The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption. The first problem is that of key distribution, which is examined in some detail in Chapter 14.

As Chapter 14 discusses, key distribution under symmetric encryption requires either (1) that two communicants already share a key, which somehow has been distributed to them; or (2) the use of a key distribution center. Whitfield Diffie, one of the discoverers of public-key encryption (along with Martin Hellman, both at Stanford University at the time), reasoned that this second requirement negated the very essence of cryptography: the ability to maintain total secrecy over your own communication. As Diffie put it [DIFF88], "what good would it do after all to develop impenetrable cryptosystems, if their users were forced to share their keys with a KDC that could be compromised by either burglary or subpoena?"

The second problem that Diffie pondered, and one that was apparently unrelated to the first, was that of *digital signatures*. If the use of cryptography was to become widespread, not just in military situations but for commercial and private purposes, then electronic messages and documents would need the equivalent of signatures used in paper documents. That is, could a method be devised that would stipulate, to the satisfaction of all parties, that a digital message had been sent by a particular person? This is a somewhat broader requirement than that of authentication, and its characteristics and ramifications are explored in Chapter 13.

Diffie and Hellman achieved an astounding breakthrough in 1976 [DIFF76 a, b] by coming up with a method that addressed both problems and was radically different from all previous approaches to cryptography, going back over four millennia.[1]

In the next subsection, we look at the overall framework for public-key cryptography. Then we examine the requirements for the encryption/decryption algorithm that is at the heart of the scheme.

### Public-Key Cryptosystems

Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic:

- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

---

[1]Diffie and Hellman first *publicly* introduced the concepts of public-key cryptography in 1976. Hellman credits Merkle with independently discovering the concept at that same time, although Merkle did not publish until 1978 [MERK78a]. In fact, the first unclassified document describing public-key distribution and public-key cryptography was a 1974 project proposal by Merkle (http://merkle.com/1974). However, this is not the true beginning. Admiral Bobby Inman, while director of the National Security Agency (NSA), claimed that public-key cryptography had been discovered at NSA in the mid-1960s [SIMM93]. The first *documented* introduction of these concepts came in 1970, from the Communications-Electronics Security Group, Britain's counterpart to NSA, in a classified report by James Ellis [ELLI70]. Ellis referred to the technique as *nonsecret encryption* and describes the discovery in [ELLI99].

In addition, some algorithms, such as RSA, also exhibit the following characteristic.

- Either of the two related keys can be used for encryption, with the other used for decryption.

A **public-key encryption** scheme has six ingredients (Figure 9.1a; compare with Figure 2.1).

