Security

Security with Distributed Systems

Why Security?

- The need for security mechanisms in distributed systems arises from the desire to share resources
- Resources must be protected against unauthorized access, as enemies (attackers) have access to the same networks as we do
- We need to protect the integrity and privacy of information, as well as enforce security policies against the most determined attacks
- The main goal of security is to restrict access to information and resources to just those principals that are authorized to have access

Meet the Principals

Alice	First participant
Bob	Second participant
Carol	Participant in three- and four-party protocols
Dave	Participant in four-party protocols
Eve	Eavesdropper
Mallory	Malicious attacker
Sara	A server

Threats and Attacks

- **Leakage** - the acquisition of information by unauthorised recipients
- **Tampering** - the unauthorised alteration of information
- **Vandalism** - interference with the proper operation of a system without gain to the perpetrator

Threats to the Channel

- **Eavesdropping** - obtaining copies of messages without authority
- **Masquerading** - sending and receiving messages using the identity of another principal without their authority
- **Message Tampering** - intercepting messages and altering their contents before passing them on to the intended recipient (e.g., the "man-in-the-middle" attack)
- **Replaying** - storing intercepted messages and sending them at a later date
- **Denial of Service** - flooding a channel or other resource with messages in order to deny access for others

How Do Attacks Happen?

- Successful attacks depend upon the discovery of loopholes in the security of systems
- When the Internet was designed and implemented, security was NOT a priority
- Security (when used) needs to be carefully thought out
- If transmission of a message can be observed, some information can be gleaned from its mere existence

Securing Electronic Transactions: Examples

- **E-mail** - the contents of messages must be kept secret and/or the contents/sender of a message must be authenticated
- **Purchase of Goods/Services** - selecting and paying for goods via the web; delivered by downloading via the Internet (digital products); supplied via a delivery service (tangible goods)
- **Banking Transactions** - on-line banking providing all of the facilities of conventional "high street" banks
- **Micro-transactions** - non-traditional payment methods where services are paid for by a fraction of a cent (and the payment overheads must be kept correspondingly low)

Securing Web Purchases

- Authenticate the vendor to the buyer, so that the buyer can be confident that they are in contact with the correct vendor
- Keep the buyer's details (credit-card number, etc.) from falling into the wrong hands
- Ensure that the purchase details are not altered when transmitted from the buyer to the vendor
- If digital content is purchased, we need to ensure that the content is delivered unaltered to the buyer

Interesting Characteristics of Web Purchasing

- The identity of the buyer is not normally required by the vendor, only the purchase details are required (especially if delivery is not required)
- In certain cases, the credit-card details are not required by the vendor, just an assurance that the goods can be paid for
- The bank (that will pay the vendor on behalf of the buyer) does not need to know the details of the purchase (i.e., what was purchased), only how much needs to be paid
- The need for non-repudiation is important, as the vendor does not want a buyer claiming -- at a later date -- not to have purchased goods

Designing Secure Systems

- The designer's aim is to exclude all possible attacks and loopholes
- The design of secure systems remains an inherently difficult task
- Security is about avoiding disasters, minimizing mishaps and assuming the worst
- Basic technique - construct a list of threats; employ auditing methods; balance costs against threats

Just What Can Go Wrong?

- **Interfaces are exposed** - by their very nature, interfaces are "open", so an attacker can send messages to the interface, too
- **Networks are inherently insecure** - message sources and sinks can be falsified; host addresses can be "spoofed"
- **Anyone can be an attacker** - PCs are cheap, and getting more powerful

What Can Be Done?

- *Limit the lifetime and scope of secrets* - passwords and shared secret-keys need to be time-limited and their sharing needs to be restricted
- *Algorithms and code need to be available to everyone (including the bad guys)* - secrets are hard to keep, as the more widely a secret is distributed, the harder it is to keep it -- so don't even try!
- *Minimize the trusted base* - all the hardware and software upon which your security depends must be trusted, so the "trusted base" must be kept small

The Saviour: Cryptography!

- Encryption is the process of encoding a message in such a way as to hide its contents
- Security is based on the use of secrets called "keys"
- A cryptographic key is a parameter used in an encryption algorithm in such a way that the encryption cannot be reversed without having a knowledge of the key

Two Main Types of Encryption Technology

- **Shared Secret Keys** - the sender and the recipient share the same key that is itself used to encrypt/decrypt -- this is called *Conventional Symmetric Key Encryption*
- **Public/Private Keys** - two separate (but related) keys are used to encrypt/decrypt a message -- this is called *Public-Key Encryption* or *Asymmetric Key Encryption* (a message encrypted with a public-key can only be decrypted with its corresponding private-key, and vice-versa)

Encryption Key Point

Public-Key encryption algorithms typically require 100 to 1000 times as much processing power as secret-key (symmetric) algorithms, however, there are many situations where their convenience outweighs this apparent disadvantage

