Network Security: Diffie-Hellman

ENCS5322, NETWORK SECURITY PROTOCOLS

First Semester 2024-2025

STUDENTS-HUB.com

Uploaded By: anonymous

Outline

1. Discrete logarithm problem
2. Diffie-Hellman key exchange
3. Impersonation and MitM
4. Authenticated DH
5. Misbinding
6. A more realistic protocol
7. Perfect forward secrecy (PFS)

STUDENTS-HUB.com

Uploaded By: anonymous

Modulo arithmetic

- Exponentiation in multiplicative group Z_p^*:
  - Choose a large prime number p (e.g. 2048 bits long)
  - Z_p^* is the group of integers {1,…,p-1};
    group operation is multiplication modulo p
  - Exponentiation x^k means multiplying x with itself k times modulo p
  - g is a generator if g^k for k=0,1,2,3,… produces all the values 1,…,p-1
- For Diffie-Hellman, choose parameters p and g
  - Many critical details not covered here; see crypto literature!
- Exponentiation is commutative: $(g^x)^y = (g^y)^x$
  i.e. $(g^x \bmod p)^y \bmod p = (g^y \bmod p)^x \bmod p$

STUDENTS-HUB.com

Uploaded By: anonymous

Elliptic curve (EC)

- Points on an elliptic curve form an additive group
  - Commonly used curves: Curve25519, Curve448
  - See cryptography literature for details
- Point multiplication $n \cdot P$ means adding P to itself n times
  - n is an integer; P is a point on the elliptic curve
- Point G is a generator point if $k \cdot G$ for k=0,1,2,3,… produces all the values of the group or a large subgroup
- Point multiplication is commutative: $n \cdot m \cdot G = m \cdot n \cdot G$

STUDENTS-HUB.com

Uploaded By: anonymous

Discrete logarithm problem

- Discrete logarithm problem in Z_p^*: given $g^k \bmod p$, solve k
  - Believed to be a hard problem for large primes p and random k
  - Typical p 1024..8096 bits; k 256 bits
- Discrete logarithm problem in EC: given $n \cdot P$, solve n
  - Believed to be a hard problem
  - Typical point lengths are 160..571 bits, depending on the curve; multiplier n 256 bits
  - Why EC? Shorter key lengths and lower computation cost for the same level of security

STUDENTS-HUB.com

Uploaded By: anonymous

Unauthenticated Diffie-Hellman in Z_p^*

- A and B have previously agreed on g and p
- All operations are in Z_p^* i.e. modulo p

A chooses a random x and computes key share g^x
B chooses a random y and computes key share g^y

1. A → B: A, g^x
2. B → A: B, g^y

A calculates shared secret $K = (g^y)^x$

B calculates shared secret $K = (g^x)^y$

- It works because exponentiation is commutative
- Sniffer learns g^x and g^y ; cannot compute x, y, or g^{xy}

Elliptic Curve Diffie-Hellman (ECDH)

- A and B have previously agreed on a curve and G

A chooses a random d_A and computes key share $Q_A = d_A \cdot G$

B chooses a random d_B and computes key share $Q_B = d_B \cdot G$

1. A → B: A, Q_A
2. B → A: B, Q_B

A computes the shared secret $SK = d_A \cdot Q_B = d_A \cdot d_B \cdot G$

B computes the shared secret $SK = d_B \cdot Q_A = d_B \cdot d_A \cdot G$

For protocol designers, DH and ECDH are interchangeable algorithms

- It works because point multiplication is commutative
- Sniffer learns Q_A and Q_B ; cannot compute d_A, d_B, or SK

STUDENTS-HUB.com

Uploaded By: anonymous

Diffie-Hellman assumption

- Diffie-Hellman assumption in Z_p^*:

  given g^x and g^y, hard to solve $K = g^{xy}$

- Diffie-Hellman assumption in EC:

  given $d_A \cdot G$ and $d_B \cdot G$, hard to solve $K = d_A \cdot d_B \cdot G$

- Believed to be as hard as the discrete logarithm problem
  - Ability to compute discrete logarithms also breaks the DH assumption
  - Quantum computers could compute discrete logarithms

