Privacy — Eavesdropper cannot get any information about plaintext message.

Authentication — Message was sent by the person who claims to send it.

Public key
Recall RSA:

Some Considerations:

1) Choose $e = 3$ so encryption is fast.
2) Computing $d$ from $(N, e)$ is as hard as factoring $N$.
3) If $x < 3\sqrt{N}$ decrypting $y = E(x)$ is easy. The difficulty of decryption relies on $x^e$ wrapping around $N$ a large and unknown number of times.
4) Public key cryptosystems typically much slower than symmetric key schemes. Therefore use public key system to establish a private key.

5) To send a message \( m \), randomize it by appending a random string in the high order bits. \( x = r \cdot m \) where \( \cdot \) denotes concatenation.

RSA Digital Signature:

A digital signature of a message \( m \) by Alice should:

1) Prove that Alice actually signed the message \( m \), i.e. no one else should be able to produce a signed message of their choice.

2) The adversary should not be able to use the signed message \( m \) to produce a signed version of \( m' \).
To sign \( m \) using RSA, Alice simply encrypts \( m \) using her private key:

\[ S(m) = m^d \pmod{N}. \]

To verify the signature, anyone can look up her public key \((N, e)\) and check that \( S(m)^e \equiv m \pmod{N} \).

In practice, we must prepare \( m \) suitably before decrypting it. \( m \) is usually restricted to at most half the length of \( N \), and is mapped to a number between 0 and \( N-1 \) by a redundancy mapping. For example, if \( m = m_1 m_2 \ldots m_k \) is a \( k \) byte string, then the redundant message might be

\[ m_r = m_1 \Pi(m_1) m_2 \Pi(m_2) \ldots m_k \Pi(m_k) \]

where \( \Pi \) is some fixed permutation of 8 bit strings. This prevents the following type of attack:

Choose some \( x \pmod{N} \), apply Alice’s public encryption function to obtain \( y = x^e \pmod{N} \). Claim that Alice signed \( y \) to obtain \( x \).
It is very unlikely to satisfy the redundancy format.

Certificates:

When Bob is sending a message to Alice, how does he verify that the posted public key \((N_a, e_a)\) is really Alice's key? Certificates provide a way of doing this. Certificates require a trusted party who everyone trusts, and whose public key \((N, e)\) is well known. The trusted party verifies that Alice is who she claims and signs a message that says:

Alice's public key is \((N_a, e_a)\) signed ...

Now Alice can publicly post this certificate and anyone who wishes can verify the trusted party's signature to reassure themselves about the authenticity of the key.
Message Authentication Codes (MACs): In the symmetric key setting, authentication is accomplished by a MAC. In this setting Alice and Bob share a secret key $K$, and are using a secure block cipher $(E_K, D_K)$ on $n$ bit blocks.

If the message $m$ to be authenticated is shorter than $n$ bits, then it can be authenticated by simply sending $(m, E_K(m))$. Since $E_K$ is indistinguishable from a random permutation, an adversary who does not know cannot authenticate any previously unseen messages.

To authenticate a long message: $m = m_1, m_2, \ldots, m_r$ where each $m_i$ is $n$ bit long, we can use CBC mode to create an $n$ bit tag $t$. Then the authenticated message is $(m, t)$:
\textbf{CBC MAC:}

\[ C(0) = 0^n \]
\[ C(i) = E_K(C(i-1) \oplus M_i) \]
\[ \text{tag} = C(l). \]

If the message length is fixed, i.e. \( l \) is not allowed to vary, then the CBC MAC can be proved to be secure.

For variable length messages provable security can be obtained by a simple augmentation of the above procedure. Pick a second block cipher (\( E'_K, D'_K \)). Let the authentication tag be \( E'_K(C(l)) \).
Formalizing security of MACs:

We allow the adversary $A$, access to a box that will authenticate messages $m$ of his choice. Thus the adversary obtains a list $(m_1, t_1), (m_2, t_2), \ldots, (m_j, t_j)$.

The advantage of the adversary

$$\text{Adv}(A) = \Pr[A \text{ can create } (m', t') \text{ which is a valid authenticated message and } m' \neq m_i \text{ for } i = j].$$

The formal proof of security of the fixed-length (length $l$) CBC MAC says:

1) If $E_k$ is a random permutation then

$$\text{Adv}(A) \leq \frac{1.5 \cdot j^2 \cdot l^2}{2^n}$$

2) Now a simple reduction shows that if $E_k$ is a secure block cipher:

$$\text{Adv}(A) \leq \varepsilon + \frac{1.5 \cdot j^2 \cdot l^2}{2^n}$$

where $\varepsilon$ is an upper bound on the adversary's advantage in distinguishing $E_k$ and a random permutation.
Let us examine some examples of insecure MACs:

\[ C(i) = E_k(M_i) \]
\[ \text{tag} = C(1) \oplus \cdots \oplus C(l) \]

The following adversary has advantage 1:
the adversary makes no queries and creates a tag for a new message \( M = x \times \)
i.e. \( l = 2 \) and \( M_1 = M_2 = x \). Now \[ \text{tag} = E_k(x) \oplus E_k(x) = 0^m \] no matter what \( E_k \) is.

We could strengthen this MAC by letting

\[ C(i) = E_k(i \cdot M_i) \quad \text{i.e. concatenate } i \text{ with } M_i. \]

Now the above attack no longer works.

The following adversary has advantage 1:
Query \( M_1 = a_1 \cdot a_2 \quad M_2 = a_1 \cdot b_2 \quad M_3 = b_1 \cdot a_2 \)
\( t_1 \quad t_2 \quad t_3 \).

Then the tag for \( M = b_1 \cdot b_2 \) is \( t_1 \oplus t_2 \oplus t_3 \).