Internet Security Protocols: Specification and Modeling

AVISPA

Automated Validation of Internet Security Protocols and Applications

Shared cost RTD (FET open) project IST-2001-39252

Tutorial

Contents

Internet Layers, Basics

Management, Implementation or Design Errors

IETF Groups and Activities

Security Protocols: Kerberos, AAA, IPsec, IKE, WLAN

Public-Key Infrastructure (PKI)

High-level Protocol Specification Language (HLPSL)

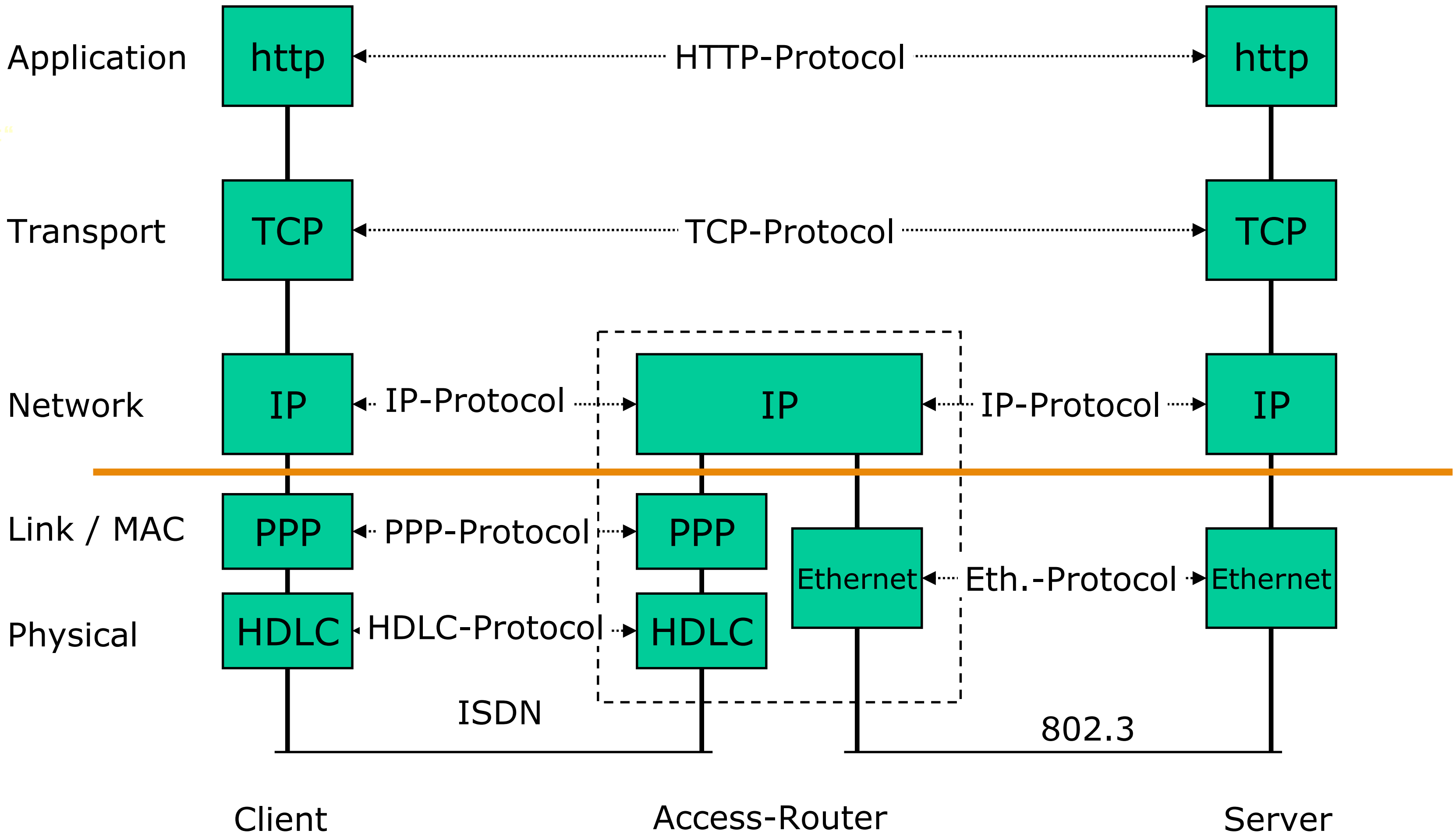
Outlook: MobileIP, DRM

Protocol layering in Internet

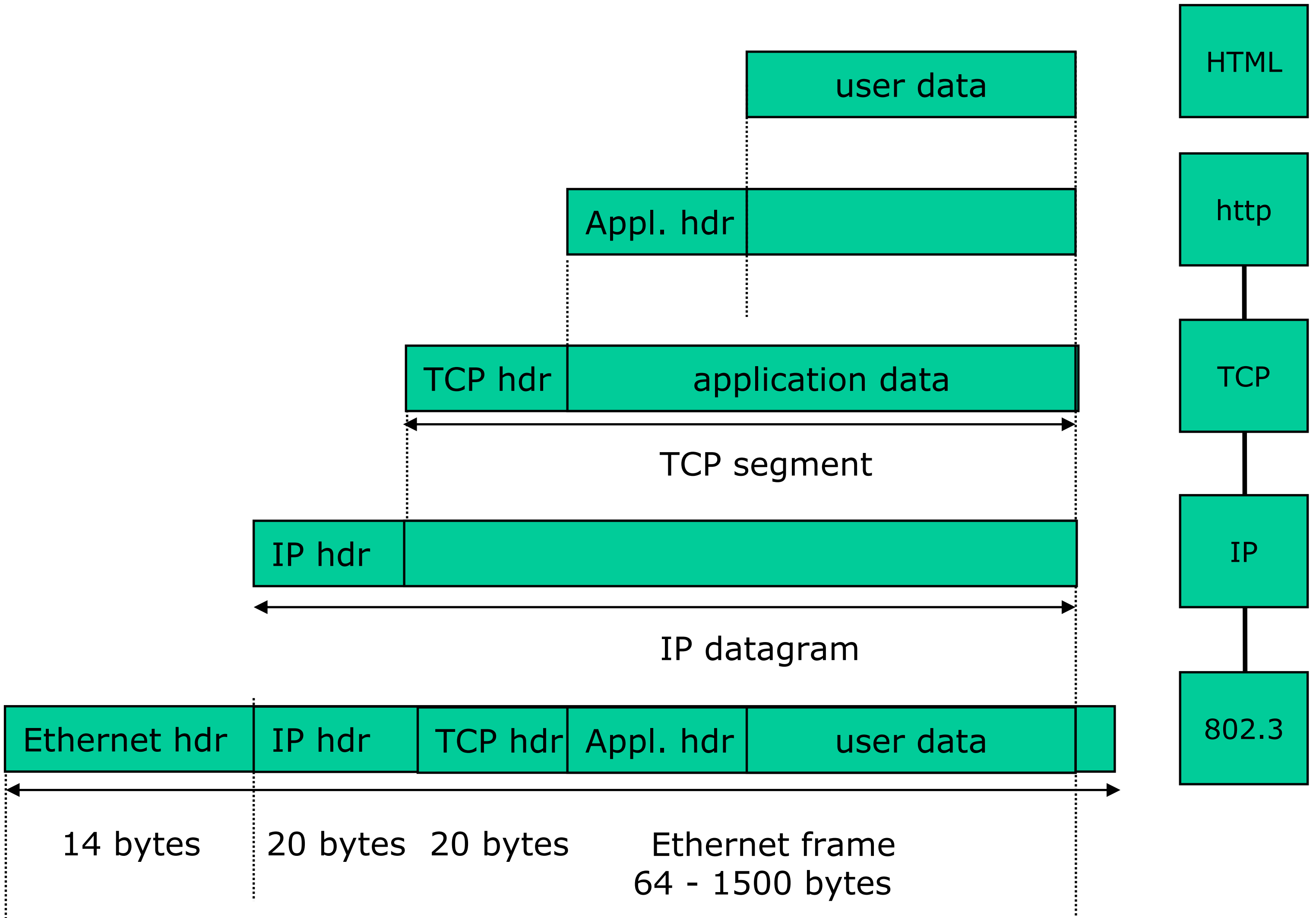
„Independent" Layers

Headers

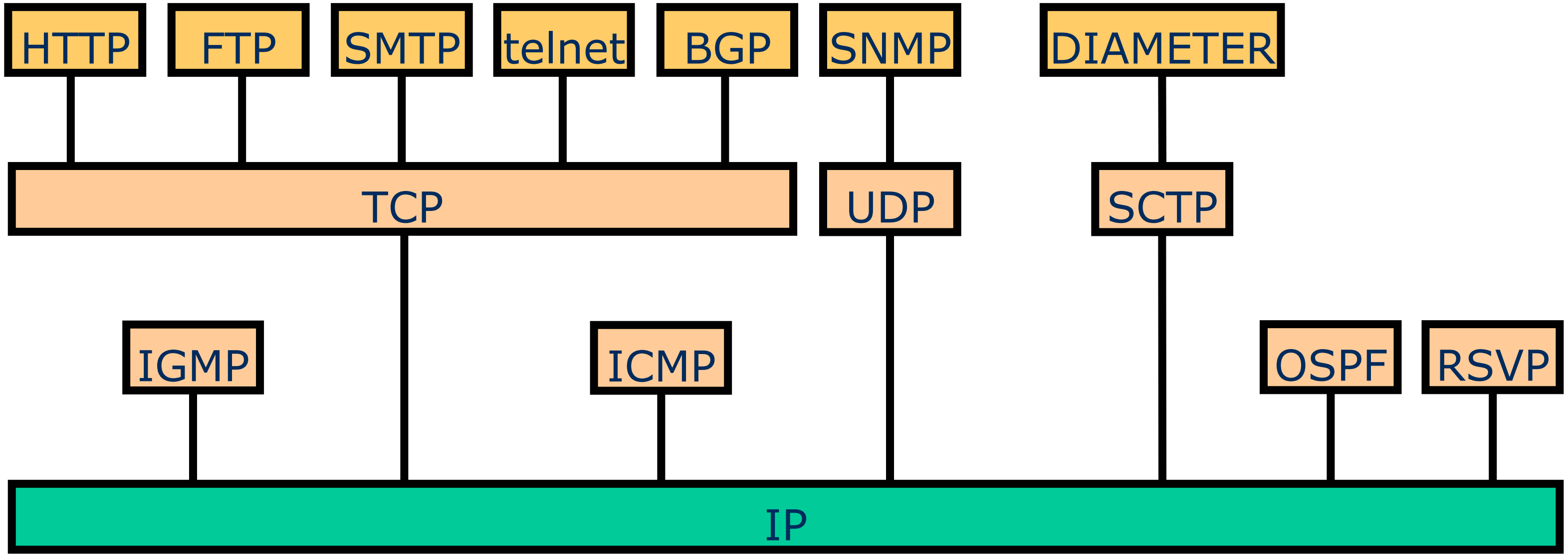
Tunneling



Encapsulation



Protocols in the TCP / IP Suite



HTTP = Hypertext Transfer Protocol

FTP = File Transfer Protocol

SMTP = Simple Mail Transfer Protocol

BGP = Border Gateway Protocol

TCP = Transmission Control Protocol

IGMP = Internet Group Management Protocol

ICMP = Internet Control Message Protocol

SNMP = Simple Network Management Protocol

UDP = User Datagram Protocol

DIAMETER = (2 x RADIUS) = New AAA Prot.

SCTP = Stream Control Transmission Protocol

OSPF = Open Shortest Path First

RSVP = Resource ReSerVation Protocol

IP = Internet Protocol

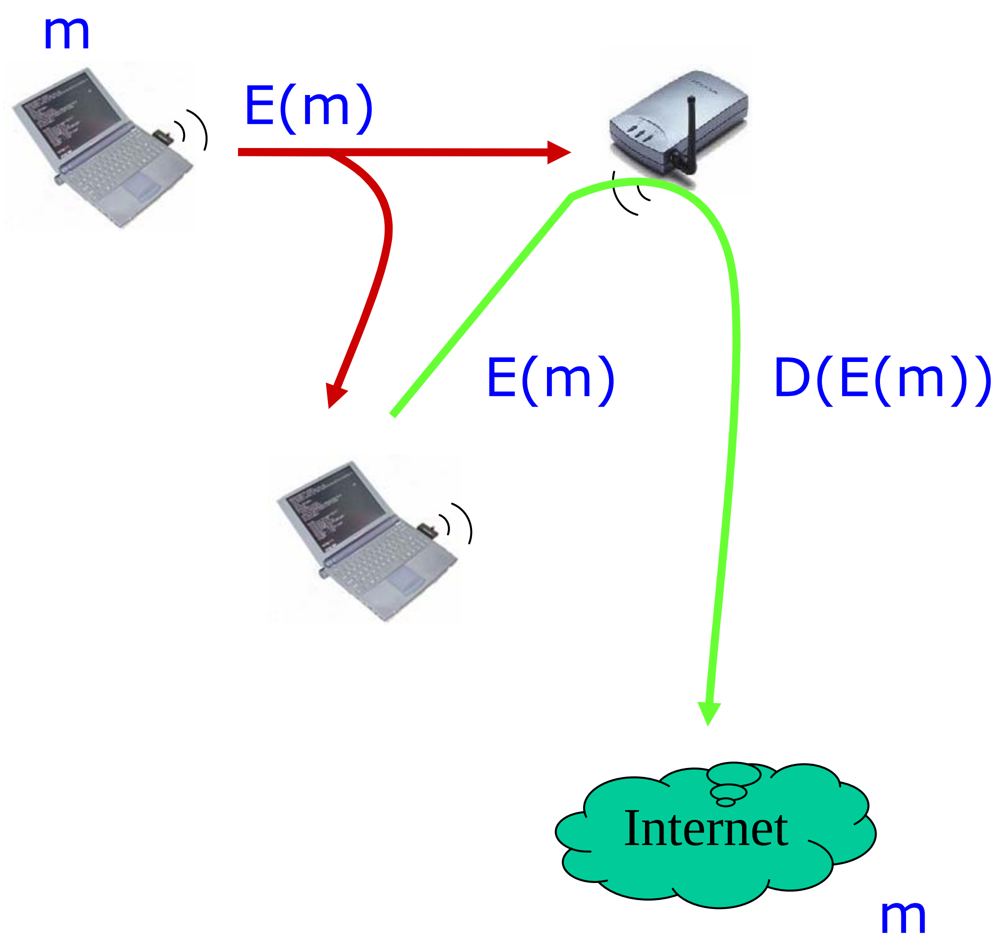
Contents

Is PKI secure? Some Management Problems

- Most users do not know what certificates are.
- Most certificates' real-world identities are not checked.
- Meaningless Certificates:
  - Which Dow Jones owns the www.WSJ.com certificate?
  - Is that certificate good for online.WSJ.com?
- Is it WHITEHOUSE.COM or WHITEHOUSE.GOV?
  - MICROSOFT.COM or MICR0S0FT.COM?
  - What about MICROSOFT.COM? (Cyrillic "O", do you see it?)
- Actually, we have no PKI for the Web.

Design Problems: WLAN / WEP

AVISPA



Denial of Service Attacks (DoS)

- Attacker does not break in, but prevents legitimate access to resources, typically by overwhelming the server.
- Many incidents reported, more are likely.
- Denial of Service Attacks are hard to prevent
  - in particular using security measures at higher layers only
- You lose if it's cheaper for the attacker to send a message than for you to process it
- Thumb rules:
  - Try to be stateless, move state to the initiator.
  - Allocate resources as late as possible.
  - Move computation load to the initiator of the protocol run.
  - Do expensive computations as late as possible.

Cryptographic puzzles

- Used to protect against resource-exhaustion attacks.
- A server requires its clients to solve a puzzle,

  e.g. brute-force search for some input bits of a one-way function, before committing its own resources to the protocol.
- The server can adjust the difficulty according to its current load.
- Solving the puzzle creates a small cost for each protocol invocation, which makes flooding attacks expensive but has little effect on honest nodes.
- Drawbacks:
  - IP layer does not know which node is the server (i.e. the responder)
  - The puzzle protocols work well only when all clients have approximately equal processing power
  - Mobile node often have limited processor and battery capacity while an attacker pretending to be a MN is likely to have much more computational resources

Setting a limit on the amount of resources

- Processor time, memory and communications bandwidth, used for location management.
- When the limit is exceeded, communication needs to be prioritized.
- Node should try to resume normal operation when attack may be over.

