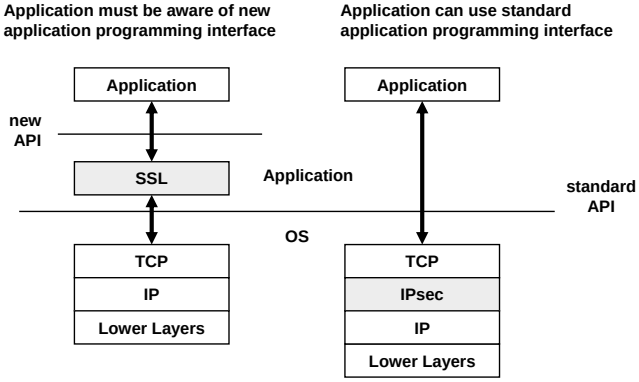
SSL, PGP, Kerberos

Secure Socket Layer (Web Security),
Pretty Good Privacy (Email Security) and Authentication

Agenda

- **SSL**
- **PGP**
- **Kerberos**

SSL versus IPsec



SSL General Aspects 1

- **Runs on top TCP**
  - TCP included in OS
    - timeout and retransmitting lost data done by TCP
    - that makes SSL a little simpler
  - therefore OS must not be changed
- **New socket layer interface**
  - SSL instead TCP
  - application must be adapted
- **Originally developed**
  - by Netscape to protect WEB transactions between client and server
    - version 3.0 or 3.1 is currently implemented in Web browsers

SSL General Aspects 2

- **Web transaction security is based on SSL**
  - HTTPS means standard HTTP over SSL
  - TCP port number 443 used
  - HREF = https://…
  - SSL protocols are activated in browser and server
- **Although SSL is not restricted**
  - for usage in Web Browsers
    - note: SSL can provide a secure connection to any application
- **Web browsers are SSL´s the most common application**

SSL General Aspects 3

- **SSL idea was taken by IETF**
  - and further developed -> TLS
- **Transport Layer Security**
  - RFC 2246 (TLS Protocol)
  - RFC 2478 (Secure SMTP)
  - RFC 2595 (IMAP, POP3)
  - RFC 2712 (Kerberos Ciphersuite for TLS)
  - RFC 2817 (HTTP 1.1)
  - RFC 3268 (AES Ciphersuite for TLS)
  - RFC 3546 (TLS Service Extensions)
- **TLSv1.0 and SSLv3.0 are not interoperable**
  - TLS uses DH and DSS, SSL uses RSA
  - TLSv1.0 = SSLv3.1

Institute of Computer Technology - Vienna University of Technology

L96 - SSL, PGP, Kerberos

What SSL does? 1

- **Establishes a secure connection in 4 phases**
  - parameter negotiation between client and server
    - session key generation method, authentication method and encryption algorithms to be used for data transfer phase
  - mutual authentication of client and server
    - note: client authentication may be optional
  - session key building and activation of cipher suite
    - integrity key and encryption key
- **Secure connection can then be used for the actual data transfer**
  - protected by session keys build during establishment

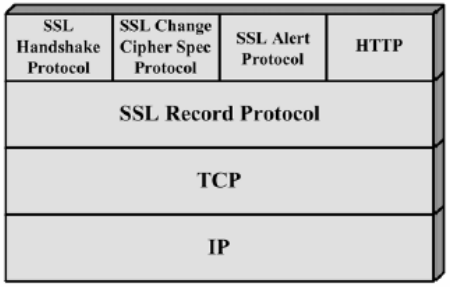
What SSL does? 2

- **Data transfer protection mechanism**
  - integrity of data exchange by HMAC
    - keyed-SHA-1
    - keyed-MD5
  - confidentiality (privacy) of data exchange by encryption
    - DES-40
    - DES-CBC,
    - 3DES-EDE, 3DES-CBC,
    - RC4-40, RC4-128
- **SSL Session-ID allows**
  - to differentiate between a new session and a session to be resumed by caching session-ID´s
    - usually not more than 24 hours lifetime

What SSL does? 3

- **Four methods for session keys generation**
  - RSA
    - shared secret S encrypted with public-key of partner
  - Fixed DH key exchange
    - fixed public-DH value contained in DC (certificate)
    - session keys are based on the same base parameters
  - Ephemeral DH key exchange (DHE)
    - actual public-DH value signed with private-key of sender
    - best protection because every session will have a completely different set of generated keys
  - Anonymous DH key exchange
    - basic DH key exchange without signatures and certificates
    - no protection against man-in-the-middle-attack