STUDENTS-HUB.com

Uploaded By: anonymous

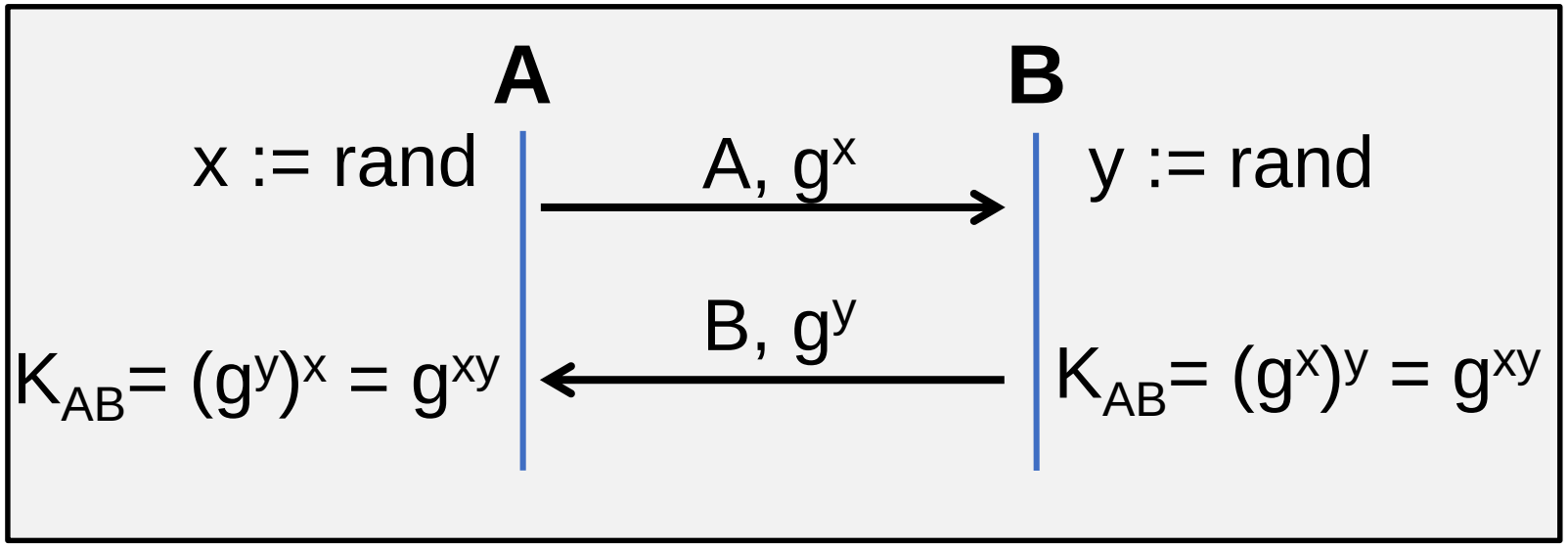
Domain parameters

- Domain parameters in Diffie-Hellman:
  - In Z_p^*, A and B must agree on the prime p and generator g
  - In ECDH, A and B must agree the curve and generator point G
- How to agree on the domain parameters?
  - Method 1: standardized parameters for each protocol or application
  - Method 2: one party chooses and signs the parameters
  - Method 3: negotiation where one party offers parameters, and the other party chooses from them
- Protocol standards usually allow many key lengths or ECDH curves, and the key-exchange starts with parameter negotiation

STUDENTS-HUB.com

Uploaded By: anonymous

Sniffing

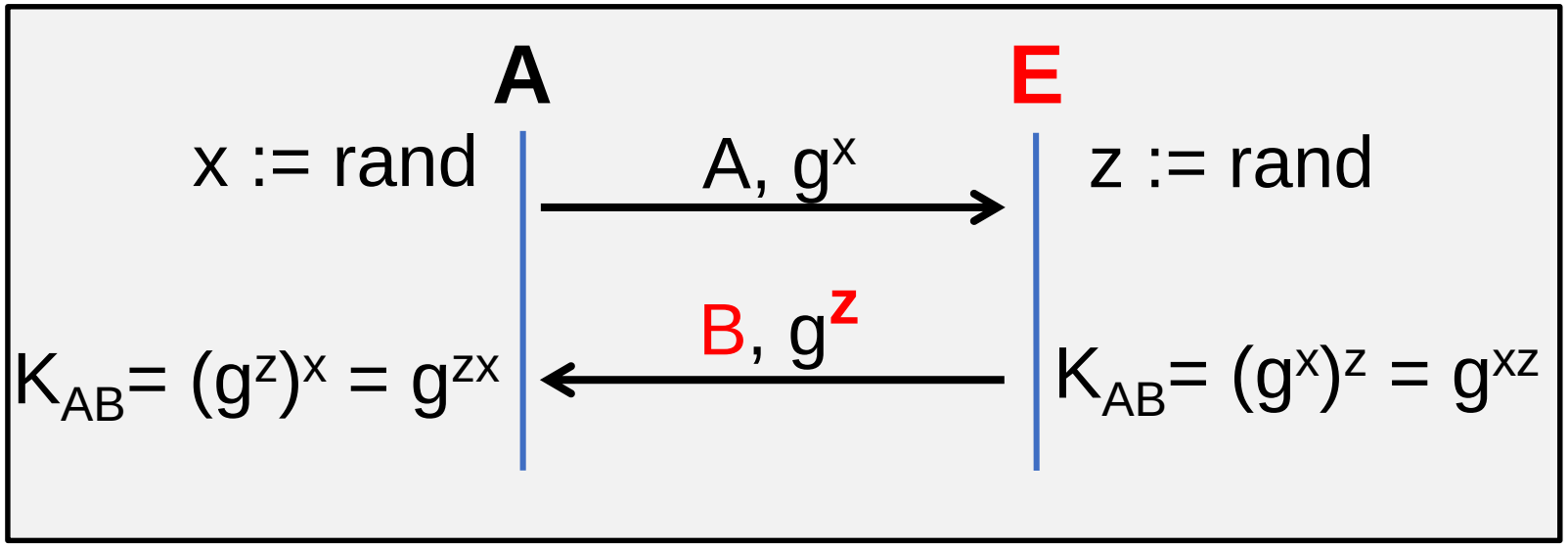


- Unauthenticated Diffie-Hellman is secure against passive attackers
  - Not possible to discover the shared secret K_{AB} by sniffing the key shares

