Project: IEEE P802.20 Working Group on Mobile Broadband Wireless Access

Submission Title: [Distributed Security Proposal Certicom]

Date Submitted: [5 March, 2003]

Source: [René Struik] Company [Certicom Corp.]
Address [5520 Explorer Drive, 4th Floor, Mississauga, ON Canada L4W 5L1]
Voice:[+1 (905) 501-6083], FAX: [+1 (905) 507-4230], E-Mail:[rstruik@certicom.com]

Abstract: [This document describes elements of a highly flexible security architectural framework for IEEE 802.20 based on the characteristics of this network and its intended operational usage.]

Purpose: [Highlight issues that need to be addressed to flexibly facilitate IEEE 802.20 security.]

Distributed Security Proposal for IEEE 802.20 Mobile Broadband Wireless Access

(Conceptual Discussion)

René Struik, Certicom Research

Outline

1. Security Concepts – A Short Introduction
2. Security Architectures – Distributed vs. Centralized Model
3. Security Architectural Framework
4. The Security Architecture at Work – Operational Description
5. Cryptographic Protocols – Example Implementation

Security Concepts – A Short Introduction

Outline

- Basic Security Services
- Message Authentication
- Entity Authentication
- Data Confidentiality
- Symmetric-Key Cryptography
- Public-Key Cryptography
- Digital Certificates

Basic Security Services

- **Authenticity**

Evidence as to the true source of information or the true identity of entities:

- *Message authentication*.

Evidence regarding the true source of information:

(1) No undetectable modifications, deletions, and injections of messages by external parties (data integrity);

(2) No confusion about who originated the message (source authenticity).

- *Entity authentication*.

Evidence regarding the true identity of entities and on their active involvement:

(1) No confusion about whom an entity is really communicating with (authenticity);

(2) Proof that entity is actively participating in communications (i.e., is 'alive').

- **Secrecy**

Prevention of external parties from learning the contents of information exchanges:

(1) Logical separation of information between parties that may have access to info and those that do not.

(2) No confusion about whom those privileged parties are (authenticity).

Cryptographic Building Blocks - Authentication (1)

- **Message authentication**

Evidence regarding the true source of information:

(1) No undetectable modifications, deletions, and injections of messages by external parties (data integrity);

(2) No confusion about who originated the message (source authenticity).

Realizations:

- *Keyed hash function (or Hash Message Authentication Code (HMAC)).*

Mapping of arbitrary messages (of any length) to *compact representative image* hereof, using a secret key.

(1) Data integrity, since difficult to find distinct messages with same MAC value.

(2) Source authentication, since only parties that share the secret key can produce MAC-value (assuming there is no confusion about who has access to this key).

- *Un-keyed hash function.*

Mapping of arbitrary messages (of any length) to *compact representative image* hereof (digital fingerprint, or message digest), without secret key.

(1) Data integrity, since difficult to find distinct messages with same hash value.

(2) Source authentication, *only if* message digest is communicated authentically.

Cryptographic Building Blocks - Authentication (2)

- **Entity authentication**

Evidence regarding the true identity of entities and on their active involvement:

(1) No confusion about whom an entity is really communicating with (authenticity);

(2) Proof that entity is actively participating in communications (i.e., is 'alive').

Realizations:

- *Entity authentication protocol (challenge response protocol).*

(1) Source authentication, since only parties that share the secret key can produce proper responses to unpredictable challenges (assuming there is no confusion about who has access to this key).

(2) Aliveness, since challenge messages are unpredictable and never repeated. (Hence, replaying previously recorded protocol messages does not leak info.)

Cryptographic Building Blocks - Secrecy

- **Secrecy**

Prevention of external parties from learning the contents of information exchanges:

(1) Logical separation of messages between parties that may have access to info and those that do not.

(2) No confusion about whom those privileged parties are (authenticity).

Realizations:

- *Symmetric-key cryptography.*

Logical separation of information, since only parties that share the secret key can learn the contents hereof (assuming there is no confusion about who has access to this key). Note that the symmetric key is used both for encryption and for decryption.

- *Public-key cryptography.*

Logical separation of information, since only parties that have access to the private decryption key can learn the content of encrypted messages (assuming there is no confusion about who has access to this private key). Note that any party may obtain access to the public encryption key, since it does not reveal the decryption key.

Cryptographic building blocks – Authenticity and Secrecy (1)

- **Symmetric-key cryptography**

Security services based on exchange of secret and authentic keys:

(1) Logical separation of messages, by exchanging secret keys between privileged parties only;

(2) Authenticity of privileged parties by checking credentials of each party, by non-cryptographic means (certified mail, courier, face-to-face, etc.)

- **Public-key cryptography**

Security services based on exchange of authentic public keys:

(1) Logical separation of messages, by restricting access to each private key to the privileged party only (in practice, there is only 1 privileged entity);

(2) Authenticity of privileged parties, by checking credentials of each party by non-cryptographic means and (if successful) by subsequently binding the public key to this party (*certification of public keys).*

Certification is done by a so-called Trusted Third Party, who vouches for the authenticity of the binding between an entity and its public key.

Cryptographic building blocks – Authenticity and Secrecy (2)

- **Public-key cryptography (cont'd)**

Certification of public keys depends on appropriately checking the credentials of a party and constitutes the following:

(1) Check, by cryptographic means, that the entity A that claims to have access to the public key P_A, has access to the corresponding private key S_A;
(2) Check, by non-cryptographic means, the claimed identity Id_A of A.

Certification is done by a so-called trusted third party:

- *Digital certificates (cryptographic binding).*

(1) Authenticity of binding, via signature over the pair (Id_A,P_A) by trusted party;
(2) Verification of authenticity of public keys *by any party*, by verifying signature of trusted party in the digital certificate (assuming the authentic storage of trusted party's signature verification string on each verifying device);
(3) Unrestricted transfer of certificates possible (hence, off-line certification possible).

- *Manual 'certificates'(non-cryptographic: pushing button, low power mode, etc.).*

(1) No cryptographic verification of the authenticity of public keys possible;
(2) No transfer of certificates possible (hence, on-line 'certification' only).

The Need for a
Distributed Security Architecture

Outline

- Underlying Network Technology
- (Potential) Network Security Objectives
- Devices and their Roles
- Security Policy
- Access Control to the Network
- The Need for a Distributed Security Model

Underlying Network Technology (details TBD)

Communication technology

- Transmission technology (e.g., frequency band for radio transmissions if wireless)
- Anticipated data rates (including, e.g., breakdown depending on device type)
- Hearing range (i.e., range over which direct communication between static and moving devices may be possible without routing)
- Cost factors (e.g., low power, low cost, low complexity devices)
- Special communication types (e.g., beacon and beacon-less communication)

Devices

- Anticipated application area of devices, networks

Network Characteristics

- Network topology (e.g., star network, mesh networks)
- Communication patterns (e.g., peer-to-peer, multicast, broadcast)
- Devices and their roles and capabilities (e.g., ordinary device, network coordinator)
- *Intra*-network behavior and *inter*-network behavior (roaming)

Interaction with outside world

- Gateway to other networks (e.g, node that communicates MAC service data units back and forth)
- Access to infrastructure (e.g., no online access to external infrastructure (external to network))

NOTE: Presentation relies on concepts only, but assumes distinct network coordinator role

(Potential) Network Security Objectives

• **Access control to the Network itself**

Restriction of access to scarce network resources to authorized devices only, to ensure objectives including the following:

- proper bandwidth allocation;
- protection of commands (e.g., those regulating network membership);
- power drain savings.

• **Control of access to message traffic between Network devices**

Restriction of access to information secured between members of a group of Network devices to precisely these group members. This includes any of the following objectives:

- *Confidentiality*.
  Prevent external parties from learning the content of exchanged messages.
- *Data integrity/message authentication.*
  Prevent external parties from modifying or injecting messages in undetected way.

• **Other** …

Devices and their Roles (1)

Role model

- *Security manager.* Sole source of local trust management.
  -Facilitates establishment of symmetric keying material between ordinary devices;
  -Facilitates maintenance of keying relationships;
  -Enforces security policy.
- *Ordinary device.* Part of Network or could become part hereof.
  - Responsible for secure processing and storage of keying material;
  - Responsible for maintenance of its own Access Control List (ACL) and other lists.
- *Coordinator.* Potential source of local message control (in Network).
  -Facilitates admission of ordinary devices to the Network;
  -Allocates time slots for message exchanges between devices;
  -No security responsibilities (apart from access control to the Network).
- *External trusted party.* Sole source of global trust management.
  -Facilitates establishment of public keying material between ordinary devices;
  -Facilitates maintenance of public keying relationships;
  -Enforces security policy.

