IEEE 802.11 TGai
–
Some Notes and Thoughts on TGai Security Properties

Date: 2011-12-15

Authors:

Name	Company	Address	Phone	email
René Struik	Struik Security Consultancy	723 Carlaw Avenue Toronto ON Canada M4K 3K8	USA: +1 (415) 690-7363 Toronto: +1 (647) 867-5658 Skype: rstruik	rstruik.ext@gmail.com

Actors

Actors

A

Client ♦ This device may move in and out of networks (that may be alien to it) and may have little network management functionality on board. *Key words*: nomadic, promiscuous, constrained.

B

Access point ♦ This device may be more tied into a relatively stable infrastructure and may have more support for network management functionality or have reliable access hereto (e.g., via a back-end system). *Key words*: anchor, semi-stable connectivity, access portal.

KDC

Server ♦ This device provides stable infrastructure and network management support, either intra-domain or inter domain (thereby, offering homogeneous or even heterogeneous functionality). *Key words*: core function, high availability, human-operator support.

CA

CA ♦ This device vouches for trust credentials, usually in offline way. *Key words*: trust anchor.

Initiator/responder model

All peer-to-peer protocols are role-symmetrical (i.e., the role of initiator/responder roles are interchangeable). Protocols involving a third party assume communications with this third party to take place via the access point (since being the device more tied into infrastructure).

Cautionary NOTE − On Limitations of Cryptography

- Cryptographic techniques may provide logical assurances as to a device's identity, where and when communications originated, whom it was intended for, whom this can be read by, etc.
- Cryptographic techniques do, however, only provide *mechanical assurances* and can generally not substitute human *authorization* decision elements (unless the latter are not important, such as with random, ad-hoc networks).

Desired Protocol Properties

Security-Related

- Parties executing a security protocol should be explicitly aware of its security properties
- Compromise of keys or devices should have limited effect on security of other devices or services
- Attacks should not have a serious impact beyond the time interval/space during/in which these take place

Communication flows

- Security protocols should allow to be run locally, without third party involvement, if at all possible
- The number of message exchanges for a joining client device should be reduced

Computational cost

- Security protocols should not impose an undue computational burden, esp. on joining client devices (An exception here may arise, when recovering from an event seriously impacting availability of the network.)

Device capabilities

- Dependency on an accurate time-keeping mechanism should be reduced
- Computational/time latency trade-offs should be tweaked to benefit those of joining client, if possible
- Dependency on “homogeneous trust models” should be reduced, without jeopardizing security properties

Network Joining – Authentication vs. Authorization (1)

Actors

A

Client ♦ This device may move in and out of networks (that may be alien to it) and may have little network management functionality on board. *Key words*: nomadic, promiscuous, constrained.

B

Access point ♦ This device may be more tied into a relatively stable infrastructure and may have more support for network management functionality or have reliable access hereto (e.g., via a back-end system). *Key words*: anchor, semi-stable connectivity, access portal.

KDC

Server ♦ This device provides stable infrastructure and network management support, either intra-domain or inter domain (thereby, offering homogeneous or even heterogeneous functionality). *Key words*: core function, high availability, human-operator support.

CA

CA ♦ This device vouches for trust credentials, usually in offline way. *Key words*: trust anchor.

Protocols involving a third party assume communications with this third party to take place via the access point (since being the device more tied into infrastructure).

Device Enrolment Steps:

Device authentication. Client A and Access Point B authenticate each other and establish a shared key (so as to ensure on-going authenticated communications). *This may involve server KDC as third party.*

Authorization. Access Point B decides on whether/how to authorize device A (if denied, this may result in loss of bandwidth). *Authorization decision may be delegated to server KDC or other 3rd-party device.*

Configuration/Parameterization. Access Point B distributes configuration information to Client A, such as ♦ IP address assignment info; ♦ Bandwidth/usage constraints; ♦ Scheduling info (including on re-authentication policy details). *This may originate from other network devices, for which it acts as proxy*.

Network Joining – Authentication vs. Authorization (2)

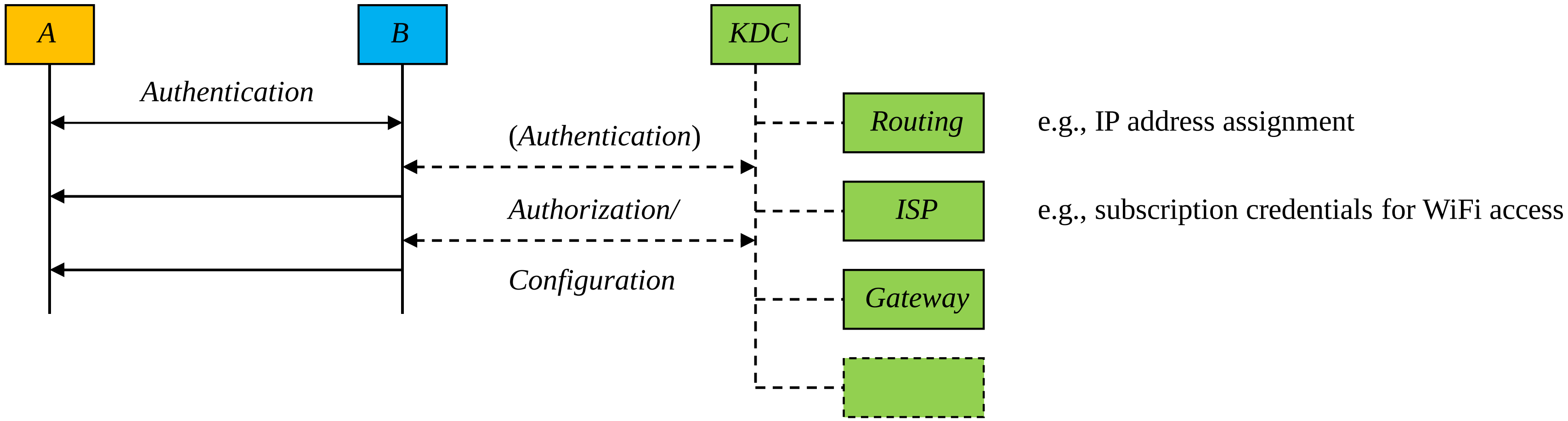
Device Enrolment Steps:

Device authentication. Client A and Access Point B authenticate each other and establish a shared key (so as to ensure on-going authenticated communications). *This may involve server KDC as third party.*

Authorization. Access Point B decides on whether/how to authorize device A (if denied, this may result in loss of bandwidth). *Authorization decision may be delegated to server KDC or other 3rd-party device.*

Configuration/Parameterization. Access Point B distributes configuration information to Client A, such as ♦ IP address assignment info; ♦ Bandwidth/usage constraints; ♦ Scheduling info (including on re-authentication policy details). *This may originate from other network devices, for which it acts as proxy.*

Sequential Enrolment vs. Combined Steps



Aggressive scheme: Initiate authorization/configuration processes as soon as (presumed) device identity becomes available (invisible to Client A). Access Point B can deny bandwidth if authorization negative.

Note: Communication of configuration info depends on secure channel with Client A.

Network Joining – *With* Authentication by Third Party

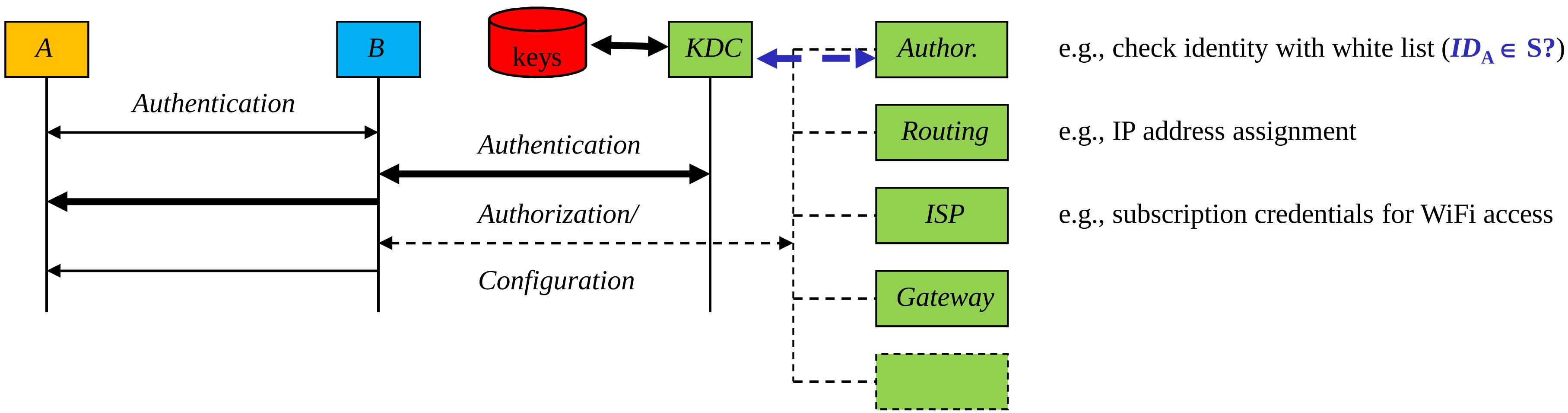
Device Enrolment Steps:

Device authentication. Client A and Access Point B authenticate each other and establish a shared key (so as to ensure on-going authenticated communications). *This may involve server KDC as third party.*

Authorization. Access Point B decides on whether/how to authorize device A (if denied, this may result in loss of bandwidth). *Authorization decision may be delegated to server KDC or other 3rd-party device.*

*Configuration/Parameterization.*Access Point B distributes configuration information to Client A, such as ♦ IP address assignment info; ♦ Bandwidth/usage constraints; ♦ Scheduling info (including on re-authentication policy details). *This may originate from other network devices, for which it acts as proxy.*

Sequential Enrolment vs. Combined Steps



<u>Aggressive scheme:</u> Initiate authorization/configuration processes as soon as (presumed) device identity becomes available (invisible to Client A). Access Point B can deny bandwidth if authorization negative.

