Internet Security Protocols: Specification and Modeling

Service Oriented Architectures Security

Ernesto Damiani

Università degli Studi di Milano

Contents

Internet Layers, Basics

Management, Implementation or Design Errors

Designing Correct Protocols: The Avispa contribution

IETF Groups and Activities

Sec Protocols: Kerberos, AAA,

IPsec, IKE, IKEv2, Wlan,

PKI, TLS

High-level Protocol Spec. Language (hlpsl): Syntax,

Semantics, Goals, Examples

Outlook: MobileIP, HIP, Pana

Brisbane, Sep 2003

Conclusions

Internet offers agent many identities

- user, ip, 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 DoS attacks

- flodding, bombing, starving, disrupting

Many types of properties

- besides authentication and secrecy
- “Incomplete protocols” (need to add extra messages to prove authentication goals)
- key control, perferct forward secrecy, ...
- layered properties
  - –if attacker ... then ..., if attacker ... then ...

Internet

Protocols define

Format and order of msgs sent and received among network entities, and

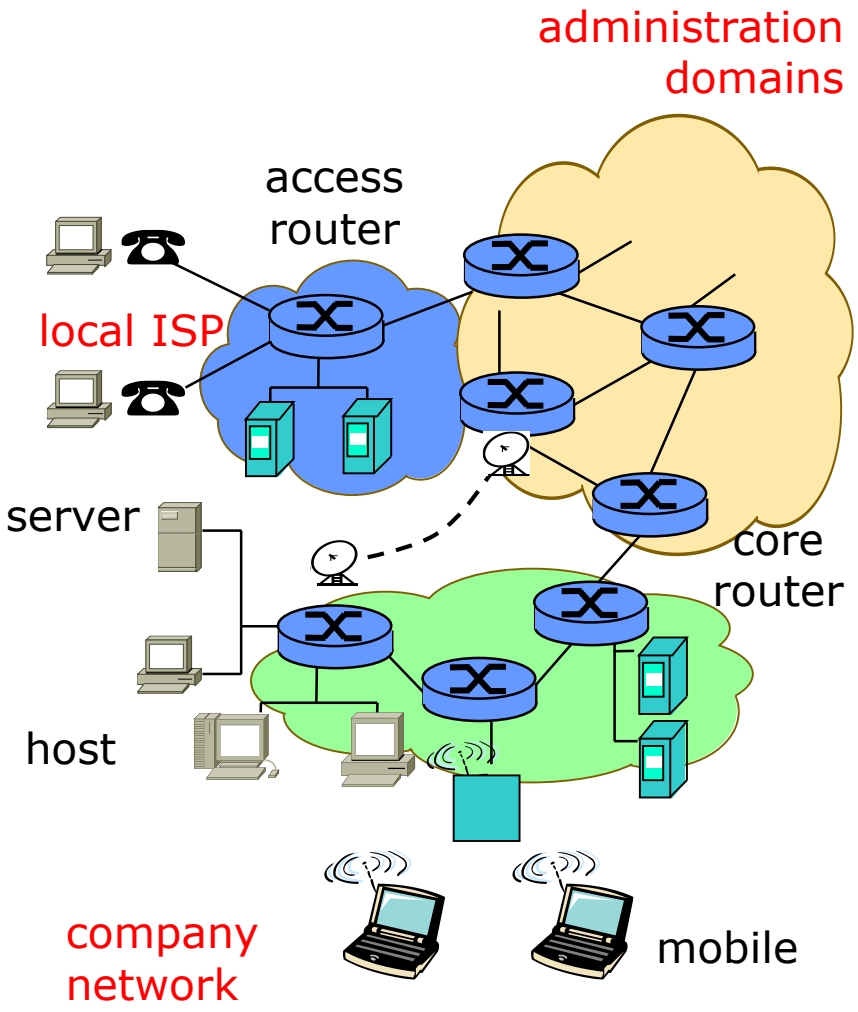
actions taken on msg transmission, receipt

- Examples: TCP, IP, HTTP, FTP,PPP

Internet: "network of networks"

Standards

- RFC: Request for Comments
- IETF: Internet Engineering Task Force

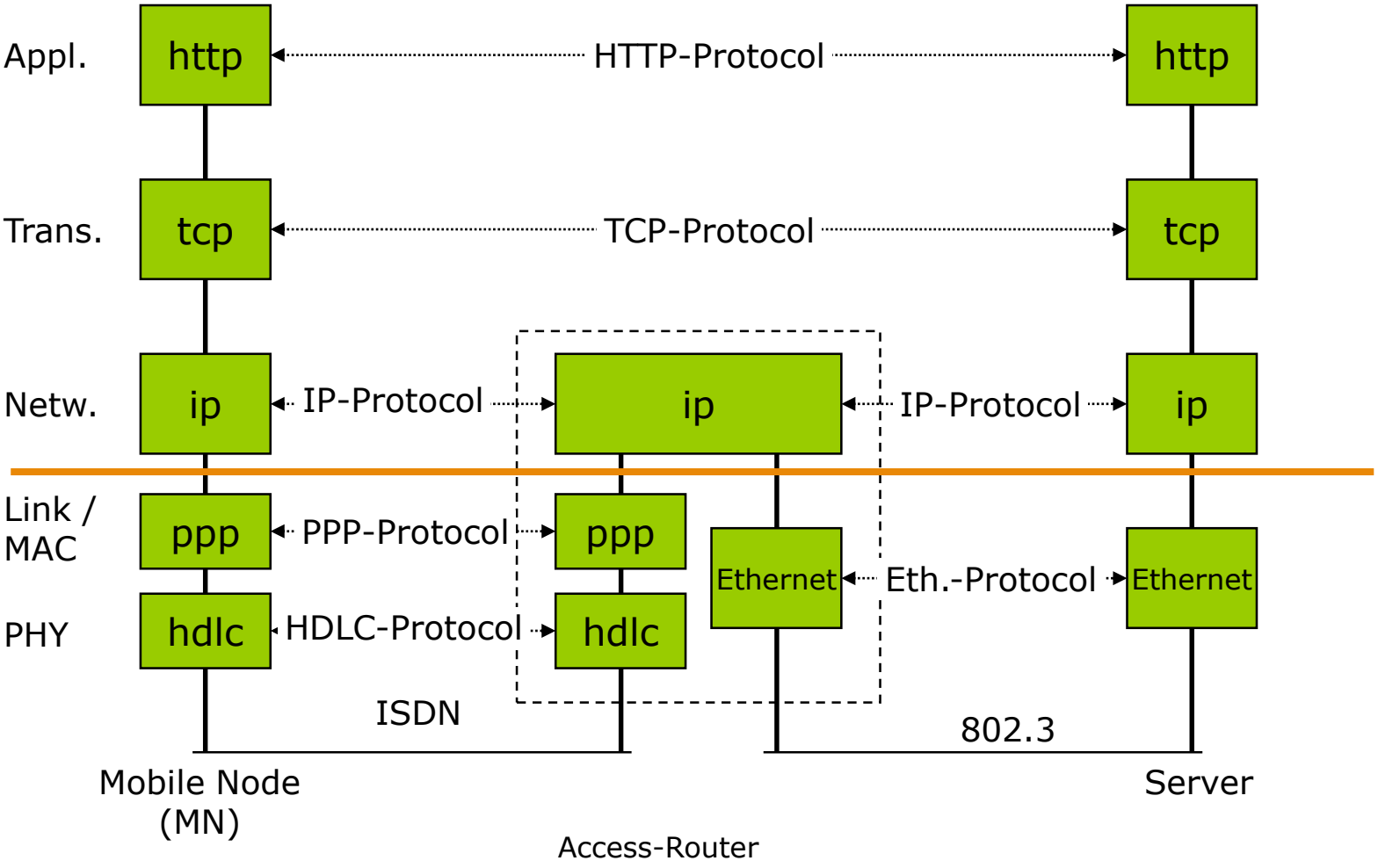


Protocol layering in Internet

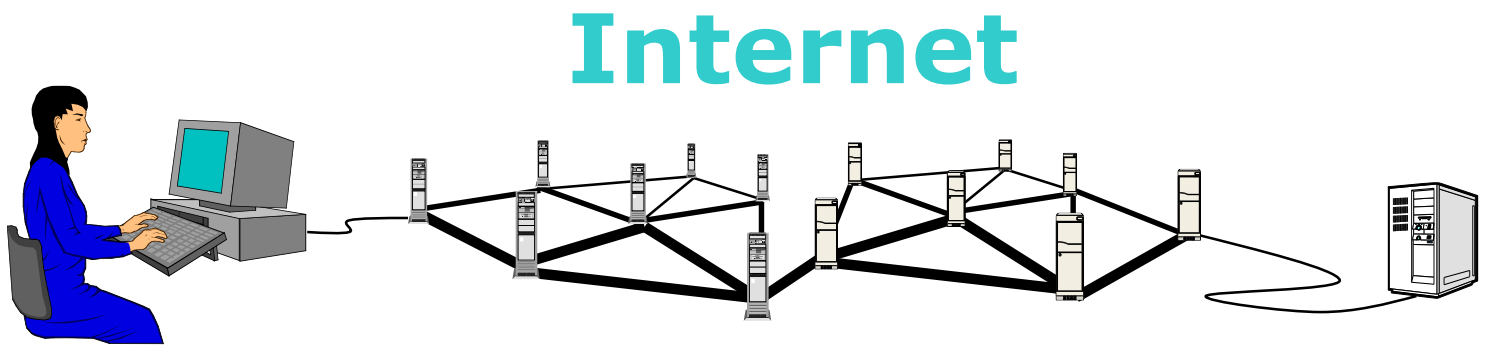
„Independtent" Layers

Headers

Tunneling

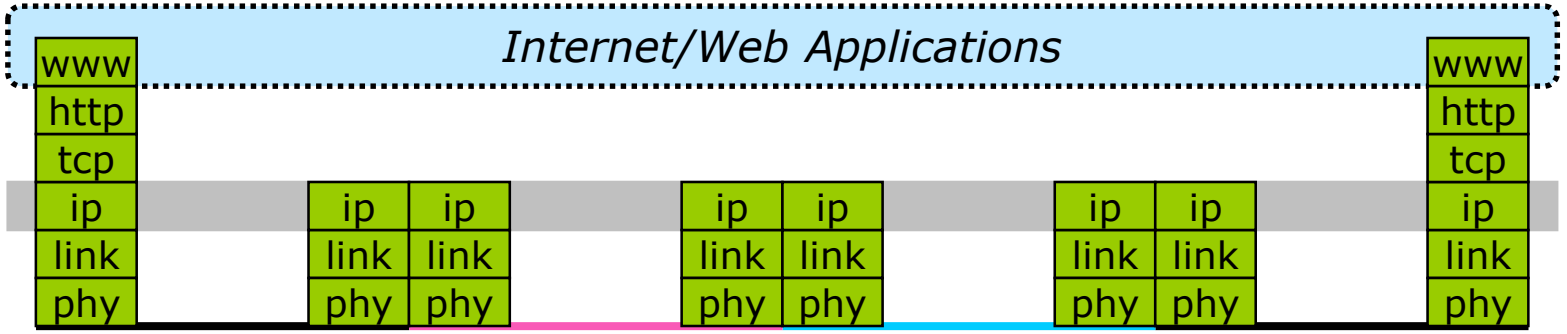


Internet Network Architecture

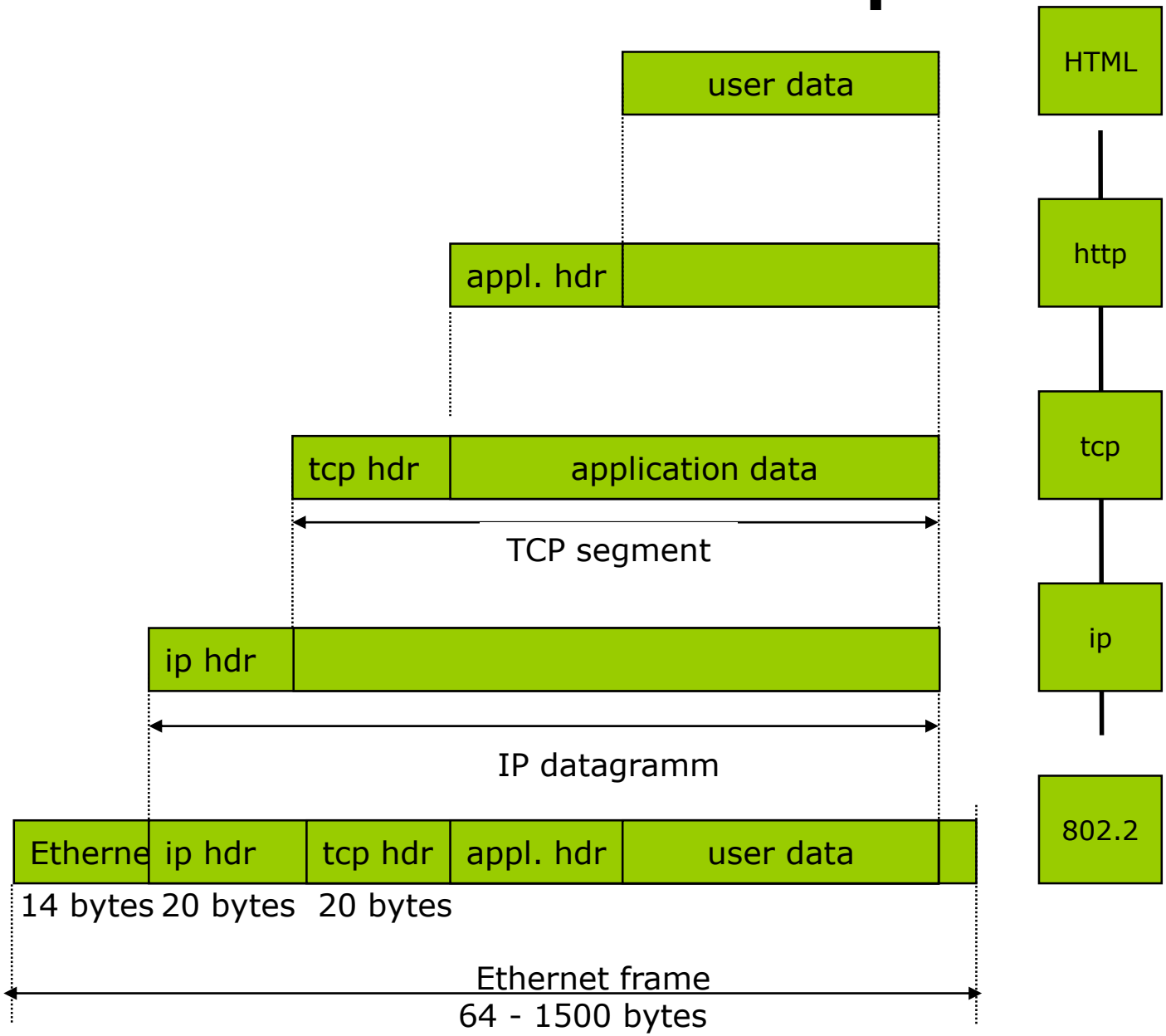


Peer
(Client)

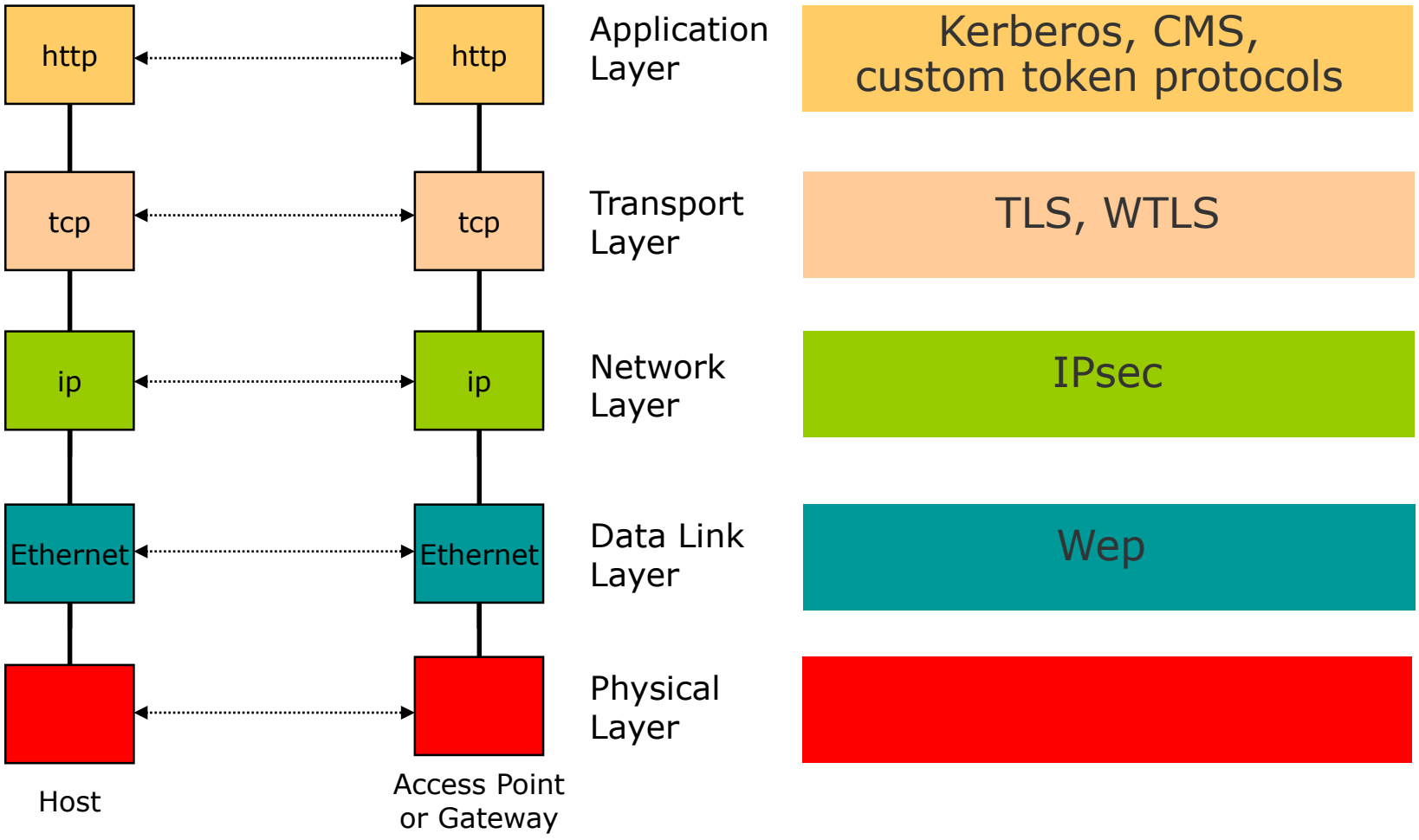
Peer
(Server)



Encapsulation



At which layer security?



Separation of Concerns

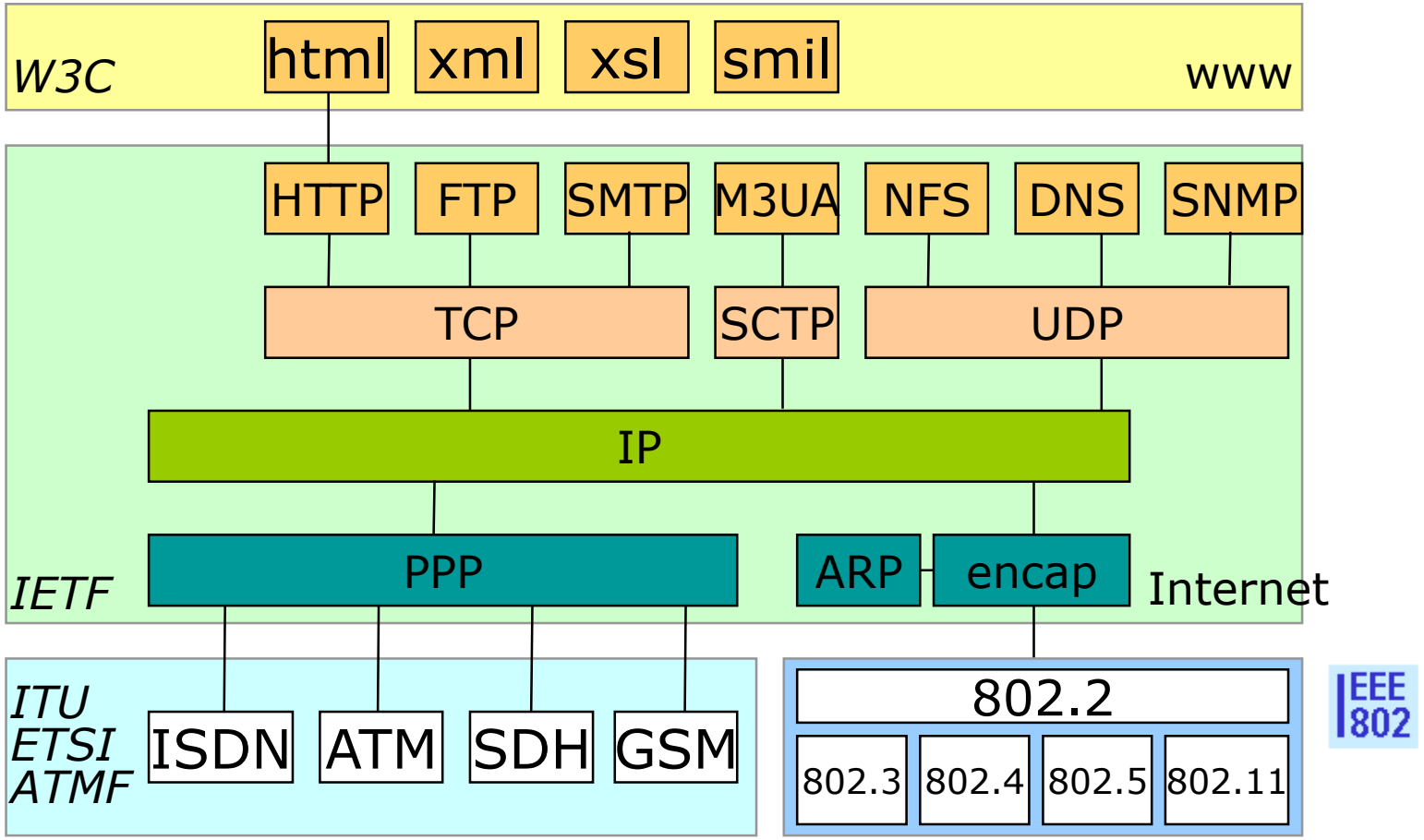
Most security protocols today are separated into two parts:

1) Authentication and key exchange protocols

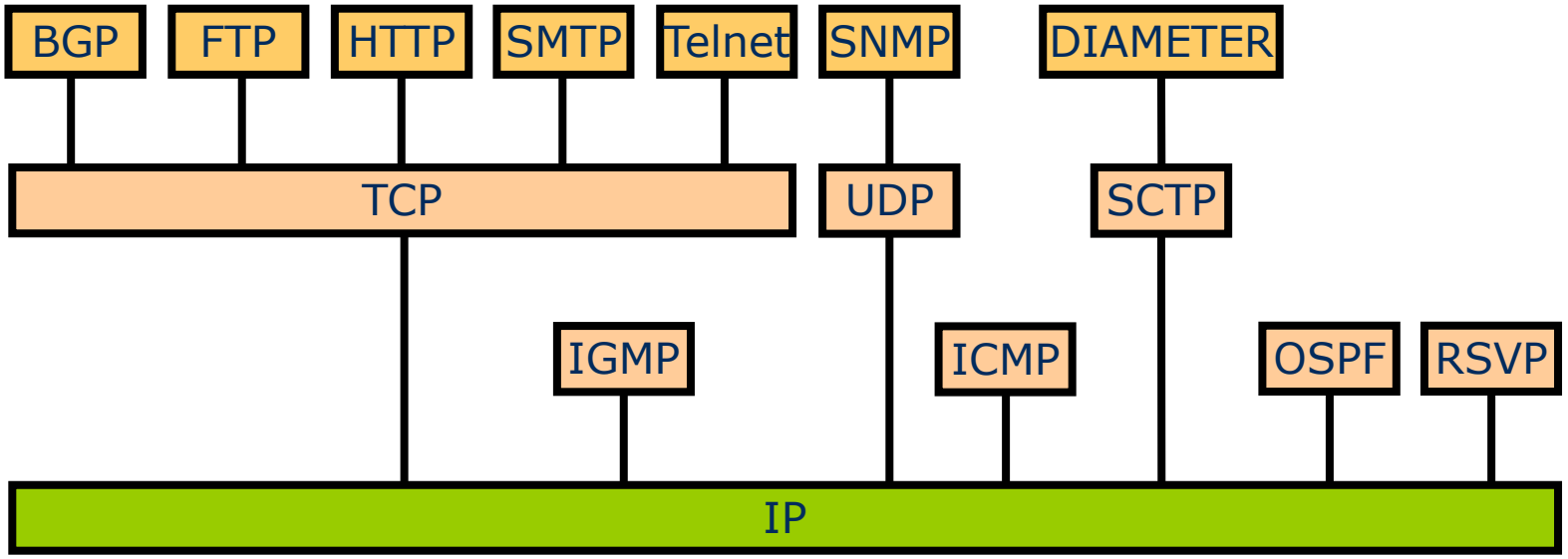
2) Protection of data traffic

Step (1) is usually the most difficult one. Sometimes this step is again separated into sub-steps for performance reasons.

Internet protocol architecture



Some protocols in the TCP/IP Suite



BGP = Border Gateway Protocol

DIAMETER = (2 x RADIUS) = New AAA Protoc

FTP = File Transfer Protocol

HTTP = Hypertext Transfer Protocol

ICMP = Internet Control Message Protocol

IGMP = Internet Group Management Protocol

IP = Internet Protocol

MIME = Multi-Purpose Internet Mail Extension

OSPF = Open Shortest Path First

RSVP = Resource ReSerVation Protocol

SMTP = Simple Mail Transfer Protocol

SNMP = Simple Network Management Protocol

TCP = Transmission Control Protocol

TCP = Transmission Control Protocol

UDP = User Datagram Protocol

Securing the Infrastructure

- Applications need complex, reliable protocols for service discovery, session control, guaranteeing QoS, etc.
- Network control mechanisms and routing protocols have minimal or no authentication at all
- Infrastructure mechanisms often may not use IPSec or TLS to secure their operations

AAA Definitions

Authentication

Verifying an identity (distinguishing identifier) claimed by or for a system entity. This is done presenting authentication information (credentials) that corroborates the binding between the entity and the identifier. (2828)

Entity authentication

Assuring one party (through acquisition evidence) of the identity of a second party involved in a protocol, and that the second has actually participated (i.e., is active at, or immediately prior to, the time the evidence is acquired).

AAA Definitions

Message authentication

A party is corroborated as the source of specified data created at some (typically unspecified) time in the past, and data integrity, but no uniqueness or timeliness guarantees.

Methods for providing data origin authentication include:

- 1. message authentication codes (MACs)
- 2. digital signature schemes
- 3. appending (prior to encryption) a secret authenticator value to encrypted text.

A difference btw. entity and msg authentication:

- message authentication provides no timeliness guarantee
- entity authentication implies actual communications with verifier during execution of the protocol

AAA Definitions

Authorization

An "authorization" is a right or a permission granted to an entity to access a system resource. An "authorization process" is a procedure for granting such rights. (2828) Here: Policy-based. Others: ACL, capability tokens.

Accounting

The collection of resource consumption data for the purposes of capacity and trend analysis, cost allocation, auditing, and billing. Accounting management requires that resource consumption be measured, rated, assigned, and communicated between appropriate parties.

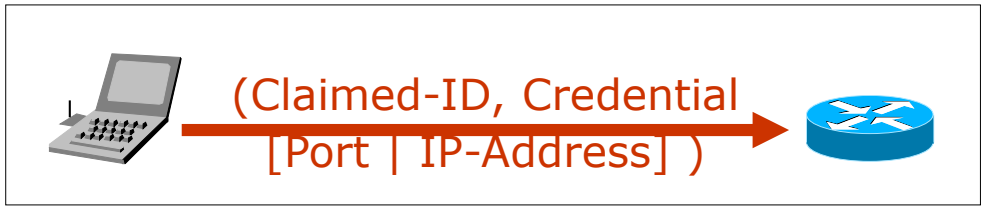
AAAA Definitions

Accountability

The property of a system (including all of its system resources) that ensures that the actions of a system entity may be traced uniquely to that entity, which can be held responsible for its actions. (2828)

Authentication

This makes no sense.
Credentials belong to claimed-ID, so what?
(Probably I knew that before)

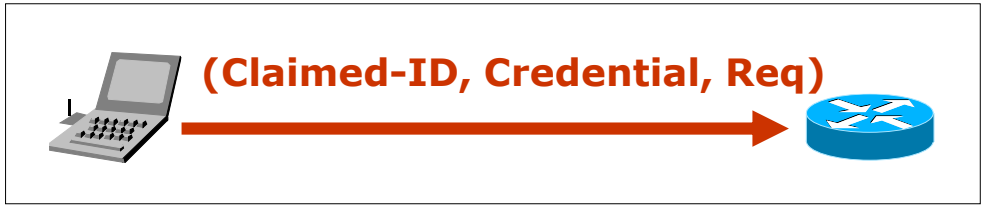


This, alone, makes no sense.
Claimed-ID is now at port xyz,so what?
In the next message?

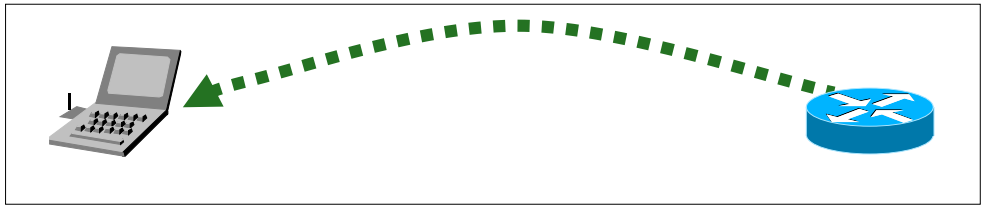
This makes sense.
Claimed-ID is requesting this or telling that.

In connectionless communication, entity authentication without a meaningful message other than the claim of being a particular entity **makes no sense**.

Security Relations



How can the router verify the Credentials and check that the Req is forn Claimed-ID?



The router has to know something special about the Claimed-ID: he has to have a Security Relation (pre-established) or obtain one.

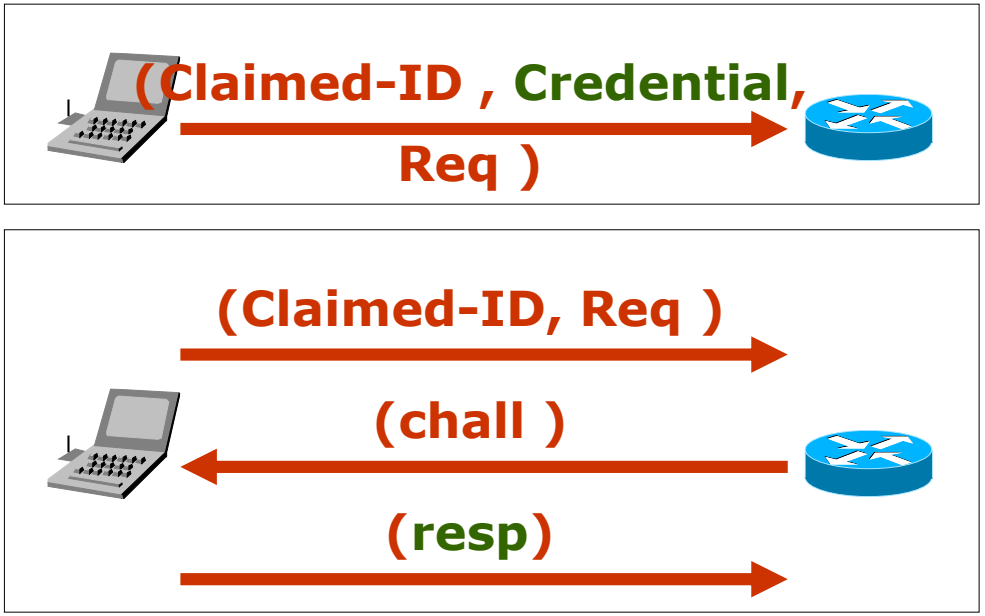
Examples:

- Knowledge of the validity of a Public Key (Digital certificates, PKI)
- Shared secret (password, key) Note: in this case the SR is bidirectional

Authentication Credentials

Examples:

- Digital certificates (PKI)
- f(secret key, time-stamp)
- rsp := f(secret key, chall), i.e. responses to Challenges



Key Establishment

Protocol whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use.

Subdivided into

- key transport and
- key agreement

Key transport: one party creates or otherwise obtains a secret value, and securely transfers it to the other(s).

Key agreement: a shared secret is derived by two (or more) parties as a function of information contributed by, or associated with, each of these

Key Establishment

Authentication term	Central focus
authentication	depends on context of usage
entity authentication	identity of a party, and aliveness at a given instant
data origin (=msg) authentication	identity of the source of data (+integrity)
(implicit) key authentication	identity of party which may possibly share a key
key confirmation	evidence that a key is possessed by some party
explicit key authentication	evidence an identified party possesses a given key

Key Agreement -- Properties

(Implicit) Key authentication:

- one party is assured that no other party aside from a specifically identified second party (and possibly additional identified trusted parties) may gain access to a particular secret key.
- independent of the actual possession of such key by the second party.

Key confirmation:

- One party is assured that a second (possibly unidentified) party actually has possession of a particular secret key.

Explicit key authentication: both

- (implicit) key authentication and
- key confirmation hold.

Key Agreement -- Properties

Authenticated key establishment

- key establishment protocol which provides key authentication.

Identity-based key establishment

- identity information (e.g., name and address, or an identifying index) of the party involved is used as the party's public key.

Identity-based authentication protocols may be defined similarly.

Session Keys

An ephemeral secret, i.e., restricted to a short time period, after which all trace of it is eliminated.

Reasons:

1. to limit available ciphertext (under a fixed key) for cryptanalytic attack;
2. to limit exposure, with respect to both time period and quantity of data, in the event of (session) key compromise;
3. to avoid long-term storage of a large number of distinct secret keys (in the case where one terminal communicates with a large number of others), by creating keys only when actually required;
4. to create independence across communications sessions or applications

Key Agreement -Classification

1. Nature of the authentication:

a. entity authentication,

b. key authentication, and

c. key confirmation.

2. Reciprocity of authentication. If provided, entity authentication, key authentication, and key confirmation may be unilateral or mutual

3. Key freshness. A key is fresh (from the viewpoint of one party) if it can be guaranteed to be new, as opposed to possibly an old key being reused through actions of either an intruder or authorized party. This is related to key control

Key Agreement - Classification

4. Key control:

the key is derived from joint information, and neither party is able to control or predict the value of the key

5. Efficiency.

(a) number of message exchanges
(b) bandwidth (total number of bits)

(c) complexity of computations

(d) precomputation to reduce on-line computational complexity

Key Agreement - Classification

6. Third party requirements

(a) on-line (real-time),

(b) off-line, or

(c) no third party;

(d) degree of trust required in third party (e.g., trusted to certify public keys vs. trusted not to disclose long-term secret keys).

7. Type of certificate used and manner by which initial keying material is distributed

8. Non-repudiation

some type of receipt that keying material has been exchanged.

Perfect forward secrecy and known-key attacks

Perfect forward secrecy

- compromise of long-term keys does not compromise past session keys.
  - –Previous traffic is locked securely in the past.
  - –It may be provided by a Diffie-Hellman procedure.
- If long-term secret keys are compromised, future sessions **are** subject to impersonation by an active intruder

Immunity to known-key attack: When past session keys are compromised, do not allow

- Passive attacker to compromise future session keys
- impersonation by an active attacker in the future

(Known-key attacks on key establishment protocols are analogous to known-plaintext attacks on encryption)

Many types of keys

Sealing key: a shared secret key used for computing cryptographic checkvalues (MACs)

Signature key: a private key used for signing,

Verification key: a public key used for checking signatures, or a secret key used for checking MACs

Encipherment key: either secret or public key,

Decipherment key: either secret or private key.

Keys shold be used only for one purpose

Contents

Internet Layers, Basics

Management, Implementation or Design Errors

Designing Correct Protocols: The Avispa contribution

IETF Groups and Activities

Sec Protocols: Kerberos, AAA,

IPsec, IKE, IKEv2, Wlan,

PKI, TLS

High-level Protocol Spec. Language (hlpsl): Syntax,

Semantics, Goals, Examples

Outlook: MobileIP, HIP, Pana

Management Problems: Passwords, Cards, Tokens

Passwords are

- often shared
- guessable
- written down on pieces of paper

Smart cards and hand-held tokens are

- expensive
- People forget them
- Card readers draw too much power from hand-helds

Management Problems: WLAN/WEP

WEP is optional,

- many real installations never even turn on encryption
  - –irrelevant how good the cryptography is if it is never used.

By default, WEP uses a single shared key for all users

- often stored in software-accessible storage on each device
- If any device is stolen or compromised, change the shared secret in all of the remaining devices
- WEP does not include a key management protocol

Is PKI secure? More Management Problems

Most users don't know what certificates are.

Most certificates' real-world identities aren't checked by users.

Meaningless Certificates:

- Why should Dow, Jones own the www.wsj.com certificate?
- Is that certificate good for interactive.wsj.com?

Is it NASA.COM or NASA.GOV?

- MICROSOFT.COM or MICR0S0FT.COM?
- What about MICROSOFT.COM? (Cyrillic "O", do you see it?)