STUDENTS-HUB.com

Uploaded By: anonymous

Impersonation attack

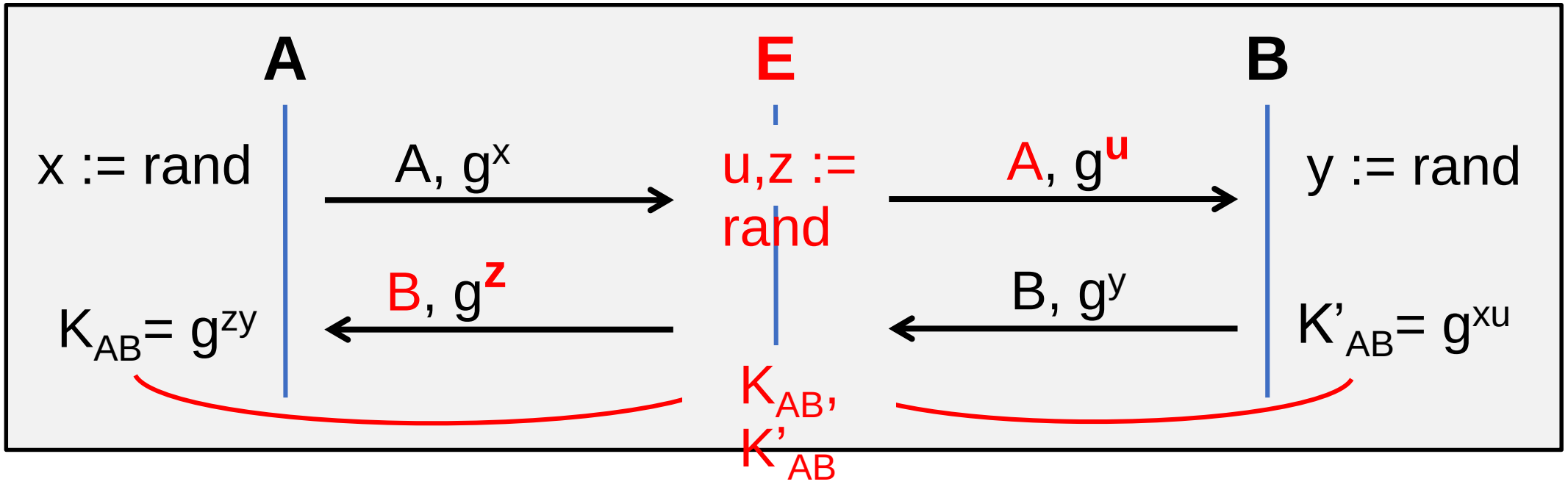


- Unauthenticated Diffie-Hellman is vulnerable to an active attacks such as impersonation:
  - Shared secret key was created, but with whom?

STUDENTS-HUB.com

Uploaded By: anonymous

Man in the Middle (MitM)



- Attacker impersonates A to B, and B to A
- Attacker creates shared session keys with both A and B
- Later, attacker can forward data between the two "secure" sessions

STUDENTS-HUB.com

Uploaded By: anonymous

Authenticated DH

1. $A \rightarrow B$: $A, g^x, S_A(g^x), Cert_A$
2. $B \rightarrow A$: $B, g^y, S_B(g^y), Cert_B$

$SK = h(g^{xy})$

Note: This is still an impractical toy protocol. Please read further

- $S_A(g^x)$ = A's signature
- $Cert_A$ = standard (X.509) public-key certificate or certificate chain
  - Subject name in the certificate must be A
  - B verifies the signature with A's public key from the certificate
- $h(g^{xy})$ = key material for deriving all necessary session keys
- Authentication prevents impersonation and MitM attacks

STUDENTS-HUB.com

Uploaded By: anonymous

Authenticated DH with key confirmation

- Three messages needed for authentication and key confirmation

1. A → B: A, B, N_A, g^x
2. B → A: A, B, N_B, g^y, S_B("Msg2", N_A, N_B, g^x , g^y), $Cert_B$,
3. A → B: A, B, S_A("Msg3", N_A, N_B, g^x , g^y), $Cert_A$

SK = h(N_A, N_B, g^{xy})