<u>Note:</u> <u>Communication of configuration info depends on secure channel with Client A.</u>

Network Joining – *Only* Authorization by Third Party

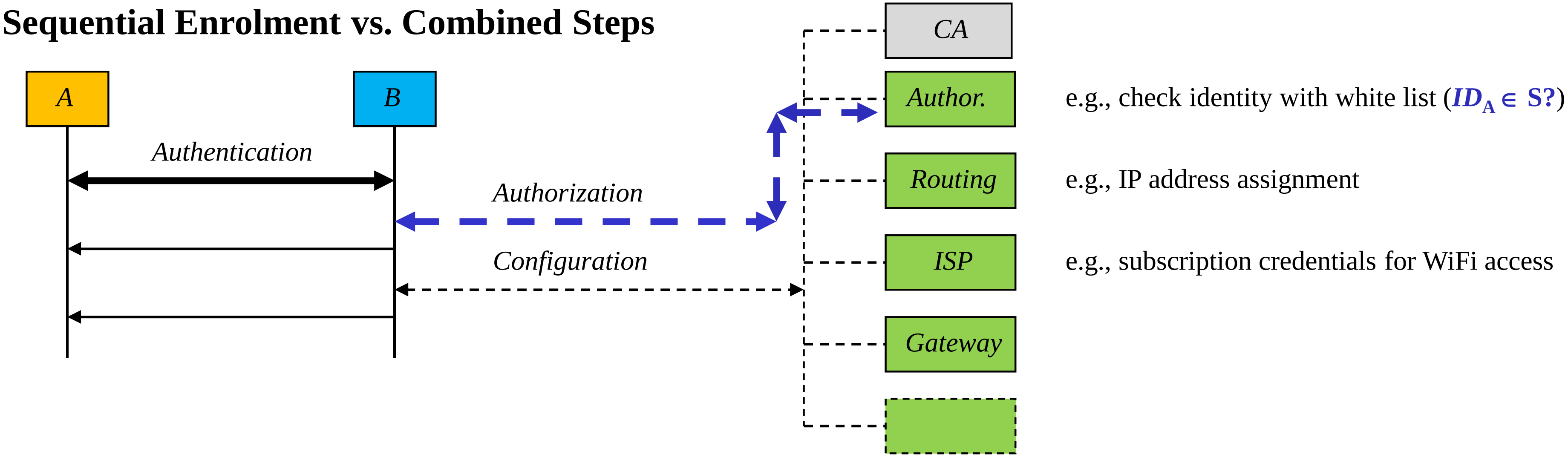
Device Enrolment Steps:

Device authentication. Client A and Access Point B authenticate each other and establish a shared key (so as to ensure on-going authenticated communications). *This may involve server KDC as third party.*

Authorization. Access Point B decides on whether/how to authorize device A (if denied, this may result in loss of bandwidth). *Authorization decision may be delegated to server KDC or other 3rd-party device.*

*Configuration/Parameterization.*Access Point B distributes configuration information to Client A, such as ♦ IP address assignment info; ♦ Bandwidth/usage constraints; ♦ Scheduling info (including on re-authentication policy details). *This may originate from other network devices, for which it acts as proxy.*

Sequential Enrolment vs. Combined Steps



Aggressive scheme: Initiate authorization/configuration processes as soon as (presumed) device identity becomes available (invisible to Client A). Access Point B can deny bandwidth if authorization negative.

Note: Communication of configuration info depends on secure channel with Client A.

Security Definitions

Key Establishment ♦ Protocol whereby a shared secret becomes available to two or more parties for subsequent cryptographic use

Key Transport ♦ Key establishment technique where one party creates/obtains the secret and securely transfers it to other(s)

Key Agreement ♦ Key establishment technique where the shared secret is derived based on information contributed by each of the parties involved, ideally so that no party can predetermine this secret value

Implicit Key Authentication♦ Assurance as to which specifically identified parties possibly *may* gain access to a specific key

Key Confirmation ♦ Assurance that second (possibly unknown) party has possession of a particular key

Explicit Key Authentication ♦ Combination of implicit key authentication and key confirmation

Unilateral Key Control ♦ Key establishment protocol whereby one party can influence the shared secret

Forward Secrecy ♦ Assurance that compromise of long-term keys does not compromise past session keys

Entity Authentication ♦ Assurance of active involvement of second explicitly identified party in protocol

Mutual vs. Unilateral ♦ Adjective indicating symmetry, resp. asymmetry, of assurances amongst parties

Identity Protection ♦ Assurance as to which specifically identified parties may gain access to identity info

Certificate ♦ Credential that vouches for authenticity of binding between a public key and other information, including the identity of the owner of the public key in question

Key Possession♦ Assurance that a specific (possibly unknown) party has possession of a particular key

<u>Esoteric properties:</u>

Unknown Key Share Resilience, Session Key Retrieval, Key Compromise Impersonation

Key Establishment Options

The following protocol options for key establishment are provided:

Symmetric-Key Key Agreement:

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

(a) Both devices do share a secret (master) key beforehand.
(b) Both devices do not share a secret key, but each shares a key with a mutually trusted third party.
(c) Both devices do not have certificates, but each shares a key with a mutually trusted third party.

Public-Key Key Agreement:

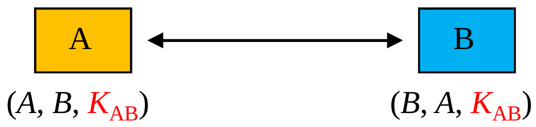
Two devices A and B derive a shared key (key agreement) and show that these have computed correctly (key confirmation) in each of the following scenarios:

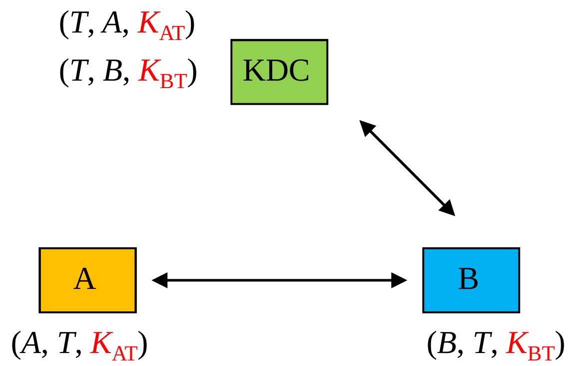
(a) Both devices do have (access to) a certificate of their public key, issued by a mutually trusted third party (certificate authority).
(b) Both devices do not have (access to) a certificate of their public key.
(c) Both devices do share a ***weak*** secret key.
(d) Both devices do have (access to) a certificate of their public key, but cannot verify each other’s certificate.

