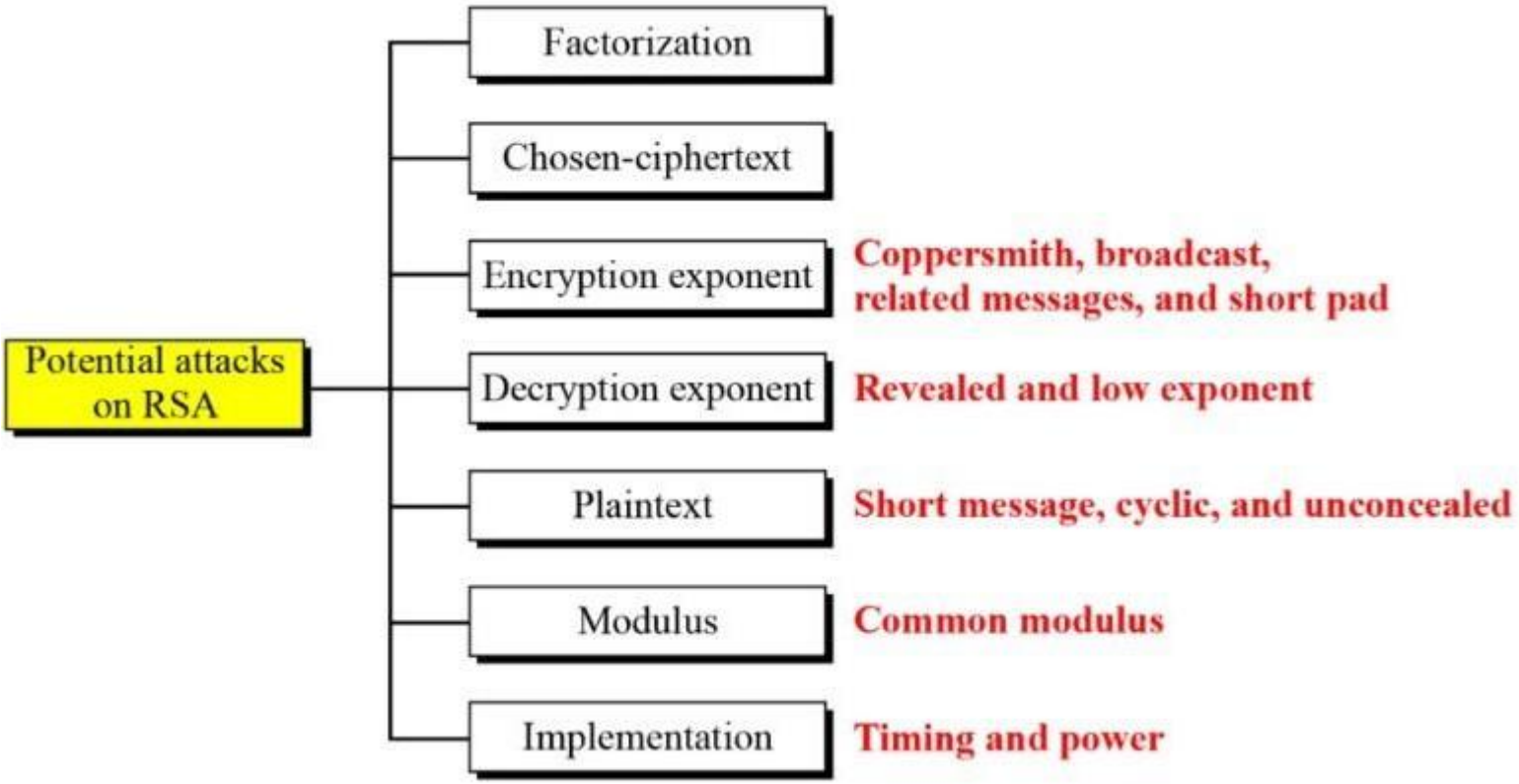
Jennifer creates a pair of keys for herself. She chooses $p$ = 397 and $q$ = 401. She calculates $n$ = 159197. She then calculates $\phi(n)$ = 158400. She then chooses e = 343 and d = 12007. Show how Ted can send a message to Jennifer if he knows $e$ and $n$.

Suppose Ted wants to send the message "NO" to Jennifer. He changes each character to a number (from 00 to 25), with each character coded as two digits. He then concatenates the two coded characters and gets a four-digit number. The plaintext is 1314.

## *Encryption and decryption in Example*



# Attacks on RSA



# Factorization Attack

The security of RSA is based on the idea that the modulus is so large that it is infeasible to factor it in a reasonable time. Bob selects $p$ and $q$ and calculates $n = p \times q$. Although $n$ is public, $p$ and $q$ are secret. If Eve can factor $n$ and obtain $p$ and $q$, she can calculate $\phi(n) = (p - 1)(q - 1)$. Eve then can calculate $d = e^{-1} \bmod \phi(n)$ because $e$ is public. The private exponent $d$ is the trapdoor that Eve can use to decrypt any encrypted message.