NOTE: For discussion's sake, we assume that the Network Coordinator always performs access control to the Network (not implemented if not needed).

Devices and their Roles (2)

Mapping of roles to devices

Devices need way to recognize which role(s) other devices play or should play.

- *Static mapping*. Mapping may be defined at initialization.
- *Dynamic mapping*. Mapping must be realized by securely associating roles to devices, allowing dynamic verification (e.g., via attribute certificates).

'Permanent' mappings of roles to devices

The following mapping of roles to devices are *always* in effect:

- Each device assumes the role of ordinary device (for all devices);
- The Network coordinator device assumes the role of coordinator (for all Network devices);
- Each device may assume the role of (alternate) coordinator;
- Each device may assume the role of security manager (for any subset of devices that include itself).

Devices and their Roles (3)

The role of the external trusted party includes facilitating the generation of authentic [public] keying material for each device. As such, it includes
- (facilitating) the generation of a public/private key pair for each device, if needed;
- generation of certificates for each device's public key;
- (facilitating) the storage of an authentic copy of the trusted party's own public key signature verification key in each device, prior to its operational deployment.

There is (conceptually) only 1 entity that assumes the role of external trusted party (for all devices).

NOTE: If there is actually more than 1 external trusted party, each device is assumed to have access to the other external trusted party's 'root' key, either directly, e.g., via a list of CA root keys, or indirectly, e.g., via cross-certification techniques.

The role of the external trusted party is implemented outside the network (CA functionality).

Devices and their Roles (4)

Actual mapping of roles to devices
The actual mapping of the Coordinator and Security Manager role to a device might change over time.

EXAMPLES:

I. Centralized security model (Master/slave model)

There is a single device that assumes the role of Security Manager (for all devices).

II. Distributed security model (Our proposed mapping of roles to devices)

Each device assumes the role of Security Manager (for itself only).

NOTE: **(Hybrid security model)**
If desired, one can 'relax' this mapping by postulating that each device assumes the role of Security Manager for himself and for all other devices that trust him ('friendship' scenario).

A detailed discussion of properties to follow later.

Security Policy (1)

The security policy specifies rules that must be adhered to to keep security properties of system invariant, in the event of security events.

Discussions are relative to a specific set of Network devices (group).

Security events

1. Change of group structure.
   (a) Exclusion of an old group member from the group:
   - *Expiration of group membership*. Disassociation due to time-out.
   - *Cancellation of group membership*. Disassociation due to cancellation request.
   - *Denial of access.* Disassociation due to enforcement of security policy.

   (b) Introduction of a new group member to the group:
   - *Subscription of the member*. Authentication of newly associated device.

2. Change of (security relevant) role.
   (a) Hand over of coordinator;
   (b) Changes to list of devices that are trusted as Security Manager for specific device;
   (c) Changes to list of devices that are trusted as External Trusted Party for specific device.

Simultaneous changes to the group structure and to the security relevant role are conceptually thought of as to occur subsequently (in any order).

Security Policy (2)

I. Effect of security events - change of group structure

Scenario where information shared between group members is secured via a common (symmetric) group key.

Security invariant

At any given moment of time, access to information shared between members of a group is restricted to precisely these group members. As such, this includes access to integrity information.

Security rule

Changes to the group structure shall invoke a change to the common group keys.

Rationale

1. This prevents a new group member from gaining access to secured information communicated *prior to* the moment he obtained access to the key-sharing group.
2. This prevents an old group member from gaining access to secured information communicated *after* the moment he was denied access to the key-sharing group.

Security Policy (3)

I. Effect of security events - change of group structure (cont'd)

Key storage invariant
At any given moment of time, devices maintain symmetric keying relationships with groups to which they belong only.

Key storage rule
Changes to the group structure shall invoke the secure destruction of the old group key(s) and the secure and authentic storage of the new group key(s).

Rationale
This limits the impact of the potential compromise of symmetric keying material to exposure of information to which the device already has access as a legitimate group member.

Security Policy (4)

II. Effect of security events - change of security relevant role

Scenario where information shared between group members is secured via a common (symmetric) group key.

1. Changes between a group member's role as coordinator and as ordinary device have no impact on the group structure, hence these do not impact the group key(s).

2. Changes to the list of devices that are trusted to assume the Security Manager role of the device have no impact on the group structure, hence these do not impact the group key(s).

NOTE: Exclusion of a security manager from a device's list of trusted security managers **does** have an impact on key usage, as follows:

Key usage rule

(a) If a device excludes a security manager (i.e., does not trust it any more), it stops using any keying material that originated or will originate from this security manager, for encryption purposes (use for decryption purposes may still occur, though).

(b) If a device includes a security manager (i.e., it trusts it as key distribution source), it starts using any keying material that will originate from this security manager from that instance in time onwards, both for encryption and decryption purposes.

Access Control to the Network (1)

The access control policy specifies how devices shall communicate in the Network.

Discussions are relative to a particular Network and do assume this (sub)network to operate in one of its secure modes.

Admission to the Network

Admission to the Network is based on the outcome of the following protocols between the prospective joining device and the Network Coordinator (not necessarily in order):

1. Mutual entity authentication protocol.
   The device and the Network Coordinator engage in a mutual entity authentication protocol based on public-key or symmetric-key techniques. This protocol provides evidence regarding the true device identity of both the joining device and the Coordinator, based on authentic public or symmetric keys.

2. (optional) Additional authorization techniques between both devices, based on, e.g., Access Control Lists (ACLs).

If devices have been positively authenticated and have been authorized, these are admitted to the Network. Addressing of these devices may take place using their local Id (shorthand Id), rather than their global Id (IEEE MAC Address). For this an ‘fresh’ (unused) local Id is assigned to the joining device.

Access Control to the Network (2)

Remark

Devices in the piconet fully depend on information provided by the Coordinator regarding which devices have been admitted to the piconet (since admission is based on communication between the Network Coordinator and a joining device only).

Corollary (Effect of improper device list in broadcast scenario)

Assume the following scenario:

- All devices in the Network share a common broadcast key;
- The list of admitted devices to this network is

L:={(local device Id, global device Id)}.

Then failure to obtain the complete and authentic list of admitted devices has the following consequences:

- *'Fly on the wall' problem.*
- *'Switchboard' problem.*

Remark (generalization of threat scenario)

This property also holds in other settings where a key-generating party does not share complete and authentic information on the composition of the key-sharing group itself with the other members of this group.

Access Control to the Network (2a)

Corollary (Effect of improper device list in broadcast scenario)

Assume the following scenario:

- All devices in the Network share a common broadcast key;
- The list of admitted devices to the Network is L:={(local device Id, global device Id)}.

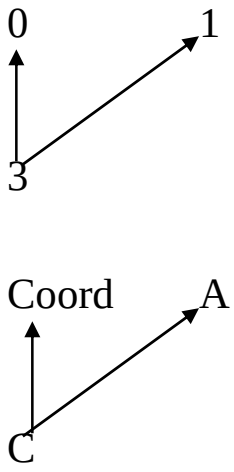
Then failure to obtain the complete and authentic list of admitted devices has the following consequences:

- *'Fly on the wall' problem.*

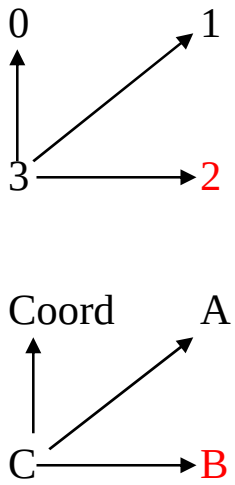
  If a device obtains an incomplete list $L' \subset L$ ($L' \neq L$) of admitted devices, all devices in the complementary set $L \setminus L'$ are 'invisible' to the device. Hence, the device might mistakenly think to share secured information only with devices from the list L', whereas actually it is with other devices of the set L as well, *and unknowingly* so. This obviously violates sound security practice.

