Information Security Fall 2022

Week # 5

Lecture # 13, 14 and 15

Dr. Aqsa Aslam

PART ONE: Computer Security Technology and Principles

CHAPTER 2

CRYPTOGRAPHIC TOOLS

Topics from text book:

- **Chapter # 2**
- **Chapter # 20**
- **Chapter # 21**

Public-Key Encryption

- Public-key algorithms are based on **mathematical functions**
  - Rather than on simple operations on bit patterns, such as are used in symmetric encryption algorithms.
- Public-key cryptography is asymmetric, involving the use of **two separate keys**
  - in contrast to symmetric encryption, which uses only one key
- The use of two keys has profound consequences in the areas of:
  - **Confidentiality**
  - **Key distribution**
  - **Authentication**

Public-Key Encryption Structure

- **Misconceptions concerning public-key encryption.**

1. Public-key encryption is more secure from cryptanalysis than symmetric encryption. In fact, the security of any encryption scheme depends on
   - *The length of the key*
   - *The computational work involved in breaking a cipher.*
   - There is nothing in principle about either symmetric or public-key encryption that makes one superior to another from the point of view of resisting cryptanalysis.

2. A second misconception is that public-key encryption is a general-purpose technique that has made symmetric encryption obsolete. On the contrary, because of the computational overhead of current public-key encryption schemes, there seems no foreseeable likelihood that symmetric encryption will be abandoned.

3. Finally, there is a feeling that key distribution is trivial when using public-key encryption, compared to the rather cumbersome handshaking involved with key distribution centers for symmetric encryption.

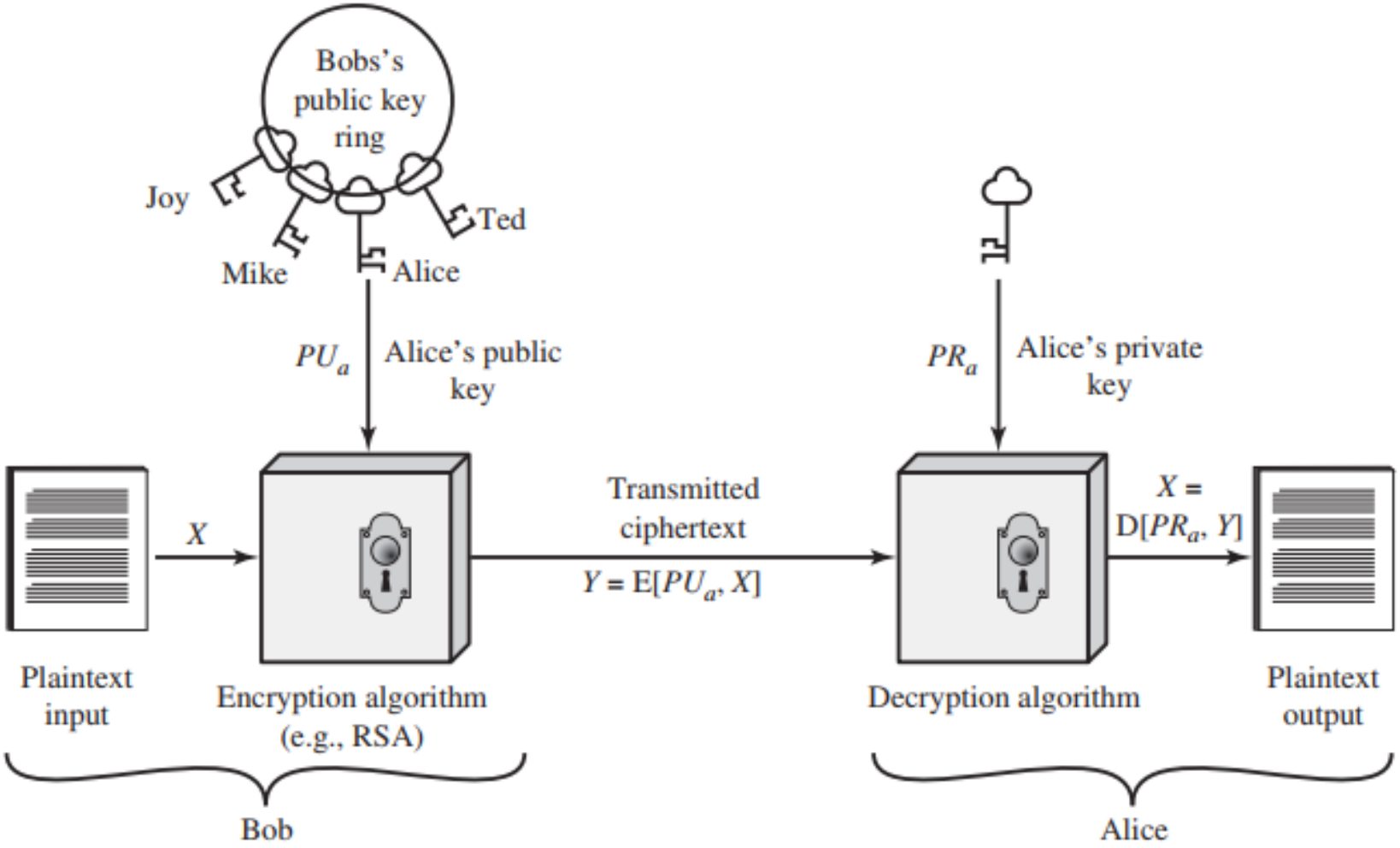
4. For public-key key distribution, some form of protocol is needed, often involving a central agent, and the procedures involved are no simpler or any more efficient than those required for symmetric encryption