Effectively, we have no PKI for the Web.

DoS Attacks against Authentication Protocols

Flooding attacks: Spoofed messages cause target to perform expensive cryptographic operations:

Attacker gets the nodes to perform PK operations. It may spoof a large number of "signed messages" with random numbers instead of signatures

- –Target will verify the signatures before rejecting the messages.
- –Symmetric encryption, hash functions and non-cryptographic computation are rarely the performance bottleneck (unless the cryptographic library is optimized only for bulk data)

If a node creates a state during protocol execution, the attacker may start an excessive number of protocol runs and never finish them

The stronger the authentication, the easier it may be for an attacker to use the protocol features to exhaust target's resources.

SYN Flooding: Implementation Issues

Host accepts TCP open requests, from spoofed locations

Half-open connection queue fills up

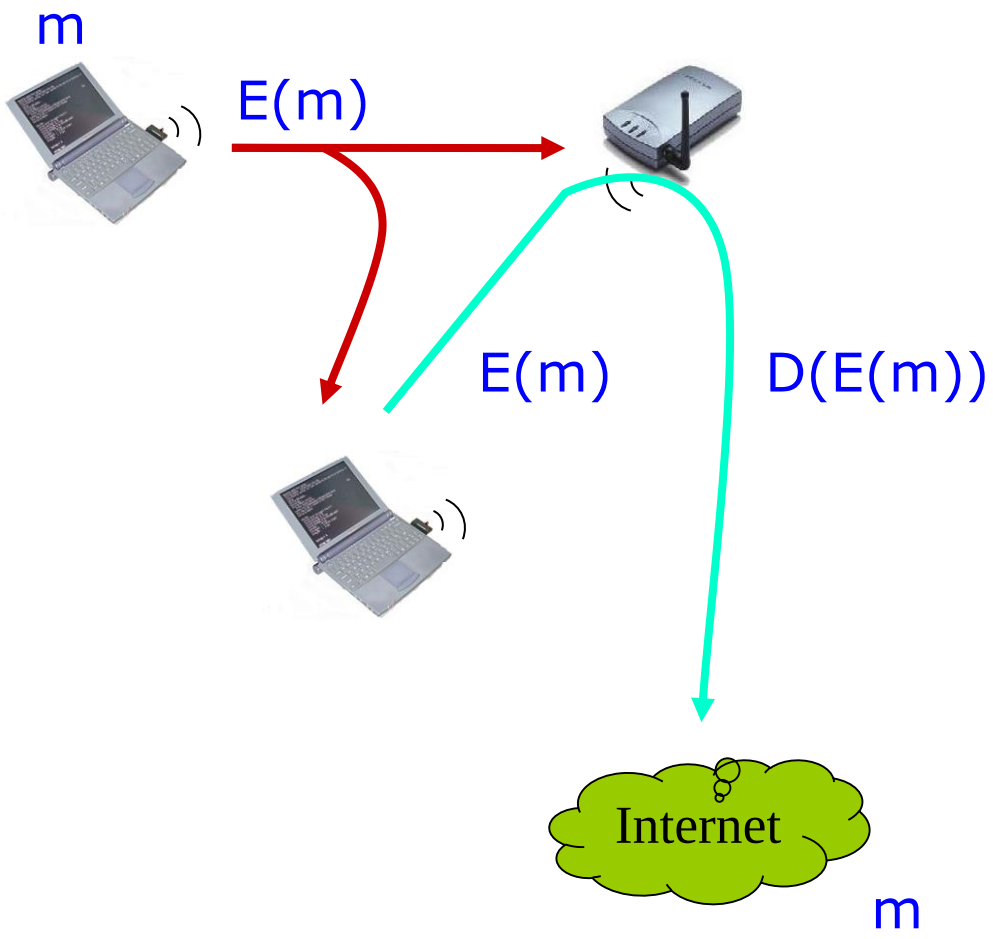
Legitimate open requests are dropped

Implementation issues

Mostly solved:

- use cheaper data structure for queue,
- random drop when queue is full

Design Problems: WLAN/WEP



No perfect Security

Many different types of Attacks

Many different types of Security Mechanisms

- at different SW layers
- with different strength

Management, Implementation or Design Errors

- Design errors affect more people

Some risks

- may be acceptable (low damage or very low risk)
- too expensive to fully prevent

Authentication Levels

None **(no authentication)**

- SASL Anonymous [RFC2245]
- Authentication based on source IP address
- Diffie-Hellman

Weak **(vulnerable against eavesdroppers)**

- FTP USER/PASS
- POP3 USER/PASS

Limited **(vulnerable against active attacks)**

- One-time Passwords
- HTTP Digest Authentication
- IMAP/POP Challenge/Response

Strong **(protection against active attacks)**

- Kerberos
- SRP Telnet Authentication
- Public Key Authentication

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

Attackers

Most are joy hackers.

Soon also Terrorists?

Spies? Governments? Industrial spies?

For profit?

Some businesses report targeted attempts:

- Vendor prices changed on a Web page
- ISP hacked by a competitor
- Customers on pay-per-packet nets targets of packet storms

Well known Attacks: DOS

Denial of Service Attacks

Attacker doesn't break in

- he denies you access to your own resources.

Many incidents reported, more are likely.

You lose:

- if it's cheaper for the attacker to send a message
- than for you to process it

Denial of Service Attacks are hard to prevent

- in particular using security measures at higher layers only

Thumbrules:

- Try to be stateless – allocate resources as late as possible.
- Do expensive computations as late as possible.
- Move heavy computation to the initiator of the protocol run.

DOS Example: “Smurf” Attack

- Attacker sends “ping” to intermediate network’s broadcast address.
- Forged return address is target machine.
- All machines on intermediate network receive the “ping”, and reply, clogging their outgoing net and the target’s incoming net.
- Firewalls at target don’t help -- the line is clogged before it reaches there.

Well known Attacks: Sniffers

Password collection

Credit card numbers

NFS file handle collection

DNS spoofing

Attacks to the infrastructure: Routing Attacks

Routers advertise

- own local nets,
- what they've learned from neighbors

Routers trust dishonest neighbors

Routers further away must believe everything they hear

First solutions in the literature

GSM Today

AV = (chall, resp, C), C = Cipher Key

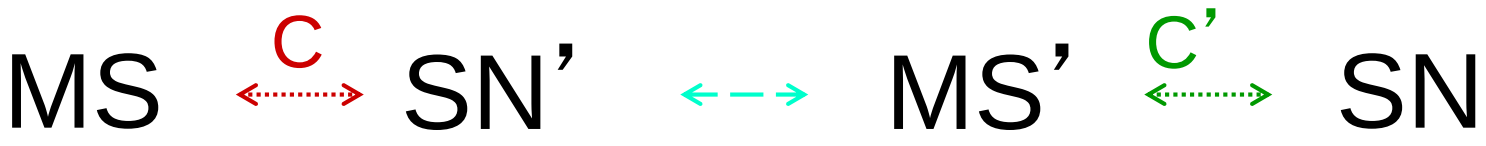
AV generation is not so fast => batch generation

MS is able to calculate: $C = C_k(chall)$

Therefore MS and SN share C.

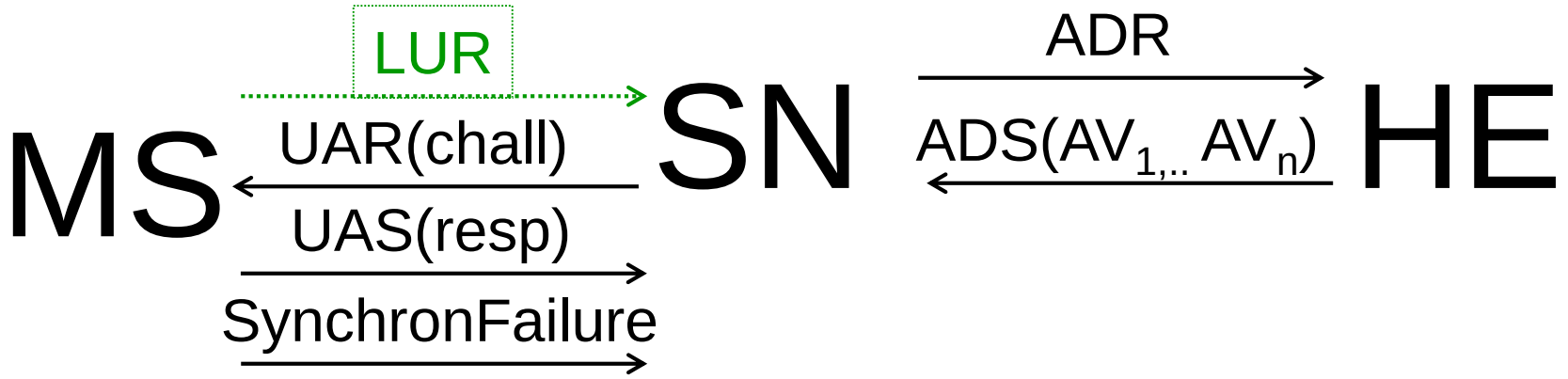
MS authenticates to SN, but SN does not authenticate to MS

GSM Today: Problem



- 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)
  - decipher/encipher
  - 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'

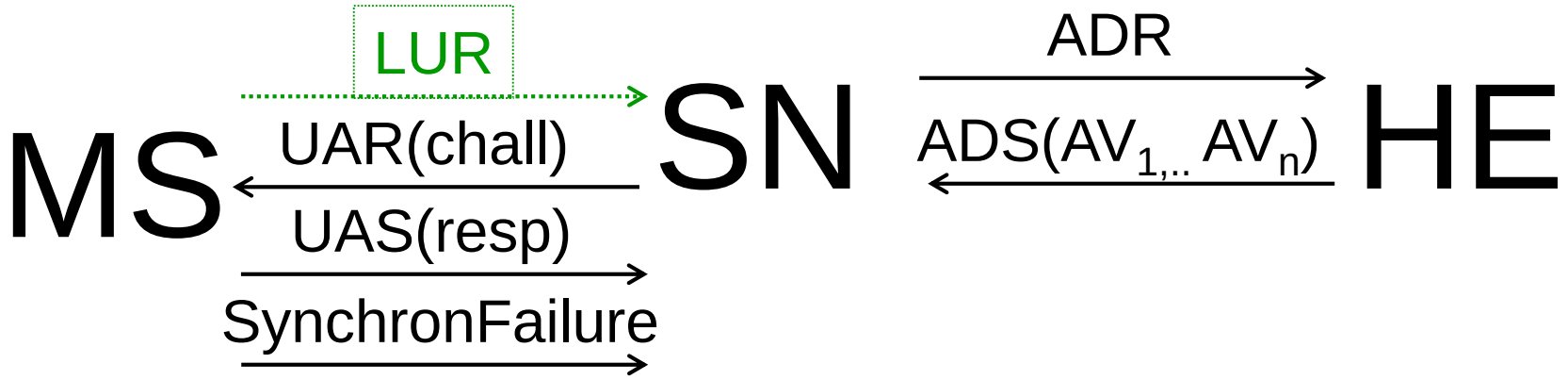
UMTS: Idee



- 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

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

UMTS: Idee



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

Is there no MiM Attack?
Is there no deadlock?
Such design errors would be very expensive:
Replace firmware in many towers
and in millions of Cellular Phones

Contents

Internet Layers, Basics

Management, Implementation or Design Errors

Designing Correct Protoc.: The Avispa contribution

IETF Groups and Activities

Sec Protocols: Kerberos, AAA,

IPsec, IKE, IKEv2, Wlan,

PKI, TLS

High-level Protocol Spec. Language (hlpsl):

Syntax,

Semantics, Goals, Examples

Outlook: MobileIP, HIP, Pana

Avispa

http://www.avispa-project.org

U. of Genova,

LORIA-Lorraine,

ETHZ,

Siemens AG

Shared-cost RTD (FET Open) Project IST-2001-39252

Started on Jan 1, 2003

PBK Construction

Alice sends m1, m2, …, mN,
Bob is able to recognize they have same source

Alice constructs a public/private key pair PBK = (p,s)

Alice disclosed the public key p to Bob along with the initial packet

Bob verifies messages signed with the private key s=inv(p)

Bob knows the messages were sent by *one* node

If replay protection: sequence number or timestamp

Is there a cheaper way?

Generalized PBK: Requirements

Bob receives m1, m2, …, mN,
authentically generated by *one* source

If the first message A → B arrives without modification, all other messages shall be protected in a way that B recognizes alteration

MiM attack in the first message:
A → E → B : B is receiving messages from E

But if first message is OK, the system should protect against MiM

DoS:
If attacker can only insert messages: DoS resilience

1. Hash Construction

If Alice knows in advance which messages she wants to send:
m1, m2, …, mN:

{mi} := <mi , H(mi+1)> (Send mi together with H(mi+1)).
1. Quiz: OK?

No. An attacker can replace $\{m_i\} := \langle m_i , H(m_{i+1})\rangle$ by $\{m_i\} := \langle m_i , H(\mu_{i+1})\rangle$

And then replace $\{m_{i+1}\} := \langle m_{i+1} , H(m_{i+2})\rangle$ by $\{\mu_{i+1}\} := \langle \mu_{i+1} , H(\mu_{i+2})\rangle$ etc.

$\{m_i\} := \langle m_i , H(m_{i+1}, m_{i+2}, \ldots, m_N)\rangle$ 2. Quiz: OK?

I think, yes, this seems easy to prove.

2. Hash Construction

Alice chooses a hash sequence:

$h1 = H(h2) = H(H(h3)) = H^{i}(hi+1) = .. = H^{N-1}(hN)$:

$\{mi\} := \langle mi , H(mi , hi) \rangle$

What is wrong? (Too trivial for a quiz!)

Bob has no means to check HAshes.

$\{m_i\} := \langle m_i , H(m_i , h_i), h_i \rangle$ 3. Quiz: OK?

No. Attacker replaces $\{m_i\} := \langle m_i , H(m_i , h_i), h_i \rangle$ by $\langle \mu_i , H(\mu_i , h_i), h_i \rangle$

3. Hash Construction

Hash sequence: $h1 = H(h2) = H(H(h3)) = \ldots = H^{i}(hi+1) = .. = H^{N-1}(hN)$
$\{mi\} := \langle mi , H(mi , hi), hi-1 \rangle$

4. Quiz: OK?

No. Attacker intercepts 2 consecutive messages
$\{m_i\} := \langle m_i , H(m_i , h_i), h_{i-1} \rangle$ $\{m_{i+1}\} := \langle m_{i+1} , H(m_{i+1} , h_i), h_i \rangle$
replaces
$\{m_i\}$ by $\langle \mu_i , H(\mu_i , h_i), h_{i-1} \rangle$

Idea: Alice waits for an Acknowledge $\{ack_i\} := \langle H(m_i , \hat{h}_i), \hat{h}_i \rangle$
(Bob uses seq: $\hat{h}_1 = H(\hat{h}_2) = H(H(\hat{h}_3)) = \ldots = H^{i}(\hat{h}_{i+1}) = .. = H^{N-1}(\hat{h}_N)$)

5. Quiz: OK?

I think, yes. Is somebody sure? What is not nice about the solution?

That B is forced to use a hash series, one for each peer. (DoS)

4. Hash Construction

Hash sequence: h1= H(h2)= H(H(h3)) =… = Hi(hi+1) =.. = HN-1(hN)

{mi} := <mi , H(mi , hi), hi-1 >

Alice waits for an Acknowledge {acki} := <H(mi , ĥi), ĥi , H(ĥi+1)>

6. Quiz: OK?

I think, yes. Is somebody sure?

Another idea: instead of acknowledgments, use time frames. This will work for multimedia. Both A and B divide their time in intervals: A sends at the beginning of his intervals, B discards messages that arrive too late.

7. Quiz: Dos that work?

I think, yes. Is somebody sure?

Motivation for the project

There are many techniques for the automatic analysis of security protocols, BUT

- tools usually come with specific working assumptions (specification language, security Goals, modelling assumptions, bounds, . . .)

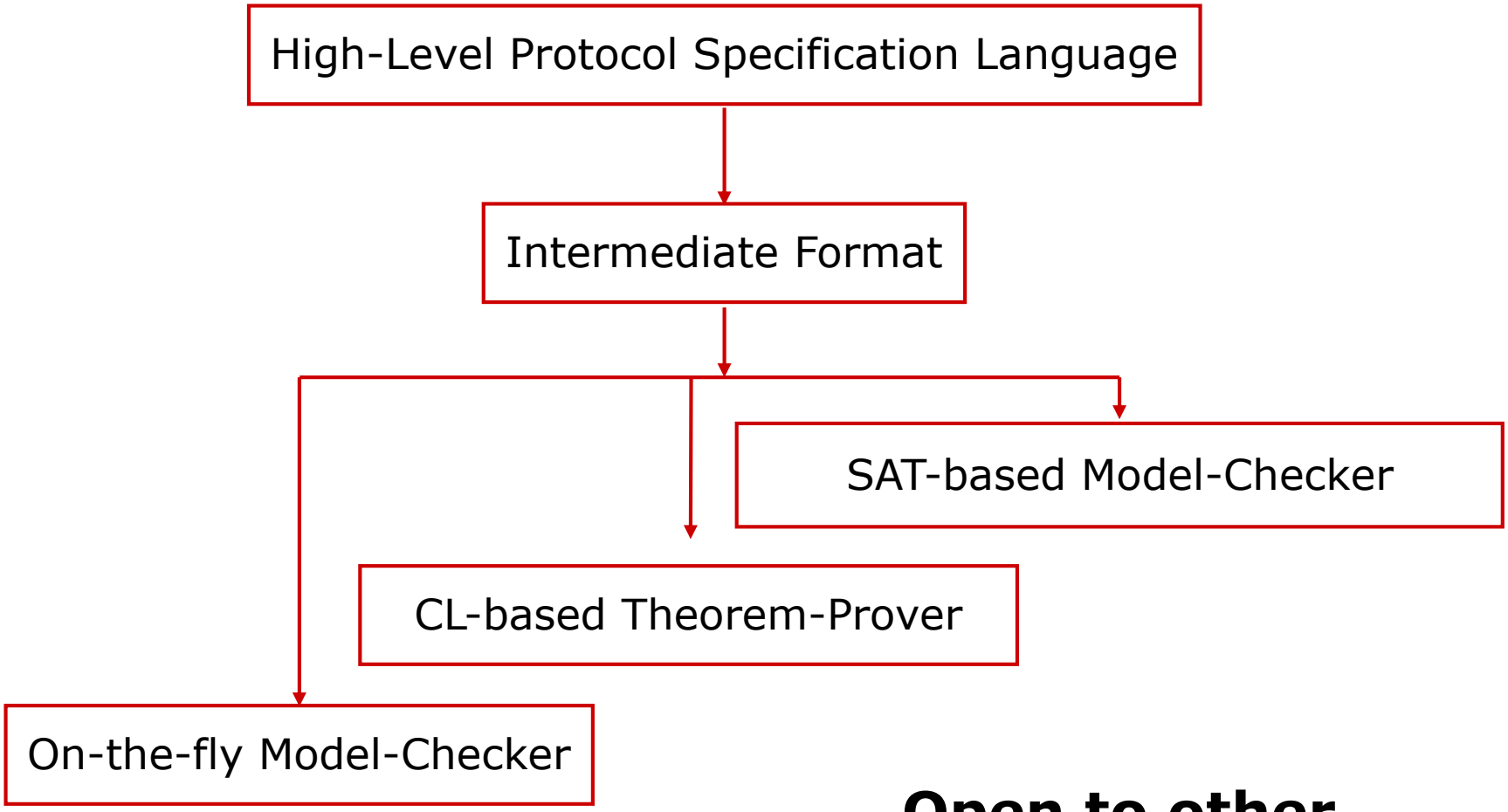
This makes it very difficult

- to use the tools productively (for the non-expert user) and
- to assess and compare the potential of the proposed techniques.

Objectives of the AVISPA Project

1. **Build a open architecture supporting**
   a) design of security protocols using a comfortable notation and web-based user-friendly interface
   b) seamless integration and systematic assessment of new automated techniques for the validation of security protocols.
2. **Build and make publicly available a library of formalized IETF protocols and associated security problems.**
3. **Develop and tune three promising and complementary state-of-the-art technologies for automatic formal analysis:**
   a) On-the-fly Model-Checking
   b) Constraint Theorem-Proving
   c) SAT-based Model-Checking

Architecture of the AVISPA Tool



Open to other technologies

On-the-fly Model-Checking

Context: On-the-fly model checking supports the incremental exploration of very large or infinite state systems. Lazy evaluation in languages like Haskell provides a powerful platform for building flexible, efficient search engines.

Approach: Lazy evaluation is combined with symbolic (unification-based) methods to build on demand, and explore, the protocol search space.

Advantages:

- Declarative specification of infinite data structures, reduction methods, and heuristics.
- Modular design, easy integration of heuristics/improvements.

Constraint Theorem-Proving

Context: Rewrite-based, first-order theorem provers have recently appeared as very effective tools for equational reasoning. daTac combines rewriting with constraints to handle properties such as associativity/commutativity.

Approach: Messages exchanges and intruder activities can be directly translated into rewrite rules. Searching for an attack amounts to deducing a contradiction.

Advantages:

- Protocol representation is simple and intuitive.
- Advancements in deduction can be easily incorporated.
- Fast prototyping of model enhancements (e.g. algebraic properties of operators).

SAT-based Model-Checking

Context: Dramatic speed-up of SAT solvers in the last decade:

- Problems with thousands of variables are now solved routinely in milliseconds.

Approach: Bounded model-checking of security protocols based on a constructive translation of the IF into SAT with iterative deepening on the number of steps.

Advantages:

- Most of the generated SAT instances are solved in milliseconds.
- Declarative.
- Plug and play integration of different SAT solvers $\Rightarrow$
- Improvements of SAT technology can be readily exploited.

Contents

Internet Layers, Basics

Management, Implementation or Design Errors

Designing Correct Protocols: The Avispa contribution

IETF Groups and Activities

Sec Protocols: Kerberos, AAA,

IPsec, IKE, IKEv2, Wlan,

PKI, TLS

High-level Protocol Spec. Language (hlpsl): Syntax,

Semantics, Goals, Examples

Outlook: MobileIP, HIP, Pana

Internet History

1961-1972: Early packet-switching principles

1961: Kleinrock - queueing theory shows effectiveness of packet-switching

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 History

1972-80: Internetworking, new and proprietary nets

1970: ALOHAnet satellite network in Hawaii

1973: Metcalfe's PhD thesis proposes Ethernet

1974: Cerf and Kahn - architecture for interconnecting networks

late70's: proprietary architectures: DECnet, SNA, XNA

late 70's: switching fixed length packets (pre ATM)

1979: ARPAnet 200 nodes

Cerf and Kahn's internetworking principles:

- minimalism, autonomy - no internal changes required to interconnect networks
- best effort service model
- stateless routers
- decentralized control

define today's Internet architecture

Internet History

1980-1990: new protocols, a proliferation of networks

1983: deployment of TCP/IP

1982: SMTP e-mail

1983: DNS name-to-IP-address translation

1985: FTP

1986, Jan: first IETF meeting 21 attendees

1986, Oct: 4th IETF, first IETF with non-government vendors

1987, Feb: 5th IETF: Working Groups were introduced

1987, Jul: 7th IETF, > 100 attendees

1988: TCP congestion control

New national networks: Csnet, BITnet, NSFnet, Minitel

100,000 hosts connected to confederation of networks

1993 July: IETF met in Amsterdam, first IETF meeting in Europe

US/non-US attendee split was (+is) nearly 50/50.

Internet Organizations

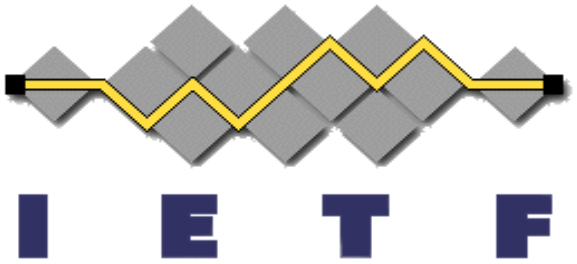
ISOC (Internet Society)
political, social, technical aspects of the Internet
http://www.isoc.org/

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 reporting, and
  - open IESG meeting.

Proceedings of each IETF plenary

Meeting minutes,

working group charters (which include the working group mailing lists),

are available on-line see www.ietf.org.

IETF Current Areas

Applications (APP) - Protocols seen by user programs, such as e-mail and the Web

Internet (INT) - Different ways of moving IP packets and DNS information

Operations and Management (OPS) Administration and monitoring

Routing (RTG) - Getting packets to their destinations

Security (SEC) - Authentication and privacy

Transport (TSV) - Special services for special packets

User Services (USV) - Support for end users and user support organizations

General (GEN) - Catch-all for WGs that don't fit in other areas (which is very few)

IETF procedures

The IETF is a group of individual volunteers (~ 4 000 woldwide)

Work is being done on mailing lists (plus 3 meetings/year)

No formal membership, no formal delegates

Participation is free and open

>110 working groups with well defined tasks and milestones

Major US vendors dominate the IETF

IETF does not decide about the market, but: the 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
- Mailing list and face-to-face meetings

Consensus made via e-mail

- (no "final" decisions made at meetings)

Contents

Internet Layers, Basics

Management, Implementation or Design Errors

Designing Correct Protocols: The Avispa contribution

IETF Groups and Activities

Sec Protocols: *Kerberos*, AAA,

IPsec, *IKE*, IKEv2, *Wlan*,

PKI, TLS

High-level Protocol Spec. Language (hlpsl): Syntax,

Semantics, Goals, Examples

Outlook: *MobileIP*, HIP, Pana

Kerberos

An authentication system for distributed systems

Introduction

Based on Needham - Schroeder

Three-Party Protocol

Extensions according to Denning - Sacco.

Developed at MIT as part of the project Athena

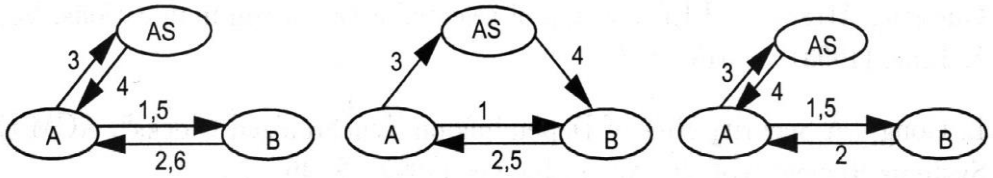
Versions 1 - 3 internal

Currently the following Kerberos Version are published:

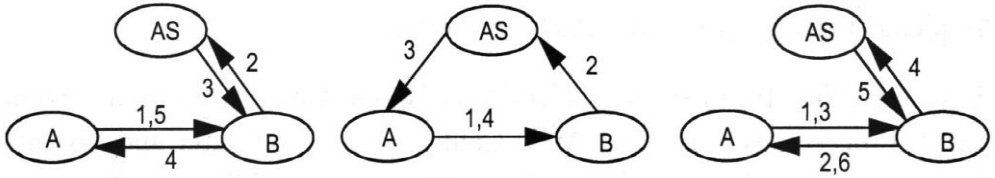
- Kerberos v4
- Kerberos v5

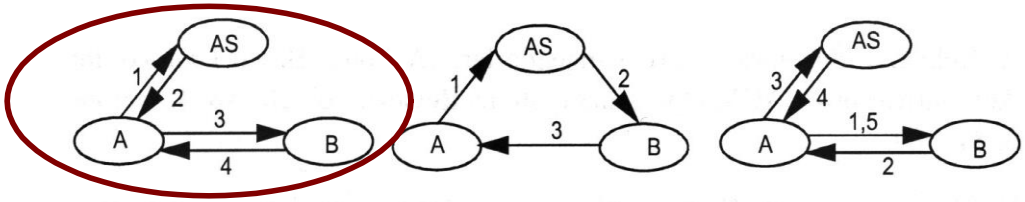
Kerberos v5 Clarifications/Revisions (not finished)

Three Party Protocols

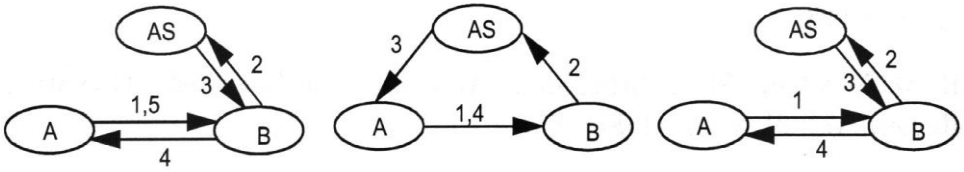


Nonce-based Protocol



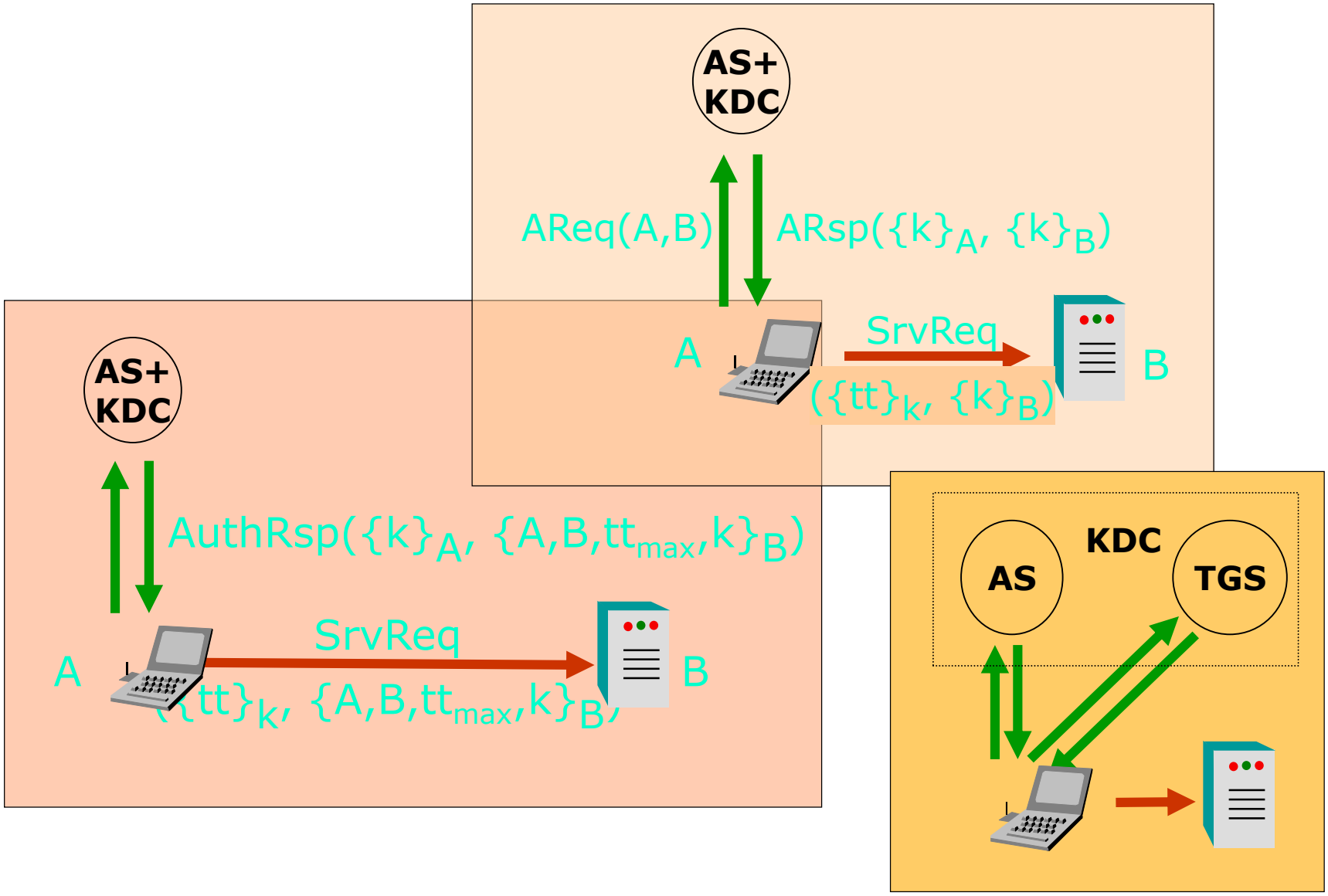


Kerberos Timestamp-based Protocol



Y. Ding and H. Petersen: "Eine Klassifikation von Authentifikationsmodellen", Proc. Trust Center'95, DuD Fachbeiträge, 292 - 302, 1995.

Kerberos in three Acts



Kerberos Single-Sign-On

Obtaining additional tickets

- Don't want to present user's password each time the user performs authentication with a new service
- Caching the user's password on the workstation is dangerous
- Cache only tickets and encryption keys (collectively called credentials) for a limited period, typically ~8 hours

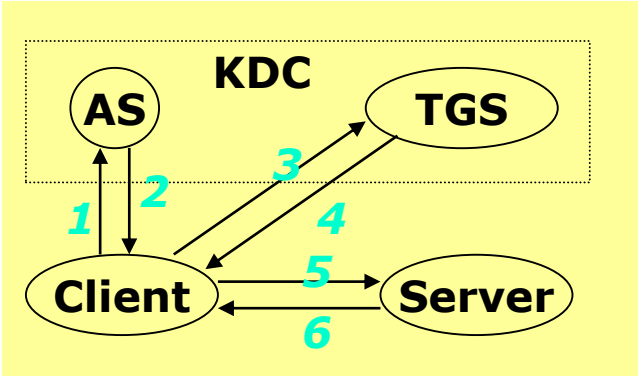
When the user first logs in, an authentication request is issued and a ticket and session key for the ticket granting service is returned by the authentication server

A special ticket, called a ticket granting ticket, is used to subsequently request a session key with a new verifier

The TGT may be cached

Complete Kerberos

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



Protocol

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

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

< client communicate 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 order server.
   - request = ({C, timestamp}Kct, {ticket}Kt, S) S: server key
4. TGS checks the ticket, If it is valid TGS generate a new random session key Kcs.
   TGS returns a ticket for S encrypted by Ks along with a session key Kcs.
   - return = ({ticket}Ks, {Kcs}Kct) ticket = (C, S, start-time, end-time, Kcs)

< client communicate with the server to access an application >

   client decrypt {Kcs}Kct with Kct to get Kcs.
   client generate authenticator A with the information from ticket.
   - A = ({C, S, start-time, end-time, address}Kcs)

5 . Client sends the service request to the server along with the ticket and A.
   - ({ticket}Ks, {C, S, start-time, end-time, address}Kcs, request

6. The server decrypt ticket using Ks and check if C, S, start, end times are valid
   If service request is valid, use Kcs in the ticket to decrypt authenticator
   Server compares information in the ticket and in the authenticator. If agreement, the service request accepted.

Kerberos Entities

Kerberos Key Distribution Center (KDC) consists of

- Kerberos Authentication Server (AS)
- Kerberos Ticket Granting Server (TGS)
- KDC supplies tickets and session keys

Realm

- Kerberos Administrative Domain that represents a group of principals
- A single KDC may be responsible for one or more realms

Principal

- Name of a user or service
- Principal Identifier: Unique identity for a principal (service/host@realm_name)
- Example: krbtgt/SYSSEC.UNI-KLU@SYSSEC.UNI-KLU

The Kerberos Ticket

A Kerberos Ticket contains of two parts:

- Unencrypted part
- Encrypted part

Fields of the unencrypted part:

- Version number for the ticket format
- Realm that issued a ticket
- Server identity

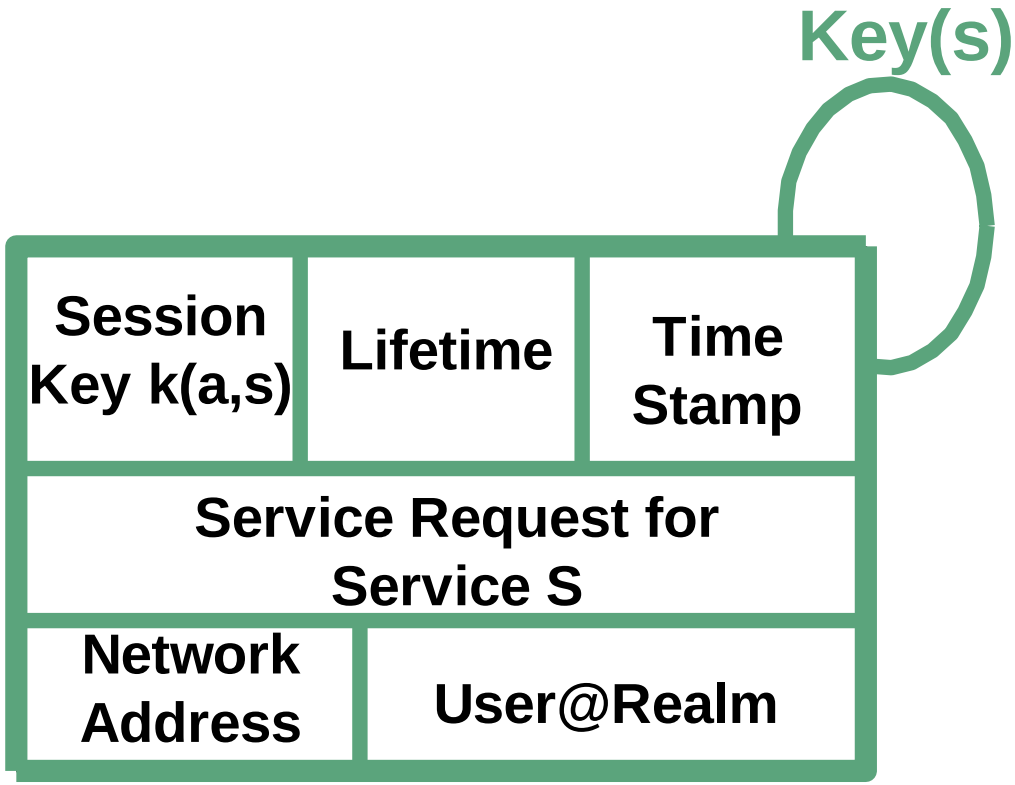
Fields of the encrypted part:

- Flags
- Key
- Client name/Client realm
- Transited
- Start-time, End-time, Renew-till
- Host addresses and authorization data

Example: Service Ticket

Service Ticket is encrypted with the secret key of the service S.