C wants to broadcast info based on his local address book

Intended behavior: to A, Coordinator



Actual behavior: to A, B, Coordinator



Coordinator list

	Global Id	Local Id
A	0x31415926	0x01
B	0x27173921	0x02
C	0x45612343	0x03

C's local list

	Global Id	Local Id
A	0x31415926	0x01
C	0x45612343	0x03

Access Control to the Network (2b)

Corollary (Effect of improper device list in broadcast scenario)

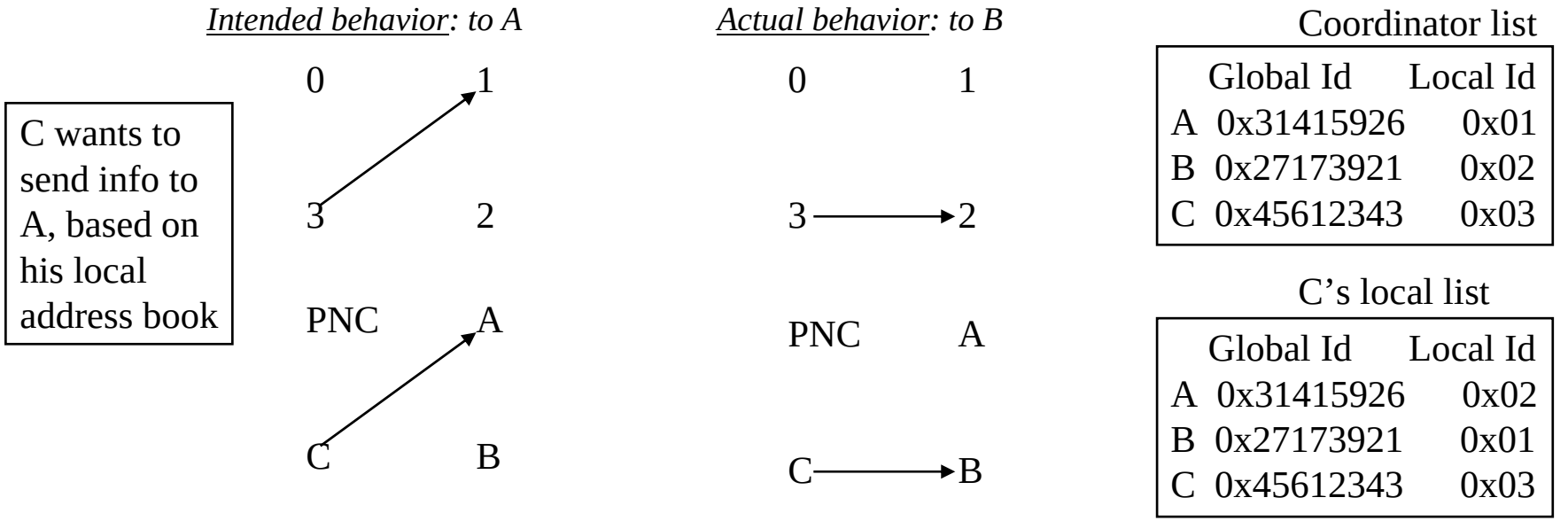
Assume the following scenario:

- All devices in the Network share a common broadcast key;
- The list of admitted devices to the Network is *L*:={(local device Id, global device Id)}.

Then failure to obtain the complete and authentic list of admitted devices has the following consequences:

- *'Switchboard problem'.*

  If the binding between the local device Id and the global device Id is incorrectly received (e.g., 2 entries are interchanged) a device might direct information to the improper device and so compromise the intended security.



Access Control to the Network (3)

Consequences (Effect of improper device lists on security policy)

According to the security policy,

"changes to the group structure shall invoke a change to the common group keys."

This rule can only be enforced if each device takes one of the following two stands:

- Completely rely on the Network Coordinator and on all key generating devices for key-sharing groups to which he belongs, to provide up-to-date and authentic information on the current group composition. This requires a complete dependency on the key generating devices and on the Network Coordinator.
- Maintain up-to-date and authentic information on 'aliveness' of devices with whom the device shares keying material himself. This requires no reliance on the key generating devices, nor on the Network Coordinator. It does, however, require regular re-authentication of all key-sharing devices (the so-called 'heartbeat' scenario, where two devices verify each other's 'aliveness').

Solution

Since complete trust in a Network Coordinator is not realistic in all usage scenarios, this threat can only be diverted properly as follows:

- Each device generates its own keys for its intended audience;
- Each device regularly performs a 'heartbeat' function, to obtain semi-continuous authentication information.

The Need for a Distributed Security Model

A centralized security model is completely unacceptable from a security perspective, except for the most simple usage scenarios (e.g., fixed hierarchical networks).

I. Centralized security model (Master/Slave model)

The Security Manager role is identified with 1 single device for all devices, hence one has the following:

- Concentration of all trust in 1 device:
  - each device must trust the same Security Manager;
  - each device must *trust each subsequent* Security Manager (in case the security manager moves).
- Change of Security Manager:
  - potentially expensive re-establishment of keying relationship between devices and the Security Manager.
- At any given moment in time, the Network Coordinator must provide each Network device with complete and authentic information on the current composition of the Network membership (in reality: at regular time intervals). Each device must regularly check that the Network Coordinator is still 'alive'.

II. Distributed security model (Our proposed mapping of roles to devices)

The Security Manager role is identified with each individual device, hence one has the following:

- No reliance on other devices for trust functionality:
  - each device need only trust himself as Security Manager.
- Change of Security Manager does not have a major impact on individual keying relationships ('smoothness').
- At any given moment in time, each device must re-authenticate with each of its key-sharing parties, to obtain 'aliveness' guarantees (in reality: at regular time intervals). Similarly, each device must regularly check that its Security Managers are still 'alive'.

Security Architectural Framework

Outline

- Overview
- Key Establishment
- Key Transport
- Data Transport

Security Architectural Framework: Overview (1)

Security mechanisms:

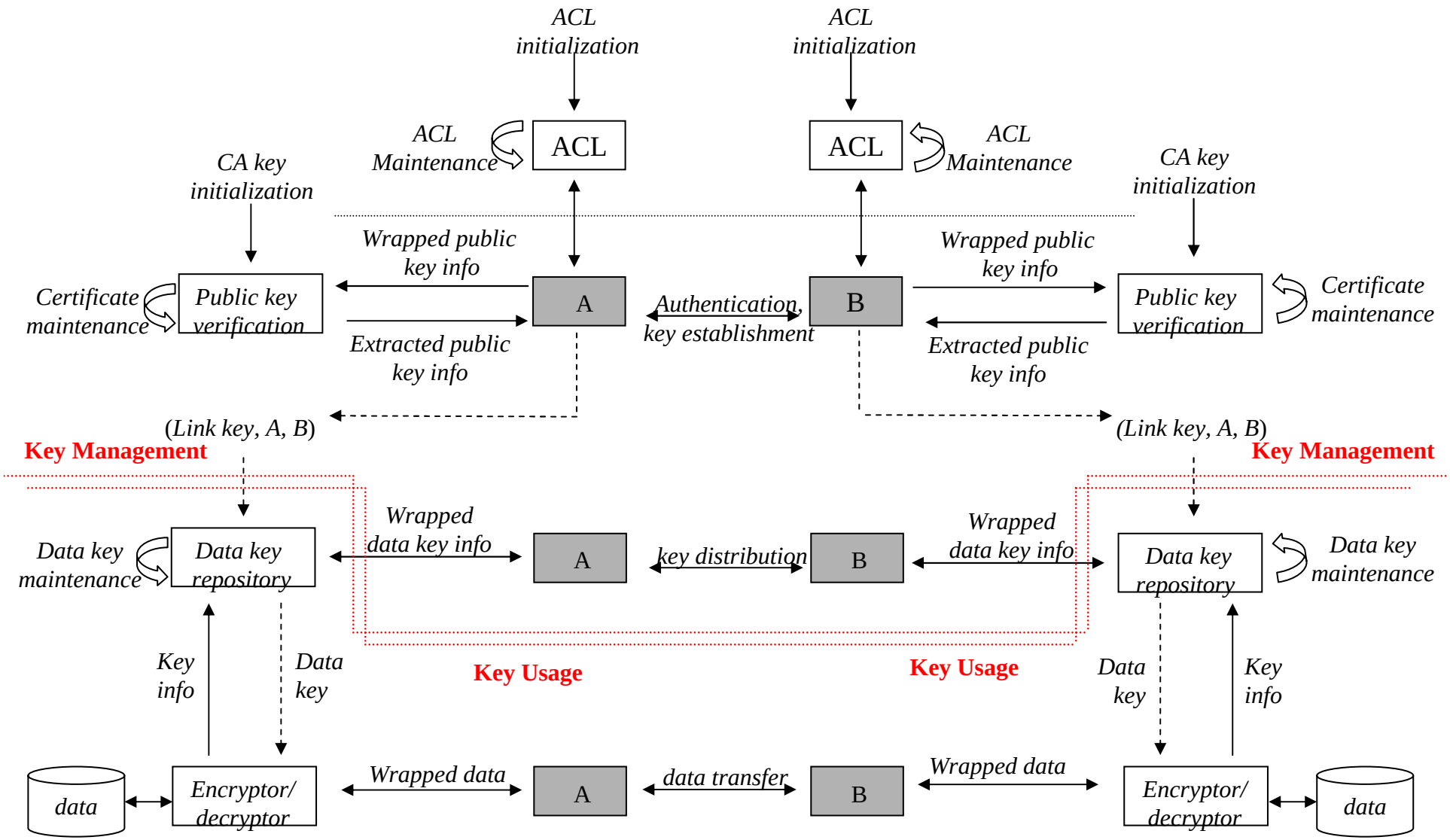
1. Public-key *key establishment* mechanism. Derivation of link key between two devices, based on authentic public keys of both parties, including evidence on whom this link key is shared with.
2. Symmetric-key *key transport* mechanism. Secure transfer of data key from one device to other(s), based on link key(s) between sender and recipient(s).
3. Symmetric-key *data transfer* mechanism(s). Secure and/or authentic data transfer between devices that share the data key (confidentiality/data integrity/authenticity).

