FIT5037: Network Security
Authentication Methods and AAA protocols

Faculty of Information Technology
Monash University

Copyright Notice

Lecture Topics:

- Symmetric key cryptography
- Asymmetric key cryptography
- Pseudorandom Number Generators and hash functions
- **Authentication Methods and AAA protocols**
- Security at Network layer
- Security at Network layer (continued)
- Security at Transport layer
- Security at Application layer
- Computer system security and malicious code
- Computer system vulnerabilities and penetration testing
- Intrusion detection
- Denial of Service Attacks and Countermeasures / Revision

Outline

- General Authentication Approaches
- Symmetric key method: Kerberos
- Brief overview of other related protocols
  - Lightweight Directory Access Protocol
  - Simple Authentication and Security Layer
- Network Access Server and AAA protocol requirements
- RADIUS and Diameter protocols
- Asymmetric key method: X.509 Certificates

MONASH University

Authentication Methods

- Use one of the following
  - SYK: Something You know
    - Username and Password
  - SYH: Something you have
    - Token
    - Digital Certificate
  - SYA/SYD: Something you are/do
    - biometric
- and a protocol to specify
  - message format, types and content
  - acceptable parameters and primitives
  - verification method
  - etc. (how the two entity communicate to prove the authenticity of one or both parties)

Two General Approaches

- Using Symmetric Key Encryption
  - One side proves to the other that it is in possession of a secret key
    - Client to server
    - key is derived from user/machine password
    - a challenge-response protocol is used
  - A Key Distribution Centre (KDC) acts as a Trusted Third Party (TTP)
    - KDC has long-term shared master keys with all participants
    - KDC generates session keys
- Using Public Key Encryption
  - Since public keys must be authentic a TTP is needed
  - A Certificate Authority is the TTP
    - participants prove their identities to CA
    - participants provide proof of possession of their private key
    - participants provide their public keys
    - CA signs the digital document containing all the relevant information
  - Any entity who trusts the CA can verify the validity of certificates issued by that CA using public key of CA

Symmetric key method: Kerberos

- Provides a centralised authentication server to authenticate users to servers and servers to users
  - allows users access to services distributed through network
  - without needing to trust all workstations
  - rather all principles trust a central authentication server
- Relies on conventional/Symmetric encryption
  - makes no use of public-key/Asymmetric encryption
- Encryption and Checksum Specifications for Kerberos Version 5 RFC 3961 (February 2005) lists DES, 3DES, RC4 and AES as ciphers
  - RFC 6649 and RFC 8429 deprecate DES, 3DES, and RC4 (and other weak algorithms)
- The first published report identified its requirements as:
  - **Security:** secure enough to prevent eavesdropping
  - **Reliability:** highly reliable to ensure the availability
  - **Transparency:** user should not be aware of authentication taking place
  - **Scalability:** capable of supporting large number of clients and servers

MONASH University

Kerberos Overview

- Employs an Authentication Server (AS)
  - maintains a database of principals (users, machines, etc.) and their secret keys
  - users initially authenticate with AS to identify self
  - AS provides an authentication credential (ticket granting ticket TGT) using a symmetric key primitive
  - AS stores all the passwords/secret keys of all principles (users, machines etc.)
- Employs a Ticket Granting server (TGS)
  - users subsequently request access to other services from TGS on basis of users' TGT
- In practice the same server provides both AS and TGS services

A Simple Authentication Dialogue

(1) **C** $\rightarrow$ **AS** : $ID_c||P_C||ID_V$

C = client

V = server

AS = authentication server

$\mathbf{ID_C}$ = identifier of user on C

$\mathbf{P_C}$ = password of user on C

$\mathbf{ID_V}$ = identifier of server V

C asks user for the password

AS checks that user supplied the right password

Message 2 of our Dialogue

(2) $\mathbf{AS \rightarrow C : Ticket}$

$\mathbf{Ticket = E(K_V, [ID_C||AD_C||ID_V])}$

AS = Authentication server

K_V = secret encryption key shared by AS and V

AD_C = network address of C

Ticket cannot be altered by C or an adversary

Message 3 of our Dialogue

(3) $C \rightarrow V : ID_C || Ticket$

Server V decrypts the ticket and checks various fields

AD_C in the ticket binds the ticket to the network address of C

However this authentication scheme has problems

Problems of the above Dialogue

Each time a user needs to access a different service he/she needs to enter their password - Read email several times - Print, mail, or file server + Assume that each ticket can be used only once (otherwise open to replay attacks) + Password sent in the clear

Authentication Dialogue (version II)

Introducing a Ticket Granding Server (TGS)...