The ticket itself does not provide authentication. This is the responsibility of the Authenticator.



Comparison Kerberos V4/V5 (1/3)

Limitations with V4	Improvements with V5
Weak Timestamp mechanism	Nonce-based replay protection with KRB_PRIV and KRB_SAFE. Replay protection for the client in the AS and TGS msgs.
No authentication forwarding	Right delegation via forwardable and proxiable tickets
Reuse of “session keys” possible	No reuse possible, real session keys for KRB_PRIV and KRB_SAFE messages with sub-keys in AP_REQ
Flawed DES in cipher-block chaining mode	Standard DES in CBC mode

The AS and TGS response msgs are not double-encrypted in Krb V5 => U2U Auth.

Comparison Kerberos V4/V5 (2/3)

Limitations with V4	Improvements with V5
Limitations with principal naming	Less restrictions with a multi-component principal naming
Available for IP only	Multi-protocol support introduced
Cross-realm authentication requires n*(n-1)/2 keys between communicating realms	Hierarchy of realms introduced.
Only DES encryption algorithm available (export restrictions)	Generic interface supports several algorithms, still limitations exist
Problems with the Kerberos V4 pseudo-random number generator used for the session key generation (2^56 -> 2^20)	Problems fixed in Kerberos V5

Comparison Kerberos V4/V5 (3/3)

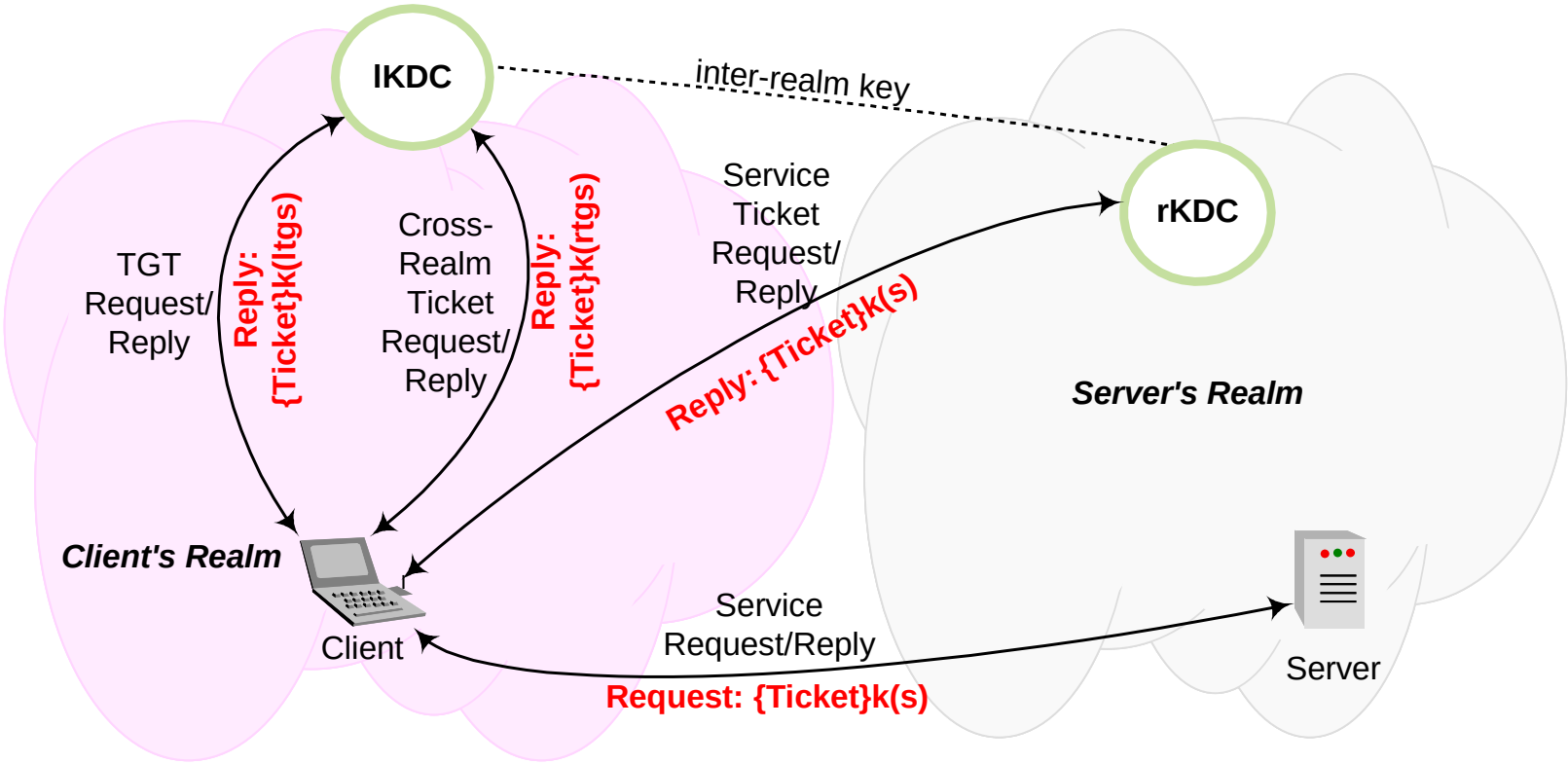
Limitations with V4	Improvements with V5
Sender encodes messages in his native format.	Messages are described and encoded with the ASN.1 syntax.
No batch processing support for tickets available.	Batch processing available with the help of postdated tickets.
Limited ticket lifetime(~21h)	Time format based on NTP -> very long lifetime
Weak message digest/checksum routines (CRC-32)	Several message digest routines available
No support for handheld authenticators (One-time Passwords)	Support added via the pre-authentication data field

Improvements with V5

Limitations with V4

Kerberos V4 Cross-Realm Authentication

Ticket Flow

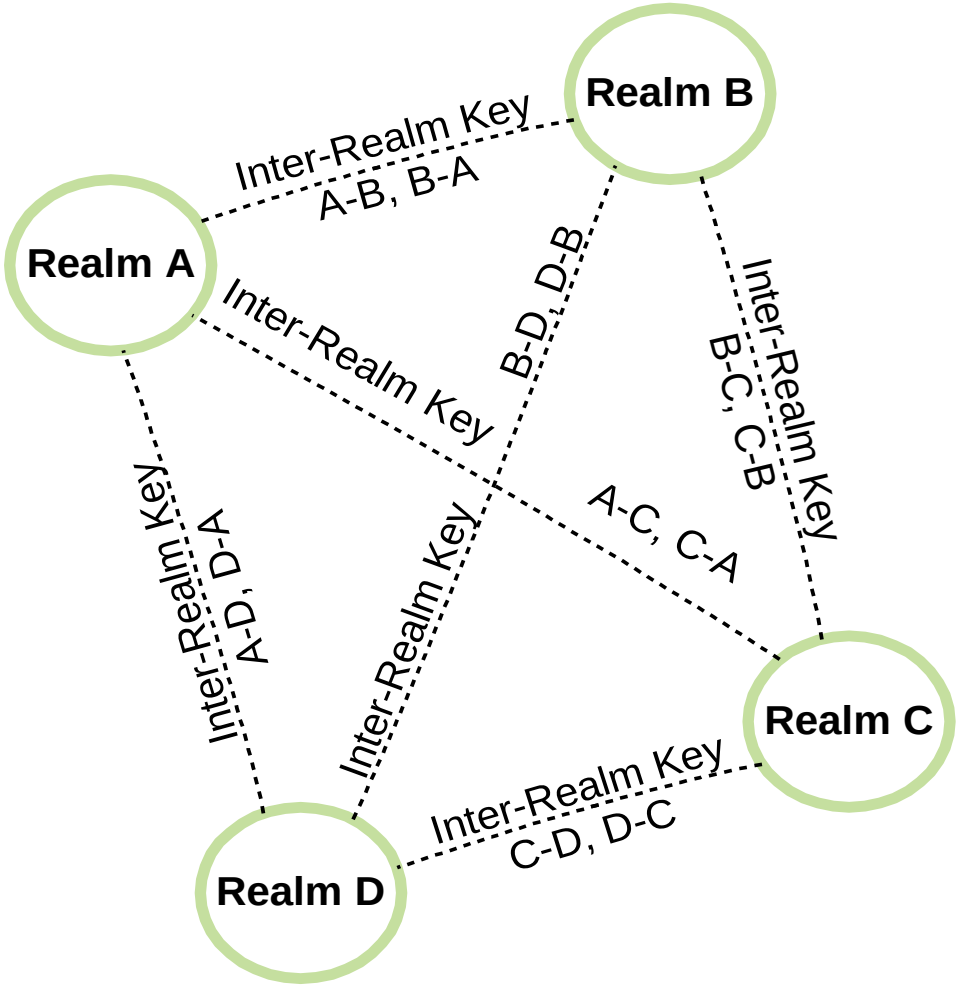


Kerberos V4 Cross-Realm

Realm navigation does not assume a realm-structure.

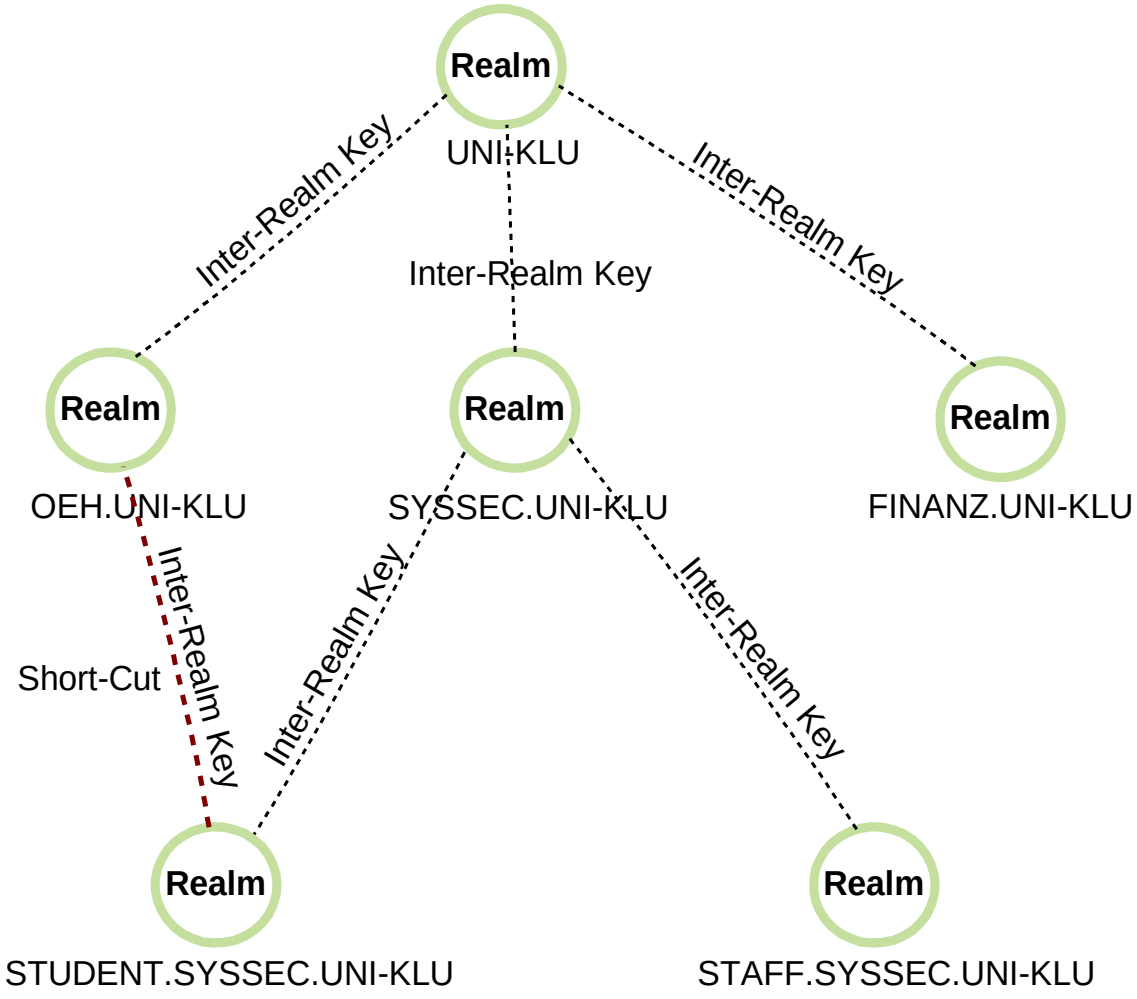
KDC must share a inter-realm key with all neighboring realms it wants to communicate with.

Scalability problems due to the complex key distribution.



Kerberos V5 Cross-Realm Improvement

- **Hierarchical structure may be used.**
- **Consulting a database is an alternative**
- **The client and the KDC run the same algorithm to determine the authentication path.**
- **Short-cuts limit the number of requests.**



Kerberos V5 Cross-Realm Authentication

The sequence of realms used in the authentication process is referred as the authentication path.

The client determines the authentication path by using a realm-naming convention similar to the DNS naming convention. The server runs the same algorithm but he may return a TGT that is closer to the final realm (if available).

Example:

- Client located at STUDENT.SYSSEC.UNI-KLU
- Server located at FINANZ.UNI-KLU
- Required TGTs:
  - krbtgt/STUDENT.SYSSEC.UNI-KLU@STUDENT.SYSSEC.UNI-KLU
  - krbtgt/SYSSEC.UNI-KLU@STUDENT.SYSSEC.UNI-KLU
  - krbtgt/UNI-KLU@SYSSEC.UNI-KLU
  - krbtgt/FINANZ.UNI-KLU@UNI-KLU

The transited path is the list of realms that were actually used to obtain the current ticket.

Kerberos V5 Ticket Types

Initial Ticket

- Indicates that this ticket is the result of a initial authentication.
- Used for ticket issued by the KDC and not by the TGS.
- Required by some programs (e.g. password changing programs)
- Gives the assurance that the user has typed in his password recently.

Invalid Ticket

- Validated by the KDC in a TGS request.
- Often used with postdated tickets

Postdated Ticket

- Purpose: Request a ticket for later use I.e. batch jobs
- Invalid until the start ticket has been reached
- Ticket must be sent to the KDC to convert it to a valid one.

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 sends a new ticket with a new session key back.

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

Where is Kerberos used?

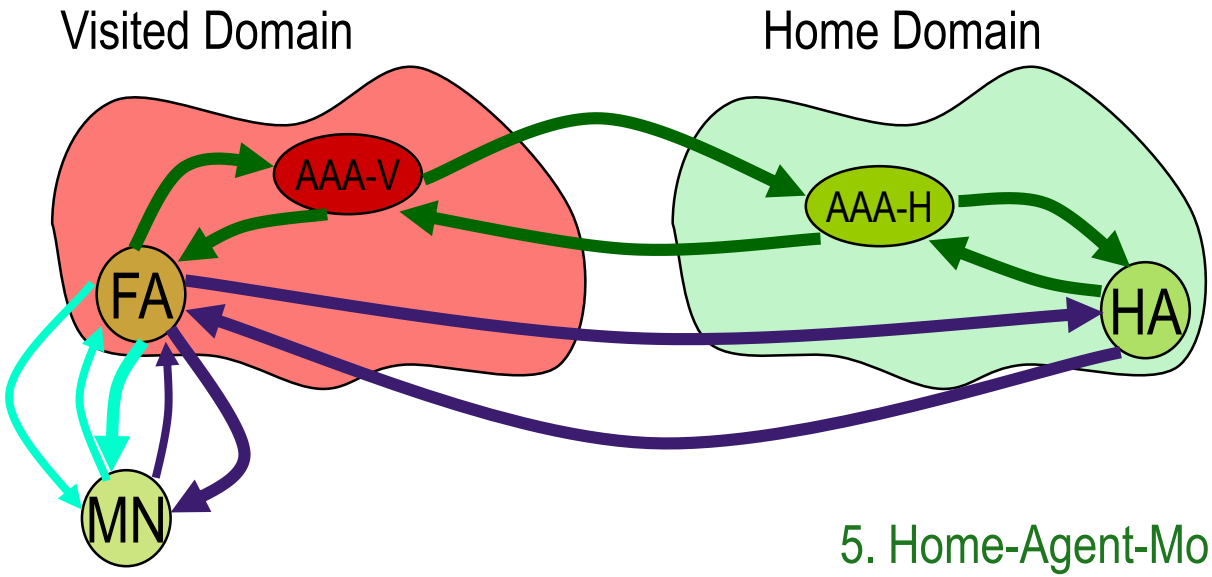
Architecture:
PacketCable

Operating Systems:
Unix
Windows 2000 for all authentication procedures
Windows CE .NET

Protocols (examples):
Resource Reservation Protocol (RSVP)
Telnet; NFS; FTP; SNMP; TLS; KINK; DNS

APIs / Carriers for Authentication Protocols
GSS-API; SASL; EAP;

AAA (Diameter) for MobIP V4



1. Agent advertisement + Challenge
2. Registration Request
3. AA-Mobile-Node-Request
4. Home-Agent-MobileIP-Request

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

5. Home-Agent-MobileIP-Answer
6. AA-Mobile-Node-Answer
7. Registration Reply
8. Registration Request
9. Registration Reply

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

What is IPSec?

IPSec is the standard suite of protocols for network-layer confidentiality and authentication of IP packets.

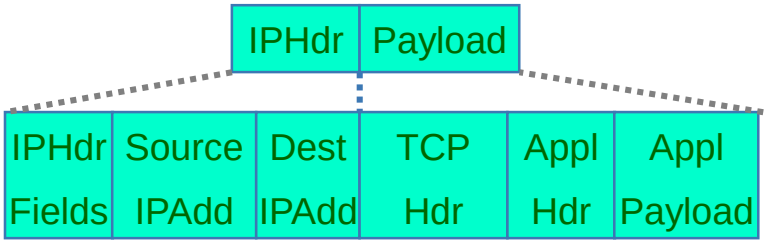
IPSec = AH + ESP + IPComp + IKE

In particular the following features are provided:

- Connectionless integrity
- Data origin authentication
- Replay Protection (window-based mechanism)
- Confidentiality
- Traffic flow confidentiality (limited)

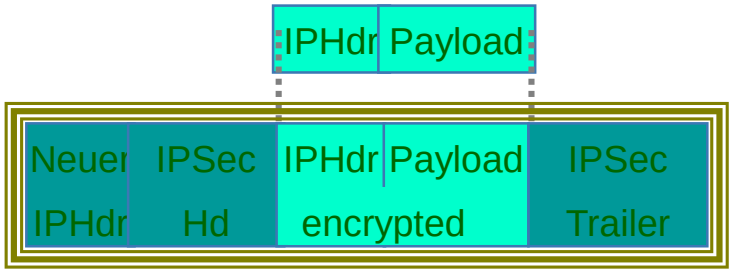
An IPv6 standard compliant implementation must support IPsec.

Insecured Messages vs. Secured Messages

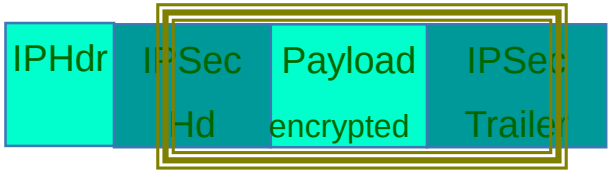


IP Spoofing
Session hijacking
Man-in-the-middle

Eavesdropping
Message modification

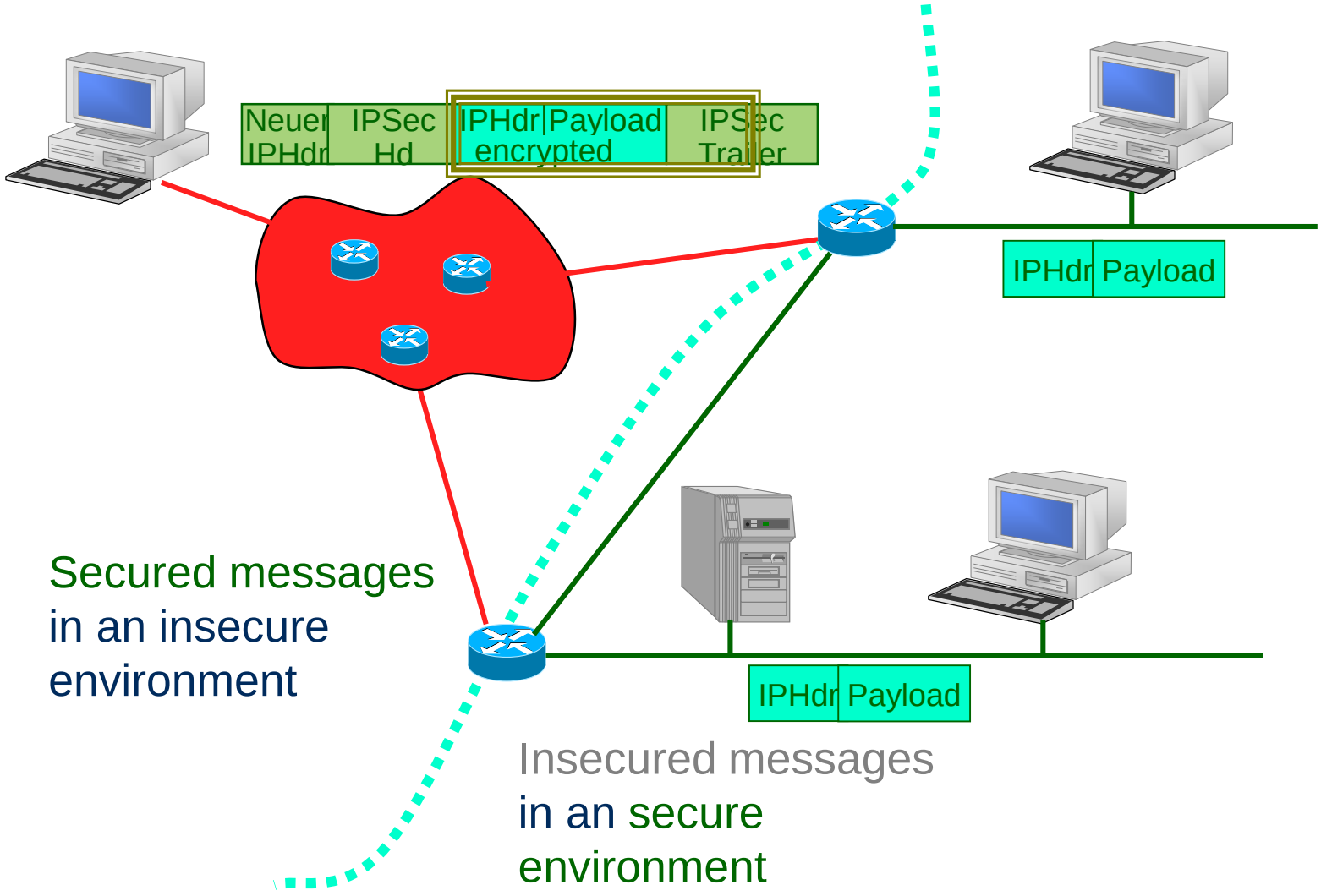


Tunnel mode:
the whole package is being encapsulated
in a new package



Transport mode (less expensive)
new IPSec Header (+ evtl Trailer)
provides somewhat less security

Use of IPSec: Tunnel Mode



Why IPSec?

Users want a secure, private network by

- disallowing communication to untrusted sites,
- encrypting packets that leave a site,
- authenticating packets that enter a site.

By implementing security at the IP level, all distributed applications can be secured (including many security-ignorant, legacy applications).

Typically, the following threats are prevented:

- Impersonation (IP Spoofing);
- Session hijacking;
- Man-in-the-middle Attacks;
- Injecting or re-ordering of IP packets
- Eavesdropping;
- Message modification

Tunnel Mode

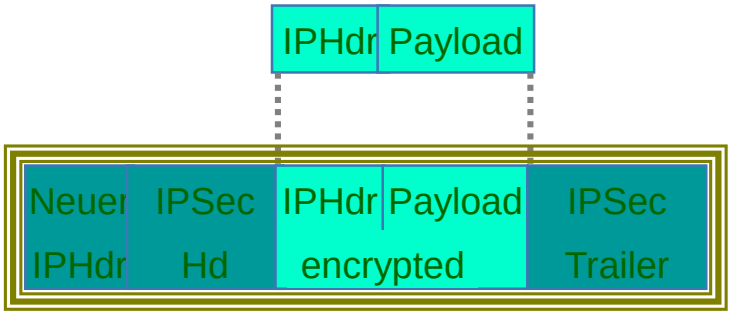
Tunnel Mode

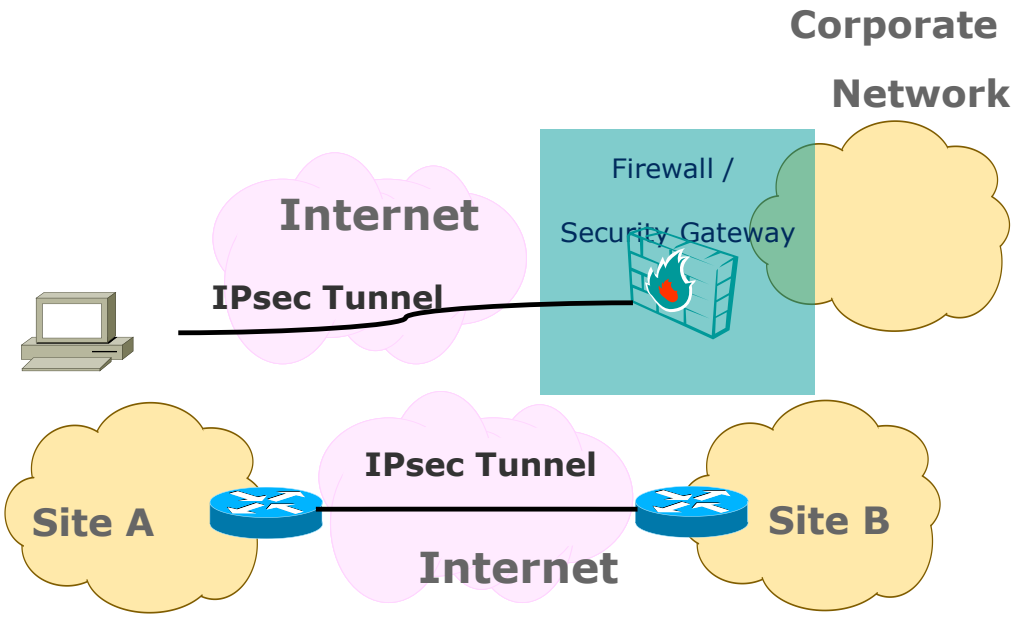
Tunnel mode has an "outer IP header" and "inner IP header"

- AH protects part of the outer header as well

Authentication is between remote host and firewall (or Security Gateway), or between two firewalls

User has access to entire internal network (VPN)





Transport vs. Tunnel Mode

Transport Mode

no additional header to the IP packet.

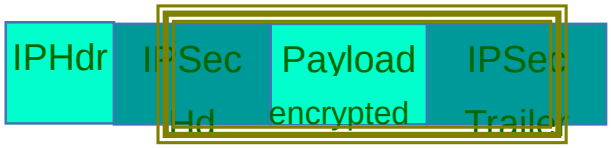
Authentication Header (AH) offers no confidentiality protection but protects parts of the IP header.

Encapsulating Security Payload (ESP) provides confidentiality protection.

Transport mode must be host to host

- adequate for upper layer protocols
- Gateways cannot handle fragmentation or multiple routes

Hosts share a secret key



IPSec SA

A Security Association (SA) is a data structure. The SA provides the necessary parameters to secure data. SAs can be established manually or dynamically (e.g. IKE).

An IPsec SA is uniquely identified by:

- Security Parameter Index, SPI (32 bit)
- Destination IP Address
- Protocol (AH or ESP)

IPsec SAs can support:

- Transport mode
- Tunnel mode

How to establish IPSec Security Associations?

Default Key Management Protocol:

The Internet Key Exchange Protocol (IKE)

Alternatives:

- Kerberized Internet Negotiation of Keys (KINK) (see http://www.ietf.org/html.charters/kink-charter.html)
- IKEv2 (SON-of-IKE)
- Host Identity Payload (HIP) (http://homebase.htt-consult.com/HIP.html; http://homebase.htt-consult.com/draft-moskowitz-hip-05.txt)
  - –HIP adds new namespace and provides a protocol for IPsec ESP SA establishment – not fully conformant to IPsec

Internet Key Exchange (IKE)

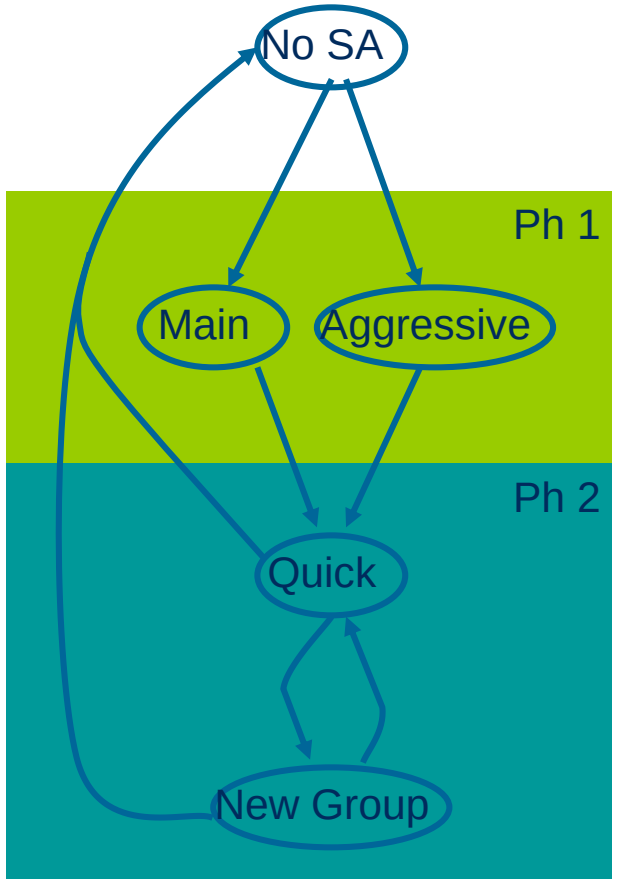
ISAKMP Phases and Oakley Modes

- Phase 1 establishes an ISAKMP SA
  - –Main Mode or Aggressive Mode
- Phase 2 uses the ISAKMP SA to establish other SAs
  - –Quick Mode
  - –New Group Mode

Authentication with

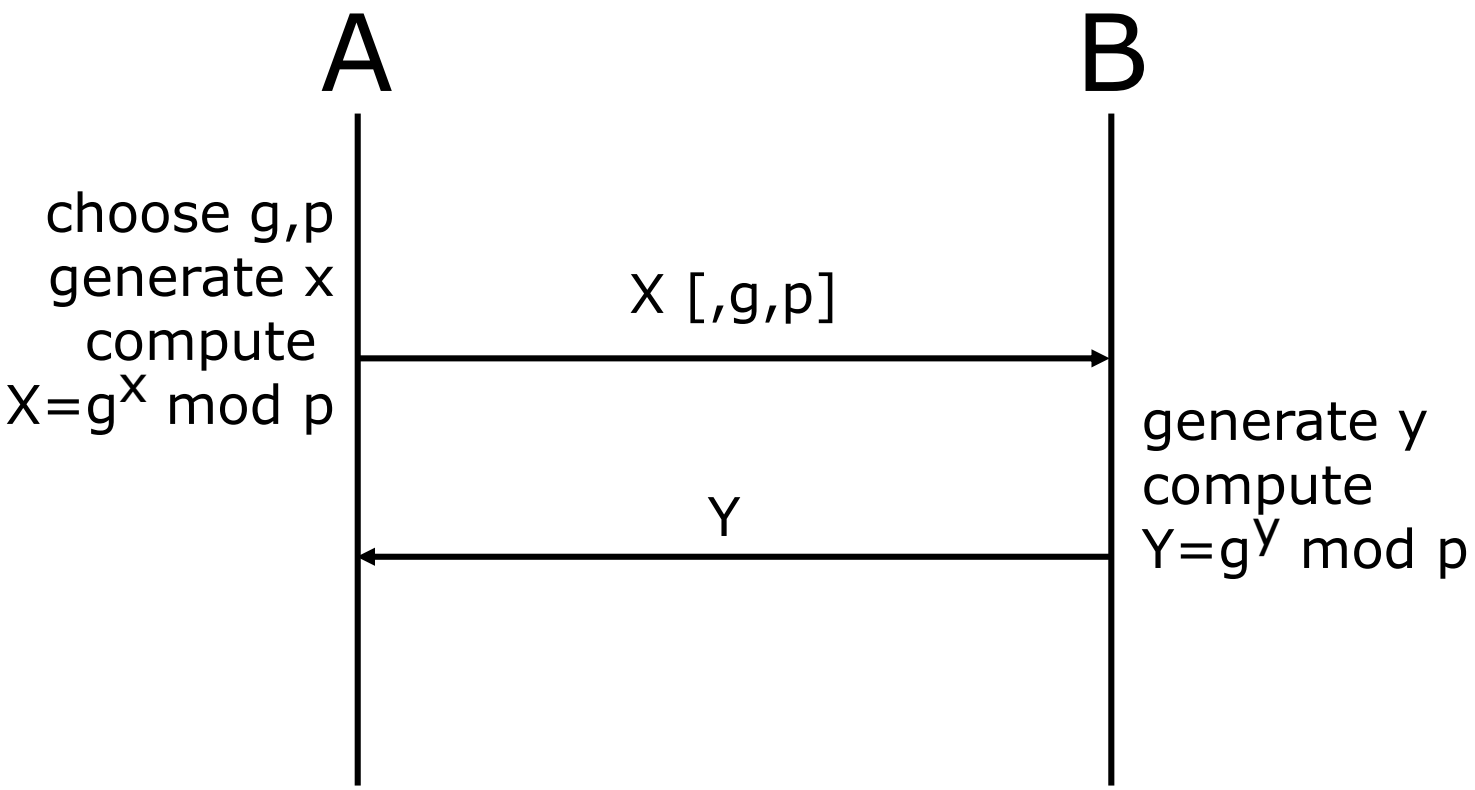
- Signatures
- Public key encryption
  - –Two versions
  - –Based on ability to decrypt, extract a nonce, and compute a hash
- Pre-shared keys