Security policy:

…

Note: Security architectural framework with symmetric-key key establishment is very similar to that with public-key key establishment (details omitted here).

Security Architectural Framework: Overview (2)



Security Architectural Framework: Authorization (1)

Authorization and key establishment is based on each of the following:

(1) Evidence regarding the true identity of the other device;

(2) Evidence whether one wants to communicate with this explicitly identified device.

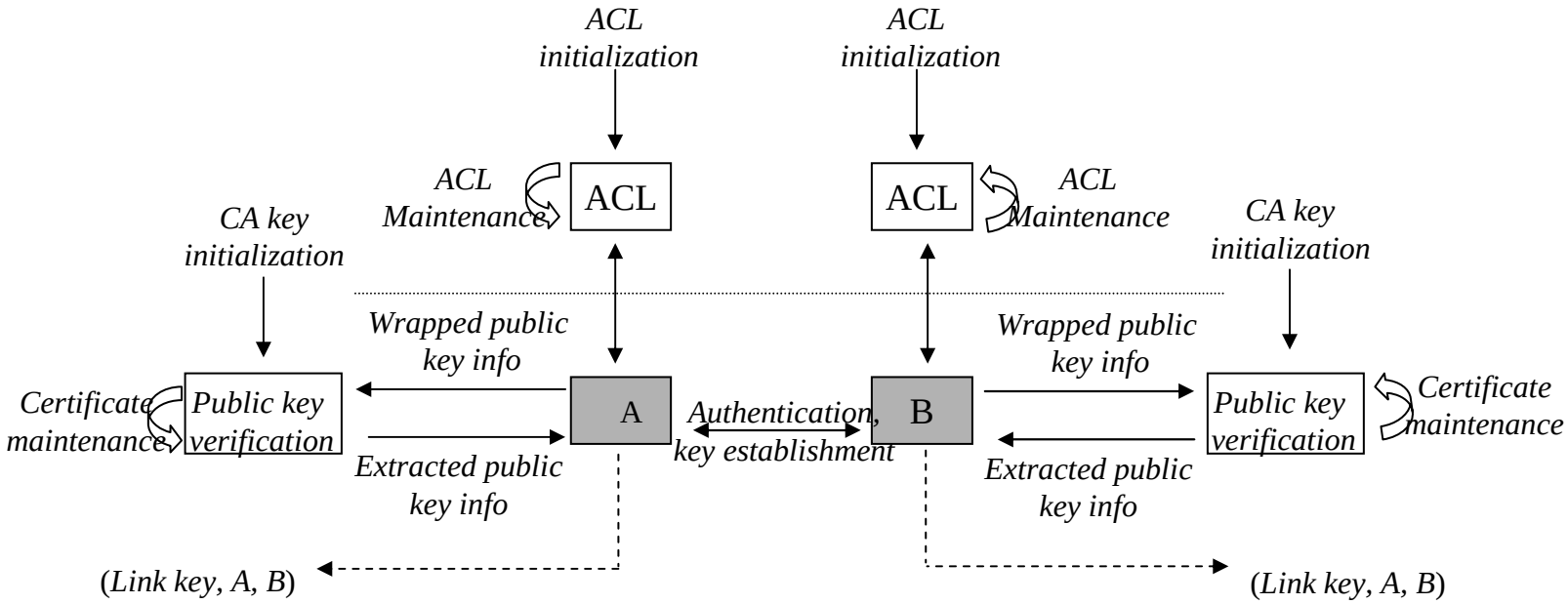
Cryptographic mechanisms:

1. Public-key *key establishment* mechanism. Derivation of link key between two devices, based on authentic public keys of both parties, including evidence on whom this link key is shared with.
2. Symmetric-key *key establishment* mechanism. Derivation of link key between two devices, based on secret and authentic pre-shared key between both parties, including evidence on whom this link key is shared with.

Non-cryptographic mechanisms:

1. Acceptability *test*. Establishment whether a particular device is to be accepted, based on a membership test of a so-called Access Control List (ACL).

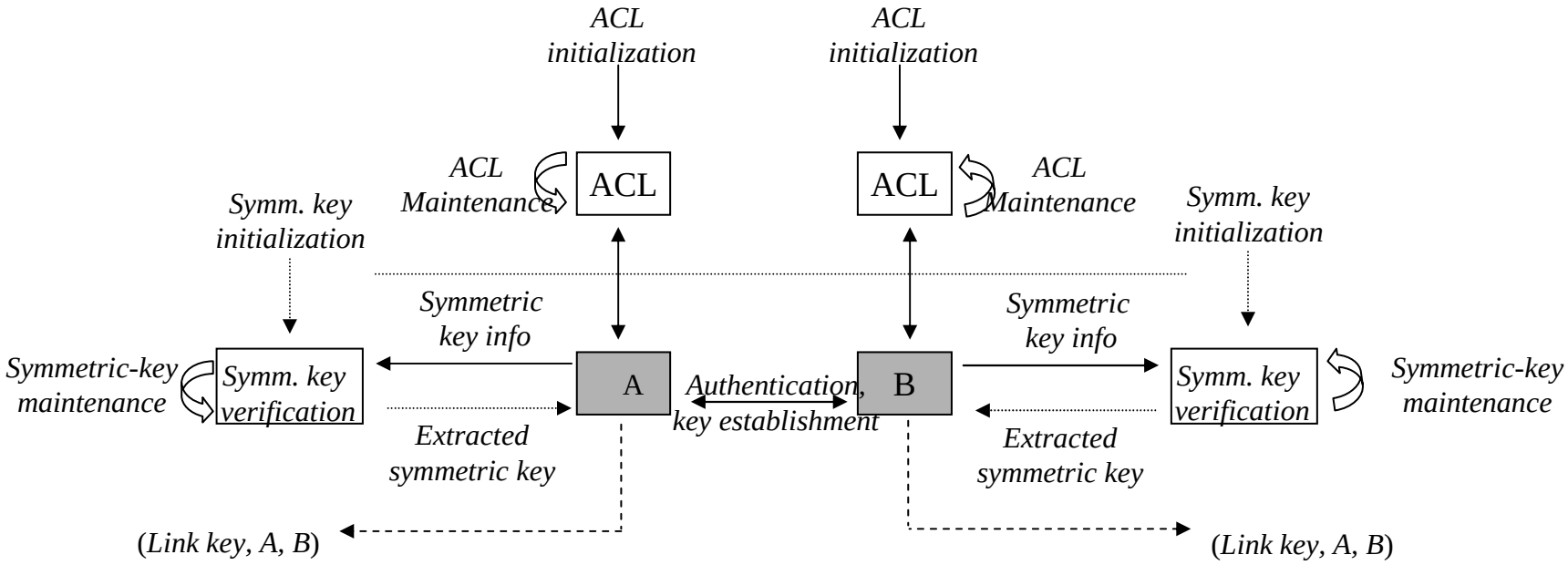
Security Architectural Framework: Authorization (2a) (public-key scenario)



Notes:

- The authentication protocol establishes a symmetric link key between the devices (since it is an authenticated key establishment protocol).
- Authenticated key establishment is based on a specific public-key protocol (e.g., ECC-based), using manual, explicit (e.g., X509), or implicit certificates.
- Certificate maintenance and ACL maintenance are not discussed any further here.

Security Architectural Framework: Authorization (2b) (symmetric-key scenario)



Notes:

- The authentication protocol establishes a symmetric link key between the devices (since it is an authenticated key establishment protocol).
- Authenticated key establishment is based on a specific symmetric-key protocol, using pre-shared secret keys.
- Symmetric-key maintenance and ACL maintenance are not discussed any further here.