This taxonomy includes all “trust bootstrapping scenarios” that may result in cryptographic assurances.

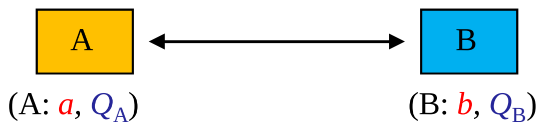
Peer-to-Peer, or with Involvement Third Party

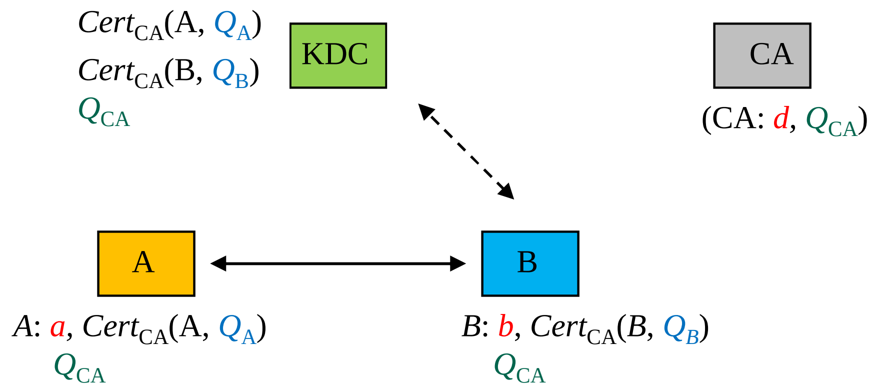
Symmetric-Key Key Agreement





Public-Key Key Agreement





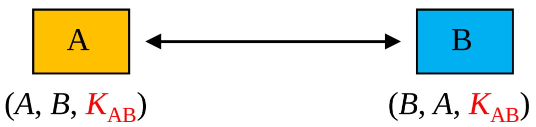
Symmetric-Key Key Agreement (1)

Symmetric-Key Key Agreement:

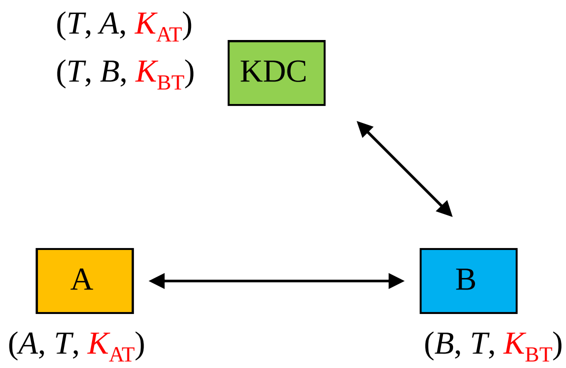
Two devices A and B derive a shared key (key agreement) and show that these have computed correctly (key confirmation) in each of the following scenarios:

(a) Both devices do share a secret (master) key beforehand.
(b) Both devices do not share a secret key, but each shares a key with a mutually trusted third party.

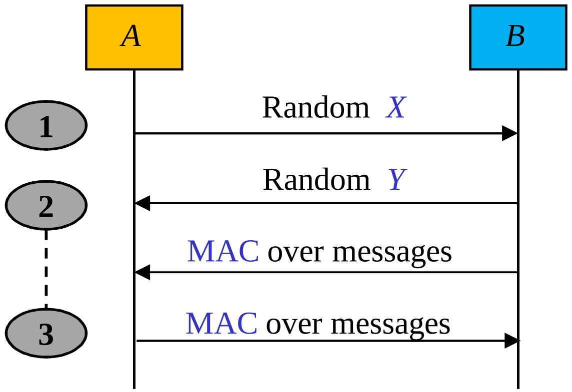
(a) Peer-to-Peer Key Establishment



(b), (c) Key Establishment with Inline Third Party



Symmetric-Key Key Agreement: (a) Peer-to-Peer (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 of the 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 Key Agreement: (a) Peer-to-Peer (2)

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

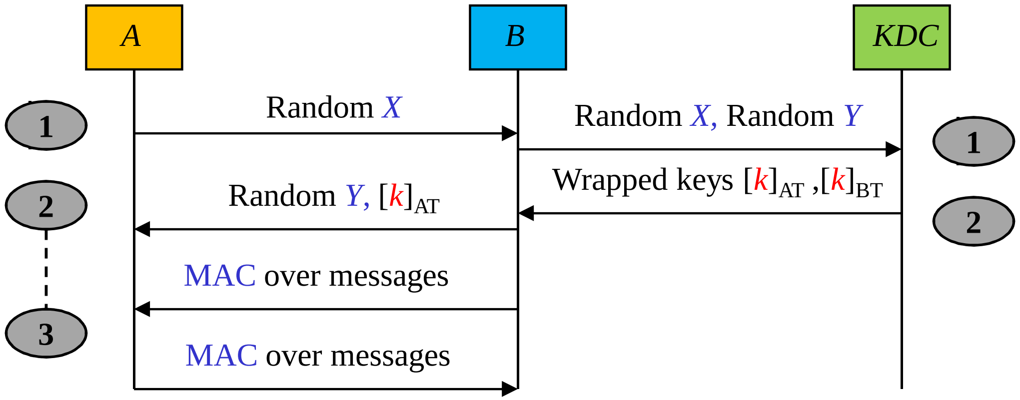
Constraints

- X and Y shall be generated at random (random challenges)
- K_{AB} private to Parties A and B