Public-Key Encryption Structure

- A public-key encryption scheme has six ingredients
  - **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 key:** This is a pair of keys that have been selected so if one is used for encryption, the other is used for decryption.
    - The exact transformations performed by the encryption 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.

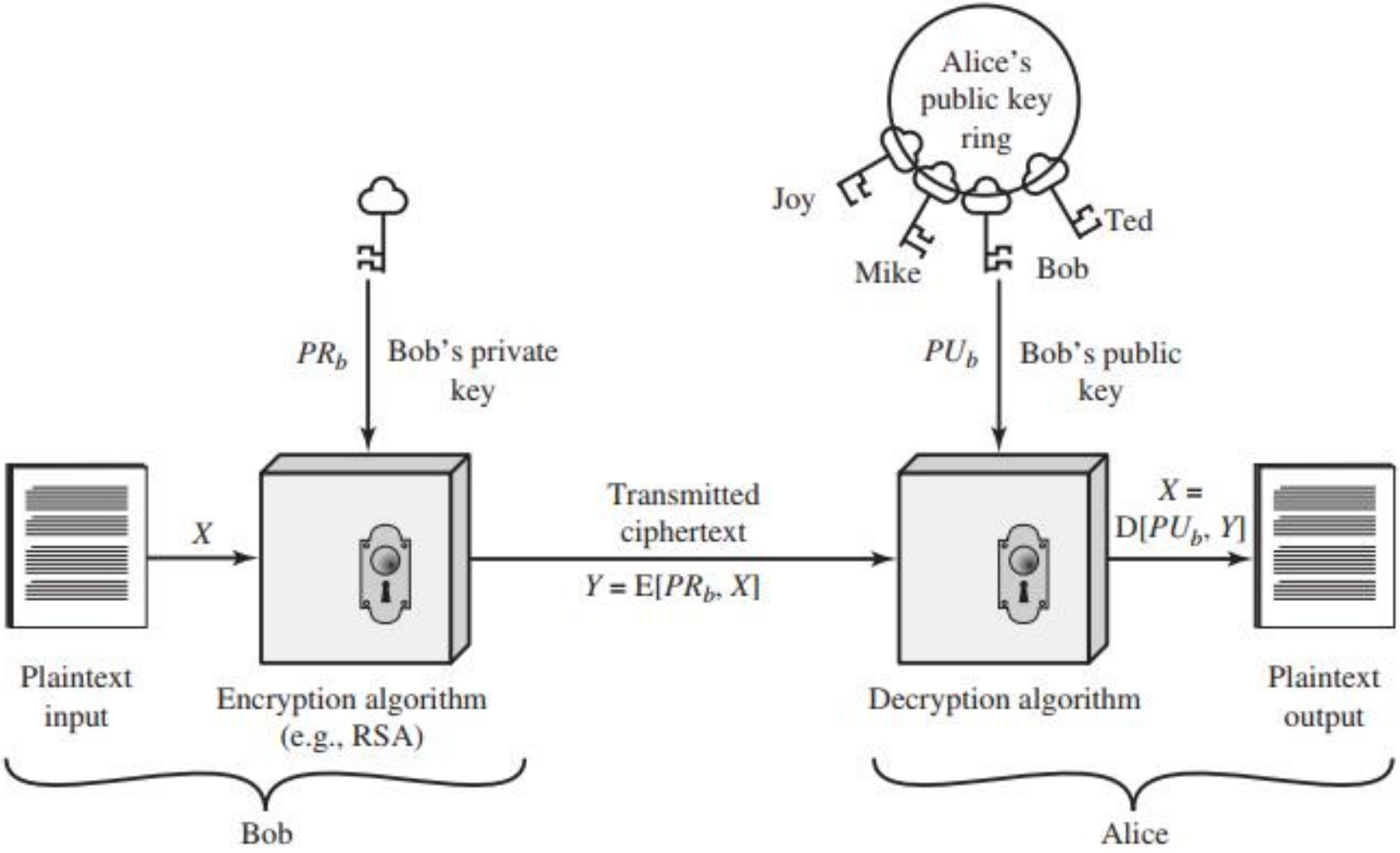
Public-Key Encryption Structure

- The public key of the pair is made public for others to use, while the private key is known only to its owner.
- A general-purpose public-key cryptographic algorithm relies on **one key for encryption and a different but related key for decryption.**
- **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.
     - (Figure 2.6a) each user maintains a collection of public keys obtained from others
  3. If Bob wishes to send a private 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.



(a) Encryption with public key

Figure 2.6 **Public-Key Cryptography**



(b) Encryption with private key

Figure 2.6 **Public-Key Cryptography**

Table 2.3 **Applications for Public-Key Cryptosystems**

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie–Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

Requirement for Public Key Cryptography

Self Reading

Requirements for Public-Key Cryptography

The cryptosystem illustrated in Figure 2.6 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 [DIFF76]:

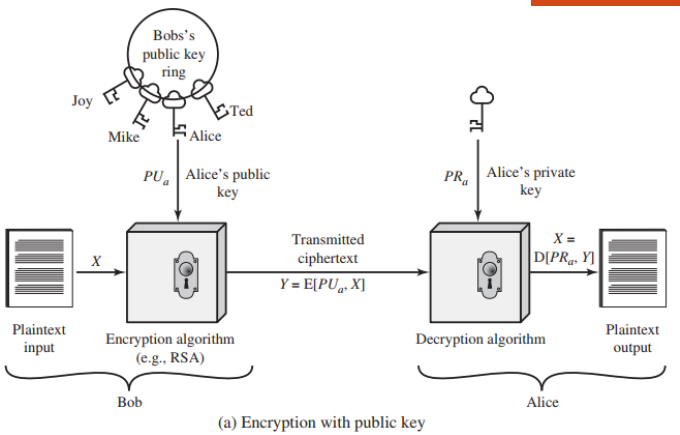
1. It is computationally easy for a party B to generate a pair (public key PU_b, private key PR_b).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext:

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

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



(a) Encryption with public key

Asymmetric Encryption Algorithms