Security Architectural Framework: Key transport (1)

Key transport is based on each of the following:

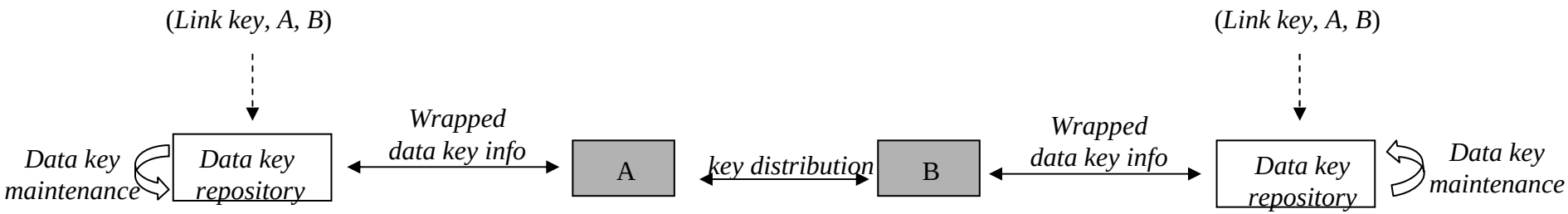
(1) Availability of a shared link key with the recipient;

(2) Evidence whether one wants to communicate with this explicitly identified device.

Cryptographic mechanisms:

1. Symmetric-key *key transport* mechanism. Secure transfer of data key from one device to other(s), based on link key(s) between sender and recipient(s).

Security Architectural Framework: Key transport (2)



Notes:

- Authenticated key transport may be based on the data protection mode that yields both confidentiality and authenticity.
- Key transport must include authentic info on, e.g., the key originator, the distribution group (key-sharing parties), and the version of the key.
  (The string *Key Id:=*(*Key originator* || *KeySeqNo*) seems to be a good choice.)
- Key storage and key update mechanisms are not discussed any further here.

Security Architectural Framework: Data transport (1)

Data transport is based on each of the following:

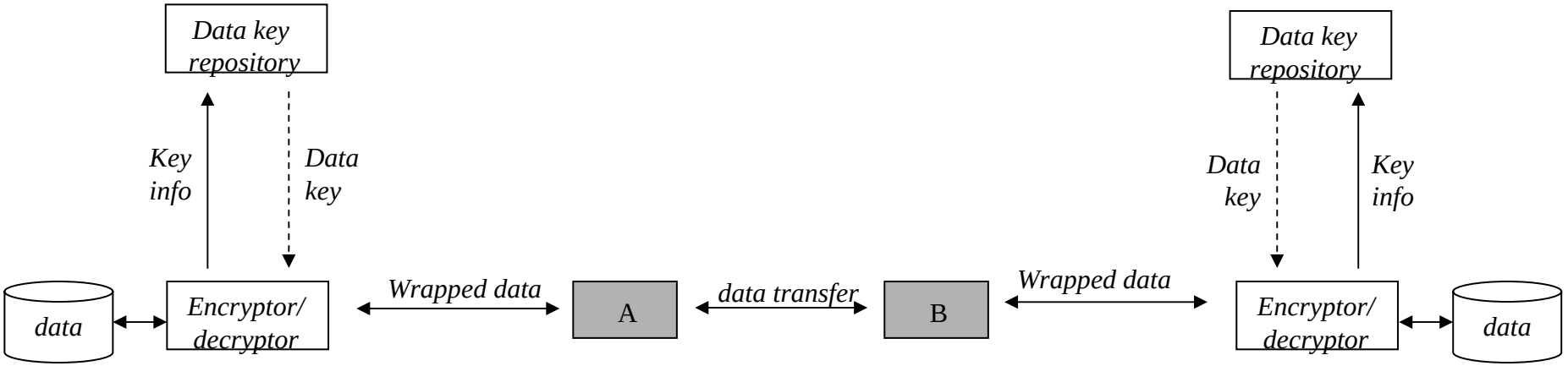
(1) Availability of a shared data key with the recipient(s);

(2) Evidence whether one wants to communicate with this explicitly identified device.

Cryptographic mechanisms:

1. Data transfer mechanism(s). Secure and/or authentic data transfer between devices that share the data key (confidentiality/data integrity/authenticity).

Security Architectural Framework: Data transport (2)



Notes:

- Data transport may be based on any negotiated data protection mode that yields a combination of confidentiality and authenticity (see next two slides for taxonomy)
- Data transport must include authentic info on, e.g., the used data key(s), the sender, and a message sequence number (to prevent replay attacks).

Data Transport – Protection of Messages (1)

Unsecured transport:

1. Initial set-up: none
2. Message: $A \rightarrow B$: *msg*
3. Security services: none

$Encrypt_{key}$: encryption function
$HMAC_{key}$: keyed hash function

Secure transport:

1. Initial set-up: Establishment of shared data encryption key *key* between *A* and *B*
2. Message: $A \rightarrow B$: $Encrypt_{key}$ *(msg)*
3. Security services: Secure transfer of message *msg*

Authentic transport:

1. Initial set-up: Establishment of shared data integrity key *key* between *A* and *B*
2. Message: $A \rightarrow B$: *msg*, $HMAC_{key}$ *(msg)*
3. Security services: Authentic transfer of message *msg*

Secure and authentic transport:

1. Initial set-up: Establishment of shared encryption key key_1 between *A* and *B*
   Establishment of shared data integrity key key_2 between *A* and *B*
2. Message: $A \rightarrow B$: $msg_1 \,||\, Encrypt_{key1}\,(msg_2 \,||\, HMAC_{key2}\,(msg_1 \,||\, msg_2))$
3. Security services: Authentic transfer of messages msg_1 and msg_2
   Secure transfer of message msg_2

Data Transport – Protection of Messages (2)

Assumptions on capabilities:

1. Sender *A* should be able to encrypt messages and to compute keyed hash values hereover.
2. Recipient *B* should be able to decrypt messages and to verify keyed hash values.

Header info may be bound to message to be authenticated if needed, e.g.,

1. Algorithm Ids: specifies the particular cryptographic primitives used;
2. Key Ids: prevents use of improper data keys;
3. Sequence No.: prevents undetected reordering (or replay) of message frames;
4. Message length: prevents misalignment in decryption and verification process.

The Security Architecture at Work

–

Operational Description

Outline

- Informational Elements
- Examples – Distributed vs. Centralized Model
- Separation of Concerns

Operational Description – Informational Elements (1)

Informational elements (provided by device itself)

(1) *Global device Id.*
Each device has own *static* global device Id (IEEE MAC address).

(2) *Access control list (ACL) (if desired).*
Each device has own set of devices it may wish to establish a secure peer-to-peer link key with.

(3) *TrustSet (in dynamic-trust scenario).*
Each device has own set of devices it trusts to assume the role of security manager.

(4) *Public key (in public-key scenario).*
Each device has its own public/private key pair (P_A, S_A).
{The public key P_A need not to be stored on the device itself.}

(5) *CA-Set (in dynamic-trusted party scenario).*
Each device has own set of devices it trusts to assume the role of trusted third party.

Informational elements (provided by other devices)

(1) *Global device Ids.*
Each device may have access to *static* global device Ids (IEEE MAC Address) of other devices.

(2) *Public keys (in public-key scenario)*
Each device may have access to public keys of other devices.

(3) *Key-sharing sets (in dynamic-trust scenario)*
Each device may have access to list of devices it shares a key with; provided by its Security Manager(s).

GroupList:=(Group Id, {global device Id}).

Operational Description – Informational Elements (2)

Informational elements (provided by Network in which device partakes)

(1) *Local device Id.*

Each device has *dynamic* local device Id (shorthand Id of, e.g., 2 octets); assigned by Network Coordinator, upon admittance to piconet.

(2) *Network Id.*

Each device has access to the Network Id (e.g., 2-octet dynamic Id); assigned by Network Coordinator (controller).

(3) *Network DeviceList.*

Each device has list of devices admitted to the Network; obtained from Network Coordinator.

DeviceList:={(local device Id, global device Id)}.

Informational elements (provided by trusted party of device)

(1) *Public-key certificates (in public-key scenario)*

Each device may have access to certificates; provided by a trusted third party T:

(a) *Digital certificates.*

Certificate $Cert_T(Id_B, P_B)$, together with signature verification key P_T.

{Two choices: ordinary certificate, implicit certificate.}

(b) *Manual certificates.*

Certificate $User_T(Id_B, P_B)$, together with info on way binding was established.

{pushing button, pre-installed, etc.}.