Four of the five Oakley groups



IKE states (simplified) modes and phases

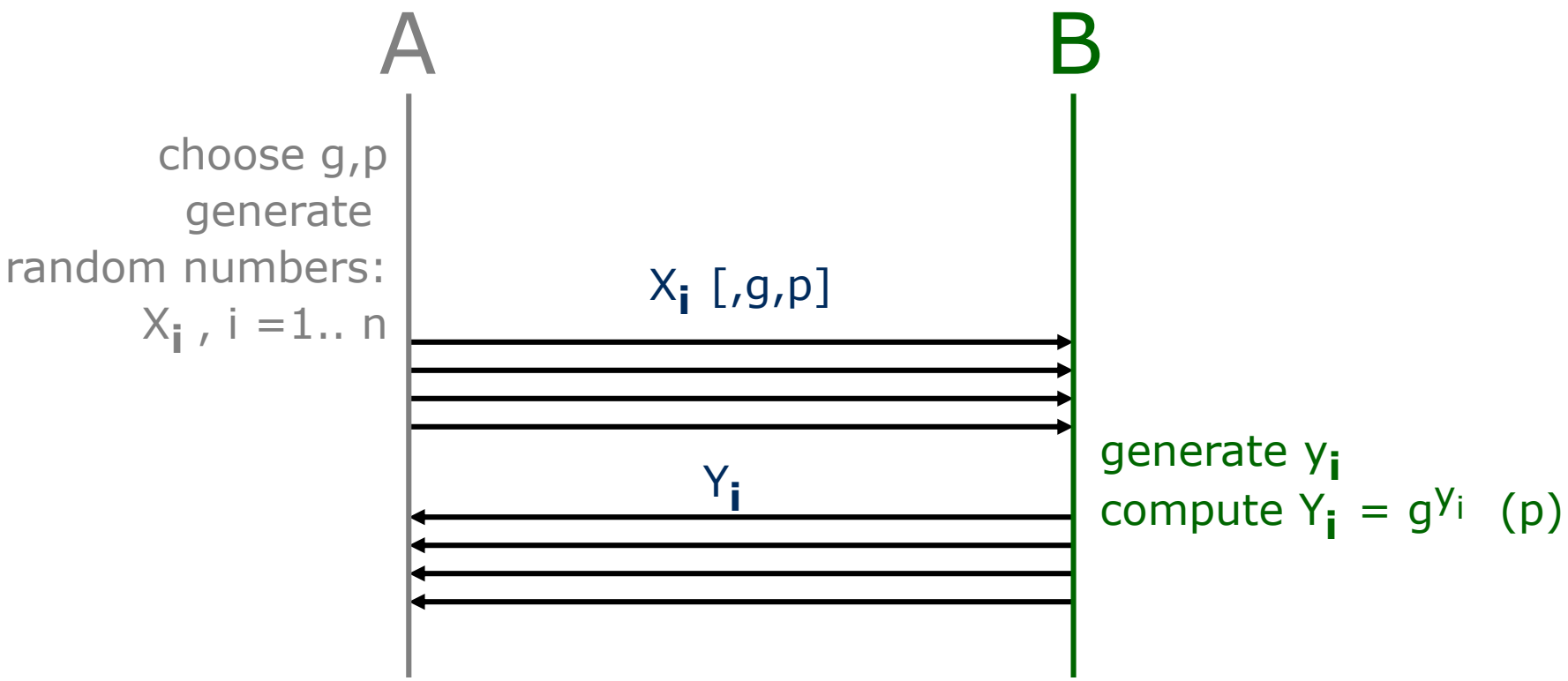
Diffie-Hellman



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

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

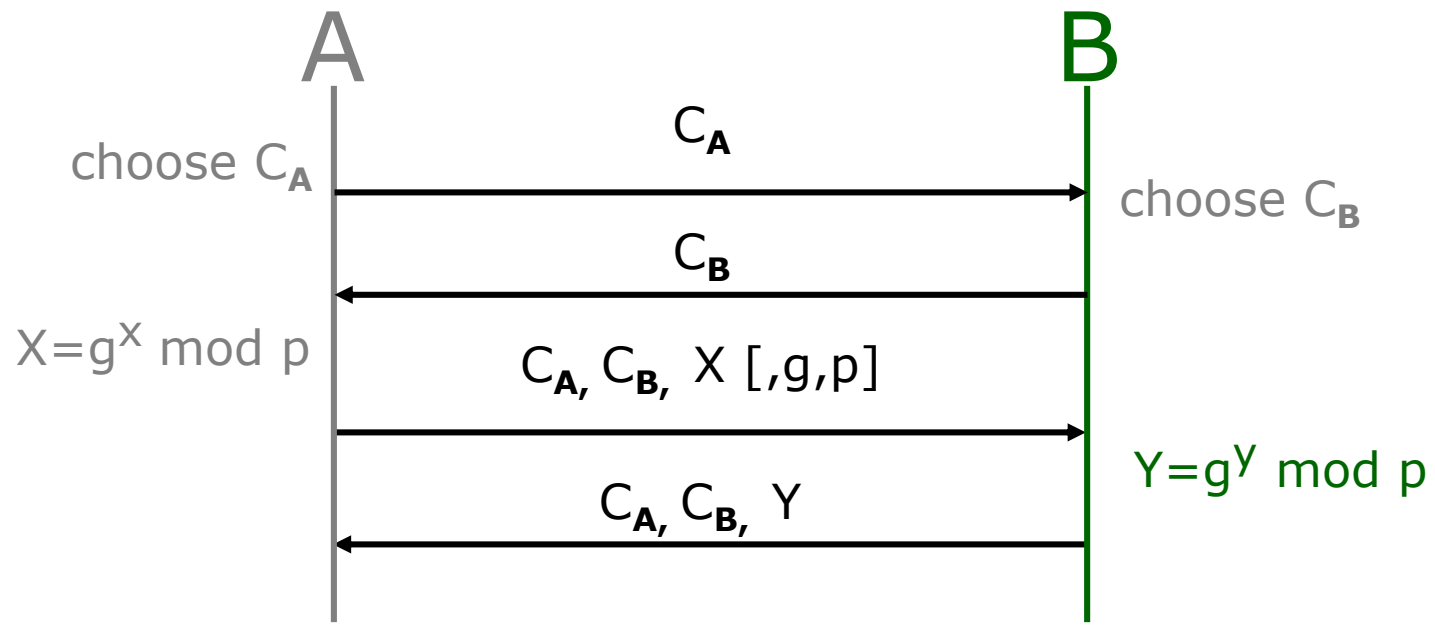
Denial of Service (Flodding)



DOS:

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

Dos Protection (Cookies)



Return routability proof:

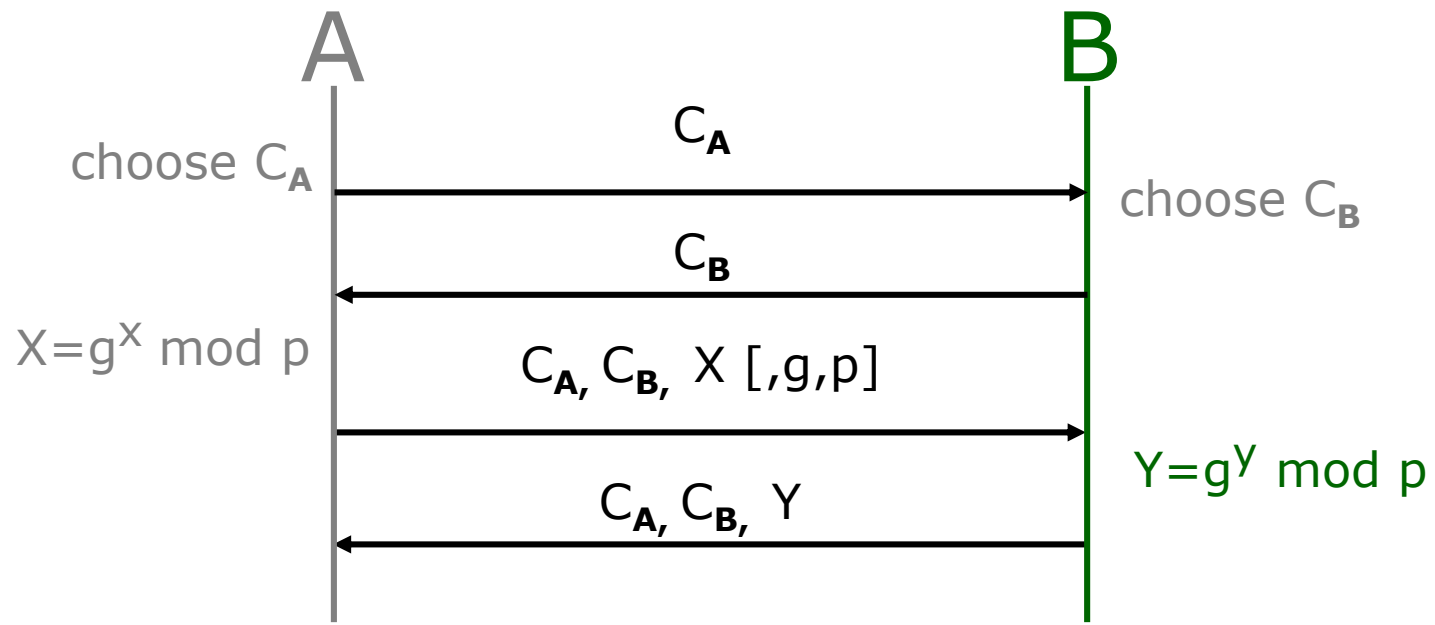
A has to have seen C_B to send the next msg

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



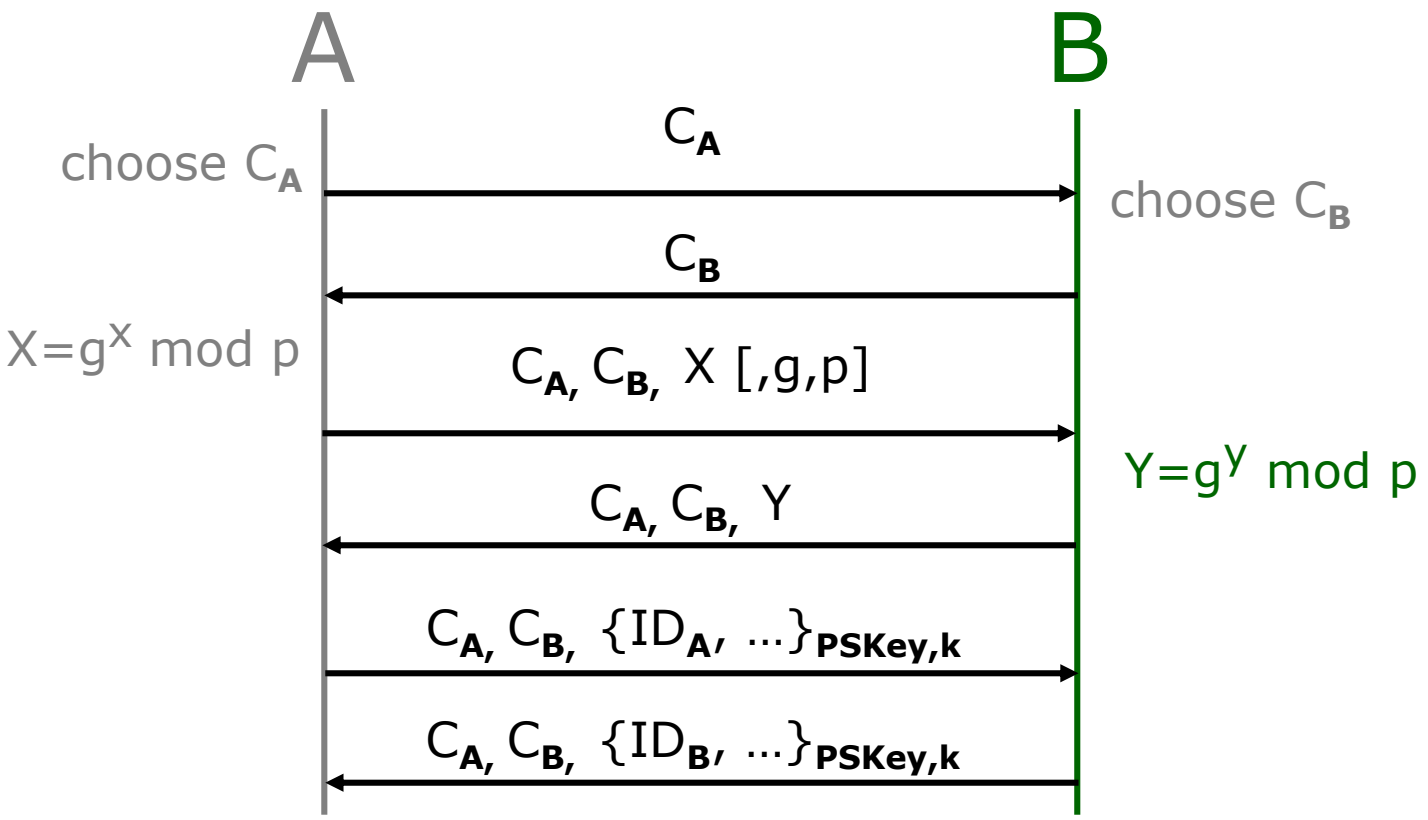
If A uses repeatedly same Address:

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

Problem remains:

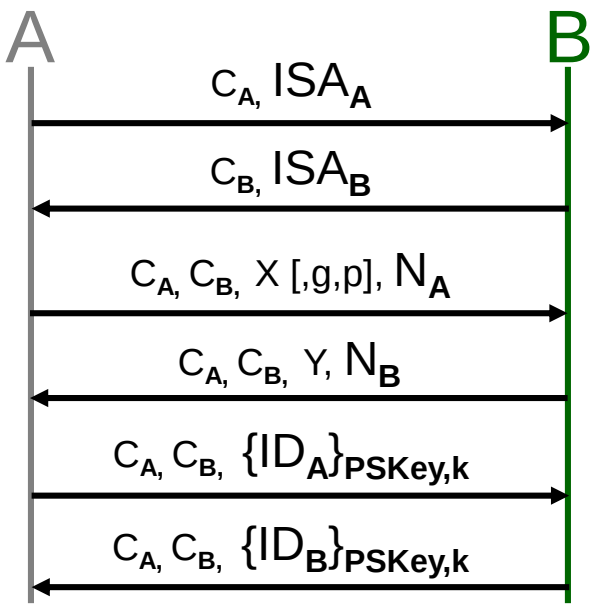
- Unauthenticated key-exchange:
  - man-in-the-middle

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



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

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

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

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

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

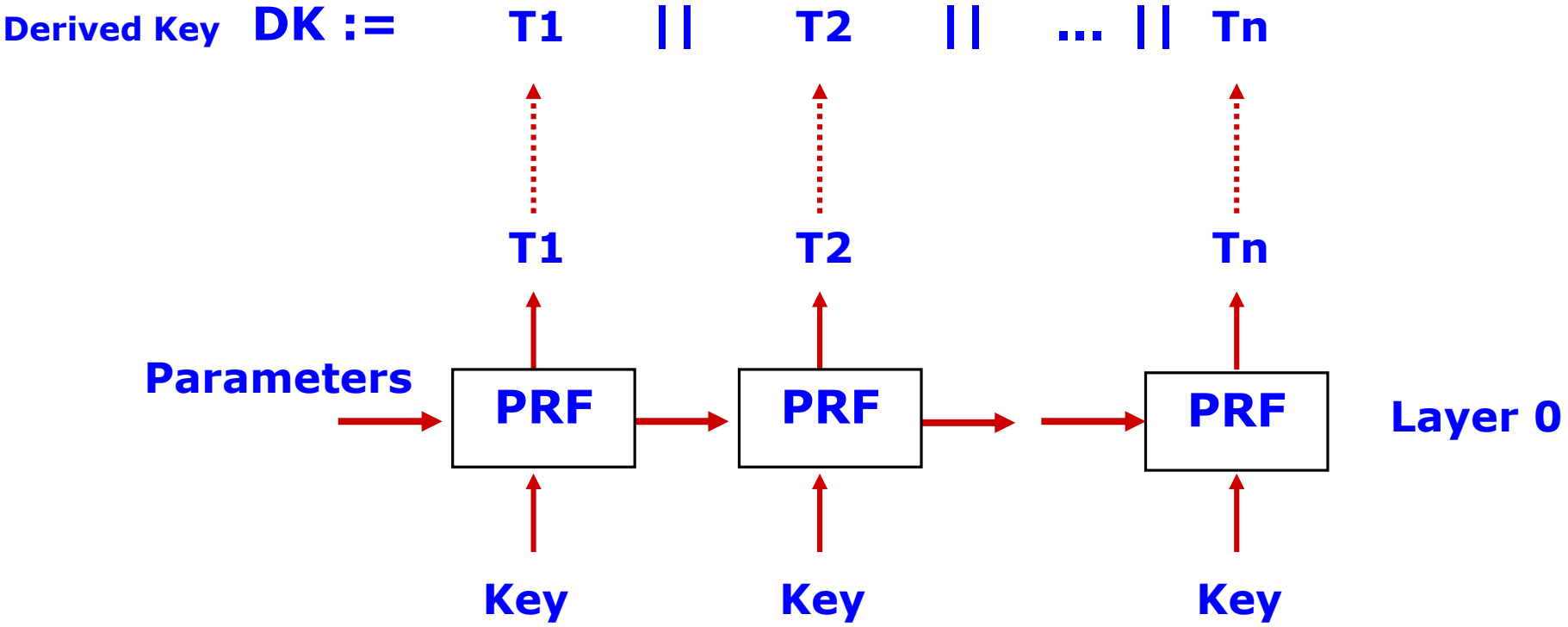
$\{ID_A\}_{PSKey,k} = (ID_A \mid Hash_A)$

ISA_A, ISA_B are ISAKMP SA Data, used by IKE to negotiate:

- encryption algorithm
- hash algorithm
- 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.

IKE (5): Key Derivation



Properties:

- IKE uses a key derivation procedure without a hierarchy.
- Key derivation provides key material of arbitrary length for the individual keys (encryption keys, integrity keys, IVs, etc. for different directions).
- The same key derivation routine is used to create an ISAKMP and an IPsec SA.

Internet Key Exchange (IKE) Summary (1/2)

Phase I

- The two peers establish a secure channel for further communication by negotiating ISAKMP SAs.

Phase II

- Protected by the SA negotiated in Phase I, the peers negotiate SAs that can be used to protect real communication; that is, the IPsec SA.

Internet Key Exchange (IKE) Summary (2/2)

IKE defines two Phase I modes:

- MAIN MODE gives authenticated key exchange with identity protection.
- AGRESSIVE MODE gives quicker authenticated key exchange without identity protection.

For Phase I, IKE defines (for main and aggressive modes) four different authentication methods:

- 1. authentication with digital signatures;
- 2. authentication with public key encryption;
- 3. authentication with a revised mode of public key encryption; and
- 4. authentication with a pre-shared key.

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 (optional) DoS protection

NAT handling covered in the core document

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

IPsec: Firewall to Firewall

Implement VPNs over the Internet.

Deployment already in progress; may some day largely replace private lines.

Caution: still vulnerable to denial of service attacks.

IPsec: Host to Firewall

Primary use: telecommuters dialing in.

Also usable for joint venture partners, clients, customers, etc.

But today's firewalls grant permissions based on IP addresses; they should use certificate names.

IPsec: Host to Host

Can we manage that many certificates?

Can servers afford it?

Can today's hosts protect their keys?

Limits to IPsec

Encryption is not authentication; we must still control access.

- Firewalls can't peek inside encrypted packets

Traffic engineers want to look inside packets, too.

New techniques for handling unusual links -- satellite hops, wireless LANs, constant bit rate ATM, etc. -- require examining, replaying, and tinkering with packets.

NAT boxes incompatible with end-to-end IPsec.

Use key recovery technology?

IPsec: IP security

Issues for IKE update (only minor corrections):

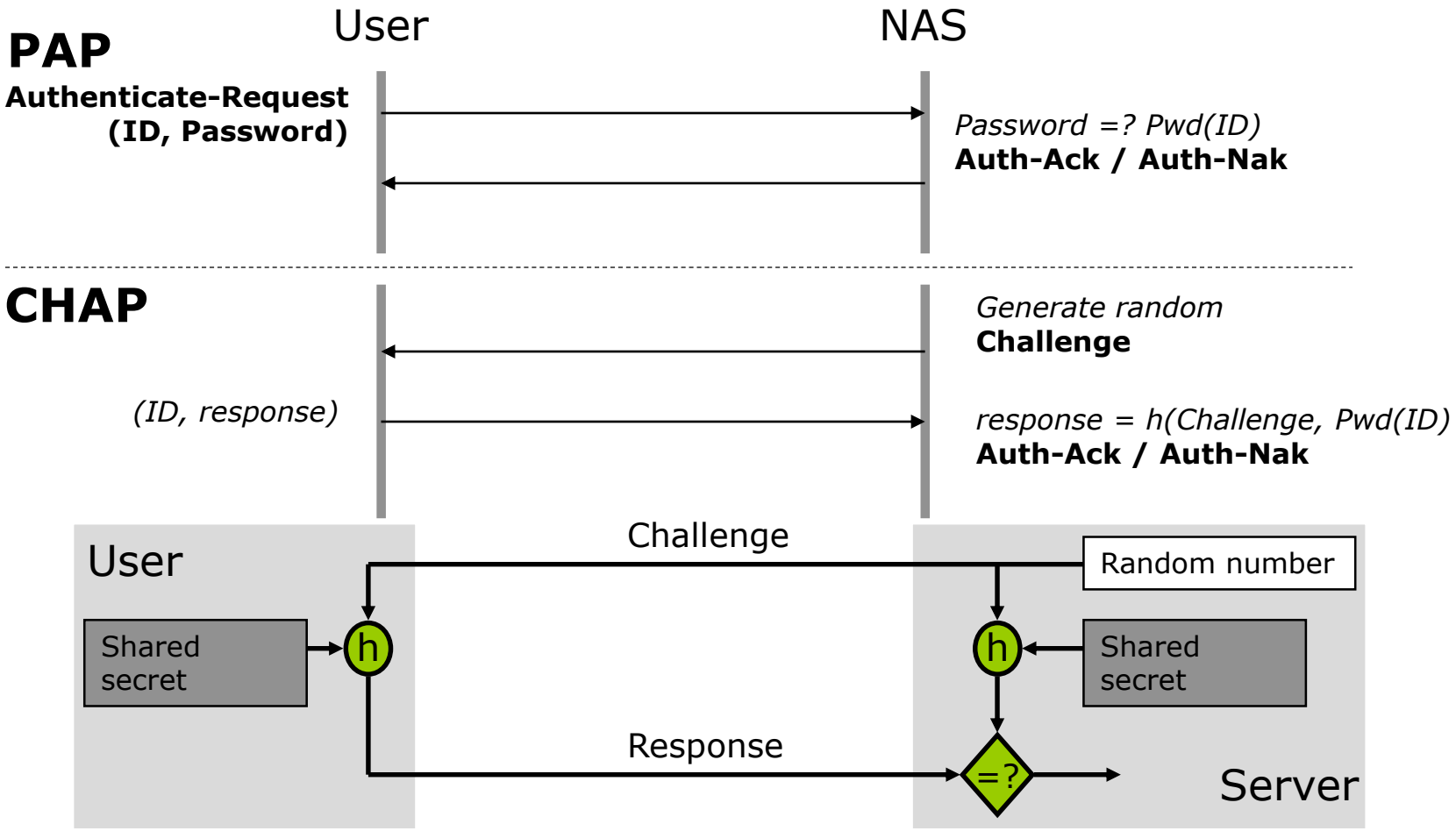
- NAT/Firewall traversal
- SCTP

Proposals for IKEv2 features/simplifications (new version):

- remote access
- dead-peer detection
- client puzzles for DoS protection
- remove most of the authentication methods
- remove perfect forward secrecy
- only one phase
- backwards compatibility
- …

Much discussion and several sets of proposals related to IKEv2

Network Access Example



Wireless Environments

Traditional network access procedures are not well suited for wireless environments.

Hence wireless network have to use different mechanism.

What about the security of IEEE 802.11?

IEEE 802.11 Background

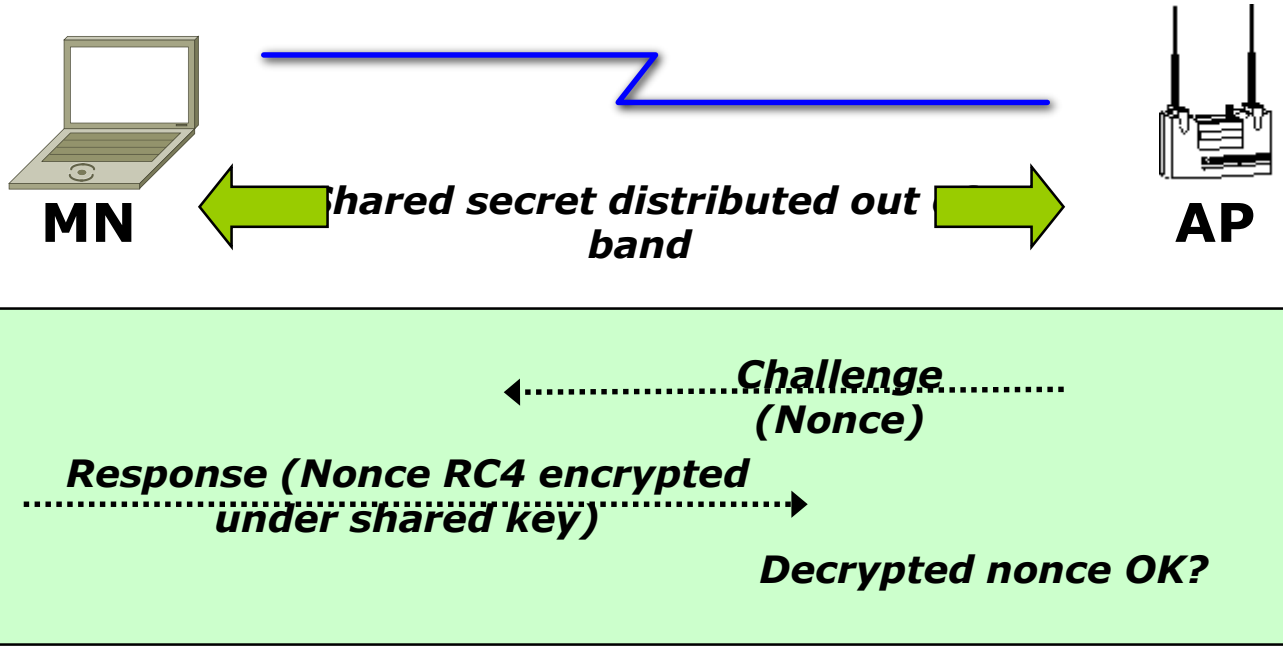
WEP (Wired Equivalent Privacy)

- Goal was: protection equivalent to the protection granted by wired LAN
- Secret key is shared between AP and all stations (40 or 104 Bit)
- Authentication based on Chall/Resp, but not mandatory
- No key distribution mechanisms
- WEP was developed behind closed doors
  - –as opposed to widespread practice today

Link layer security

- WEP key consists of Initialisation Vector (IV) concatenated with shared key
- IV is 24 Bit long, no rules about usage
- Encryption is based on RC4 (a stream cipher)
  - –Generates an "endless" key stream
  - –Key stream is bit-wise XORed with plaintext
  - –General Rule: never use key stream twice, but: 24 Bit revolves quickly

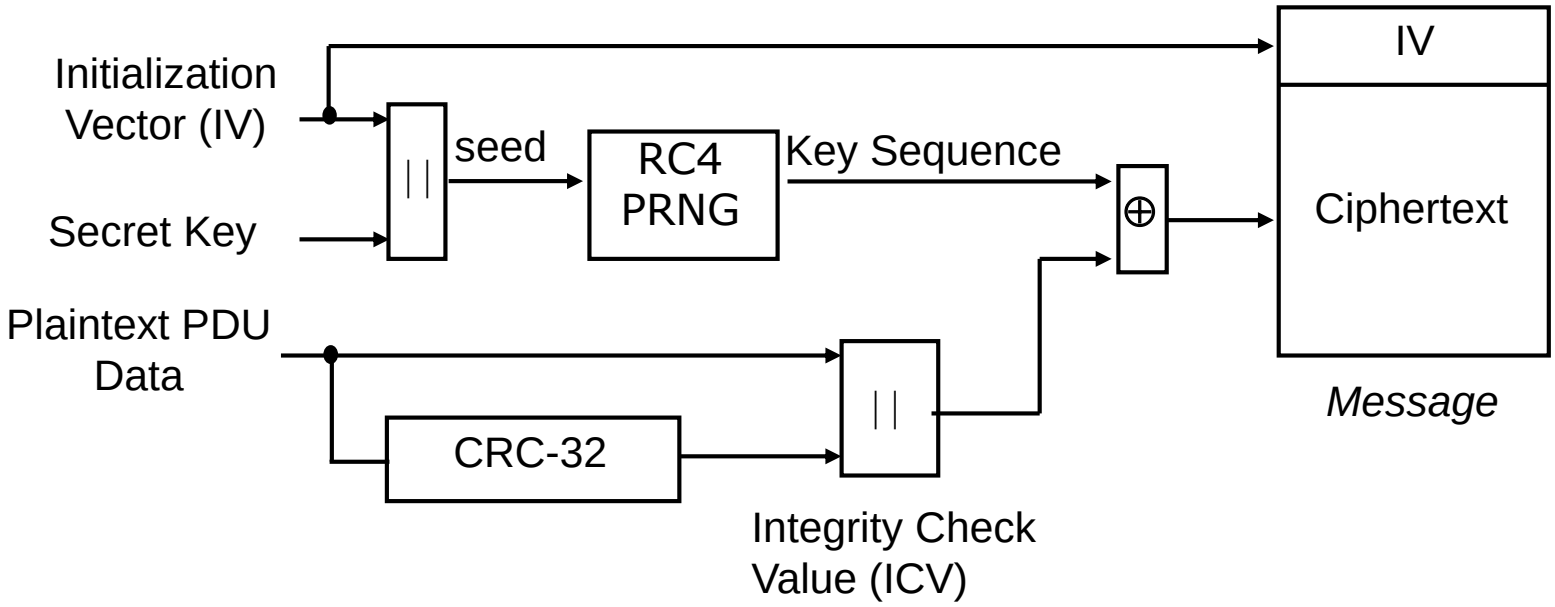
Wireless Equivalence Privacy (WEP) Authentication



802.11 Authentication Summary:

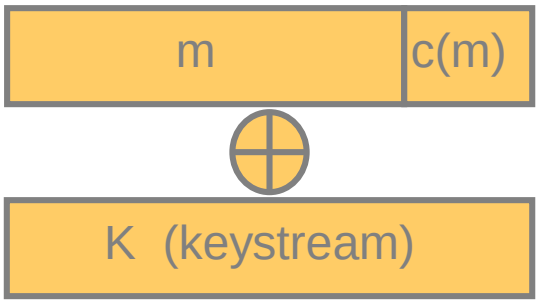
- Authentication key distributed out-of-band
- Access Point generates a “randomly generated” challenge
- Station encrypts challenge using pre-shared secret

WEP Encryption



WEP in brief:

Sender and receiver share a secret key k.



To transmit m:

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

Recipient:

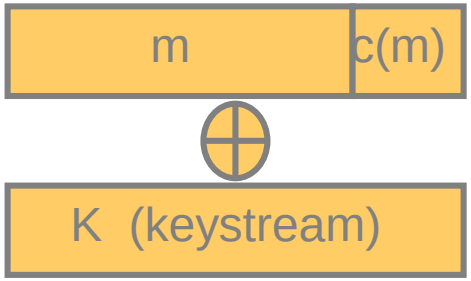
- Use the transmitted iv and k to generate K = rc4(iv,k)
- <m',c'> := C ⊕ K $=^{\text{ifOK}}$ = (M ⊕ K) ⊕ K = M
- If c' = c(m'), accept m' as the message transmitted

Attacks involving keystream reuse (collision)

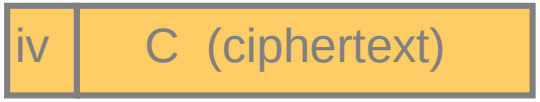
If m1 and m2 are both encrypted with K,

⇒ C1 ⊕ C2 = m1 ⊕ K ⊕ m2 ⊕ K
= m1 ⊕ m2

⇒ intruder knows ⊕ of two plaintexts!



Pattern recognition methods:
know m1 ⊕ m2 ⇒ recover m1, m2.



K = rc4(iv,k).

k changes rarely and shared by all users

Same iv ⇒ same K ⇒ collision

iv cleartext ⇒ 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++): low iv values get reused often

Decryption Dictionaries

pings, mail ⇒ intruder knows one pair ciphertext and the plaintext for some 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

The next time he sees a packet with iv in the table, he can just look up the K and calculate $M = C \oplus K$

size of the table depends only on the number of different iv you see.

It doesn't matter if you're using 40-bit keys or 104-bit keys

If the cards reset iv to 0, the dictionary will be small!

Message “Authentication” in WEP

The checksum algorithm used is CRC-32

CRC's detect random errors; useless against malicious errors:

- It is independent of k and iv
- It is linear: $c(m \oplus D) = c(m) \oplus c(D)$

Message Modification

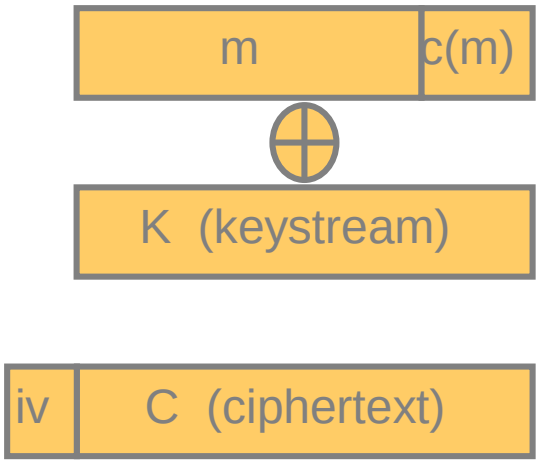
Assume IV and C are known to intruder .

Intruder wants the receiver to accept fake message

F = m ⊕ d

for some chosen d

($$ in a funds transfer)



D := (d | c(d)), then (C ⊕ D) = K ⊕ (M ⊕ D)

C' := C ⊕ D transmit (iv,C') to the receiver.

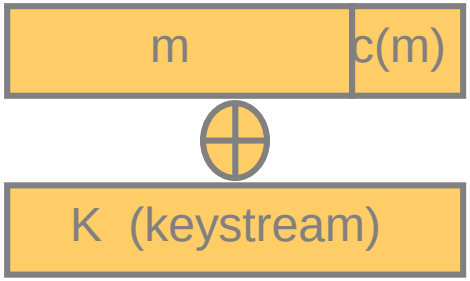
Receiver checks:

C' ⊕ K = C ⊕ D ⊕ K = M ⊕ D = <F, c(F)>

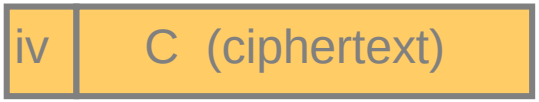
OK!

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
$C' = \langle F, c(F) \rangle \oplus K$.

Transmits (iv,C').

Message Injection

Note that we only used that the CRC does not depend on the key. The attack would work just as well if the CRC were replaced by, say, SHA-1.

The Authentication Protocol

AP sends challenge

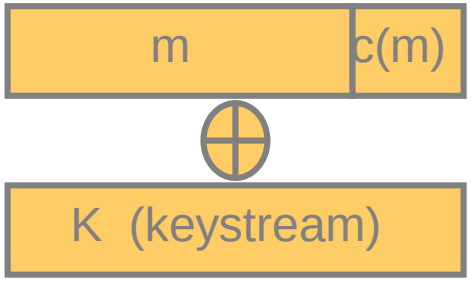
The client sends back the challenge, WEP-encrypted with the shared secret k

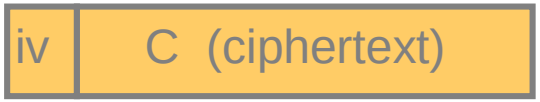
AP checks if the challenge is correctly encrypted

Intruder: has now both the plaintext and the ciphertext of this challenge!

Authentication Spoofing

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

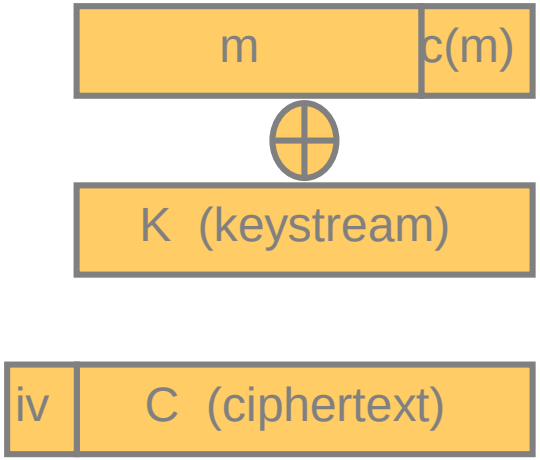




Now he connects to the network himself:

- AP sends a challenge m' to intruder
- Intruder replies with iv, <m',c(m')> ⊕ K
- This is in fact the correct response, so AP accepts intruder
- Without knowing k

Message Decryption



Intruder can trick AP into decrypting the packet, and telling him the result :

Double-encryption

IP Redirection

Reaction attacks

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 => silently dropped
- But if the TCP checksum on the modified packet is correct, 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

Attacking the WEP Algorithm

Passive attacks

- Eavesdropping packets with same IV $\Rightarrow$ yields XOR of two (or more) plaintexts and allows conclusions about plaintext
- Eavesdropping packets with "special IVs" $\Rightarrow$ allows to reconstruct the WEP key (=> Airsnort attack)

Active attacks

- Injecting know plaintext packets from the Internet (packet sent with selected IV for a known key stream) $\Rightarrow$
  - –Allows to decrypt all packets with same IV
  - –Allows to encrypt own plaintext with same IV
  - –Allows to built a lookup table for many (all) IVs (space required for all IVs ~15GB)
- Authentication possible without knowledge of the key (Known plaintext attack - challenge / response)

IEEE 802.11 Security weaknesses

- The properties provided by IEEE 802.11 do not meet today's security objectives
- The missing user identification and the non-existing appropriate key management makes it difficult to detect unusual activity.
- Authentication is based on the MAC address and not on the user identity.
- Mutual authentication not provided (false base-station attacks possible)
- No keyed message digest used
- 40-bit RC4 key length too short for today's application (because of US export restriction)
- Too short Initialization Vector (24 bits)
- Known (and partially known) plain-text attacks possible

Current Status of WLAN Security

802.11 Task Group i deals with enhanced security for 802.11 WLANs

Standard expected for end 2003

Short-term solution: TKIP (Temporal Key Integrity Protocol)

- Idea: reuse 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

Authentication and key management: 802.1X "Port-based access control"

- Mutual authentication between STA and backend authentication server
- Establishment of individual per-session keys between STA and AP

Long-term solution: AES-CCMP (AES-Counter-Mode/CBC-MAC protocol)

- Robust security solution
- Requires new hardware

WEP Security: Lessons

WEP designers selected well-regarded algorithms, such as RC4

But used them in insecure ways

The lesson is that security protocol design is very difficult

- best performed with an abundance of caution,
- supported by experienced cryptographers and security protocol designers
- and tools!

IEEE 802.1X Security Properties

- Support flexible security framework based on EAP (RFC 2284) and RADIUS
- Enable plug-in of new authentication, key management methods without changing NIC or Access Point
- Enables customers to choose their own security solution
- Can implement the latest, most sophisticated authentication and key management techniques with modest hardware
- Enables rapid response to security issues
- Per-session key distribution

IEEE 802.1X Security Properties

Enables use of Kerberos v5 for authentication

Allows fine-grain authorization:

- Authorization can include bandwidth limits, Virtual LAN, QoS, etc.