Attacks to the infrastructure: Routing Attacks

- Routers advertise
  - own local nets,
  - what they've learned from neighbours
- Routers trust dishonest neighbours
- Routers further away must believe everything they hear

GSM Today

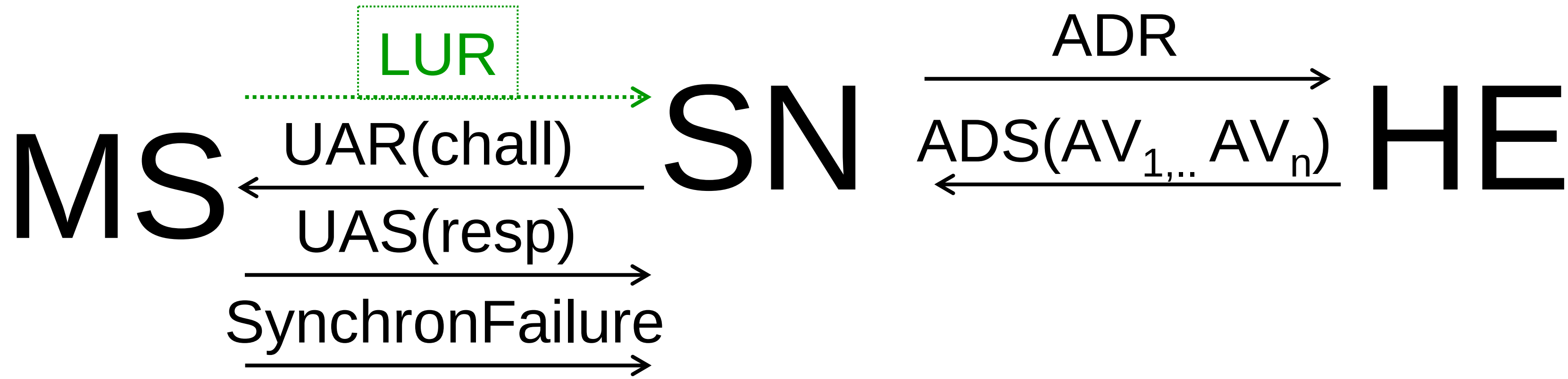
- AV = (chall, resp, C), C = Cipher Key, K = secret of MS and HE
- AV generation is not so fast → batch generation
- MS is able to calculate: C = $\text{cipherkey}_k(\text{chall})$
  Therefore MS and SN share C.
- MS authenticates to SN, but
- SN does not authenticate to MS

GSM Today: Problem

MS $\xleftrightarrow{C}$ SN' $\longleftrightarrow$ MS' $\xleftrightarrow{C'}$ SN

- If attacker gets hold of one (for instance, used) AV:
  - he may create false base station SN'
  - force MS to communicate to SN' (using C)
  - decrypt/encrypt
  - use another (legal) user MS' (with key C')
- Possible:
  - says(A,B,m) /\ notes(C,A,m) /\ C != B
  - notes(A,B,m) /\ says(B,A,m') /\ m' != m

UMTS: Idea



- MS is able to check that challenge is consistent: $cons_k(chall)$
- AV_i also contain a sequence number, that may be reconstructed by the MS: $seq = seq_k(chall)$
- MS accepts AV_i only if

$$seq_{MS} < seq_k(chall) \leq seq_{MS} + \Delta$$

UMTS: Idea

Is there a MitM attack?
Is there deadlock?

Such design errors would be very expensive:
Replace firmware in many towers
and in millions of Cellular Phones

Answer: No

Formal Proof!

Variable Security

- Different security mechanisms
  - variable levels of guarantees
  - variable security properties
  - variable cost
- Challenge:
  - find an acceptable level of protection
  - at affordable price
- Find:
  - most inexpensive security mechanisms
    - even if they are weak!
  - that solve your problem

Contents

Internet History

1961-1972: Early packet-switching principles

- 1961: Kleinrock - queuing theory shows effectiveness of packet-switching (“data highway”)
- 1964: Baran - packet-switching in military nets
- 1967: ARPAnet conceived by Advanced Research Projects Agency
- 1969: first ARPAnet node operational
- 1972:
  - ARPAnet demonstrated publicly
  - NCP (Network Control Protocol) first host-host protocol
  - first e-mail program
  - ARPAnet has 15 nodes

Internet Organizations

ISOC (Internet Society)
political, social, technical aspects of the Internet
http://www.isoc.org/

IAB (Internet Architecture Board)
oversight of Internet architecture and standards process;
liaisons with e.g. ITU-T, ISO
http://www.iab.org/iab/

IETF
(Internet Engineering Task Force)
standardizes Internet protocols;
open community for engineers,
scientists, vendors, operators
http://www.ietf.org/

IRTF
(Internet Research Task Force)
pre-standards R&D
http://www.irtf.org/

IETF

- 3 meetings a year.
  - working group sessions,
  - technical presentations,
  - network status reports,
  - working group reports, and
  - open Internet Engineering Steering Group (IESG) meeting.

- Proceedings of each IETF plenary,
- Meeting minutes, and
- Working group charters (which include mailing lists)

are available at www.ietf.org.

IETF procedures

- The IETF is a group of individual volunteers (~ 4.000 worldwide, ~5% from universities)
- Work is being done on mailing lists (plus 3 meetings/year)
- No formal membership, no formal delegates
- Participation is free and open
- 126 (as of March 2005) working groups with well-defined tasks and milestones
- Major US vendors dominate the IETF
- IETF does not decide about the market, but:
  approval of the IETF is required for global market success.

Protocol design is done in working groups

- Basic Principles
  - Small, focused efforts preferred to larger, comprehensive ones
  - Preference for a limited number of options
- Charter
  - Group created with a narrow focus
  - Published Goals and milestones
  - Mailing list and chairs' addresses
- "Rough consensus" (and running code!)
  - No formal voting (IESG decides)
  - Disputes resolved by discussion and demonstration
  - Consensus when no explicit opposition
  - Mailing list and face-to-face meetings
- „Final" decisions made not at meetings by via e-mail

indiv. draft

group draft

last call

IESG accept

RFC

standard

Coverage of Protocols

The IETF needs tools that cover a wide range of protocols and security properties:

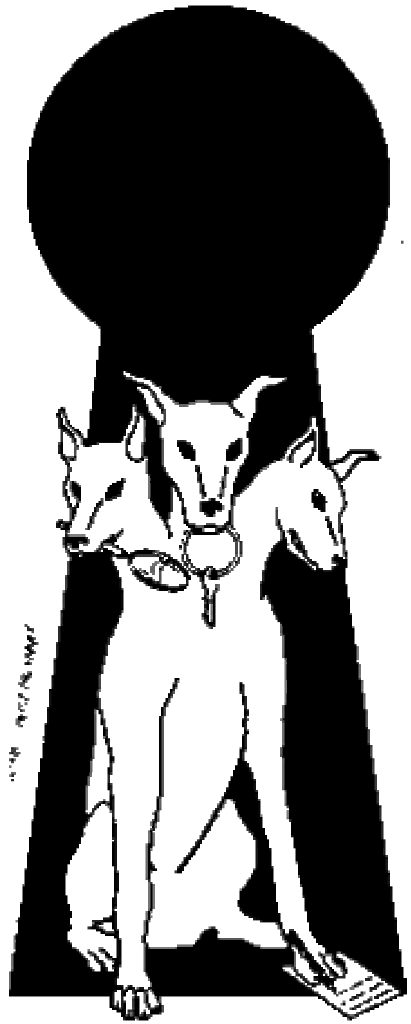
- 6 different areas in 33 groups:
  - Applications Area (app)
  - Internet Area (int)
  - Operations and Management Area (ops)
  - Routing Area (rtg)
  - Security Area (sec)
  - Transport Area (tsv)
- 5 IP layers
- > 20 security goals (as understood at IETF, 3GPP, OMA, etc)

Protocol Areas

- Infrastructure (DHCP, DNS, BGP, stime)
- Network Access (WLAN, pana)
- Mobility (Mobile IP, UMTS-AKA, seamoby)
- VoIP, messaging, presence (SIP, ITU-T H530, impp, simple)
- Internet Security (IKE (IPsec Key agreement), TLS, Kerberos, EAP, OTP, Sacred, ssh, telnet,...)
- Privacy (Geopriv)
- AAA, Identity Management, Single Sign-On (Liberty Alliance)
- Security for QoS, etc. (NSIS)
- Broadcast/Multicast Authentication (TESLA)
- E-Commerce (Payment)

Contents

Kerberos

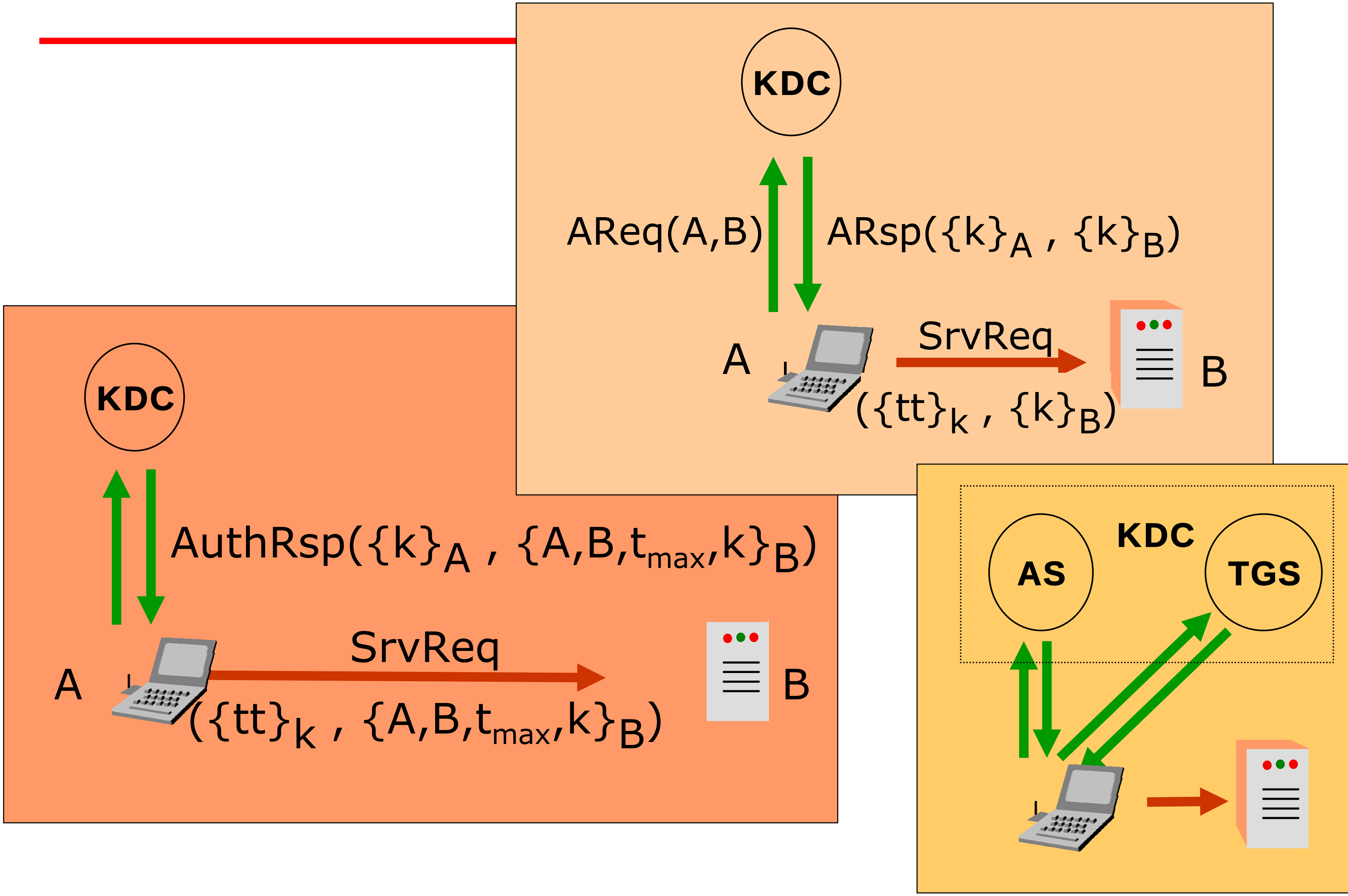


An authentication system for distributed systems

Purpose: Single Sign-On

Access Control

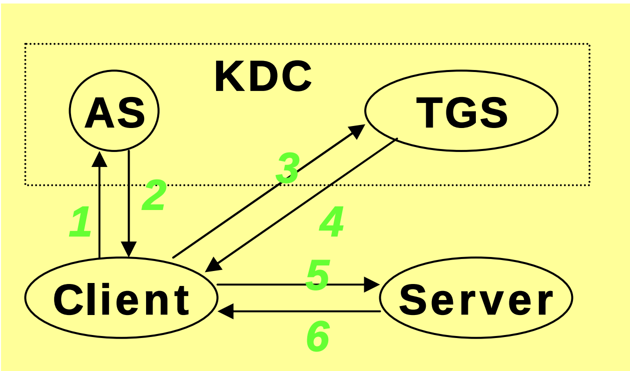
Kerberos in three Acts



Complete Kerberos

AVISPA

(from: B. C. Neuman + T. Ts'o: IEEE Communications Magazine SEP. 1994)



Protocol

< client communicates with AS to obtains a ticket for access to TGS >

1. Client (with id C) requests AS of KDC to supply a ticket in order to communicate with TGS.
   - request **(C, TGS)**
2. AS returns a ticket encrypted with TGS key Kt along with a TGS session key Kct encrypted with client key Kc:
   - return = **({ticket1}Kt, {Kct}Kc)**
   - **ticket1= (C, TGS, start-time, end-time, Kct)**

< client communicates with TGS to obtain a ticket for access to other server >

3. Client requests TGS of KDC to supply a ticket in order to communicate with the server.
   - request = **({C, timestamp}Kct, {ticket1}Kt, S)** S: server key
4. TGS checks the ticket. If it is valid, TGS generates a new random session key Kcs.
   TGS returns a ticket for S encrypted with Ks along with a session key Kcs.
   - return = **({ticket2}Ks, {Kcs}Kct)** **ticket2 = (C, S, start-time, end-time, Kcs)**

   client decrypts {Kcs}Kct with Kct to get Kcs.

< client communicates with the server to access an application >

client generates authenticator A with the information from ticket.
- **A = ({C, S, timestamp, address}Kcs)**

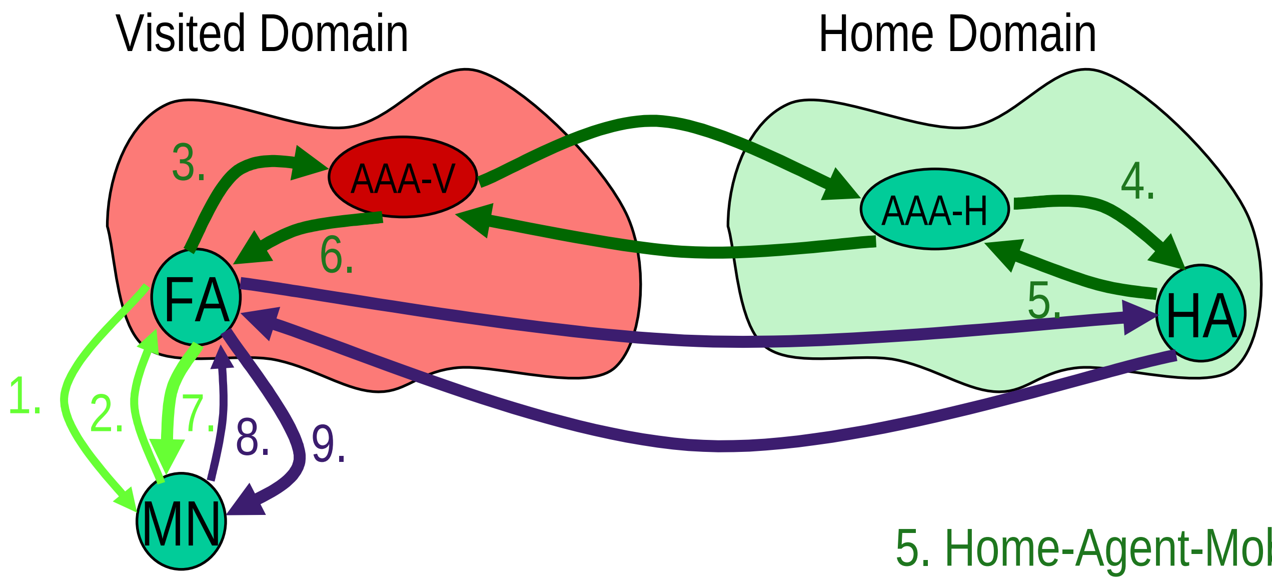
5 . Client sends the service request to the server along with the ticket and A.
- **({ticket2}Ks, A, request)**

6. The server decrypts ticket using Ks and checks if C, S, start, end times are valid.
   If service request is valid, uses Kcs in the ticket to decrypt authenticator.
   Server compares information in the ticket and in the authenticator.
   If they agree, the service request is accepted and the server acknowledges this to the client.

