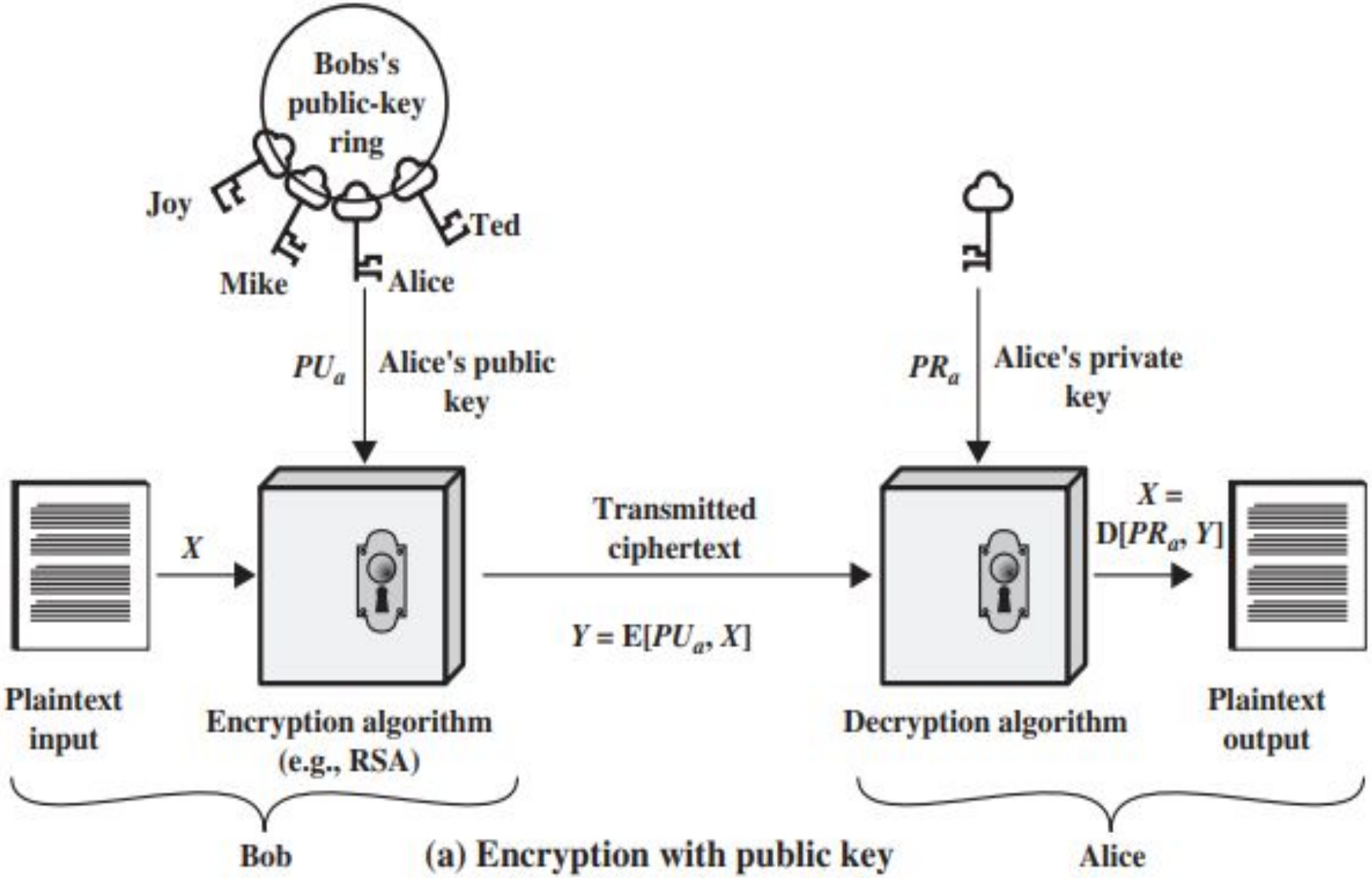



(a) Encryption with public key

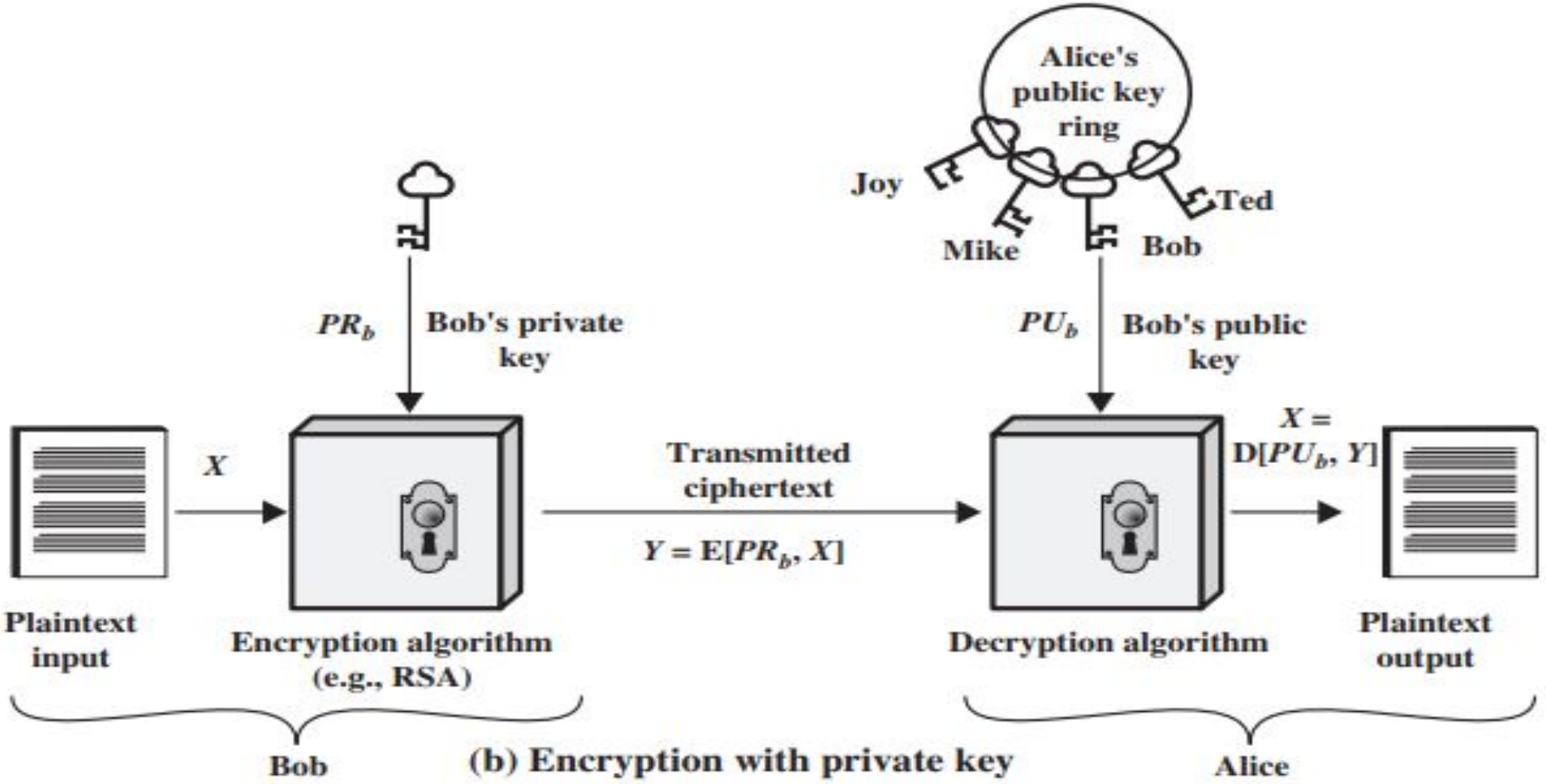



Figure 9.1 Public-Key Cryptography

Asymmetric Key Cryptography have following characteristics

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

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

## Public Key Encryption

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 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.