The Network at Work – Examples (1)

DeviceList :={A,B,C,D,E,F,G,H}
TrustSet(A):={A,B,C}

Groups in which A participates:

	A	B	C	D	E	F	G	H		
Group1'	x	x	x						Key source: C	encryption/decryption
Group2'	x			x					Key Source: D	decryption

Fig 1. Group structures as seen by A

Key Usage Rules:
(1) A accepts all keys transferred to him by others, for decryption purposes:
(2) A only accepts keys transferred to him by devices DEV ∈ TrustSet(A), for encryption purposes

Consequences:
(1) A uses Group1' group key for encryption/decryption; Group2' group key for decryption only.
(2) If A wants to communicate to Group2' members, he should generate a new group key and distribute these to the members of Group2': $\{E_D(key) \| E_{key}(msg)\}$

	A	B	C	D	E	F	G	H			
Group1	x	x	x						Key source: C	encryption/decryption	
Group2	x			x		$			Key Source: D	decryption	$: hidden node
Group3	x			x					Key source: A	encryption/decryption	('fly on the wall')

Fig 2. Group structures, as actually realized

The Network at Work – Examples (2)

DeviceList :={A,B,C,D,E,F,G,H}
TrustSet(A):=Universe (since A is an altruistic device) {Centralized model}
TrustSet(D):={D} (since D is an egocentric device) {Decentralized model}

Groups in which A participates:

	A	B	C	D	E	F	G	H		A	D
Group1'	x	x	x						Key source: C	encryption/decryption	
Group2'	x			x			x		Key Source: G	encryption/decryption	decrypt
Group3'				x	x				Key Source: E		decrypt

Fig 1. Group structures as seen by A and D

Consequences:
(1) A uses all group keys for encryption/decryption (since A is an altruistic device)
(2) D uses group keys for decryption purposes only (since B did not generate these himself)

	A	B	C	D	E	F	G	H		A	D
Group1	x	x	x						Key source: C	encryption/decryption	
Group2	x			x		$	x		Key Source: G	wrong view of group!	does not matter
Group3'				x	x				Key Source: E		does not matter

Fig 2. Group structures, as actually realized

$: hidden node ('fly on the wall')

The Network at Work – Separation of Concerns

Distributed Security Model

(1) *Admission to the Network.*

Network Coordinator regulates access of device to the network, based on

- proper (static) device Id;
- other info (e.g., from Access Control List).

(2) *Access to actual information*.

Security manager regulates access of group of devices to information, based on

- proper (static) device Id;
- other info (e.g., from Access Control List).

User scenario ('Starbucks'):

1. Admission to the Network based on charging airtime/bandwidth fee (similar to that for cell-phones).
2. Admission to information based on charging for content:
   a. From fixed Network Coordinator in ceiling:
      - multicast to subscribing devices only;
      - logical separation of content in different subscription packages.
   b. From device to device(s):
      - up to local devices.

NOTE: Separation of the role of Network Coordinator and that of Security Manager allows charging models that differentiate between airtime/bandwidth cost & content/subscription cost. These charging models might be operated by different entities.

Similar user scenarios: Network in fitness club, movie theatre, casino

Cryptographic Protocols
for
Implementing
the
Distributed Security Architecture
-
(for illustrational purposes)

Outline

- Public-Key Authenticated Key Agreement
- Symmetric-Key Authenticated Key Agreement
- Mutual Entity Authentication

Cryptographic Protocols

The following protocol options are provided:

- **Key agreement protocols:**

Two devices A and B derive a shared link key (key agreement) and show that these have computed correctly (key confirmation) in each of the following scenarios:

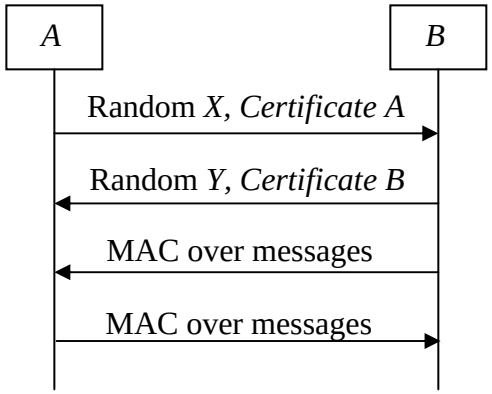
(1) Public-key based: Both devices do have (access to) a certificate of their public key, issued by a common trusted party (certificate authority).

(2) Symmetric-key based: Both devices do share a secret (master) key beforehand.

- **Entity authentication protocol:**

Two devices A and B derive evidence that the other device has access to a shared secret and is actively involved in the execution of the protocol (heartbeat function).

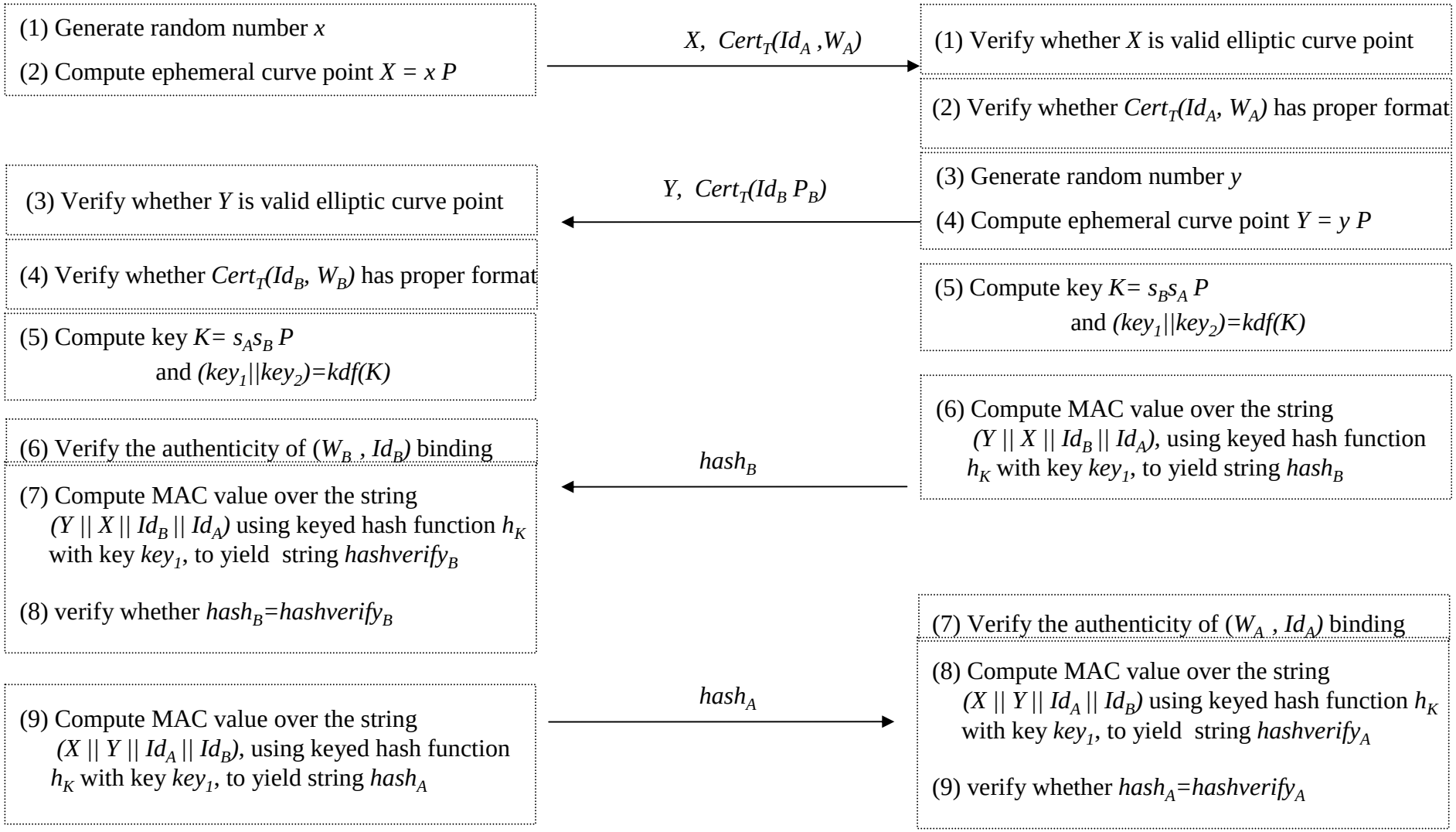
ECMQV Public-Key Authenticated Key Agreement Protocol (1)



Note: Certificate of the static public keys do not need to be communicated, if pre-established between parties. This does, however, require storage of status information.

- *Key contributions.* Each party randomly generates a short-term (ephemeral) public key pair and communicates this ephemeral public key to the other party (but not the private key).
- *Key establishment.* Each party computes the shared key based on the static and ephemeral elliptic curve points it received from the other party and based on the static and ephemeral private keys it generated itself. Due to the properties of elliptic curve, either party indeed arrives at the same shared key.
- *Key authentication.* Each party verifies the authenticity of the long-term static key of the other party, to obtain evidence that the only party that may be capable of computing the shared key is, indeed, its perceived communicating party.
- *Key confirmation.* Each party communicates a message authentication check value over the strings communicated by the other party, to prove possession of the shared key to the other party. This confirms to each party the true identity of the other party and proofs that that party successfully computed the shared key.