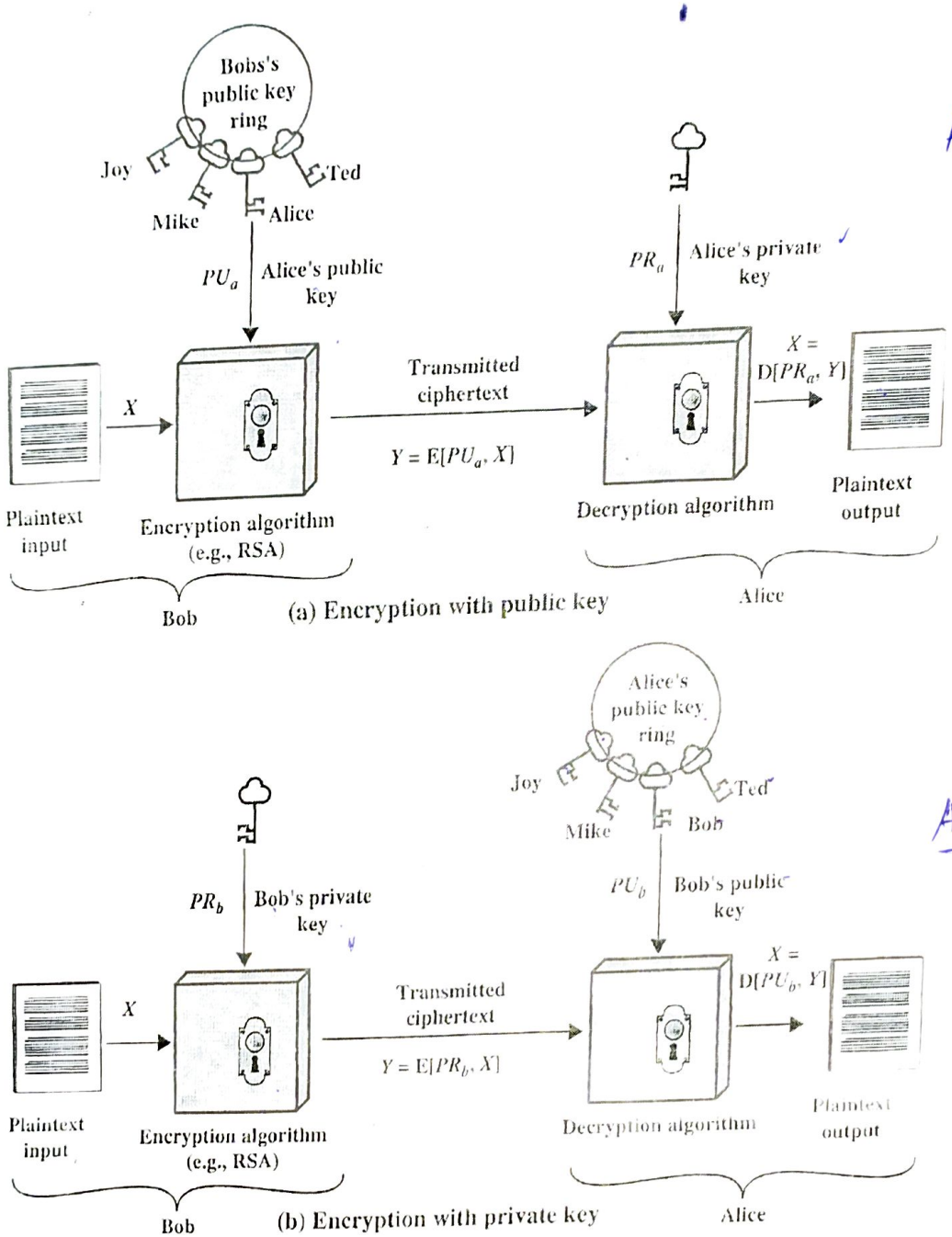



**Figure 9.1** Public-Key Cryptography

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

The essential steps are the following.

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As Figure 9.1a suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user's private key remains protected and secret, incoming communication is secure. At any time, a system can change its private key and publish the companion public key to replace its old public key.

Table 9.2 summarizes some of the important aspects of symmetric and public-key encryption. To discriminate between the two, we refer to the key used in symmetric encryption as a **secret key**. The two keys used for asymmetric encryption are referred to as the **public key** and the **private key**.[2] Invariably, the private key is kept secret, but it is referred to as a private key rather than a secret key to avoid confusion with symmetric encryption.

Let us take a closer look at the essential elements of a public-key encryption scheme, using Figure 9.2 (compare with Figure 2.2). There is some source A that

---

[2]The following notation is used consistently throughout. A secret key is represented by $K_m$, where $m$ is some modifier; for example, $K_a$ is a secret key owned by user A. A public key is represented by $PU_a$, for user A, and the corresponding private key is $PR_a$. Encryption of plaintext $X$ can be performed with a secret key, a public key, or a private key, denoted by $E(K_a, X)$, $E(PU_a, X)$, and $E(PR_a, X)$, respectively. Similarly, decryption of ciphertext $C$ can be performed with a secret key, a public key, or a private key, denoted by $D(K_a, X)$, $D(PU_a, X)$, and $D(PR_a, X)$, respectively.

**Table 9.2** Conventional and Public-Key Encryption

| Conventional Encryption | Public-Key Encryption |
|---|---|
| *Needed to Work:* | *Needed to Work:* |
| 1. The same algorithm with the same key is used for encryption and decryption. | 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. |
| 2. The sender and receiver must share the algorithm and the key. | 2. The sender and receiver must each have one of the matched pair of keys (not the same one). |
| *Needed for Security:* | *Needed for Security:* |
| 1. The key must be kept secret. | 1. One of the two keys must be kept secret. |
| 2. It must be impossible or at least impractical to decipher a message if no other information is available. | 2. It must be impossible or at least impractical to decipher a message if no other information is available. |
| 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key. | 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key. |

produces a message in plaintext, $X = [X_1, X_2, \ldots, X_M]$. The $M$ elements of $X$ are letters in some finite alphabet. The message is intended for destination B. B generates a related pair of keys: a public key, $PU_b$, and a private key, $PR_b$. $PR_b$ is known only to B, whereas $PU_b$ is publicly available and therefore accessible by A.

With the message $X$ and the encryption key $PU_b$ as input, A forms the ciphertext $Y = [Y_1, Y_2, \ldots, Y_N]$:

