CSE 410/565 Computer Security
Spring 2022

Lecture 11: Key Establishment and Applications

Department of Computer Science and Engineering
University at Buffalo

Authentication and Key Establishment

- Entity authentication allows us to establish that entities are who they claim they are
  - but this not enough for secure remote communication
- Building secure channels combines authentication with key establishment
  - key establishment is a mechanism of agreeing on a secret key that can be used for consecutive communication
  - if a key is established in an authenticated way, the parties are assured that they communicate with proper entities
    - key agreement can be performed with or without authentication

Key Establishment

- Let's look at the simplest communication of a secret key using public-key encryption
  - Bob has a public-private key pair with his public key being pk_B
  - Alice picks a session key s, encrypts it with Bob's key, and sends the result $\mathsf{Enc}_{pk_B}(s)$ to Bob
  - Bob decrypts it and now they share the same key
  - this can work, but we need to address at least two issues
    - this assumes that Alice can reliably obtain a correct version of Bob's public key
    - this doesn't take into account active adversaries
- This type of interaction is the basis for some key agreement mechanisms

Key Distribution Mechanisms

- There are many possibilities for key distribution
  - assume that we have an insecure network of n users
  - there is also a trusted authority (TA)
    - the TA's responsibilities could include checking user identities, issuing certificates, transmitting keys, etc.
- We divide all approaches in 3 categories
  - key pre-distribution
    - a TA distributes keying information during the setup phase using a secure channel
    - a pair of users is then able to compute a key known only to them

Key Distribution Mechanisms

- Types of key distribution (cont.)
  - session key distribution
    - on request, an online TA chooses session keys and distributes them to the users
    - this is done by encrypting the new keys using previously distributed secret keys
    - session keys are used for a fixed, rather short period of time
  - key agreement (a.k.a. key establishment or key exchange)
    - network users employ an interactive protocol to construct a session key
    - no TA's help is used
    - such protocols can be based on secret-key or public-key schemes

Key Distribution Mechanisms

- The difference between key distribution and key agreement:
  - in key distribution, one party (e.g., a TA) chooses a key and transmits it to one or more parties
    - key transmission is performed in an encrypted form
    - e.g., Kerberos
  - in key agreement, two or more parties jointly establish a secret key
    - communication is performed over a public channel
    - each participant contributes to the value of the resulting key
    - the key is not sent from one party to another
    - e.g., SSH, SSL

Key Distribution Mechanisms

- In the network, users may have long-lived keys
  - they could be secret keys known to a pair of users or to a user and the TA
  - they also could be private keys corresponding to public keys stored on users' certificates
- Pairs of users often employ short-lived session keys
  - a session key is used for a particular session and is discarded at the end of it
  - session keys are normally secret keys for a symmetric encryption scheme or MAC
  - we don't want compromise of a session key lead to compromise of long-term key

Key Distribution Mechanisms

- Since the network is insecure, we need to protect against attackers
  - the adversary might be one of the users in the network
- An active adversary can:
  - modify messages being transmitted on the network
  - save messages for later use
  - try to masquerade as another user in the network
- Adversary's goal might be:
  - fool someone into accepting an invalid key as valid
  - learn some information about the key being established
  - use another user's identity to establish a shared key with someone

Key Agreement

- Diffie-Hellman key agreement (simplified)
  - Alice and Bob want to compute a shared key, which must be unknown to eavesdroppers
  - they share public parameters: modulus p, element g $(1 < g < p)$, and modulus q for computation in the exponent
  - Alice chooses random $a \in \mathbb{Z}_q$, computes $g^a \bmod p$, and sends it to Bob
  - Bob chooses random $b \in \mathbb{Z}_q$, computes $g^b \bmod p$, and sends it to Alice

A		B
	$g^a \bmod p$ →	
secret a		secret b
	← $g^b \bmod p$	
compute		compute
$g^{ab} = (g^b)^a$		$g^{ab} = (g^a)^b$