Kerberos V5 Ticket Types

- Renewable Ticket
  - Used for batch jobs.
  - Ticket has two expiration dates.
  - Ticket must be sent to the KDC prior the first expiration to renew it.
  - The KDC checks a “hot list” and then returns a new ticket with a new session key and a new second expiration date.
- Proxiable Ticket
  - Makes it possible for a server to act on behalf of the client to perform a specific operation (e.g. print service)
  - Purpose: granting limited rights only
- Forwardable Ticket
  - Similar to proxiable ticket but not bound to a specific operation
  - Mechanism to delegate user identity to a different machine/service
  - Sample application: telnet

AAA (Diameter) for MobIPv4



1. Agent advertisement + Challenge
2. Registration Request
3. AAA-Mobile-Node-Request
4. Home-Agent-MobileIP-Request
5. Home-Agent-MobileIP-Answer
6. AAA-Mobile-Node-Answer
7. Registration Reply
8. Registration Request
9. Registration Reply

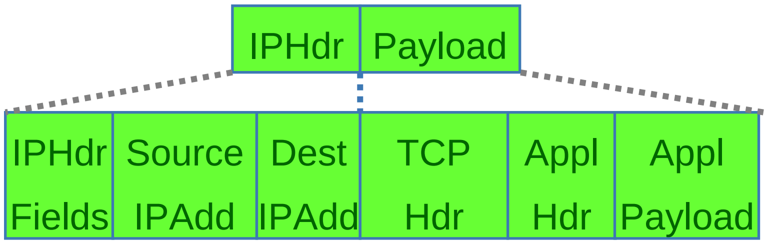
ad 7. Now there are SA:
MN-FA, MN-HA, FA-HA

(8. + 9. Renewal with extensions:
MN-FA-, MN-HA-,FA-HA-Auth)

IPsec

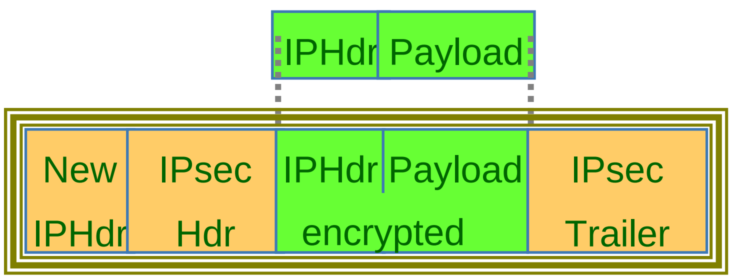
- IPsec is the standard suite of protocols for network-layer confidentiality and authentication of IP packets.
- IPsec = IKE + Authentication Header (AH)
  | Encapsulating Payload (ESP)
- In particular the following features are provided:
  - Connectionless integrity
  - Data origin authentication
  - Replay Protection (window-based mechanism)
  - Confidentiality
  - Traffic flow confidentiality (limited)
- An IPv6 compliant implementation must support IPsec.

Unsecured Messages vs. Secured Messages

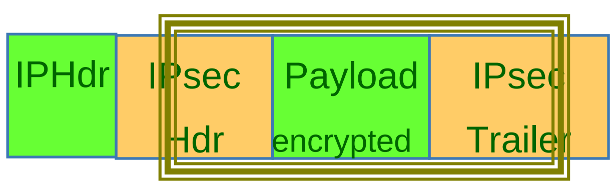


IP Spoofing
Session hijacking
Man-in-the-middle

Eavesdropping
Message modification

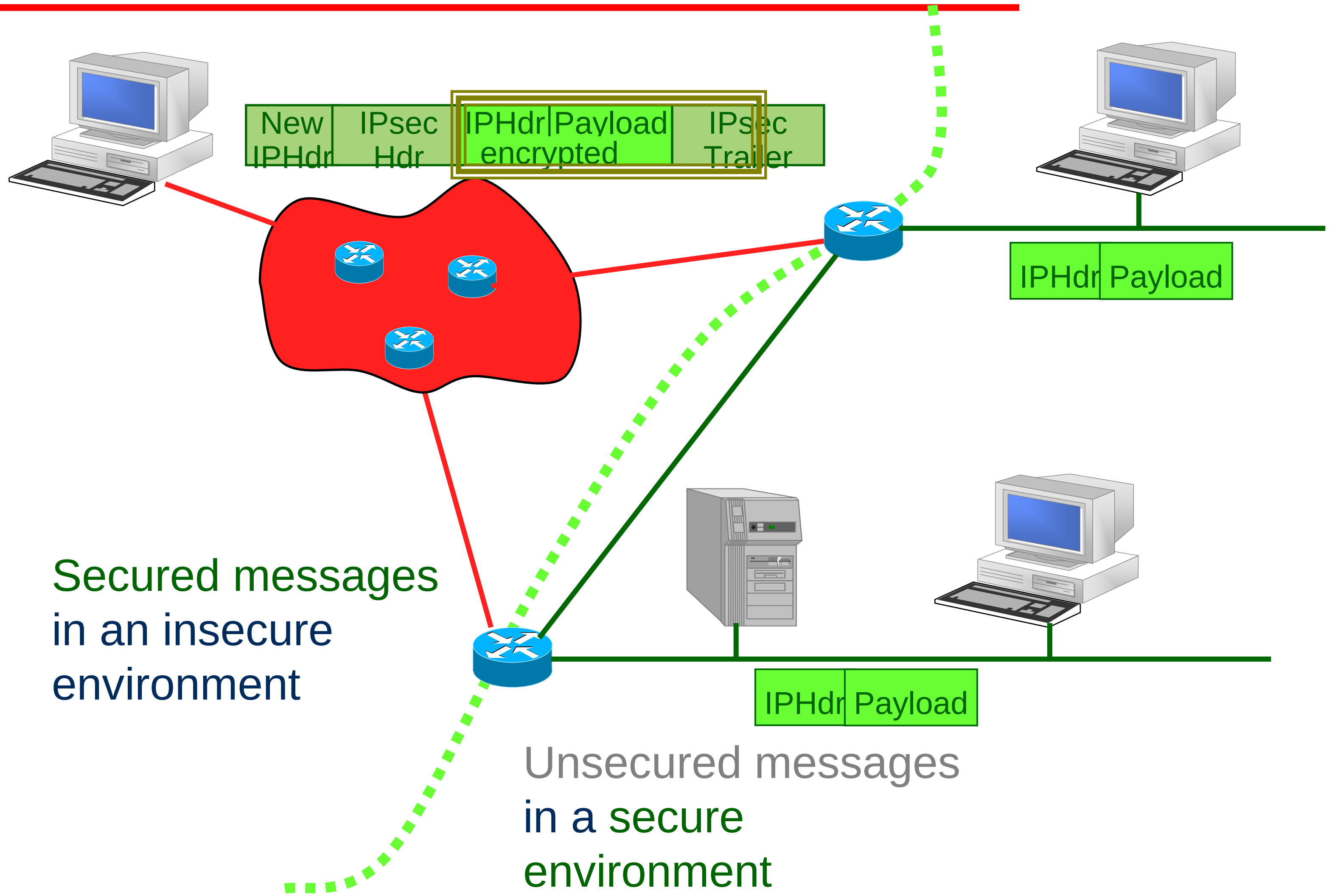


Tunnel mode:
the whole package is being encapsulated
in a new package



Transport mode (less expensive)
new IPsec Header (+ maybe Trailer)
provides somewhat less security

Use of IPsec: Tunnel Mode

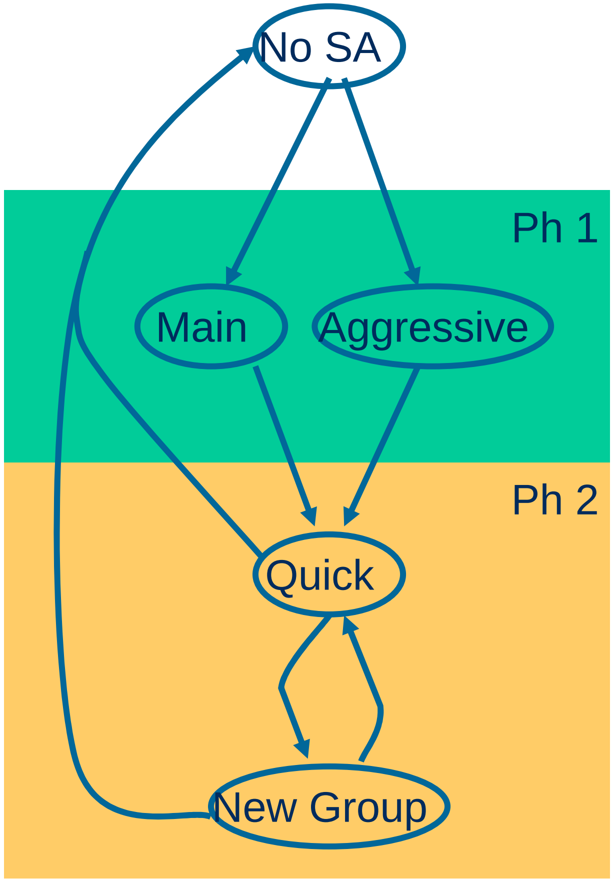


IPSec SA

- A **Security Association** (SA) is a data structure. The SA provides the necessary parameters (algorithms, modes, keys, addresses, timeouts) to secure data. SAs can be established manually or dynamically (e.g. IKE, IKEv2).
  - Alternatives being discussed:
    - Kerberized Internet Negotiation of Keys (KINK) : Status??
    - Host Identity Protocol (+Payload) (HIP)
- An IPsec SA is uniquely identified by:
  - Security Parameter Index, SPI (32 bit)
  - Destination IP Address
  - Protocol (AH or ESP)

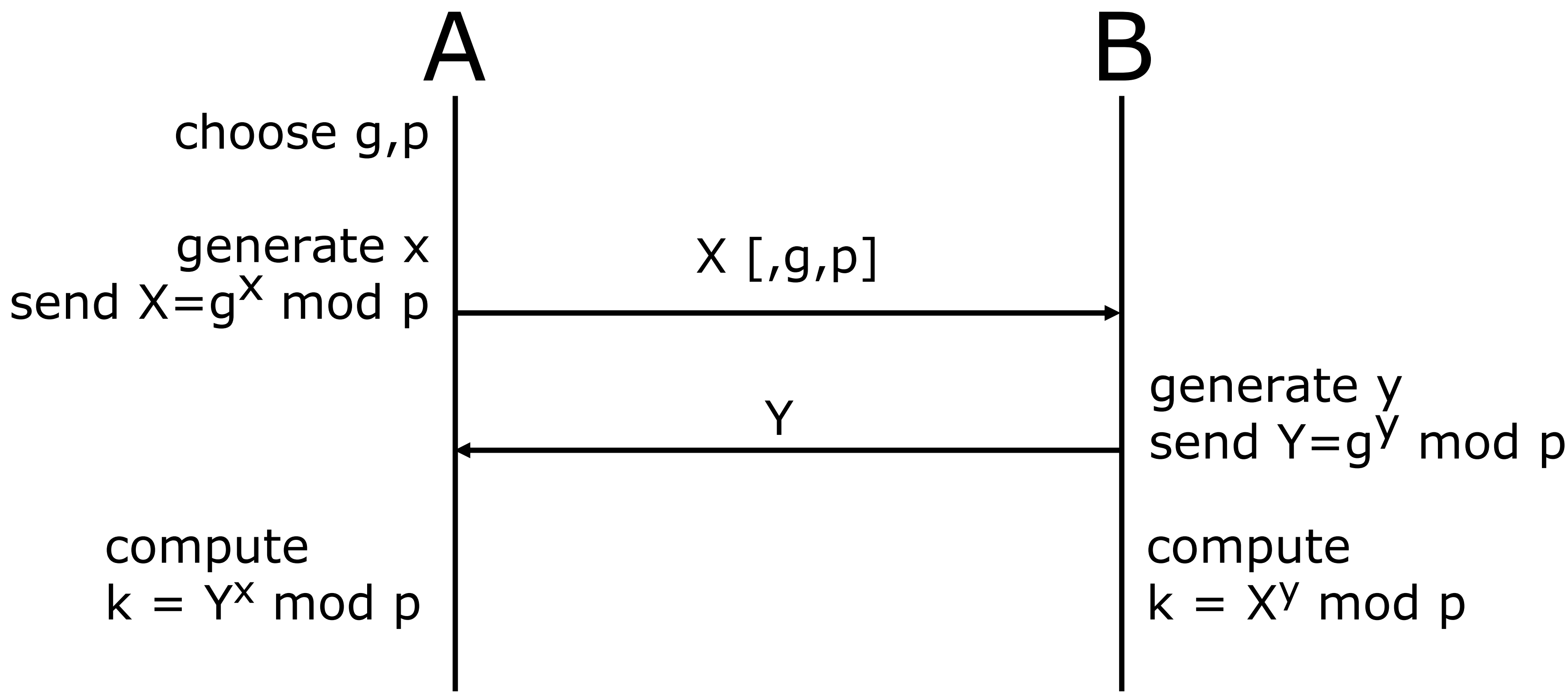
Internet Key Exchange (IKE)

- ISAKMP Phases and Oakley Modes
  - Phase 1 establishes an ISAKMP SA
    - Main Mode or Aggressive Mode
  - Phase 2 uses the ISAKMP SA to establish other (IPsec, etc.) SAs
    - Quick Mode
    - New Group Mode
- Authentication in Phase 1 with
  - Signatures (DSS/RSA)
  - Public key encryption
    - Two variants with RSA
    - Based on ability to decrypt, extract a nonce, and compute a hash
  - Pre-shared keys
- Four of the five Oakley groups



(simplified) IKE
modes and phases

Diffie-Hellman Key Exchange

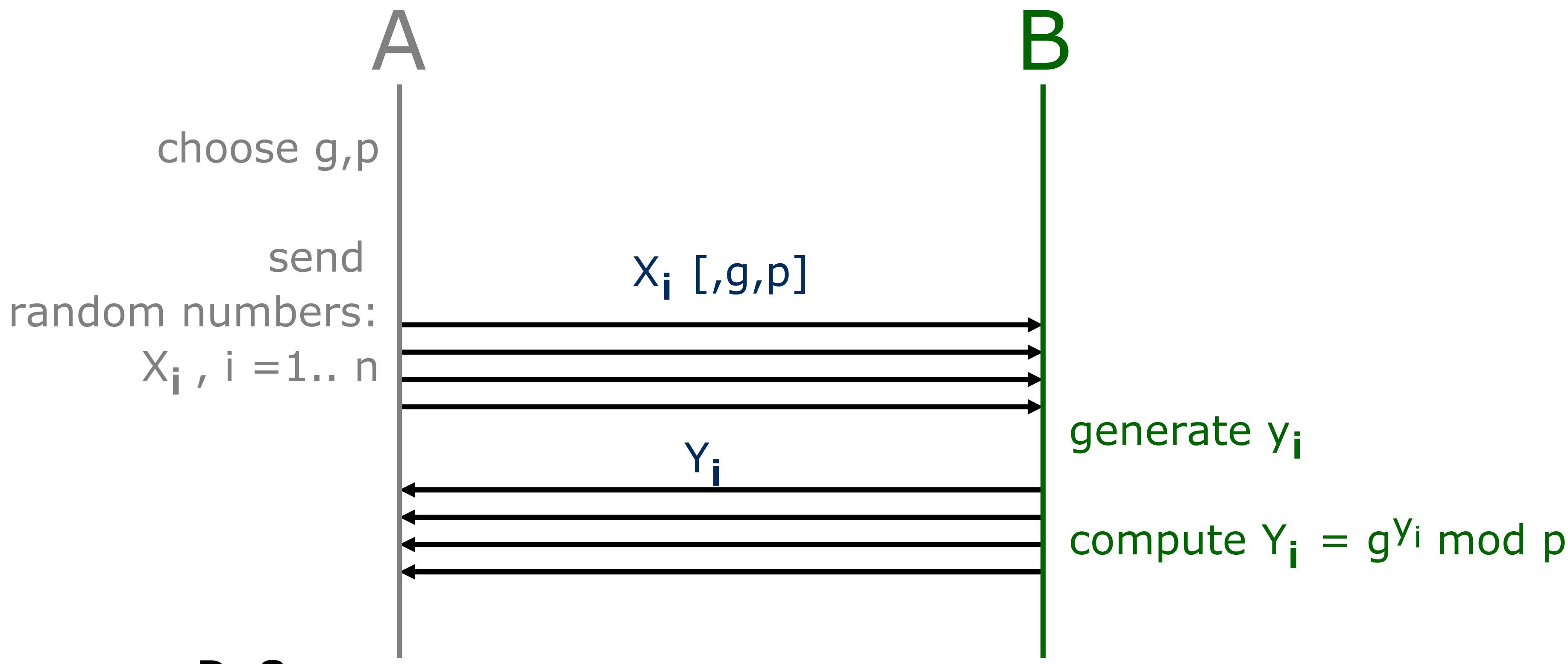


$$k = Y^x \bmod p = (g^y)^x \bmod p = (g^x)^y \bmod p = X^y \bmod p = k$$

The parameters g and p are typically known to all communication partners.

Diffie-Hellman offers Perfect Forward Secrecy !

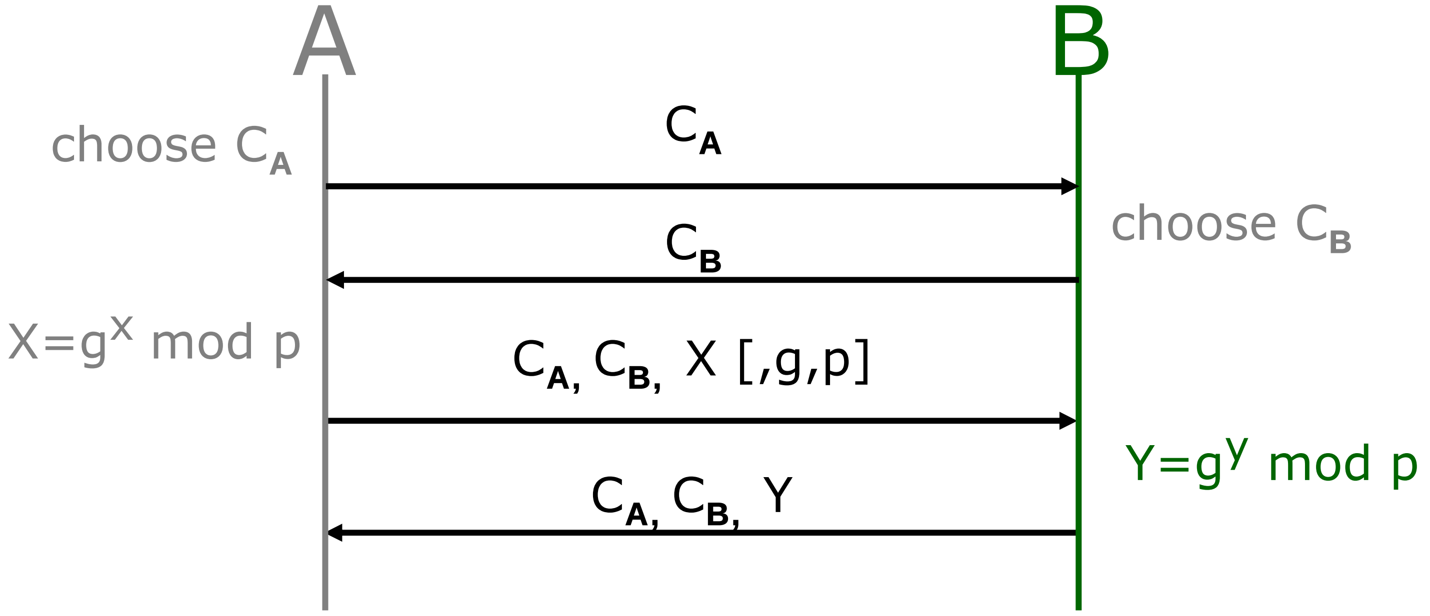
Denial of Service (Flooding)



DoS:

- B: expensive computation
- B: Memory allocation
- A: IP spoofing to prevent traceability.