$$Y = \mathrm{E}(PU_b, X)$$

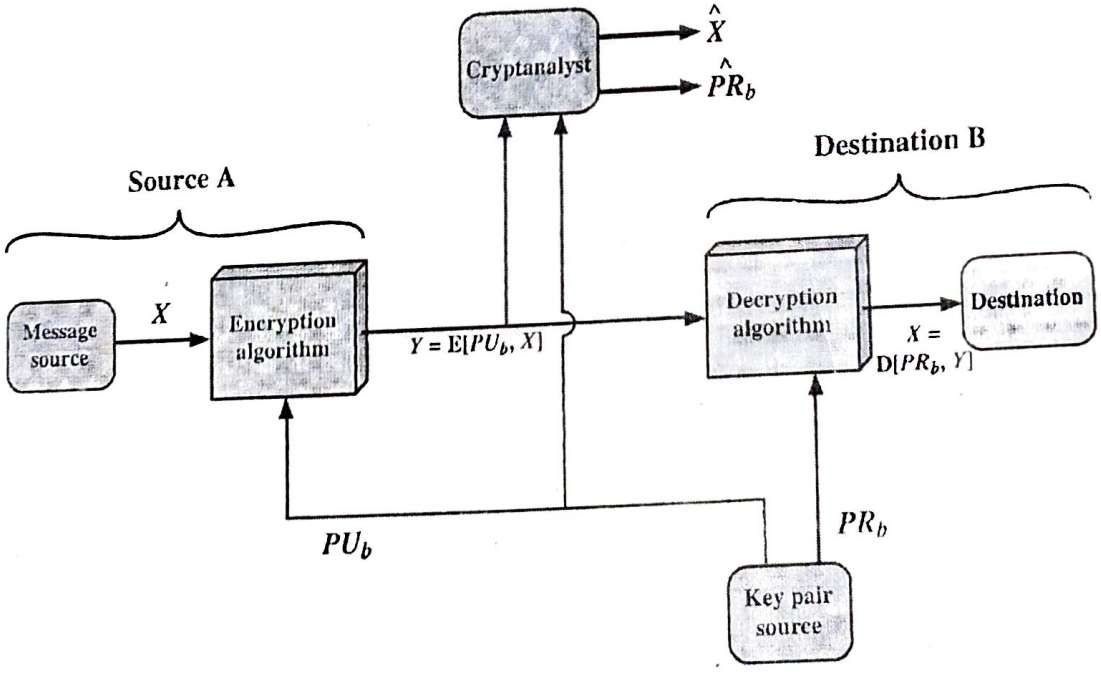



**Figure 9.2** Public-Key Cryptosystem: Secrecy

The intended receiver, in possession of the matching private key, is able to invert the transformation:

$$X = \mathrm{D}(PR_b, Y)$$

An adversary, observing $Y$ and having access to $PU_b$, but not having access to $PR_b$ or $X$, must attempt to recover $X$ and/or $PR_b$. It is assumed that the adversary does have knowledge of the encryption (E) and decryption (D) algorithms. If the adversary is interested only in this particular message, then the focus of effort is to recover $X$ by generating a plaintext estimate $\hat{X}$. Often, however, the adversary is interested in being able to read future messages as well, in which case an attempt is made to recover $PR_b$ by generating an estimate $\hat{PR}_b$.

We mentioned earlier that either of the two related keys can be used for encryption, with the other being used for decryption. This enables a rather different cryptographic scheme to be implemented. Whereas the scheme illustrated in Figure 9.2 provides confidentiality, Figures 9.1b and 9.3 show the use of public-key encryption to provide authentication:

$$Y = \mathrm{E}(PR_a, X)$$

$$X = \mathrm{D}(PU_a, Y)$$

In this case, A prepares a message to B and encrypts it using A's private key before transmitting it. B can decrypt the message using A's public key. Because the message was encrypted using A's private key, only A could have prepared the message. Therefore, the entire encrypted message serves as a **digital signature**. In addition, it is impossible to alter the message without access to A's private key, so the message is authenticated both in terms of source and in terms of data integrity.