SSL Protocols 1



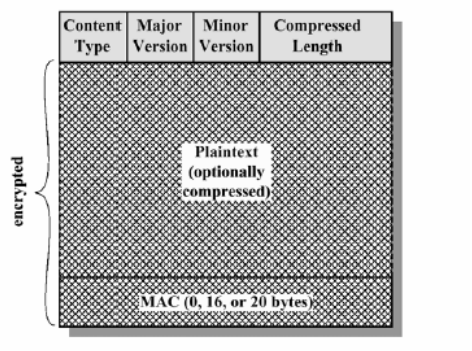
SSL Protocols 2

- **SSL Record Protocol**
  - using the reliable octet stream service provided by TCP
  - partitions these octet stream into records
    - maximum 16384 bytes per record
  - every record starts with a header (type/length) and is cryptographic protected
    - integrity
    - privacy
  - four record types (content type field)
    - handshake message (for connection setup and resume)
    - change cipher spec (for activating new security parameter)
    - alert (for error messages or notification of connection closure)
    - user data

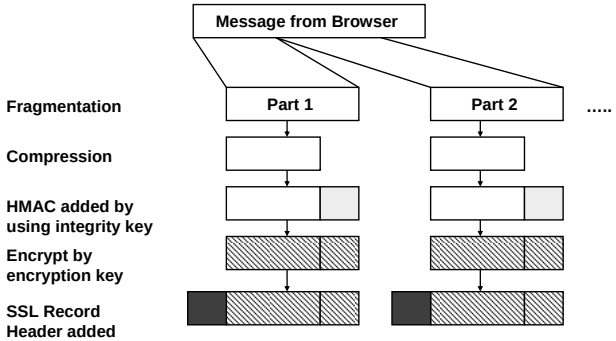
SSL Protocols 3

- **SSL Record Protocol**
  - sub protocol for three other protocols and application data transfer
- **SSL Handshake Protocol**
  - for authentication and parameter negotiation
    - methods and keys
- **SSL Change Cipherspecification Protocol**
  - for signalling of a change of the cipher suite to be used
- **SSL Alert Protocol**
  - for error signalling

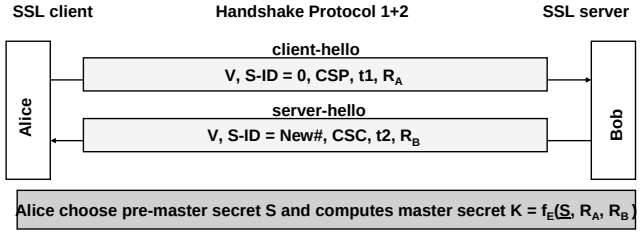
SSL Record Header



SSL in Action



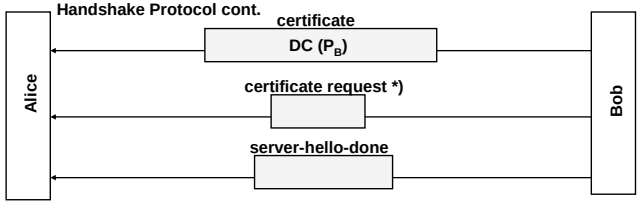
Protocol Procedures Phase 1 - All Modes



Alice choose pre-master secret S and computes master secret K = $f_E(\underline{S}, R_A, R_B)$

V … SSL version number
S-ID … Session-Identifier (0 for new session, if unequal 0 then resuming a former established session -> phase 2 and 3 are not used
CSP … cipher suite proposal of client
CSC … cipher suite chosen by server
R_A, R_B … random, number
t1, t2 … timestamps

Protocol Procedures Phase 2 - RSA

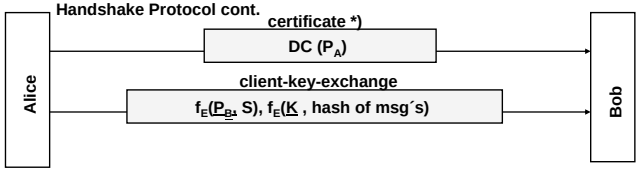


Alice verifies that she is talking to Bob by proofing the certificate

DC = P_B + $f_E(\underline{S}_{Cert}, P_B)$ … Digital Certificate of Bob's Public Key P_B and chain of certificates to reach a well-known root-CA (certification authority) if necessary