Cryptographic Notation

K_A	Alice's secret key
K_B	Bob's secret key
K_{AB}	Secret key shared between Alice and Bob
K_{Apriv}	Alice's private key (known only to Alice)
K_{Apub}	Alice's public key (published by Alice for all to read)
$\{M\}_K$	Message M encrypted with key K
$[M]_K$	Message M signed with key K

Uses of Cryptography

- **Security** - the secrecy of the encrypted message is maintained for as long as the decryption key is not compromised
- **Integrity** - the integrity of the message is maintained assuming the encryption process includes some redundant information that can be used as a checksum
- **Authentication** – a check can be performed to ensure that the message originator is who they say they are

Alice Communicates with Bob

- Alice and Bob share a secret-key
- Alice uses the key and an encryption algorithm to encrypt the message for Bob
- Bob uses the key and a decryption algorithm to decrypt the message from Alice
- How can Alice send the secret-key to Bob securely (especially over an insecure channel)?
- How does Bob know that the message he got is in fact from Alice and not a replay of an earlier message (from Alice) that is now originating from Mallory?

Cryptography and Authentication

- Cryptography supports mechanisms for authenticating communication between pairs of principals
- A successful decryption authenticates the decrypted message as coming from a particular sender

Hybrid Cryptographic Protocols

- Symmetric Key Cryptography is fast, but requires a shared secret-key
- Asymmetric Key Cryptography is slow, but does not require the prior distribution of a secret-key
- Hybrid cryptographic protocols combine *the best of both worlds*
- A shared secret-key is generated, then sent from the Alice to Bob using public-key cryptographic techniques
- Further communication employs symmetric-key technologies using the time-limited, shared secret-key (known as a "session-key")

The Man-In-The-Middle Attack

- Alice acquires Bob's public-key from a "trusted" key distribution service
- Mallory intercepts Alice's request, and responds with his own public-key, claiming that it is Bob's
- All subsequent messages from Alice can then be intercepted by Mallory!
- This can only be guarded against by ensuring that the "trusted" key distribution service signs any keys it distributes with its own private-key

Digital Signatures

- Rationale - to verify to a third party that a message or a document is an unaltered copy of one produced by the signer
- This uses a secure digest function (based on one-way hash functions)
- Similar to a checksum function, with the characteristic that no two messages are likely to produce a similar digest
- Signatures are typically generated with a sender's private-key, then checked by the receiver using the sender's public-key

How Digital Signatures Work

- Alice computes a fixed-length digest of the document using Digest(M)
- Alice encrypts the digest with her private-key, appends it to the original message, giving M, $\{Digest(M)\}K_{A_{priv}}$ (the signed message)
- Bob receives the signed message, extracts M and computes Digest(M)
- Bob decrypts $\{Digest(M)\}K_{A_{priv}}$ using Alice's public-key, $K_{A_{pub}}$, and compares the result with his calculated Digest(M)
- A match confirms that the signature is valid

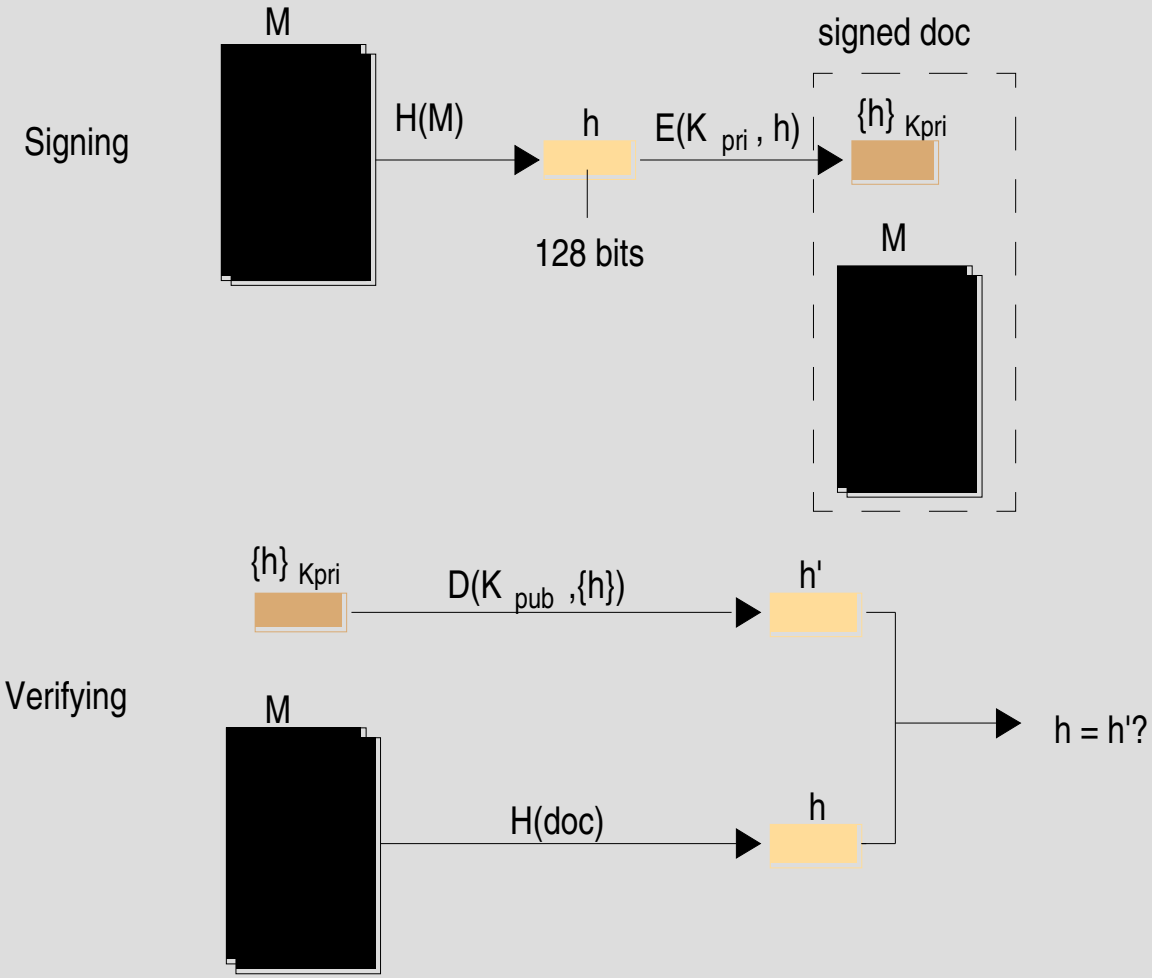
Is the Public-Key Valid?