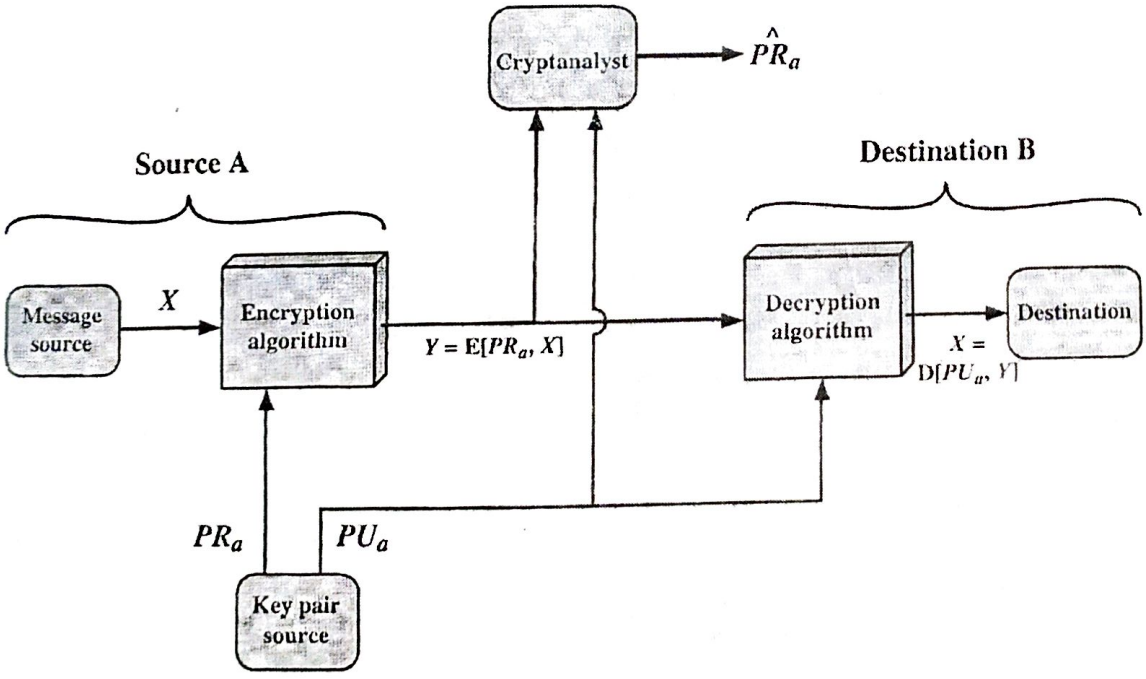



Figure 9.3 Public-Key Cryptosystem: Authentication

In the preceding scheme, the entire message is encrypted, which, although validating both author and contents, requires a great deal of storage. Each document must be kept in plaintext to be used for practical purposes. A copy also must be stored in ciphertext so that the origin and contents can be verified in case of a dispute. A more efficient way of achieving the same results is to encrypt a small block of bits that is a function of the document. Such a block, called an authenticator, must have the property that it is infeasible to change the document without changing the authenticator. If the authenticator is encrypted with the sender's private key, it serves as a signature that verifies origin, content, and sequencing. Chapter 13 examines this technique in detail.

It is important to emphasize that the encryption process depicted in Figures 9.1b and 9.3 does not provide confidentiality. That is, the message being sent is safe from alteration but not from eavesdropping. This is obvious in the case of a signature based on a portion of the message, because the rest of the message is transmitted in the clear. Even in the case of complete encryption, as shown in Figure 9.3, there is no protection of confidentiality because any observer can decrypt the message by using the sender's public key.

It is, however, possible to provide both the authentication function and confidentiality by a double use of the public-key scheme (Figure 9.4):

$$Z = E(PU_b, E(PR_a, X))$$

$$X = D(PU_a, D(PR_b, Z))$$

In this case, we begin as before by encrypting a message, using the sender's private key. This provides the digital signature. Next, we encrypt again, using the receiver's public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus, confidentiality is provided. The

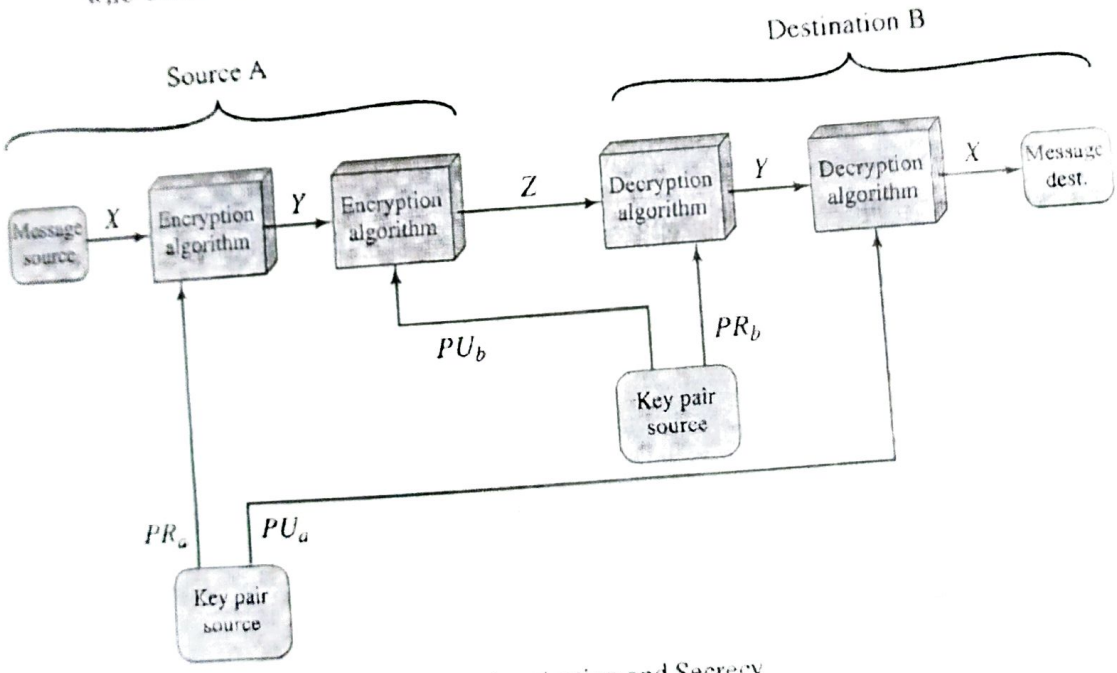



**Figure 9.4** Public-Key Cryptosystem: Authentication and Secrecy

disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

### Applications for Public-Key Cryptosystems

Before proceeding, we need to clarify one aspect of public-key cryptosystems that is otherwise likely to lead to confusion. Public-key systems are characterized by the use of a cryptographic algorithm with two keys, one held private and one available publicly. Depending on the application, the sender uses either the sender's private key or the receiver's public key, or both, to perform some type of cryptographic function. In broad terms, we can classify the use of **public-key cryptosystems** into three categories:

- **Encryption/decryption:** The sender encrypts a message with the recipient's public key.
- **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
- **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

Some algorithms are suitable for all three applications, whereas others can be used only for one or two of these applications. Table 9.3 indicates the applications supported by the algorithms discussed in this book.