***) message only present if server wants client authentication**

Protocol Procedures Phase 3 - RSA

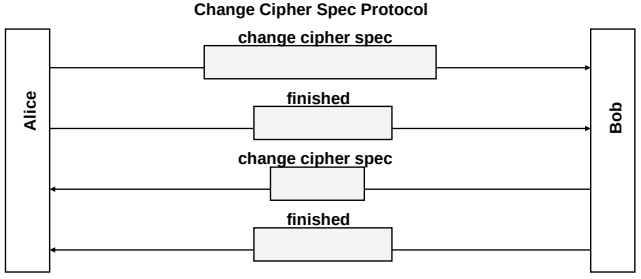


Alice sends S (pre-master secret) encrypted with Bob's public key.
Hash of msg's encrypted with master-secret K both to prove that she knows the key and that tampering of the handshake message would be detected.

Bob decrypts S, computes master secret K = $f_E(S, R_A, R_B)$ and verifies that Alice really knows the secrets and communication was not tampered

*) message only present if server wants client authentication

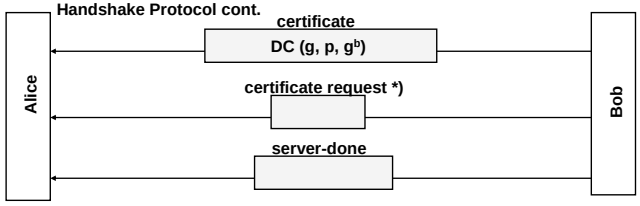
Protocol Procedures Phase 4 - All Modes



Alice and Bob now compute the necessary session keys for integrity and privacy aspects based on the exchanged master secret K and R_A / R_B

change cipher spec messages activate these keys for usage by the SSL record protocol in order to encrypt data and to achieve data integrity

Protocol Procedures Phase 2 - Fixed DH

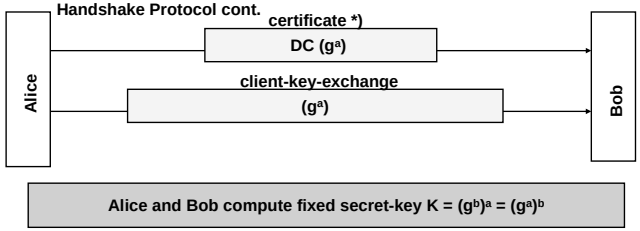


Alice verifies that she is talking to Bob by proofing the certificate

DC = g, p, g^b + $f_E(S_{Cert}$, g, p, g^b) … Digital Certificate of Bob's Public DH values g^b and g, p and chain of certificates to reach a well-known root-CA if necessary

*) message only present if server wants client authentication

Protocol Procedures Phase 3 - Fixed DH

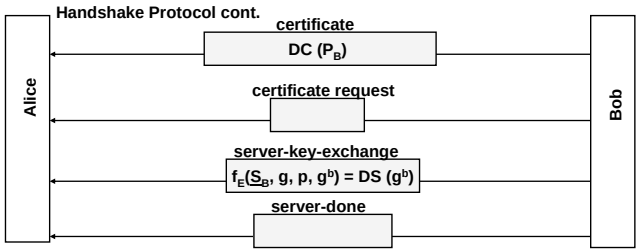


Alice and Bob compute fixed secret-key K = $(g^b)^a = (g^a)^b$

*) message only present if server wants client authentication, client-key-exchange message is empty in this case

DC = g^a + $f_E(S_{Cert}$, g^a) … Digital Certificate of Alice's Public DH value g^a and chain of certificates to reach a well-known root-CA if necessary

Protocol Procedures Phase 2 - Eph. DH

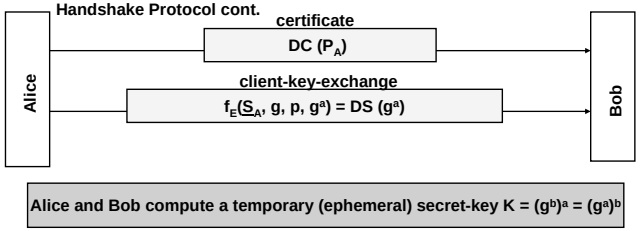


$DC = P_B + f_E(S_{Cert}, P_B)$ … Digital Certificate of Bob's Public Key P_B and chain of certificates to reach a well-known root-CA (certification authority) if necessary

S_B = Bob's private-key

DS = Digital Signature -> (g, p and Public-DH value g^b) signed with Bob's private-key S_B