Key Agreement

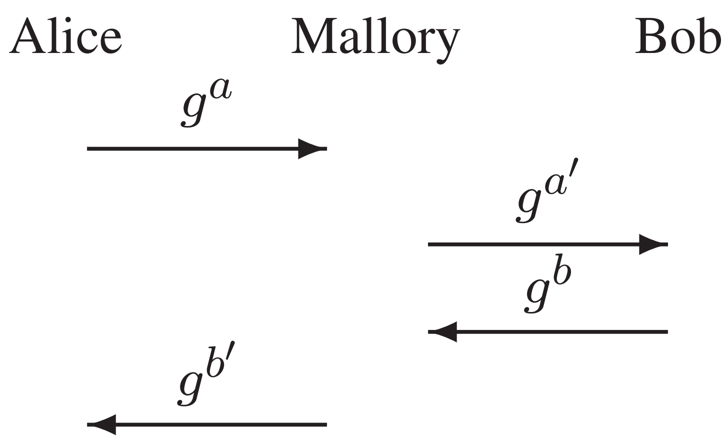
- Diffie-Hellman key agreement
  - the shared key is set to $g^{ab} \bmod p$
    - Alice computes $(g^b)^a \bmod p = g^{ab} \bmod p$
    - Bob computes $(g^a)^b \bmod p = g^{ab} \bmod p$
  - it is believed to be infeasible for an eavesdropper to compute g^{ab} given g^a and g^b
    - this assumption holds in groups in which the discrete log problem is hard and is known as Computational Diffie-Hellman assumption

Diffie-Hellman Key Exchange

- Diffie-Hellman key exchange
  - the security property holds only against a passive attacker
  - the protocol has a serious weakness in the presence of an active adversary
    - this is called a man-in-the-middle attack
    - Mallory will intercept messages between Alice and Bob and substitute her own
    - Alice establishes a shared key with Mallory and Bob also establishes a shared key with Mallory

Diffie-Hellman Key Exchange

- Man-in-the-middle attack on Diffie-Hellman key exchange



  - Alice shares the key $g^{ab'}$ with Mallory
  - Bob shares the key $g^{a'b}$ with Mallory
  - Alice and Bob do not share any key
  - what is Mallory capable of doing?

Diffie-Hellman Key Exchange

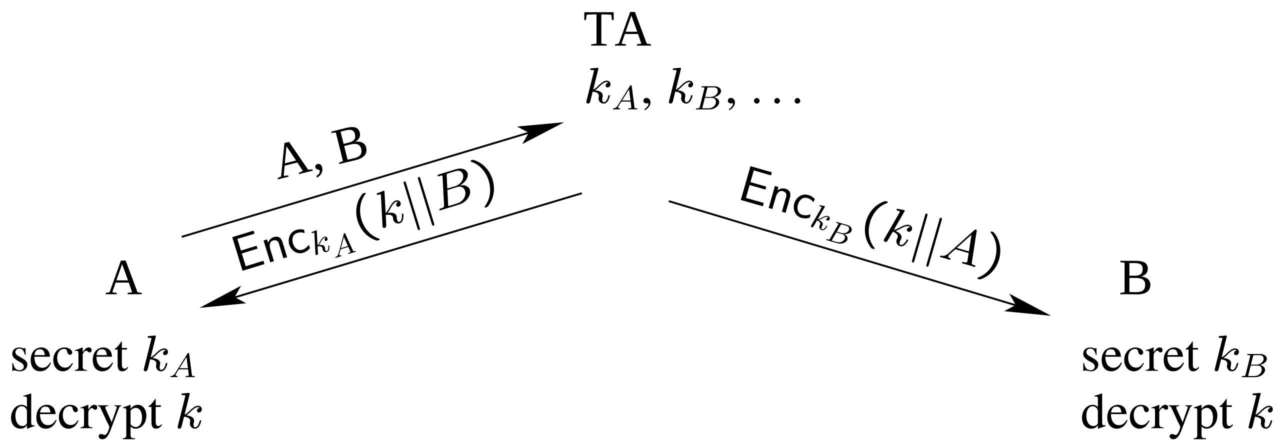
- Alice and Bob need to ensure they are exchanging messages with each other
  - there is a need for authentication
  - preceding this protocol with an authentication scheme is not guaranteed to solve the problem
    - authentication needs to be a part of the key exchange
    - this is called authenticated key exchange