DoS Protection: Cookies



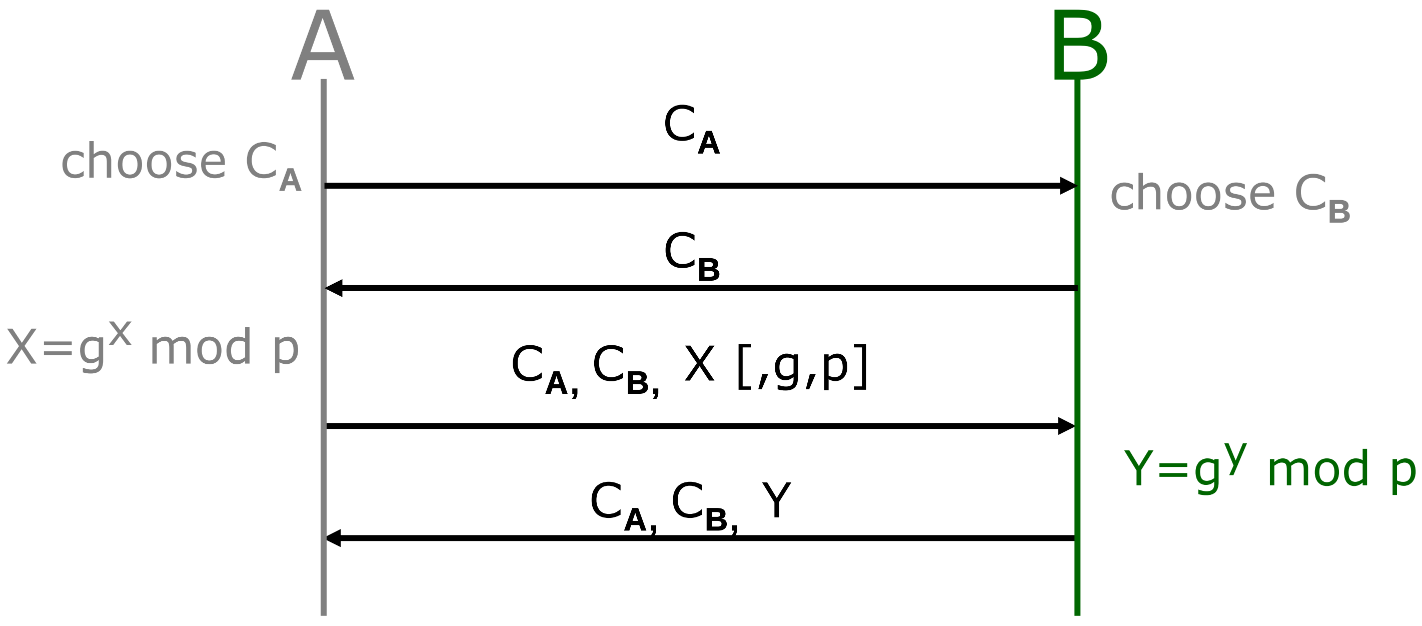
Return routability proof:

- A has to have seen C_B to send the next msg
- If A spoofs Add_i, it has to sit on path Add_i ... B

Close to Add_i : not many active addresses

Close to B

IKE: Cookies



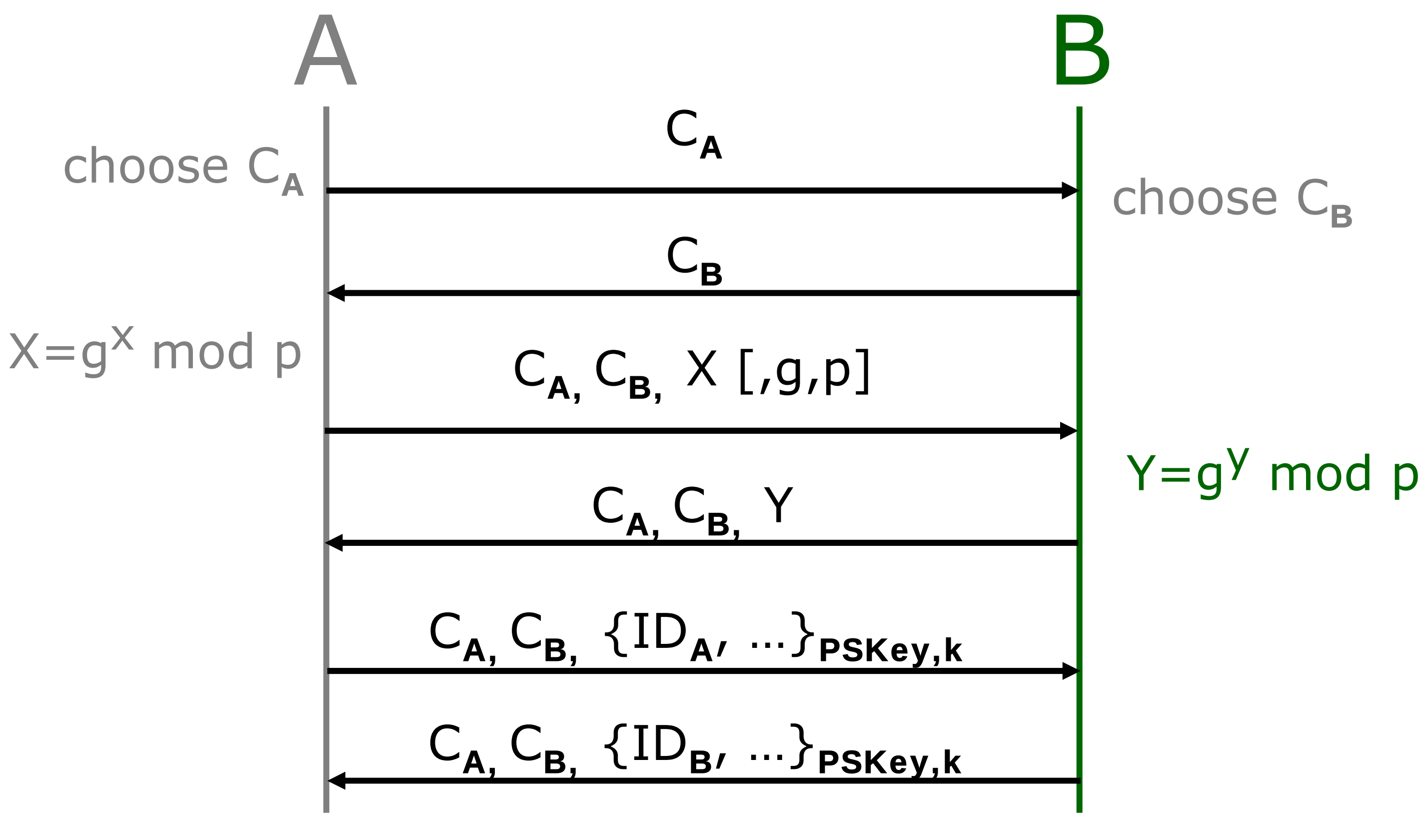
If A uses repeatedly same Address:

- Same or unknown cookie: B discards
- Different cookies: A must wait

Problem remains:

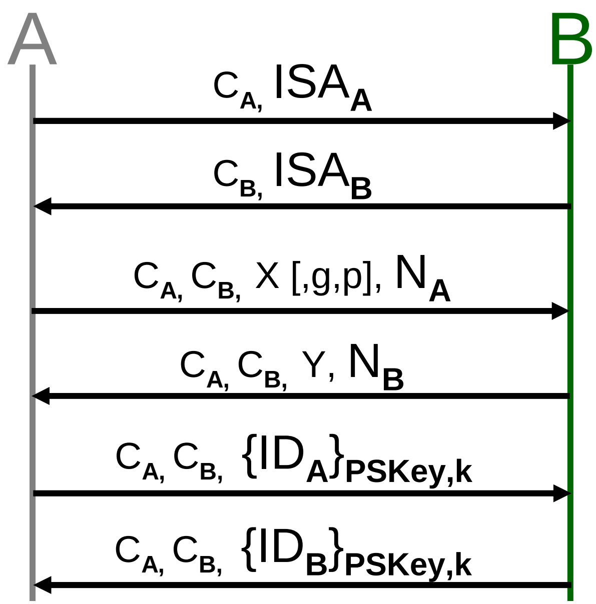
- Key exchange is unauthenticated (man-in-the-middle)

Authenticated Key Exchange



If A and B share a key PSKey then they may use it, together with k (the D-H result) to encrypt and authenticate the ID (and other param).

Main Mode for shared key: Negotiation, Key Derivation



$SKey = h_{PSKey}(\ N_A \mid N_B)$ Key control!

$SKey_d = h_{SKey}(\ k \mid C_A \mid C_B \mid 0\ )$

$SKey_a = h_{SKey}(\ SKey_d \mid k \mid C_A \mid C_B \mid 1\ )$

$SKey_e = h_{SKey}(\ SKey_d \mid k \mid C_A \mid C_B \mid 2\ )$

$Hash_A = h_{SKeya}(\ X \mid Y \mid C_A \mid C_B \mid\ ISA_A \mid ID_A\ )$

$\{ID_A\}_{PSKey,k} = (\ ID_A \mid Hash_A\ )$

ISA_A, ISA_B are ISAKMP SA data, used by IKE to negotiate encryption and hash algorithms and authentication method

The negotiated parameters pertain only to the ISAKMP SA and not to any SA that ISAKMP may be negotiating on behalf of other services.

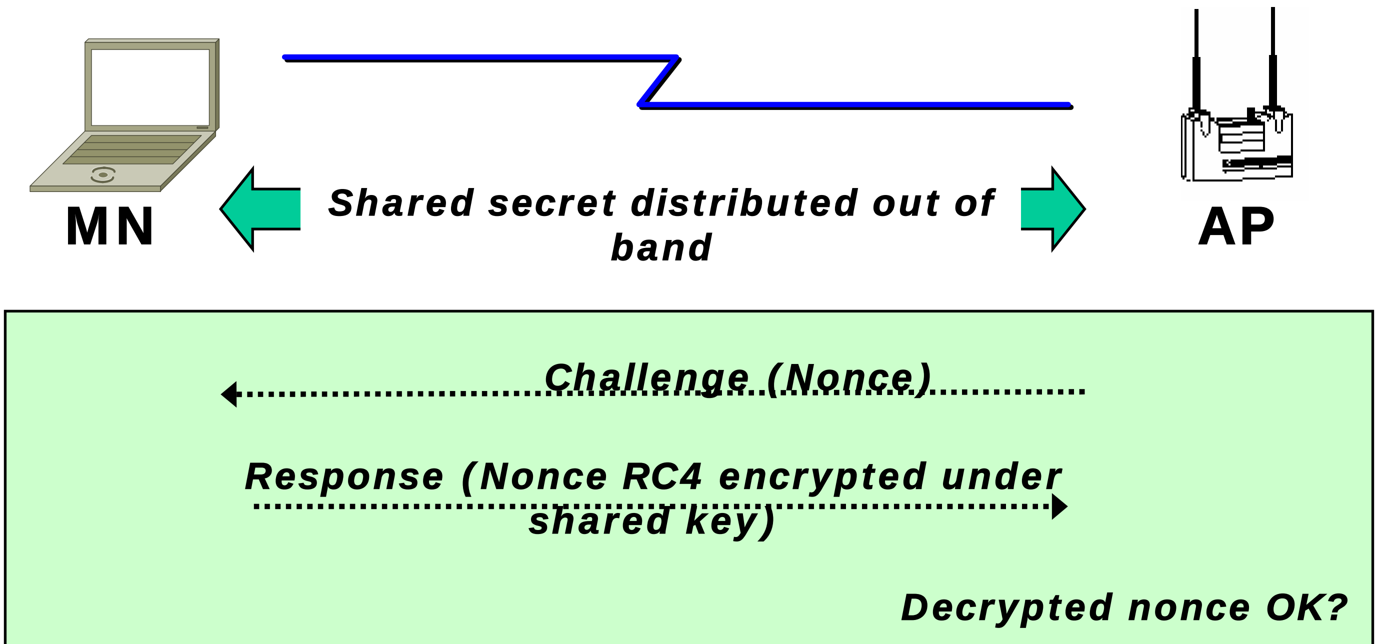
IKEv2 – What's new? (1/2)

- Number of authentication modes reduced : Only one public-key based and a pre-shared secret based method
- Establishes two types of SAs (IKE-SA and Child-SAs)
- User identity confidentiality supported
  - Active protection for responder
  - Passive protection for initiator
- Number of roundtrips are reduced (piggy-packing SA establishing during initial IKE exchange)
- Better DoS protection (optional)
- NAT handling covered in the core document
- Other improvements …

IKEv2 – What's new? (2/2)

- Legacy authentication and IPSRA results have been added to the core document.
  This allows OTP and other password-based authentication mechanisms to be used
- To support legacy authentication a two-step authentication procedure is used.
- Traffic Selector negotiation improved
- IPComp still supported
- Configuration exchange included which allows clients to learn configuration parameters similar to those provided by DHCP.
- EC-groups supported

Wireless Equivalence Privacy (WEP) Authentication

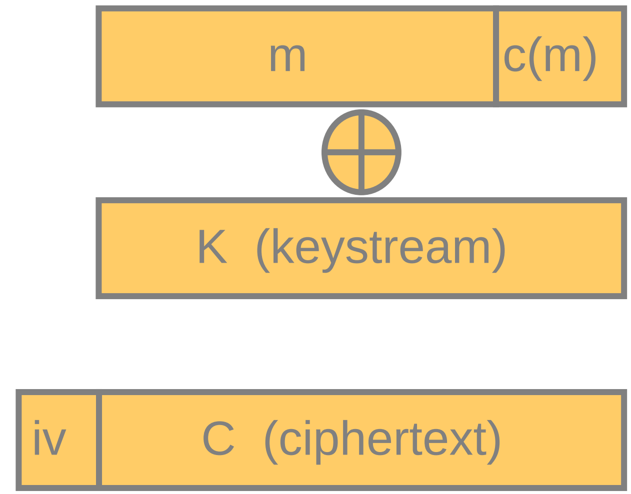


802.11 Authentication Summary:

- Authentication key distributed out-of-band
- Access Point generates a “randomly generated” challenge
- Station encrypts challenge using pre-shared secret

WEP in brief:

Sender and receiver share a secret key k.



To transmit m:

- Compute a checksum c(m), append to m:
$$M = (m \mid c(m))$$
- Pick iv, and generate a keystream
$$K := rc4(iv,k)$$
- ciphertext = C := $M \oplus K$
- Transmit (iv | ciphertext)

Recipient:

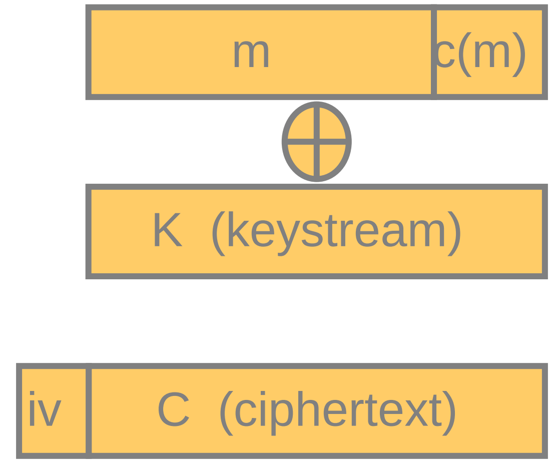
- Use the transmitted iv and k to generate K = rc4(iv,k)
- $\langle m',c'\rangle := C \oplus K \overset{\text{ifOK}}{=} (M \oplus K) \oplus K = M$
- If c' = c(m'), accept m' as the message transmitted

Decryption Dictionaries

- AP sends challenge
- MN responds with challenge, encrypted with the shared secret k
- Intruder: has now both the plaintext and the ciphertext!
- pings, mail $\Rightarrow$ intruder knows one pair of ciphertext and the corresponding plaintext, for one iv.
- $C := M \oplus K \Rightarrow$ he knows $K = M \oplus C$.