- How does Bob know that Alice's public-key in his possession does in fact belong to Alice and not Mallory?
- The public-key is included in a *Certificate*
- Certificates are essentially a statement of fact signed (using digital signature technologies) by a **Certificate Authority** (CA) whom we are assumed to trust
- If the CA tells Bob that a public-key is valid, then Bob agrees to trust that information
- The CA signs certificates that it issues with it's private-key, and Bob authenticates this with his copy of the CA's public-key

Recursive Problem of Authenticity

- How can Bob be sure that he has the correct public-key for the CA and not one from Mallory?
- Bob can't, but Bob has to trust someone sometimes
- Bob has to acquire the public-key of the CA in a manner within which he can have confidence

Digital Signatures with Public Keys



The Birthday Attack

- Alice prepares two versions of a contract, M and M', for Bob
- M is favourable to Bob, M' is not
- Alice makes several subtly different versions of M and M' that are indistinguishable from one another, each time comparing the hash of M and M' to see if they match
- If Alice gets a match, she proceeds, otherwise, she keeps subtly changing M and M' and computing hashes until a match occurs
- When M and M' compute the same hash value, Alice gives the favourable document to Bob which he then accepts and signs with his private-key
- When Bob returns the signed document, Alice substitutes the favourable M for the unfavourable M'

How Likely is the Birthday Attack?

- A hash-size of 64 bits requires 2^{32} versions of M and M' (4294967296), which isn't very many (considering the speed of modern desktop computers)
- A hash-size of 128 bits should be considered the smallest size to guard against this type of attack
- *Where does the name come from?*
- Given a set of *N* people, how many people do we need before we have an even chance of finding a pair of people with the same birthday?
- Turns out it is ONLY 23 people! However, a set of 253 is required to try and find a person with a birthday on any given day

Popular Technologies

- **RSA** – Rivest, Shamir and Adelman -- now RSA Security Inc. -- produces a number of highly regarded cryptographic technologies
- **TEA** - Tiny Encryption Algorithm, developed at Cambridge University
- **DES** - Data Encryption Standard, developed by IBM for the National Bureau of Standards, initially considered to be very secure, but with a 56 bit key it is now no longer safe to depend on DES -- cracked in 1997 (in less than 3 days) using brute-force techniques
- **IDEA** - International Data Encryption Algorithm, developed as a replacement for DES, uses a 128 bit key and operates at about three times the speed of DES -- it has no known weaknesses

More Encryption Technologies

- **RC4** - developed by Ronald Rivest, easy to implement, supports large keys -- however, a flaw/weakness in how RC4 is used caused problems for early adopters of WiFi 802.11 networks
- **AES** - Advanced Encryption Standard, developed for NIST (National Institute for Standards and Technology) by Daemen/Rijmen -- also known as "Rijndael", very widely implemented and popular
- **Triple DES (3DES)** - a more secure version of DES -- E(D(E(M)K1)K2)K3