Diffie-Hellman Key Exchange

- Authenticated Diffie-Hellman solves this problem using certificates and digital signatures
  - Alice and Bob have public-private key pairs, with the
  - their public keys are signed by an authority
  - during the protocol, they sign the values transmitted along with the identities

Session Key Distribution

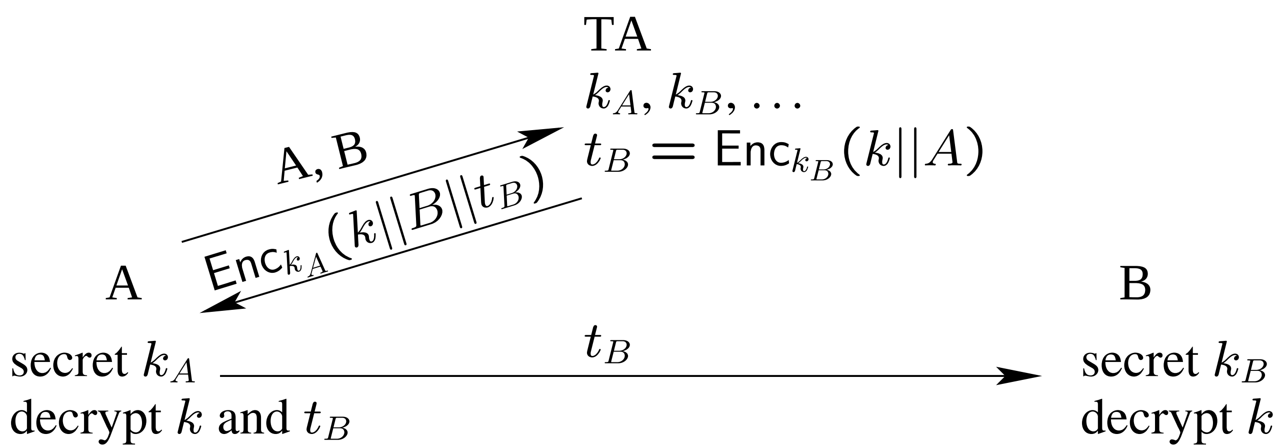
- Assume that the TA has a shared key with each user on the network
  - k_A is the key shared with Alice, k_B is the key shared with Bob, etc.
- The TA chooses session keys and distributes them in encrypted form upon user requests



  - is this enough?

Session Key Distribution

- Another way of achieving this is for the TA to communicate with Alice only and she will relay the TA's key to Bob



- Still the same problem: no freshness

Kerberos

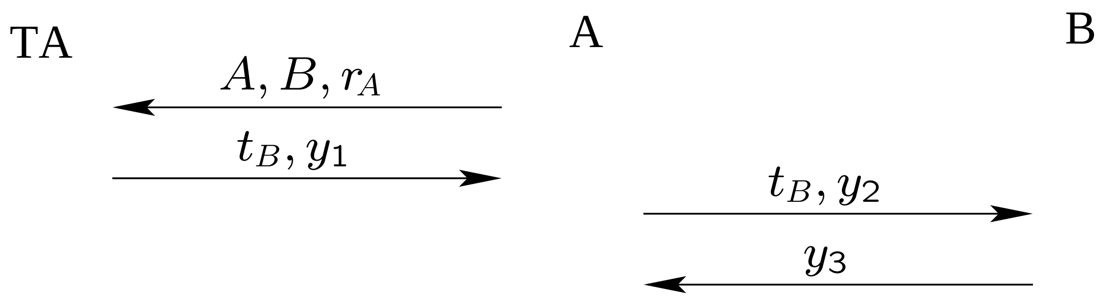
- Kerberos is a series of schemes for session key distribution developed at MIT in the 80–90s, based on Needham-Schroeder SKDS (1978)

