Asymmetric-Key Cryptography

19CSE311 Computer Security

Jevitha KP
Department of CSE

Symmetric vs Asymmetric-Key Cryptography

- Symmetric and asymmetric-key cryptography are complements of each other

- The differences between the two systems are based on how these systems keep a secret.

- In symmetric-key cryptography, the **secret must be shared** between two persons.

- In asymmetric-key cryptography, the **secret is unshared**; each person creates and keeps his or her own secret

Symmetric vs Asymmetric-Key Cryptography

- For n people, **n(n – 1)/2 shared secrets** are needed for **symmetric-key cryptography**; only **n personal secrets** are needed in **asymmetric-key cryptography.**

- For a population of 1 million, symmetric-key cryptography would require **half a billion shared secrets**; asymmetric-key cryptography would require **1 million personal secrets**

- Symmetric-key cryptography is based on **sharing secrecy;**

- asymmetric-key cryptography is based on **personal secrecy.**

Symmetric vs Asymmetric-Key Cryptography

Symmetric-key cryptography

- Secret must be shared - Sharing Secrecy
- Based on **substitution and permutation** of symbols (characters or bits),
- Plaintext and ciphertext are thought of as a combination of **symbols**.
- Encryption and decryption **permute** these symbols or substitute a symbol for another.

Asymmetric-key cryptography

- Secret is unshared - Personal Secrecy
- Based on applying **mathematical functions to numbers**.
- Plaintext and ciphertext are **numbers**;
- encryption and decryption are **mathematical functions** that are applied to numbers to create other numbers.

Need for both

- Asymmetrickey (public-key) cryptography **does not eliminate the need** for symmetric-key (secretkey) cryptography.

- Asymmetric-key cryptography, which uses mathematical functions for encryption and decryption, is much **slower** than symmetric-key cryptography.

- For **encipherment of large messages**, **symmetric-key cryptography i**s still needed.

- The speed of symmetric-key cryptography does not eliminate the need for asymmetric-key cryptography.

- Asymmetric-key cryptography is still needed for **authentication, digital signatures, and secret-key exchanges.**

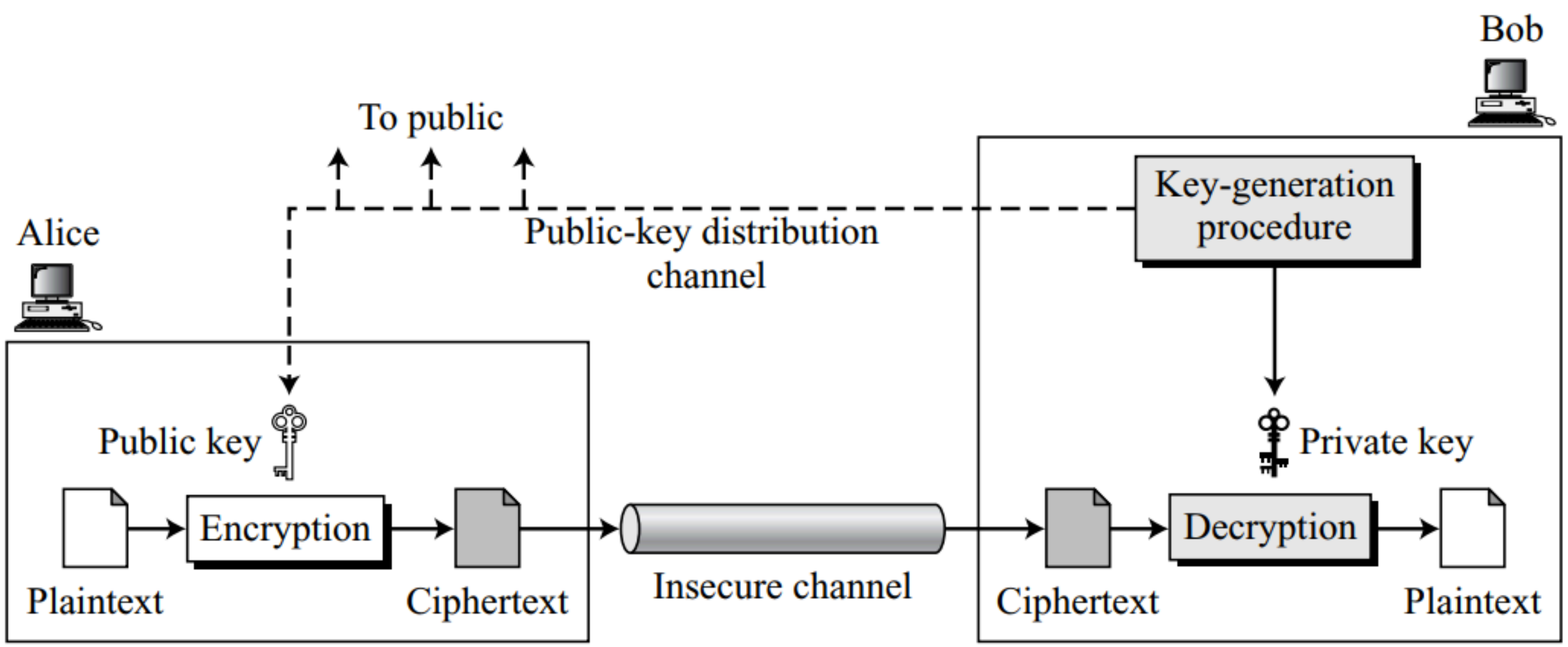
- This means that, to be able to use all aspects of security today, we need both symmetric-key and asymmetric-key cryptography.

