IEEE 802.11 TGai

–

Some Notes and Thoughts on TGai Security Properties

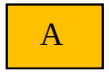
Date: 2011-10-27

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



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.

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.

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 ♦ 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.

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

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

Esoteric properties:
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.

Public-Key Key Agreement:

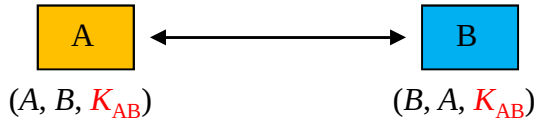
Two devices A and B derive a shared key (key agreement) and show that these have computed correctly (key confirmation) in each of the following scenarios:

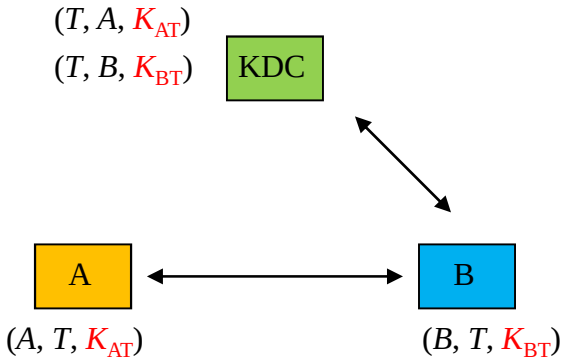
(a) Both devices do have (access to) a certificate of their public key, issued by a mutually trusted third party (certificate authority).
(b) Both devices do not have (access to) a certificate of their public key.
(c) Both devices do share a ***weak*** secret key.
(d) Both devices do have (access to)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

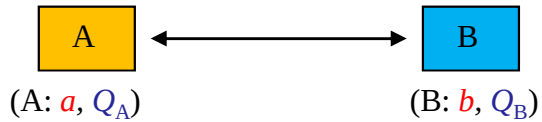
Peer-to-Peer, or with Involvement Third Party

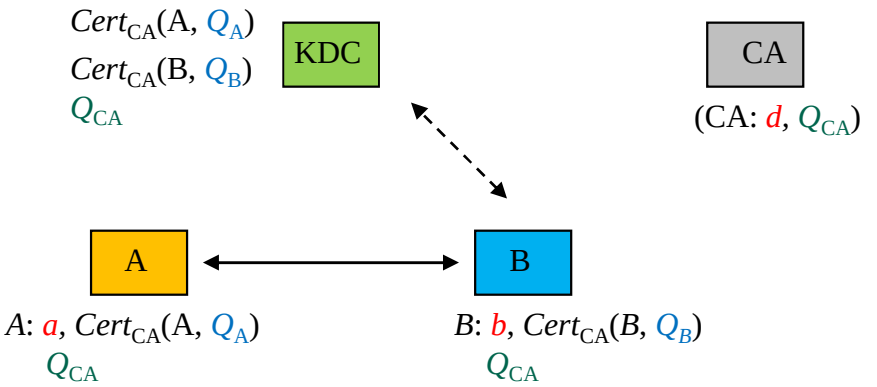
Symmetric-Key Key Agreement





Public-Key Key Agreement





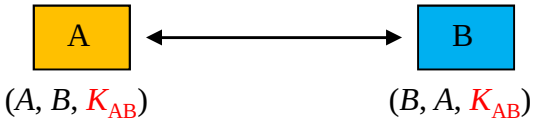
Symmetric-Key Key Agreement (1)

Symmetric-Key Key Agreement:

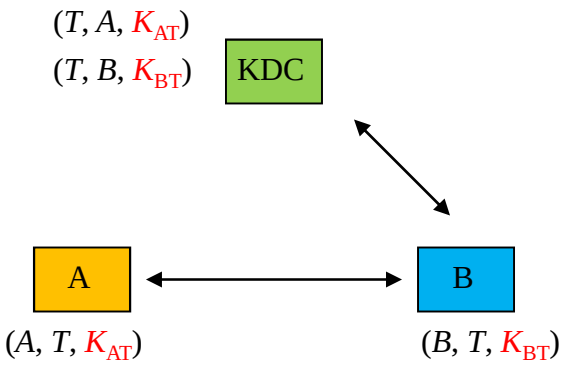
Two devices A and B derive a shared key (key agreement) and show that these have computed correctly (key confirmation) in each of the following scenarios:

(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.