RSA One of the first public-key schemes was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978 [RIVE78]. The RSA scheme has since reigned supreme as the most widely accepted and implemented approach to public-key encryption. RSA is a block cipher in which the plaintext and ciphertext are integers between 0 and $n - 1$ for some n.

used a public-key size (length of n) of 129 decimal digits, or around 428 bits. This result does not invalidate the use of RSA; it simply means that larger key sizes must be used. Currently, a 1024-bit key size (about 300 decimal digits) is considered strong enough for virtually all applications.

FACT 1. **Prime generation is easy:** It's easy to find a random prime number of a given size.

FACT 2. **Multiplication is easy:** Given p and q, it's easy to find their product, $n = pq$.

CONJECTURE 3. **Factoring is hard:** Given such an n, it appears to be quite hard to recover the prime factors p and q.

RSA Algorithm

Description of the Algorithm

One of the first public-key schemes was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978 [RIVE78]. The RSA scheme has since that time reigned supreme as the most widely accepted and implemented approach to public-key encryption. RSA is a block cipher in which the plaintext and ciphertext are integers between 0 and $n - 1$ for some n.

Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C:

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

Both sender and receiver must know the values of n and e, and only the receiver knows the value of d. This is a public-key encryption algorithm with a public key of $PU = \{e, n\}$ and a private key of $PR = \{d, n\}$. For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:

1. It is possible to find values of e, d, n such that $M^{ed} \bmod n = M$ for all $M < n$.

RSA Algorithm

2. It is relatively easy to calculate M^e and C^d for all values of $M < n$.
3. It is infeasible to determine d given e and n.

The first two requirements are easily met. The third requirement can be met for large values of e and n.