- One complements the other.

Keys

- Asymmetric key cryptography uses two separate keys: **one private and one public**.

- The term Secret key is best used with Symmetric key crypto systems (string of symbols) vs private key (set of numbers)



Keys

- The burden of providing security is mostly on the shoulders of the **receiver** (Bob, in this case).
- Bob needs to create two keys: one private and one public.
- Bob is responsible for **distributing the public key to the community**.
- This can be done through a **public-key distribution channel**.
- Although this channel is not required to provide secrecy, it must provide **authentication and integrity.**
- **Eve should not be able to advertise her public key to the community pretending that it is Bob's public key**

Keys

- Two entities cannot use the **same set of keys for two-way communication**.
- Each entity in the community should create its own private and public keys.
- If Bob wants to respond, Alice needs to establish her own private and public keys.
- Bob needs only one private key to receive all correspondence from anyone in the community, but Alice needs n public keys to communicate with n entities in the community, one public key for each entity.
- Alice needs a **ring of public key**

Plaintext / Ciphertext

- Plaintext and ciphertext are treated as integers in asymmetric-key cryptography.

- The message must be encoded as an integer (or a set of integers) before encryption; the integer (or the set of integers) must be decoded into the message after decryption.

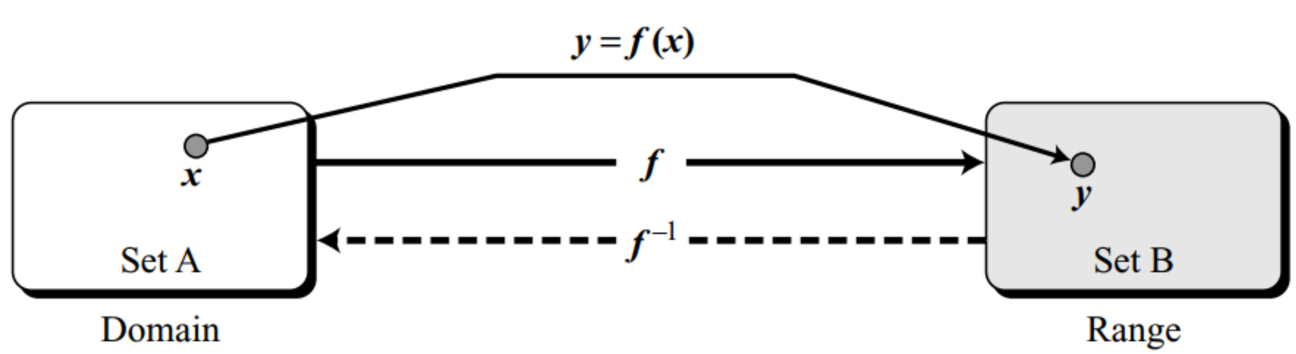
- Asymmetric-key cryptography is normally used to **encrypt or decrypt small pieces of information,** such as the cipher key for a symmetrickey cryptography.

- Normally is used for ancillary goals instead of message encipherment.

Function

- A **function** is a rule that associates (maps) one element in set A, called the **domain**, to one element in set B, called the **range**

- An **invertible function** is a function that associates **each element** in the **range** with **exactly one element** in the **domain.**



One-Way Function

- A one-way function (OWF) is a function **f** that satisfies the following two properties:
  - **f** is easy to compute(i.e) given x, **y = f (x)** can be easily computed.
  - f^{-1} is **difficult to compute** (i.e) given y, it is computationally infeasible to calculate **x =** $\mathbf{f^{-1}(y)}$**.**

One-Way Function

- **Eg:**
  - When n is large, **$n = p \times q$ is a one-way function.**
  - **Function x - tuple (p, q) of two primes** and **y is n**.
  - Given p and q, it is always easy to calculate n; given n, it is very difficult to compute p and q. This is the factorization problem.
  - There is not a polynomial time solution to the f^{-1} function.