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.

# Public Key Cryptosystem

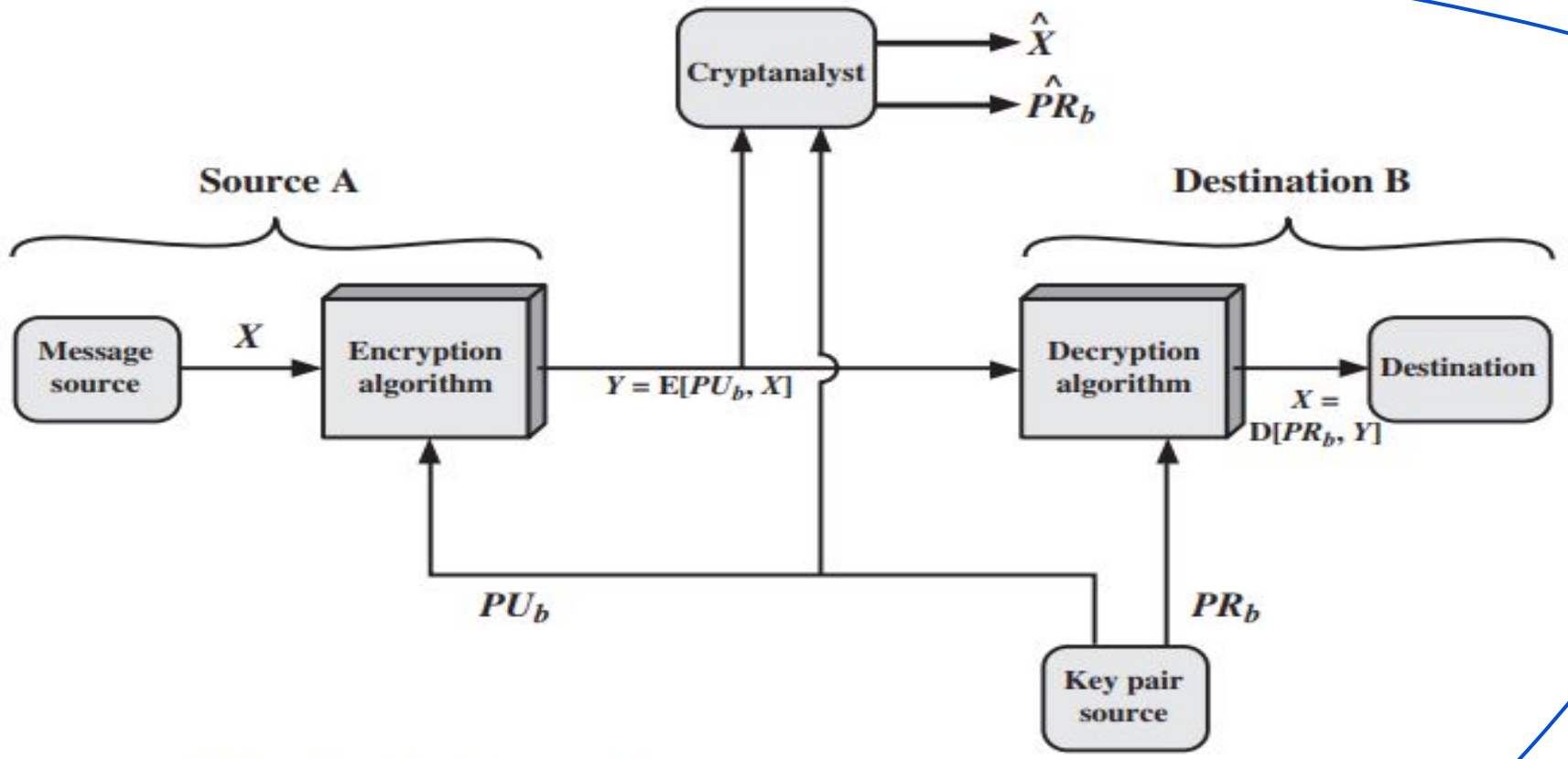



Figure 9.2 Public-Key Cryptosystem: Secrecy

# Public Key CryptoSystem

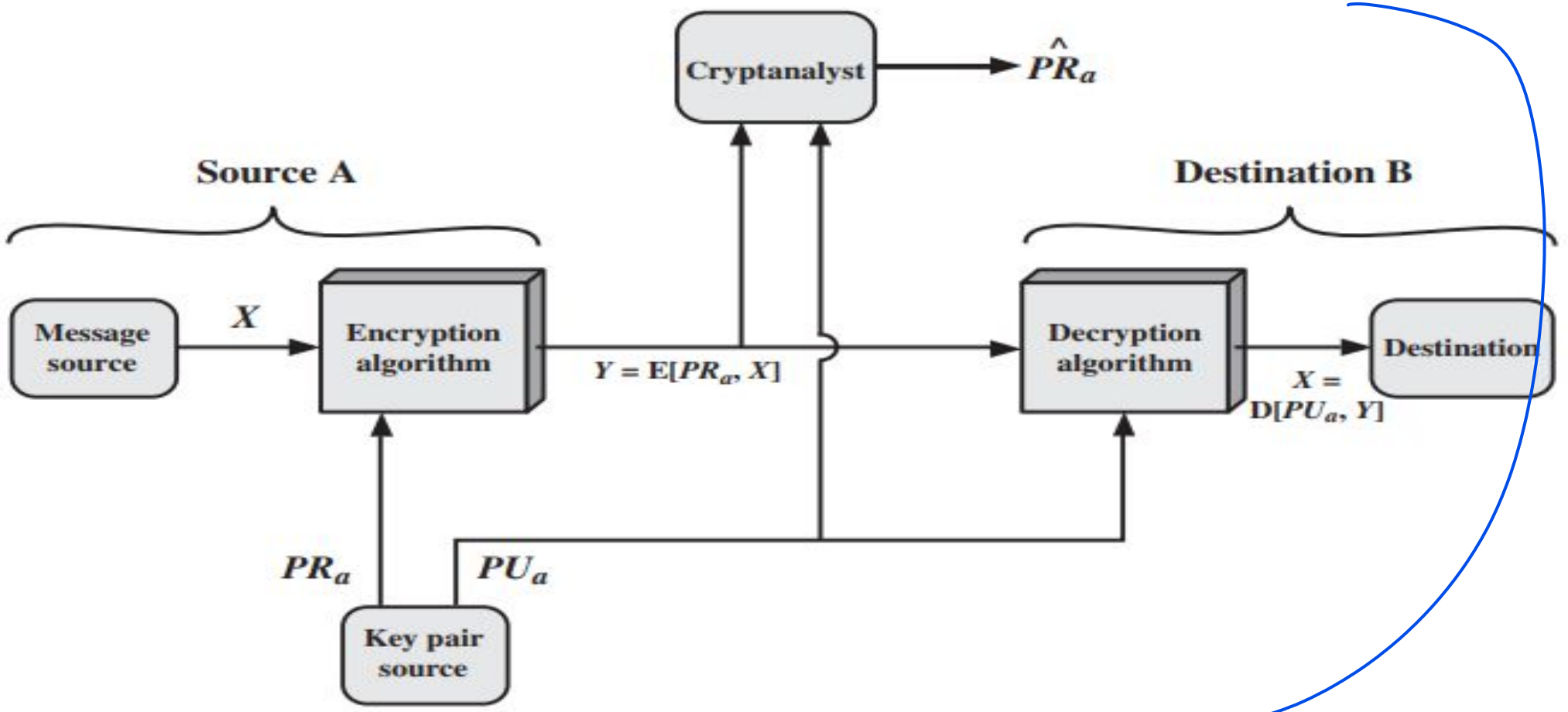



Figure 9.3 Public-Key Cryptosystem: Authentication

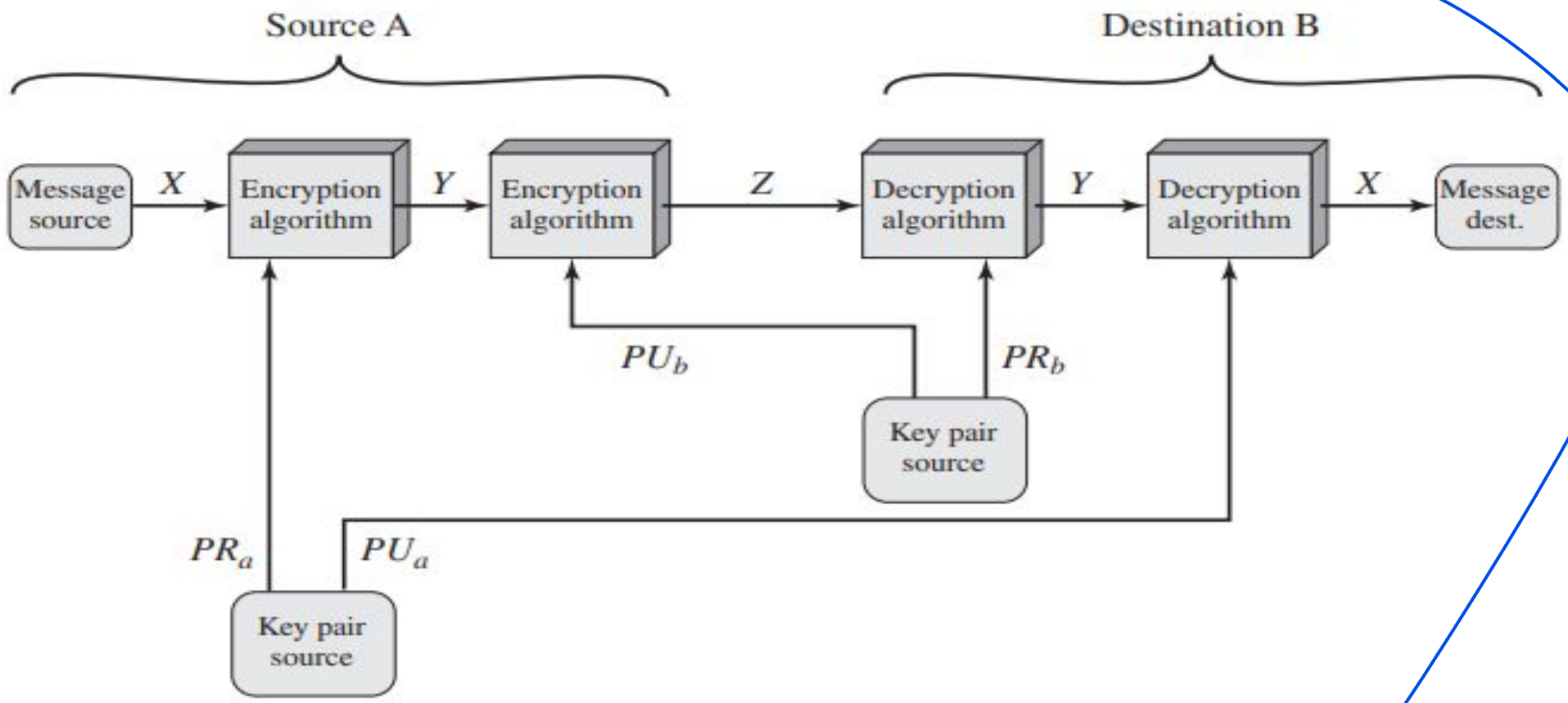



Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

Table 9.2 Conventional and Public-Key Encryption

| Conventional Encryption | Public-Key Encryption |
|---|---|
| *Needed to Work:*<br>1. The same algorithm with the same key is used for encryption and decryption.<br>2. The sender and receiver must share the algorithm and the key.<br>*Needed for Security:*<br>1. The key must be kept secret.<br>2. It must be impossible or at least impractical to decipher a message if the key is kept secret.<br>3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key. | *Needed to Work:*<br>1. One algorithm is used for encryption and a related algorithm for decryption with a pair of keys, one for encryption and one for decryption.