- Simplified Kerberos V5

  - Alice chooses a random r_A and sends A, B, and r_A to the TA
  - the TA chooses a random session key k and a validity period L
  - the TA computes a ticket to Bob $t_B = \mathsf{Enc}_{k_B}(k||A||L)$ and $y_1 = \mathsf{Enc}_{k_A}(r_A||B||k||L)$ and sends them to Alice
  - Alice decrypts y_1 using k_A, obtains k, computes $y_2 = \mathsf{Enc}_k(A||\mathsf{time})$, and sends t_B and y_2 to Bob
  - Bob decrypts t_B, obtains k, and uses it to decrypt y_2 and obtain time
  - Bob computes $y_3 = \mathsf{Enc}_k(\mathsf{time} + 1)$ and sends it to Alice

Kerberos

- Information flow in Kerberos V5



  - where $t_B = \mathsf{Enc}_{k_B}(k||A||L)$, $y_1 = \mathsf{Enc}_{k_A}(r_A||B||k||L)$, $y_2 = \mathsf{Enc}_k(A||\mathsf{time})$, and $y_3 = \mathsf{Enc}_k(\mathsf{time} + 1)$
  - several checks are necessary
    - Alice checks that y_1 is in the correct form and has her r_A
    - Bob checks that $\mathsf{time} \leq L$ in y_2
    - Alice checks that y_3 decrypts to $\mathsf{time} + 1$

Kerberos

- The purpose of the validity period L is to prevent an attacker from storing old messages and retransmitting them at a later time
  - this limits the time period during which a certain type of attack discovered on Needham-Schroeder scheme can be carried out
- Kerberos requires users to have synchronized clocks
  - the current time is used to determine the validity of the session key k
  - perfect synchronization is hard to achieve in practice, so small amount of variation should be allowed
- There are many other solutions to all of key pre-distribution, session key distribution, and key agreement

Applications for Building Secure Channels

- Internet Security (IPsec) and Internet Key Exchange (IKE) protocols
  - provides authentication and encryption mechanisms at low network layer
- Kerberos
  - a network authentication tool used in Windows operating system
- Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocols
  - public-key based authentication technique with wide use on the web
- Secure Shell (SSH)
  - public-key authentication protocol suite for secure remote login

Internet Security (IPsec)

- We often think of techniques for protecting messages transmitted over networks as used at the application level
- Protection at low level, however, can be effective for securing communication over the Internet
  - if we can protect packet addressing information, we can prevent manipulation of this information
  - the suite of authentication protocols standardized for internet security is called Internet Key Exchange (IKE)
  - IKE includes a variety of algorithms that can be used for integrity and confidentiality protection
- Suppose two parties wish to conduct confidential communications by using end-to-end encryption

Internet Security (IPsec)

- If end-to-end encryption operates at the application level, then information in the IP header is the subject of attacks
  - e.g., IP spoofing (masquerading by using a fake source IP address)
- Virtually all active attacks that we've seen so far require Mallory to perform manipulation at the IP level
  - message interception, man-in-the-middle, etc.
- Internet Protocol Security (IPsec) standard has been developed to add cryptographic protection to the IP header
  - it stipulates a mandatory authentication protection for IP header
  - it also allows for optional confidentiality protection for the endpoint-identity information

Internet Security (IPsec)

- Thus, IPsec effectively prevents many active attacks since IP header manipulation can now be detected

- To use IPsec for authentication, there will be an additional header called the Authentication Header (AH)
  - it is positioned between the IP header and the higher-layer data
  - it provides data integrity and data origin authentication and covers the IP header fields that don't change in transit and packet payload

- Optional confidentiality protection can be added
  - datagrams called Encapsulating Security Payloads (ESP) are specified for this purpose
  - ESP follows AH in a packet and does not protect the packet header

Secure Shell (SSH)

- Secure Shell (SSH) is a public-key based authentication protocol suite
  - it enables a user to securely login onto a remote server host machine from a client machine through an insecure network
  - it also allows to securely execute commands on the remote host and move files from one host to another
- SSH is in wide use today
  - a server can be run on many operating systems
    - a server will work if the operating system supports interactive command sessions for remote users
  - a client can be run on any operating system

Secure Shell (SSH)