### Requirements for Public-Key Cryptography

The cryptosystem illustrated in Figures 9.2 through 9.4 depends on a cryptographic algorithm based on two related keys. Diffie and Hellman postulated this system without demonstrating that such algorithms exist. However, they did lay out the conditions that such algorithms must fulfill [DIFF76b].

1. It is computationally easy for a party B to generate a pair (public key $PU_b$, private key $PR_b$).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, $M$, to generate the corresponding ciphertext:

$$C = \mathrm{E}(PU_b, M)$$

**Table 9.3** Applications for Public-Key Cryptosystems

| Algorithm | Encryption/Decryption | Digital Signature | Key Exchange |
|---|---|---|---|
| RSA | Yes | Yes | Yes |
| Elliptic Curve | Yes | Yes | Yes |
| Diffie-Hellman | No | No | Yes |
| DSS | No | Yes | No |

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

$$M = \mathrm{D}(PR_b, C) = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

4. It is computationally infeasible for an adversary, knowing the public key, $PU_b$, to determine the private key, $PR_b$.
5. It is computationally infeasible for an adversary, knowing the public key, $PU_b$, and a ciphertext, $C$, to recover the original message, $M$.

We can add a sixth requirement that, although useful, is not necessary for all public-key applications:

6. The two keys can be applied in either order:

$$M = \mathrm{D}[PU_b, \mathrm{E}(PR_b, M)] = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

These are formidable requirements, as evidenced by the fact that only a few algorithms (RSA, elliptic curve cryptography, Diffie-Hellman, DSS) have received widespread acceptance in the several decades since the concept of public-key cryptography was proposed.

Before elaborating on why the requirements are so formidable, let us first recast them. The requirements boil down to the need for a trap-door one-way function. A **one-way function**[3] is one that maps a domain into a range such that every function value has a unique inverse, with the condition that the calculation of the function is easy, whereas the calculation of the inverse is infeasible:

$$Y = \mathrm{f}(X) \quad \text{easy}$$

$$X = \mathrm{f}^{-1}(Y) \quad \text{infeasible}$$

Generally, *easy* is defined to mean a problem that can be solved in polynomial time as a function of input length. Thus, if the length of the input is $n$ bits, then the time to compute the function is proportional to $n^a$, where $a$ is a fixed constant. Such algorithms are said to belong to the class **P**. The term *infeasible* is a much fuzzier concept. In general, we can say a problem is infeasible if the effort to solve it grows faster than polynomial time as a function of input size. For example, if the length of the input is $n$ bits and the time to compute the function is proportional to $2^n$, the problem is considered infeasible. Unfortunately, it is difficult to determine if a particular algorithm exhibits this complexity. Furthermore, traditional notions of computational complexity focus on the worst-case or average-case complexity of an algorithm. These measures are inadequate for cryptography, which requires that it be infeasible to invert a function for virtually all inputs, not for the worst case or even average case. A brief introduction to some of these concepts is provided in Appendix 9B.

[3]Not to be confused with a one-way hash function, which takes an arbitrarily large data field as its argument and maps it to a fixed output. Such functions are used for authentication (see Chapter 11).

We now turn to the definition of a **trap-door one-way function**, which is easy to calculate in one direction and infeasible to calculate in the other direction unless certain additional information is known. With the additional information the inverse can be calculated in polynomial time. We can summarize as follows. A trap-door one-way function is a family of invertible functions $f_k$, such that

$$Y = f_k(X) \quad \text{easy, if } k \text{ and } X \text{ are known}$$
$$X = f_k^{-1}(Y) \quad \text{easy, if } k \text{ and } Y \text{ are known}$$
$$X = f_k^{-1}(Y) \quad \text{infeasible, if } Y \text{ is known but } k \text{ is not known}$$

Thus, the development of a practical public-key scheme depends on discovery of a suitable trap-door one-way function.

### Public-Key Cryptanalysis

As with symmetric encryption, a public-key encryption scheme is vulnerable to a brute-force attack. The countermeasure is the same: Use large keys. However, there is a tradeoff to be considered. Public-key systems depend on the use of some sort of invertible mathematical function. The complexity of calculating these functions may not scale linearly with the number of bits in the key but grow more rapidly than that. Thus, the key size must be large enough to make brute-force attack impractical but small enough for practical encryption and decryption. In practice, the key sizes that have been proposed do make brute-force attack impractical but result in encryption/decryption speeds that are too slow for general-purpose use. Instead, as was mentioned earlier, public-key encryption is currently confined to key management and signature applications.