More should be said about the first requirement. We need to find a relationship of the form

$$M^{ed} \bmod n = M$$

The preceding relationship holds if e and d are multiplicative inverses modulo $\phi(n)$, where $\phi(n)$ is the Euler totient function. It is shown in Appendix B that for p, q prime, $\phi(pq) = (p - 1)(q - 1)$. $\phi(n)$, referred to as the Euler totient of n, is the number of positive integers less than n and relatively prime to n. The relationship between e and d can be expressed as

$$ed \bmod \phi(n) = 1$$

This is equivalent to saying

$$ed \bmod \phi(n) = 1$$
$$d \bmod \phi(n) = e^{-1}$$

That is, e and d are multiplicative inverses mod $\phi(n)$. According to the rules of modular arithmetic, this is true only if d (and therefore e) is relatively prime to $\phi(n)$. Equivalently, $\gcd(\phi(n), d) = 1$; that is, the greatest common divisor of $\phi(n)$ and d is 1.

Figure 21.7 summarizes the RSA algorithm. Begin by selecting two prime numbers, p and q, and calculating their product n, which is the modulus for encryption and

RSA Algorithm

Key Generation	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$

Decryption	
Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

Figure 21.7 **The RSA Algorithm**

RSA Algorithm: Example

Suppose user A has published its public key and user B wishes to send the message M to A. Then B calculates $C = M^e \pmod{n}$ and transmits C. On receipt of this ciphertext, user A decrypts by calculating $M = C^d \pmod{n}$.

An example, from [SING99], is shown in Figure 21.8. For this example, the keys were generated as follows:

1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \times 11 = 187$.
3. Calculate $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$.
5. Determine d such that $de \bmod 160 = 1$ and $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = (1 \times 160) + 1$.

The resulting keys are public key $PU = \{7, 187\}$ and private key $PR = \{23, 187\}$. The example shows the use of these keys for a plaintext input of $M = 88$. For encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows:

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$
$$88^1 \bmod 187 = 88$$
$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$
$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$
$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

For decryption, we calculate $M = 11^{23} \bmod 187$:

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$
$$11^1 \bmod 187 = 11$$
$$11^2 \bmod 187 = 121$$
$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$

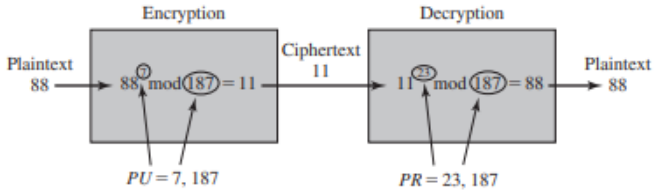


Figure 21.8 **Example of RSA Algorithm**

Digital Signature And Key Management

- Public-key algorithms are used in a variety of applications.
- In broad terms, these applications fall into two categories:
  - digital signatures, and various techniques to do with key management and distribution.
- With respect to key management and distribution, there are at least three distinct aspects to the use of public-key encryption in this regard:
  - The secure distribution of public keys
  - The use of public-key encryption to distribute secret keys
  - The use of public-key encryption to create temporary keys for message encryption

Digital Signature

- Public-key encryption can be used for authentication with a technique known as the **digital signature.**
- NIST FIPS PUB 186-4 [Digital Signature Standard (DSS), July 2013] defines a digital signature as follows:
  - *The result of a cryptographic transformation of data that, when properly implemented, provides a mechanism for* ***verifying:***
    - ***origin authentication***
    - ***data integrity and***
    - ***signatory non-repudiation.***
- Thus, a digital signature is a data-dependent bit pattern, generated by an agent as a function of a file, message, or other form of data block. Another agent can access the data block and its associated signature and verify:
  1. ***data block has been signed by the alleged signer,***
  2. ***the data block has not been altered since the signing.***
  3. ***the signer cannot repudiate the signature.***

Digital Signature

FIPS 186-4 specifies the use of one of three digital signature algorithms:

1. **Digital Signature Algorithm (DSA):** The original NIST-approved algorithm, which is based on the difficulty of computing discrete logarithms.
2. **RSA Digital Signature Algorithm:** Based on the RSA public-key algorithm.
3. **Elliptic Curve Digital Signature Algorithm (ECDSA):** Based on elliptic-curve cryptography.

Digital Signature

Steps

- Bob wants to send a message to Alice. He wants Alice to be certain that the message is indeed from him. For this purpose:
  1. Bob uses a secure hash function, such as SHA-512, to generate a hash value for the message.
  2. That hash value, together with Bob's private key, serve as input to a digital signature generation algorithm that produces a short block that functions as a digital signature.
  3. sends the message with the signature attached.