Protocol Procedures Phase 3 - Eph. DH

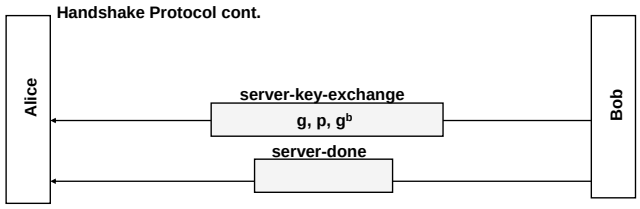


$DC = P_A + f_E(S_{Cert}, P_A)$ … Digital Certificate of Alice's Public Key P_A and chain of certificates to reach a well-known root-CA (certification authority) if necessary

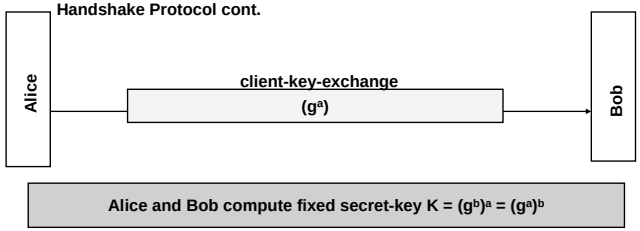
S_A = Alice's private-key

DS = Digital Signature -> (g, p and Public-DH value g^b) signed with Bob´s private-key S_B

Protocol Procedures Phase 2 – Anony. DH



Protocol Procedures Phase 3 - Anony. DH



SSL in Web Browsers

- **Preconfigured with public-keys of various "trusted" organisations (root CA)**
  - e.g. Verisign
- **User may modify this list**
  - adding, deleting
- **Server will sent a certificate**
  - which is checked against the list and verified if there is a matching entry
- **If no match or no verification then Pop-up window will appear**
  - user should say what to do either to import to the list of trusted root CA´s or cancel

Agenda

- **SSL**
- **PGP**
- **Kerberos**

Pretty Good Privacy (PGP)

- **PGP is a complete E-mail security package providing**
  - privacy, authentication, digital signature, compression
  - in an easy to use form
- **Designed by Phil Zimmermann**
  - roots in the 80´s
  - first release 1991
  - 1993 released for free private usage in the public domain
  - US government investigation against Phil on breaking the US export rules
  - patent problems (RSA and IDEA)

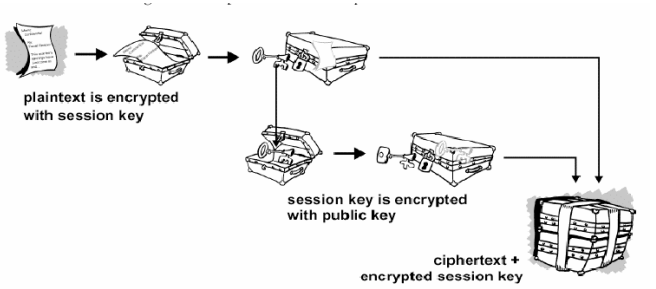
Pretty Good Privacy (PGP)

- **Because of these problems several versions of PGP exist today**
  - PGP classic (described in this module)
    - oldest and simplest version
  - Open PGP (RFC 2440)
  - GNU Privacy Guard (CPG)
    - Free Software Foundation
    - http://www.gnupg.org/
      - "GNU Handbuch zum Schutz der Privatsphäre"
    - revocation of public keys is possible
  - PGP product
    - company "PGP" is now owned by Network Associates
    - -> www.pgp.com, www.nai.com/default_pgp.asp
    - -> www.pgpi.com (Freeware)

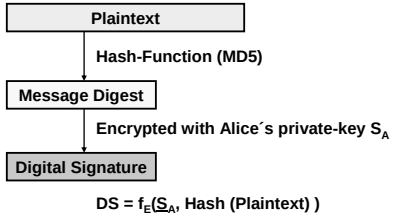
What Does PGP?