# Chosen Cipher text Attack

A potential attack on RSA is based on the multiplicative property of RSA. Assume that Alice creates the ciphertext $C = P^e \bmod n$ and sends C to Bob. Also assume that Bob will decrypt an arbitrary ciphertext for Eve, other than C. Eve intercepts C and uses the following steps to find P:

a. Eve chooses a random integer X in $Z_n^*$.
b. Eve calculates $Y = C \times X^e \bmod n$.
c. Eve sends Y to Bob for decryption and get $Z = Y^d \bmod n$; This step is an instance of a chosen-ciphertext attack.
d. Eve can easily find P because

$$Z = Y^d \bmod n = (C \times X^e)^d \bmod n = (C^d \times X^{ed}) \bmod n = (C^d \times X) \bmod n = (P \times X) \bmod n$$
$$Z = (P \times X) \bmod n \quad \rightarrow \quad P = Z \times X^{-1} \bmod n$$

Eve uses the extended Euclidean algorithm to find the multiplicative inverse of X and eventually the value of P.

# Attacks on Encryption Exponent

- These attacks do not generally result in a breakdown of the system, but they still need to be prevented. To thwart these kinds of attacks, the recommendation is to use $e = 2^{16} + 1 = 65537$ (or a prime close to this value).
- Coppersmith Theorem Attack The major low encryption exponent attack is referred to as the Coppersmith theorem attack. This theorem states that in a modulo-n polynomial f(x) of degree e, one can use an algorithm of the complexity log n to find the roots if one of the roots is smaller than $n^{1/e}$.

# Broadcast Attack

The **broadcast attack** can be launched if one entity sends the same message to a group of recipients with the same low encryption exponent. For example, assume the following scenario: Alice wants to send the same message to three recipients with the same public exponent $e = 3$ and the moduli $n_1$, $n_2$, and $n_3$.

$$C_1 = P^3 \bmod n_1 \qquad C_2 = P^3 \bmod n_2 \qquad C_3 = P^3 \bmod n_3$$

Applying the Chinese remainder theorem to these three equations, Eve can find an equation of the form $C' = P^3 \bmod n_1 n_2 n_3$. This means that $P^3 < n_1 n_2 n_3$. This means $C' = P^3$ is in regular arithmetic (not modular arithmetic). Eve can find the value of $C' = P^{1/3}$.

# Related Message Attack

- The related message attack, discovered by Franklin Reiter, can be briefly described as follows. Alice encrypts two plaintexts, P1 and P2, and encrypts them with e = 3 and sends C1 and C2 to Bob. If P1 is related to P2 by a linear function, then Eve can recover P1 and P2 in a feasible computation time.

# Short pad Attack

The **short pad attack,** discovered by Coppersmith, can be briefly described as follows. Alice has a message M to send to Bob. She pads the message with $r_1$, encrypts the result to get $C_1$, and sends $C_1$ to Bob. Eve intercepts $C_1$ and drops it. Bob informs Alice that he has not received the message, so Alice pads the message again with $r_2$, encrypts it, and sends it to Bob. Eve also intercepts this message. Eve now has $C_1$ and $C_2$, and she knows that they both are ciphertexts belonging to the same plaintext. Coppersmith proved that if $r_1$ and $r_2$ are short, Eve may be able to recover the original message M.

# Attacks on Decryption exponent

**Revealed Decryption Exponent Attack** It is obvious that if Eve can find the decryption exponent, $d$, she can decrypt the current encrypted message. However, the attack does not stop here. If Eve knows the value of $d$, she can use a probabilistic algorithm (not discussed here) to factor $n$ and find the value of $p$ and $q$. Consequently, if Bob changes only the compromised decryption exponent but keeps the same modulus, $n$, Eve will be able to decrypt future messages because she has the factorization of $n$. This means that if Bob finds out that the decryption exponent is compromised, he needs to choose new value for $p$ and $q$, calculate $n$, and create totally new private and public keys.

---

**In RSA, if $d$ is comprised, then $p$, $q$, $n$, $e$, and $d$ must be regenerated.**

---

**Low Decryption Exponent Attack** Bob may think that using a small private-key $d$, would make the decryption process faster for him. Wiener showed that if $d < 1/3\ n^{1/4}$, a special type of attack based on *continuous fraction*, a topic discussed in number theory, can jeopardize the security of RSA. For this to happen, it must be the case that $q < p < 2q$. If these two conditions exist, Eve can factor $n$ in polynomial time.