Once per user logon session

(1) $C \rightarrow AS : ID_C || ID_{TGS}$

(2) $AS \rightarrow C : E(K_C, [Ticket_{TGS}])$

$Ticket_{TGS}$ is equal to $E(K_{TGS}, [ID_C || AD_C || ID_{TGS} || TS_1 || Lifetime_1]$

TGS $=$ Ticket-granting server

ID_{TGS} $=$ Identifier of the TGS

$Ticket_{TGS}$ $=$ Ticket-granting ticket or TGT

TS_1 $=$ timestamp

$Lifetime_1$ $=$ lifetime for the TGT

K_C $=$ key derived from user's password

Authentication Dialogue (version II) continues...

Once per type of service

(3) $C \rightarrow TGS : ID_C || ID_V || Ticket_{TGS}$

(4) $TGS \rightarrow C : Ticket_V$

$Ticket_V$ is equal to $E(K_V, [ID_C || AD_C || ID_V || TS_2 || Lifetime_2])$

K_V: key shared between V and TGS

Is called the service-granting ticket (SGT)

Authentication Dialogue (version II) continues . . .

Once per service session

(5) $C \rightarrow V : ID_C || Ticket_V$

C says to V : *I am* ID_C *and have a ticket from the TGS . Let me in.*

Seems secure, but.. There are problems

Problems .. again

1. Lifetime of the TGT
    - Short : user is repeatedly asked for their password
    - Long : open to replay attack
    - Oscar captures TGT and waits for the user to logoff
    - Sends message (3) with network address IDC (network address is easy to forge)
2. The lifetime of the SGT has the same problem

How to solve them ?

A network service (TGS or server) should be able to verify that

1. person using the ticket is the same as the person that the ticket was issued to
   - Remedy : use an authenticator
2. Server should also authenticate to user. Otherwise can setup a "fake" server

- Eg. A "fake" tuition payment server and capture the student's credit card
   - Remedy : use a challenge-response protocol

Kerberos v4 Authentication Dialogue

Authentication Service Exchange to obtain Ticket-Granting Ticket:

1. $\mathbf{C} \rightarrow \mathbf{AS}: \; ID_c || ID_{tgs} || TS_1$
2. $\mathbf{AS} \rightarrow \mathbf{C}: \; E(K_c, [K_{c,tgs} || ID_{tgs} || TS_2 || Lifetime_2 || Ticket_{tgs}])$
   $Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} || ID_c || AD_c || ID_{tgs} || TS_2 || Lifetime_2])$