<br>2. The sender and receiver must each have one of the matched pair of keys (not the same one).<br>*Needed for Security:*<br>1. One of the two keys must be kept secret.<br>2. It must be impossible or at least impractical to decipher a message if one of the keys is kept secret.<br>3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key. |

# Applications of Public Key Cryptography

**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.

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 |

# Requirements for public key cryptography

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 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:

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

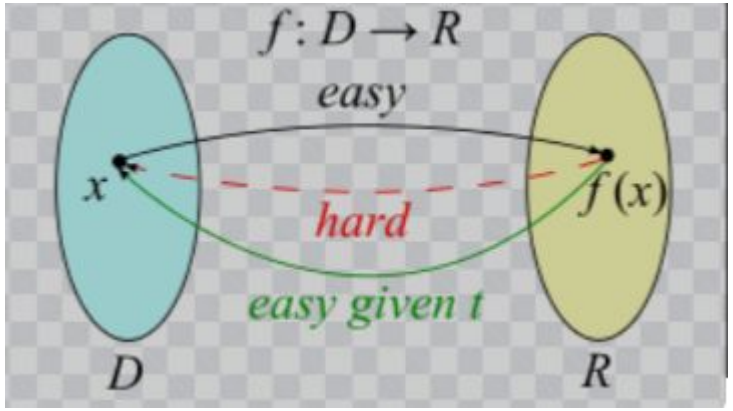
# One-Way Function

- A one-way function is a mathematical function that is significantly easier to compute in one direction (the forward direction) than in the opposite direction (the inverse direction).