Security Services

- Key agreement between A and B on the shared key $K=h(K_{AB}, X, Y, A, B)$
- 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 compromise compromises all past and future keys)
- No unilateral key control by either party

Symmetric-Key Key Agreement: (b) Inline 3rd Party (1)



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 a session key distributed by the third party. 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 of the 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 Key Agreement: (b) Inline 3rd Party (2)

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

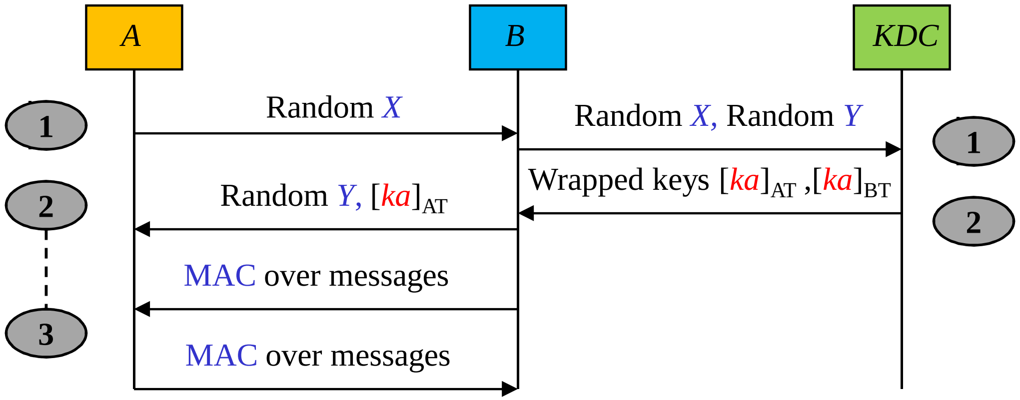
Constraints

- X and Y shall be generated at random (random challenges)
- K_{AT} private to Parties A and KDC; K_{BT} private to Parties B and KDC
- k private to Parties A, B, and KDC.

Security Services

- Key transport from KDC to A and B of the key k, based on key wrap using K_{AT}, resp. K_{BT}
- Key agreement between A and B on the shared key $K = h(k, X, Y, A, B)$
- 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 compromise compromises all past and future keys)
- No unilateral key control by either party A and B, irrespective of key control by KDC

Public-Key Key Agreement: (c) with Inline 3rd Party (1)



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 ephemeral elliptic curve point it received from the other party and based on the ephemeral private key 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 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.

Public-Key Key Agreement: (c) with Inline 3rd Party (2)

Initial Set-up

- Publication of system-wide parameters
- Publication of elliptic curve domain parameters
- Publication of key derivation function *kdf* used

Constraints

- X and Y shall be generated at random (ephemeral elliptic curve points)
- Short-term private keys x and y private to Party A, resp. Party B and *valid during execution of protocol*
- K_{AT} private to Parties A and KDC; K_{BT} private to Parties B and KDC and *valid during execution of protocol*
- *ka* private to Parties A, B, and KDC *during execution of the protocol*

Note: (x, X) and (y, Y) are short-term public key pairs of A, resp. B

Security Services

- Key transport from KDC to A and B of the key *ka*, based on key wrap using K_{AT}, resp. K_{BT}
- Key agreement between A and B on the shared key $K=KeyMap(x, Y)=KeyMap(y, X)$ and $K_s=kdf(K,ka,A,B)$
- 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 (and rely on the KDC to disclose *ka* to A and B only)
- Mutual key confirmation between A and B
- Perfect forward secrecy
- No unilateral key control by either party, irrespective of key control by KDC
- Esoteric properties: unknown key-share resilience

Public-Key Key Agreement (1)

Public-Key Key Agreement:

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

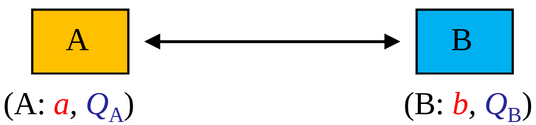
(a) Both devices do have (access to) a certificate of their public key, issued by a mutually trusted third party (certificate authority).

(b) Both devices do not have (access to) a certificate of their public key.

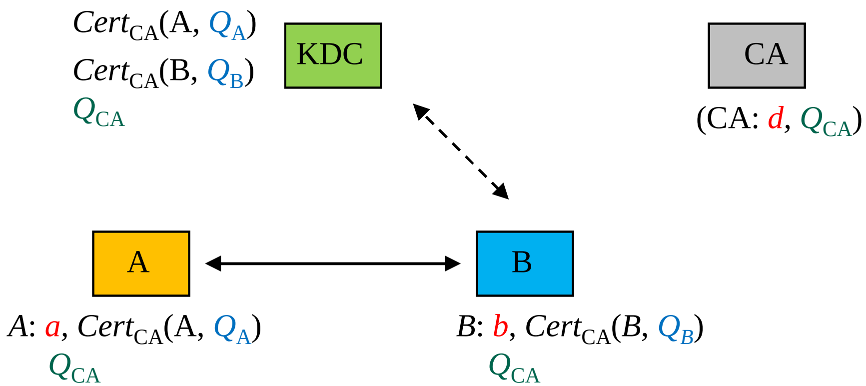
(c) Both devices do have access do share a ***weak*** secret key.