Ticket-Granting Service Exchange to obtain Service-Granting Ticket

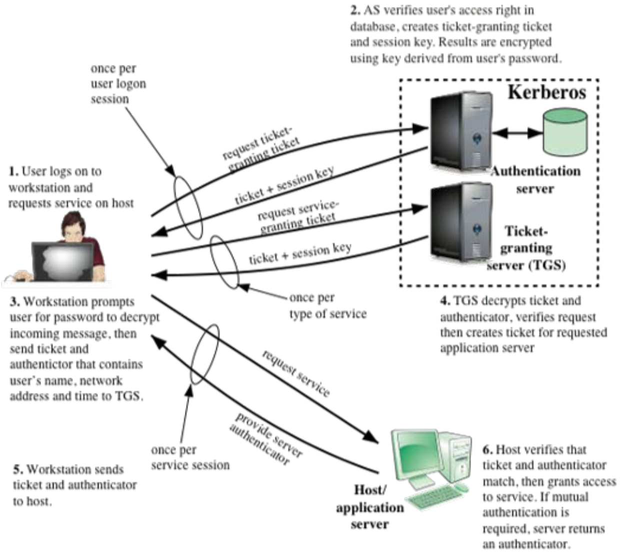
3. $\mathbf{C} \rightarrow \mathbf{TGS}: \; ID_v || Ticket_{tgs} || Authenticator_c$
   $Authenticator_c = E(K_{c,tgs}, [ID_c || AD_c || TS_3])$
4. $\mathbf{TGS} \rightarrow \mathbf{C}: \; E(K_{c,tgs}, [K_{c,v} || ID_v || TS_4 || Ticket_v])$
   $Ticket_v = E(K_v, [K_{c,v} || ID_c || AD_C || ID_v || TS_4 || Lifetime_4)$

Client/Server Authentication Exchange to obtain service

5. $\mathbf{C} \rightarrow \mathbf{V}: \; Ticket_v || Authenticator_c \; \ldots . \; Authenticator_c = E(K_{c,v}, [ID_c || AD_c || TS_5])$
6. $\mathbf{V} \rightarrow \mathbf{C}: \; E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

MONASH University

Kerberos 4 overview

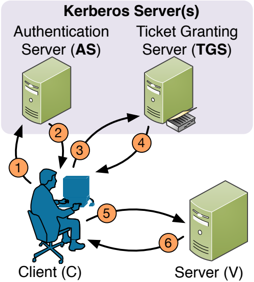


Authentication Service Exchange to obtain Ticket-Granting Ticket

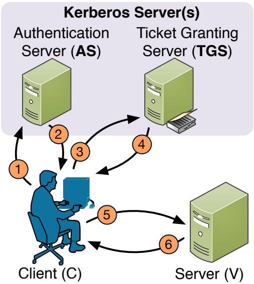
1. **C** $\rightarrow$ **AS** : $ID_c||ID_{tgs}||TS_1$
   - Client requests a TGT from AS
2. **AS** $\rightarrow$ **C** : $E(K_c, [K_{c,tgs}||ID_{tgs}||TS_2||Lifetime_2||Ticket_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs}||ID_c||AD_c||ID_{tgs}||TS_2||Lifetime_2])$

- AS sends a message encrypted with K_c
  - message includes the ticket which itself is encrypted with K_{tgs}
  - message delivers a session key $K_{c,tgs}$ for communication with TGS
  - message includes timestamp TS_2 to protect against replay
  - message and ticket include $Lifetime_2$ to indicate validity period of the ticket



Ticket-Granting Service Exchange to obtain Service-Granting Ticket



3 **C** $\rightarrow$ **TGS** : $ID_v || Ticket_{tgs} || Authenticator_c$

$$Authenticator_c = E(K_{c,tgs}, [ID_c || AD_c || TS_3])$$

- Client requests a Service-Granting ticket
  - request contains the TGT
  - request contains an authenticator encrypted with $K_{c,tgs}$
  - $K_{c,tgs}$ was delivered to client encrypted in AS response (step 2)
  - authenticator proves client's identity and has short lifetime

4 **TGS** $\rightarrow$ **C** : $E(K_{c,tgs}, [K_{c,v} || ID_v || TS_4 || Ticket_v])$

$$Ticket_v = E(K_v, [K_{c,v} || ID_c || AD_C || ID_v || TS_4 || Lifetime_4])$$

- TGS sends back a message encrypted with $K_{c,tgs}$
  - $K_{c,tgs}$ was delivered to TGS in $Ticket_{tgs}$ which was encrypted with K_{tgs}
  - message contains $K_{c,v}$ a session key to be used between client and requested server
  - message contains Service-Granting ticket $Ticket_v$ encrypted with K_v