- The basic idea behind SSH protocol
  - a user on a client machine downloads a public key of a remote server
  - he establishes a secure channel between the client and the server using the downloaded key and the user's cryptographic credentials
  - even if the user's credentials are only a password, they will be sent encrypted to the server
- The SSH protocol suite consists of three major components
  - SSH Transport Layer Protocol
    - provides server authentication to the client
    - the server host uses its public-private key pair and the client uses the host's public key

Secure Shell (SSH)

- The SSH protocol major components (cont.)
  - SSH User Authentication Protocol
    - it runs over the unilateral authentication channel established by the transport layer protocol
    - it achieves entity authentication from a client-side user to the server
    - it can be based on a public-key, password, or other mechanisms
    - at the end of it a mutually authenticated secure channel is formed
  - SSH Connection Protocol
    - materializes an encrypted communication channel
    - tunnels it into several secure logical channels for various communication purposes

Secure Shell (SSH)

- SSH-1 was discovered to have design flaws and should be avoided
- SSH-2 provides better security and has been standardized by IETF
- SSH Transport Layer Protocol
  - a server host maintains a public-private key pair for each required signature algorithm
  - a client must have a priori knowledge of the server's host public key
  - SSH supports two trust models on servers' public keys:
    - the client has a local database of host names and the corresponding public keys
    - the host name and its public key is certified by a trusted certification authority

Secure Shell (SSH)

- The list of known hosts' public keys is stored in `$HOME/.ssh/known_hosts`
- An entry might look like this:

```
timberlake.cse.buffalo.edu,128.205.36.8 ssh-rsa AAAAB3
NzaC1yc2EAAAABIwAAAIEArov5ZnZlpAETHjEvmLk7J/1g65JYIHYq
r6lfYWTHlTT20+IxfcWGX4vtsfcYwwpzLxhwlWTjah7/fK2MwgU1Lo
/HDDcjDZrpCFXN4pTAosLUdmV5uqadwNFbbtDTAESrjxJ/beAEwYZ/
Gvy/V36rZRFFWeFBMrDUiTXirc0NP80=
```

Secure Shell (SSH)

- When a user connects to a machine, the public key of which is not stored locally or has changed, SSH will ask the user to verify the public key
- Such verification happens in the form of a fingerprint
  - fingerprint(host key)$=h$(host key), where h is an agreed upon cryptographic hash function
- To verify the key, you might
  - have an authenticated copy of the key on another machine
  - have generated the key (for your workstation) and know it
  - call the security administrator and verify the fingerprint over the phone

Secure Shell (SSH)

- SSH key exchange protocol
  - each key exchange is initiated by the client
    - the server listens on a specific port, commonly port 22
  - SSH-2 uses Diffie-Hellman key exchange for session key agreement
  - we'll use the following notation:
    - C is the client and S is the server
    - p is a large prime, and $g \in \mathbb{Z}_p^*$ is an element with necessary properties
    - V_C and V_S are the client's and the server's versions, respectively
    - pk_S is the server's public key
    - I_C and I_S are the client's and the server's initial messages exchanged before this part begins

Secure Shell (SSH)

- SSH key exchange protocol
  - C chooses a random number x_C, computes $y_C = g^{x_C}$ and sends it to S
  - S chooses a random number x_S, computes

$$y_S = g^{x_S},\ k = y_C^{x_S} = g^{x_C x_S},$$
$$H = h(V_C||V_S||I_C||I_S||pk_S||y_C||y_S||k),\ \mathsf{sig}_{sk_S}(H)$$

    and sends pk_S, y_S and $\mathsf{sig}_{sk_S}(H)$ to C
  - C verifies that pk_S is really the host key for S
  - C then computes $k = y_S^{x_C} = g^{x_C x_S}$, H (as above), and verifies the signature $\mathsf{sig}_{sk_S}(H)$, and accepts if the verification passes
- After the key exchange, the parties use k to secure the communication

Secure Shell (SSH)

- The next step is to authenticate the client using a suitable mechanism
- One of the goals of SSH is to improve security on the internet in a progressive manner
  - this is why any suitable method of verifying the server's key is permitted
  - a variety of user authentication mechanisms is supported as well
  - this allows for quick deployment and backward compatibility
  - this is why it's been popularly implemented and widely used