(d) Both devices do have (access to) a certificate of their public key, but cannot verify each others certificate.

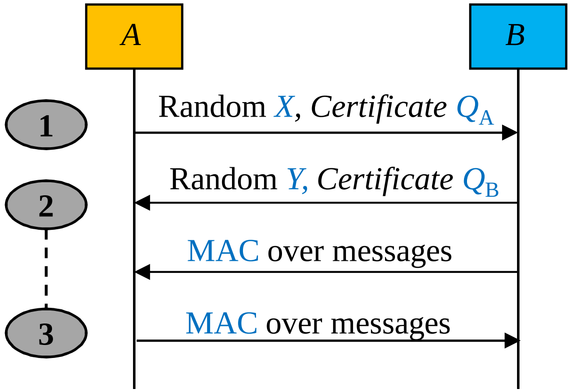
(a), (b), (c) Peer-to-Peer Key Establishment



(d) Key Establishment with Online Third Party



Public-Key Key Agreement: (a) with Certificates (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.

Public-Key Key Agreement: (a) with Certificates (2)

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 Q_A and Q_B, using certificates

Constraints

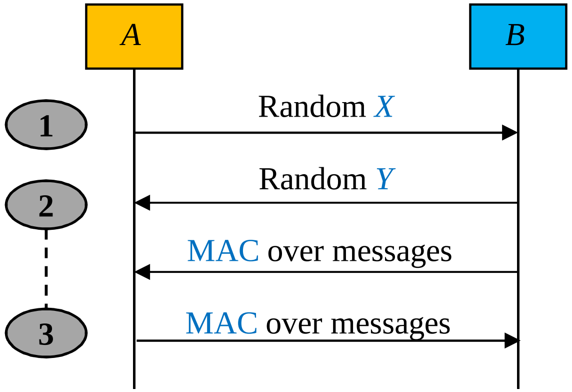
- X and Y shall be generated at random (ephemeral elliptic curve points)
- Long-term private keys d_A and d_B private to Party A, resp. Party B, and *valid during execution of protocol*
- Short-term private keys x and y private to Party A, resp. Party B and *valid during execution of protocol*
- Each party shall have access to the public key Q_{CA} used to certify the other party's long-term key

Note: (d_A, Q_A), (x, X) and (d_B, Q_B) , (y, Y) are long-term and short-term public key pairs of A, resp. B

Security Services

- Key agreement between A and B on the shared key $K=KeyMap(d_A, x, Q_B, Y)=KeyMap(d_B, y, Q_A, X)$
- 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
- No unilateral key control by either party
- Esoteric properties: unknown key-share resilience, session key retrieval resilience

Public-Key Key Agreement: (b) without Certificates (1)



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 ephemeral elliptic curve point it received from the other party and based on the ephemeral private key 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 short-term key of the other party via non-cryptographic means, 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.

Public-Key Key Agreement: (b) without Certificates (2)

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

Constraints

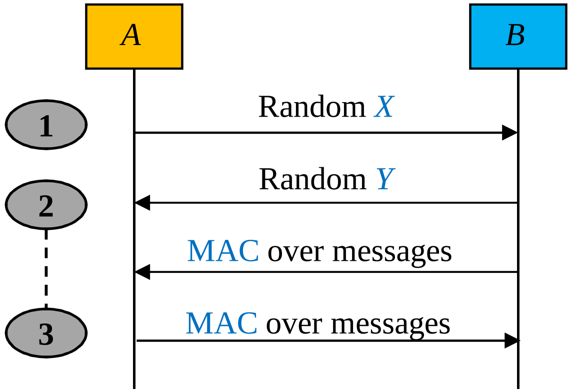
- X and Y shall be generated at random (ephemeral elliptic curve points)
- Short-term private keys x and y private to Party A, resp. Party B and *valid during the system's lifetime*

Note: (x, X) and (y, Y) are short-term public key pairs of A, resp. B

Security Services

- Key agreement between A and B on the shared key $K=KeyMap(x, Y)=KeyMap(y, X)$
- 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
- Perfect forward secrecy
- No unilateral key control by either party
- Esoteric properties: unknown key-share resilience

Public-Key Key Agreement: (c) with Shared Password (1)



Key contributions. Each party randomly generates a short-term (ephemeral) public key pair using shared password to determine some of elliptic curve domain parameters 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 ephemeral elliptic curve point it received from the other party and based on the ephemeral private key it generated itself. Due to properties of elliptic curve and shared domain parameters, either party indeed arrives at the same shared key.

Key authentication. Each party verifies the authenticity of the password 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.