MONASH University

Client/Server Authentication Exchange to obtain service

5. **C** $\rightarrow$ **V** : $Ticket_v || Authenticator_c$

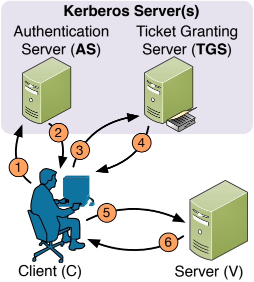
$Authenticator_c = E(K_{c,v}, [ID_c || AD_c || TS_5])$

$Ticket_v = E(K_v, [K_{c,v} || ID_c || AD_C || ID_v || TS_4 || Lifetime_4)$

- Client requests service from server V
  - request includes an authenticator encrypted with $K_{c,v}$
  - $K_{c,v}$ was delivered to client encrypted in TGS response (step 4)
  - $K_{c,v}$ is delivered to server in $Ticket_v$ encrypted with K_v

6. **V** $\rightarrow$ **C** : $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

- Server sends back a response encrypted with $K_{c,v}$
  - response is the encryption of $TS_5 + 1$
  - response proves the server's ability to recover TS_5 which requires $K_{c,v}$



Kerberos v5

- Kerberos v5 latest RFC4120 and updated by few other RFCs
- defines the concept of `realm` to specify scope of operation
  - allows cross-realm operation
  - cross-realm authentication is achieved using `inter-realm` keys
    - a client authenticated with local realm can prove its identity to servers in other realms
    - TGS in each realm is registered as a principal in the other

To use Kerberos:

- need to have a Key Distribution Centre (KDC) on your network
- need to have Kerberised applications running on all participating systems
  - the applications may use direct calls to kerberos library functions
  - the applications may use Generic Security Service API (GSS-API)
    - RFC 1964: The Kerberos Version 5 GSS-API Mechanism
- preferably a time server
  - Kerebros is sensitive to clock skews
- protocol is subject to US export restrictions [1]

[1] As of October 2003, MIT is no longer restricting downloads of Kerberos to the U.S. and Canada however export laws and regulations may still apply

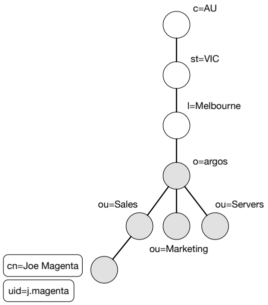
Directory Service Brief Overview

- Kerberos can store principal name and secret keys (what about other information?)
- ITU-T X.500 standard[2] provides a conceptual model for directory service
  - *Directory*: a (distributed) collection of open systems in cooperation to hold a logical database of objects
    - various administrative authorities control access to their portion of the information
    - replication can improve performance and reliability
  - *Directory Information Base (DIB)*: the collective information held by the Directory
  - *users* of the Directory, with proper permission, can read and modify the information
- ITU-T X.519 defines the directory protocol

[2] ITU-T X.500 standard

Directory Information Base

- organised in the form of a tree *Directory Information Tree (DIT)*
  - entries higher in the tree often represent objects such as
    - countries and organisations (C country, ST state, and L location) or
    - domain name hierarchies (DC domain component)
  - entries lower in the tree represent people, devices, application processes, etc.
- every entry has a *distinguished name* which uniquely identifies the entry
  - derived from the tree structure and the entry's attribute(s)
- *Directory schema*: set of rules to make sure DIB remains well-formed over time
  - prevents entries to have wrong type of attributes (for its object class)
  - prevents attribute values being of the wrong form for its type
- RFC 4519 defines LDAP Schema for User Applications



MONASH University

Lightweight Directory Access Protocol (LDAP)

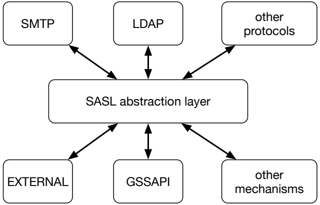
- RFC 4511 defines the protocol
- LDAP must act in accordance with X.500 standard service model
- Directory user accesses the Directory through
  - LDAP client and a Directory User Agent (DUA)
- RFC 4512 describes the X.500 Directory Information Models as used in LDAP
- RFC 4513 defines authentication methods and security mechanisms
- LDAP provides the following security mechanisms (as described in RFC 4513)
  - Bind operation
    - anonymous
    - unauthenticated
    - name/password
    - Simple Authentication and Security Layer (SASL)
  - supports Transport Layer Security
    - integrity and confidentiality
    - server authentication