- When Alice receives the message plus signature, she:
  1. calculates a hash value for the message;
  2. provides the hash value and Bob's public key as inputs to a digital signature verification algorithm. If the algorithm returns the result that the signature is valid, Alice is assured that the message must have been signed by Bob.

- No one else has Bob's private key, and therefore no one else could have created a signature that could be verified for this message with Bob's public key.

- In addition, it is impossible to alter the message without access to Bob's private key, so the message is authenticated both in terms of source and in terms of data integrity.

- The digital signature does not provide confidentiality.
  - That is, the message being sent is safe from alteration, but not safe from eavesdropping.

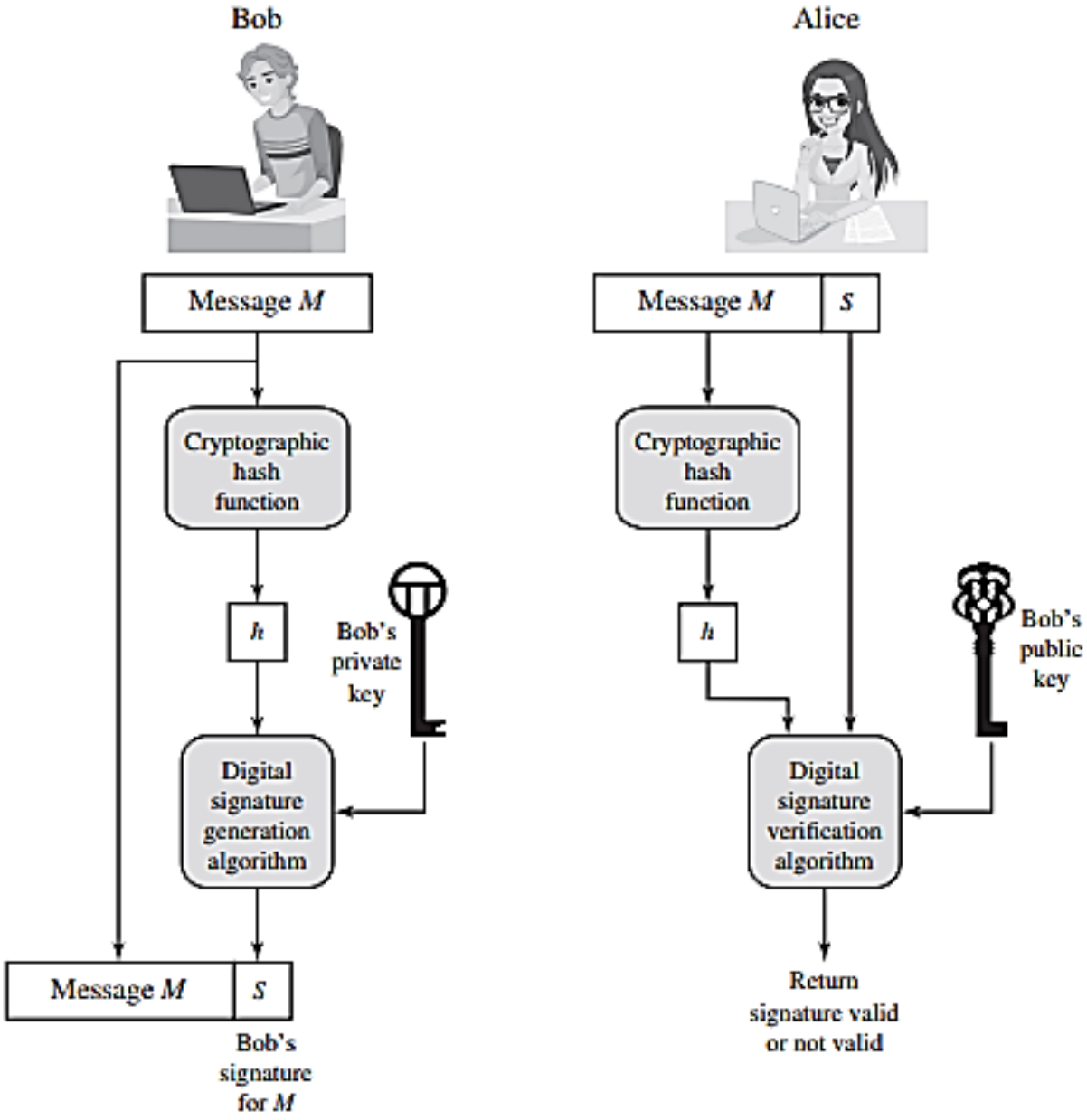


Figure 2.7 **Simplified Depiction of Essential Elements of Digital Signature Process**

Public-Key Certificates

- Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement. That is, some user could pretend to be Bob and send a public key to another participant or broadcast such a public key. Until such time as Bob discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for Bob and can use the forged keys for authentication.