- Encryption of files using a pass-phrase as key
- Create public/private key pairs
- Provide compression
- Provide Radix-64 encoding for mail friendly delivery
- Send/receive encrypted email
- Compute digital signatures
- Manage a public-key database, including certificates
- Certify public-keys (for others)
  - Can use PGP Internet key servers

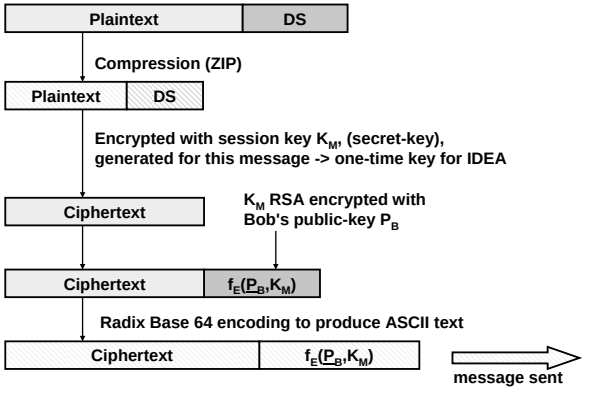
How PGP Privacy Works



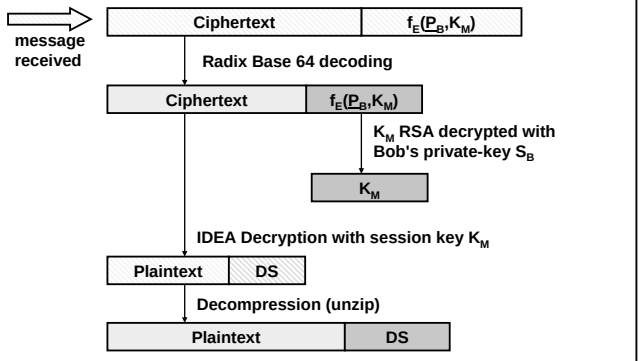
PGP Sender Side (Authentication+Integrity) 1



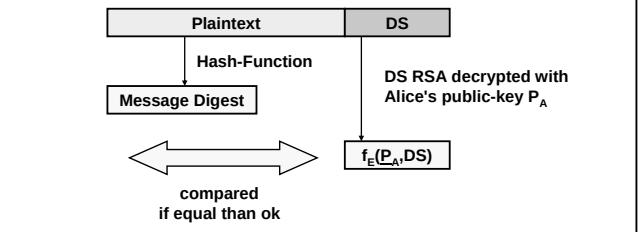
PGP Sender Side (Privacy) 2



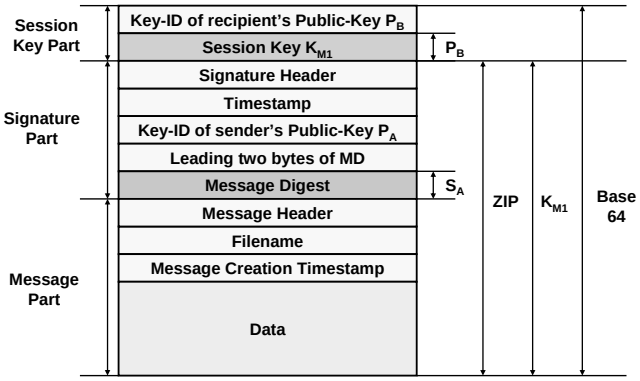
PGP Receiver Side 1



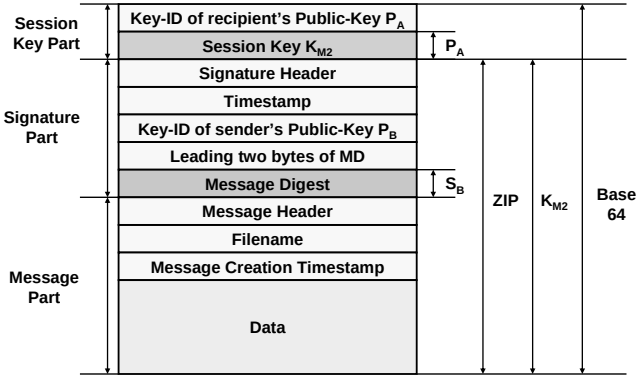
PGP Receiver Side (Identity + Integrity Check) 2



PGP Message Format (Alice->Bob)



PGP Message Format (Bob->Alice)



Performance / Security

- **RSA (asymmetric, slow) is used only for 256 bits**
  - encryption of 128-bit MD5 as signature
  - encryption of 128-bit IDEA-key as session-key
- **IDEA (symmetric, fast) is used**
  - for bulk encryption
- **PGP supports four RSA key lengths**
  - Casual (384 bits):
    - can be broken easily today
  - Commercial (512 bits):
    - breakable by three letter organizations
  - Military (1024 bits):
    - not breakable by anyone on earth
  - Alien (2048 bits):
    - not breakable by anyone on other planets, either