MONASH University

Simple Authentication and Security Layer (SASL)[a]

[a]Simple Authentication and Security Layer (SASL)

- defined in RFC 4422
- a framework for providing authentication and data security services
  - used for connection-oriented protocols
  - provides a structured interface between protocols and mechanisms
    - allows new protocols to use existing mechanisms without redesigning the mechanism
    - allows old protocols to make use of new mechanisms without redesigning the protocol
  - the interface between protocols and mechanisms allows authentication exchanges to be carried out



- Protocols and Mechanisms Requirements
  - to use SASL each protocol provides
    - a method for identifying which mechanism to be used
    - a method for exchange of mechanism-specific messages (server-challenges and client responses)

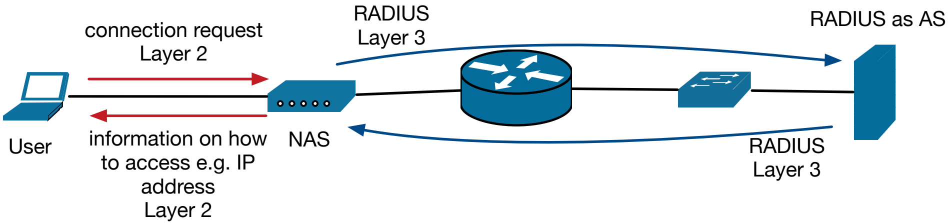
Network Access Server (NAS) and AAA Protocol Requirements

- NAS is users' point of access to a network
  - ISP DSLAM
  - Ethernet Network Switch
  - Wireless Access Point
- AAA protocols are designed for
  - Authentication: verify user's identity
  - Authorisation: whether the user is allowed access and what kind of access the user can have
  - Accounting: keep log of user activity for accounting purposes
- RFC 3169[3] defines criteria for evaluating NAS protocols
- RFC 2989[4] provides criteria for evaluating AAA protocols for network access

[3] Criteria for Evaluating NAS Protocols
[4] Criteria for Evaluating AAA Protocols for Network Access

RADIUS

- RADIUS: Remote Authentication Dial In User Service defined in RFC 2865
  - is a AAA protocol that satisfies some of the specified criteria
  - designed to provide a single database of users for
    - authentication
    - specifying the type of service provided
  - used in ISPs with modem pools
  - can be used with DSL, VPN, wireless users etc.



RADIUS Overview

- a AAA protocol that allows a NAS to communicate with an authentication server
  - NAS is the client of RADIUS server
- Supports various authentication mechanisms
  - when acting as AS itself supports
    - Password Authentication Protocol (PAP)
    - Challenge Handshake Authentication Protocol (CHAP)
    - Unix login etc.
  - Extensible Authentication Protocol
    - EAP messages between machine requesting access and NAS
    - EAP messages encapsulated in RADIUS between NAS and RADIUS server
    - Allows for more authentication methods supported by EAP