Another form of attack is to find some way to compute the private key given the public key. To date, it has not been mathematically proven that this form of attack is infeasible for a particular public-key algorithm. Thus, any given algorithm, including the widely used RSA algorithm, is suspect. The history of cryptanalysis shows that a problem that seems insoluble from one perspective can be found to have a solution if looked at in an entirely different way.

Finally, there is a form of attack that is peculiar to public-key systems. This is, in essence, a probable-message attack. Suppose, for example, that a message were to be sent that consisted solely of a 56-bit DES key. An adversary could encrypt all possible 56-bit DES keys using the public key and could discover the encrypted key by matching the transmitted ciphertext. Thus, no matter how large the key size of the public-key scheme, the attack is reduced to a brute-force attack on a 56-bit key. This attack can be thwarted by appending some random bits to such simple messages.

## 9.2 THE RSA ALGORITHM

The pioneering paper by Diffie and Hellman [DIFF76b] introduced a new approach to cryptography and, in effect, challenged cryptologists to come up with a cryptographic algorithm that met the requirements for public-key systems. A number of

A No. of Concepts from Number Theory are essential in the design of Public-key cryptographic algorithms.

① Prime No's :-

A Central Concern of No. Theory is the study of Prime No's. An integer $P > 1$ is a prime no. if & only if its only divisors are $\pm 1$.

Any integer $a > 1$ can be factored in Unique way as

$$a = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_t^{a_t}$$

where $p_1 < p_2 < \cdots p_t$ are prime no's & where each $a_i$ is a positive integer.

Eg :-

$$91 = 7 \times 13$$
$$3600 = 2^4 \times 3^2 \times 5^2$$

} Prime factorization

② Relatively Prime no's :- No Common Divisor apart from 1.

Eg :- 8 & 15

$8 - 1, 2, 4, 8$
$15 - 1, 3, 5, 15$ } 1 Common factor

$\text{Gcd}(8, 15) = 1$

General Formula $\boxed{a = \prod_{p \in P} p^{a_p}}$ where each $a_p \geq 0$.

GCD :- Prime factorization [comparisons & least Powers]

Eg :- $300 = 2^1 \times 3^2 \times 5^2$
$18 = 2^1 \times 3^2 \times 5^0$ $\Longrightarrow$ ?

$\text{GCD}(300, 18) = 2^1 \times 3^2 \times 5^0$
$= 18.$

If

- p is prime &
- 'a' is positive integer not divisible by p
  - i.e ( gcd (a, p) = 1 )

then

$$a^{p-1} \equiv 1 \pmod{p} \text{ or } \frac{a^p}{a} \equiv 1 \pmod{p}$$

or ~~$a^{p-1} \pmod{p} = 1$~~

$$\boxed{a^p = a \pmod{p}} = a.$$

Eg :-

(i) $13^{16} \bmod 17 = 1$ ✓

( 13 not div by 17 & 17 is Prime)

$a = 13$
$p = 17$.

~~(ii) $13^{15} \bmod 15$~~

$17-1 \; ! = 15$

So, we cannot use Fermat's Thm.

iii) $5^{18} \bmod 19$ if we don't know this Thm very Diff. to Solve.

then $p = 19$
$a = 5$ i.e gcd (5, 19) = 1

So $a^{p-1} \bmod P$

$a^{p-1} \equiv 1 \bmod P$

$5^{19-1} \bmod 19 = 1$ ✓

(iii) $5^{19} \bmod 19 = 5$ ✓

(iv) $5^{20} \bmod 19$.

$5^{19 \times 1 + 1} \bmod 19$

$5^{19} \cdot 5^1 \bmod 19 \Rightarrow 5^{19} \bmod 19 \cdot 5 \bmod 19$

$\Rightarrow 5 \cdot 5 \Rightarrow 25$

## EULER'S Thm :-

. It states that for every 'a' & 'n' that are relatively Prime :

$$a^{\phi(n)} = 1 \pmod{n}$$

where $\phi(n)$ is Euler's Totient fx. → very much useful in RSA Algo

↓

It is the count of no's which are relatively prime to n there less than n.

Eg :- $\phi(10) = 4$.

Factors :- 2, 5

1 ✗ 3 4 5 6 7 8 9 10

= Ans is 4

Case 1 :- if n is Prime

$$\phi(n) = n-1$$

Eg :- $\phi(7) = 6$

Nos ~~are~~ are 1, 2, 3, 4, 5, 6 relatively Prime with 7

Count is 6.

Case 2 :- if 'n' is a product of 2 Primes p & q

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

eg :- $\phi(21) = \phi(7) \times \phi(3)$
$= (7-1) \times (3-1) = 6 \times 2 = 12$

Count is 12.

Nos relatively Prime with 21 are 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20

Case 3:- if $n = p^e$ where $p$ is a

$$\phi(n) = p^e - p^{e-1}$$

Eg:- $\phi(8) = \phi(2^3)$

$= 2^3 - 2^2 = 8 - 4 = 4$ ✓

(No's relatively Prime with 8 are 1, 3, 5, 7

Count is 4)

# RSA Algorithm

- Most Popular Public key Algo
- Named After Rivest, Shamir, Adleman
- Encryp & Decryp. based on No. Theory
- Variable key Length; Long for Enchanced Security & Short for Efficiency (512 bits)

In Sym – Same key in both Sides.