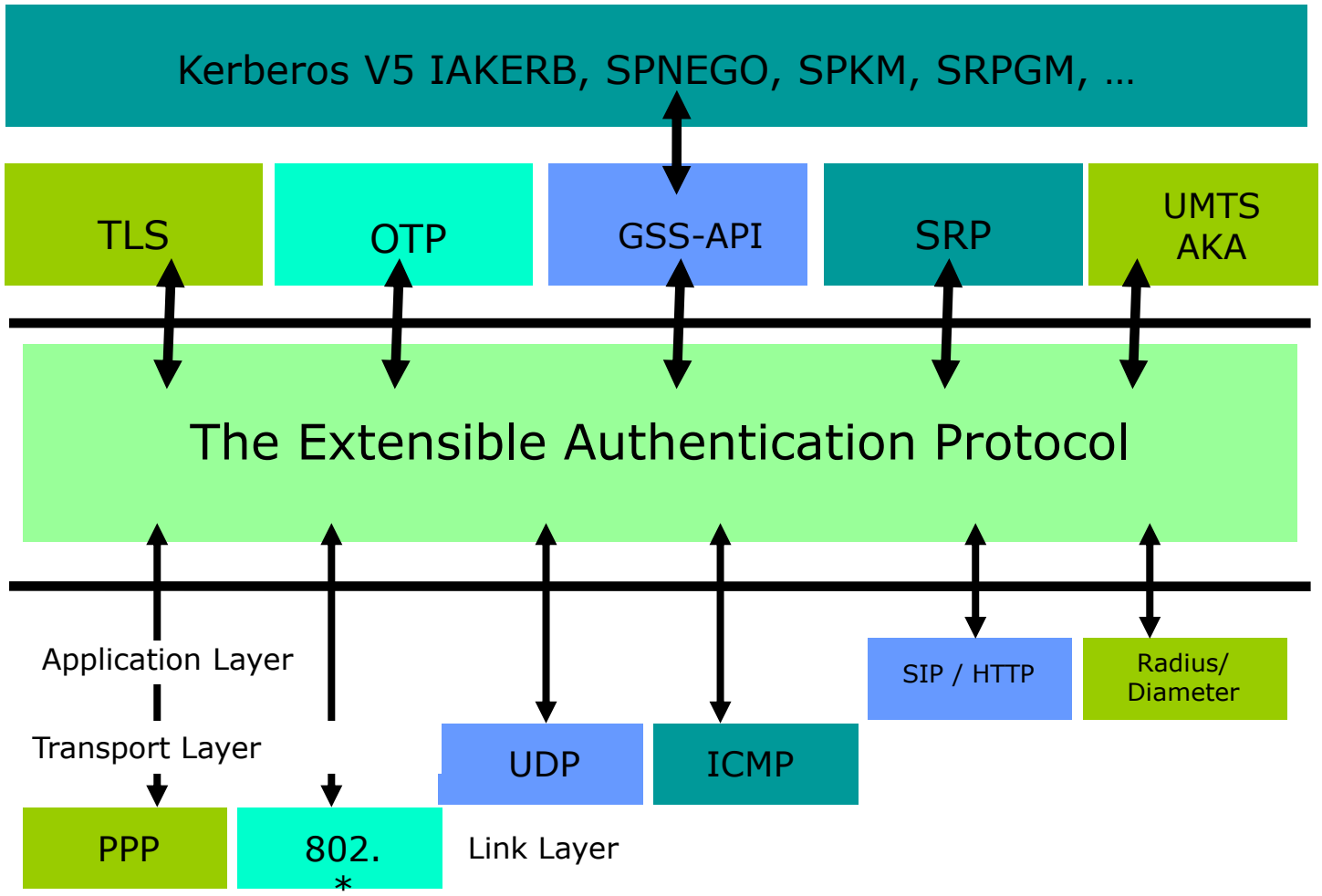
User-based identification

- Identification based on NAI (Network Access Identifier, RFC 2486)
- Allows cross-realm access in public places

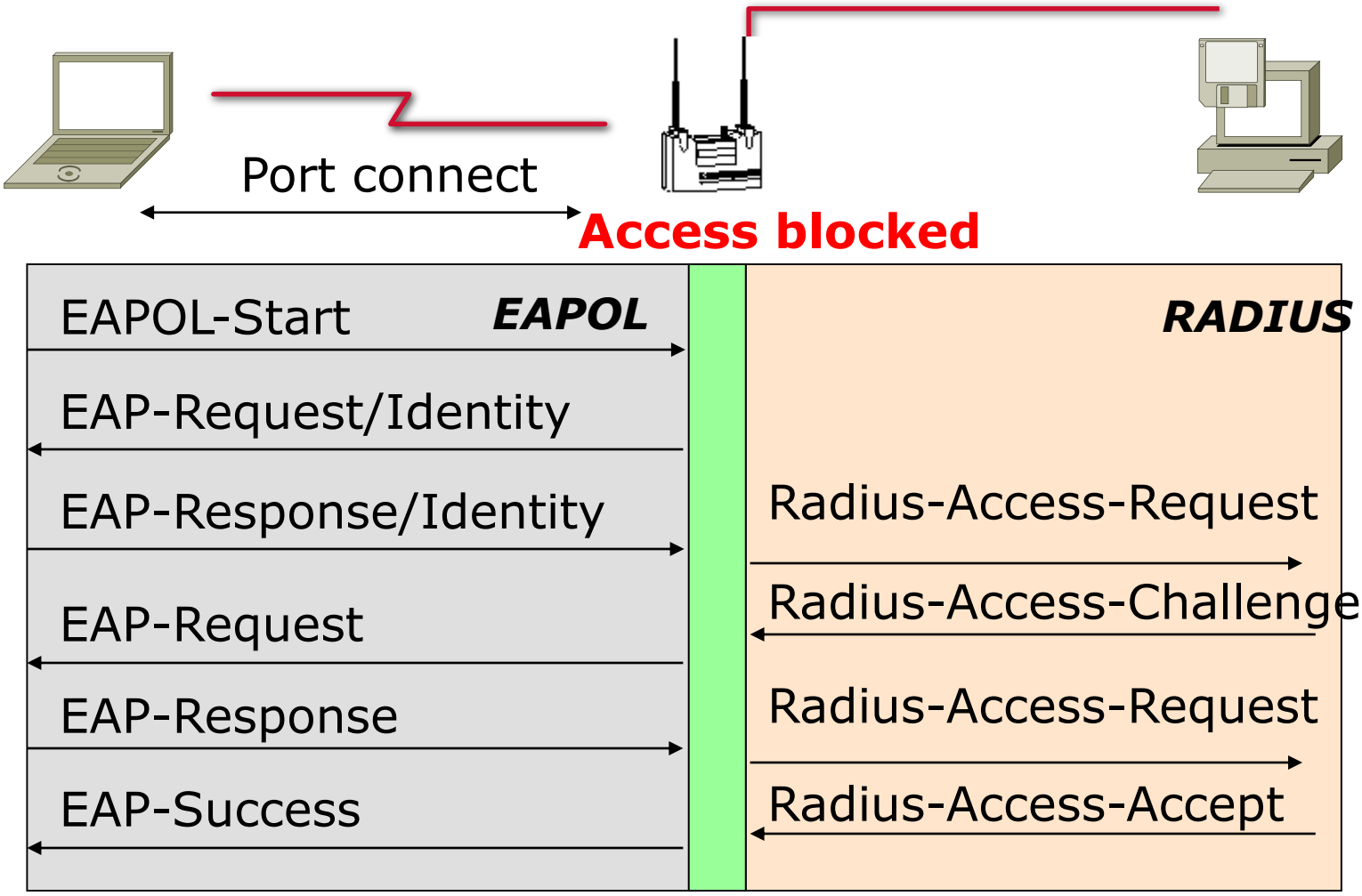
Receives wide support in the industry

- 3Com, Intel, HP, MERIT, Microsoft, Nortel, Cisco

EAP Architecture



IEEE 802.1X EAP/Radius Conversation



Purpose of Digital Certificates

Scalability

Trusted validation of parties

Transmission and storage of public keys can be insecure

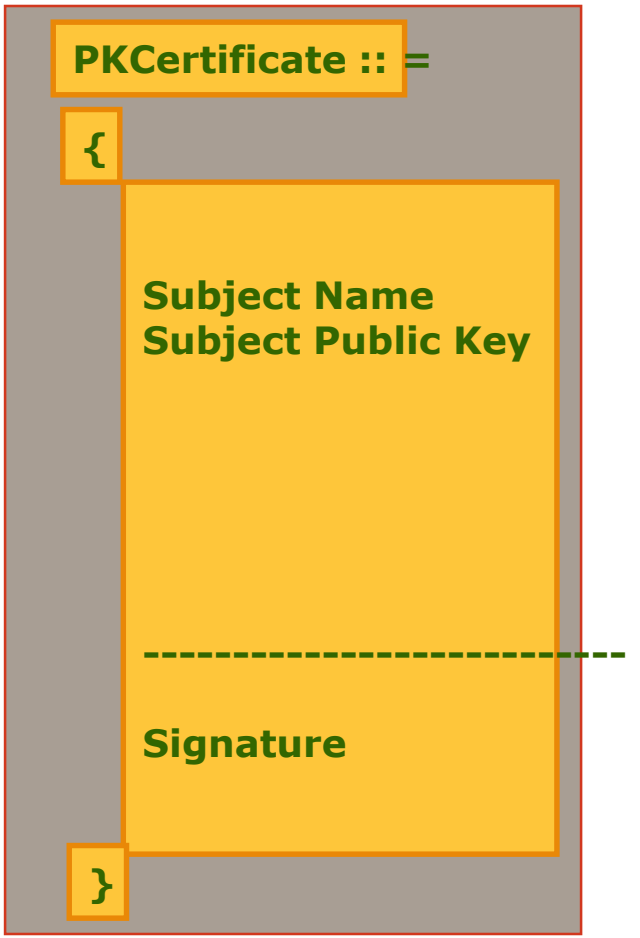
Can provide permissions (Authorizations)

X.509 is part of the ITU-T Directory series of recommendations (= ISO/IEC 9594).

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



X.509 Public Key Cert V.1

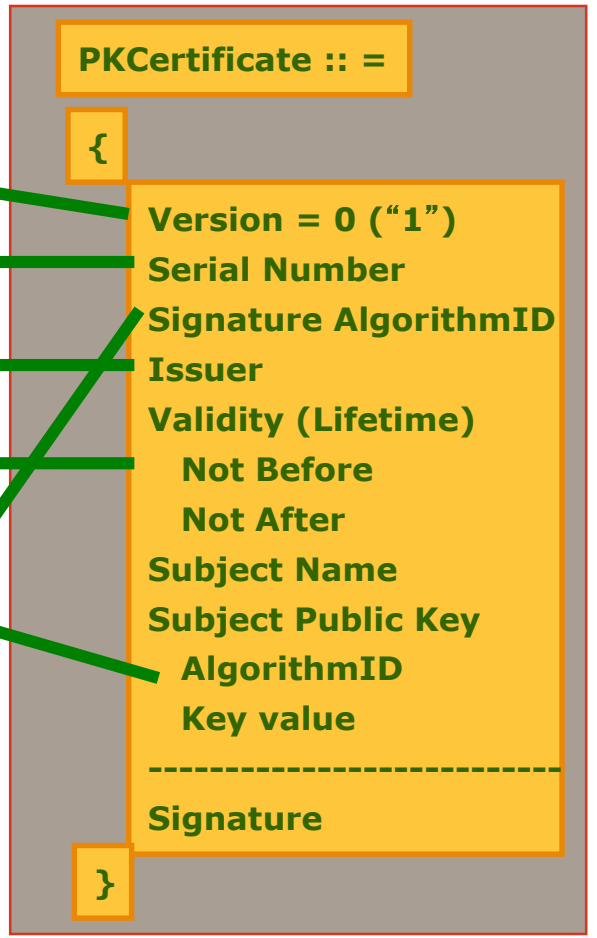
Version 1 from 1988

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)



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

(Single) Certificate Validation

Check the Certificate Integrity

Validity Period

Key Usage and Applicability according to policies

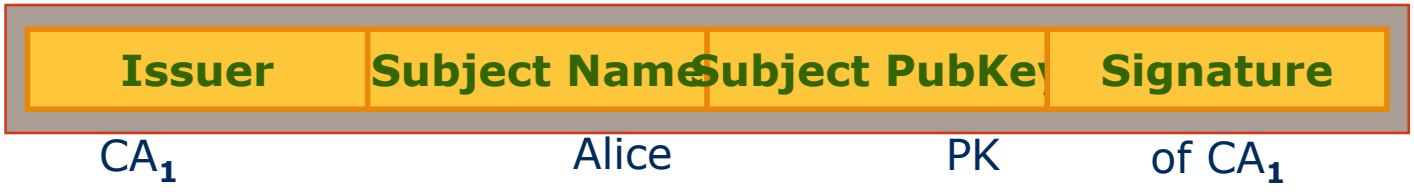
Certificate Status

How do I Verify this Certificate?

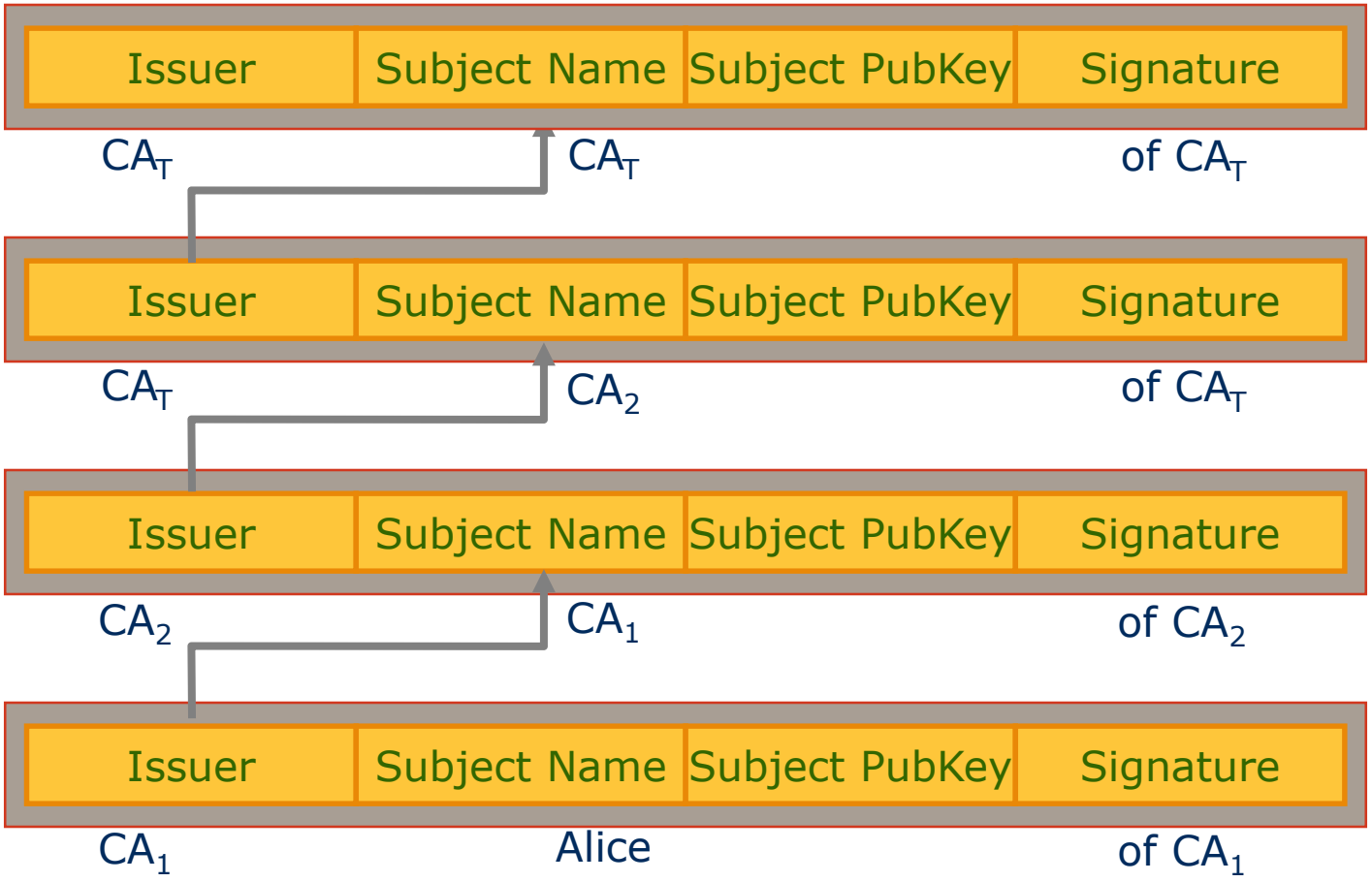
Alice wants me to believe that she owns a certain public key PK.

For that, she presents me a Certificate, issued by her company "CA_1".

But who is that company, "CA_1"?
Is CA_1 trustworthy?
Is "Signature of CA1" really the signature of CA1?

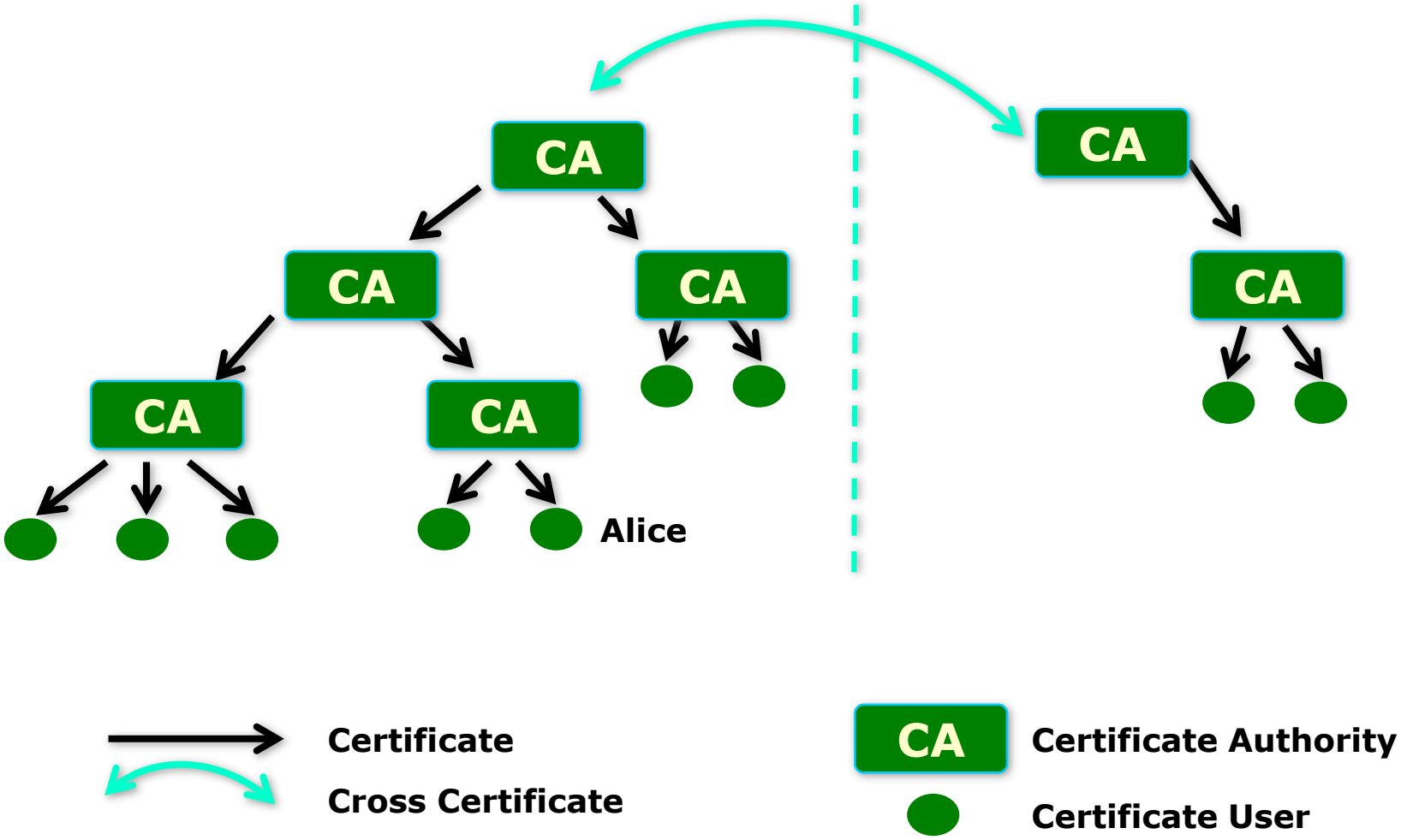


Path Construction and Path Discovery

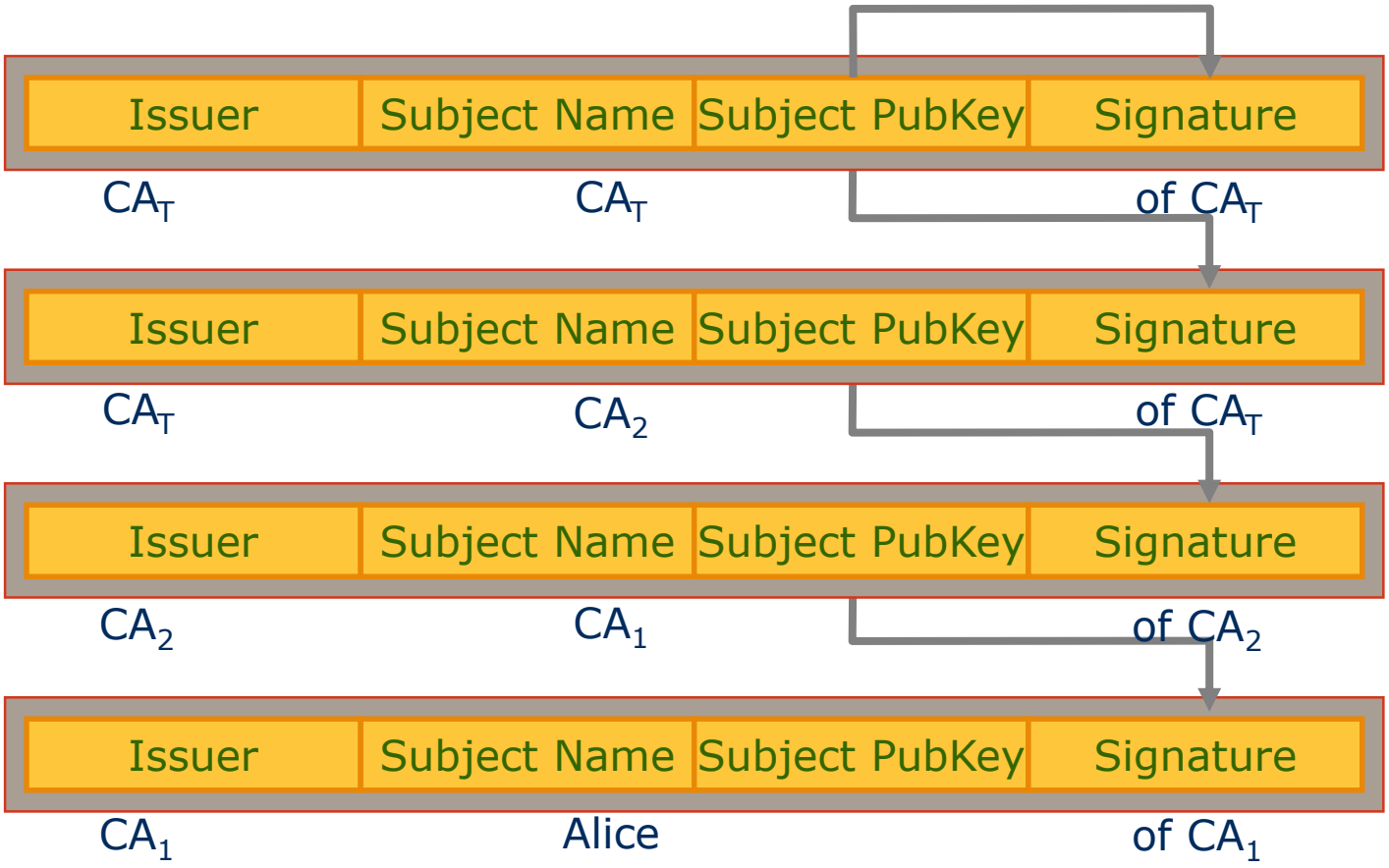


Easy, in hierarchical PKIs, If not: may need construct several paths

CA Hierarchy and Cross-Certification



Verify the Certificate: Path Validation



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

prove 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 from 1992

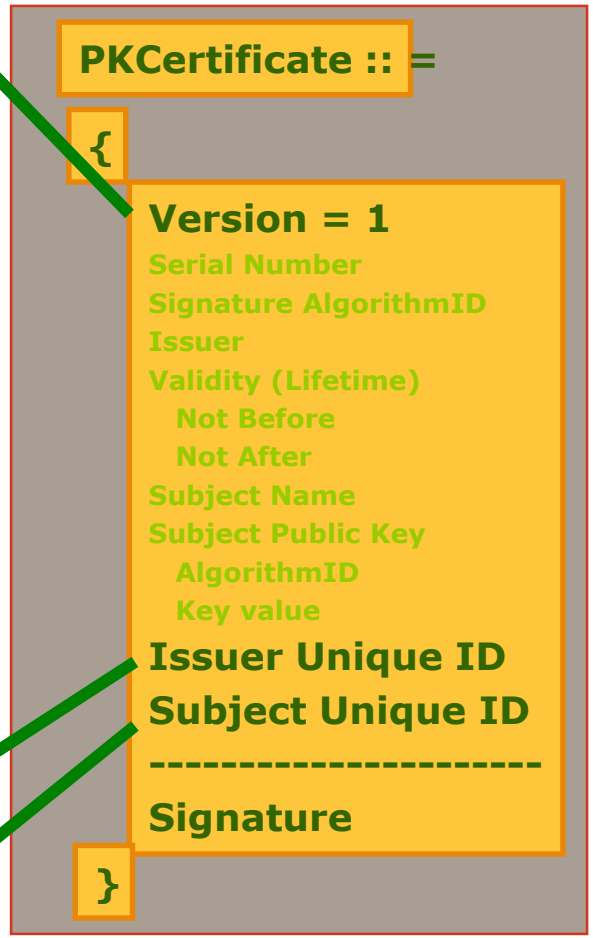
There may be several “Trustme-Cert Inc.” worldwide,
or several “Bob Hope” in our company

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.

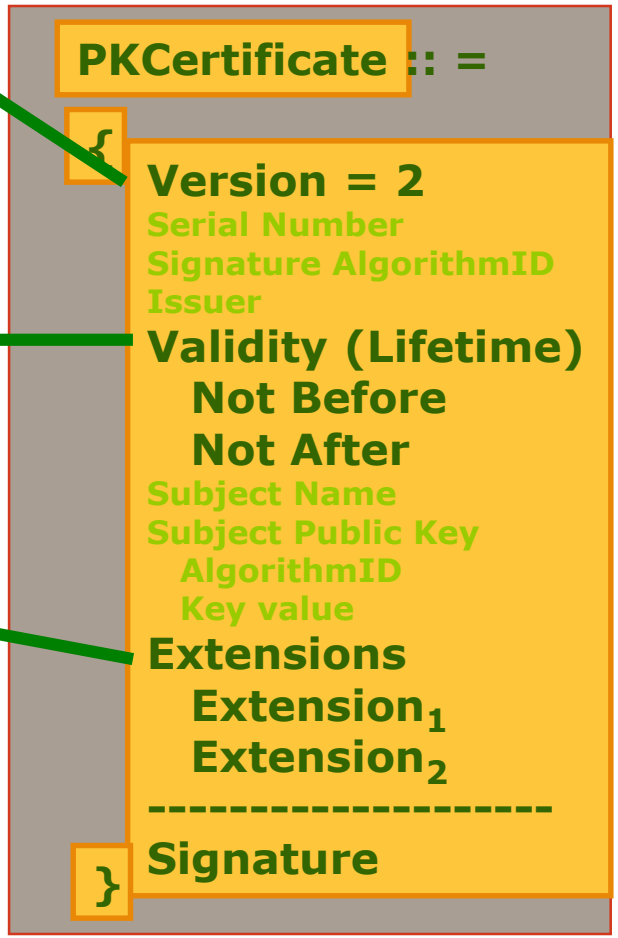


X.509 Public Key Cert V.3

Version 3 from 1998

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

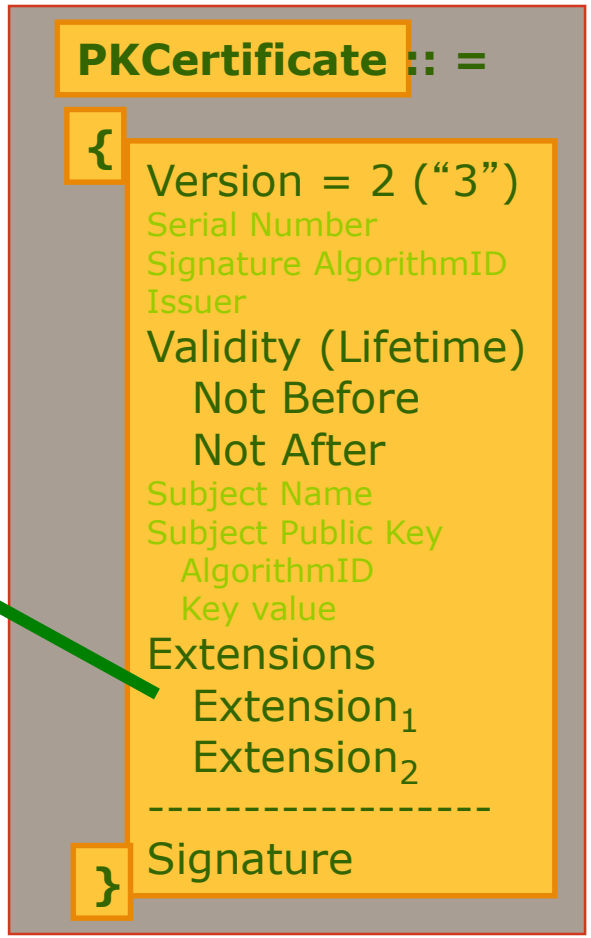
Standard extensions for: KeyID,
Key usage intention / restriction,
subject/issuer alternate names or attributes
(DNS name, email addr., URL, IP addr.)
policies
certification path
Private Extensions also possible



Key Usage

```
KeyUsage ::= BIT STRING {
  digitalSignature        (0),
  nonRepudiation          (1),
  keyEncipherment         (2),
  dataEncipherment        (3),
  keyAgreement            (4),
  keyCertSign             (5),
  cRLSign                 (6),
  encipherOnly            (7),
  decipherOnly            (8) }
```

X.509 Public Key Certificate V.3



Fields: Type
(critical | non critical)
value

Problems:

Issuer does not only check your identity, it also checks what you are allowed

Size of cert (say, in wireless applications)
Do not need all extensions always
More extensions => faster to revocate

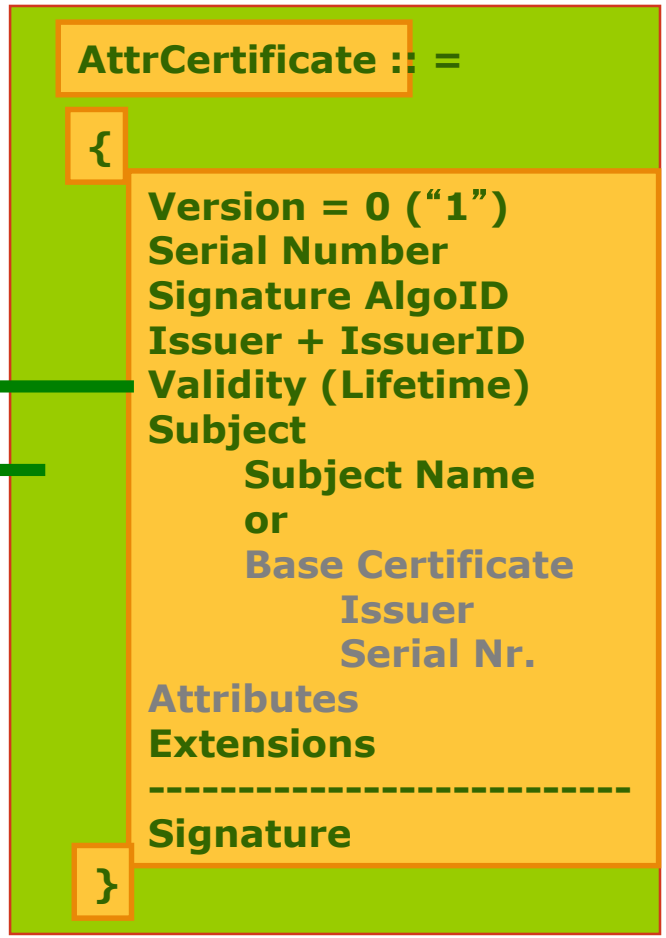
X.509 Attribute Cert V.1 (current)

Generalized Time: YYYYMMDD

No field for a public Key

May have different CA from PKCert
- different lifetime (shorter)
- authorization information
- role, etc.

Not (yet?) in wide use



Other Extensions

Basic constraints

- Identifies whether the certificate subject is a CA;
- how deep a certification path may exist through that CA.

Name constraints (only for CA certificates)

- Indicates name space within which all subject names in subsequent certificates in a certification path must be located.

Certificate management

Certificate management covers:

- the responsibilities and actions of the Certification Authority,
- the 'certification process',
- distribution and use of certificates,
- certification paths,
- certificate revocation.

Two parallel sets of standards cover interactions between users and a CA:

- IETF RFCs 2510/2511
- ISO/IEC 15945.

IETF leads the way - ISO/IEC has adopted proposals of RFCs.

The Certification Authority

The CA is responsible for:

- identifying entities before certificate generation
- ensuring the quality of its own key pair,
- keeping its private key secret.

The CA, before generating a certificate, checks that a user

- knows the corresponding private key to its claimed public key.

On keeping those commitments depends the notion of trust

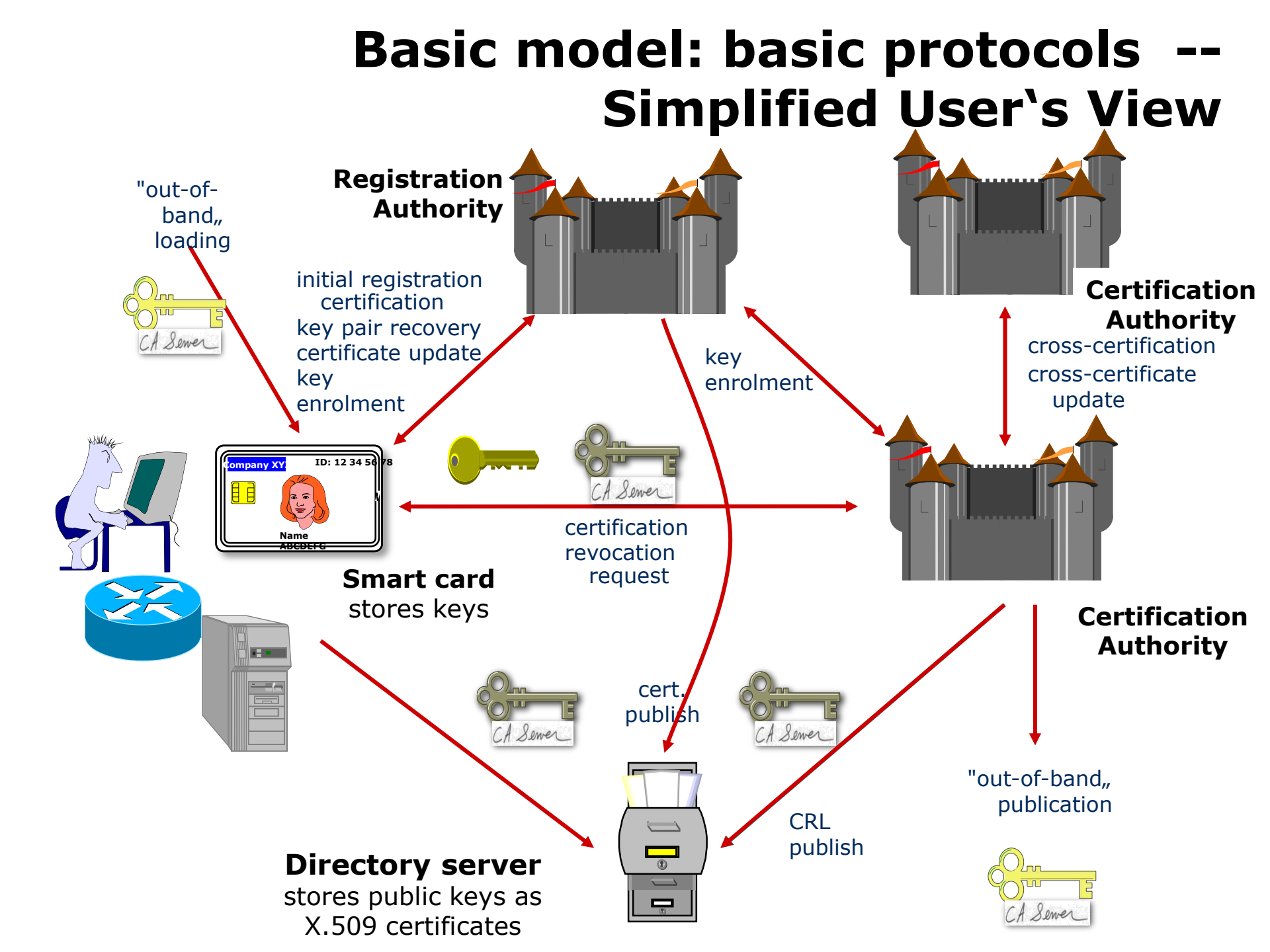
What is an End Entity?