MONASH University

RADIUS Message Format

- Uses UDP

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                         Authenticator                         |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Attributes ...
+-+-+-+-+-+-+-+-+-+-+-+-+-
```

- Code: specifies the type of RADIUS packet
- Identifier: is used to match the requests and responses
- Length: the packet length including the Code, Identifier, Authenticator, and Attribute fields
- Authenticator: is used to authenticate the reply from RADIUS server
- Attribute: carry the specific authentication, authorisation, information and configuration details for the request and reply
    - vendor-specific attributes allows for more flexibility

RADIUS Message Types

- Access-Request
  - used to determine if a user is allowed to access a NAS
- Access-Accept
  - if all attributes in a previous `Access-Request` are acceptable then an `Access-Accept` is sent
  - conveys information about delivering service to the user
- Access-Reject
  - if any of the attributes in a previous `Access-Request` are unacceptable then an `Access-Reject` is sent
- Access-Challenge
  - is used in a challenge-response method
  - the response to a challenge will be sent as an `Access-Request` message

Diameter

- Next generation of AAA protocol (RADIUS replacement)
  - designed to address new demands
  - RFC 6733: Diameter Base Protocol
  - RFC 7155: Diameter Network Access Server Application
- AAA requirements specified in RFC 2989
  - supports application layer acknowledgements and defines failover algorithms
  - supports TLS and DTLS to provide transmission-level security
  - runs over TCP
  - provides support for agents
    - relay, proxy, redirect or translate
  - supports server-initiated messages (to implement re-authN and re-authZ on demand)
  - provides backward compatibility with RADIUS
  - provides support for error handling, capability negotiation and mandatory/non-mandatory Attribute-Value Pairs (AVPs)
  - supports peer discovery

Diameter Header

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |                 Message Length                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Command Flags |                  Command Code                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Application-ID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Hop-by-Hop Identifier                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      End-to-End Identifier                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  AVPs ...
+-+-+-+-+-+-+-+-+-+-+-+-+-
```

- Version: set to 1
- Command Flags (msb to lsb): Request, Proxiable, Error, T: potentially retransmitted message, reserved (last 4 bits)
- Application-ID: identify the application for which the message is applicable (e.g. authN, authZ, or acct)
- Hop-by-Hop Identifier: must be unique for a given connection (value must match between requests and replies)
- End-to-End Identifier: is used to detect duplicate messages (unique on each message)

Diameter Command Code

- Each command Request/Answer pair has a command code
  - the sub-type is identified by command flag `R`

command codes defined in base protocol

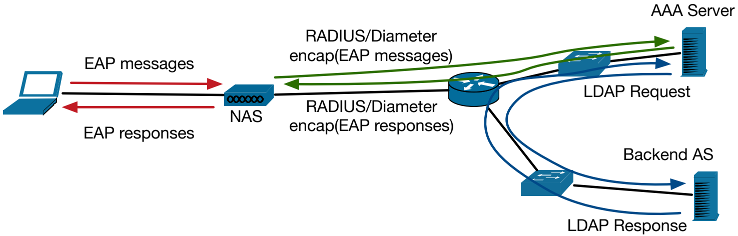
Command Name	Abbrev.	Code
Abort-Session-Request	ASR	274
Abort-Session-Answer	ASA	274
Accounting-Request	ACR	271
Accounting-Answer	ACA	271
Capabilities-Exchange-Request	CER	257
Capabilities-Exchange-Answer	CEA	257
Device-Watchdog-Request	DWR	280
Device-Watchdog-Answer	DWA	280
Disconnect-Peer-Request	DPR	282
Disconnect-Peer-Answer	DPA	282
Re-Auth-Request	RAR	258
Re-Auth-Answer	RAA	258
Session-Termination-Request	STR	275
Session-Termination-Answer	STA	275

command codes defined in NAS application

Command Name	Abbrev.	Code
AA-Request	AAR	265
AA-Answer	AAA	265
Re-Auth-Request	RAR	258
Re-Auth-Answer	RAA	258
Session-Termination-Request	STR	275
Session-Termination-Answer	STA	275
Abort-Session-Request	ASR	274
Abort-Session-Answer	ASA	274
Accounting-Request	ACR	271
Accounting-Answer	ACA	271

Extensible Authentication Protocol (EAP)

- Originally defined in RFC 2284 as PPP EAP (obsoleted by RFC 3748)
- RFC 3784 defines EAP as an authentication framework which supports multiple authentication methods
- runs directly over data link layers without requiring IP
  - PPP
  - IEEE 802
- allows defining new authentication methods without changing authentication protocol
- a sample scenario