  Note that he does not learn the value of the shared secret k.
- He stores (iv, K) in a table (dictionary).
- This table is 1500 * 2^24 bytes = 24 GB
- Independent of 40-bit keys or 104-bit keys
- Next time he sees iv in the table, look up K and calculate $M = C \oplus K$
- Size of table depends only on the number of different iv.
- If the cards reset iv to 0, the dictionary will be small!

Attacks involving keystream reuse (collision)



If m1 and m2 are both encrypted with K,

$\Rightarrow$ C1 $\oplus$ C2 = m1 $\oplus$ K $\oplus$ m2 $\oplus$ K
= m1 $\oplus$ m2

$\Rightarrow$ intruder knows $\oplus$ of two plaintexts!

Pattern recognition methods:
know m1 $\oplus$ m2 $\Rightarrow$ recover m1, m2.

K = rc4(iv,k).

k changes rarely and shared by all users

Same iv $\Rightarrow$ same K $\Rightarrow$ collision

iv cleartext $\Rightarrow$ intruder can tell when collision happens.

There are 2^24, (16 million) possible values of iv.

50% chance of collision after only 4823 packets!

Cards reset iv to 0 on each activation (then iv++)

$\Rightarrow$ low iv values get reused often

Message Modification

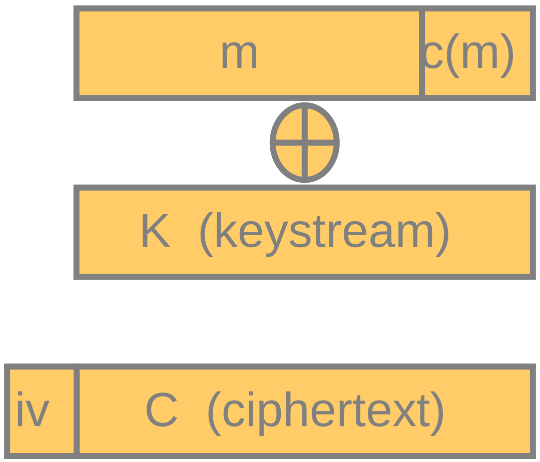
- Assume IV and C are known to intruder.
- Intruder wants the receiver to accept fake message

  f = m ⊕ d for some chosen d

  ($$ in a funds transfer)
- D := (d | c(d)), C' := C ⊕ D
- Transmit (iv,C') to the receiver.
- Receiver checks:

  C' ⊕ K = (C ⊕ D) ⊕ K = K ⊕ (M ⊕ D) ⊕ K = M ⊕ D =

  (m ⊕ d | c(m) ⊕ c(d)) = (m ⊕ d | c(m ⊕ d)) = (f | c(f))
- OK!



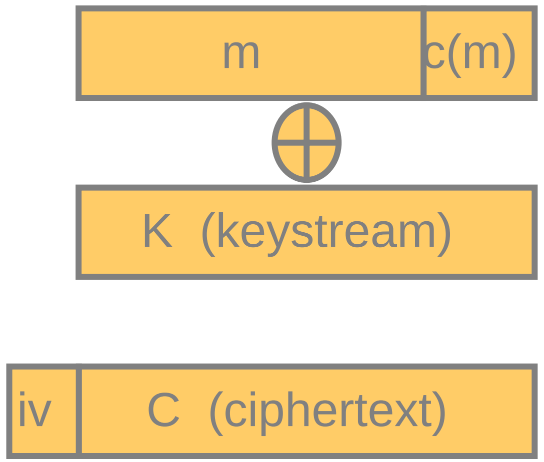
Message Injection

Assume: Intruder

knows a plaintext,

and corresponding encryption

(pings or spam provide this)



The encrypted packet is (iv,C),

plaintext is (m | c(m)),

intruder computes

$$K = C \oplus (m \mid c(m)).$$

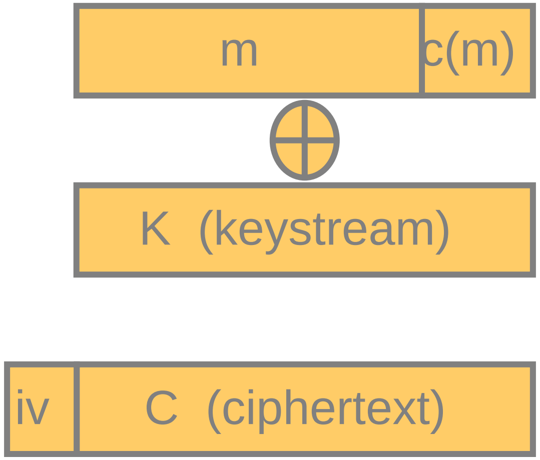
Now he can take any message F, compute c(F),

and compute $C' = (f \mid c(f)) \oplus K$.

Transmits (iv,C').

Authentication Spoofing

- Once intruder sees a challenge/response pair for a given key k, he can extract iv and K .



- Now he connects to the network himself:
  - AP sends a challenge m' to intruder
  - Intruder replies with iv, <m',c(m')> $\oplus$ K
  - This is in fact the correct response, so AP accepts intruder
  - Without knowing k

Reaction Attacks

Assume the packet to be decrypted is a TCP packet

Do not need connection to the Internet

Use the fact: TCP checksum invalid $\Rightarrow$ silently dropped

TCP checksum is correct $\Rightarrow$ ACK

We can iteratively modify a packet and check if the TCP checksum valid

Possible to make the TCP checksum valid or invalid exactly when any given bit of the plaintext message is 0 or 1

So each time we check the reaction of the recipient to a modified packet, we learn one more bit of the plaintext

Current Status of WLAN Security

802.11 Task Group

- Short-term solution: TKIP (Temporal Key Integrity Protocol)
  - Idea: **use existing hardware** by software-/firmware-update only
  - 128 bit key, 48 bit Extended IV, IV sequencing rules (~10^10 years)
  - Key mixing function (creates new seed for RC4 for each packet)
  - New Message Integrity Code
- Long-term solution: Authentication and key management: 802.1X "Port-based access control": MN ↔ AP ↔ AS
  - Mutual authentication: MN ↔ AS (authentication server)
  - Establishment of individual per-session keys MN ↔ AP
  - AES-CCMP (AES-Counter-Mode/CBC-MAC protocol)
  - Requires new hardware

WEP Security: Lessons

- WEP designers selected well-regarded algorithms, such as RC4
- But used them in insecure ways
- $\Rightarrow$ security protocol design is very difficult
  - best performed with an abundance of caution,
  - supported by experienced cryptographers and security protocol designers
  - use verification tools!

Contents

Internet Layers, Basics

Management, Implementation or Design Errors

IETF Groups and Activities

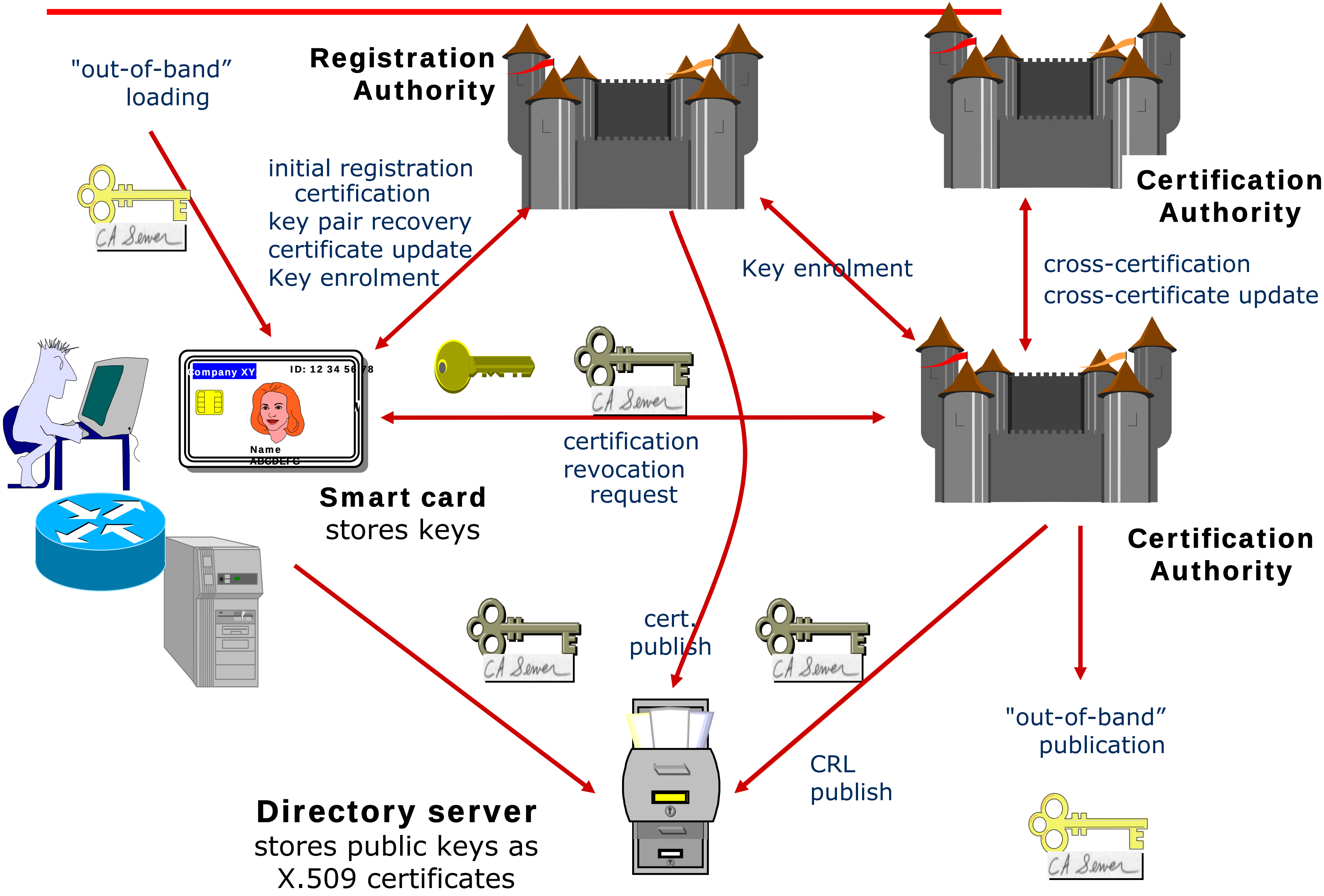
Security Protocols: Kerberos, AAA, IPsec, IKE, WLAN

Public-Key Infrastructure (PKI)

High-level Protocol Specification Language (HLPSL)

Outlook: MobileIP, DRM

Basic model: basic protocols -- Simplified User‘s View



The minimal Public Key Certificate

A data structure that binds
- a subject
- a public key

Binding done by trusted CA:
- verifies the subject's identity
- signs the certificate

PKCertificate ::=

{

Subject Name
Subject Public Key

Signature

}

X.509 Public Key Cert V.1

AVISPA

Version 1 of 1988

To uniquely identify cert. Never reused

X.500 DN of CA, e.g., {C=de, S=.., O=Comp}

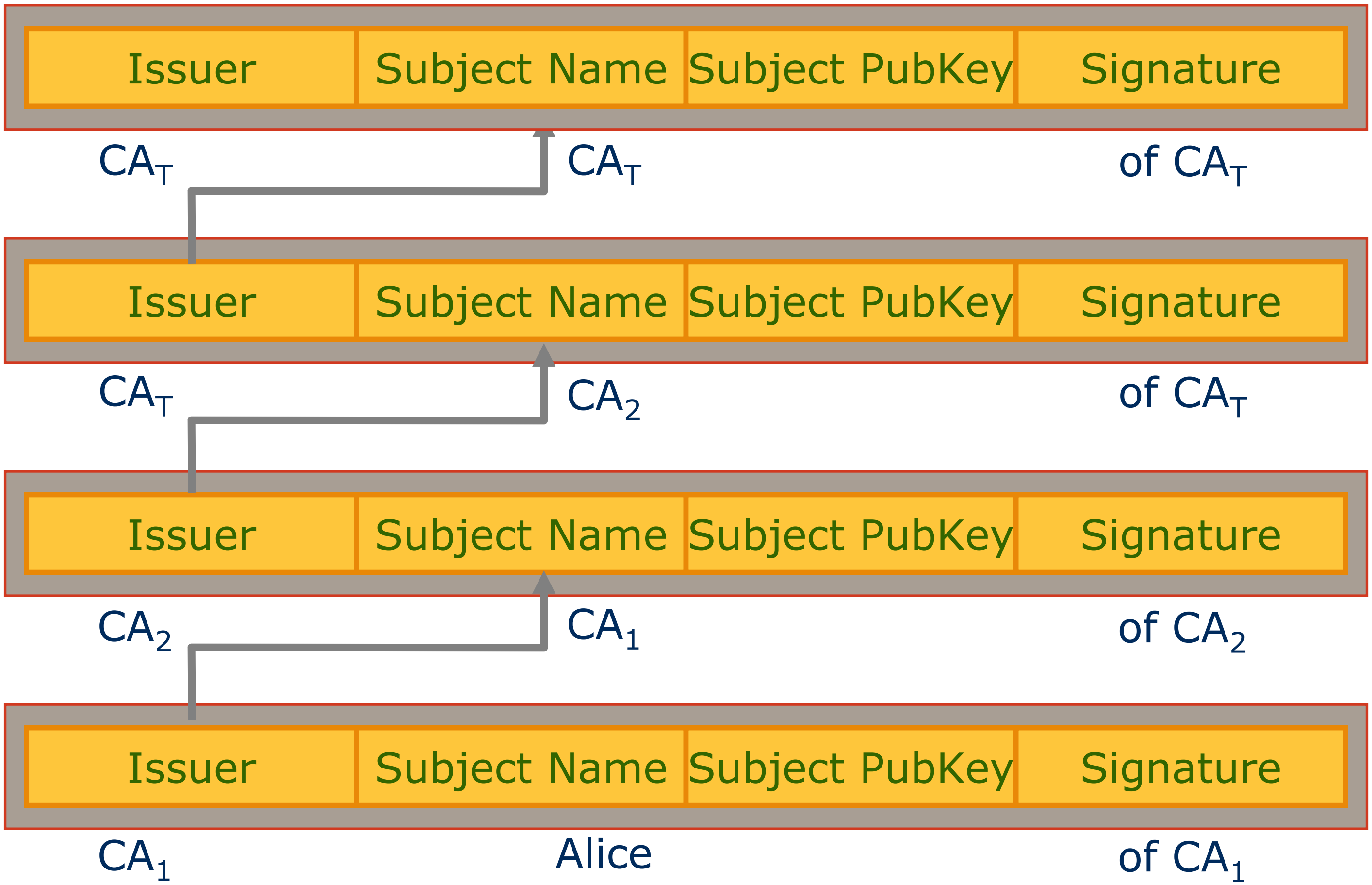
YYMMDD; HHMM{SS}: ("Y2K" problem)

AlgorithmID is a pair: encrypt + hash (+ opt. parameters)

```
PKCertificate :: =
{
  Version = 0 ("1")
  Serial Number
  Signature AlgorithmID
  Issuer
  Validity (Lifetime)
     Not Before
     Not After
  Subject Name
  Subject Public Key
     AlgorithmID
     Key value
  --------------------------
  Signature
}
```

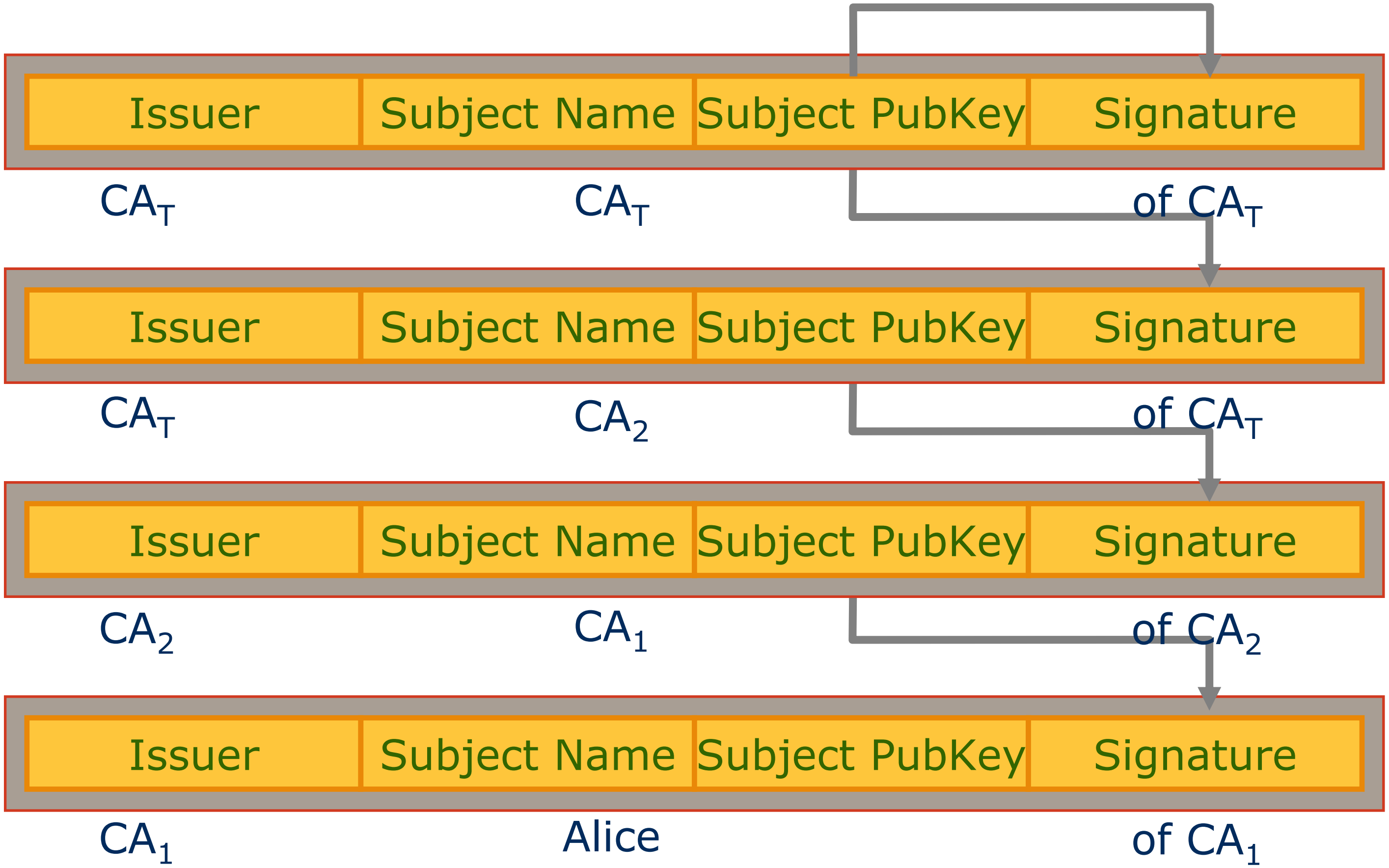
- Format of certificate is ASN.1
- DER (Direct Encoding Rules) produces octets for transmission

Path Construction and Path Discovery



Easy in hierarchical PKIs; if not: may need construct several paths

Verify the Certificate: Path Validation



Relying on a trusted/local copy of the root certificate, by induction:

Issuer owns the claimed PubKey, CA_2 , CA_1 trustworthy.