---

**In RSA, the recommendation is to have $d \geq 1/3\ n^{1/4}$ to prevent low decryption exponent attack.**

---

# Plaintext Attack

- Plaintext and cipher-text in RSA are permutations of each other because they are integers in the same interval (0 to n − 1). In other words, Eve already knows something about the plaintext.This characteristic may allow some attacks on the plaintext.Three attacks have been mentioned in the literature: short message attack, cycling attack, and unconcealed attack.

# Short Message Attack

In the **short message attack,** if Eve knows the set of possible plaintexts, she then knows one more piece of information in addition to the fact that the ciphertext is the permutation of plaintext. Eve can encrypt all of the possible messages until the result is the same as the ciphertext intercepted. For example, if it is known that Alice is sending a four-digit number to Bob, Eve can easily try plaintext numbers from 0000 to 9999 to find the plaintext. For this reason, short messages must be padded with random bits at the front and the end to thwart this type of attack. It is strongly recommended that messages be padded with random bits before encryption using a method called OAEP

# Cycling Attack

- The cycling attack is based on the fact that if the ciphertext is a permutation of the plaintext, the continuous encryption of the ciphertext will eventually result in the plaintext.
- However, Eve does not know what the plaintext is, so she does not know when to stop. She needs to go one step further.When she gets the ciphertext C again, she goes back one step to find the plaintext.

Intercepted ciphertext: C

$C_1 = C^e \mod n$

$C_2 = C_1^{\,e} \mod n$

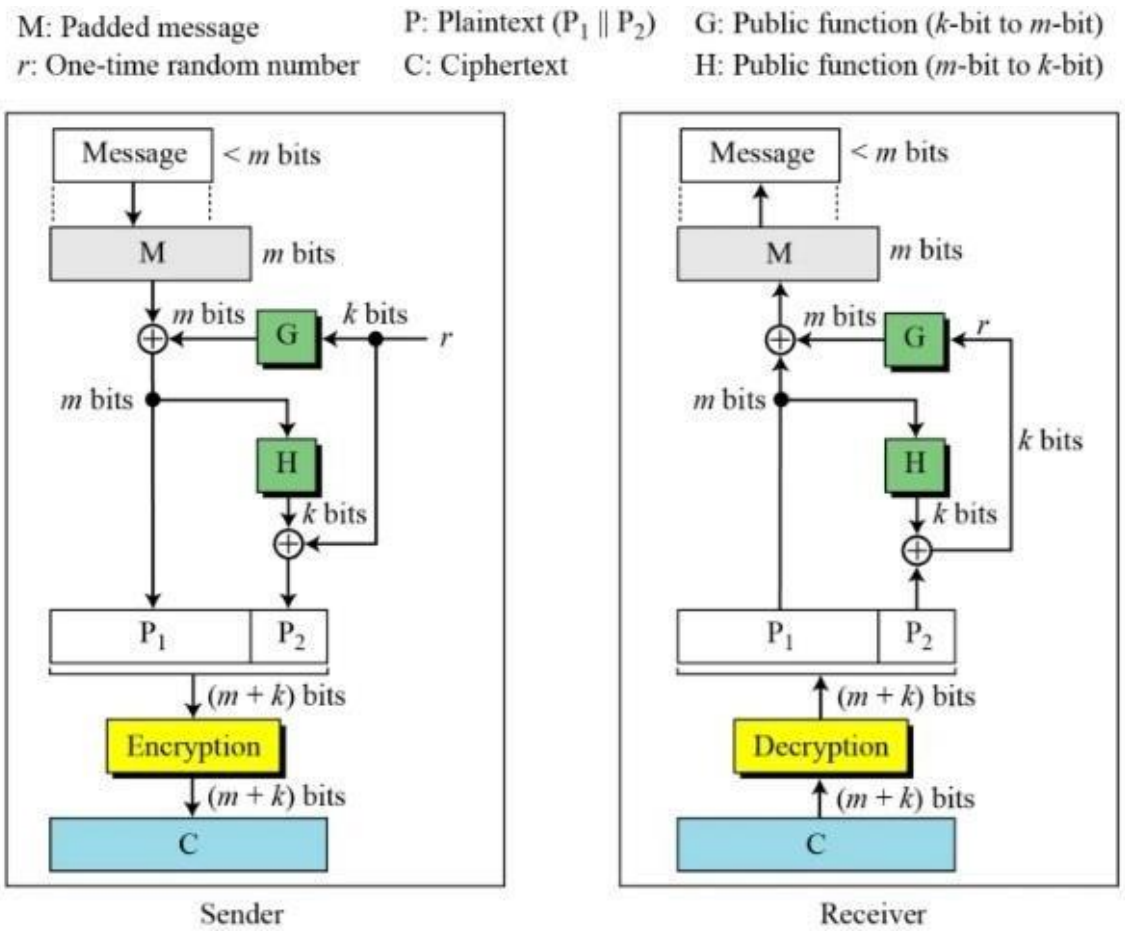
...