Public-Key Key Agreement: (c) with Shared Password (2)

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

Constraints

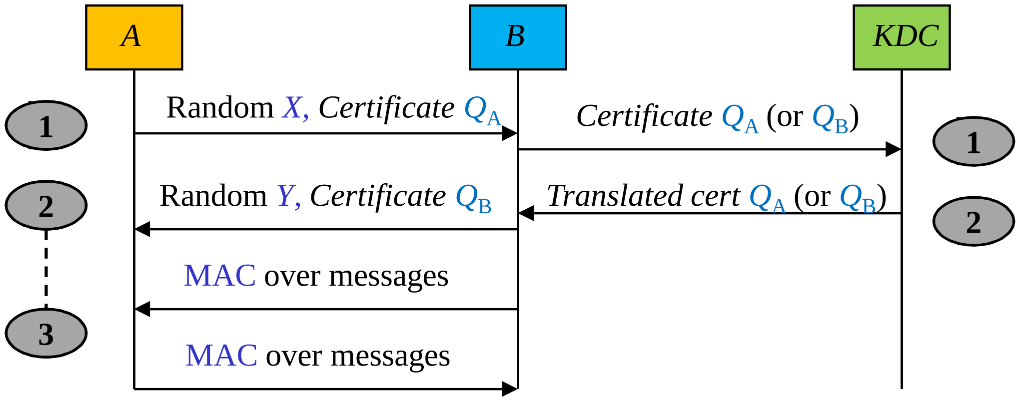
- X and Y shall be generated at random (ephemeral elliptic curve points)
- Short-term private keys x and y private to Party A, resp. Party B and *valid during the system's lifetime*

Note: (x, X) and (y, Y) are short-term public key pairs of A, resp. B

Security Services

- Key agreement between A and B on the shared key K=*KeyMap*(x, Y)=*KeyMap*(y, X)
- 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 party one has shared the password with (e.g., using NFC or key pad). 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
- Perfect forward secrecy
- No unilateral key control by either party
- Esoteric properties: unknown key-share resilience

Public-Key Key Agreement: (d) with Inline 3rd Party (1)



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.

Public-Key Key Agreement: (d) with Inline 3rd Party (2)

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 Q_A and Q_B, using certificates

Constraints

- X and Y shall be generated at random (ephemeral elliptic curve points)
- Long-term private keys d_A and d_B private to Party A, resp. Party B, and *valid during execution of protocol*
- Short-term private keys x and y private to Party A, resp. Party B and *valid during execution of protocol*
- Each party *does not need* access to the public key Q_{CA} used to certify the other party's long-term key

Note: (d_A, Q_A), (x, X) and (d_B, Q_B) , (y, Y) are long-term and short-term public key pairs of A, resp. B

Security Services

- Key agreement between A and B on the shared key $K = KeyMap(d_A, x, Q_B, Y) = KeyMap(d_B, y, Q_A, X)$
- 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
- No unilateral key control by either party
- Esoteric properties: unknown key-share resilience, session key retrieval resilience

Network Joining –
Peer-to-Peer vs. Third-Party-Assisted Authentication,
Authorization, and Configuration
(with IEEE 802.11 Architecture)

Network Joining – *with* Authentication by Third Party

Device Enrolment Steps:

Device authentication. Client A and Access Point B authenticate each other and establis[…] as to ensure on-going authenticated communications). *This may involve server KDC*

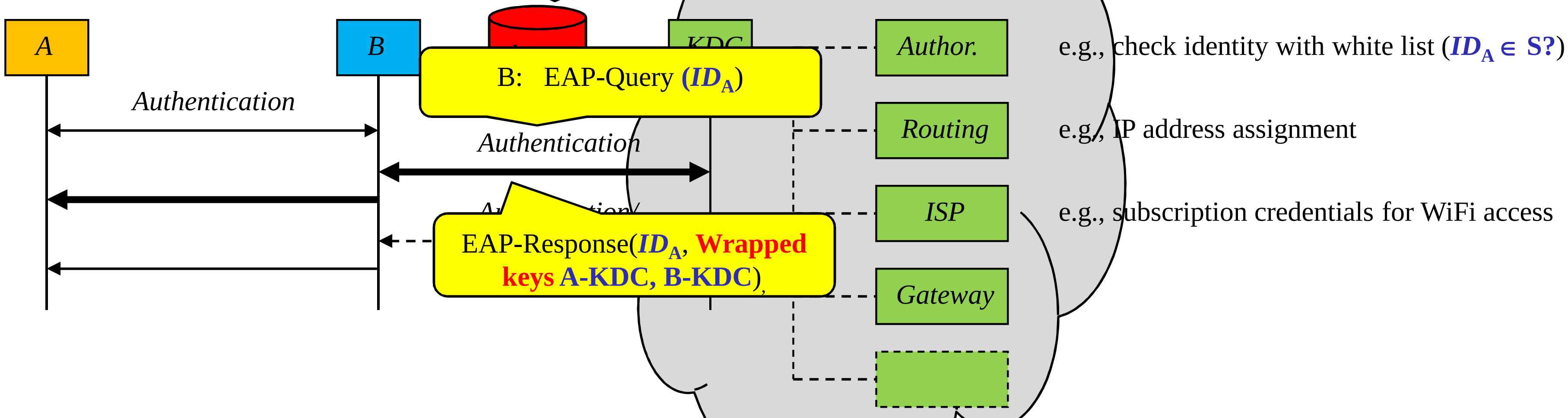
Authorization. Access Point B decides on whether/how to authorize device A (if denie[…] loss of bandwidth). *Authorization decision may be delegated to server KDC or other*

Configuration/Parameterizatio[…] […] configuration information to […] ♦ IP address assignment info; […]nts; ♦ Scheduling info (includ[…] authentication policy details) […]*ther network devices, for which it acts as proxy.*

Sequential Enrolment vs. Co[…]

All nonlocal communications with existing 802.11 frames (EAP)

Does require *online* database with keys. Key provisioning difficult in heterogeneous trust settings (may only work with lock-in by, e.g., ISP and X-trust)



Aggressive scheme: Initiate authorization/configuration processes as soon as (presumed) device identity becomes available (invisible to Client A). Access Point B can deny bandwidth if authorization negative.