Still not a good protocol! Please read further

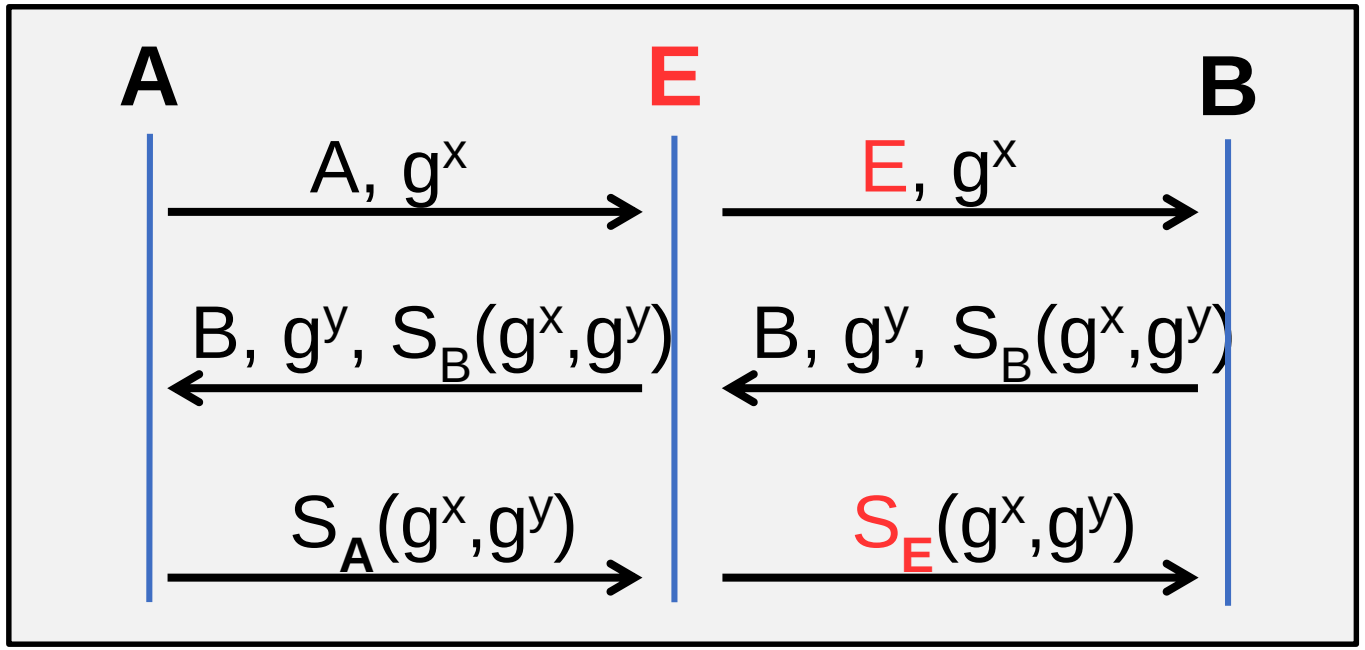
- Signatures on fresh data authenticate the endpoints
- Key confirmation: signatures prove that each endpoint knows all the parameters needed to compute the session key
  - Endpoints must trust each other about knowing the exponent

STUDENTS-HUB.com

Uploaded By: anonymous

Misbinding attack

- Misbinding of the initiator: B thinks it is connected to E. In fact, A and B are connected



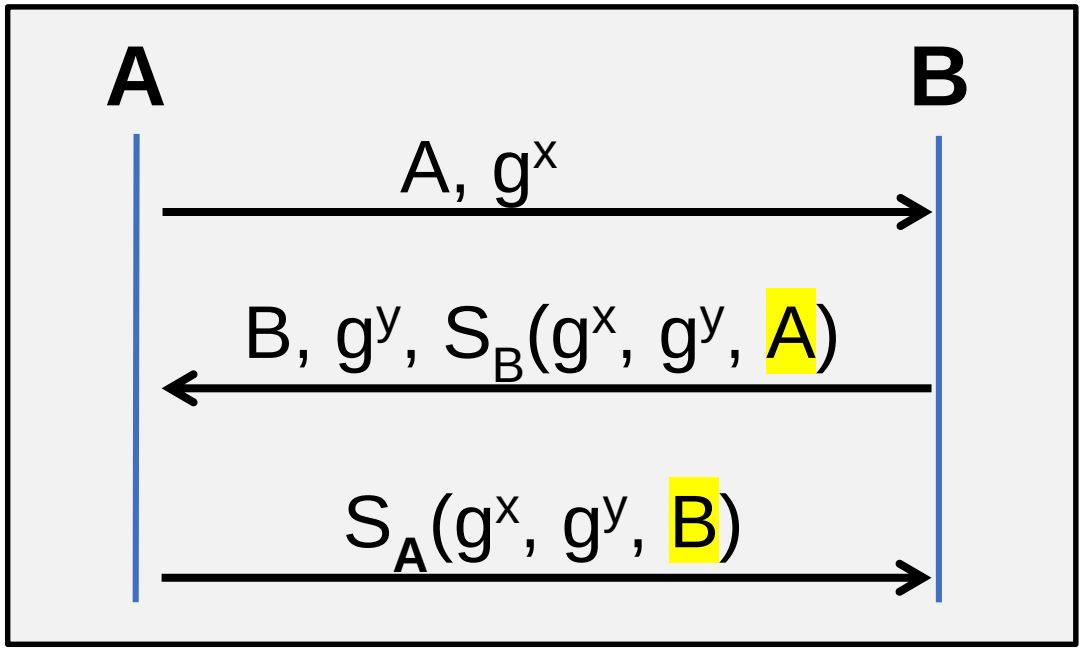
- E is a dishonest insider (E can legitimately connect to B)
- Misbinding of the responder is similarly possible