Digital Signature Technologies

- **MD5** - developed by Ronald Rivest, produces a 128-bit digest and is one of the most efficient algorithms currently in use
- **SHA-1** - developed for NIST, produces a 160 bit digest and is based on Rivest's MD4 (which -- not surprisingly -- is very similar to MD5)
- SHA-1 is slower than MD5, but can be used to produce longer digests if required, offering better protection against birthday attacks
- SHA-1's predecessors have been shown to be vulnerable and a replacement has been announced with availability from 2010 onwards

Certificate Standards

- **X.509**, developed by CCITT, is the most widely used certificate format, and is part of the X.500 suite of standards
- X.509 is used with TLS (the secure sockets technology)
- Certificate Authorities store X.509 certificates for "customers" -- both VeriSign and CREN are examples of these types of organisations

Cryptographic Performance

	Key size/hash size (bits)	*Extrapolated speed (kbytes/sec.)*	*PRB optimized (kbytes/s)*
TEA	128	700	-
DES	56	350	7746
Triple-DES	112	120	2842
IDEA	128	700	4469
RSA	512	7	-
RSA	2048	1	-
MD5	128	1740	62425
SHA	160	750	25162

Cryptography and Politics

- The emergence of cryptographic software was strongly resisted by the US government, especially the NSA and FBI ...
- Cryptographic software was classified as a munition in the United States and was subject to very stringent export restrictions
- Cryptographic technologies for export could not have a key size greater than 40 bits!
- In 2000, the law was changed in the US -- keys can now be exported up to 64 bits for encryption and up to 1024 bits for signing (message digests)
- As can be expected, these US-based laws only hurt US software exports

Security Case Studies

- The Needham-Schroeder Authentication Protocol
- Kerberos
- TLS
- 802.11 (and what can go wrong)

The Needham-Schroeder Authentication Protocol

- The authentication protocols, devised by Needham-Schroeder in 1978, are at the heart of many security techniques
- Developed in response for the need for a secure means to manage keys and passwords within a networked setting
- Proposed a solution to authentication and key distribution based on an "authentication server" that supplies secret-keys to clients

How Needham-Schroeder Works

- The authentication server (AS) maintains a table containing a name and a secret-key for each of the principals known to the system
- The protocol is based on the generation and transmission of tickets by the AS
- A "nonce" is an integer value that is added to a message to demonstrate its freshness
- Nonces are used only once and are generated on demand

The Needham-Schroeder Protocol in Action

Header	*Message*	*Notes*
1. A->S:	A, B, N_A	A requests S to supply a key for communication with B.
2. S->A:	$\{N_A, B, K_{AB}, \{K_{AB}, A\}_{KB}\}_{KA}$	S returns a message encrypted in A's secret key, containing a newly generated key K_{AB} and a 'ticket' encrypted in B's secret key. The nonce N_A demonstrates that the message was sent in response to the preceding one. A believes that S sent the message because only S knows A's secret key.
3. A->B:	$\{K_{AB}, A\}_{KB}$	A sends the 'ticket' to B.
4. B->A:	$\{N_B\}_{KAB}$	B decrypts the ticket and uses the new key K_{AB} to encrypt another nonce N_B.
5. A->B:	$\{N_B - 1\}_{KAB}$	A demonstrates to B that it was the sender of the previous message by returning an agreed transformation of N_B.

Success with Needham-Schroeder

- When completed, both Alice and Bob can be sure that any message encrypted in K_{AB} comes from the other
- Any message sent using K_{AB} can be understood by the other (or by the AS)
- This is because the messages sent containing K_{AB} were encrypted in Alice's or Bob's secret-key