ECMQV Public-Key Authenticated Key Agreement Protocol (2)



ECMQV Public-Key Authenticated Key Agreement Protocol (3)

Initial Set-up

- Publication of system-wide parameters
- Publication of elliptic curve domain parameters
- Publication of keyed hash function h_k used
- Publication of un-keyed hash function h used
- Distribution of authentic long-term public keys W_A and W_B

Constraints

- X and Y shall be generated at random (ephemeral elliptic curve points)
- w_A and w_B private to Party A, resp. Party B
- w_A and w_B valid during execution of the protocol

(Note: the (w_A, W_A) and (w_B, W_B) are public key pairs of A, resp. B)

Security Services

- Key agreement between A and B on the shared key $K=K_{AB}$
- Mutual entity authentication of A and B
- Mutual implicit key authentication between A and B
- Mutual key confirmation between A and B
- Perfect forward secrecy; unknown key-share resilience, known-key security
- No unilateral key control by either party

ECMQV Public-Key Authenticated Key Agreement Protocol (4)

• **Implicit Certificates**

Public key derived from publicly available information, i.e.,

- Reconstruction data D_A of party A involved;
- Identity Id_A of the device A;
- Authentic public key W_T of trusted party (who issued implicit certificate).

Formula: $W_A = g(Id_A, D_A, W_T) = rD_A + W_T$, where $r = hash(D_A || Id_A || Id_T)$

• **ECMQV Key Agreement**

$$K = s_A s_B P$$
$$= (x + w_A \cdot map(X))(y + w_B \cdot map(Y))P$$
$$= (x + w_A \cdot map(X))(Y + map(Y)W_B)$$
$$= (y + w_B \cdot map(Y))(X + map(X)W_A)$$

• **System-wide parameters**

- P is a fixed point on the shared elliptic curve E
- *map* converts an elliptic curve point to an integer

ECMQV Public-Key Authenticated Key Agreement Protocol (5)

- **ECMQV Key Agreement**

$$K = s_A s_B P$$
$$= (x + w_A \cdot map(X))(y + w_B \cdot map(Y))P$$
$$= (x + w_A \cdot map(X))(Y + map(Y)W_B)$$
$$= (y + w_B \cdot map(Y))(X + map(X)W_A)$$
$$= \alpha X + \beta W_A$$

- **Implicit Certificate Processing**

$W_A = g(Id_A, D_A, W_T) = rD_A + W_T$, where $r = hash(D_A \,||\, Id_A \,||\, W_T)$

- **ECMQV Key Agreement using Implicit Certificates**

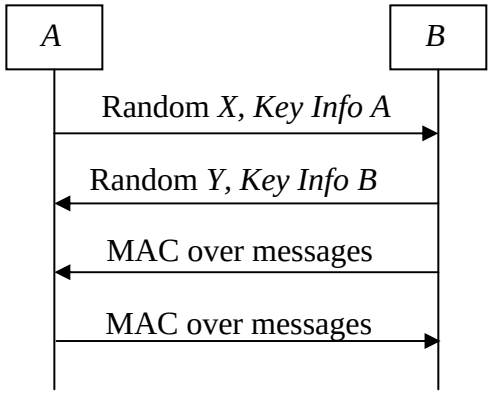
$$K = s_A s_B P$$
$$= \alpha X + \beta W_A = \alpha X + \beta(rD_A + W_T) = \alpha X + \gamma D_A + \delta W_T$$

Computational remarks:

Speed-up of computation K and W_A using, e.g., Shamir's trick (or Solinas's work)

Result: Certificate processing *virtually free* if computations of K and W_A combined

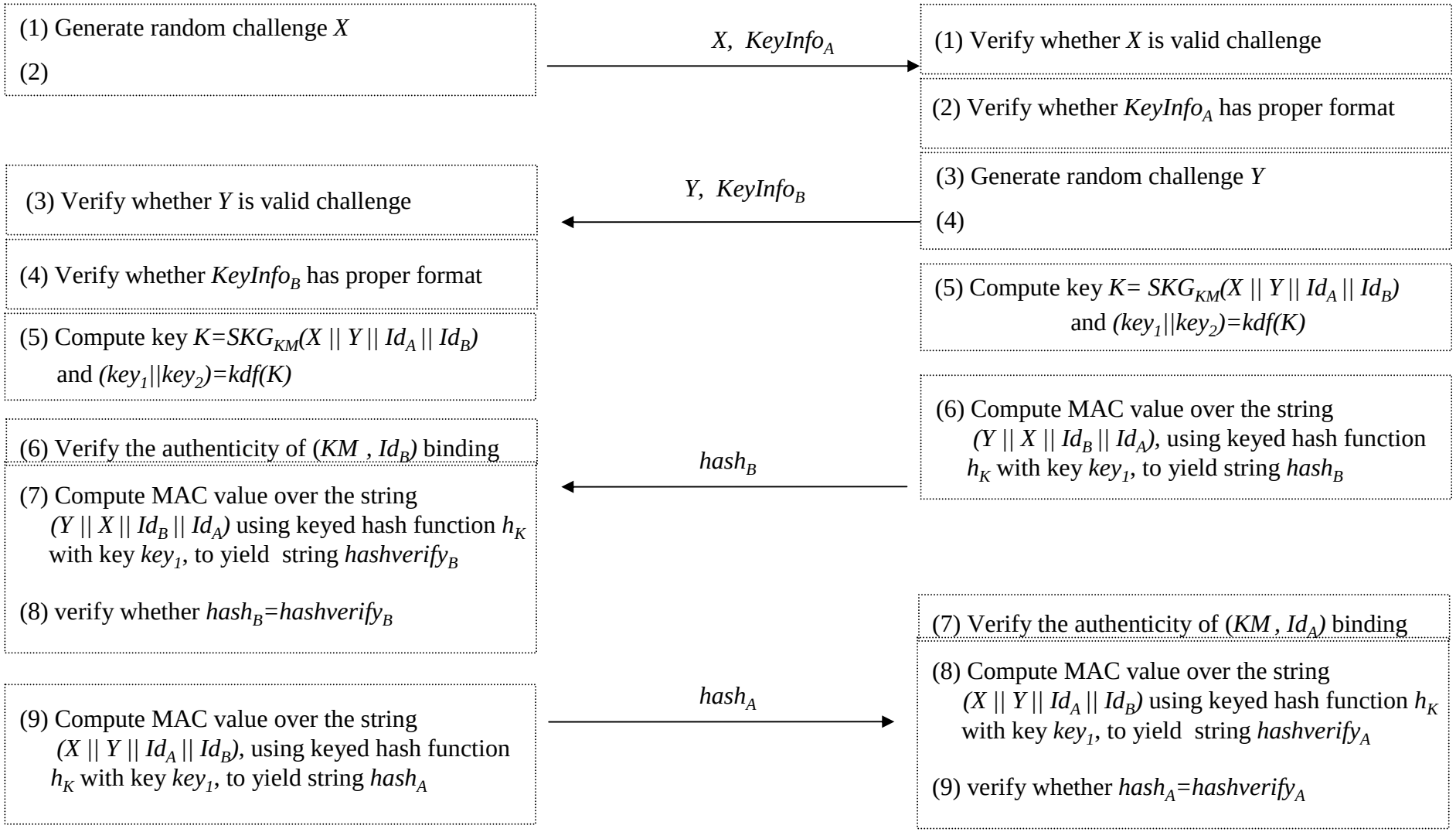
Symmetric-Key Authenticated Key Agreement Protocol (1)



Note: Key Info of the pre-shared keys does not need to be communicated, if pre-established between parties. This does, however, require storage of status information.

- *Key contributions.* Each party randomly generates a random bit string and communicates this random challenge to the other party.
- *Key establishment.* Each party computes the shared key based on the random challenges generated and received and based on their respective identities, and their shared pre-established key. Due to the properties of the secret key generator, either party indeed arrives at the same shared key.
- *Key authentication.* Each party verifies the authenticity pre-established key allegedly shared with the other party, to obtain evidence that the only party that may be capable of computing the shared key is, indeed, its perceived communicating party.
- *Key confirmation.* Each party communicates a message authentication check value over the strings communicated by the other party, to prove possession of the shared key to the other party. This confirms to each party the true identity of the other party and proofs that that party successfully computed the shared key.