STUDENTS-HUB.com

Uploaded By: anonymous

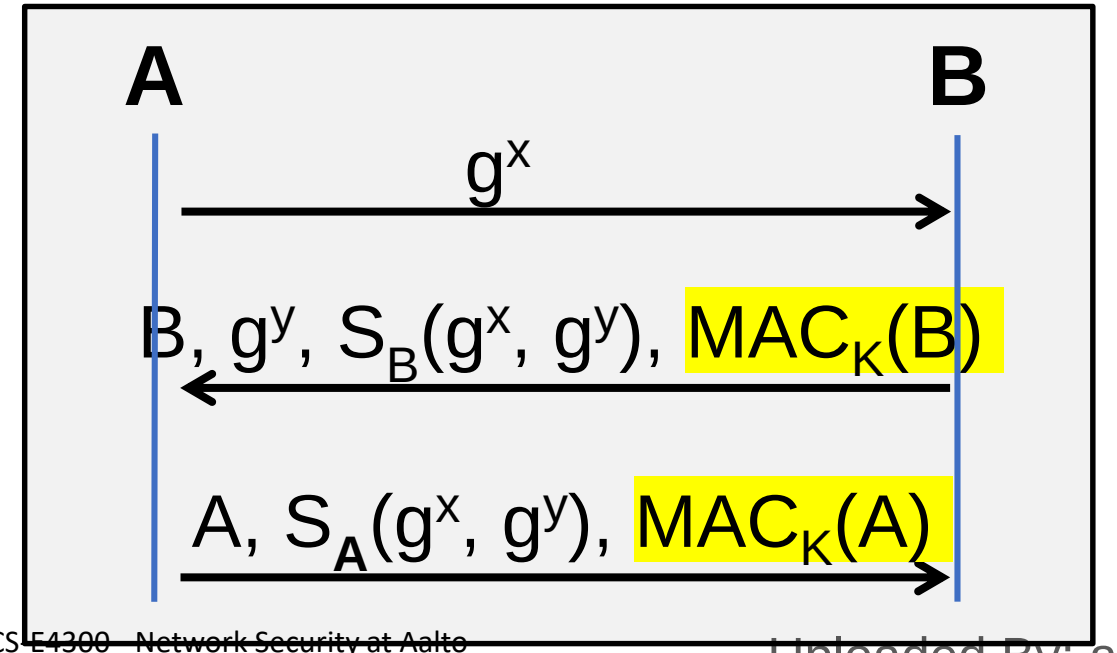
Solutions to misbinding:
check peer identifier