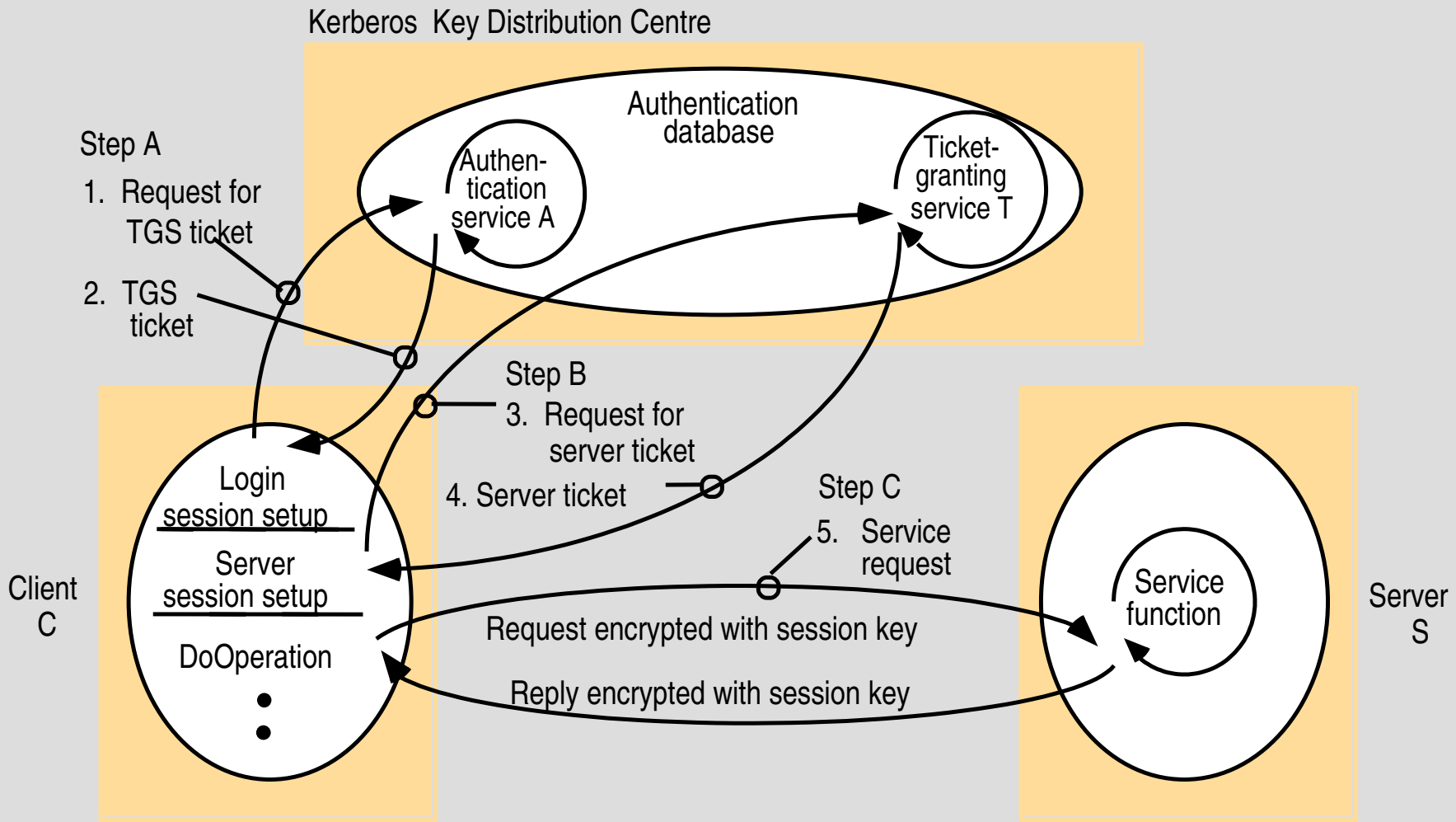
The Needham-Schroeder Weakness

- Bob has no reason to believe that message 3 is fresh, that is, when Alice sends her ticket to Bob
- If the ticket is "captured", it can be reused later to impersonate Alice and trick Bob
- Needham-Schroeder just didn't consider this as a possible threat
- The protocol can be protected by adding a nonce that timeouts to the ticket (this is what happens in Kerberos)

Kerberos

- Kerberos version 5 is on the Internet standards track, and is now used by many companies and universities
- It is included in many versions of Linux, the OSF Distributed Computing Environment (DCE) and Microsoft Windows 2000
- Extensions have been proposed to incorporate the use of public-key certificates for the initial authentication of principals

Kerberos's System Architecture



Kerberos Security Objects

- **Ticket** - a token issued to the client by the TGS for presentation to the target server -- tickets include an expiry time and a session-key
- **Authentication** - the ticket proves the identity of the client and can be used only once
- **Session Key** - used to encrypt communication with those servers that demand it

Kerberos Details

- Most tickets are granted to clients with a lifetime of several hours so that they can be used for interaction with a particular server until they expire
- The Kerberos Server is referred to as a "Key Distribution Centre" (KDC)
- The KDC contains the Authentication Server (AS) and the Ticket Granting Service (TGS)
- On login, uses are authenticated by the AS, supplied with a ticket granting ticket and a session-key for communicating with the TGS
- The Needham-Schroeder protocol is followed quite closely in Kerberos

The Application of Kerberos

- Developed as part of Project Athena at MIT, providing services to more than 5000 users at MIT
- The trustworthiness of clients and servers cannot be assumed (other than the Kerberos server, that is)
- Kerberos provides virtually all the security in the Athena system
- Clients and servers within the environment have to be extended to support the use of the TGS
- Users' passwords are known to the user and to the Kerberos AS
- Services have secret-keys that are known only to Kerberos and to the servers that provide the service

What Happens at Login?