Trapdoor One-Way Function

- The main idea behind asymmetric-key cryptography is the concept of the **trapdoor oneway function**

- A trapdoor one-way function (TOWF) is a **one-way function** with a third property:

  - **Given y and a trapdoor (secret), x can be computed easily.**

  - With other two properties:

  - **f** is easy to compute(i.e) given x, **$y = f(x)$** can be easily computed.

  - f^{-1} is **difficult to compute** (i.e) given y, it is computationally infeasible to calculate **$x = f^{-1}(y)$.**

Trapdoor One-Way Function

- **Eg:**
  - When n is large, the function $\mathbf{y = x^k \bmod n}$ is a **trapdoor one-way function**.
  - Given x, k, and n, it is easy to calculate y using the fast exponential algorithm
  - Given y, k, and n, it is very difficult to calculate x. This is called the **discrete logarithm problem .**
  - There is not a polynomial time solution to the f^{-1} function.
  - However, if we know the **trapdoor, k′** such that $\mathbf{k \times k' = 1 \bmod \phi(n)}$, we can use $\mathbf{x = y^{k'} \bmod n}$ to find x.

Knapsack Cryptosystem

- First idea of public-key cryptography from Merkle and Hellman, in knapsack cryptosystem.

- System was found to be insecure with today's standards, but they formed the precursor to recent public-key cryptosystems.

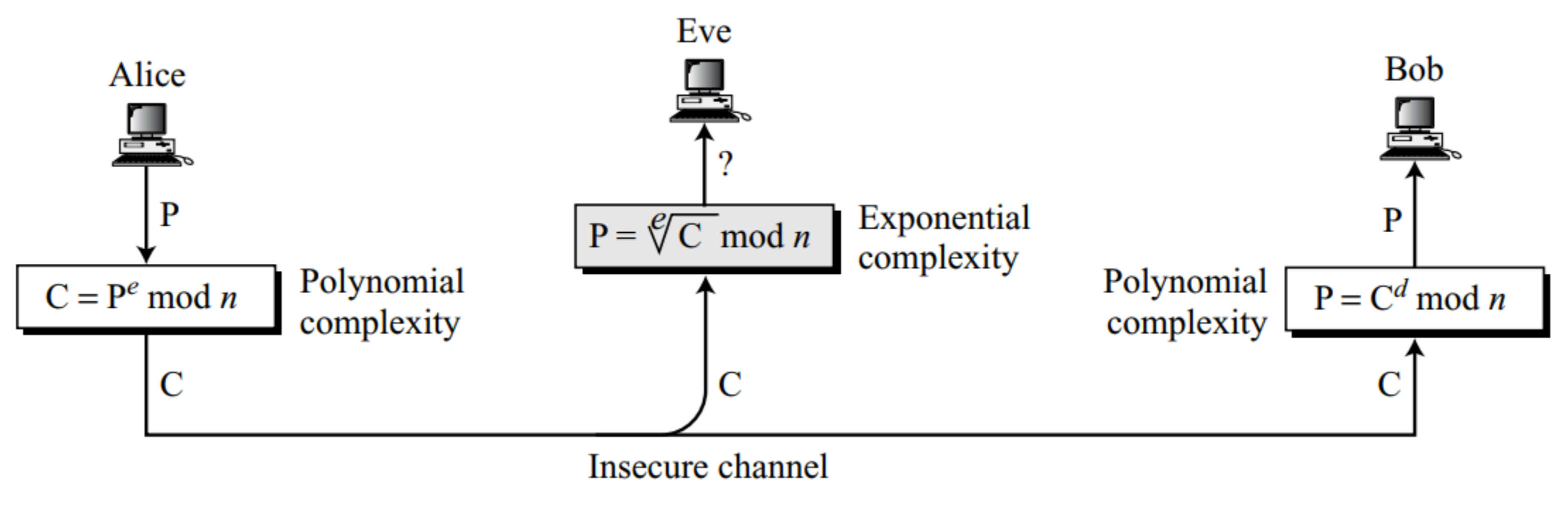
- If we are told which **elements**, from a predefined set of numbers, are in a **knapsack**, we can easily calculate the **sum of the numbers;**

- if we are told the sum, it is difficult to say which elements are in the knapsack.

