Chapter 8 Network Security

Key Predistribution

- To use ciphers and authenticators, the communicating participants need to know what keys to use
  - For symmetric-key cipher, how does a pair of participants obtain the key they share?
  - For public-key cipher, how do participants know what public key belongs to a certain participant?

- A session key is a key used to secure a session—a single, relatively short episode of communication
  - Each distinct session between a pair of participants uses a new short-lived session key, which is always a symmetric-key for speed.
  - The participants determine what session key to use by means of a protocol—a session key establishment protocol
  - A session key establishment protocol needs its own security which is based on the longer-lived predistributed keys

- Motivation for using short-lived session keys
  - Less time for computationally intensive attacks
  - Less ciphertext for cryptanalysis
  - Less information exposed should the key be broken

- Predistribution of public keys
  - The algorithms to generate a matched pair of public and private keys are publicly known, and software that does it is widely available.
  - So a user could generate her own pair of public and private keys, keep the private key hidden, and publicize the public key

- How can a user publicize her public key in such a way that other participants can be sure it really belongs to her?
A complete scheme for certifying bindings between public keys and identities—what key belongs to who—is called a Public Key Infrastructure (PKI).

- A PKI starts with the ability to verify identities and bind them to keys out of band (i.e., not using the network)

Suppose Bob gets Alice’s public key \( x \) out of band, then Bob could create and publish a digitally signed statement that Alice’s key is \( x \)

- Such a digitally signed statement of a public key binding is called a public key certificate, or simply a certificate

When someone wants to verify Alice’s public key, they could do so by getting a copy of the certificate, as long as they trust Bob and know his public key

- Bob is playing the role often referred to as a certification authority (CA)

A certificate must include

- the identity of the entity being certified
- the public key of the entity being certified
- the identity of the signer
- the digital signature
- a digital signature algorithm identifier (which cryptographic hash and which cipher)
- an expiration time for the certificate (optional)

Certificates are often issued for email addresses and DNS domains

There are different ways a PKI could formalize the notion of trust

- Certification Authorities: trust is binary
- Web of Trust: trust is a matter of degree

Certification Authorities

- You either trust someone completely or not at all
- A certification authority (CA) is an entity claimed (by someone) to be trustworthy for verifying identities and issuing public key certificates.
There are commercial CAs, governmental CAs, and even free CAs.

To use a CA, you must know its key. You can learn that CA’s key if you can obtain a chain of CA-signed certificates that starts with a CA whose key you already know.

- Then you can believe any certificate signed by that new CA

Certification revocation

- If someone has discovered your private key, you would want to revoke the certificates that bind your compromised key to your identity
- Each CA can issue a certification revocation list (CRL), which is a digitally signed list of certificates that have been revoked
  - The CRL is periodically updated and made publicly available
- If all certificates have unlimited life spans, you could never take a certificate off the CRL. So, it is common to attach an expiration date to a certificate when it is issued
  - A revoked certificate can be removed from the CRL as soon as its expiration date is passed

Pre-Distribution of Symmetric Keys

- If Alice wants to use a symmetric-key cipher to communicate with Bob, she can’t just pick a key and send it to him because, without already having a key, they can’t encrypt this key to keep it confidential and they can’t authenticate each other

Pre-distribution is harder for symmetric keys than for public keys for two reasons:

- While only one public key per entity is sufficient for authentication and confidentiality, there must be a symmetric key for each pair of entities who wish to communicate. If there are N entities, that means N(N – 1)/2 keys.
- Unlike public keys, symmetric keys must be kept secret.
The most common solution is to use a Key Distribution Center (KDC). A KDC is a trusted entity that shares a secret key with each other entity. These keys can be established out of band. When Alice wishes to communicate with Bob, the KDC authenticates Alice and Bob, and generates a new session key for them to use. Kerberos is a widely used system based on this approach.

Authentication Protocols

- Authenticators alone are not enough to provide authentication for two reasons:
  - In a *replay attack*, an adversary retransmits a copy of a message that was previously sent → message is not original
  - In a *suppress-replay attack*, an adversary merely delays your message so that it is received at a time when it is no longer appropriate → the message is not timely

- How to ensure originality and timeliness in authentication protocols?
  - Include a timestamp in the message to ensure timeliness
    ▪ Distributed clock synchronization required
  - Include a *nonce* – a random number used only once – in the message to ensure originality
    ▪ Must keep track of past nonces
  - Use a timestamp or nonce in a *challenge-response* protocol
    ▪ Alice sends Bob a timestamp (or a nonce), n, challenging Bob to encrypt it in a response message
    ▪ Alice computes the decrypted value in the response message, if it is equal to n, then Bob’s identity is verified