Check Lifetime, Policies and Revocation Lists!

X.509 Public Key Cert V.2

Version 2 of 1992

There may be several "Trust-my-Cert Inc." worldwide, or several "Bob Hope" in our company

If "Bob Hope" leaves our company and a new "Bob Hope" is hired, how to make sure that the new one does not inherit the old authorizations?

To uniquely identify Issuer

To uniquely identify Subject

Nobody uses that. There are better solutions.

```
PKCertificate ::=
{
  Version = 1 ("2")
  Serial Number
  Signature AlgorithmID
  Issuer
  Validity (Lifetime)
    Not Before
    Not After
  Subject Name
  Subject Public Key
    AlgorithmID
    Key value
  Issuer Unique ID
  Subject Unique ID
  ---------------------
  Signature
}
```

X.509 Public Key Cert V.3

AVISPA

Version 3 of 1998

UCTTime: YYMMDD: If YY < 50 then add 2000
else add 1900
OR
Generalized Time: YYYYMMDD

Standard extensions for: KeyID,
Key usage intention / restriction,
subject/issuer alternate names or attributes
(DNS name, email addr., URL, IP addr.)
policies
certification path
Private Extensions also possible

```
PKCertificate :: =
{
  Version = 2 ("3")
  Serial Number
  Signature AlgorithmID
  Issuer
  Validity (Lifetime)
     Not Before
     Not After
  Subject Name
  Subject Public Key
     AlgorithmID
     Key value
  Extensions
     Extension1
     Extension2
  --------------------
  Signature
}
```

X.509 Public Key Cert V.3

Problems:

Fields: Type (critical | non critical) value

- Issuer does not only check your identity, it also sees what you are allowed
- Size of cert (say, in wireless applications)
- Do not need all extensions always
- More extensions → earlier need to revocate

```
PKCertificate :: =
{
  Version = 2 ("3")
  Serial Number
  Signature AlgorithmID
  Issuer
  Validity (Lifetime)
     Not Before
     Not After
  Subject Name
  Subject Public Key
     AlgorithmID
     Key value
  Extensions
     Extension1
     Extension2
  ------------------
  Signature
}
```

Reasons for Revocation

- Compromise of subject's private key
- Change in subject name
- Change in authorizations given in certificate
- Change of subject's affiliation
- Violation of CAs policies
- Compromise of CAs private key
- Termination of entity, etc.

- Need to inform all users by some means.
- Note: Revocation before expiry!

How to check revocation status?

- Options from PKIX
  - OCSP (Online certificate status protocol)
  - OCSP with extensions:
    - Delegated Path Validation (DPV)
    - Delegated Path Discovery (DPD)
  - DPD or DPV are also possible without OCSP
  - Simple Certificate Verification Protocol (SCVP)

Contents

Syntax: TLA Normal Form

AVISPA

A TLA formula in normal form is:

$\exists \ldots$ st_pred $\wedge \Box$ ((event $\Rightarrow$ tr_pred) $\wedge$ (event $\Rightarrow$ tr_pred) $\wedge \ldots$)

Our HLPSL is close to this TLA form.

Note: conjunction of TLA normal forms is (wlog) normal form

Conjunction is parallel composition of module/role instances

Two types of variables:

- flexible variables (state of the system)
- rigid variables (parameters, constants, may be bound at some point later)

TLA Example

AVISPA

$V=\{x,y\}$

Let $Prg(x) = (x=0) \wedge \square (x' \neq x \Rightarrow x'=x+1)$

Then the following traces are in Tr(Prg):

(0,3), (0,7), (0,5), (0,6), (0,6), …

(0,4), (1,4), (2,4), (3,0), (4,7), …

(0,0), (1,1), (2,2), (3,3), (4,4), …

(0,4), (0,3), (1,2), (1,1), (2,0), …

The program talks about variable x only, y may behave arbitrarily.

All traces of Prg are generated by the following "symbolic trace":

(0,*), (1,*), (2,*), (3,*), (4,*), …

by: taking a prefix (including all)

introducing any number of x-stuttering steps,

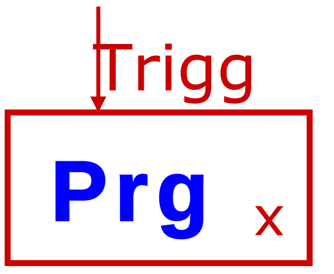
repeating (x,*) any number of times (even infinite)

replacing the do-not-cares "*" by any values of y

simpl. HLPSL Example

$Prg(x) = (x=0) \wedge \Box (x' \neq x \Rightarrow x'=x+1)$

Using a signal "Trigg":



role Prg(Trigg,x) def=

owns x

init x = 0

transition

Trigg $\Rightarrow$ x' = x +1

The var x is modified only by Prg,

but it may be seen outside.

TLA Example

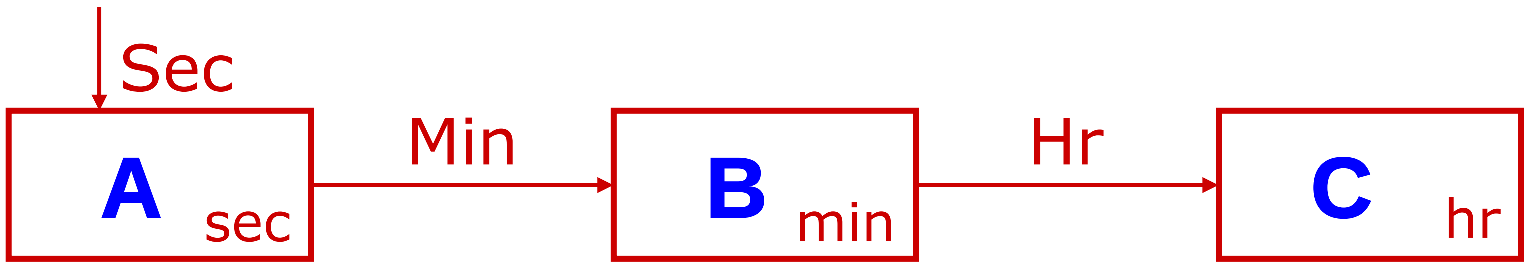
$V=\{x,y\}$

Let $Prg(x) = (x=0) \wedge \square (x' \neq x \Rightarrow x'=x+1)$

Let $New(x,y) := Prg(x) \wedge Prg(y)$

Exercise: What are the traces of this program?

TLA Example, modeling channels



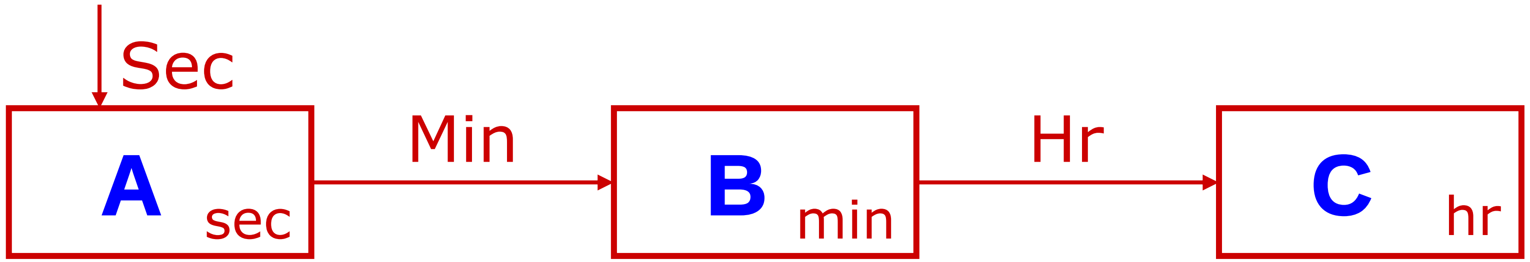
V={sec:{0,...59} ,min :{0,...59},hr :{0,...23} }

Sec := (sec'≠sec), etc. *Events*

Clock: = A $\wedge$ B $\wedge$ C

$$
\begin{aligned}
A &:= (sec = 0) \wedge \Box\, (\wedge\ Sec \Rightarrow sec' = sec + 1 \pmod{60} \\
&\qquad \wedge\ Sec \wedge sec' = 0 \Rightarrow Min) \\
B &:= (min = 0) \wedge \Box\, (\wedge\ Min \Rightarrow min' = min + 1 \pmod{60} \\
&\qquad \wedge\ Min \wedge min' = 0 \Rightarrow Hr) \\
C &:= (hr = 0) \wedge \Box\, (\ Hr \Rightarrow hr' = hr + 1 \pmod{24})
\end{aligned}
$$

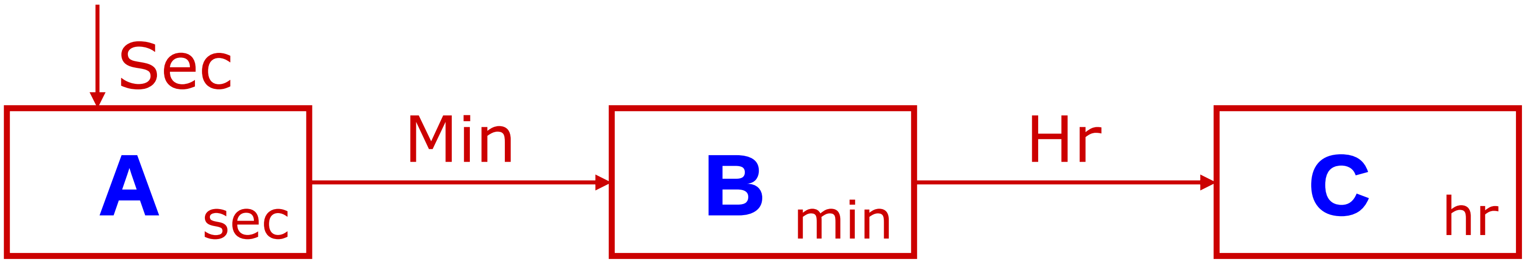
simpl. HLPSL Example, the clock



Clock: = A ∧ B ∧ C

```
Role A(Sec,sec,Min) :=
Init sec = 0
Trans Sec ⇒ sec' = sec +1 (mod 60)
      Sec ∧ sec' = 0 ⇒ Min
```

Implementing the clock with local variables



Who owns the minutes?

Separate Min + min, etc

Redefine Min := v_Min′ ≠v_Min

```
role A(Sec,sec,Min) :=
owns sec, Min
init sec = 0
trans Sec ⇒ sec' = sec +1
      Sec ∧ sec' = 0 ⇒ Min
```

```
A = (sec = 0) ∧ □ (∧ Sec ⇒ sec' = sec +1
                   ∧ Sec ∧ sec' = 0 ⇒ Min
                   ∧ sec ≠ sec' = 0 ⇒ Sec
                   ∧ Min ⇒ Sec ∧ sec' = 0 )
```

Basic Roles: Semantics

```
role Basic_Role (…) :=
  owns {θ: Θ}
  local {ε}
  init Init
  accepts Accept
  transition
    event1 ⇒ action1
    event2 ⇒ action2
    …
end role
```

```
θ(Basic_Role)      := θ
Trigg(Basic_Role) := event1 ∨ event2 ∨ …         %% This is also an event!
Init(Basic_Role)  := Init
Accept(Basic_Role):= Accept
Mod(x,Basic_Role) := ∨ {eventi | x' ocurrs in actioni (or in a LHS channel val)}
Step(Basic_Role)  := Trigg(Basic_Role) ∧ (event1 ⇒ action1) ∧ (event2 ⇒ action2) ∧ …
TLA(Basic_Role)   := ∃ ε { Init ∧ □ [ (event1 ⇒ action1) ∧ (event2 ⇒ action2) ∧ …
                        ∧ (∧ _(θ∈Θ) θ'≠ θ ⇒ Mod(θ,Basic_Role)) ] }
```

Semantic of Composed Roles: flattening approach

$A \otimes B$ = Composition(A,B):

Parallel, Sequential (+taking ownership, hiding)

flatten: hlpsl-Programs $\rightarrow$ hlpsl-Programs

For basic roles:

flatten(A) = A

For composed roles:

flatten($A \otimes B$) = arrange(flatten(A),flatten(B))

Composed Roles: parallel

```
role Par_Role ( parameters; variables, channels) :=
   % Parallel Composition of A and B
  owns {θ:Θ}
  local {ε}
  init Init
  accepts Accept
    A ∧ B
end role
```

θ(Par_Role) := θ(A) ∪ θ(B) ∪ θ

Trigg(Par_Role) := Trigg(A) $\vee$ Trigg(B)

Init(Par_Role) := Init(A) $\wedge$ Init(B) $\wedge$ *Init*

Accept(Par_Role) := Accept(A) $\wedge$ Accept(B) $\wedge$ *Accept*

Mod(x,Par_Role) := Mod(x,A) $\vee$ Mod(x,B)

TLA(Par_Role) := $\exists$ ε {Init(Par_Role) $\wedge$ TLA(A) $\wedge$ TLA(B)

$\wedge \Box$ [($\wedge$ _($\theta \in \Theta$) $\theta' \neq \theta \Rightarrow$ Mod(θ, Par_Role))] }

Composed Roles: sequential

```
role Seq_Role ( parameters; variables, channels) :=  %Sequential Composition of A and B
  owns {θ:Θ}
  local {ε}
  init Init
  accepts Accept
    A ; B
end role
```

```
Trigg(Seq_Role)  := (flag  = 0 ∧ Trigg(A)) ∨ (flag  = 1 ∧ Trigg(B))
Init(Seq_Role)   := flag  = 0 ∧ Init(A)   ∧  Init
Accept(Seq_Role) := Accept(B) ∧  Accept
Mod(x,Seq_Role)  := (flag  = 0 ∧ Mod(x,A))  ∨ (flag  = 1 ∧ Mod(x,B))
TLA(Seq_Role)    := ∃ ε,flag {Init(Seq_Role)
                     ∧ □ [(Trigg(A) ⇒flag=0) ∧ (Trigg(B) ⇒flag=1)
                          (flag' ≠ flag =>   flag' = 1
                                           ∧ Accept_A’
                                           ∧ Init_B')
```

NSPK: Needham-Schroeder Public-Key Protocol (1978)

Alice → Bob: {Na,A}Kb

Bob → Alice: {Na,Nb}Ka

Alice → Bob: {Nb}Kb

Ka, Kb: public key of Alice, Bob

Na, Nb: Nonce (= Number used only Once,
i.e. a fresh random value)

Security goals:

Alice freshly authenticates Bob

Bob freshly authenticates Alice

Jorge Cuellar, David von Oheimb, Sebastian Mödersheim, Luca Viganò

NSPK: Alice in HLPSL

```
role Alice (A, B: agent, Ka, Kb: public_key,
               SND, RCV: channel (dy))
played_by A def=
  local State : nat, Na: text (fresh), Nb: text
  init State = 0
  transition
    0.  State = 0 /\ RCV(start) =|>
        State'= 2 /\ SND({Na'.A}_Kb)
                  /\ witness(A,B,na,Na')
    2.  State = 2 /\ RCV({Na.Nb'}_Ka) =|>
        State'= 4 /\ SND({Nb'}_Kb)
                  /\ request(A,B,nb,Nb')
end role
```

NSPK: Bob in HLPSL

```
role Bob(A, B: agent, Ka, Kb: public_key,
         SND, RCV: channel (dy))
played_by B def=
  local State : nat, Na: text, Nb: text (fresh)
  init State = 1
  transition
    1.  State = 1 /\ RCV({Na'.A}_Kb) =|>
        State'= 3 /\ SND({Na'.Nb'}_Ka)
                  /\ witness(B,A,nb,Nb')
    3.  State = 3 /\ RCV({Nb}_Kb) =|>
        State'= 5 /\ request(B,A,na,Na)
end role
```

