Kerberos V5
ASN.1

- data representation language:
  - data structure definition (∝ C struct, union), but variable length-arrays, optional elements, labeling, …
  - data representation “on the wire” (transfer syntax):
    - **BER**: basic encoding rules ➞ self-describing, verbose
    - **DER**: distinguished encoding rules = canonical BER
    - **PER**: packed encoding rules ➞ length/value
  - wire format not mappable to C (or Ada… ) data structures
- others: XDR, Internet ad-hoc (network byte order, ASCII + CRLF)
- use: PKCS, Kerberos V5, SNMP, H.323, …
ASN.1: Simple Types

**BOOLEAN:** TRUE or FALSE

**INTEGER:** infinite precision

\[
\text{ContentLength ::= INTEGER} \\
\text{Version ::= INTEGER \{ v1988 (0) \}} \\
\text{length ContentLength ::= 100}
\]

**REAL:** arbitrary precision

**BIT STRING:** any number of bits

**OCTET STRING:** any number of bytes

**NULL:** placeholder

**PrintableString:** printable characters
**T61String:** eight-bit (T.61)

**IA5String:** ASCII

**UTCTime:** GMT (UTC) time: 960813003058Z

**OBJECT IDENTIFIER:** hierarchical identifier:

```
iso (1) member-body (2) US (840) rsadsi (113549) pkcs (1)
```
**ASN.1: Constructor Types**

**SEQUENCE:** structure

```
Validity ::= SEQUENCE {
    start UTCTime,
    end UTCTime
}
```

**SEQUENCE OF:** dynamic array

**CHOICE:** union

**SET:** unordered collection $\geq 1$

**SET OF:** unordered collection

**ANY:** any data type, unspecified

tagging:
distinguish elements of same type

universal tag: designate standard types (1...28)

application-wide

context-specific: within constructor

private tag: enterprise

version [0]

additional wrapping of data unless IMPLICIT
ASN.1: BER/DER Transfer Syntax

tag: class (universal, application, ...), primitive/constructed, tag (5 bits); use more bytes if needed

length: • definite length: length of length + length (base 256) or length < 127
  • indefinite length: 0, data, 00

value: BIT STRING: bits unused, bits
  OCTET STRING: simply bytes
  OID: base 128 (high bit set: more bytes)
ASN.1

- general
- must be parsed recursively
- not aligned
- not space efficient
Delegation of Rights

- transfer rights to object, for limited time
- can’t delegate: contain network address of requestor
- V5: ask for TGT for different node or any node (audit!)
- may grant TGT or ticket to specific service
- forwardable: exchange for TGT with different address
- may ask for TGT that can again be forwarded
Ticket Lifetimes

- unlimited lifetime instead of 21 hours
  - start time (may be *postdated* into the future)
  - end time (may be adjusted)
  - authorization time (initial TGT)
  - renew-till = upper bound on renewal
  - postdating may require revalidation

- renewable ticket

- can’t renew expired ticket
Key Protection

- single password in all realms \(\rightarrow\) same masterkey
- compromise one KDC \(\rightarrow\) compromise all
- solution: master key depends on realm
Optimizations

- V4: ticket encrypted ➞ unnecessary
- ticket target (“Bob”) no longer in ticket
Cryptographic Algorithms

- V4: DES only export-controlled, limited security
- V5: algorithm indication, but not negotiation
- only as secure as weakest algorithm accepted
- *should* use MD(secret|message)
Kerberos V5 Integrity: rsa-md5/md4-des

1. confounder $C = 64$-bit random number
2. compute MD5 (MD4) on $C|m$ ➞ 128-bit digest
3. prepend confounder to message digest
4. derive key $K'$ from KDC shared secret $K$ by $⊕$ing
5. $K'|$message} using DES CBC, IV = 0

 تعال $192$-bit MIC
Integrity: des-mac

1. *confounder* $C = 64$-bit random number

2. prepend confounder to message

3. DES CBC residue using $K$ and IV = 0  $\Rightarrow$ 64-bit residue $R$

4. modified key $K' = K \oplus f0f0f0f0f0f0f0f0f016$

5. DES CBC on $K'\{C|R\}$, IV = 0

$\Rightarrow$ 128-bit MIC
Privacy and Integrity

1. confounder $C = 64$-bit random number
2. checksum($C|0\ldots0|m$), where checksum $\in \{\text{CRC-32, MD4, MD5}\}$
3. fill in $0\ldots0$ with checksum
4. pad
5. encrypt using CBC, IV = 0
Hierarchy of Realms

- V4: each realm must be registered in “origin” realm
- V5: allow chaining
- e.g., Alice in \( A \) talk to Carol in \( C \); \( C \) not registered in \( A \)
- \( B \) registered in \( A \), \( C \) in \( B \)
- allows realm \( B \) to impersonate anybody
- list transit domains (reject if KDC named doesn’t match key)
- trust: transit or for principals
- realm tree: share key with parents, children
- allow only shortest path through tree (lowest common ancestor)
- identify tree based on names (domain hierarchy)
- cross links as shortcuts
Password-Guessing Attacks

*human* keys subject to guessing:

- V4: cleartext request for TGT for Alice $\Rightarrow$ password guessing
- prove possession of Alice’s master key (?)

use own TGT to ask for ticket to human principal

- mark human principals $\Rightarrow$ don’t hand out tickets
- doesn’t work with email

note: off-line guessing still possible (Bellovin/Merritt)
Double TGT Authentication

- ticket encrypted with Bob’s *master* key
- Bob may want to forget master key (but keep TGT, session key)

Solution:

- Alice should ask Bob for TGT (encrypted with KDC’s master key)
- Alice sends $TGT_{\text{Alice}}, TGT_{\text{Bob}}$
- KDC issues ticket encrypted with Bob’s *session* key

Application: X client (app.) writing to X server (screen control)
GSS API, Version 2

- API, not protocol
- RFC 2078
- may use any method, but Kerberos V5, X.509 are outlined
- language-independent, ASN.1-like data structures
- language binding: RFC 1509 (Version 1) for C

client

GSS_Acquire_cred();
GSS_Init_sec_context();
GSS_Wrap(data);
GSS_GetMIC();

server

GSS_Acquire_cred();
GSS_Accept_sec_context();
GSS_Unwrap();
GSS_VerifyMIC();
GSS_Delete_sec_context();