Note: Communication of configuration info depends on secure channel with Client A.

Network Joining – *only* Authorization by Third Party

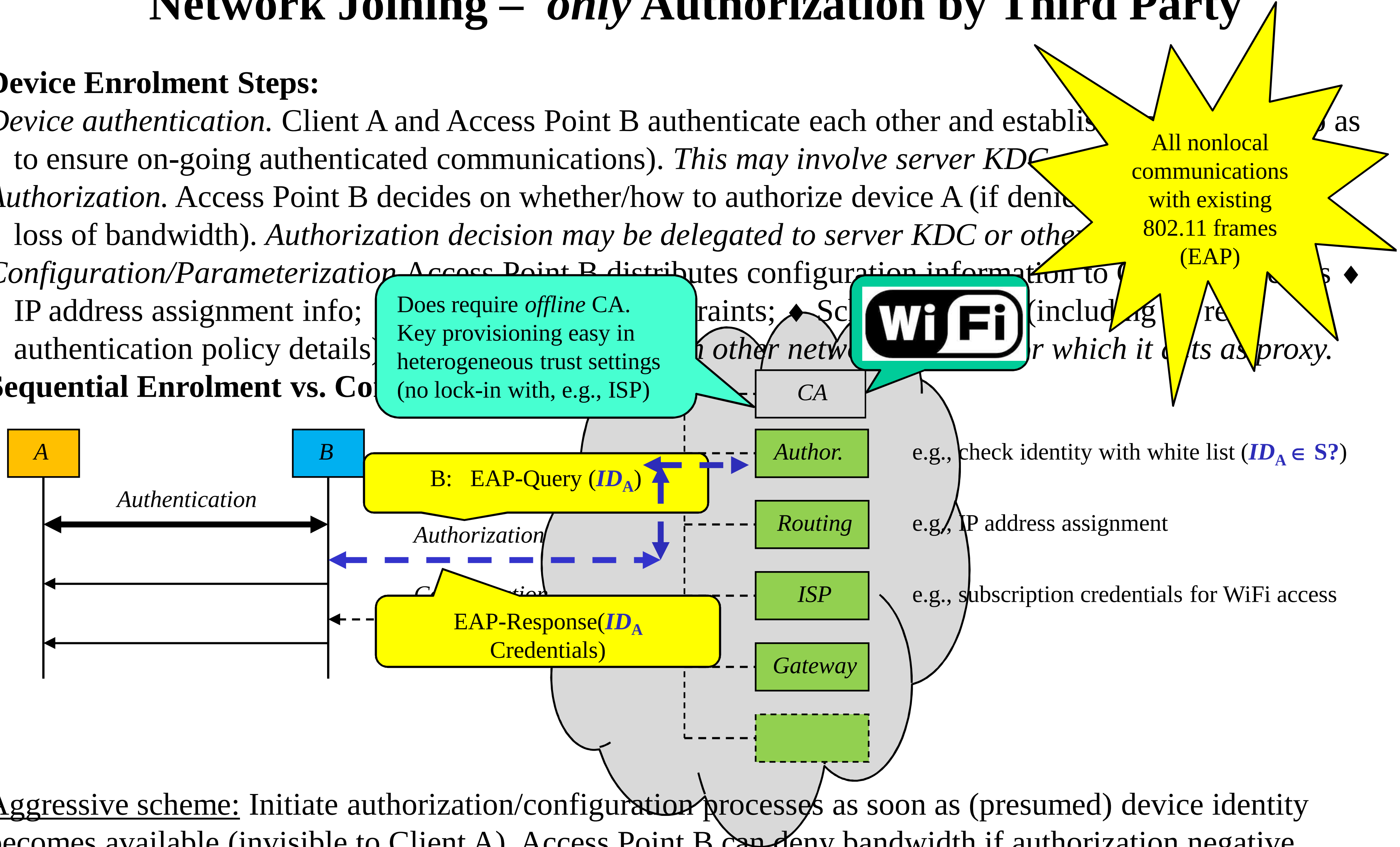
Device Enrolment Steps:

Device authentication. Client A and Access Point B authenticate each other and establis… as to ensure on-going authenticated communications). *This may involve server KDC*

Authorization. Access Point B decides on whether/how to authorize device A (if deni… loss of bandwidth). *Authorization decision may be delegated to server KDC or othe…*

Configuration/Parameterization Access Point B distributes configuration information to C… ♦ IP address assignment info; … raints; ♦ Sc… (includ… authentication policy details) … other netw… r which it … ts as proxy.

Sequential Enrolment vs. Co…



Aggressive scheme: Initiate authorization/configuration processes as soon as (presumed) device identity becomes available (invisible to Client A). Access Point B can deny bandwidth if authorization negative.

Note: Communication of configuration info depends on secure channel with Client A.

Security Concepts – A Short Introduction

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).

<u>Realizations:</u>

- *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).

<u>Realizations:</u>

- *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).

Note: with manual certificates, one usually implements ACL lists with public keys