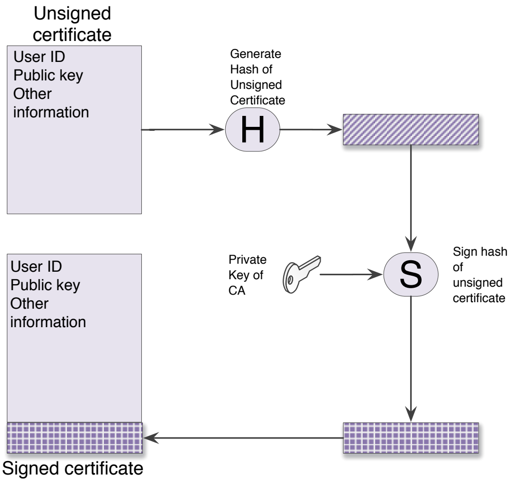
EAP Authentication Advantages[a]

[a] RFC3748 Extensible Authentication Protocol

- Can support multiple authentication mechanisms without having to pre-negotiate a particular one
- Devices such as switches or access points do not need to understand each authentication method (can act as pass-through)
- Separates the authenticator from backend authentication server (simplifies key management)

Public key method: X.509 Certificate

- defined as part of X.500 series of standard
- X.509 defines a framework of authentication services provided by X.500 to its users
- X.509 Certificate format is now used in a broader context for authentication
  - IPSec, TLS, S/MIME, etc.
- To get a certificate
  - the entity proves its identity to CA
  - presents the public key
  - CA creates the certificate document and signs it with its private key
- Any entity with access to CA can get a certificate
- Only the issuing CA can modify a certificate



X.509 Format

- RFC 5280 defines the version 3 format of X.509 certificates

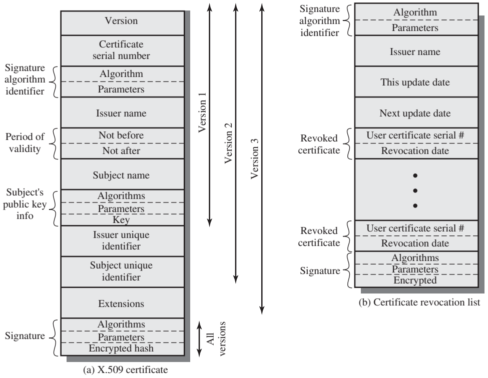


Figure 1: (a) X.509v3 Certificate Format (b) Certificate Revocation List Format[5]

[5] Figure is borrowed from "Network Security Essentials-Application Standards", 5th Edition by William Stallings

Certificate Path and Trust

- if a user of a security service does not have an authentic copy of the CA
  - a chain of multiple certificates may be needed
- Basic path validation (RFC 5280)
  - verifies that a sequence of n certificates satisfy:
    - for all x certificates the subject of x is the issuer of $x+1$
    - certificate 1 is the trust anchor (certificate of a trusted root CA)
    - certificate n is the one being validated
    - for all certificates in the path it is valid at the time in question
- obtaining the certificates in the path is outside the scope of RFC 5280
- *cross-certificates*
  - issuer and subject are different entities
  - describe a trust between two CAs
- *self-signed* : they are used to start the certificate path

Certificate Revocation

- Certificates have a period of validity
- May need to be revoked before expiry, e.g.:
  - user's private key is compromised
  - user is no longer certified by this CA
  - CA's certificate is compromised
- CAs maintain list of revoked certificates
  - the Certificate Revocation List (CRL)
- Users should check certificates with CA's CRL

References

- RFC 3748 Extensible Authentication Protocol
- RFC 3169 Criteria for Evaluating NAS Protocols
- RFC 2989 Criteria for Evaluating AAA Protocols for Network Access
- RFC 4422 Simple Authentication and Security Layer (SASL)
- ITU-T X.500 standard
- RFC 4511 Lightweight Directory Access Protocol (LDAP): The Protocol
- RFC 4512 Lightweight Directory Access Protocol (LDAP): Directory Information Models
- RFC 4513 Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms
- RFC 4519 Lightweight Directory Access Protocol (LDAP): Schema for User Applications

A good discussion on Kerberos
Chapter 15 of Cryptography and Network Security Principles and Practice 5th (or later) Edition by William Stallings