RSA CRYPTOSYSTEM

- The most common public-key algorithm is the RSA cryptosystem, named for its inventors (Rivest, Shamir, and Adleman).
- RSA uses two exponents, **e and d,** where **e is public** and **d is private.**
- P is the plaintext and C is the ciphertext.
- $\mathbf{C = P^e \bmod n}$
- $\mathbf{P = C^d \bmod n}$
- The modulus n, a very large number, is created during the key generation process

RSA CRYPTOSYSTEM



- **Alice uses a one-way function (modular exponentiation) with a trapdoor known only to Bob.**
- Eve, who does not know the trapdoor, cannot decrypt the message.
- If a polynomial algorithm for eth root modulo n calculation is found, **modular exponentiation will not be a one-way function any more**.

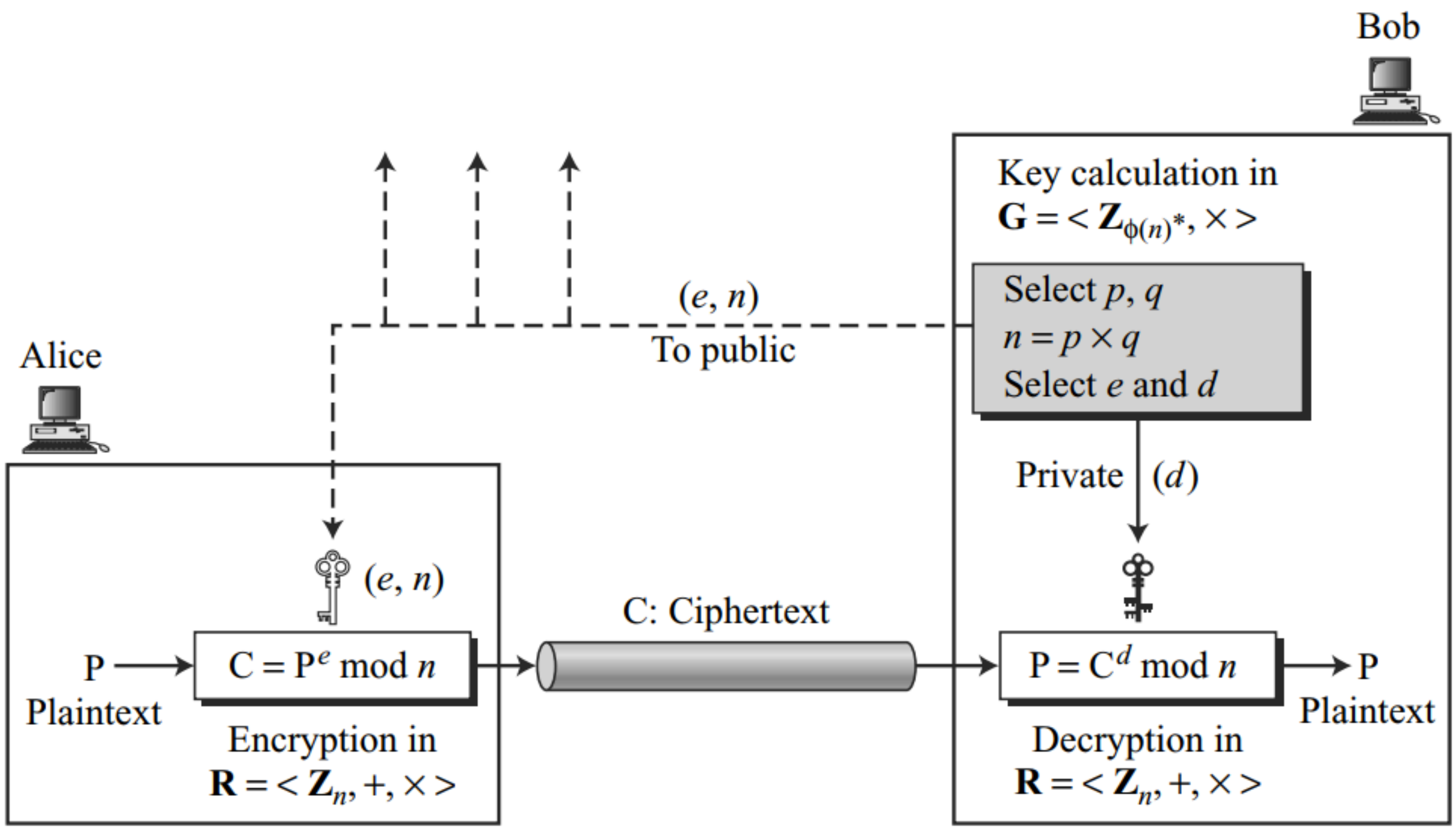
RSA CRYPTOSYSTEM

- Encryption and decryption use modular exponentiation.
- **Modular exponentiation** is feasible in **polynomial time** using the fast exponentiation algorithm.
- **Modular logarithm i**s as hard as factoring the modulus, for which there is no polynomial algorithm yet.
- Alice can encrypt in **polynomial time** (e is public), Bob also can decrypt in **polynomial time** (because he knows d),
- Eve cannot decrypt because she would have to calculate the **eth root of C** using modular arithmetic.