$C_k = C_{k-1}^{\,e} \mod n \rightarrow$ If $C_k = C$, stop: the plaintext is $P = C_{k-1}$

# Unconcealed Message Attack

unconcealed message is a message that encrypts to itself (cannot be concealed). It has been proven that there are always some messages that are encrypted to themselves. Because the encryption exponent normally is odd, there are some plaintexts that are encrypted to themselves such as P = 0 and P = 1. Although there are more, if the encrypting exponent is selected carefully, the number of these message is negligible. The encrypting program can always check if the calculated ciphertext is the same as the plaintext and reject the plaintext before submitting the ciphertext.

# Optimal Asymmetric encryption Padding



# Encryption

1. Alice pads the message to make an $m$-bit message, which we call M.
2. Alice chooses a random number $r$ of $k$ bits. Note that $r$ is used only once and is then destroyed.
3. Alice uses a public one-way function, G, that takes an $r$-bit integer and creates an $m$-bit integer ($m$ is the size of M, and $r < m$). This is the mask.
4. Alice applies the mask $G(r)$ to create the first part of the plaintext $P_1 = M \oplus G(r)$. $P_1$ is the masked message.
5. Alice creates the second part of the plaintext as $P_2 = H(P_1) \oplus r$. The function H is another public function that takes an $m$-bit input and creates an $k$-bit output. $P_2$ is used to allow Bob to recreate the mask after decryption.
6. Alice creates $C = P^e = (P_1 \,||\, P_2)^e$ and sends C to Bob.

# Decryption

The following shows the decryption process:

1. Bob creates $P = C^d = (P_1 \| P_2)$.
2. Bob first recreates the value of $r$ using $H(P_1) \oplus P_2 = H(P_1) \oplus H(P_1) \oplus r = r$.
3. Bob uses $G(r) \oplus P = G(r) \oplus G(r) \oplus M = M$ to recreate the value of the padded message.
4. After removing the padding from M, Bob finds the original message.

# Example (Realistic)

**Here is a more realistic example. We choose a 512-bit $p$ and $q$, calculate $n$ and $\phi(n)$, then choose $e$ and test for relative primeness with $\phi(n)$. We then calculate $d$. Finally, we show the results of encryption and decryption. The integer $p$ is a 159-digit number.**

| | |
|---|---|
| $p =$ | 96130345313583504574191581280615427909309845594996215822583150879647940455056470638491257160180347503120986666064924201918087806674210960633542199266612 09 |
| $q =$ | 12060191957231446918276794204450896001555925054637033936061798321731482148483764659215389453209175225273226830107120695604602513887145524969000359660045617 |

**The modulus $n = p \times q$. It has 309 digits.**

$n =$

1159350417396761496889250986461588752377145737545414477548552613761478854083263508172768788159683251684688493006254857641112501624145523391829271625076567727274600970827141277304349605005563472745666280600999240371029914244722922157727985317270338393813346926841373276220009666766718318310883734208234443709 53

**$\phi(n) = (p - 1)(q - 1)$ has 309 digits.**

$\phi(n) =$

1159350417396761496889250986461588752377145737545414477548552613761478854083263508172768788159683251684688493006254857641112501624145523391829271625076567510542336084929167520344826279881175547876570139234444057169895817281960982263610754672118646121713591073586406140088851702653772772644673410662438576641 28

**Bob chooses e = 35535 (the ideal is 65537) and tests it to make sure it is relatively prime with ϕ(n). He then finds the inverse of *e* modulo ϕ(n) and calls it *d*.**

| | |
|---|---|
| $e$ = | 35535 |
| $d$ = | 5800830286003776393609366128967791759466906208965096218042286611138059385282235873170628691003002171085904433840217072986908760061153062025249598844480475682409662470814858171304632406440777048331340108509473852956450719367740611973265574242372176176746207763716420760033708533328853214470885955136670294831 |

**Alice wants to send the message "THIS IS A TEST", which can be changed to a numeric value using the 00–26 encoding scheme (26 is the space character).**

| | |
|---|---|
| P = | 190708182608182600261904 1819 |

**The ciphertext calculated by Alice is $C = P^e$, which is**

| | |
|---|---|
| C = | 475309123646226827206365550610545180942371796070491716523239243054<br>452960613199328566617843418359114151197411252005682979794571736036<br>101278218847892741566090480023507190715277185914975188465888632101<br>148354103361657898467968386763733765777465625079280521148141844048<br>141844308127730590046928742485591664621086 56 |

**Bob can recover the plaintext from the ciphertext using $P = C^d$, which is**

| | |
|---|---|
| P = | 1907081826081826002619041819 |

**The recovered plaintext is "THIS IS A TEST" after decoding.**