Management of Keys

- **After installing PGP on Alice's machine**
  - a RSA public/private key pair is generated
- **Storage of keys**
  - public-key is stored on a data structure called public-key ring referenced by User-ID (Alice) and Key-ID (least significant 64 bits of public-key)
  - private-key is stored on the private-key ring in encrypted form together with User-ID and copy of corresponding public-key
  - Alice is asked for a corresponding pass-phrase in order to get access to (to decrypt) her private-key
  - after the private-key is used it is immediately discarded from memory of the used machine

Private-Key Protection

- **Alice's pass-phrase**
  - is used to generate a 128-bit MD5 message digest which in turn is used as 128-bit IDEA key
- **Private-Key**
  - is encrypted by IDEA algorithm with key based on the pass-phrase and then stored on the private-key ring
- **Pass-phrase and IDEA key are then discarded**
  - to protect the private-key in case of breaking into Alice's computer
- **Whenever Alice wants to sign a message**
  - she must again specify the pass-phrase in order to IDEA decrypt the private-key

Public-Key Ring

- **Storage place for public-keys**
  - of all partners to which Alice wants to communicate using PGP
  - even her own public-key is stored here in order to be given to partners on request

Handling of Keys at the Receiver Side

- **Bob's storage place for private-keys**
  - is his private-key ring
- **If a message is received**
  - Bob must provide his pass-phrase to get access to his private-key
  - Bob's private-key is then used to decrypt the IDEA one-time session key
    - better would be the name message key because there is not anything like a session in PGP
- **After IDEA decryption**
  - Bob will retrieve Alice's public-key from his public-key ring and verifies the signature of the message

Public-Key Management

- **Originally**
  - decentralized, user-controlled approach
    - some call it an anarchy
    - against centralized PKI schemas
  - level of trust is introduced
    - each user decides which keys to trust
    - each user decides which users to trust
      - levels are none, partial and complete
  - public-keys of others may be signed with own private-key
    - signed public-keys (= certificate) from trusted users maybe again to be trusted
- **Today**
  - PGP versions are interoperable with PKI infrastructure
    - CA and X.509

How to get Public-Key Securely?

- **The problem is the man-in-the-middle attack**
- **Therefore**
  - physically get the key on floppy disk or cdrom
  - get and verify a key via telephone
    - authentication based on voice recognition and then dictation of the key over phone
  - get the key in an email
    - generate a fingerprint of the received key
    - call the partner and tell him to dictate the fingerprint over the phone, if the two fingerprints match, the key is certified
  - get the key signed by a trusted person
  - get the key from a key server and verify the fingerprint directly with the corresponding partner out-band
  - get the key signed from a trusted key server

Other Email Security Techniques

- **PEM (Privacy Enhanced Mail)**
  - developed in late 1980´s (RFC 1421-1424)
  - same topics covered as PGP
  - some differences
    - keys are certified by X.509 certificates issued by CA
    - rigid CA hierarchy starting at a single root
    - nobody want to support this single root (political problem)
  - at the end PEM approach collapsed finding no root

Other Email Security Techniques

- **S/MIME (Secure Multipurpose Internet Mail Extensions)**
  - next IETF approach but learning the lessons avoiding the rigid CA hierarchy of PEM
  - RFC 2632-2634 (obsoleted)
  - RFC 3850-3855 (actual)
  - trust anchors instead single root
  - user can have multiple so called trust anchors
  - PGP type certifications are possible but only in 1:1 relation

Agenda

- **SSL**
- **PGP**
- **Kerberos**

Introduction

- **Kerberos (old):**
  - is the watchdog of Hades, whose duty it was to guard the entrance against whom or what does not clearly appear; Kerberos is known to have had three heads
- **Kerberos (today):**
  - is an encryption-based security system that provides mutual authentication between the workstation users (clients) and the servers in a network environment in a secure way without having servers configured with tons of passwords (secrets)
  - is an authentication and authorization system
  - developed at the MIT for project Athena (1983)

Introduction

- **Kerberos (today): cont.**
  - version 4
    - symmetric cryptography (uses DES-CBC)
    - IP only
    - RFC 1411
  - version 5
    - symmetric cryptography (uses modified DES-CBC)
      - Plaintext Cipher Block Chaining (PCBC)
    - public-key cryptography as well
    - RFC 1510
    - ASN.1 syntax
  - used in many real systems
    - e.g. for Unix
    - e.g. for Windows NT, Windows 2000