- The login program sends the user's name to the Kerberos AS, which replies with a session-key and a nonce encrypted in the user's password, as well as a ticket for the TGS
- The login program decrypts the session-key and the nonce using the password that the user enters
- If all is OK, the ticket now authenticates the user and the password is no longer needed
- Note that the password is never transferred across the network

Accessing Services with Kerberos

- When access to a new service is required, a ticket for the service is requested from the TGS
- The ticket can then be sent to the server hosting the service to allow access to the required functionality
- The service uses a time-limited session-key (which is included with the ticket) to communicate with the user application

Implementation of Kerberos

- The Kerberos server is assumed to run on a secure machine
- (Remarkably) DES encryption is used throughout Kerberos, but is implemented as a separate module that can be replaced with "stronger" technology
- Kerberos is scalable, supporting the notion of domains (which is calls "realms")
- Realms can cross-authenticate, in that principals can authenticate themselves to servers in other realms through their local TGS
- The lifetime of TGS granted tickets needs to be long enough to avoid re-authentication when they expire (resulting in rejection)
- In practice, ticket lifetimes of 12 hours are typically used

Critiques of Kerberos

- It uses DES!
- Some of the synchronization protocols used in Kerberos have been shown to have potential vulnerabilities (improvements do not rely on synchronized clocks, and this can help)
- To work within a Kerberos environment, applications (both client-parts and server-parts) need to be "kerberised" (which may not always be possible nor practical)

Secure Sockets and Transactions

- SSL (secure sockets layer) was developed by Netscape in 1996
- An extended version of SSL is now an Internet standard (RFC 2246) and is called TLS (transport layer security)
- SSL and TLS are (for all intents and purposes) the same thing
- TLS is now the de-facto standard secure-channel technology on the Internet (e.g., HTTPS)

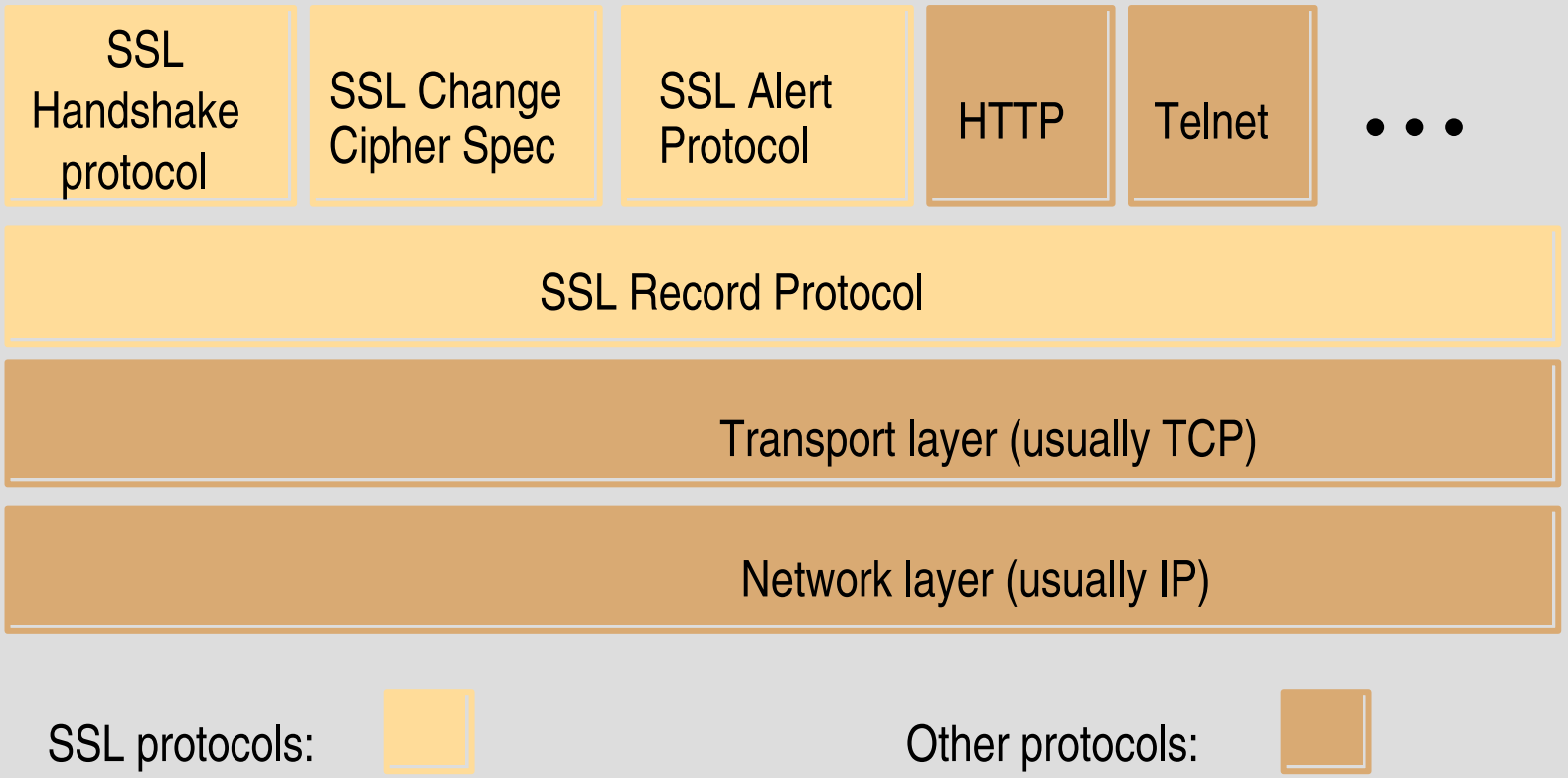
TLS Negotiation

- We cannot assume that all clients and servers will operate the same software that uses a particular encryption technology
- During the initial handshake, the encryption algorithms and authentication technologies are negotiated
- If agreement cannot be reached, the connection will fail