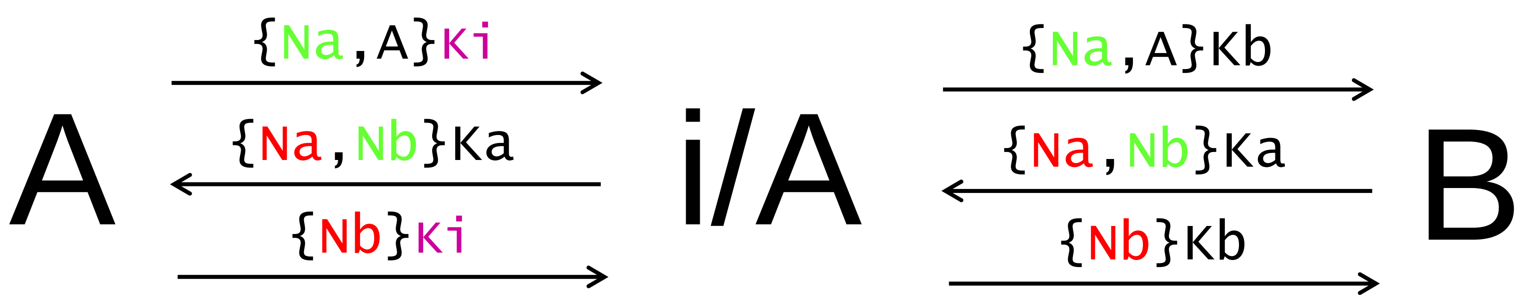
NSPK: Sessions in HLPSL

```
role Session(A, B: agent, Ka, Kb: public_key) def=
  local SA, RA, SB, RB: channel (dy)
  composition
      Alice(A,B,Ka,Kb,SA,RA)
    /\ Bob  (A,B,Ka,Kb,SB,RB)
end role

role Environment() def=
    const a, b: agent, ka, kb, ki: public_key
    knowledge(i) = {a, b, ka, kb, ki, inv(ki)}
    composition
        Session(a,b,ka,kb)
     /\ Session(a,i,ka,ki)
     /\ Session(i,b,ki,kb)
end role
```

NSPK: Lowe's attack (1995)

```
goal
  Alice authenticates Bob on nb
  Bob authenticates Alice on na
end goal

Environment()
```

Man-in-the-middle attack:



NSPK: OFMC output

```
% OFMC
% Version of 2005/01/18
SUMMARY
  UNSAFE
DETAILS
  ATTACK_FOUND
PROTOCOL
  NSPK.if
GOAL
  b authenticates a on na
%Request b a na Na(1)
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.05s
  visitedNodes: 27 nodes
  depth: 3 plies
```

```
ATTACK TRACE
i -> (a,6): start
(a,6) -> i: {Na(1),a}ki
i -> (b,3): {Na(1),a}kb
(b,3) -> i: {Na(1),Nb(2)}ka
i -> (a,6): {Na(1),Nb(2)}ka
(a,6) -> i: {Nb(2)}ki
i -> (b,3): {Nb(2)}kb

% Reached State:
% Request b a na Na(1)
% Witness b a nb Nb(2)
% Witness a i na Na(1)
...
```

Contents

Internet Layers, Basics

Management, Implementation or Design Errors

IETF Groups and Activities

Security Protocols: Kerberos, AAA, IPsec, IKE, WLAN

Public-Key Infrastructure (PKI)

High-level Protocol Specification Language (HLPSL)

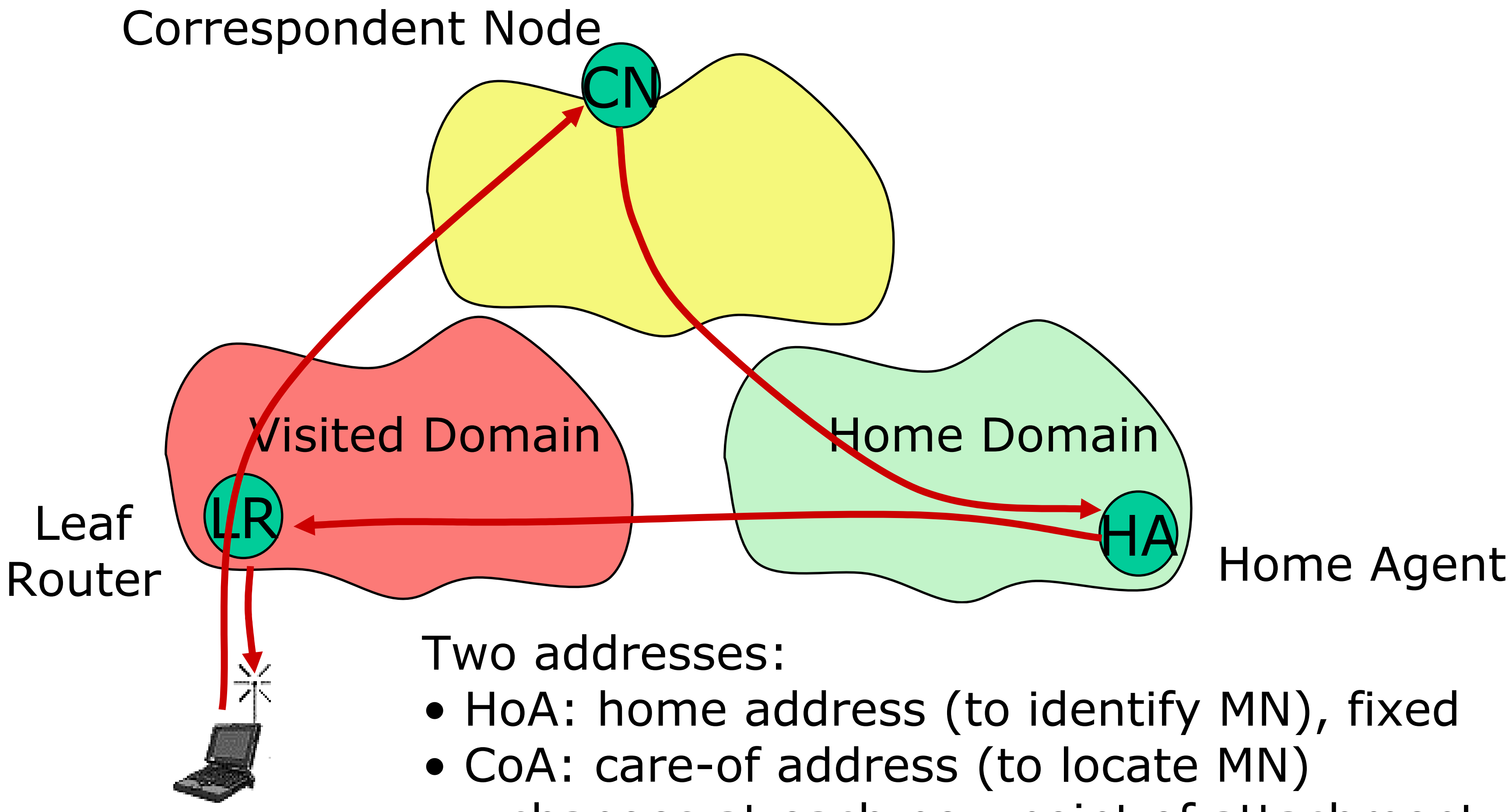
Outlook: MobileIP, DRM

IP mobility

- MN moves from one IP address to another
  - moves between network coverage areas or media types,
  - its logical point of network access changes, or
  - a whole sub-network moves (not covered in MobileIP).
- Mobility protocols
  - maintain existing connections over location changes
  - ensure that MN can be reached at its new location.
- Location management: mechanism for informing other nodes about MN's current address.

  Approaches:
  - a directory service where MN's location is maintained or
  - direct notifications to the nodes that need to know about the new location.

Mobility Management

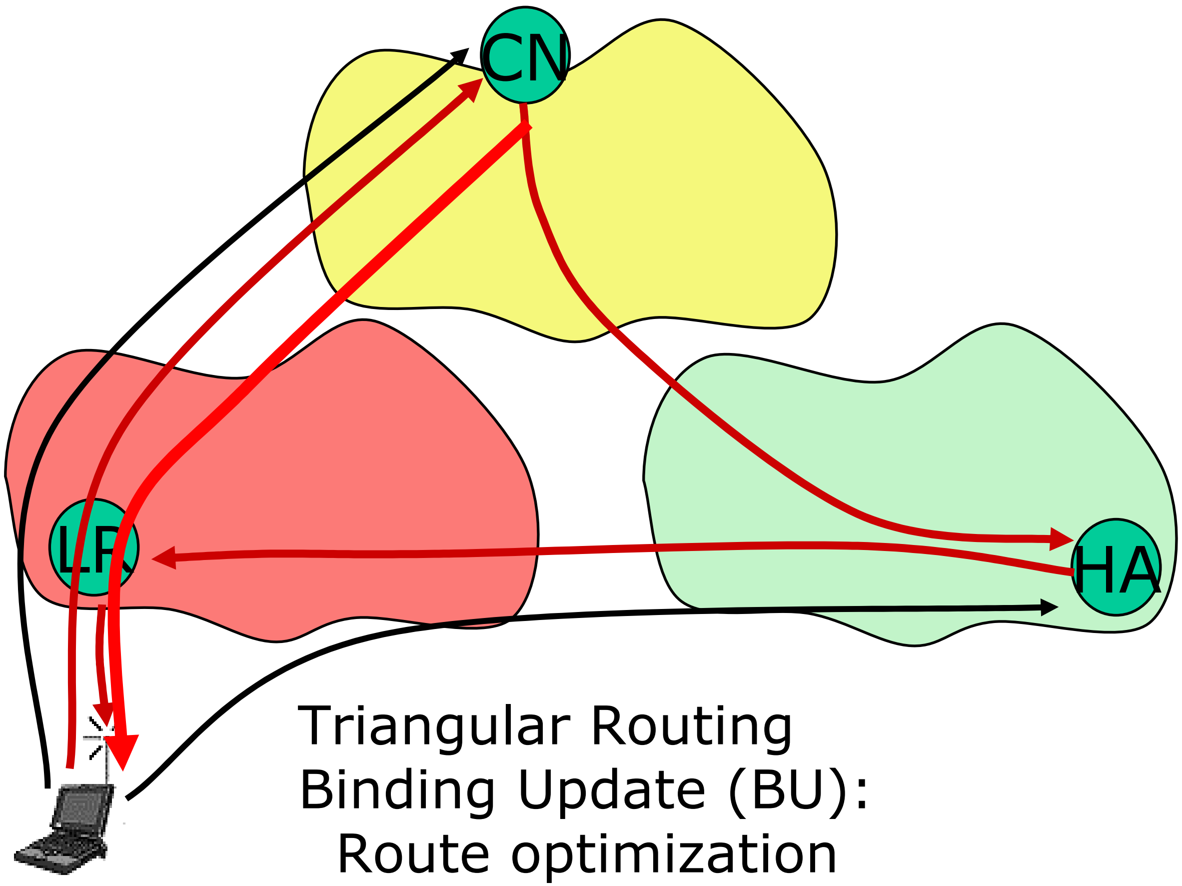


Two addresses:

- HoA: home address (to identify MN), fixed
- CoA: care-of address (to locate MN) changes at each new point of attachment.

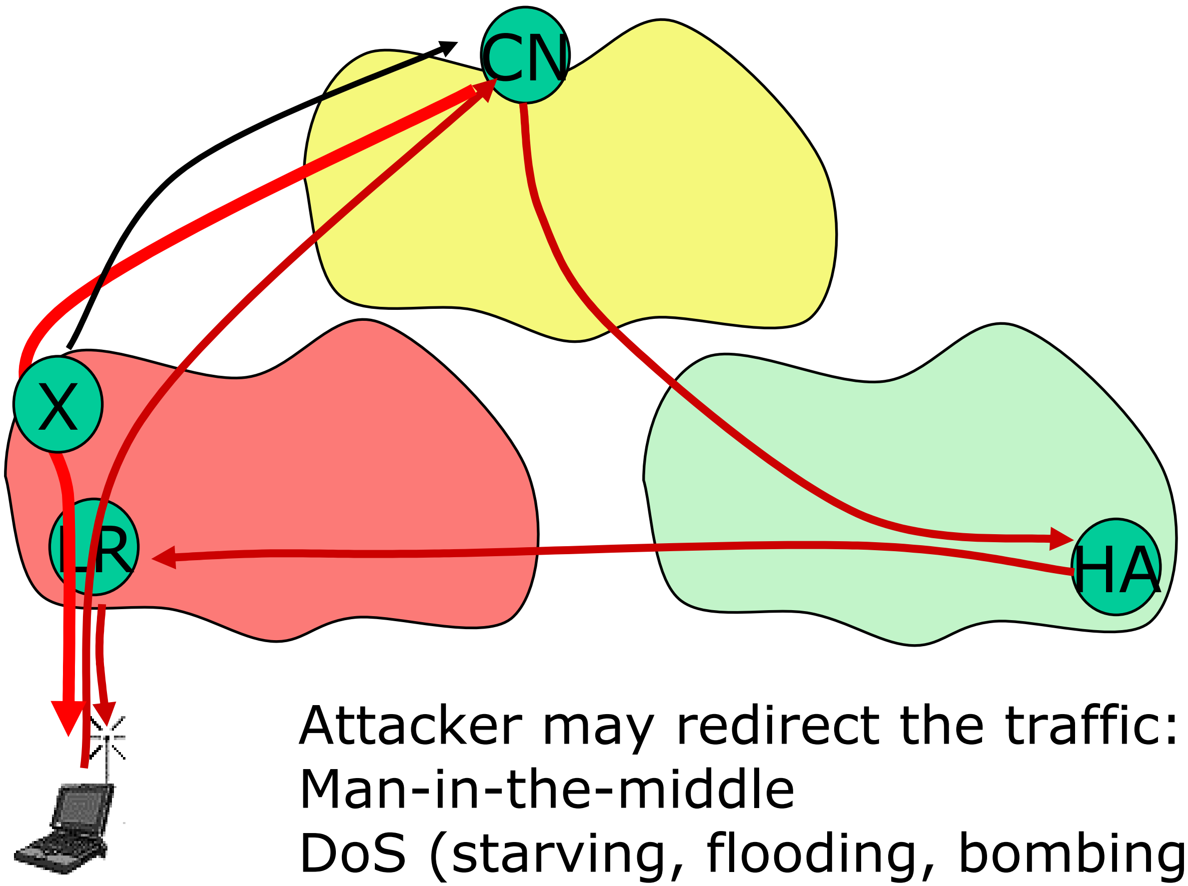
How are such „Bindings" created / modified?

Mobility Management



Triangular Routing
Binding Update (BU):
Route optimization

Security Problems



Attacker may redirect the traffic:
Man-in-the-middle
DoS (starving, flooding, bombing)

IPv6

- Address size increased from 32 to 128 bits.
- Auto-configuration to generate locally CoA:

Routing prefix	(e.g. MAC Address)

- 64-bit routing prefix, which is used for routing the packets to the right network
- 64-bit interface identifier,
  - which identifies the specific node
  - can essentially be a random number.

Mobile IPv6

- MN is identified by a home IP address (HoA)
- IP addresses in MIPv6 can identify either a node or a location on the network, or both.
- Home Agent (HA, a router)
  - acts as MN's trusted agent and
  - forwards IP packets between MN's correspondent nodes (CN) and its current location, the care-of address (CoA)
- The MIPv6 protocol also includes a location management mechanism called Binding Update (BU).
- When moving, MN sends BUs to CN and HA to notify them about the new location so that they can communicate directly
- MN may also be triggered to sending a BU when it receives a packet from a new CN via HA.

Binding Updates

- MN and HA have a permanent trust relationship and a preconfigured security association for encrypted and authenticated communication.
- MN informs HA about its location via this secure tunnel.
- MN and its HA can co-operate to send BUs to CNs, with which they often have no pre-existing relationship.
- CN stores the location information in a binding cache entry, which needs to be refreshed regularly by sending a new BU.

Threats

- Misinform CN about MN’s location
  - Redirect packets intended for MN
    - compromise secrecy and integrity
    - denial-of service (MN unable to communicate).
- Send bogus BUs, may use own address as CoA
  - impersonate MN.
    - hijack connections between MN and its CNs, or
    - open new ones.
- Redirect packets to a random or non-existent CoA (DoS).

  MN has to send a new BU every few minutes to refresh the binding cache entry at CN.

Preventing Replay Attacks

- Time stamps would be problematic because MNs may not be able to maintain sufficiently accurate clocks.
- Sequence-numbered BUs, on the other hand, could be intercepted and delayed for later attacks.
- A nonce-based freshness mechanism seems practical because many related authentication and DoS protection mechanisms use nonces anyway.