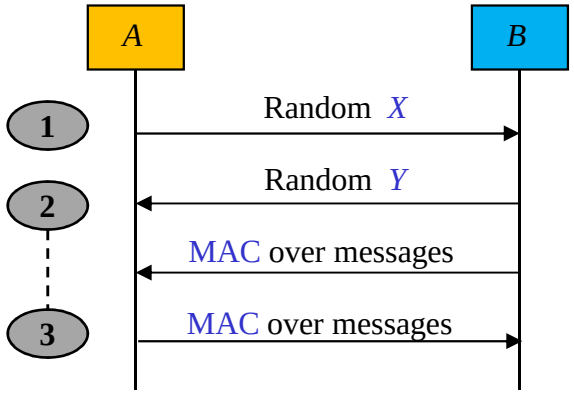
(a) Peer-to-Peer Key Establishment



(b) 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

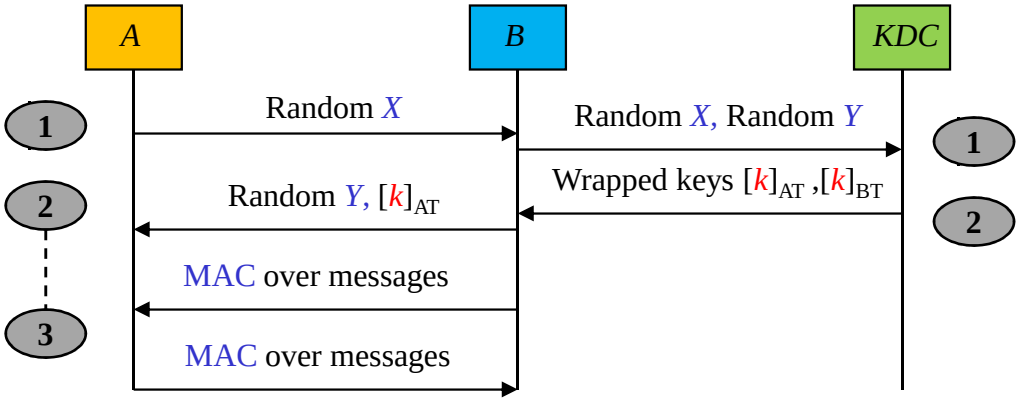
Constraints

- 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

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

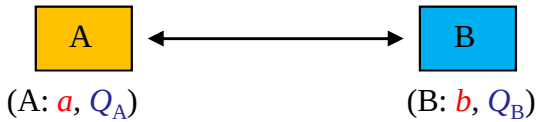
Public-Key Key Agreement (1)

Public-Key Key Agreement:

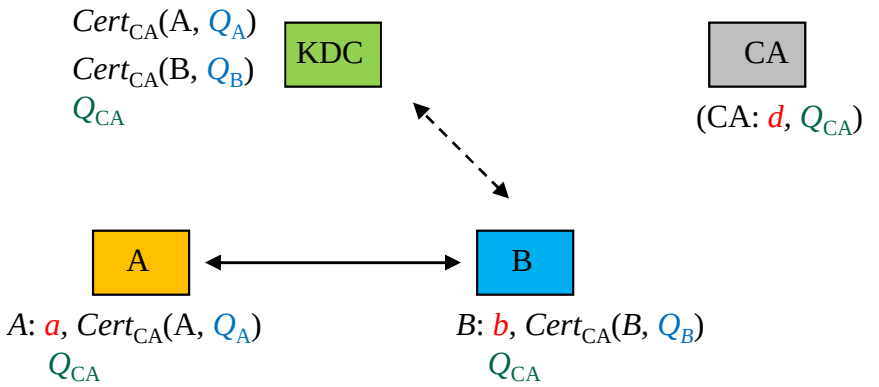
Two devices A and B derive a shared key (key agreement) and show that these have computed correctly (key confirmation) in each of the following scenarios:

(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.