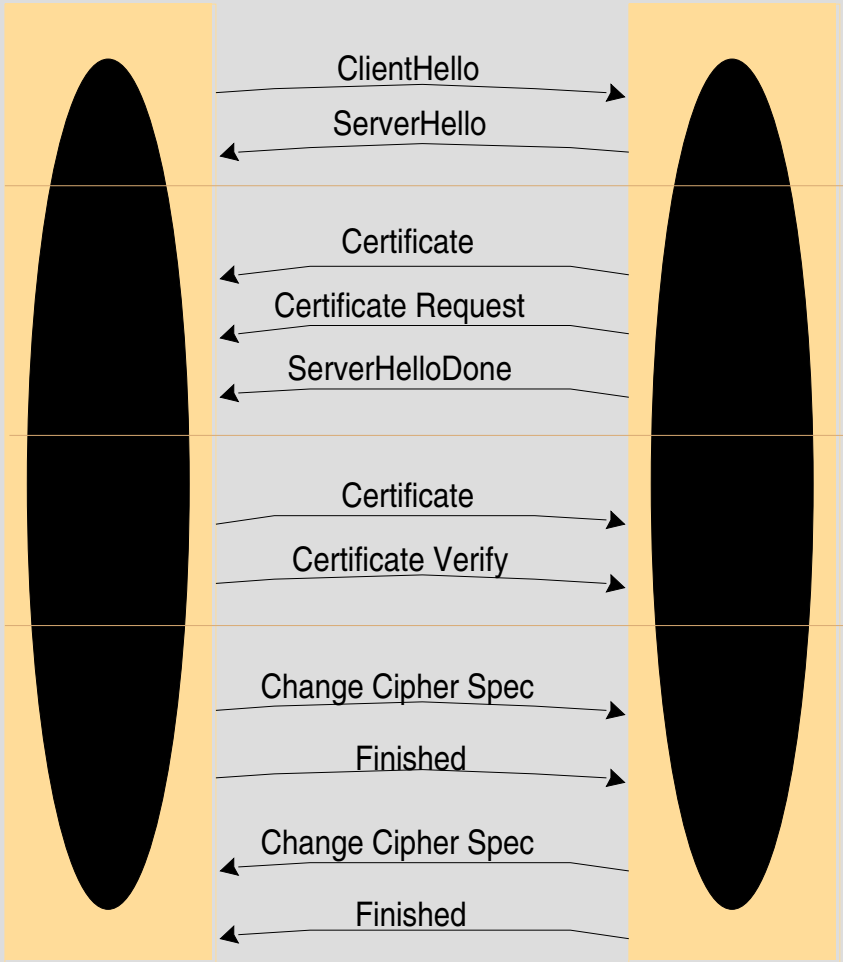
How Communication Occurs

- Clear-text messages start the communication session
- Public-Key technologies are then used to establish a session-key
- The session-key is then used by conventional symmetric technologies for the remainder of the session
- Each switch is optional and includes negotiation

The TLS Protocol Stack



The TLS Handshake Protocol



Establish protocol version, session ID, cipher suite, compression method, exchange random values

Optionally send server certificate and request client certificate

Send client certificate response if requested

Change cipher suite and finish handshake

TLS Handshake Configuration Options (Cipher Suite)

Component	*Description*	*Example*
Key exchange method	the method to be used for exchange of a session key	RSA with public-key certificates
Cipher for data transfer	the block or stream cipher to be used for data	IDEA
Message digest function	for creating message authentication codes (MACs)	SHA