X.509v3 certificates are used by protocols such as S/MIME, TLS and IKE, when authentication requires public keys.
(End Entity = Natural Person)

When two routers or security gateways or servers, etc. wish to communicate, they exchange certificates to prove their identity

- thus removing the need to manually exchange public keys or shared keys with each peer
- End Entity = Router, Printer, Gateway, Server, Device
- The certificate provides the equivalent of a digital ID card to each device.

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



Recall: Purpose of Digital Certificates

Scalability: get public keys only when really needed

Trusted validation of parties: by induction, I believe party is who he claims to be (erroneously: "trust is transitive")

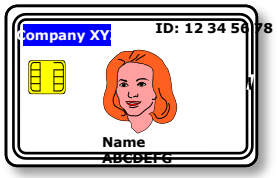
Transmission and storage of public keys can be insecure:
replace storing securely many keys with:

- store insecurely many certificates
- store securely the root certificate
- store securely the private key

Can provide permissions (Authorizations): later

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

Secured applications client e.g.
- Encrypted e-mail
- Encrypted web-access
- E-commerce using certificates
- VPN authentication using certificates



Need all:
Secure
networks,
services,
applications, and
devices

Secured application servers, e.g.
- Encrypted e-mail
- Encrypted web-access
- E-commerce using certificates
- VPN authentication using certificates

Reasons for Revocation

Compromise of subject's private key

Change in subject name

Change in Authorizations in Cert

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!

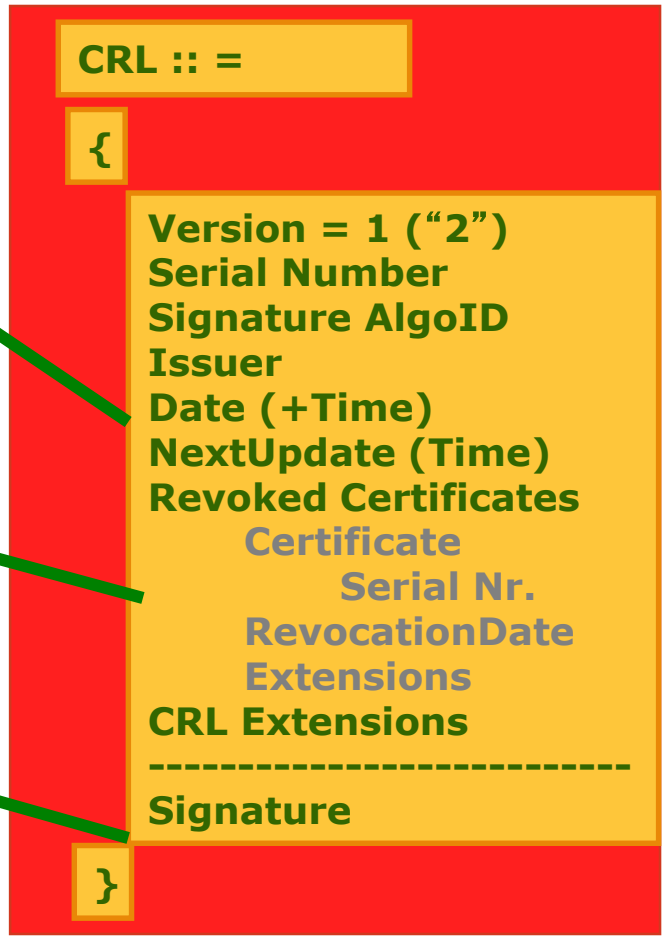
Certificate Revocation List, Version 2 (current)

- Time-stamped and frequently updated
- Rate may vary according to security of the transaction (say, 4 times a day)
- Must be scheduled regardless of change in status

- List of revoked certificate's serial numbers

- Signed by CA

- Stored on CA, directory service or OSCP (Online Cert Status Prot) server

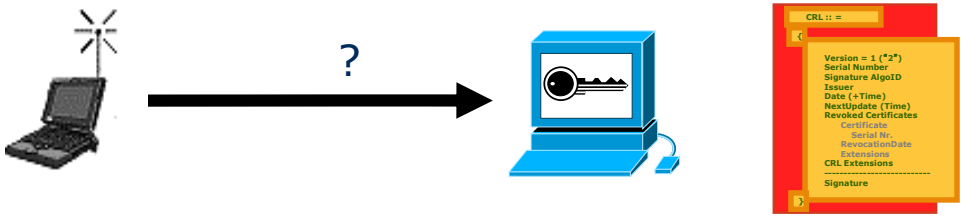


Distribution of CRLs

- Polling
  - Client polls according to advertised interval
  - CA or directory server can be polled
  - Black hole between revocation and next scheduled update

- Push
  - Broadcast
  - Reliable transport
  - Bandwidth Intensive
- Who needs them?

- On-line status checking
  - Client initiated
  - On-line query
  - Info available 24 x 7



Problems

PDAs, Cellular Phones, Laptops:

- Intermittent Network Access
- Low communication Bandwidth
- Low Computational Power

Ideally:

- Connect for a short time, download messages, SW, etc
- Validate Certificates
- Proceed with off-line operations
- But:
- Need Public Keys
- Path Discovery, Verification

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)

Online certificate status protocol

OCSP, RFC 2560, enables certificate status to be queried.

- The protocol specifies data exchanged between entity checking certificate status and the T3P providing that status.

OCSP may provide more timely revocation information than is possible with CRLs.

Entity issues status request to T3P and suspends acceptance of certificate until T3P gives response. (Some seconds, not real-time)

Client sends list of cert ids to a responder

Responder returns status for each:

- Good (simply means that responder has no record of the cert's revocation)
- Revoked
- Unknown (responder has no knowledge of the cert)

(Version 2 fixes the way cert ids are sent)

DPD: Delegated Path Discovery

For clients that don't want to do build a complete cert chain

- Memory or bandwidth constraints

Client request parameters:

- On the path construction
  - –Trust anchors
  - –Name constraints
  - –Name forms
- Validation Parameters
  - –Type of revocation status info (CRL or OCSP)

Responder builds a chain for the client:

- Client sends cert id
- Responder builds and returns chain - does not validate

Why DPD but no DPV? Client does not trust the responder

DPV: Delegated Path Validation

For clients that don't want to do validate a complete cert chain

- CPU, memory or bandwidth constraints
- Central policy management

Responder builds chain (but does not return) and gives status of cert sent as for OCSP

Client can specify trust points through which chain must be built

Client completely trusts the responder, but

- Can use signed response for non-revocation

Issue: trust delegation

SCVP

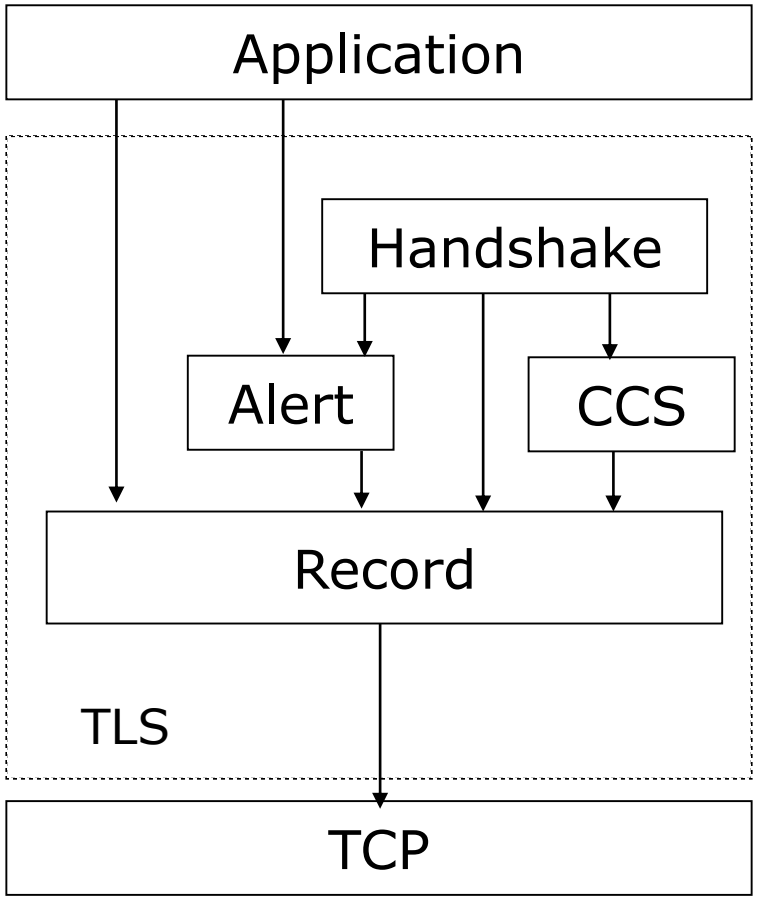
The amount that the responder does can be varied

- Client can offload all processing to the SCVP server
- Client can just use SCVP for chain building

Client sends up complete certs and what it expects:

- TypesOfCheck
  - –tells the server what types of checking the client expects the server to perform on the on the query item(s).
- WantBack
  - –tells the server what the client wants to know about the query item(s).

TLS Sub-Protocols



TLS Handshake Overview

Ciphers:

- RSA, DSS, and DH
- Elliptic curves, Kerberos, and Fortezza
- RC4, DES, 3DES, IDEA

RC4 is the default encryption algorithm

- Lots of old 40-bit software around
- Very weak.

HMAC MD5 or HMAC SHA-1 are the common MAC

The TLS Handshake Protocol

hello request $B \rightarrow A : ()$

sent by the server at any time, simple notification that the client should begin the negotiation process anew by sending a client hello

This message should not be included in the message hashes which are used in the finished messages and the certificate verify message.

client hello $A \rightarrow B : A; Na; Sid; Pa$

nonce Na, called client random,
session identifier Sid. The model makes no assumptions about the structure of agent names such as A and B.
Pa is A's set of preferences for encryption and compression;
both parties can detect if Pa has been altered during transmission (using the message hashes in finished messages and the certificate verify message).

The TLS Handshake Protocol

server hello B $\rightarrow$ A : Nb; Sid; Pb

nonce Nb (called server random).
Same session identifier
Pb his cryptographic chice, selected from Pa.

server certificate B $\rightarrow$ A : certificate(B;Kb)

The server's public key, Kb, in a cert signed by a trusted CA

Server key exchange message B $\rightarrow$ A : g^y

sent by the server only when the server certificate message does not contain enough data to allow the client to send a PMS. This message (may) contain the DH parameter of B "g^y", for calculating the PMS. (Another variant, not discussed here)

The TLS Handshake Protocol

certificate request B $\rightarrow$ A : certificate_types, certificate_authorities

server hello done B $\rightarrow$ A : ()

client certificate* A $\rightarrow$ B : certificate(A; Ka)

either client key exchange A $\rightarrow$ B : g^x

or encrypted premaster secret A $\rightarrow$ B : $\{PMS\}_{Kb}$

certificate verify* A $\rightarrow$ B : Sig_{Ka} (Hash {Nb; B; PMS})

Optional messages are starred (*)

In certificate verify, A authenticates herself to B by signing HAsh of some relevant messages to the current session.

Paulson: Important only to hash Nb, B and PMS.

The TLS Handshake Protocol

M = PRF(pre_master_secret, "master secret",
Client_random + Server_random)

Both parties compute the master-secret M from PMS, Na and Nb

finished A $\rightarrow$ B : PRF(M, "**client finished**" ,
hash(handshake_messages))
finished B $\rightarrow$ A : PRF(M, "**server finished**" ,
hash(handshake_messages))

The TLS Handshake Protocol

According to the TLS specification, client hello does not mention the client's name. But server needs to know where the request comes from; in practice gets this information from TCP. That it is not protected and could be altered by an intruder.

The master secret is hashed into a sequence of bytes, which are assigned to the MAC secrets, keys, and non-export IVs required by the current connection state:

- a client write MAC secret,
- a server write MAC secret,
- a client write key,
- a server write key,
- a client write IV, and
- a server write IV

The TLS Handshake Protocol

The symmetric client write key is intended for client encryption, while server write key is for server encryption; each party decrypts using the other's key.

Once a party has received the other's finished message and compared it with her own, she is assured that both sides agree on all critical parameters, including M and the preferences Pa and Pb.
Only now she may begin sending confidential data.

The TLS Handshake Protocol

The TLS specification erroneously states that she can send data immediately after sending her own finished message, before confirming these parameters;

- An attacker may have changed the preferences to request weak encryption.
- This is exactly the cipher-suite rollback attack, which the finished messages are intended to prevent.
- TLS corrects this error.

The TLS Handshake Protocol

- For session resumption, the hello messages are the same.
- After checking that the session identifier is recent enough, the parties exchange finished messages and start sending application data.
- Each party has to store the session parameters after a successful handshake and look them up when resuming a session.
- Session resumption does not involve any new message types.

Certificates, CAs, Browsers, and Servers

Many CAs' certificates pre-loaded with the browser:

- ATT, VeriSign, …
- Can be viewed in the browser, e.g.,
  - –Navigator 6: tasks, security and privacy, security manager

User surfs to https://www.mystockbroker.com/

Browser connects to port 443, sends nonce and gets back servers' cert & nonce

Certificates, CAs, Browsers, and Servers

Browser verifies cert; encrypts a pre-master secret with server's public key

- Process works if everyone is careful
  - –Some browsers come with 100+ CAs' certs; easy to mistake the name
  - –Some CAs may be unreliable
  - –Pre-master secret may be predictable
  - –Certificates expire and signatures may not check
  - –Virus may corrupt either party

Rest of the communications are protected

- Server asks for password, credit card #, tax ID #, etc.
  - –Sometimes servers get hacked and all customers' secrets get published
- And there's a lot of old "40-bit" software around

Personal Certificates and Client-Side Authentication

Clients (browsers) can have certificates too

- CA signs client's public key
- Obtained from well-known CAs:
  - –VeriSign, ATT, MCI, …
  - –Costs and policies vary
- Can be viewed in the browser, e.g.,
  - –options, security, personal certificates
- Two-way strong security
  - –No server access to user's secret
  - –Good security but not widely used
  - –Most secure web sites ask client for a simple password (encrypted)
  - –Worse, most secure Web sites only secure the "payment screen"

TLS Limitations

In all cases, have to trust other party's CA

- Usually not even aware of the choice
- How can you trust 115 CAs?

Password or credit card authentication allows unlimited guessing

Systems on both sides may get hacked or infiltrated with untrusted code

For efficiency reasons, most screens are not protected

Inherent back-end security target

- Many exposures, examples

No non-repudiation and huge dispute rates

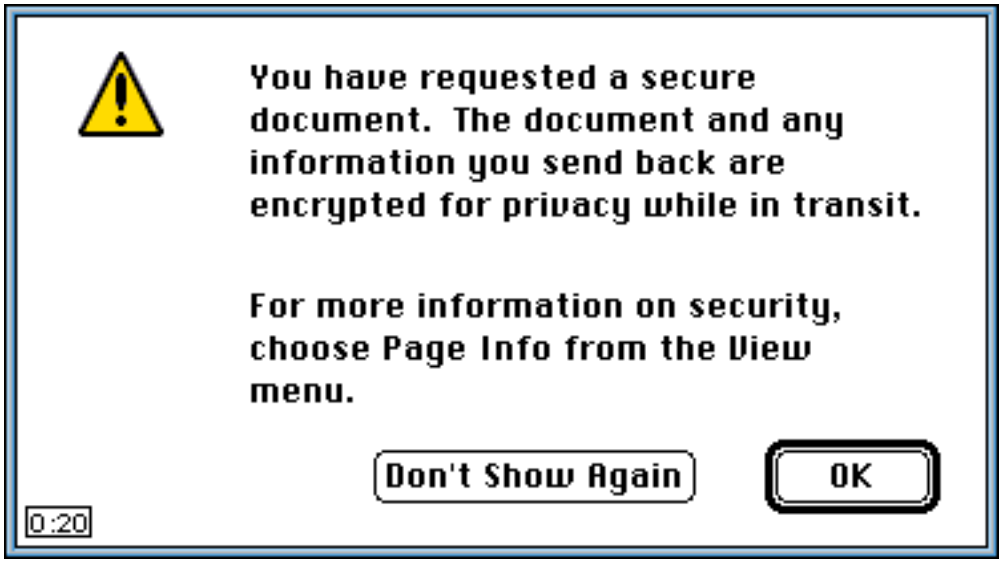
- Netscape introduced "form signing" on navigator 4.04
- Not supported by Explorer

No convenient "wallet" software

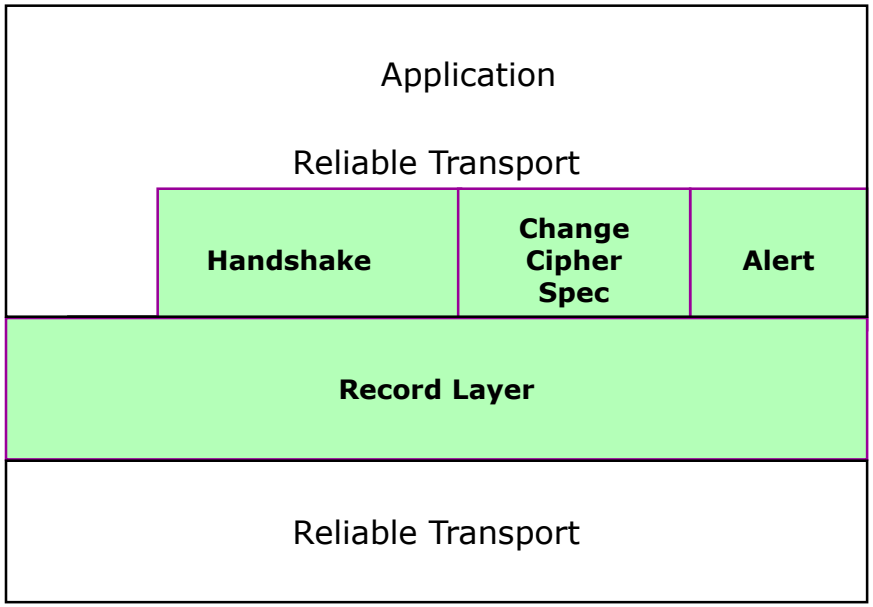
Using TLS

Warning screen from a secure page:

https://www.somewheresecure.com



TLS Architecture



- The Change Cipher Spec protocol consists of a single message that is sent by both the client and server to notify the receiving party that subsequent data will be protected under the newly negotiated Ciphersuite and keys.
- The Alert protocol specifies the TLS alert messages.

- The Record Layer provides the encapsulation of the upper layer data. The data is fragmented, optionally compressed, a MAC is appended, and data and MAC are encrypted. Each transport connection is assigned to a unique TLS session.

Contents

Internet Layers, Basics

Management, Implementation or Design Errors

Designing Correct Protocols: The Avispa contribution

IETF Groups and Activities

Sec Protocols: Kerberos, AAA, IPsec, IKE, IKEv2, Wlan, PKI, TLS

High-level Protocol Spec. Language (hlpsl): Syntax, Semantics, Goals, Examples

Outlook: MobileIP, HIP, Pana

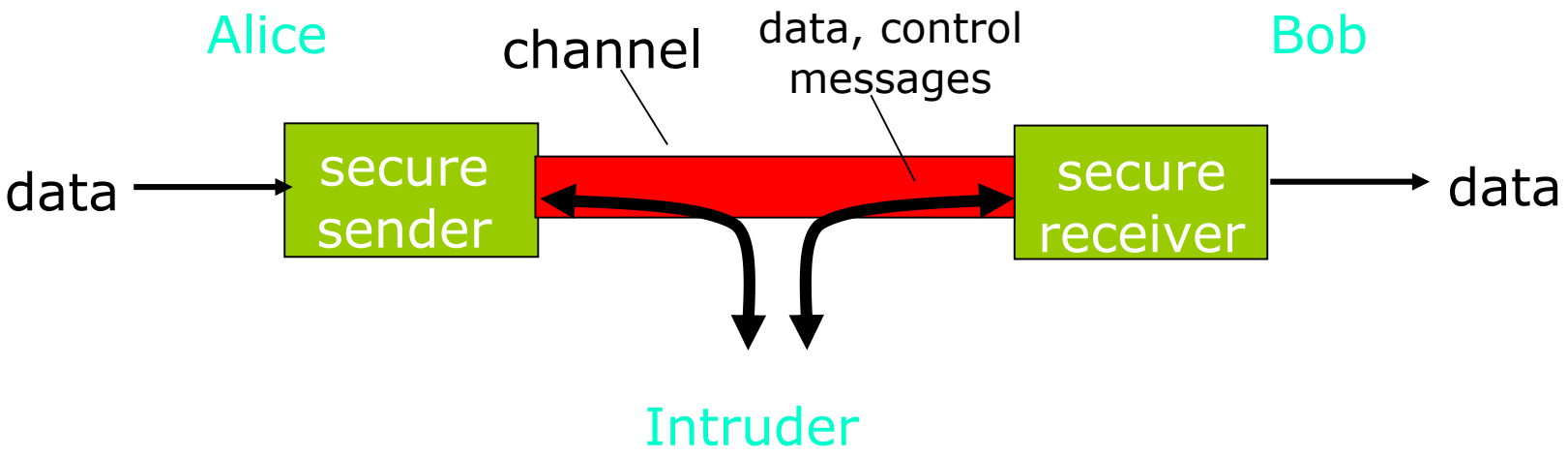
The basic Model: Alice, Bob, Intruder

Well-known in network security world

Alice + Bob want to communicate securely (privately, without modifications)

Alice and Bob are “roles”

Intruder may intercept, delete, add messages



Syntax: Roles as trace Predicates

Think of a module or "role" as a formula Alice(n;x,y)

Analogy: think of p(n,x,y) a FOL formula, like (x > y+n)

Alice(n;x,y) is not talking about single values of variables, (like p does), but about traces (sequences of values).

As you may write ξ *sat* p(n,x,y) (sat is usually written " |= "), for instance (2,10,4) *sat* (x > y+n)

You can also write τ *sat* Alice(n;x,y) for instance

((1,0,0), (1,1,0), (1,1,0), (1,1,8), (1,8,8), …)

$$\text{sat}\ (x=y=0 \ \wedge\ \Box\ (x' \neq x \Rightarrow y \leq x' \leq y' +1)\ )$$

Syntax: Variables, Predicates

Set of vars V={x,x1,x2,x3,y,y1, …} called *state variables*
(each of a determined type),
construct a copy of them called *primed variables*
{x',x1',y', …}

FOL predicates with free vars in V are called *state predicates*
and predicates with free vars in V united V' are called *transition predicates*

st_pred, tr_pred

(x=y=0) is a state predicate
($x' \neq x \Rightarrow y \leq x' \leq y' + 1$)) is a transition predicate

Syntax: Events, Stuttering

Transition predicates of the form

$(t(x) \neq t(x')) \wedge N(x,x')$ **where x is a tuple of variables**

are called *events*. Events exclude stuttering $(x=x')$

$x'=x+1$ **is not an event (*syntactical criteria*) but it excludes stuttering. It is equivalent to the event**

$x' \neq x \ \wedge \ x'=x+1$

Note that the disjunction of events is wlog also an event rewirting:

$(t(x) \neq t(x')) \wedge N(x,x')) \vee (s(x) \neq s(x')) \wedge M(x,x'))$

$((t,s)(x) \neq (t,s)(x')) \wedge (N \vee M)(x,x'))$

Syntax: TLA Normal Form

A TLA formula in normal form is:

$\exists$**... st_pred** $\wedge \Box$ **((event** $\Rightarrow$ **tr_pred)** $\wedge$ **(event** $\Rightarrow$ **tr_pred)** $\wedge$ **...)**

Our hlpsl is close to this TLA form

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

Conjuction is parallel composition of modules (roles)!

Two types of variables:

flexible variables (state of the system)

rigid variables (parameters, constants, may be instantiated at some point later)

TLA Example

V={x,y}

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

Then the following traces are in Tr(Prg):

(0,3), (0,4), (0,5), (0,6), (0,7), …

(0,3), (1,4), (2,5), (3,6), (4,7), …

(0,0), (1,1), (2,2), (3,3), (4,4), …

(0,0), (0,1), (1,2), (1,3), (2,4), …

If a TLA program talks about variable x only, it does not say anything about variable y.

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

(0,*), (1,*), (2,*), (3,*), (4,*), …

by:

take a prefix (including all)

introduce any number of x-stuttering steps,

repeat (x0,*) any number of times (even infinite)

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

hlpsl Example

Prg(x) = (x=0) ∧ □ (x'≠x ⇒ x'=x+1)

Using a signal "Trigg":

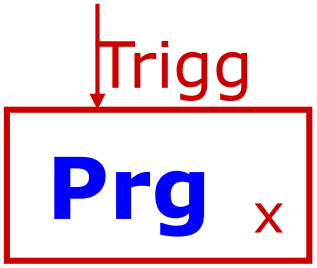
Role Prg(Trigg,x) :=

Owns x

Init x = 0

Trans

Trigg ⇒ x' = x +1



The var x is modified only by Prg, but it may seen outside.

TLA Example

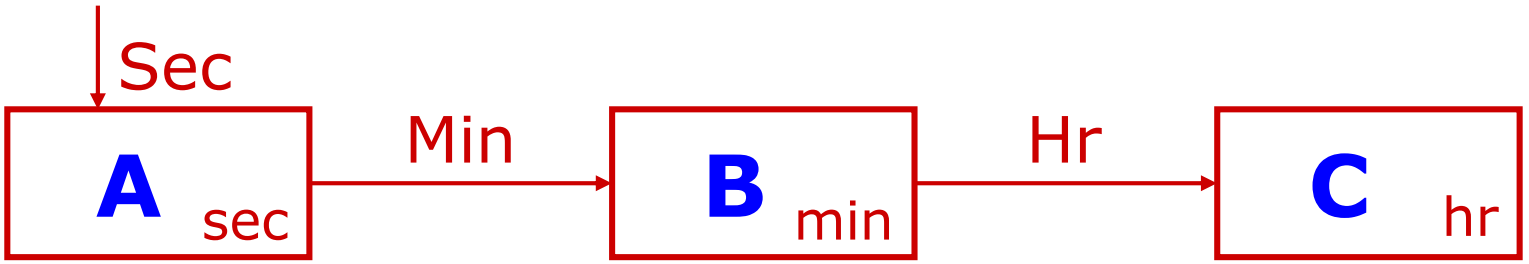
V={x,y}

Let Prg(x) = (x=0) $\wedge$ $\Box$ (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, modelling channels



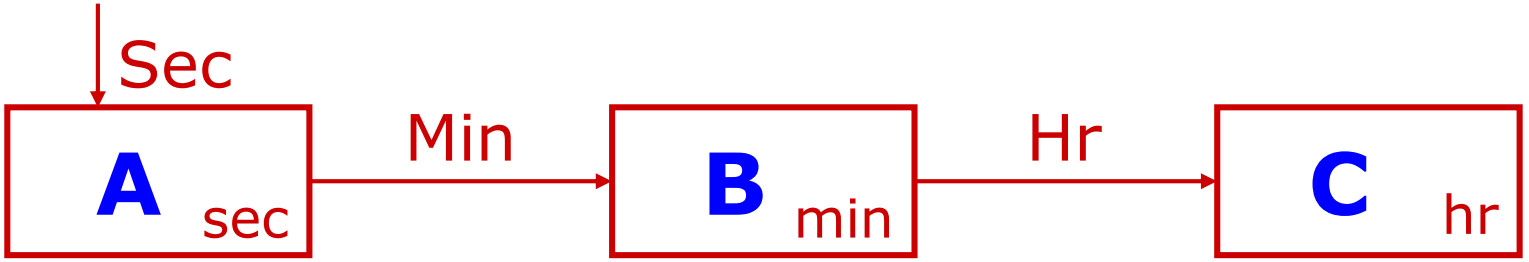
V={sec:{0,…59} ,min :{0,…59},hr :{0,…11} }

Sec := (sec'≠sec), etc. ***Events***

Clock: = A ∧ B ∧ C

```
A := (sec = 0) ∧ □ (∧ Sec ⇒ sec' = sec +1 (mod 60)
                   ∧ Sec ∧ sec' = 0 ⇒ Min)
B := (min = 0) ∧ □ (∧ Min ⇒ min' = min +1 (mod 60)
                   ∧ Min ∧ min' = 0 ⇒ Hr)
C := (hr = 0)  ∧ □ (  Hr  ⇒ hr'  = hr +1  (mod 12))
```

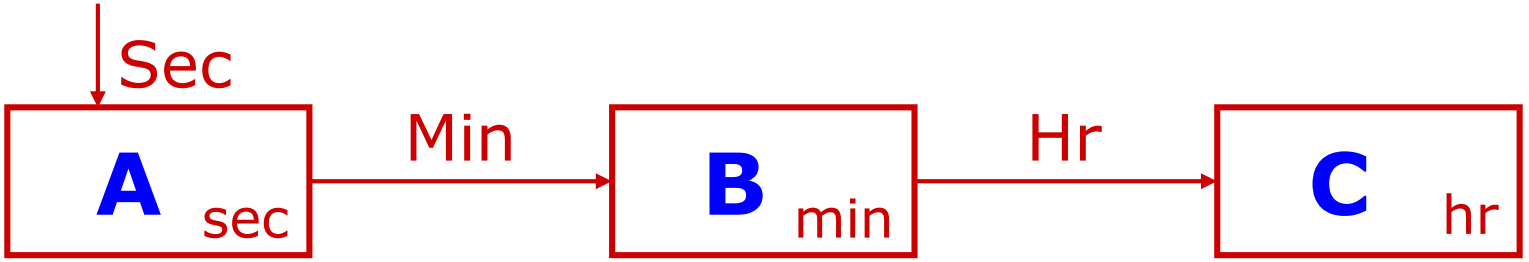
hlspl Example, the clock



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 )
```

Types of Channels

```
role A (p; v, channels: channel
  (dy|secure|ota|…) ) :=
  …
end role
```

Basic Roles: Semantics

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

```
Trigg(Basic_Role) := event1 ∨ event2 ∨ …          %% This is also an event!
Init(Basic_Role)  := Init
Accept(Basic_Role):= Accept(A) ∧ Accept(B) ∧ 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)) ] }
```

Basic Roles: Semantics

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

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

Basic Roles: Semantics

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

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

Note:
Step(A) ⇒ (event1 ∧ action1) ∨ (event2 ∧ action2) ∨ …

TLA(A)  = ∃ ∈ { Init ∧ □ [
            (event1 ⇒ action1) ∧ (event2 ⇒ action2) ∧ …
          ∧ (∧ _(θ∈Θ) θ'≠ θ ⇒ Mod(θ,A)) ] }
```

Semantic of Composed Roles: modular approach

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

Parallel, Sequential (+taking ownership, hiding)

IF-Programs $\leftarrow$ hlpsl-Programs $\rightarrow$ TLA-Formulas

IF(A) , IF (B) $\leftarrow$ A , B $\rightarrow$ TLA(A) , TLA(B)

$\downarrow$ $\downarrow$ $\downarrow$

IF(A) $\oplus$ IF (B) $\leftarrow$ A $\otimes$ B $\rightarrow$ TLA(A) $\bullet$ TLA(B)

For Parallel composition:

TLA(A) $\bullet$ TLA(B) = TLA(A) $\wedge$ TLA(B) $\wedge$ extra_glue (for ownnership)

Semantic of Composed Roles: flattening approach

A ⊗ B = Composition(A,B):

Parallel, Sequential (+taking ownership, hiding)

flatten: hlpsl-Programs → hlpsl-Programs

For basic roles: flatten(A) = A

For composed roles: flatten(A ⊗ B) = arrange(flatten(A),flatten(B))

Composed Roles: Parallel

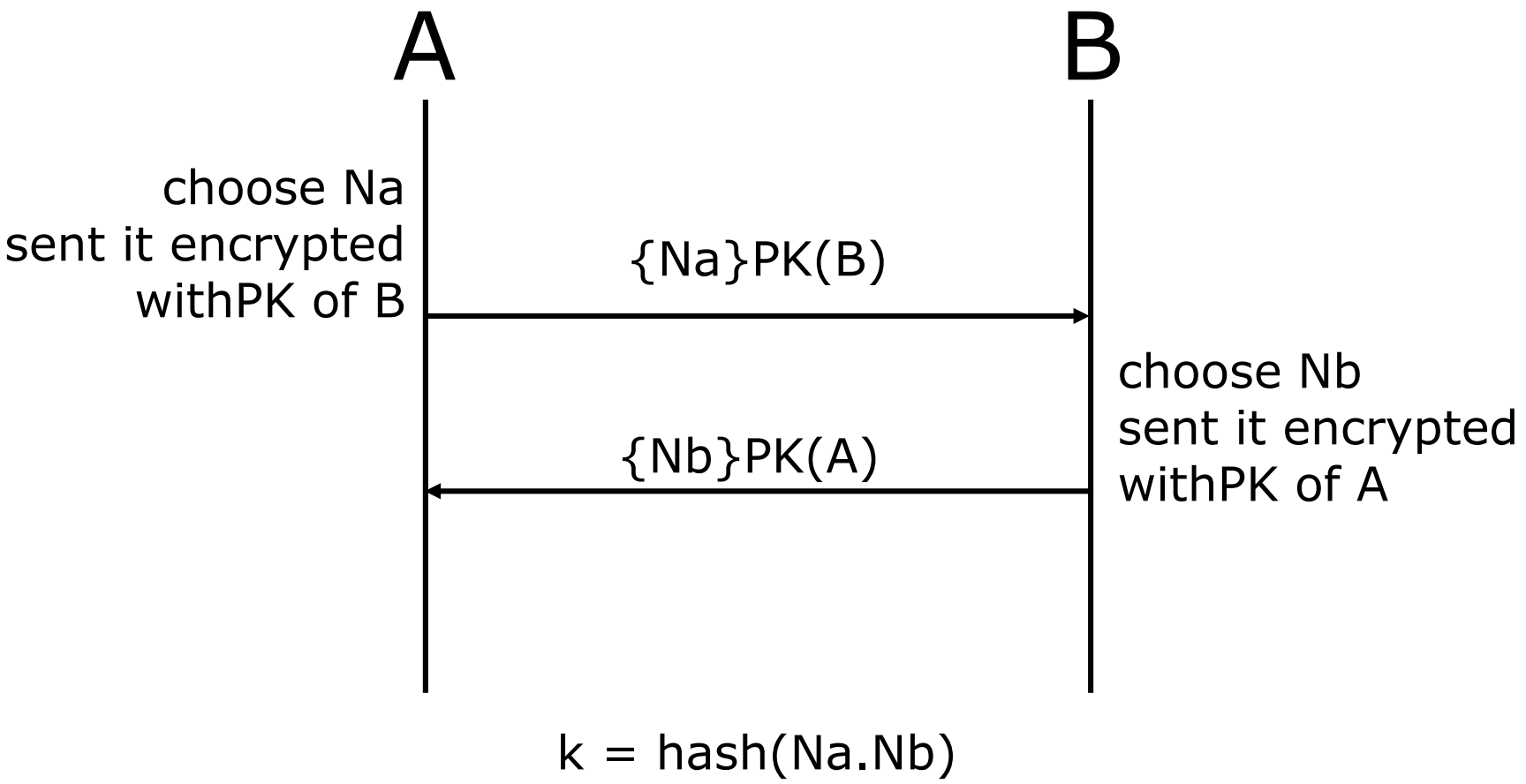
```
role Par_Role ( parameters; variables, channels) :=   % Parallel Composition of A and
   B
  owns {θ:Θ}
  local {ε}
  init Init
  accepts Accept
    A ∧ B
end role

Trigg(Par_Role)   := Trigg(A)   ∨  Trigg(B)
Init(Par_Role)    := Init(A)    ∧  Init(B)    ∧ Init
Accept(Par_Role)  := Accept(A)  ∧  Accept(B)  ∧ Accept
Mod(x,Par_Role)   := Mod(x,A)   ∨  Mod(x,B)
TLA(Par_Role)     := ∃ ∈ { Init ∧ A ∧ B
                           ∧ □ [ (∧ _(θ∈Θ) θ'≠ θ ⇒ Mod(θ, Par_Role)) ] }
```

Composed Roles: Seq

```
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
```

$$
\begin{aligned}
&\text{Trigg(Seq\_Role)} := (\mathit{flag} = 0 \wedge \text{Trigg(A)}) \vee (\mathit{flag} = 1 \wedge \text{Trigg(B)})\\
&\text{Init(Seq\_Role)} := \mathit{flag} = 0 \wedge \text{Init(A)} \wedge \mathit{Init}\\
&\text{Accept(Seq\_Role)} := \text{Accept(B)} \wedge \mathit{Accept}\\
&\text{Mod(x,Seq\_Role)} := (\mathit{flag} = 0 \wedge \text{Mod(x,A)}) \vee (\mathit{flag} = 1 \wedge \text{Mod(x,B)})\\
&\text{TLA(Seq\_Role)} := \exists\, \epsilon, \mathit{flag}\ \{\text{Init(Seq\_Role)}\\
&\qquad \wedge \Box\, [(\text{Trigg(A)} \Rightarrow \mathit{flag}{=}0) \wedge (\text{Trigg(B)} \Rightarrow \mathit{flag}{=}1)\\
&\qquad\quad (\text{flag}' \neq \text{flag} \Rightarrow \text{flag}' = 1\\
&\qquad\qquad \wedge \text{Accept\_A'}\\
&\qquad\qquad \wedge \text{Init\_B'})
\end{aligned}
$$

Example: Share protocol



hlpsl: Share: basic roles

```
role Initiator(A,B, PK: agent -> public_key; SND, RCV: channel (dy)) :=
  exists St:{0,1,2}, Na:text (fresh), Nb:text
  init St=0
  transition
    St=0 ∧ RCV(start) ⇒ St'=1 ∧ SND({Na'}PK(B))
    St=1 ∧ RCV({Nb'}PK(A)) ⇒ St'=2 ∧ secret(hash(Na,Nb'))
 goal
       secrecy %  of hash(Na,Nb)
 end goal
end role

role Responder(A,B, PK: agent -> public_key; SND, RCV: channel (dy)) :=
  exists St:{0,1,2}, Na:text, Nb:text (fresh)
  init St=0
  transition
    St=0 ∧ RCV({Na'}PK(B)) ⇒ St'=1 ∧ SND({Nb'}PK(A)) ∧ secret(hash(Na',Nb'))
goal secrecy end goal
end role
```

Explicit secrecy goals

Needham-Schroeder Public Key Protocol (NSPK): Alice

```
role Alice (A,B: agent,
            Ka, Kb: public_key,
            SND,RCV: channel (dy)) played_by A def=
  exists State : nat, Na : text (fresh), Nb: text
  init State=0
  knowledge(A) = { inv(Ka) }
  transition
    step1. State=0 /\  RCV(start)  =|> State'=1 /\ SND({Na'.A}Kb)
    step2. State=1 /\ RCV({Na.Nb'}Ka) =|> State'=2 /\ SND({Nb'}Kb)
end role
```

played_by

knowledge

start message to signal an initiator that he should start

step1 and step2 are merely labels

NSPK: Bob

```
role Bob(A: agent,
         Ka, Kb: public_key,
         SND,RCV: channel (dy)) played_by B def=
  exists State : nat, Na: text, Nb: text (fresh)
  init State=0
  knowledge(B) = { inv(Kb) }
  transition
    step1. State=0 /\ RCV({Na'.A}Kb)
       =|> State'=1 /\ SND({Na.Nb'}Ka)
    step2. State=1 /\ RCV({Nb}Ka)
       =|> State'=2
end role
```

NSPK: Composing the roles

```
role NSPK(S,R: agent -> channel (dy),
          Instances: (agent,agent, public_key,public_key) set) def=
  exists A, B: agent, Ka, Kb: public_key
  composition
     /\_{in((A,B,Ka,Kb),Instances)}
         Alice(A,B,Ka,Kb,S(A),R(A))
      /\ Bob(A,B,Ka,Kb,S(B),R(B))
end role
```

NSPK: Sessions and Goals

```
role Environment() def=
    composition
    NSPK([(a,s_a),(b,s_b)],                    % S
         [(a,r_a),(b,r_b)],                    % R
         [(a,b,ka,kb),(a,i,ka,ki)])            % Instances
end role

goal
        Alice weakly authenticates Bob on Nb
        Bob weakly authenticates Alice on Na
        secrecy of Na, Nb
end goal
```

Share: goals

1. **A->B: {NA}k(B)**
2. **B->A: {NB}k(A)**

Agents will use h(NA,NB) as shared key.