Kerberos

- Today a user might need to access various resources for related purposes
  - assume Alice is a member of an enterprise
  - at work she has access to her projects at a project server
  - she also can manage her own HR related issues at the HR server
  - additionally, she manages her patent filing at an intellectual property server, etc.
- It is infeasible to require her to maintain different credentials for each task (e.g., remember many passwords or carry many smartcards)
- A suitable network solution for this environment is the Kerberos Authentication Protocol

Kerberos

- Recall that Kerberos is a session key distribution scheme based on symmetric keys
  - each user shares a long-term secret key with a trusted third party
  - when Alice would like to talk to Bob, she requests a session key from that trusted party
- Windows uses Kerberos as its network authentication basis
  - it uses Kerberos version 5, which is a free software
  - it can be downloaded from http://web.mit.edu/kerberos/www/
  - the exportation restrictions on cryptographic software make it available only in the US

Kerberos

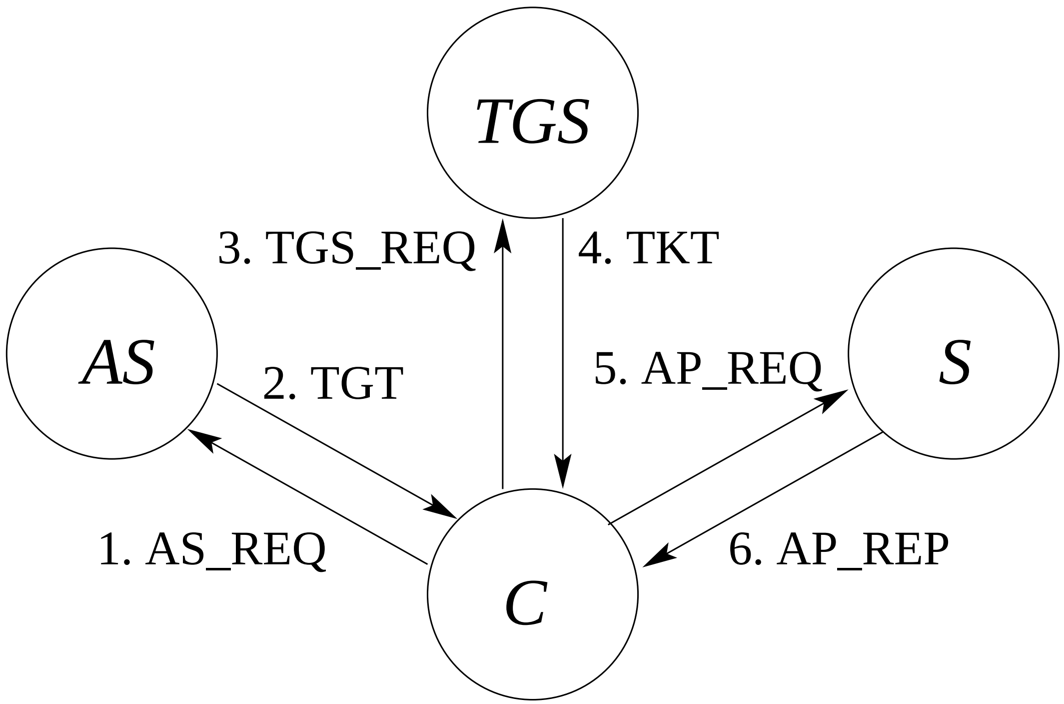
- The Kerberos Authentication Protocol consists of a suite of sub-protocols called exchanges
- Three sub-protocols are used for authentication:
  - the Authentication Service (AS) exchange
    - runs between a client C and an authentication server AS
  - the Ticket-Granting Service (TGS) exchange
    - runs between C and a ticket granting server TGS after the AS exchange
  - the client/server Authentication Application (AP) exchange
    - runs between C and an application server S after the TGS exchange