Asymm – Two keys.
- Public
- Private

Algo:- [Key Generation Process] — for More Security

- Consider Two large Prime no's $p, q$.
- Calculate $n = p * q$

Euler's Totient Fx: $\phi(n) = (p-1) * (q-1)$

- Assume $e$ (Public key used at Encryp Side)

$e$ such that $\gcd(e, \phi(n)) = 1$

- Assume $d$ (Private key used at Decryp. Side)

$d$ such that $d \equiv e^{-1} \bmod \phi(n)$ — congruent Modulo.

$\Downarrow$

$d \times e \equiv 1 \bmod \phi(n)$ — Equals 1

$\Downarrow$

$d \times e \bmod \phi(n) = 1 \bmod \phi(n)$

$d \times e \bmod \phi(n) = 1$

- Public Key = {e, n}
- Private Key = {d, n}

One key i.e e is used in Encryption Side and other key i.e d is used in Decryp. Side

**Encryption :-**

Msg → M < n.

$C = M^e \bmod n$

Ciphertext

**Decryption :-**

$M = C^d \bmod n$

PlainText

For Simple Calculations :- Consider Simple Small No's } But for Security Take Large No's.

- $p = 3, q = 5$
- $n = p * q = 15$

$\phi(n) = (p-1) * (q-1)$

$= 2 * 4$

$= 8$

$\boxed{\phi(n) = 8}$

- e

Assume for 3,5,7 Condition is satisfied

$\gcd(e, \phi(n)) = 1$

$\gcd(3, 8) = 1$

- $d * e \bmod \phi(n) = 1$

Assume (d = 3)

$d * 3 \bmod 8 = 1$

$3 * 3 \bmod 8 = 1$

$9 \bmod 8 = 1$

NOTE :- e satisfies e > 1 Less than $\phi(n)$

(iii) $\gcd(e, \phi(n)) = 1$

$\gcd(2, 12) = 1$ ✗ Wrong

↳ No Common

Public key $\{e, n\} = \{3, 15\}$

Private key $\{d, n\} = \{3, 15\}$

ENCRYP :-

Consider $M = 4 < n$

$C = M^e \bmod n$

$= 4^3 \bmod 15$

$= 64 \bmod 15 = 4$ → Remainder

$\boxed{C = 4}$

DECRYP :-

$M = C^d \bmod n$

$= 4^3 \bmod 15$

$= 64 \bmod 15$

$\boxed{M = 4}$

↑ Plain Text.

- Nounces in Authentication Protocols to Prevent Replay.
- Session Keys. Eg: during Transaction, we use One Time Pad
- Public Key generation
- Key stream for a one-Time Pad

Used in Cryptography.

Random Numbers - They play an important role in the use of encryption for various network security applications. A no. of n/w security algorithms and protocols based on cryptography make use of random binary nos.

- Key distribution and reciprocal authentication schemes
- Session Key generation
- Generation of keys for the RSA public-key encryption algo.
- Generation of a bit stream for symmetric stream encryption

These applications give rise to two distinct & not necessarily compatible requirements for a sequence of random numbers: randomness and unpredictability

1) Randomness :- The foll two criteria are used to validate that a sequence of nos is random.
   a) Uniform distribution :- freq of occurrence of 1's & 0's should be equal
   b) Independence :- No one subsequence in the sequence can be inferred from the others.

2) Unpredictability :- With true random sequences, each number is statistically independent of other nos in the sequence and therefore unpredictable. True random numbers are seldom used.

Cryptographic applications make use of algorithmic techniques for random number generation. These algos are deterministic and therefore produce sequences of nos that are not statistically random. However, if the algo is good, the resulting sequences will pass many reasonable tests

of randomness. Such numbers are referred to as pseudorandom numbers.