ISO 9798-3



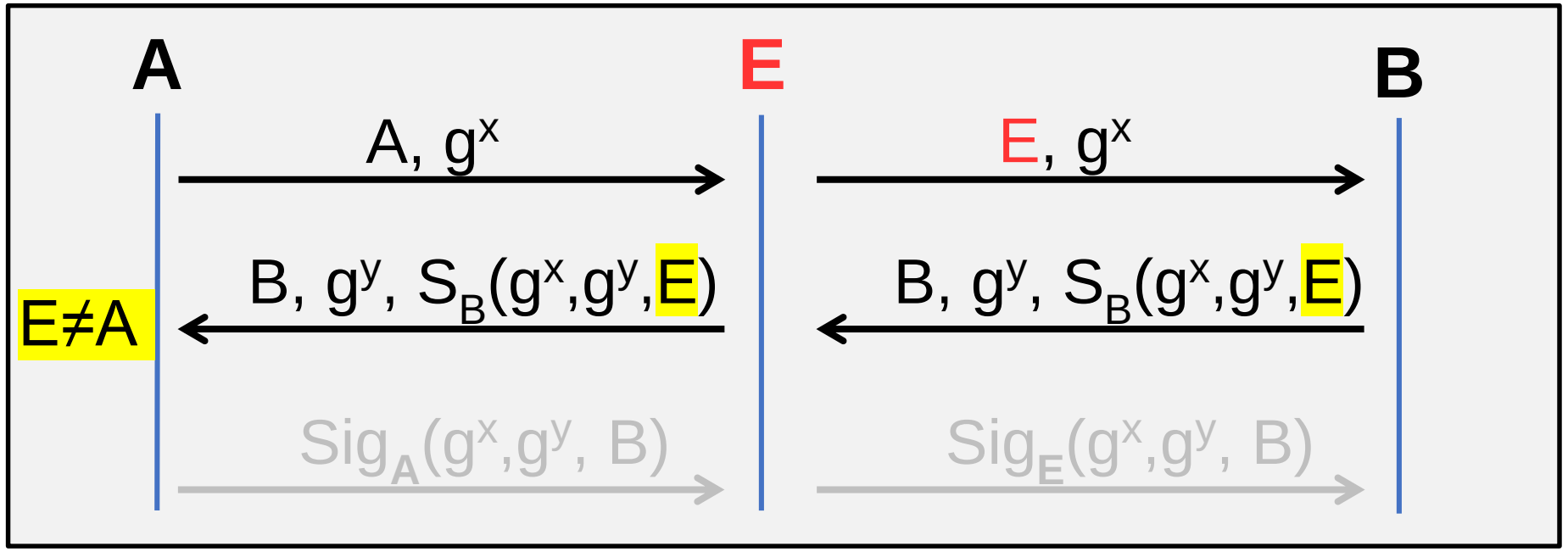
SIGMA

(easier security proofs, and slightly better protection in case of an incompetent CA)

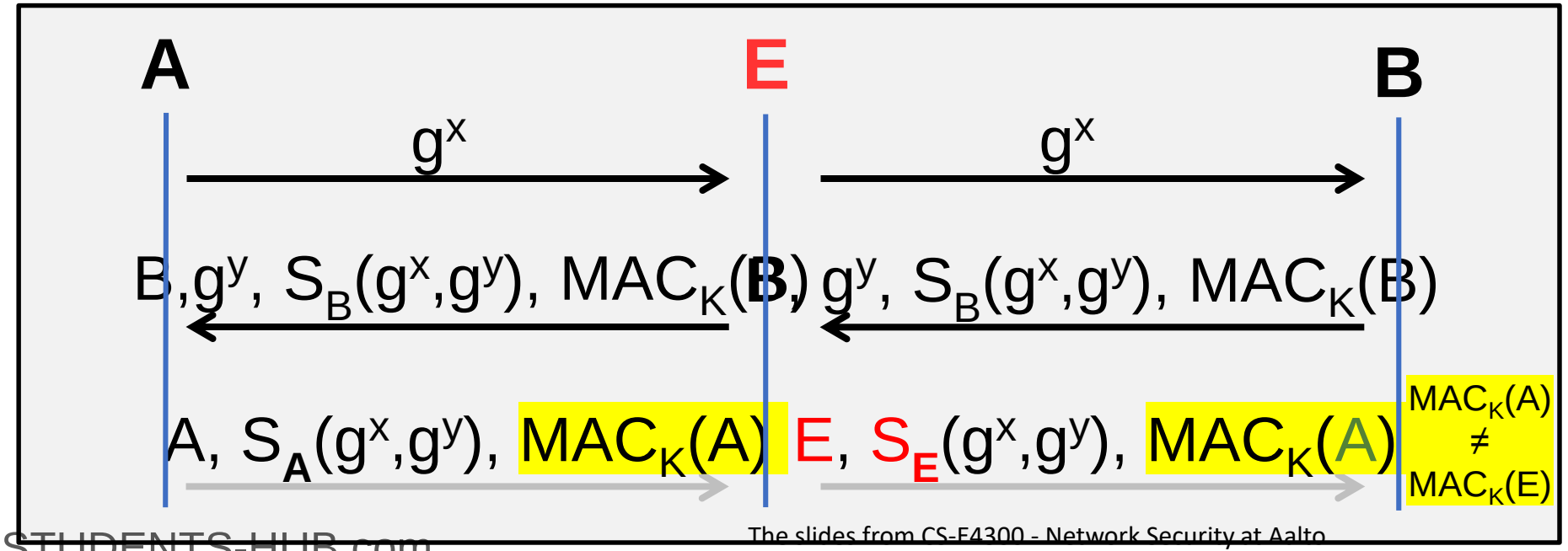


STUDENTS-HUB.com

Uploaded By: anonymous



Detecting misbinding of initiator in ISO 9798-3



Detecting misbinding of initiator in SIGMA

STUDENTS-HUB.com

Uploaded By: anonymous

A MORE REALISTIC AUTHENTICATED DIFFIE-HELLMAN PROTOCOL

STUDENTS-HUB.com

Uploaded By: anonymous

Authenticated DH

- Signed Diffie-Hellman with nonces and key confirmation:

1. $A \rightarrow B$: A, B, N_A, g, p, g^x, S_A("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. $B \rightarrow A$: A, B, N_B, g^y, S_B("Msg2", A, B, N_B, g^y), $Cert_B$, MAC_{SK}(A, B, "Responder done.")
3. $A \rightarrow B$: A, B, MAC_{SK}(A, B, "Initiator done.")

$SK = h(N_A, N_B, g^{xy})$

- Prevents impersonation, MitM and misbinding attacks
- Why so complicated?

STUDENTS-HUB.com

Uploaded By: anonymous

Authenticated DH

- Signed Diffie-Hellman with nonces and key confirmation:

1. A → B: A, B, N_A, g, p, g^x, S_A("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. B → A: A, B, N_B, g^y, S_B("Msg2", A, B, N_B, g^y), $Cert_B$, MAC_{SK}(A, B, "Responder done.")
3. A → B: A, B, MAC_{SK}(A, B, "Initiator done.")