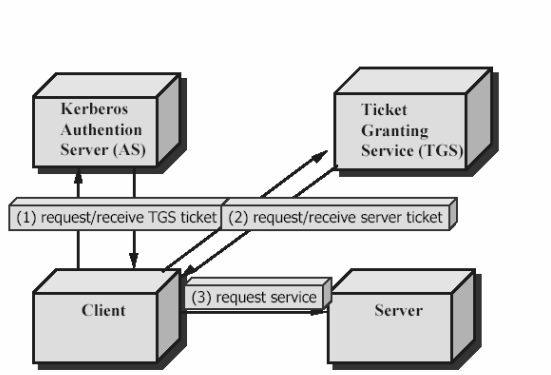
Requirements for Kerberos

- **Secure**
  - protect against eavesdropping and impersonation (need user authentication)
- **Reliable**
  - Kerberos must provide high degree of availability
- **Transparent**
  - minimal user interaction required for security
- **Scalable**
  - able to support large numbers of clients and servers in a distributed environment

Kerberos Structure

- **A distributed Trusted Third Party (TTP) authentication schema**
  - users trusted an arbitrator (Kerberos server is the trusted arbitrate; like a KDC)
  - assumes that normal servers are not trustworthy
  - of course Kerberos server must be specially secured
- **Two Kerberos server function involved**
  - Authentication Server (AS)
  - Ticket Granting Server (TGS)
- **Synchronized clocks**
  - AS, TGS, client (Alice) and server (Bob)

Kerberos Procedures Overview



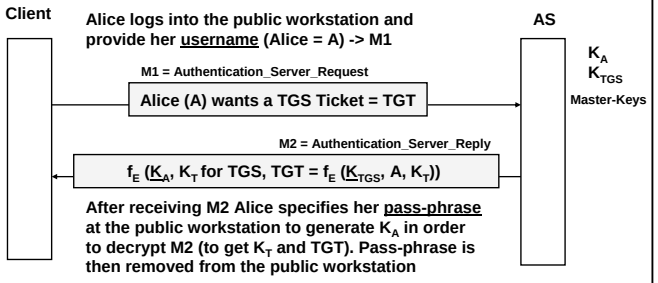
Kerberos Principles

- **Each user shares a long-term secret-key with the AS**
  - derived by hashing a user-supplied pass-phrase
  - users are clients and servers
    - e.g. Alice as client and Bob as server
- **Long-term secret-key**
  - pass-phrase is distributed (agreed) off-line
  - hashed pass-phrase is entered at start of each session
  - stored only very short on the client's workstation
  - not sent over the insecure network
  - pass-phrase is used for initial log-in of user to the client computer

Kerberos Principles

- **Authentication at the beginning of a network connection**
  - but not for the remainder of the session
- **The AS uses the long long-term secret-key**
  - to set up a short-term shared secret-key with the TGS
    - short-term means hours instead for days/months or years
- **The TGS generates**
  - shared session-keys between entities
- **Does not require client to enter password**
  - every time a service is requested service
- **Passwords are never sent in clear**

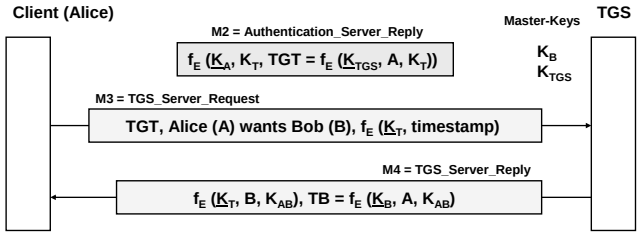
Kerberos Procedures Details 1



AS selects a short-term session-key K_T and sent it encrypted with shared secret-key K_A to Alice together with a so called Ticket Granting Ticket (TGT)

TGT-Ticket itself contains short-term session-key K_T, username A and Alice´s network address encrypted with TGS shared secret-key K_{TGS}

Kerberos Procedures Details 2

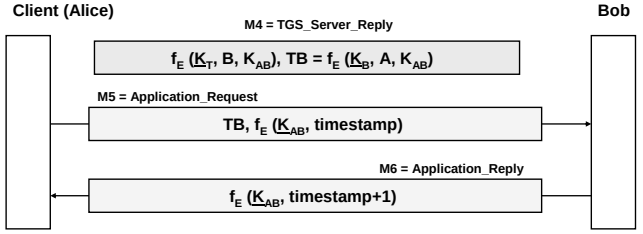


Alice provides TGT to TGS to proof that she really is Alice (only with her pass-phrase she was able to obtain TGT) and asks for a server ticket to Bob.