Kerberos

- There are five principals participating in the exchanges with the following roles:
  - U: a (human) user who gives instructions to her client process and enters password for authentication
  - C: a client (process) that requests network service on behalf of the user
  - KDC: key distribution center is a collective name for the following two authentication servers
    - AS: an authentication server that issues ticket granting tickets (TGT) for subsequent TGS exchanges
    - TGS: a ticket granting server that issues tickets for applications
  - S: an application server which provides an application resource to C

Kerberos

- Kerberos exchanges



Kerberos

- A ticket granting ticket (TGT) has two parts
  - one part for C encrypted under a key derived from password
  - another part for TGS encrypted under a key shared by AS and TGS
  - both contain a session key $k_{C,TGS}$ to be used between C and TGS
- A ticket (TKT) also has two parts
  - one part is for C to use encrypted under the session key $k_{C,TGS}$
  - another part is for S encrypted a key shared between S and TGS
  - both parts also include a session key $k_{C,S}$
- KDC is divided into AS and TGS to permit single sign-on
  - this gives flexibility and the ability to place TGSs in different domains

Federated Systems

- Kerberos can also be used to provide authentication across different administrative domains
  - within a single system, called realm, each user is registered with the Kerberos server
  - the Kerberos server also shares a key with each service provider
  - Kerberos provides a mechanism for supporting interrealm authentication to let users use external servers
  - in a federated system, one Kerberos server must trust another Kerberos server to authenticate users
  - a user can be issued a ticket granting ticket for a remote ticket granting server

Federated Systems

- Shibboleth is an open-source project that provides single sign-on capabilities
  - authorization is based on attributes and privacy is a builtin property of the authentication mechanism
  - when a user wants to authenticate with a service provider SP, it is redirected to its identity provider IdP
  - once the user authenticates, the IdP issues a response to the SP and the user is redirected back to the SP
  - Shibboleth implements the OASIS Security Assertion Markup Language
- OpenId Connect uses OAth, which is a standard for authorization of resources and does not deal with authentication

SSL and TLS

- The Secure Sockets Layer Protocol (SSL) is an important authentication protocol widely used on the web
  - the term "sockets" refers to standard communication channels linking peer processes on client/server machines
  - it runs under the application-layer protocols such as HTTP (Hypertext Transfer Protocol) and IMAP (Internet Messaging Access Protocol) and above network layer protocols (e.g., TCP/IP)
  - when socket-layer communications are secured, communications in all application-layer protocols will be secured in the same manner
  - SSL was originally developed by Netscape Communications Corporation as an integral part of the web browser and web server
  - it was later accepted by other developers

SSL and TLS

- SSL eventually evolved into an internet standard called Transport Layer Security (TLS)
  - TLS is based on SSL and is not drastically different from SSL
  - we'll focus on the standard track TLS in this lecture (SSL is currently deprecated)
- TLS is composed of two layered protocols
  - the TLS Record Protocol
  - the TLS Handshake Protocol
- The Handshake Protocol runs on top of the Record Protocol

TLS

- The TLS Record Protocol provides secure encapsulation of the communication channel by higher layer application protocols
  - it partitions data to be transmitted into blocks
  - optionally compresses the data
  - applies symmetric key algorithms to achieve confidentiality and integrity
    - used to have separate encryption and MAC algorithms, while the current version TLS 1.3 instead uses authenticated encryption
- The TLS Handshake Protocol is used to set up a secure session connection
  - it allows the client and server to authenticate each other
  - negotiate cryptographic algorithms and cryptographic keys (for integrity and confidentiality)

TLS

- TLS Handshake Protocol
  - it is stateful process running on the client and server machines
  - a stateful connection is called a session, where the communication peers perform the following steps:
    - exchange hello messages to agree on algorithms, exchange random values, and check for session resumption
    - exchange cryptographic parameters to be able to agree on a secret (called master secret)
    - exchange certificates and cryptographic information to allow them to authenticate one to another
    - generate session secrets from exchanged information
    - verify that their peer calculated the same parameters at the end of the handshake without tampering by an attacker

TLS Handshake Protocol

- The established channel is then passed on to the TLS Record Protocol for processing higher level application communications

- Simplified version of the handshake protocol:

  – $C \to S$: ClientHello;

  – $S \to C$: Server Hello,
  ServerCertificate (optional),
  ServerKeyExchange (optional),
  Certificate Request (optional),
  ServerHelloDone;

  – $C \to S$: ClientCertificate (optional),
  ClientKeyExchange,
  CertificateVerify (optional),
  ClientFinished;

  – $S \to C$: ServerFinished.