$SK = h(N_A, N_B, g^{xy})$

- Signatures and certificates for authentication, nonces for freshness, MAC for key confirmation

Authenticated DH

- Signed Diffie-Hellman with nonces and key confirmation:

1. $A \rightarrow B$: A, B, N_A, g, p, g^x, S_A("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. $B \rightarrow A$: A, B, N_B, g^y, S_B("Msg2", A, B, N_B, g^y), $Cert_B$, MAC_{SK}(A, B, "Responder done.")
3. $A \rightarrow B$: A, B, MAC_{SK}(A, B, "Initiator done.")

$SK = h(N_A, N_B, g^{xy})$

- Signatures and certificates for authentication, nonces for freshness, MAC for key confirmation

Authenticated DH

- Signed Diffie-Hellman with nonces and key confirmation:

1. A → B: A, B, N_A, g, p, g^x, S_A("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. B → A: A, B, N_B, g^y, S_B("Msg2", A, B, N_B, g^y), $Cert_B$, MAC_{SK}(A, B, "Responder done.")
3. A → B: A, B, MAC_{SK}(A, B, "Initiator done.")

$SK = h(N_A, N_B, g^{xy})$

- Signatures and certificates for authentication, nonces for freshness, MAC for key confirmation

Authenticated DH

- Signed Diffie-Hellman with nonces and key confirmation:

1. $A \rightarrow B$: A, B, N_A, g, p, g^x, S_A("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. $B \rightarrow A$: A, B, N_B, g^y, S_B("Msg2", A, B, N_B, g^y), $Cert_B$, MAC_{SK}(A, B, "Responder done.")
3. $A \rightarrow B$: A, B, MAC_{SK}(A, B, "Initiator done.")

$SK = h(N_A, N_B, g^{xy})$

- Signatures and certificates for authentication, nonces for freshness, MAC for key confirmation

Authenticated DH

- Signed Diffie-Hellman with nonces and key confirmation:

1. A → B: A, B, N_A, g, p, g^x, S_A("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. B → A: A, B, N_B, g^y, S_B("Msg2", A, B, N_B, g^y), $Cert_B$, MAC_{SK}(A, B, "Responder done.")
3. A → B: A, B, MAC_{SK}(A, B, "Initiator done.")

SK = h(N_A, N_B, g^{xy})

- Signatures and certificates for authentication, nonces for freshness, MAC for key confirmation

Ephemeral Diffie-Hellman (DHE)

- Perfect forward secrecy (PFS): session keys and data from past sessions are safe even if the long-term secrets, such as private keys, are later compromised !
  - Even participants themselves cannot recover old session keys
- Ephemeral DH (DHE): new random DH exponents for every key exchange, forget the exponent values afterwards → PFS !
  - Similarly, ephemeral ECDH (ECDHE)
  - Cost-security trade-off: replace DH exponents periodically, e.g. once in a day or an hour, and use nonces for freshness: $SK = h(N_A, N_B, g^{xy})$

STUDENTS-HUB.com

Uploaded By: anonymous

Diffie-Hellman and nonces

- Are the nonces needed in Diffie-Hellman?

1. A → B: A, B, N_A, g, p, g^x, S_A("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. B → A: A, B, N_B, g^y, S_B("Msg2", A, B, N_B, g^y), $Cert_B$, MAC_{SK}(A, B, "Responder done.")
3. A → B: A, B, MAC_{SK}(A, B, "Initiator done.")

$SK = h(N_A, N_B, g^{xy})$

- Old DH implementations reuse exponents
  → Saving on computation. Lack of PFS. Nonces needed for freshness
- After Snowden, PFS has become mandatory → Ephemeral DH. Nonces optional
- Prudent protocol design still separates the two concerns: nonces for freshness of authentication and session key, DH for secrecy and new exponents for PSF

STUDENTS-HUB.com

Uploaded By: anonymous

Network Security:
Goals of authenticated key exchange

STUDENTS-HUB.com

Uploaded By: anonymous

Purpose of key exchange

- With public keys:
  - A and B each have public-private key pairs and certificates
  - Goal: generate a symmetric shared secret session key
  - Public keys are used for the key exchange. Session keys are used for efficient protection session data (symmetric encryption and MAC or AE)
- With a shared master secret:
  - A and B share a secret master key, e.g., 128-bit random number
  - Goal: generate a shared session key for short-term use
  - Motivation: compromise of a session key is quite likely; the seldom-used master key can be better protected, e.g., SIM
- The master key and certificates (or the CA) are called roots of trust

STUDENTS-HUB.com

Uploaded By: anonymous

Basic security goals

- Create a good session key:
  - Secret i.e. known only to the intended participants
  - Fresh i.e. never seen or used before
  - Separation short-term secrets and long-term security: compromise of session keys does not endanger future authentication or secrecy
- Authentication:
  - Mutual = two-directional authentication: each party knows who it shares the session key with
  - Sometimes only one-way = unidirectional authentication

STUDENTS-HUB.com

Uploaded By: anonymous

Other common security properties

- Perfect forward secrecy (PFS)
  - Compromise of long-term secrets today should not compromise old session data
  - Typically achieved with empheral Diffie-Helmann
  - Can also be implemented with public-key encryption by creating a fresh key pair and then throwing it away

STUDENTS-HUB.com

Uploaded By: anonymous

Other common security properties

- Entity authentication: each (or one) participant knows that the other is online and participated in the protocol
- Key confirmation: each (or one) participant knows that the other knows the session key (implies entity authentication)
  - Receives proof vs. trusts the other participant

A knows SK.