The authentication goals

A authenticates B on NB (or on (NA,NB))

and

B authenticates A on NA (or on (NA,NB))

are trivially violated:

1. **i(a) -> B: {X}k(B)**
2. **B -> i(a): {nb}k(A)**

Now B believes (X,nb) is the shared key between a and him, while a is not even present.

Not a "real" attack:

- intruder does not find out the nonce nb
- and can never use the shared key

Share

Also execution of B is stuck: nobody except B knows the shared key, nobody can send messages with this key.

Same problem with the first-phase of IKE: intruder can play a MiM, but can not find out the key and the protocol execution is stuck, no second-phase protocol can be executed.

Protocol does not satisfy the authenticate goal:

when B receives the first message of the protocol, he can not be sure that it actually comes from A.

A must prove her presence by sending a message encrypted with the key h(NA,NB).

Share

See this part of protocol as a challenge, add the response:

1. A->B: {NA}k(B)
2. B->A: {NB}k(A)
3. A->B: {0,..}h(NA,NB)
4. B->A: {1,..}h(NA,NB)

"0", "1" inserted to distinguish the two messages
then intruder can not simply reflect this message 3 from A back to A

New goals:

A authenticates B on NA,NB,MA

B authenticates A on NA,NB,MB

secrecy of NA,NB,MA,MB

“Incomplete Protocols”

"the key-exchange phase of the protocol does not YET provide the authenticate itself, but rather after the first use of the key the agents authenticate each other."

We found no further attacks on SHARE.

We have taken SHARE (with the additional messages 3 and 4) as an example and could verify (within seconds!) secrecy and weak authentication (in a typed model for an unbounded number of sessions and agents).

List of Protocols

SHARE	
UMTS-AKA	3GPP
ISO Pub Key wout	T3 Party
ChapV2	AAA
EKE	cat,sasl,NWWG
SRP	cat,sasl,NWWG
EKE2	
SPEKE	
ASW	
AAA-MobileIP	mobileip
IKEv2 main mode	ipsec
Two-Party RSA Sig Schemes	
TLS	
TWSS	Liberty

Kerberos	krb-wg
HIP	HIP
Mut Auth for low-power dev	
TESLA	MSEC
sucv	mobileip
BU in IPv6	mobileip
TLS	tls
SSH	secsh
Key-Priv in Pub-Key Encr	PKIX
Payment in UMTS	3GPP
CMS Symmetric Key Mang	smime
SET	
FairZG	

Contents

Internet Layers, Basics

Management, Implementation or Design Errors

Designing Correct Protocols: The Avispa contribution

IETF Groups and Activities

Sec Protocols: Kerberos, AAA,
IPsec, IKE, IKEv2, Wlan,
PKI, TLS

High-level Protocol Spec. Language (hlpsl):
Syntax, Semantics, Goals, Examples

Outlook: MobileIP, HIP, Pana

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 subnetwork moves (not covered in MobileIP).

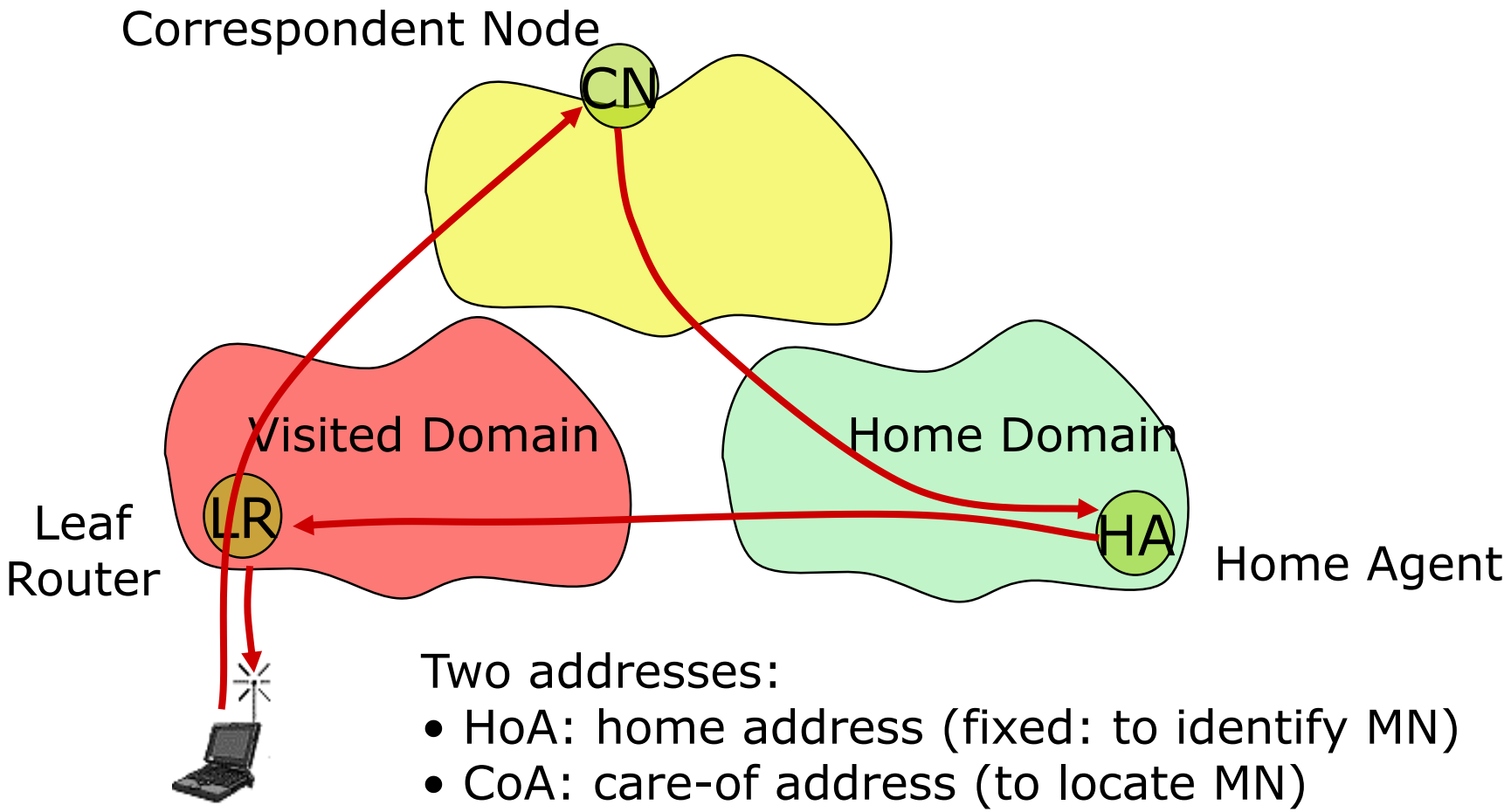
Mobility protocols

- 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

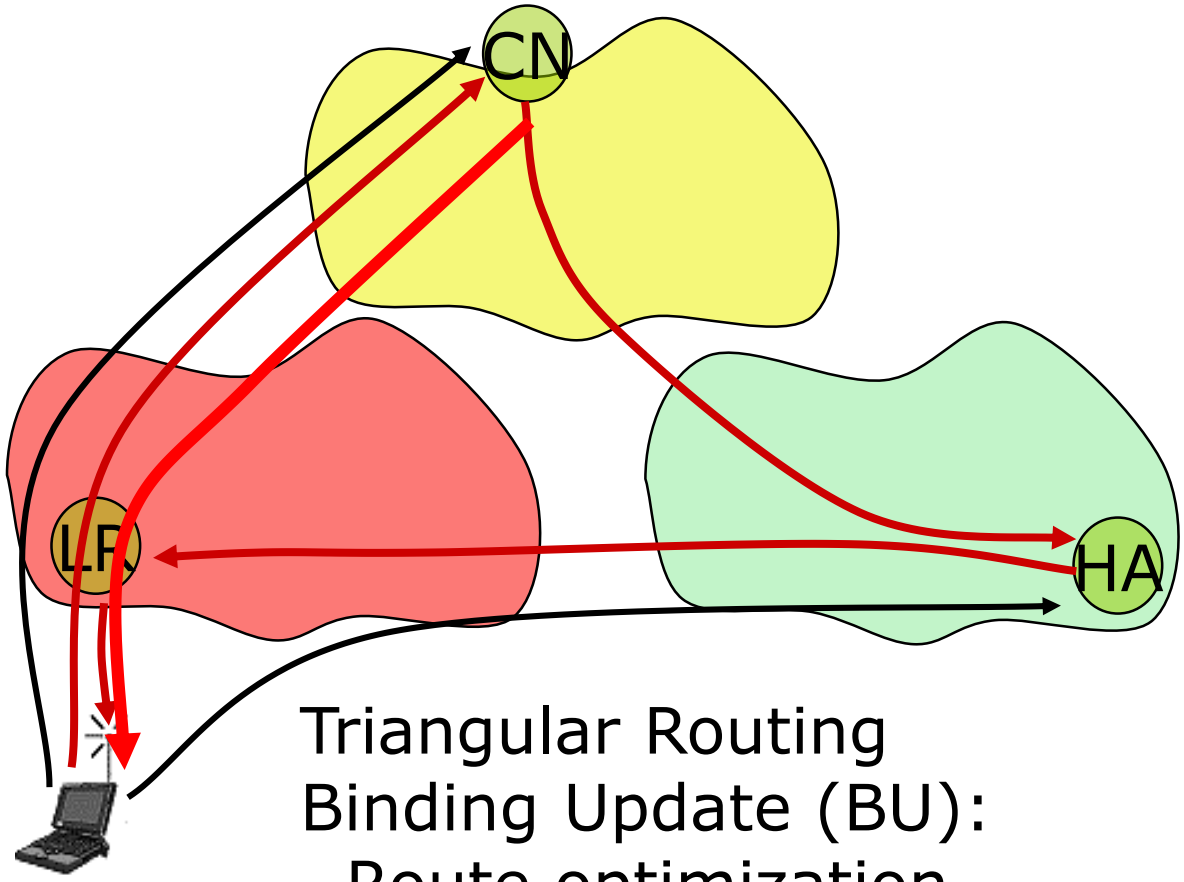


Two addresses:

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

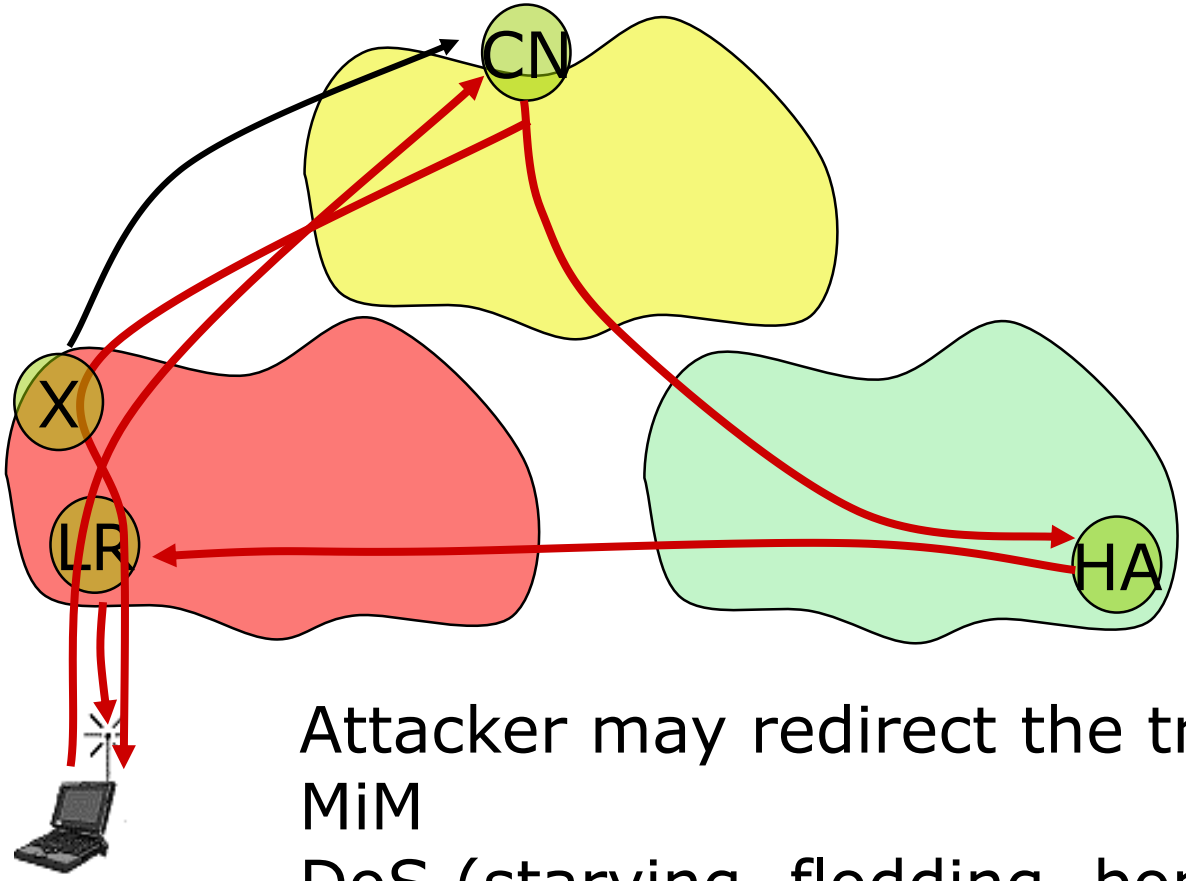
How are such „Bindings" created / modified?

Mobility Management



Triangular Routing
Binding Update (BU):
Route optimization

Security Problems



Attacker may redirect the traffic:
MiM
DoS (starving, flodding, boming)

IP V6

Adress size increased from 32 to 128 bits.

Auto-configuration to generate locally CoA:

- 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).

MN can send 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 Update

- 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 cooperate to send BUs to CNs, with which they often have no preexisting 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 of secrecy and integrity
  - –denial-of service (MN unable to communicate).

Attacker sending bogus BUs may use own address as CoA, impersonating MN.

- highjack connections between MN and its CNs or
- open new ones.

Or 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.

the attacker can make any node believe that any other node, even a non-MN one, is MN and has moved to the false CoA.

- Side effect of making mobility transparent.

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.

- A single PKI must cover the entire Internet.

Cryptographically Generated Addresses (CGAs)

- Take last 64 bits of the IP address (interface identifier) as one-way hash of a PK. MN signs its location information with the corresponding private key and sends the PK along with the data.
- The recipient hashes the public key and compares HAsh to the address before verifying the signature on the location data.
- Used without any trusted third parties, PKI, or other global infrastructure.
- Weakness: at most 64 bits of the IP address can be used for Hash. Perhaps brute force attack will become possible during the lifetime of MobIPv6.

CGAs

- Strong signature key generation expensive, but weak signature keys may be used.
- Advances in storage technology may enable the attacker to create a large enough database for finding matching keys at high probability.
- CGA do not stop the attacker from inventing new false addresses with an arbitrary routing prefix. The attacker can generate a public key and a matching IP address in any network. Thus CGA addresses prevent some packet-flooding attacks against individual addresses but not against entire networks.
- Public-key protocols (including CGA) are computationally intensive and expose the participants to DoS.

Routing-based authentication

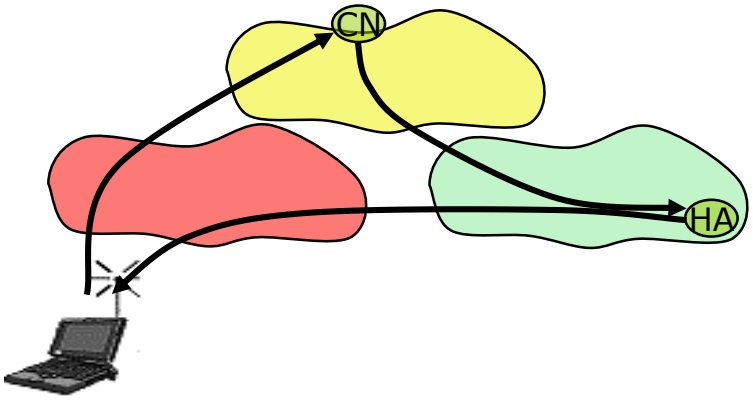
Idea: send 1st message through a relatively safe route (hope it is not intercepted).

- Here: Route between CN and HA.
- CN can send a secret key to HA (plaintext).

HA forwards key to MN (secure tunnel),

MN uses key for authenticating a BU to CN:

- MN → CN: BU with MAC (computed with secret key).



Routing-based authentication

Reasonable: very few Internet nodes can listen to or modify packets on the right routers to mount an attack against a given connection.

- At most 10-20 routers see the secret keys for a specific connection

Not secure in the classical sense

- But much better than unauthenticated situation.

HA and CN are typically located on the wired network and communication is relatively secure compared to the packets to and from a wireless MN.

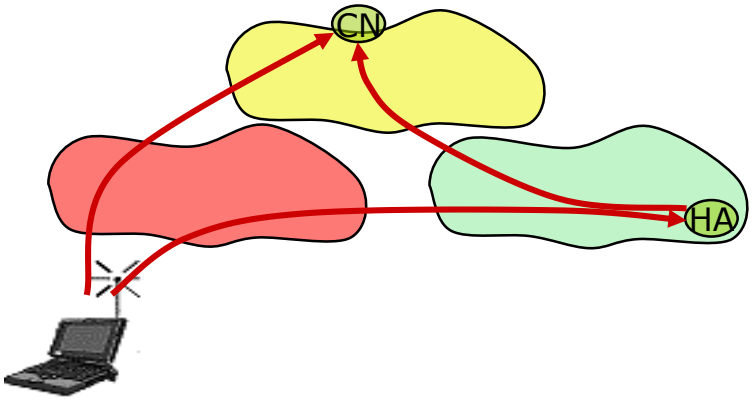
- An attacker between MN at home and a CN can mount equally damaging attacks
- Recall that the goal is to address the *additional* threats created by mobility

Weaker than CGA

Sending 2 Pieces of Authentication Data

Other proposals for BU authentication:

Send 2 pieces of authentication data between CN and MN via 2 independent routes and hoping that most attackers are unable to capture both of them.



Leap-of-faith authentication

MN sends a session key insecurely to CN at the beginning of their correspondence and the key is used to authenticate subsequent BUs, no safe route.

- Attacker can send false key before the MN sends the key
- Need a recovery mechanism for situations where MN or CN loses its state; attacker can exploit this mechanism
- Attacker can trigger the BU protocol at any time by sending to MN's home address a spoofed packet that appears to come from CN

Another DoS

Authentication does not prevent the attacker from lying about its own location.

Attacker acts as MN, sends false location data to CNs and get them to send traffic to an arbitrary IP address.

It first subscribes to a data stream (e.g. a video stream from a public web site) and then redirects this to the target address.

Bomb any Internet node or network with excessive amounts of data.

- Attack an entire network by redirecting data to a nonexistent address and congesting the link toward the network.

The attacker may even be able to spoof the (say TCP) acknowledgements

Another DoS (cont)

The attacker performs the TCP handshake itself and thus knows the initial sequence numbers. After redirecting the data to the target, it suffices to send one spoofed ack per TCP window to CN.

TCP provides some protection against this attack:

- If the target address belongs to a real node, it will respond with TCP Reset, which prompts CN to close the connection.
- If target is a non-existent address, the target network may send ICMP Destination Unreachable messages. Not all networks send this latter kind of error messages.

The attack is not specific to MIPv6:

- Dynamic updates are made to Secure DNS, there is no requirement or mechanism for verifying that the registered IP addresses are true.
- ICMP Redirect messages enable a similar attack on the scale of a local network. We expect there to be other protocols with the same type of vulnerability.

Variation: Bombing HoA

Im MIPv6 the MN has a default address, to which data will be sent when its current location is unknown.

Attacker claims to have a HoA in the target network. It starts downloading a data stream and either sends a request to delete the binding cache entry or allows it to expire. This redirects the data stream to the false HoA .

CGA prevents bombing individual addresses but not whole networks

- generate a new address with its routing prefix.

Bombing HoA

The target itself cannot do anything to prevent the attack.

- it does not help if the target stops sending or accepting BUs.

The attacker needs to find a CN that is willing to send data streams to unauthenticated recipients.

- Many popular web sites provide such streams.

A firewall on the border of the target network may be able to filter out packets to nonexistent addresses.

- However, IPv6 addressing privacy features can make such filtering difficult.

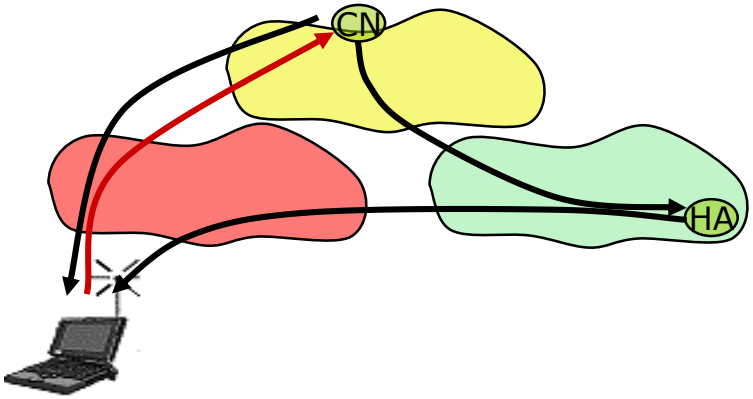
Limiting bombing attacks: Return Routability

Test the return routability (RR) of MN's new address

- CN sends a packet with a secret value to the new location and accepts the BU only if MN is able to return the value (or hash)
- Thus MN can receive packets at the claimed address
- Number of potential attackers is strongly reduced

Figure shows how a BU is authenticated using two secrets, which CN sends to MN's home and CoAs. The secret sent directly to the CoA forms the RR test.

The RR test can be seen as a variation of the cookie exchange, used in TCP, Photuris, and IKE



RR

Expiry of a binding cache

- Deleting the cache entry means that MN's new address defaults to the HoA , but since MN may have become unreachable, it is not always possible to test RR for the new address.

One solution:

- mark the cache entry as invalid and
- stop sending data to MN until the RR test succeeds
  - –Then some cache entries are never deleted.

Alternative: additional RR test for the HoA during every BU

- Invariant: a successful RR test for the HoA has been performed recently
- When the cache entry needs to be deleted, it can be deleted immediately
  - –BU cancellation, expiring cache entry, or failing BU authentication
- This limits bombing-attack targets to networks where attacker has recently visited.

RR

- In routing-based authentication (CN sends a plaintext key to MN via its HoA), the same secret key can also serve as the RR test for the HoA .
- Thus CN tests return routability of both HoA and CoA.
- RR is complementary to CGA-based BU authentication, which does not prevent bombing of the home network.

Transport layer: Flow Control

- When sending a data flow into a new route, CN could first verify that this route accepts the data
- Send first a single packet and increase the transmission rate gradually.
- TCP: reset the TCP window size to one packet when MN moves. This would, in effect, test return routability of the new route before sending large amounts of data into it.
- Adding a secure RR test to all transport protocols and changing existing implementations is not be possible in practice.
- Some transport-layer protocols either do not practice TCP compatible congestion control or allow spoofing of acknowledgments.
- Therefore: return routability test in the IP layer.

DoS Attacks against unnecessary BU Authentication

- When a MIPv6 MN receives an IP packet from a new CN via its home network, it may automatically send a BU to CN.
- The attacker can exploit this by sending MN spoofed IP packets (e.g. ping or TCP SYN packets) that appear to come from different CN addresses.
- The attacker will automatically start the BU protocol with all these CNs.
- If CN addresses are real addresses of existing IP nodes, most instances of the BU protocol will complete successfully. The entries created into the binding caches are useless.
- This way, the attacker can induce MN or CN to execute the BU protocol unnecessarily, which will drain host's resources.
- A strong cryptographic authentication protocol is more vulnerable than a weak or unauthenticated.

Reflection and Amplification

Reflection: Attacker sends data to other nodes and tricks them into sending the same number, or more (amplification), packets to the target.

- Possible even when ingress filtering prevents source address spoofing.

The location management protocols could also be used for reflection. For example, CN in Figure responds to the initial packet by sending two packets to MN (one to the HoA and one to the new address).

- If public-key authentication is used, the packets sent by CN may be significantly larger than the one that triggers them.

Preventing Resource Exhaustion: Delaying Commitment

Idea: delay committing one's resources until other party has shown its honesty

Require first a weaker authentication, such as a RR, before expensive computation.

Making the protocol parties stateless:

- usually only the responder can be stateless,
- not clear which party initiates the BU process and which one responds.
  - –MN normally initiates the authentication,
  - –this may be triggered by a packet belonging to another protocol that arrived from CN via HA.
  - –Moreover, if a packet sent by CN triggers a BU, CN's IP layer does not know that this was the case because the IP layer is stateless and does not maintain a history of sent packets.
- Make CN stateless until the BU has been authenticated.

One way in which CN can remain stateless is to derive a values Ka using a one-way function from a secret value N known only by CN and a value dependent on the MN:

- CN uses the same value of N for all MNs.
- It can discard Ka because it can recompute the values after receiving the final message.

CN generates a new secret Ni periodically.

Cryptographic puzzles

Used to protect against resource-exhaustion attacks.

A server requires its clients to solve a puzzle, e.g. bruteforce search for some input bits of a one-way function, before committing its own resources to the protocol.

The server can adjust the difficulty of the puzzles according to its 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 respondent)
- MNs often have limited processor and battery capacity while an attacker pretending to be a MN is likely to have much more computational resources

The puzzle protocols work well only when all clients have approximately equal processing power

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.

A node that exceeds the limit should stop sending or accepting BUs and allow binding cache entries to expire.

Although communication can continue via MN's home network, it is suboptimal.

Node should try to resume normal operation when attack may be over.

Ingress filtering at the attacker's local network mitigates the resource exhaustion attacks by making it easier to trace the attacker and to filter out the unwanted packets.

Favoring Regular Customers

CN's local security policy: allow BUs with some

- high-priority MNs or
- those with which it has a long-term relationship or
- recent meaningful communication.

The decision may violate the layering principle: a Web server could accept BUs from its clients after it has successfully executed the TCP handshake.

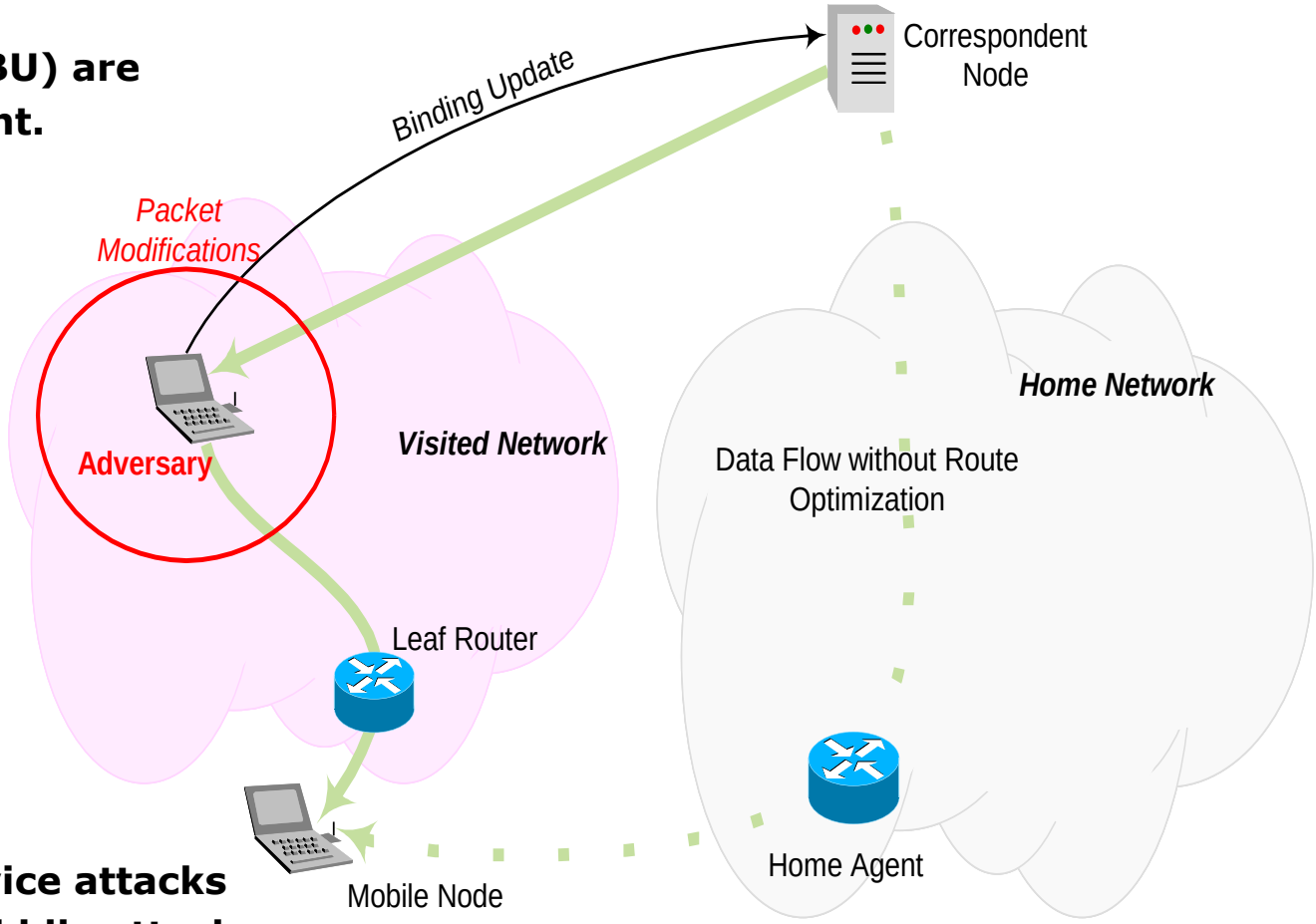
How does MN obtain its CoA?

- IPv6 stateless address autoconfiguration used to obtain an IPv6 address for MN.
- Host combines tentative interface identifier with link-local address prefix and probes address with a Neighbor Solicitation message.
- If another host is already using this address then he sends a Neighbor Advertisement message.
- An intruder can use this protocol exchange for a DoS attack.
- IETF Send WG tries to solve this problem.
- Stateful address autoconfiguration (DHCP)

Security Problems?

Binding Updates (BU) are security relevant.

⇒ BU enables source routing



Unprotected BUs:

- **Denial of Service attacks**
- **Man-in-the-Middle attacks**

Binding Updates between HA<->MN and between CN<-> MN experience different protection.

The Home Address

The home address (HoA) must be unique for each MN (global reachable IP address).

Functionality:

- Connection endpoint identifier for long-lived connection
- Is used to reach MN
- (HoA,CoA) pair used to create profile for personalization
- Can used to identify MN for billing and charging (additionally to NAI)

Selected Problem 1: Privacy [RFC2462]

Hosts selects interface identifier

Interface identifier is based on the link layer address

Since the link layer address rarely changes MN is uniquely identified

CoA Prefix reveals location of MN (source address)

HoA

- represents long-lived endpoint identifier
- is unencrypted
- revealed to CN (Route Optimization)

CoA and/or HoA enable profiling

Solutions for Privacy Problem

Bi-directional IPSec tunnel from MN to HA

- Very expensive communication

HA option encryption

- Requires modification to IPSec

IPv6 Privacy Extension

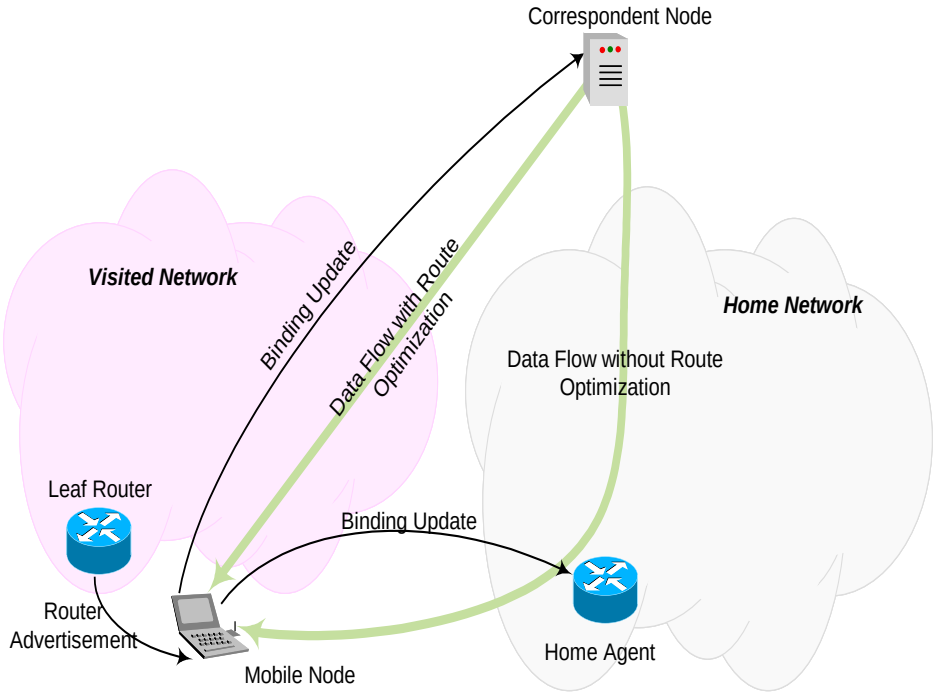
- Changing stateless address autoconfiguration

Disable Route Optimization

- Performance degradation

Castelluccia Mobile IPv6 Privacy Proposal

- Uses Temporal Mobile Identifier
- TMI changed temporarily, HoA encrypted



Standard Sec Infrastructure cannot be used

To enable route optimization

⇒ BU must be sent to CN

Consequences:

- Security Association between MN-CN required
- Previously suggested: IPsec (together with IKE)
- IPsec does not address mobility specific problems; IKE is computationally expensive;
- Public key infrastructure not available
- Protection of BU difficult
  ⇒ IPSec policies too coarse grained
- CN has to run many IKE exchanges
- CN has to store a large number of SAs
- Vulnerability against active attackers may be acceptable
  ⇒ Unauthenticated key agreement/key transport

Selected Problem 2: Address Ownership

Authorization Problem

- MN must show that it is owner of an IP address

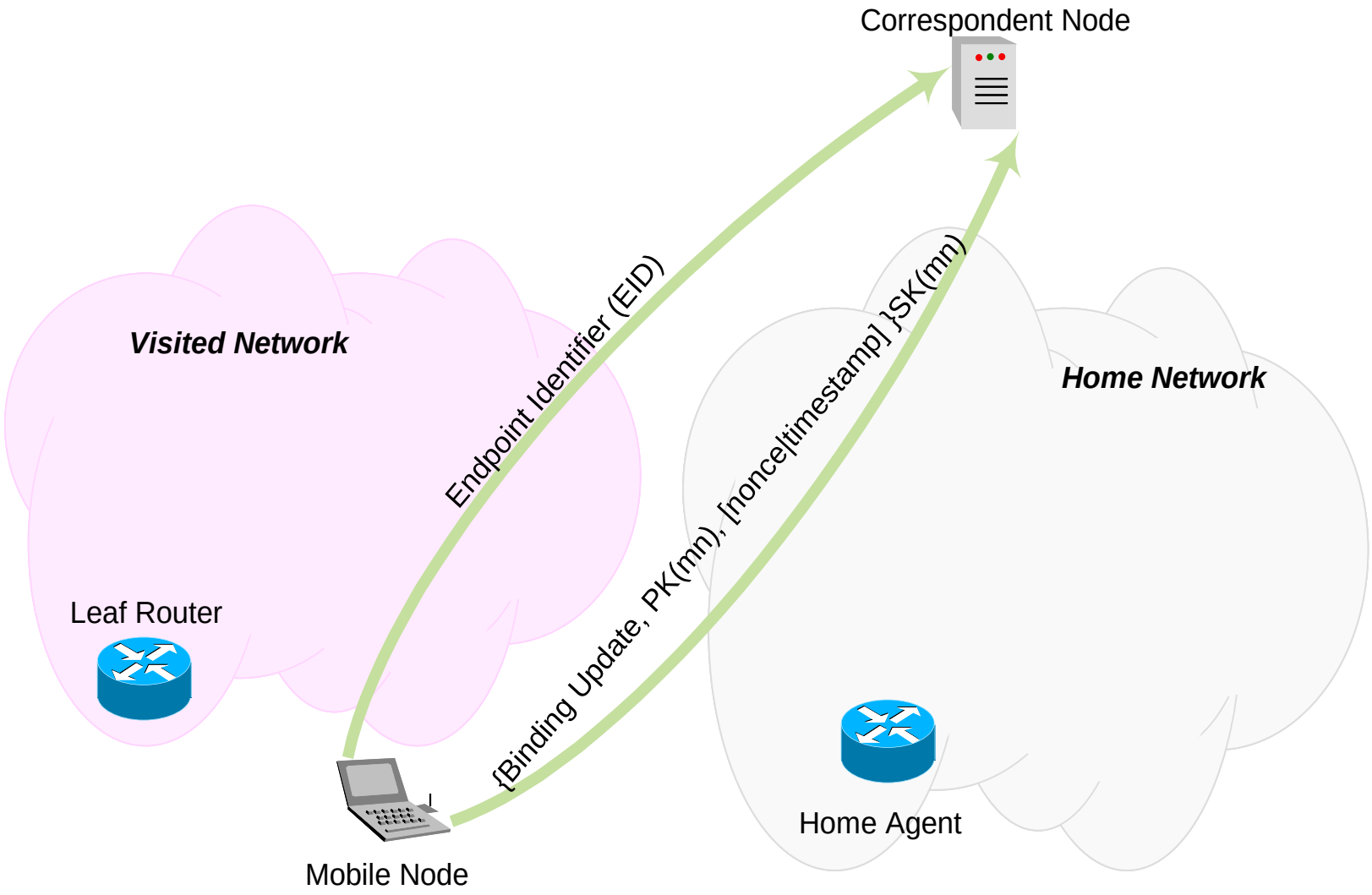
Is this MN allowed to set the (CoA,HoA)-binding?