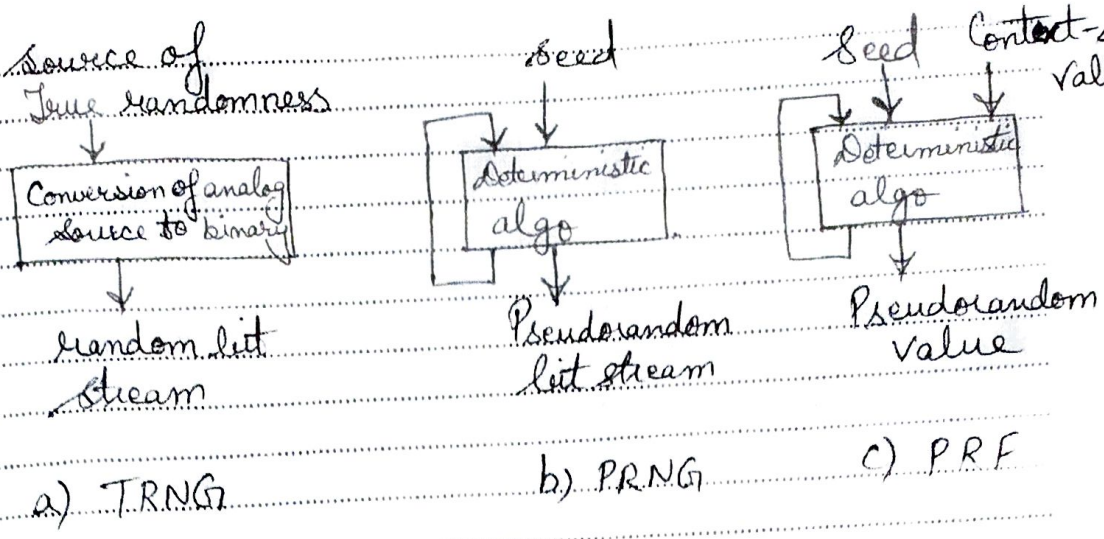
True Random Number Generator (TRNG) :- TRNG with two forms of pseudorandom number generators. A TRNG takes as input a source that is effectively random; the source is often referred to as an entropy source. The entropy source is drawn from the physical environment of the computer and could include things such as keystroke timing patterns, disk electrical activity, mouse movements and instantaneous values of the system clock. The TRNG may simply involve conversion of an analog source to a binary output. It may involve additional processing to overcome any bias in the source.

A PRNG takes as input a fixed value called the seed and produces a sequence of output bits using a deterministic algo.

Two different forms of PRNGs based on application

1) Pseudorandom number generator - An algo that is used to produce an open-ended sequence of bits is referred to as a PRNG.

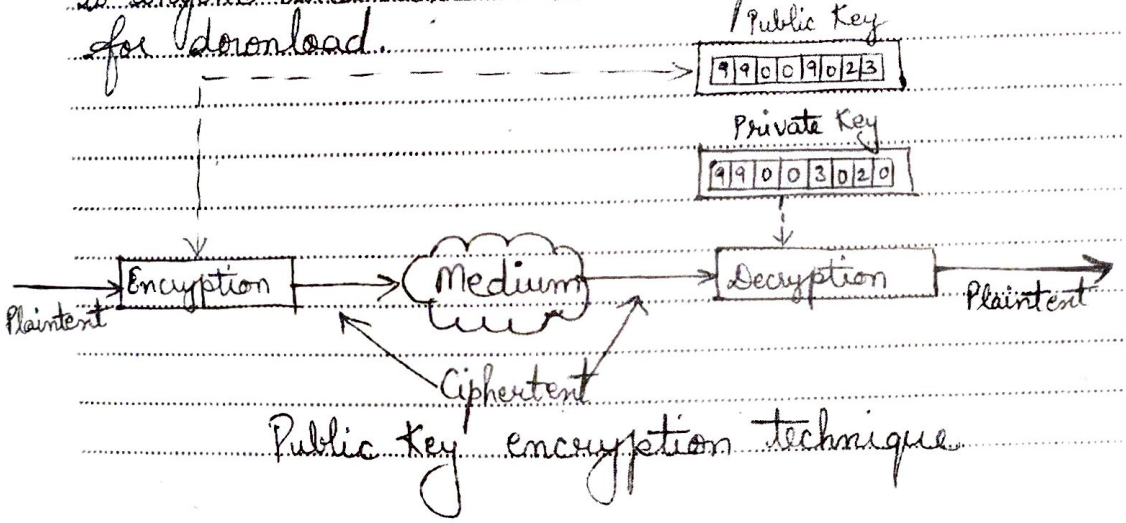
2) Pseudorandom function (PRF) :- It is used to produce a pseudo-random string of bits of some fixed length. Eg: are symmetric encryption keys and nonces.




a) TRNG b) PRNG c) PRF

Asymmetric Encryption :- It uses two different but related keys and either key can be used to encrypt or decrypt the message. If, however, key A is used to encrypt the message, only key B can decrypt it, and if key B is used to encrypt a message, only key A can decrypt it. It can be used to provide elegant solutions to problems of secrecy and verification. This technique has its highest value when one key is used as a private key which means that it is kept secret, known only to the owner of the key pair, and the other key serves as a public key, which means that it is stored in a public location where anyone can use it. That's why it is known as public - key encryption.

It is based on a hash value which is a summary of the original input values. The key used to encrypt a message is public key and to decrypt a message, the key is private. Many commonly used algorithms, such as PGP, use public key encryption. It is very easy to implement since the public key can be freely distributed to anyone and sometimes even put on a website for download.




Public key encryption technique

In public key cryptography, the public key is announced to the public, whereas the private key is kept by the receiver. The sender uses the public key of the receiver for encryption and the receiver uses his private key for decryption.

Advantages :- ① The pair of keys can be used with any other entity
② The no. of keys required is small

Disadvantages :- ① It is not efficient for long messages
② Association b/w an entity and its public key must be verified