B knows SK. B knows that A knows SK.

A knows that B knows SK. A knows that B knows that A knows SK.

…

But common knowledge is not possible in a distributed system.

STUDENTS-HUB.com

Uploaded By: anonymous

Correspondence properties

- Correspondence properties (or consistency): agreement between the states and beliefs of the two endpoints, or between the endpoints' initial intentions and final states
  - More precise definition of authentication and key confirmation
  - Example:
    If responder B accepts the session key K for communication with initiator A, then A has previously created the key K for communication with B

STUDENTS-HUB.com

Uploaded By: anonymous

Other common security properties

- Contributory key exchange: both endpoints contribute randomness to the session key; neither can decide the key alone
  - Key distribution where one party decides the key; common in broadcast and sometimes in asynchronous communication
- Algorithm agility: support for negotiating, upgrading and deprecating algorithms
  - Downgrading protection: Endpoints negotiate the best algorithms and latest protocol version supported by both, and the attacker cannot manipulate the process (never absolute protection)

STUDENTS-HUB.com

Uploaded By: anonymous

Privacy and identity issues

- Identity protection
  - Unauthenticated Diffie-Hellman first; then encrypt the identities and certificates
  - Passive sniffer cannot learn the identities of the protocol participants
  - Usually only one side can have identity protection against active attacks: one side must reveal its identity first, making its identity vulnerable to active attacks

Would you give stronger identity protection to the initiator or responder?

STUDENTS-HUB.com

Uploaded By: anonymous

Privacy and identity issues

- Non-repudiation
  - Evidence preserved, so that a participant cannot later deny taking part in the protocol (usually not an explicit goal)
- Plausible deniability
  - No evidence left of taking part (usually not an explicit goal either)

STUDENTS-HUB.com

Uploaded By: anonymous

DoS resistance

- Various denial-of-service resistance requirements:
  - The protocol cannot be used to exhaust memory or CPU of the participants
  - Not easy to spoof packets that prevent others from completing a key exchange (especially off-route attackers)
  - When an on-route MitM attacker stops dropping and breaking messages, the protocol recovers
  - The protocol cannot be used to flood third parties with data or to amplify DDoS attacks
- DoS protection is never absolute

STUDENTS-HUB.com

Uploaded By: anonymous

Authenticated DH properties

- Signed Diffie-Hellman with nonces and key confirmation:

1. $A \rightarrow B$: A, B, N_A, g, p, g^x, S_A("Msg1", A, B, N_A, g, p, g^x), $Cert_A$
2. $B \rightarrow A$: A, B, N_B, g^y, S_B("Msg2", A, B, N_B, g^y), $Cert_B$, MAC_{SK}(A, B, "Responder done.")
3. $A \rightarrow B$: A, B, MAC_{SK}(A, B, "Initiator do

$SK = h(N_A, N_B, g^{xy})$

Which security properties?

- Secret, fresh session key
- Mutual or one-way authentication
- Entity authentication, key confirmation
- Perfect forward secrecy (PFS)
- Contributory key exchange
- Downgrading protection
- Identity protection
- Non-repudiation
- Plausible deniability
- DoS resistance

STUDENTS-HUB.com

Uploaded By: anonymous

What is a protocol flaw?

- Poorly understood security requirements
- Limitations on the applicability of the protocol:
  - Is the protocol used for a new purpose or in a new environment?
  - Historical examples: insider attacks, multiple parallel executions
  - Timely example: distributed cloud implementation
- Unwritten expectations for implementations
  - Encryption in old specs is assumed to protect integrity
  - Authenticated messages should include type tags
- New attacks and security requirements arise over time:
  - DoS amplification, PFS, identity protection

STUDENTS-HUB.com

Uploaded By: anonymous

Notes on protocol engineering

- Security is just one requirement for network protocols
  - Cost, implementation complexity, performance, deployability, code reuse, time to market etc. may override some security properties
- Security protocol engineering requires experienced experts and peer scrutiny
  - Reuse well-understood solutions like TLS; avoid designing your own
  - Only use strong security solutions (privacy and DoS protection are never strong, though)
- The most difficult part is understanding the problem
  - Must understand both security and the application domain
  - When the security requirements are well understood, potential solutions often become obvious

STUDENTS-HUB.com

Uploaded By: anonymous