Timestamp with K_T encrypted for protecting against replay attack

TGS selects a session-key K_{AB} and sent it encrypted with short-term session-key K_T to Alice together with a server ticket TB for Bob.
TB contains lifetime of this ticket and network address of Alice

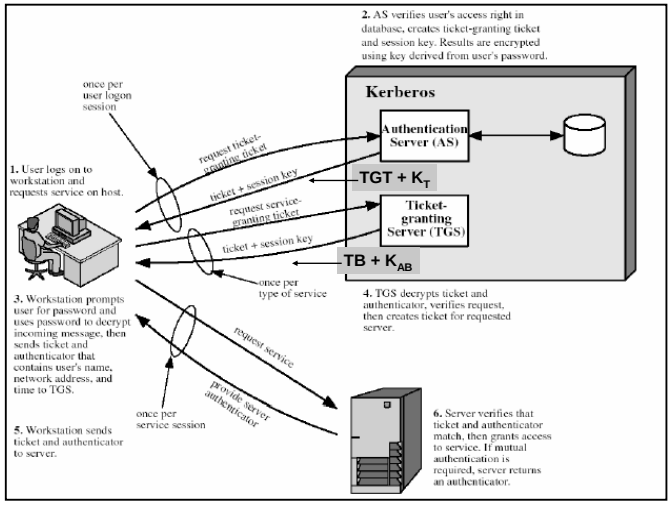
Kerberos Procedures Details 3



Alice provides TB to Bob to proof that she really is Alice (only with her pass-phrase she was able to get an TGT and TB) and asks access to server Bob.

Exchange of timestamp with K_{AB} encrypted for proofing that Alice is really talking to Bob.

Note: AS + TGS normally implemented in one machine -> KDC



Kerberos Pros

- **Attacks which Kerberos prevents:**
  - Eavesdropping
    - as all the data in the protocol is sent encrypted (or may be publicly known), any eavesdropper would not gain any information
  - Imposture
    - it is hard to imposture someone, the knowledge of the secret key is a proof of identity
  - Man-in-the-middle
    - only valid users can generate the needed output (especially to encrypt Alice's address)
  - Replay Attacks
    - due to the timestamps and the lifetime fields, it is impossible to re-send any ticket (hence receiving authentication as someone else)

Kerberos Cons

- **Kerberos Limitations:**
  - not effective against password guessing attacks
  - only protects S/W that's been modified to use it
  - requires a "trusted path" for password entry
  - does not provide authorization
  - not a host-to-host protocol
    - designed to authenticate a workstation end-user
    - bad for time sharing machines & diskless workstations
  - denial of service attacks not solved
  - old authenticators may be stored for detecting later replay, at least during the lifetime of the ticket
    - servers should store all tickets to prevent this, but can't always do so

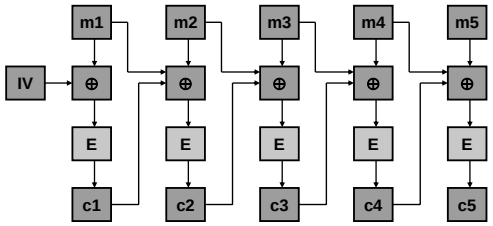
Kerberos Cons

- **Kerberos Limitations (cont.):**
  - authenticators rely upon synchronized and uncompromised clocks
    - if a host is compromised, the clock can be compromised and replay is easy
  - password guessing attacks may work
    - attackers could collect tickets and try it ...
  - relies upon trustworthy clients and servers
  - relies upon the security of the TGS and the Kerberos server
  - requires Kerberos server to work (single point of failure)

Kerberos Realms in Version 5

- **It is not scalable**
  - that the entire world will trust a single authentication server
- **Therefore multiple realms**
  - each with its own AS and TGS
- **In order to get a ticket for a server in a distant realm**
  - client asks his own TGS for a ticket accepted by the TGS in the distant realm
- **If the distant TGS has registered**
  - with the local TGS (in the same way local servers do) a valid ticket for the distant realm can be given to the client

DES - PCBC in Version 5



Encryption with Plaintext Cipher Block Chaining because DES-CBC alone cannot guarantee integrity of messages and Kerberos want to provide integrity assurance without depending on the application