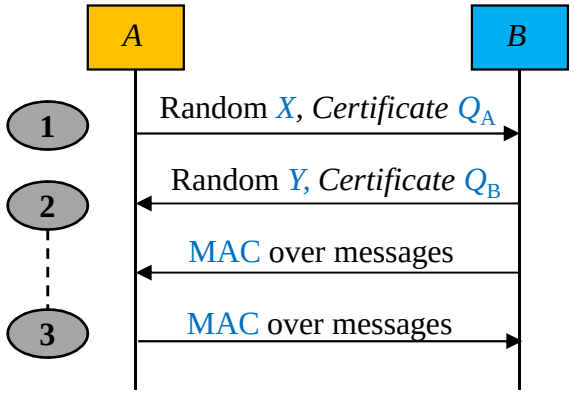
(a), (b), (c) Peer-to-Peer Key Establishment



(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

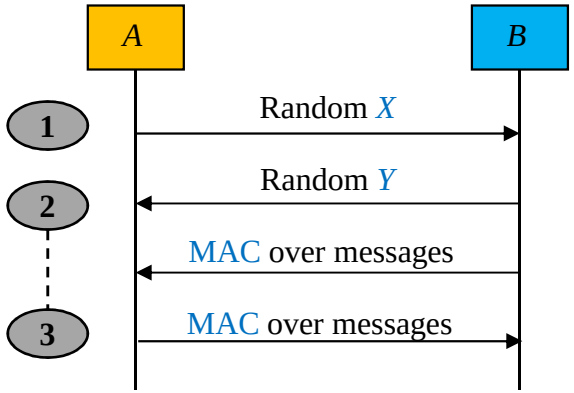
- X and Y shall be generated at random (ephemeral elliptic curve points)
- Long-term private keys d_A and d_B private to Party A, resp. Party B, and *valid during execution of protocol*
- Short-term private keys x and y private to Party A, resp. Party B and *valid during execution of protocol*
- Each 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

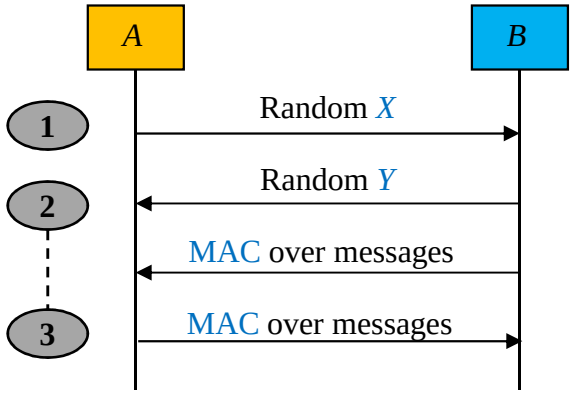
- X and Y shall be generated at random (ephemeral elliptic curve points)
- Short-term private keys x and y private to Party A, resp. Party B and *valid during the system's lifetime*

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
- *No* 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

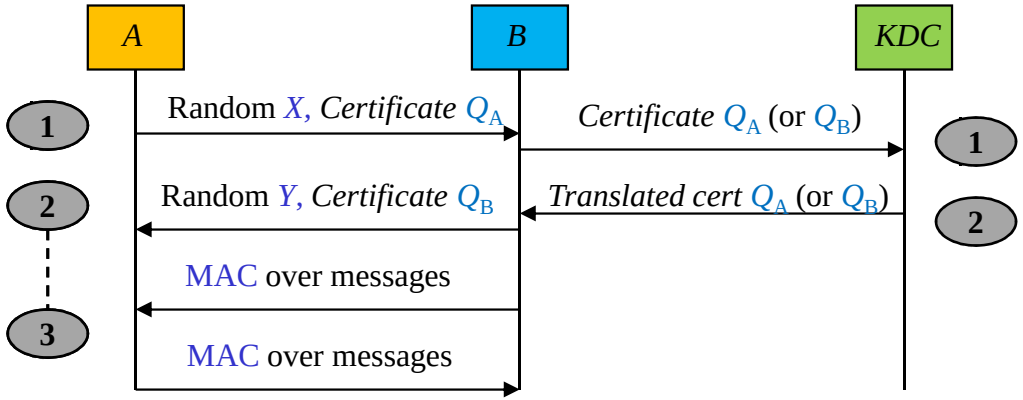
- X and Y shall be generated at random (ephemeral elliptic curve points)
- Short-term private keys x and y private to Party A, resp. Party B and *valid during the system's lifetime*

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
- *No* perfect forward secrecy
- No unilateral key control by either party
- Esoteric properties: unknown key-share resilience

Public-Key Key Agreement: (d) with Online 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 Online 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