- It might be possible, for example, to compute the function in the forward direction in seconds but to compute its inverse could take months or years, if at all possible.

Informally, a function f is a one-way function if

1. The description of f is publicly known and does not require any secret information for its operation.
2. Given x, it is easy to compute f(x).
3. Given y, in the range of f, it is hard to find an x such that f(x) = y

One Way Function




Trap Door Function: Trap door one-way function is a family of invertible functions fk

| | |
|---|---|
| $Y = \mathrm{f}_k(X)$ | easy, if $k$ and $X$ are known |
| $X = \mathrm{f}_k^{-1}(Y)$ | easy, if $k$ and $Y$ are known |
| $X = \mathrm{f}_k^{-1}(Y)$ | infeasible, if $Y$ is known but $k$ is not known |

# 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.

# RSA Algorithm

- RSA(Rivest-Shamir-Adleman) is an algorithm used for encrypt and decrypt messages.
- It is an asymmetric cryptographic algorithm. Asymmetric means that there are 2 different keys used.

**RSA algorithm** is an **asymmetric cryptography** algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key.** As the name describes that the Public Key is given to everyone and the Private key is kept private.

**An example of asymmetric cryptography:**

1. A client (for example browser) sends its public key to the server and requests some data.
2. The server encrypts the data using the client's public key and sends the encrypted data.
3. The client receives this data and decrypts it.

Since this is asymmetric, nobody else except the browser can decrypt the data even if a third party has the public key of the browser.

# 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 $d$ | $d \equiv e^{-1} \pmod{\phi(n)}$ |
| Public key | $PU = \{e, n\}$ |
| Private key | $PR = \{d, n\}$ |

**Encryption**

| | |
|---|---|
| Plaintext: | $M < n$ |
| Ciphertext: | $C = M^e \bmod n$ |

**Decryption**

| | |
|---|---|
| Ciphertext: | $C$ |
| Plaintext: | $M = C^d \bmod n$ |

# RSA-Algorithm Example