First proposal to address this mobility & security problem:

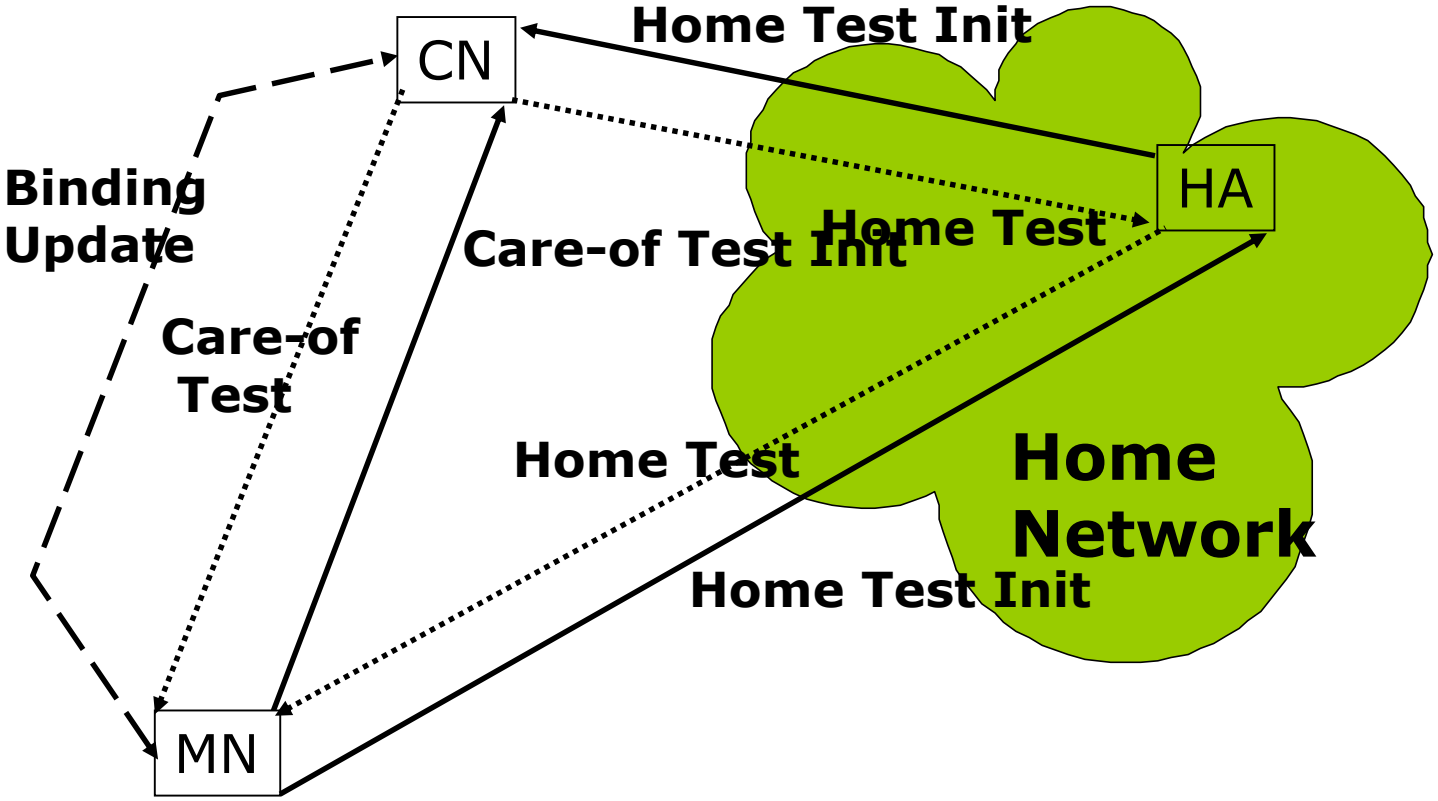
- Purpose Built Keys
- Proposal does not require a PKI or similar security infrastructure
- Does not provide “perfect” security (i.e. protection against all attacks)

After this proposal was published similar proposals have been submitted.

Purpose Built Keys



Mobile IPv6 Security
MN ⇔ CN Binding Update



Security Property: *Return Routability*
Verifies that a node is able to respond to packets sent to a given address
Assumption: *Routing infrastructure is secure*

HIP (Host Identity Payload + Protocol) Overview

Protocol proposal submitted by Bob Moskowitz.

HIP is developed independently (not within an IETF working group).

Protocol proposal contains:

A new namespace / new identity

An authentication and key exchange protocol

Architecture

HIP: A new namespace / new identity

Basic Idea: Cryptographic identity for a host

An IP address to identify a host is not the best idea (see multi-homed hosts, virtual interfaces)

Used Identities:

- Host Identity (=Public Key)
- Host Identity Tag (=hash of the public key, 128 bit)
- SPI (same as in IPSec)
- LSI (32-bit Local Scope Identity)

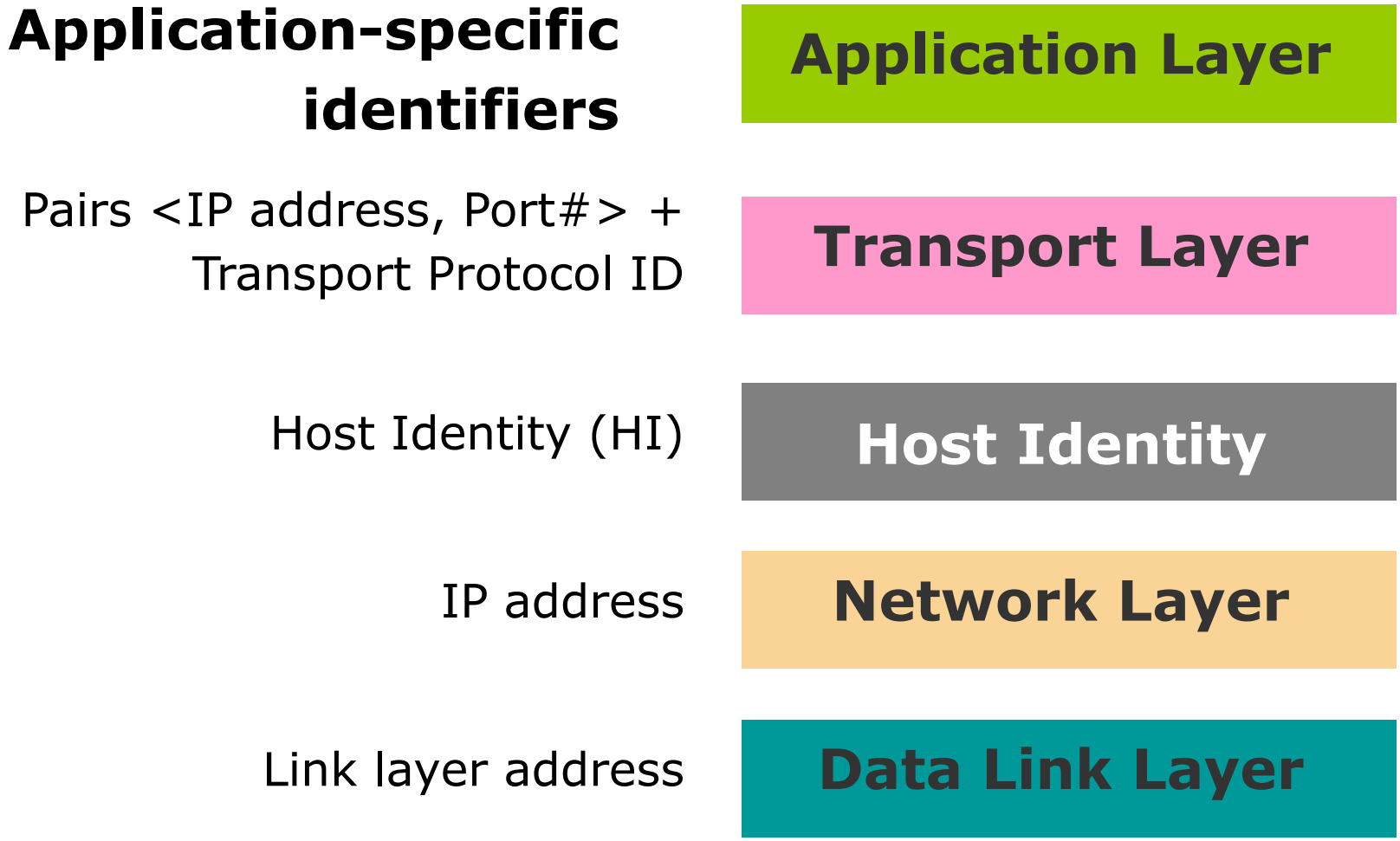
Security Association indexed by Host Identity Tag (HIT)

32 bit value (LSI) is used to support IPv4 applications

Host Identities can be well-known or anonymous

Higher layers only see identities, not addresses

HIP Architecture
An additional Identifier



HIP: Authentication and key exchange

The HIP protocol is used to create an IPSec ESP security association

The protocol has the following properties:

- DoS protection with the client-puzzle mechanism
- Re-keying provided by a separate protocol
- Digital signatures and certificates are exchanged in a DNS like data structure.
- The DNS protocol is strongly integrated with HIP
- Identities are stored into the DNS (DNS Binary Labels allow reverse mapping).

Including the HIP identity in every packet would be difficult.

Therefore HIP is always combined with IPSec ESP where the HIP Identity is "compressed" into IPsec ESP SPI.

HIP Properties

IP addresses still used for routing packets.

Bandwidth conservative

Each host must have at least one key pair

A 128 bit hash or tag to be used in system calls

End-to-end use but integration of intermediate devices planned.

HIT replaces IP address as the ‘name’ of a host

Enables mobility and allows simpler multi-homing

Addressing realm friendliness

Support for different addressing schemes, end-to-end => IPv4/IPv6 migration

What about PKI and HIP?

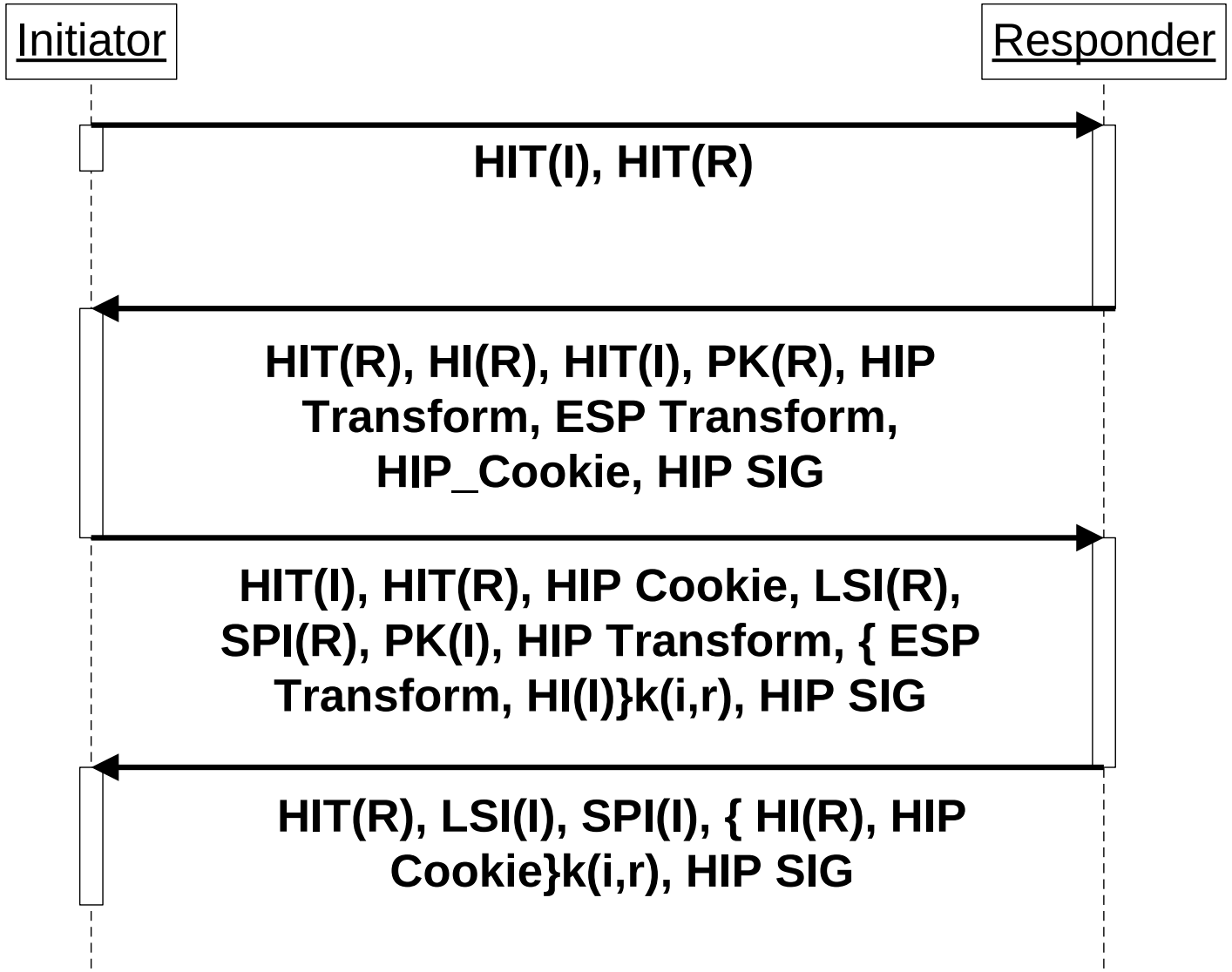
HIP assumes interaction with DNS

- Identity in KEY records
- DNSSEC required for trustable as the 3rd party authentication

Payload uses DNS RR formats

- Reuse existing code
- KEY, SIG, OPT, and A records
- Subject to change to reduce packet size

HIP Protocol Exchange



HIP Protocol Exchange Legend

Host Identity Tag – HIT

Host Identity – HI

I – Initiator

R – Responder

PK(R), PK(I) – Diffie-Hellman Public Key of Responder (Initiator)

k(i,r) – session key computed between I and R

HIP SIG – Digital Signature computed over the entire packet

HIP (ESP) Transport – List of algorithm to be negotiated (used)

HIP Cookie – Values required for the Client Puzzle

LSI – Local Scope Identity

SPI – Security Parameter Index

Special HIP Packets

Message for rekeying

Bootstrapping for the case where the initiator does not possess the HIT of the responder.

Packet to announce readdressing

- Readdressing required because of:
  - –PPP reconnect
  - –DHCP new lease, IPv6 address prefix change
  - –Mobility related readdressing
  - –IPv6 privacy related IP address change

Summary

HIP introduces new and interesting concepts.

The introduction of a new address space based on a cryptographic identity makes a lot of things easier:

- Mobility
- Multi-Homing
- IPv4/IPv6 Transition

Solutions are already there for these problems; HIP solves the problems in a different way.

Additionally HIP has security integrated into the protocol.

Open Source implementations might create an interesting alternative.

Authentication, Authorization and Accounting (AAA)

Authorization: Is a particular entity able to pay for the requested resources?

Which resource?

- Certain services
- Specific QoS
- Amount of time being online
- Data volume transmitted/received

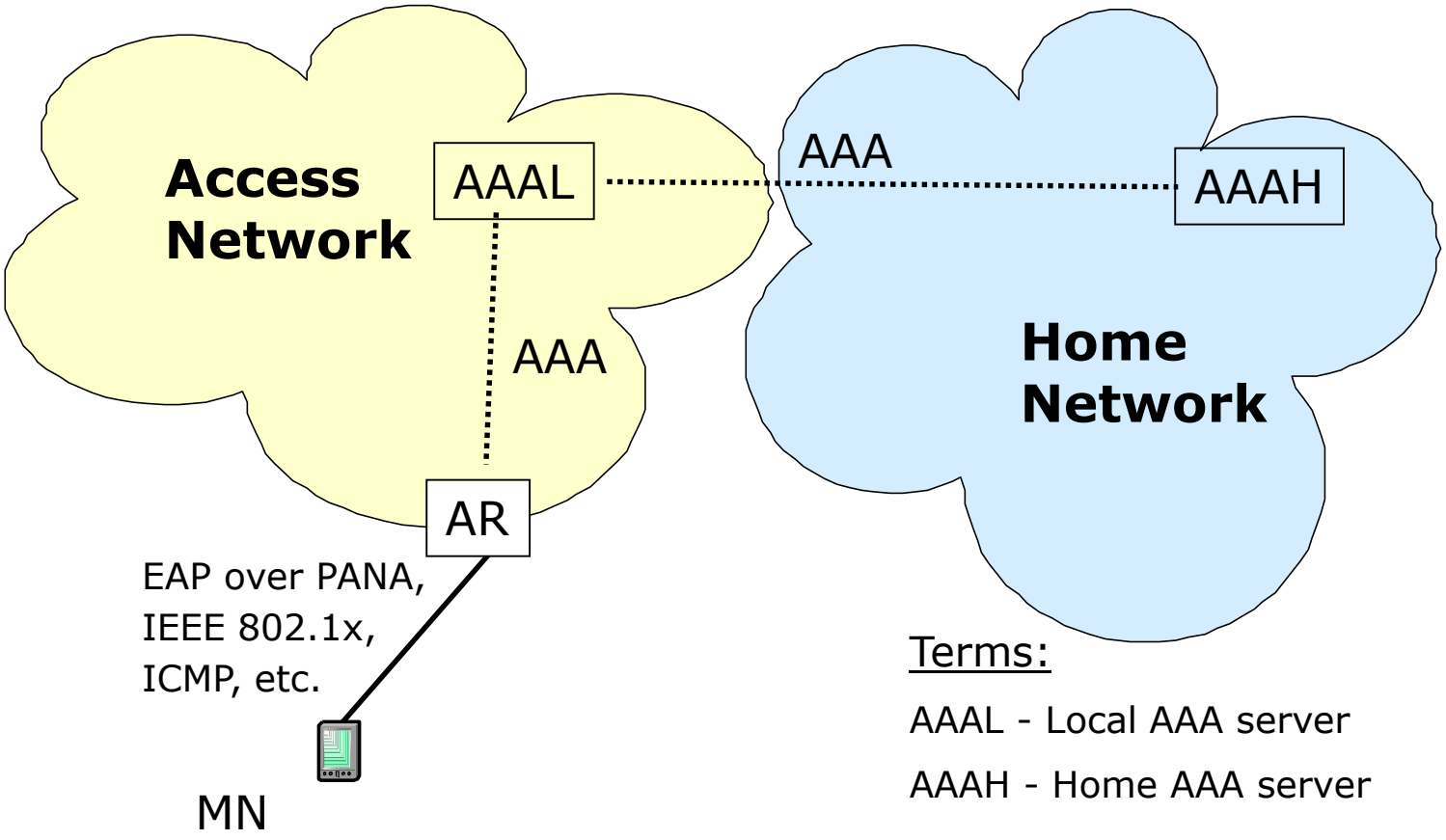
Goal:

- 1) Establishing a financial settlement
- 2) Prevent unauthorized nodes from gaining access to resources

Two basic models for (1):

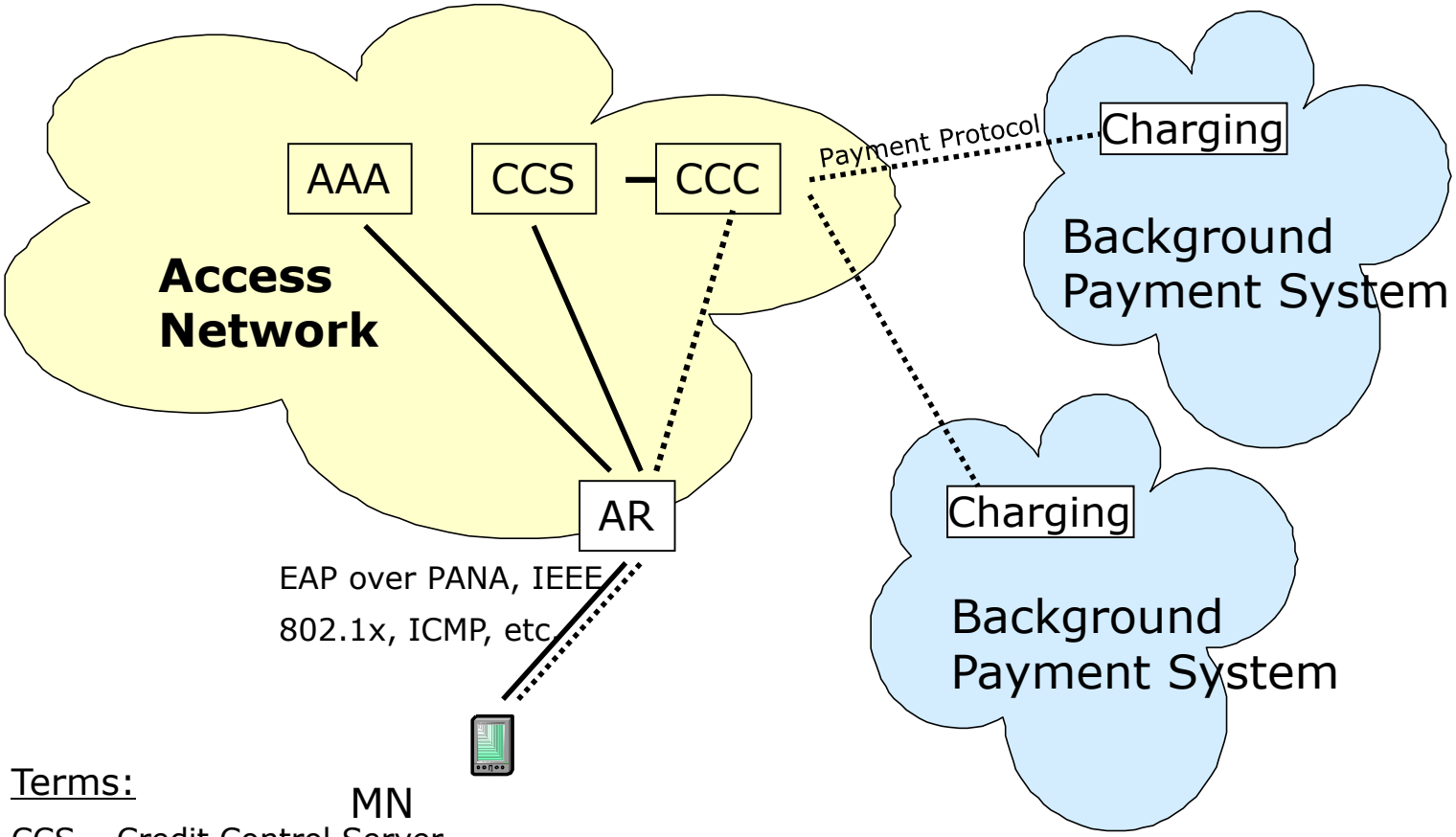
- Subscription-based Architecture
- Alternative Access Architecture

Subscription-based Architecture



- MN is registered at home network (typically secret key based).
- Several protocol proposals exist for transport of AKA information between MN and the AAA attendant.

Alternative Access Architecture



Terms:

CCS – Credit Control Server

CCC – Cost Charging Centre

PANA

Protocol for carrying Authentication for Network Access (pana)

Provides carrier for EAP messages over IP (UDP)

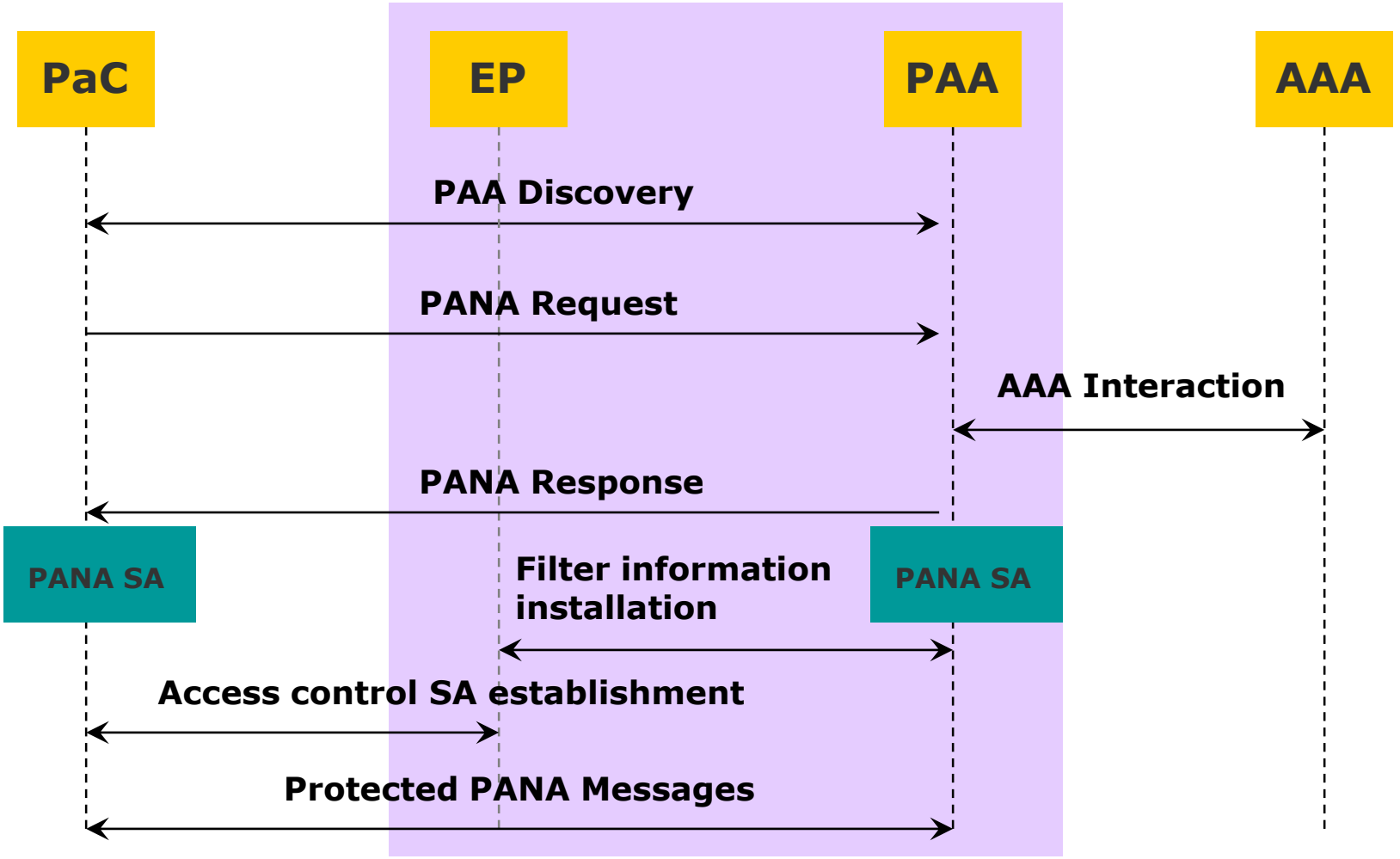
Provides in-order delivery of packets

PANA is a protocol for heterogeneous network access (link layer agnostic).

PANA provides a mechanism for the PAC to discover the PAA on the link

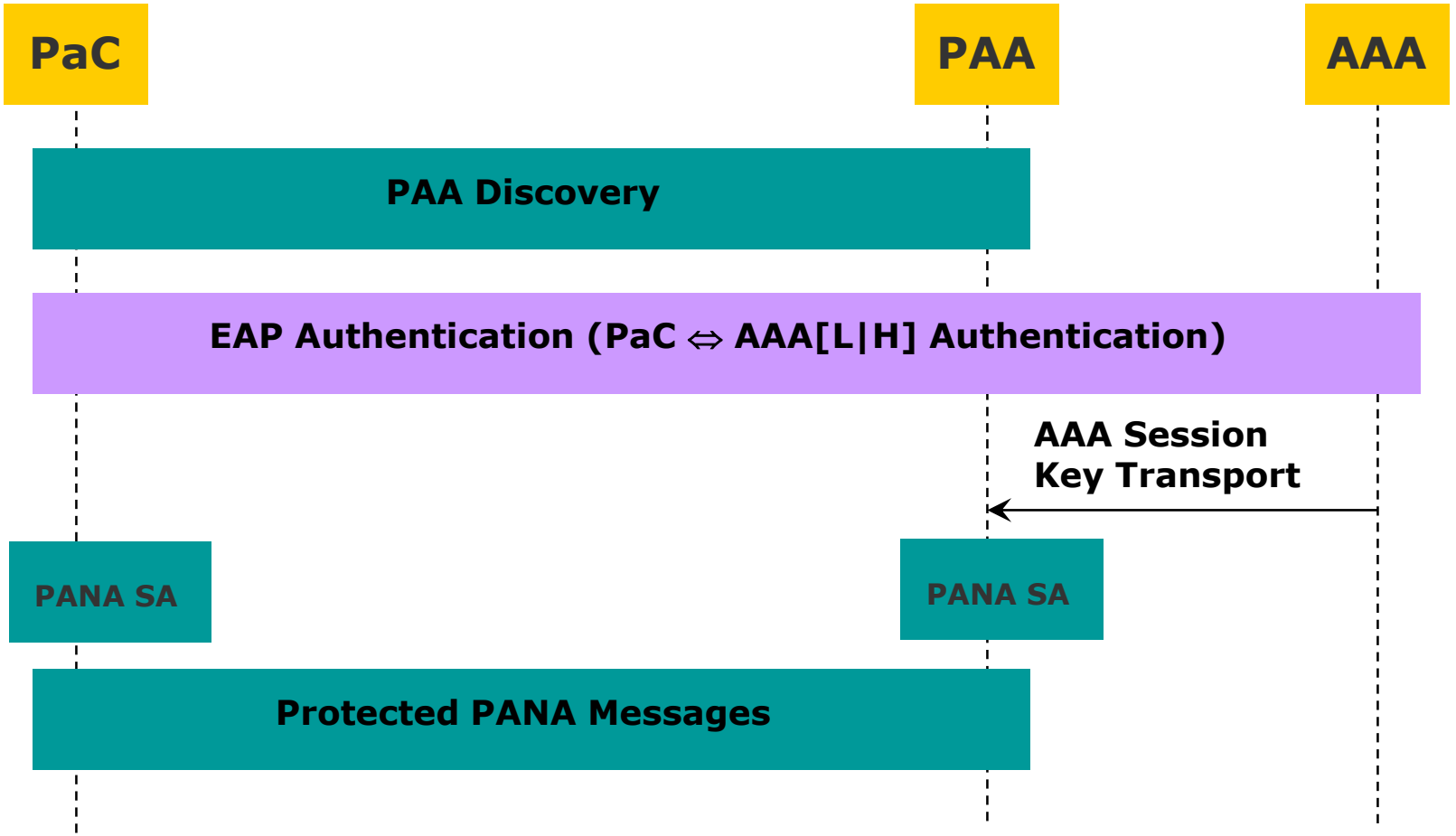
Provides different mechanisms to prevent unauthorized nodes from accessing the network (interaction with other protocols)

PANA Framework



Note that some protocol interactions are optional.
Terminology: http://www.ietf.org/internet-drafts/draft-ietf-pana-requirements-05.txt

PANA Security Association Establishment



PANA relies on EAP methods to produce keying material for PANA SA.

Literature

PANA IETF WG:

- http://www.ietf.org/html.charters/pana-charter.html

WLAN Security:

- Fluhrer, Mantin, Shamir: "Weaknesses in the Key Scheduling Algorithm of RC4"
  (see http://citeseer.nj.nec.com/fluhrer01weaknesses.html)

EAP IETF WG:

- http://www.ietf.org/html.charters/eap-charter.html

AAA IETF WG:

- http://www.ietf.org/html.charters/aaa-charter.html

PPPEXT IETF WG:

- http://www.ietf.org/html.charters/pppext-charter.html

Airsnort Software:

- http://airsnort.shmoo.com/

Open Source IEEE 802.1X (EAP) Implementation:

- http://www.open1x.org

Literature

Linux (FreeS/Wan: http://www.freeswan.org/)

Contains AH, ESP (Klips) and IKE (Pluto)

IETF IPsec WG: http://www.ietf.org/html.charters/ipsec-charter.html

AES Support: http://www.irrigacion.gov.ar/juanjo/ipsec/

IPv6 and IPSec: http://www.ipv6.iabg.de

BSD (e.g. http://www.netbsd.org/Documentation/network/ipsec/)

IKE Daemon: Racoon

Provides Traffic Selectors at a fine-grain granularity and “policy” management

PF_KEY: RFC 2367

API for the communication with the kernel-based key engine (Security Association Database (SADB) and Security Policy Database (SPD))

Literature

Kerberos IETF WG:

- http://www.ietf.org/html.charters/krb-wg-charter.html

Kerberos V4

- Steiner, B., Neuman, C., Schiller, J.: "Kerberos: An Authentication Service for Open Network Systems", USENIX Conference, (Dallas, TX), pp. 191-201, 1988.

KINK

- http://www.ietf.org/html.charters/kink-charter.html

Kerberos V5

- http://www.ietf.org/rfc/rfc1510.txt

MIT Kerberos Software implementation:

- http://web.mit.edu/kerberos/www/

Literature

IETF Mobile IP WG:

- http://www.ietf.org/html.charters/mobileip-charter.html
- Particularly interesting are the following drafts:
  - –MIPv4, MIPv6; Hierarchical Mobile IPv6 mobility management
  - –Fast Handovers for Mobile IPv6

IETF AAA WG:

- http://www.ietf.org/html.charters/aaa-charter.html

IETF Context Transfer, Handoff Candidate Discovery, and Dormant Mode Host Alerting (seamoby):

- http://www.ietf.org/html.charters/seamoby-charter.html

IRTF Routing Research Group (Micromobility)

- http://www-nrc.nokia.com/sua/irtf-mm-rr/IRTF-mm-rr.htm

Literature

Host Identity Payload and Protocol

- http://homebase.htt-consult.com/~hip/draft-moskowitz-hip-05.txt

Host Identity Payload Implementation Issues

- http://homebase.htt-consult.com/~hip/draft-moskowitz-hip-impl-01.txt

Host Identity Payload Architecture

- http://homebase.htt-consult.com/~hip/draft-moskowitz-hip-arch-02.txt

Implementations:

- HIPL: HIP for Linux http://gaijin.iki.fi/hipl/