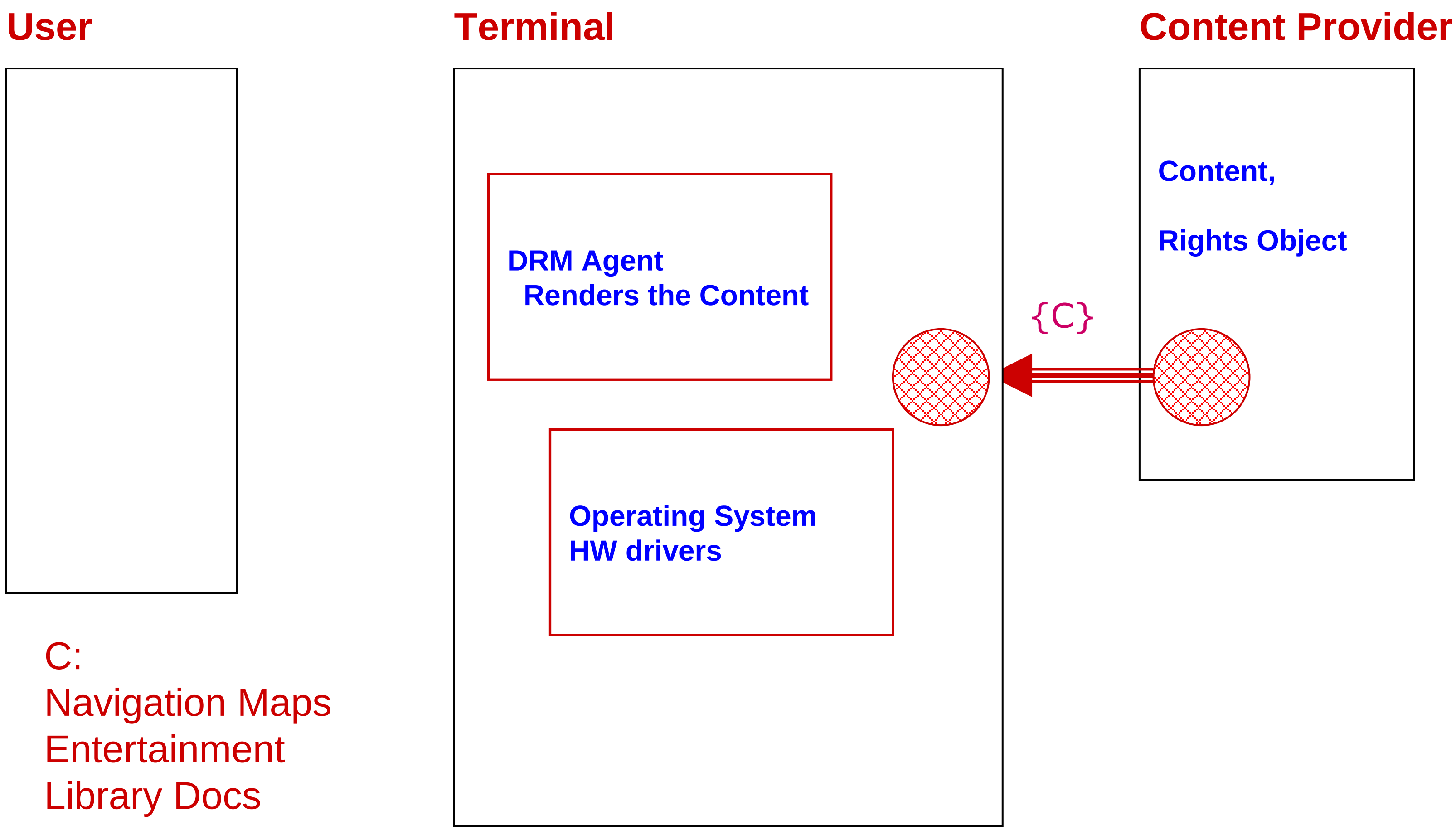
Why not IPsec, IKE, and PKI?

BU authentication: could use strong generic authentication mechanisms and infrastructure: IPsec, IKE, and PKI.

- Overhead too high for low-end mobile devices and for a network-layer signaling protocol.
- Internet mobility protocol should allow anyone to become MN and it must allow all Internet nodes as CNs

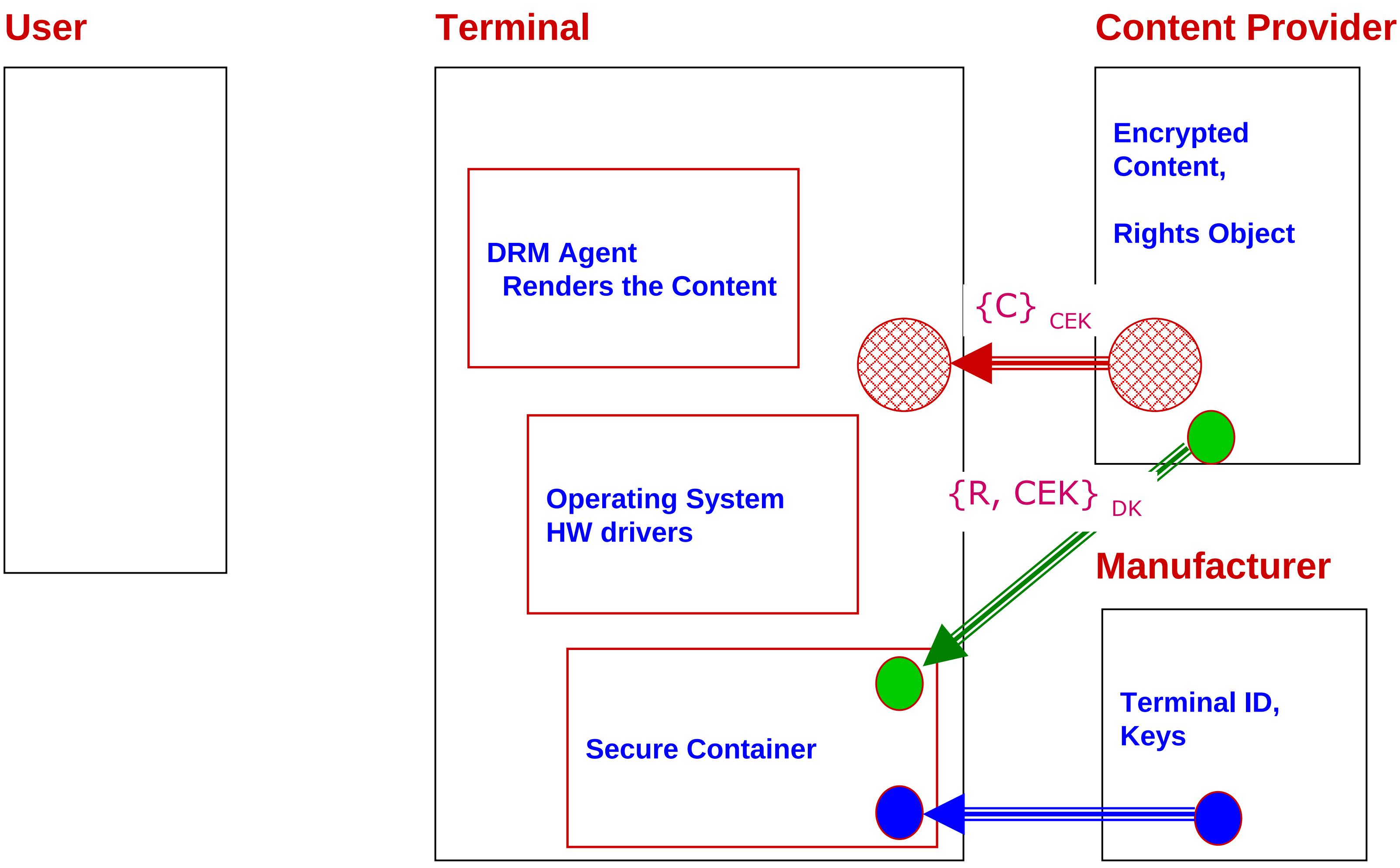
  $\Rightarrow$ A single PKI must cover the entire Internet.

DRM: The Goal

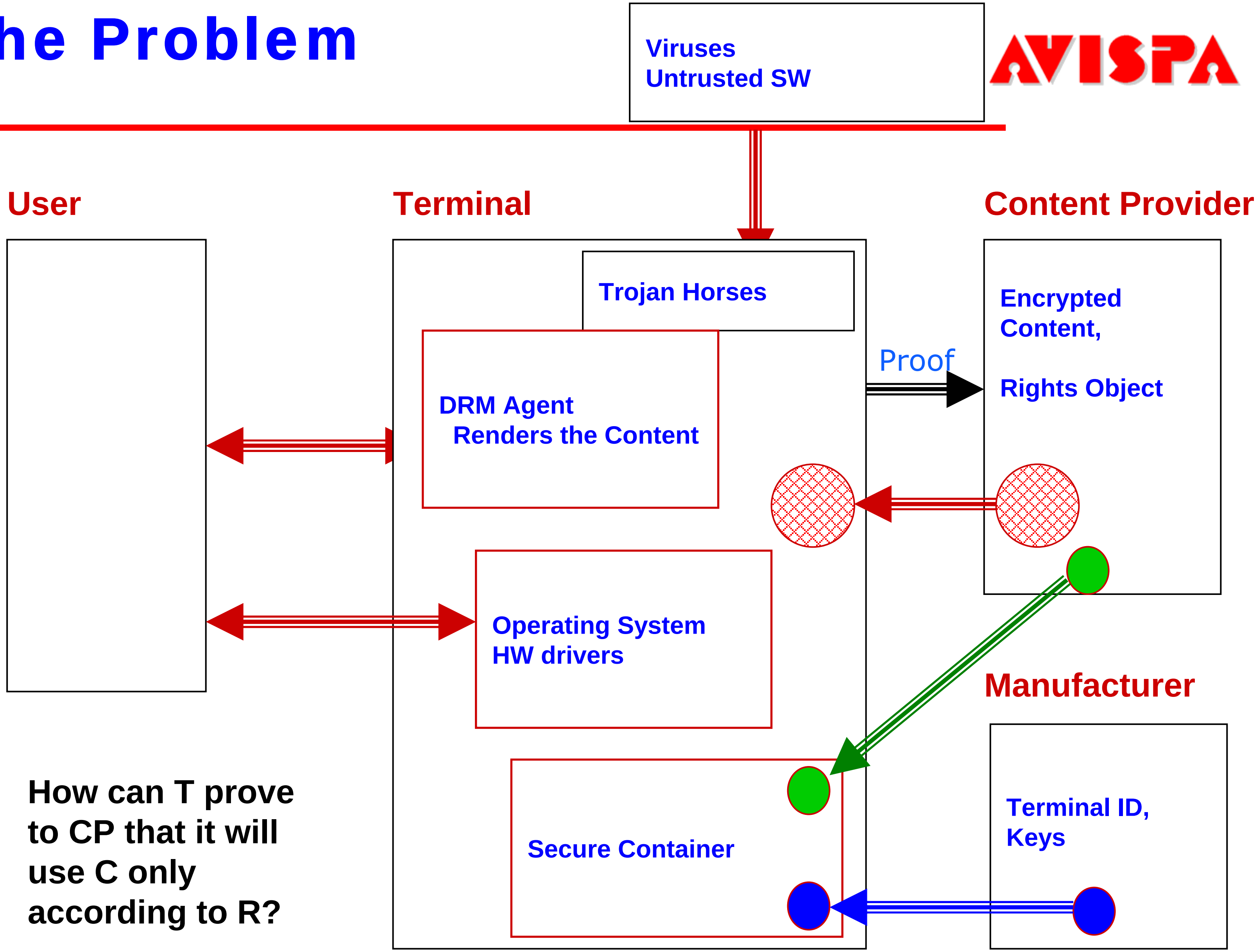


C:
Navigation Maps
Entertainment
Library Docs

OMA DRM: The Concept



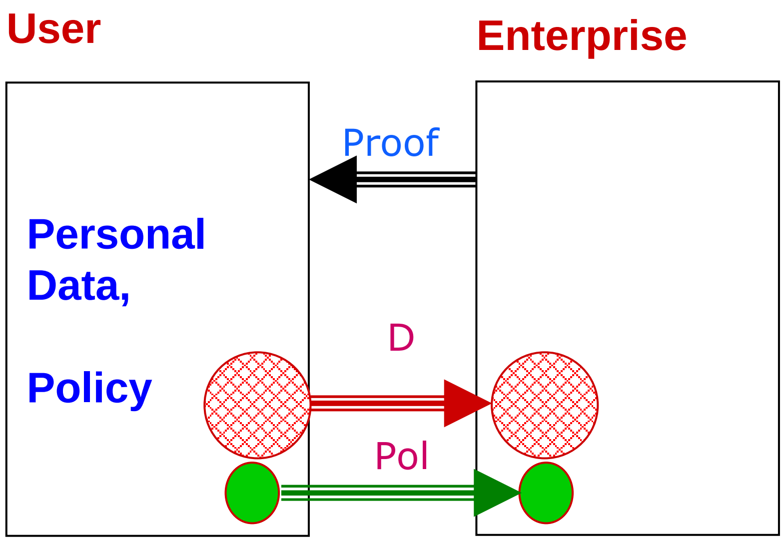
The Problem



How can T prove to CP that it will use C only according to R?

The same Problem in 3 different disguises - 1

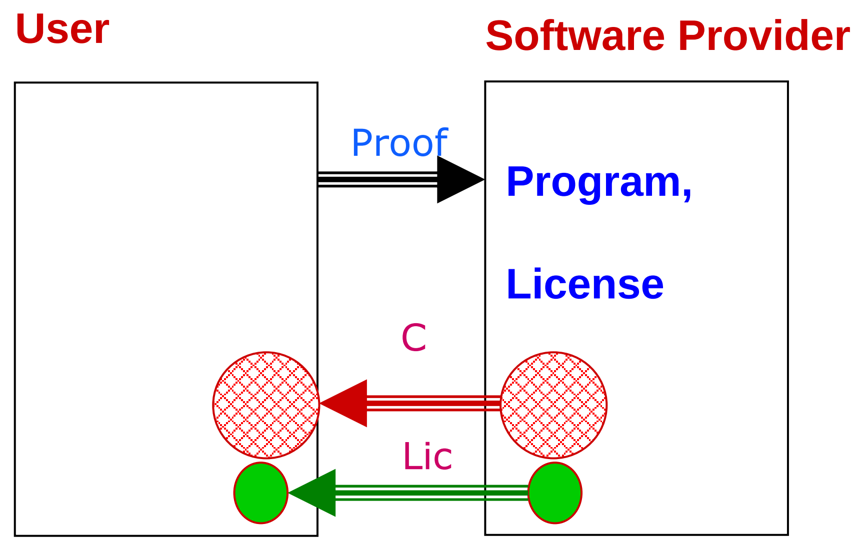
- "Privacy problem"
  - If U is to give some personal data to E, how does E prove to U that it uses the data only according to policies of U?



Document Management in Enterprises
e-Health
e-Government

The same Problem in 3 different disguises - 2

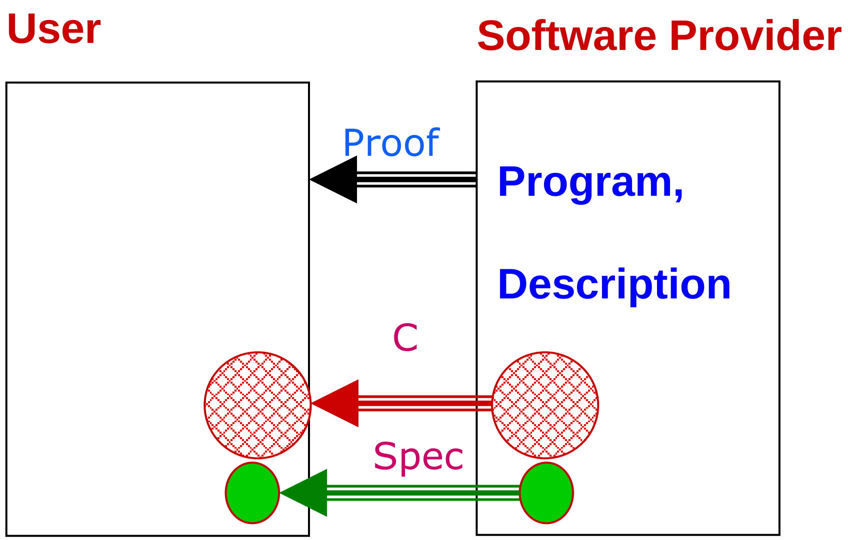
- "Software License problem"
  - If U is to receive some program p from SD, how can U assure to SD that he will use the program only according to the license agreement?



Power generation
Manufacturing
Transportation
Airplane Industry

The same Problem in 3 different disguises - 3

- "Trusted Software download problem"
  - If U is to receive some program p from SD, that is supposed to perform a certain functionality, how can SD assure to U that this program will only behave as stated in the spec (and for instance contains no virus or Trojan application)?



Radio terminal reconfiguration,
Java
…

Conclusions

Internet Layers, Basics

Management, Implementation or Design Errors

IETF Groups and Activities

Security Protocols: Kerberos, AAA, IPsec, IKE, WLAN

Public-Key Infrastructure (PKI)

High-level Protocol Specification Language (HLPSL)

Outlook: MobileIP, DRM

Loose Ends

- Internet offers agent many identities
  - user, IP addr, MAC, TCP port, ... What is “A”, “ID_A”?
- Many types of attackers (or channels)
  - over the air, authentic channels, connection channels
  - “safer” routes
- Many types of properties
  - besides authentication and secrecy “Incomplete protocols”
  - key control, perfect forward secrecy, ...
  - Layered-strength properties (graceful degradation)
    - if attacker ... then ..., if attacker ... then ...
- Many types of DoS attacks
  - flooding, bombing, starving, disrupting, …
- Modular security mechanisms
- New types of Agents (without keys!)

Formal verification is starting to make a difference