Symmetric-Key Authenticated Key Agreement Protocol (2)



Symmetric-Key Authenticated Key Agreement Protocol (3)

Initial Set-up

- Publication of system-wide parameters
- Publication of challenge domain parameters
- Publication of keyed hash function h_k used
- Publication of un-keyed hash function *h* used

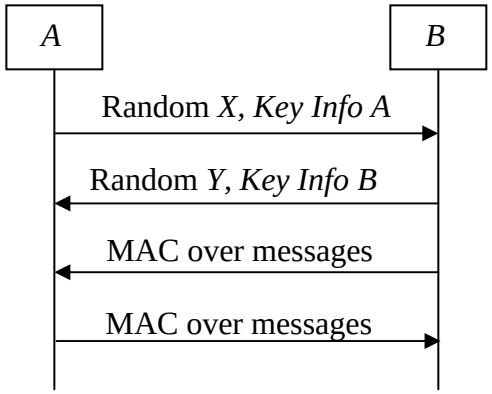
Constraints

- *X* and *Y* shall be generated at random (random challenges)
- *KM* private to Party A, resp. Party B

Security Services

- Key agreement between A and B on the shared key $K=K_{AB}$
- Mutual entity authentication of A and B
- Mutual implicit key authentication between A and B, *provided that* both parties have a non-cryptographic way of establishing the identity of the other party (Example: 'pushing buttons', where human operator controls who is executing protocol. The identities are then only known implicitly, since the human operator knows the devices he wants to securely connect to one another.)
- Mutual key confirmation between A and B
- No perfect forward secrecy (key compromises compromises all past and future keys)
- No unilateral key control by either party

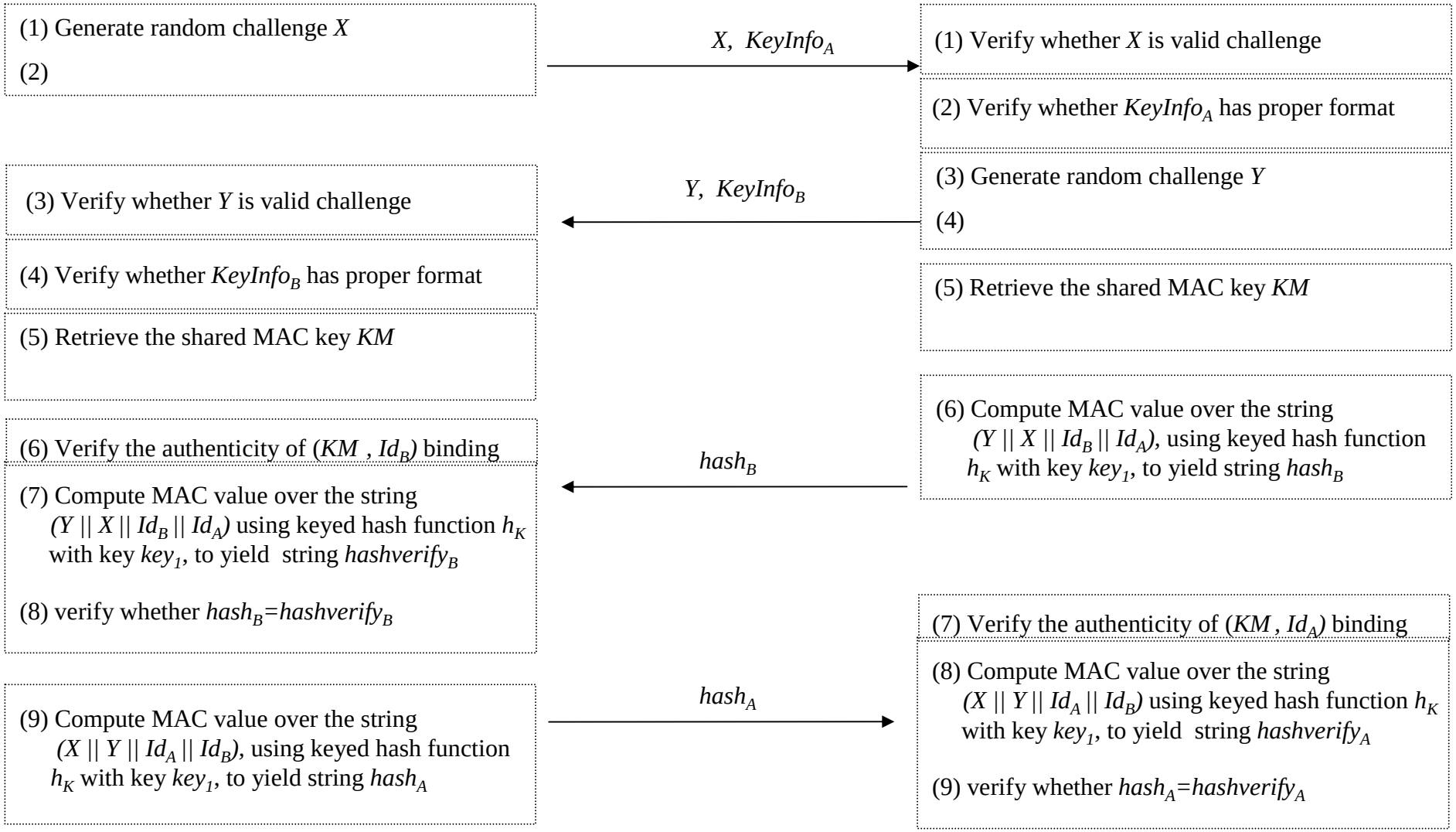
Symmetric-Key Entity Authentication Protocol (1)



Note: Key Info of the pre-shared keys does not need to be communicated, if pre-established between parties. This does, however, require storage of status information.

- *Challenge contributions.* Each party randomly generates a random bit string and communicates this random challenge to the other party..
- *Key authentication.* Each party verifies the authenticity pre-established key allegedly shared with the other party, to obtain evidence that the only party that may be capable of computing the shared key is, indeed, its perceived communicating party.
- *Key confirmation.* Each party communicates a message authentication check value over the strings communicated by the other party, to prove possession of the shared key to the other party. This confirms to each party the true identity of the other party and proofs that that party successfully computed the shared key.

Symmetric-Key Entity Authentication Protocol (2)



Symmetric-Key Entity Authentication Protocol (3)

Initial Set-up

- Publication of system-wide parameters
- Publication of challenge domain parameters
- Publication of keyed hash function h_k used
- Publication of un-keyed hash function *h* used

Constraints

- *X* and *Y* shall be generated at random (random challenges)
- *KM* private to Party A, resp. Party B

Security Services

- Mutual entity authentication of A and B, *provided that* both parties have a non-cryptographic way of establishing the identity of the other party (Example: 'pushing buttons', where human operator controls who is executing protocol. The identities are then only known implicitly, since the human operator knows the devices he wants to securely connect to one another.)

Combined Key Agreement and Entity Authentication Protocols

Implementation issues

- Efficient implementation possible
- No usage constraints
- Channel should be simplex channel both ways

Flexibility

- No restrictions between cryptographic building blocks (in particular, good fit for ECC)
- Survivability, since no status information maintained

Alternative uses using same implementation (!)

- Full ECMQV Authenticated Key Agreement Protocol
- 1-Pass ECMQV Public Key Authenticated Key Agreement Protocol
- Mutual Symmetric-Key Authenticated Key Agreement Protocol
- 1-Pass Symmetric-Key Authenticated Key Agreement Protocol
- Mutual Symmetric-Key Entity Authentication Protocol
- Unilateral Symmetric-Key Entity Authentication Protocol

Note: This covers key establishment, heartbeat function, and, e.g, secure code download!

Performance ECMQV Authenticated Key Agreement Protocol (1)

- **Communicated messages**

Step 1 (A → B): $X \,||\, Cert_T(Id_A, W_A)$

Step 2 (A ← B): $Y \,||\, Cert_T(Id_B, W_B)$

Step 3 (A ← B): $hash_B$

Step 3 (A → B): $hash_A$

- **Communications bandwidth**

- *Binary NIST-curve K-283 (128-bit security)*

ECC Point: 37 bytes

Implicit Cert: 37+8+10 = 55 bytes

Hash string: 16 bytes

3 frames needed – no fragmentation; 4 frames most efficient

	Hash	RND	Cert	Total
Step 1/2:		37 +	55	= 92/74
Step 3/4:	16			= 16
			total bytes:	216/180

- *Binary NIST-curve K-163 (80-bit security)*

ECC Point: 22 bytes

Implicit Cert: 22 + 8 +10 = 40 bytes

Hash string: 10 bytes

3 frames needed – no fragmentation; 4 frames most efficient

	Hash	RND	Cert	Total
Step 1/2:		22 +	40	= 62/44
Step 3/4:	10			= 10
			total bytes:	144/108