The TLS Record Protocol

Application data

Fragment/combine

Record protocol units

Compress

Compressed units

Hash

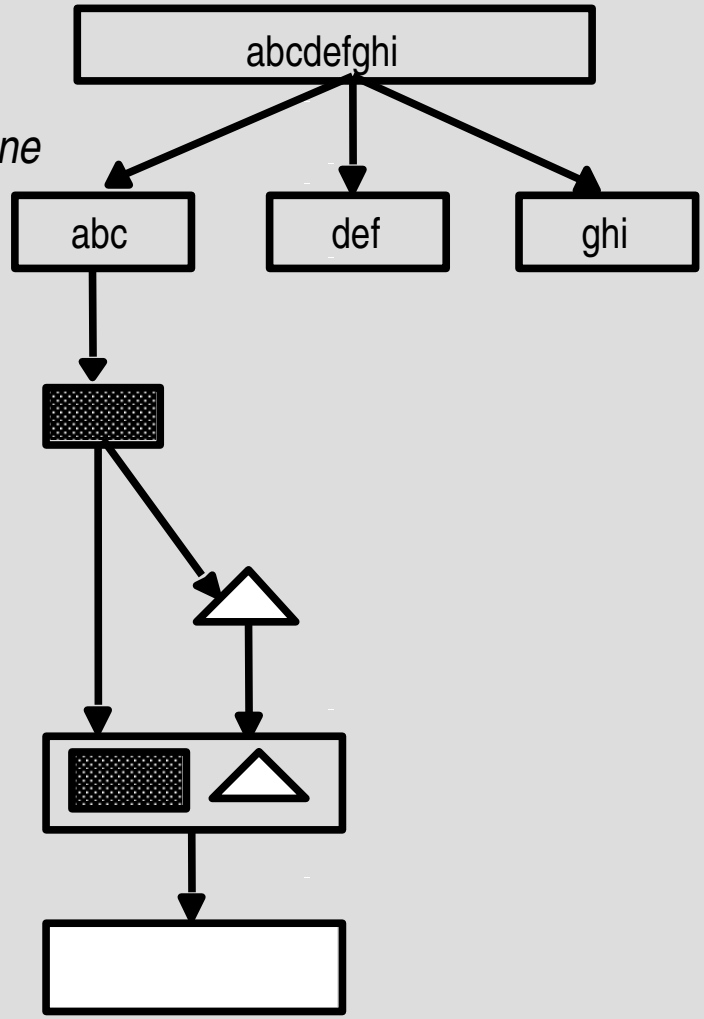
MAC

Encrypt

Encrypted

Transmit

TCP packet



TLS Summary

- TLS is a practical (and popular) implementation of a hybrid encryption scheme with authentication and key exchange based on public-keys
- Due to cipher negotiation, it does not depend on the availability of any particular algorithm
- It does not need a secure channel in order to establish a shared session-key
- Only the public-key certificates need to be recognize by both parties

IEEE 802.11 WiFi

- First released in 1999 and widely implemented in base stations, laptops and portables
- Design found to be "security weak" and inadequate for a modern networked environment

Wireless and WEP

- An device fitted with a transmitter/receiver can eavesdrop or masquerade on a wireless network
- Wired Equivalent Privacy (WEP) was initially designed to counteract these threats
- WEP Access Control - a challenge-response protocol, only base stations with the correct shared-key can connect, with a single key shared between the single base station and any number of devices
- WEP Privacy and Integrity - encryption based on RC4, with the same shared-key (40, 64 or 128 bits) used for access being used for encryption and a checksum used to ensure integrity

Deficiencies and Weaknesses, #1

- Sharing a single key with all users is a weak design!
- The key is liable to be transmitted to new users using unsecured channels
- A single careless or malicious user can compromise the security of the entire network, and go undetected
- **Solution**: use a public-key based protocol for negotiating individual keys

Deficiencies and Weaknesses, #2

- Base stations are never authenticated!
- An attacker with a compromised key can spoof a base station on the network, eavesdrop, insert or tamper traffic on the network
- **Solution**: Base stations should supply a certificate that can be authenticated using public-key technologies

Deficiencies and Weaknesses, #3

- The use of a stream cipher was inappropriate!
- The repetitions introduced in the way the stream cipher is used (especially as it relates to key values) in 802.11 WEP can allow a receiver to capture and decrypt large portions of traffic
- **Solution**: Negotiate a new key before the repetitions occur, effectively negating this problem

Deficiencies and Weaknesses, #4

- The 40 and 64 bit key lengths are too short!
- The 40 and 64 bit key lengths were included in the standard to allow US suppliers to export their products
- Brute-force attack can crack these key lengths
- **Solution**: use 128 bit key Lengths only

Deficiencies and Weaknesses, #5

- The RC4 stream cipher had been shown to exhibit weaknesses when used in certain ways!
- Even with 128 bit keys, RC4 was demonstrated to be weak (in certain circumstances)
- **Solution**: allow WEP to negotiate ciphers

Deficiencies and Weaknesses, #6

- User don't often deploy protection!
- The WEP technologies were shipped with WEP switched off and with inadequate documentation as the dangers of deploying the product without switching the security features on
- **Solution**: vendors must provide better settings and documentation (adding filtering on IP/MAC address is NOT enough)

Fixing WEP

- Updated standard released in 2004
- All of the above weaknesses (all six) were addressed
- Mutual authentication is now supported
- Dynamically negotiated pair-wise keys now works
- AES is now used for encryption

Distributed Systems Security - Summary

- Threats to the security of distributed systems are pervasive
- Be afraid, be very afraid!