TLS Handshake Protocol

- Hello message exchange
  - the server and the client exchange protocol versions, random numbers, session ID, cryptographic algorithms, and compression methods
  - the client will provide a session ID, if the session is to be resumed
- Server's certificate
  - if the server's certificate is to be authenticated, the server will send it
  - X.509.v3 certificates are used
  - each certificate contains sufficient information about the certificate owner's name and public key and the issuing certificate authority

TLS Handshake Protocol

- Server's key exchange material
  - ServerKeyExchange contains the server's public key material matching the certificate list
  - material for DH key agreement will be included here as (p, g, g^{x_S})
  - the server that provides non-anonymous services can request client's authentication here as well
- Client's response
  - the content of ClientKeyExchange message will depend on the public key algorithm agreed between the server and the client
    - in the case of RSA, the client used to generate a 48-byte master key and encrypt it under the server's certified RSA public key
  - client's certificate (if any) will be provided at this stage

TLS Handshake Protocol

- Finished message exchange
  - the client sends the ClientFinished message which used to include a keyed (under the master key) HMAC
  - it allows the server to confirm the proper handshake executed at the client side
  - the server then sends a similar ServerFinished message
- Example
  - consider a typical run of the Handshake Protocol where the client chooses to be anonymous
  - such one-directional authentication is the most common in e-commerce applications

TLS Handshake Protocol

- Example run of the TLS handshake protocol (older version)
  - $C \to S$: ClientHello.protocol_version = "TLS Version 1.0",
    ClientHello.random = T_C, N_C,
    ClientHello.session_id = "NULL",
    ClientHello.crypto_suite = "RSA: Enc, SHA-1: HMAC",
    ClientHello.compression_method = "NULL";
  - $S \to C$: ServerHello.protocol_version = "TLS Version 1.0",
    ServerHello.random = T_S, N_S,
    ServerHello.session_id = "abc123",
    ServerHello.crypto_suite = "RSA: Enc, SHA-1: HMAC",
    ServerHello.compression_method = "NULL",
    ServerCertificate = server's X.509 certificate,
    ServerHelloDone;
  - $C \to S$: ClientKeyExchange = RSA_Enc(master_secret),
    ClientFinished = SHA-1(master_secret$||C||N_C||N_S \ldots$);
  - $S \to C$: ServerFinished = SHA-1(master_secret$||S||N_S||N_C \ldots$).

TLS Handshake Protocol

- TLS 1.3 has significant changes to the Handshake Protocol from prior versions
  - the handshake uses fewer interactions
    - encryption can be used as early as in the second message
  - the specification mandates forward secrecy
    - this makes the use of many algorithms in previous versions unacceptable
  - obsolete and insecure features from TLS 1.2 were removed
    - this includes MD5, SHA-1, RC4, DES, 3DES, etc.

TLS Protocols

- Recent specifications of TLS also include other protocols
  - Alert protocol
    - used to convey TLS-related alerts (based on pre-defined codes) to the peer entity
  - Heartbeat protocol
    - ensures that the other party is still alive
    - generates activity (prevents firewall closure and enables the use of connectionless service)

SSL/TLS Security

- Over the years, many attacks on and abuses of SSL and TLS have been discoreved
  - these include attacks on the handshake and record protocols, certificate-related attacks, and others
  - the largest attack was on OpenSSL's implementation of TLS's Heartbeat Protocol
    - the Heartbleed expoit was able to read memory of a remote server
    - the memory could store private keys, user ids and passwords, and other sensitive information

Summary

- Key establishment
  - session key distribution
  - key agreement protocols
- Applications and standards
  - IPsec and IKE: IP packet level integrity and confidentiality
  - SSL and TLS: socket level secure channel for all applications
  - Kerberos: network authentication framework that allows for single sign-on
  - Secure Shell: secure remote login and file transfer