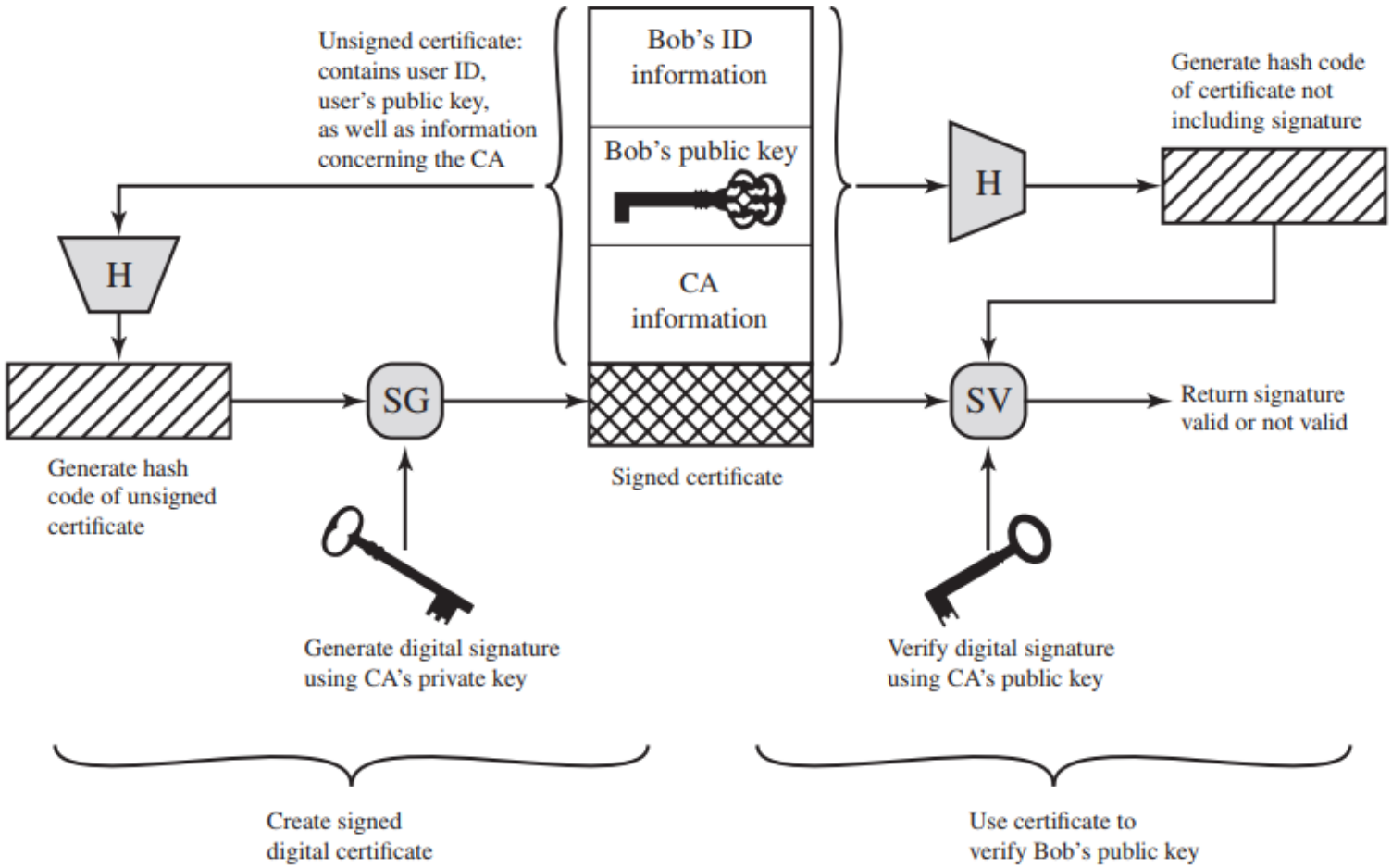


Figure 2.8 **Public-Key Certificate Use**