RSA CRYPTOSYSTEM

- **Alice uses a one-way function (modular exponentiation) with a trapdoor known only to Bob.**
- Eve, who does not know the trapdoor, cannot decrypt the message.
- If a polynomial algorithm for eth root modulo n calculation is found, **modular exponentiation will not be a one-way function any more**.

Encryption, Decryption, and Key Generation in RSA



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

}

RSA Key Generation

- After key generation, Bob announces the tuple (e, n) as his public key;
- Bob keeps the integer d as his private key.
- Bob can discard p, q, and ϕ(n); they will not be needed unless Bob needs to change his private key without changing the modulus (which is not recommended).
- To be secure, the recommended size for each prime, p or q, is 512 bits (almost 154 decimal digits).
- This makes the size of n, the modulus, **1024 bits** (309 digits)

RSA Proof

- Using second version of Euler's theorem:

Euler's Theorem

- **First Version**
- The first version of Euler's theorem is similar to the first version of the Fermat's little theorem.
- If a and n are coprime, then $\mathbf{a^{\varphi(n)} \equiv 1 \ (mod\ n)}$
- **Second Version**
- The second version of Euler's theorem is similar to the second version of Fermat's little theorem;
- It removes the condition that a and n should be coprime.
- If n = p × q, a < n, and k an integer, then
- $\mathbf{a^{k \times \varphi(n) + 1} \equiv a \ (mod\ n)}$

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

RSA Proof

- If the plaintext retrieved by Bob is P1 and prove that it is equal to P

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

$ed = k\phi(n) + 1$ // 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

- Given p = 7, q=11. Calculate the keys and encrypt Plain text 5 with the keys generated

Examples

1. Calculate $n = 7 \times 11 = 77$.

2. Calculate $\phi(n) = (7 - 1)(11 - 1) = 60$.

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

4. If $e = 13$

5. Calculate $d = e^{-1} \bmod \phi(n)$. $d = 13^{-1} \bmod 60 = 37$

6. $e \times d \bmod 60 = 1$ (they are inverses of each other).

7. Encrypt PT = 5. $C = (P^e \bmod n) = 5^{13} \bmod 77 = 26 \bmod 77$. CT = 26

8. Decrypt CT = 26. $P = (C^d \bmod n) = 26^{37} \bmod 77 = 5 \bmod 77$.

Inverse 13 , 60

q	r1	r2	r= r1 − q × r2	t1	t2	t = t1 − q × t2
4	60	13	8	0	1	-4
1	13	8	5	1	-4	5
1	8	5	3	-4	5	-9
1	5	3	2	5	-9	14
1	3	2	1	-9	14	-23
2	2	1	0	14	-23	

Inv of 13 mod 60 = -23 mod 60 = 37 mod 60

Examples

- Given p = 7, q =17. Find n, e, d. PT = 10
- n = p*q = 7 * 17 = 119
- $\phi(n) = (p-1) * (q-1) = 6 * 16 = 96$
- e = 19 , d = 91 .
- CT = 10^19 mod 119 = 31 mod 119
- PT = 31^91 mod 119 = 10 mod 119
- e = 5, d = 77
- CT = 10 ^ 5 mod 119 = 40 mod 119
- PT = 40^77 mod 119 = 10 mod 119

Inverse 19 , 96

q	r1	r2	r= r1 − q × r2	t1	t2	t = t1 − q × t2
5	96	19	1	0	1	-5
19	19	1	0	1	-5	-96

Inv of -5 mod 96 = 91 mod 96

Inverse 5 , 96

q	r1	r2	r= r1 − q × r2	t1	t2	t = t1 − q × t2
19	96	5	1	0	1	-19
5	5	1	0	1	-19	96

Inv of 5 mod 96 = -19 mod 96 = 77 mod 96

Examples

- Given $p = 7$, $q = 17$. Find n, e, d.

Examples

- Given $p = 17$, $q = 11$. Find n, e, d.