Authentication & Key Exchange

Pierre-Alain Fouque

Université de Rennes 1

November, 2020
1. Generality

2. Authentication

3. Key Exchange
   - Diffie-Hellman
   - Signed Diffie-Hellman
   - Signed Diffie-Hellman
   - TLS Protocol
How to authenticate?

How sharing and exchanging a secret common value? (and why?)
Encrypt and authenticate the communications

- essential for digital payment on the internet
- ensure confidentiality and integrity
- good performances

Use of symmetric key cryptography (shared key)

How can we compute a session key?
Key Exchange

Several Methods to establish a (secret) key

- **Key distribution**: a server chooses a key and transfers it to everyone
- **Key exchange**: everyone participates to the *negotiated* key

A *client* (Alice) authenticates herself to a server (Bob)
Key Exchange

Several Methods to establish a (secret) key

- **Key distribution**: A server chooses a key and transfers it to everyone.
- **Key exchange**: Everyone participates to the negotiated key.

A *client* (Alice) authenticates herself to a server (Bob)
Key Exchange

Several Methods to establish a (secret) key

- **Key distribution**: a server chooses a key and transfers it to everyone
- **Key exchange**: everyone participates to the negotiated key

A *client* (Alice) authenticates herself to a server (Bob)
Key Exchange

Several Methods to establish a (secret) key

- **Key distribution**: a server chooses a key and transfers it to everyone
- **Key exchange**: everyone participates to the \textit{negotiated} key

A \textit{client} (Alice) authenticates herself to a server (Bob)
A *client* authenticates himself to a *server*

**Usual situations:**
- access control
- internet connexion, VPN, mobile phone, ...

**General Principle**
The server "tests" the client by performing it an operation that only the legitimate client can do correctly
Authentication

Example (Challenge/Response)

Decrypt a random value sent by the server

Alice

Bob
Authentication

Example (Challenge/Response)

Decrypt a random value sent by the server

Alice

Alice, Hi

Bob
Example (Challenge/Response)

Decrypt a random value sent by the server

Alice

Alice, Hi

Hi, R

Bob
Example (Challenge/Response)

Decrypt a random value sent by the server

Alice

Alice, Hi

Hi, $R$

$s = RSA^{-1}(R)$

Bob

test, $s^e = R$?
Two distinct frameworks for authentication

1. **No active attack**: after the authentication, the server knows he is still discussing with the client.

2. **In case of active attack**: the attacker can substitute to the authenticated client after the authentication \(\Rightarrow\) use a key exchange.
### Definition: Key Agreement

Generating a common secret (session key) by 2 entities discussing on an unsecure channel

### Two possible attack scenerios:

- **Passive attacks**: channel *listened* only
- **Arbitrary active attacks**
Diffie-Hellman Protocol

Attempts to negotiate a common secret through a public channel

Alice

\[ x, \]

Bob

\[ y, \]
Diffie-Hellman Protocol

Allows two actors to negotiate a common secret through a public channel.

\[ X = g^x \]

Alice

Bob

x, y,
Diffie-Hellman Protocol

Allows two actors to negotiate a common secret through a public channel

Alice

Bob

\[ X = g^x \]

\[ Y = g^y \]
Diffie-Hellman Protocol

Allows two actors to negotiate a common secret through a public channel

\[ X = g^x \]
\[ Y = g^y \]
\[ X^y = g^{xy} \]
\[ Y^x = g^{xy} \]

Common key \( K = g^{xy} \)
Without authentication: we do not check with who are we talking (we trust each other)

The interested property is then the key confidentiality

1. Most important property in a key exchange
2. Which confidentiality?
   - Complete: the adversary $A$ cannot compute the key $k$
   - Indistinguishability: $A$ cannot distinguish $k$ from a random element
3. Which environment?
   - What garanties if multiple parallel sessions?
   - Passive or active attacks (concurrent sessions)?
Key exchange and authentication

Authentication: only the legitimate users can authenticate themselves

Authentication can be:

- unilateral (e.g. HTTPS)
- bilateral ou mutual: each party performs a verification
  - authenticated key exchange (AKE)
  - succeeds only if the identities are authenticated

Same questions on the environment and attacker model
Additional properties (integrity of the session key)

An active attacker must not be able to manipulate the exchanged messages and modify the final value of the session key.

Possible issues:

- The attacker can make the protocol failed.
- The attacker can force some bits of the key.
- The attacker can force the value of the key.
- The attacker can link the new key to previous ones (e.g., same unknown key established twice).
Additional properties (impossibility to control the session key)

No participant of a key exchange must be able to control the final value of the session key

Possible issues:
- A participant can force a "weak key"
- Or simply a key known by someone else

Hard to enforce in practice ...
"Session-ID" and active attacks

Simple technique to "avoid" any modification of the exchanged message in the protocol

Add in the key derivation a session-id, i.e. a quantity which depends on the set of all exchanged messages
Bad Key Exchange

Alice

$pk$

$E_{pk}(r)$

Sends her $pk$

Bob

random $r$

Common key = $r$

- Bob has the control on the key!
- Advantage: if Alice later proves her knowledge of the key $r$, she authenticates herself
- used in SSH v1 and SSL
**Authenticated key exchange**

**Key exchange protocols proposed in 1976**

- based on the *one-way function* concept
- non authenticated in the basic version
Authenticated Diffie-Hellman key exchange

Alice

- $g^x$
- $g^y$
- random $x$

Bob

- $g^x$
- $g^y$
- random $y$

Common secret value: $K = g^{xy} = (g^x)^y = (g^y)^x$

- mod $p$ or over an elliptic curve
- **forward secure**: no long-term keys involved
Ephemeral keys and forward secrecy

Forward secrecy: the session keys are not put in danger by leakage of long-term keys.

Simple mechanism to achieve it: use ephemeral public keys to protect the session key.
- optionally in combination with long-term keys
- ephemeral secret keys have to be erased as soon as the exchange is over
- e.g. "Server key" in SSH v1
Diffie-Hellman Key Exchange and Active Attacks

DH protocol is vulnerable to active attacks (man-in-the-middle):

**Alice**

\[ g^x \]

\[ g^{y'} \]

**Bob**

\[ g^{x'} \]

\[ g^y \]

random \( x \), \( K = g^{x'y} \)

random \( y \), \( K = g^{x'y} \)
Signed Diffie-Hellman version 1

Alice

Bob

\[ A, g^x, \text{sign}_A(g^x) \]
\[ B, g^y, \text{sign}_B(g^y) \]

random \( x \) \hspace{1cm} random \( y \)

Avoid man-in-the-middle attacks:
- impossible for an attacker to forge \( \text{sign}_A(g^{x'}) \)

Replay attacks (Non fresh values):
- if a pair \((g^x, \text{sign}_A(g^{x'}))\) is obtained, we can use it to authenticate as \( A \) several times
- but we are not able to compute the corresponding key
Station-to-station protocol (Signed Diffie-Hellman version 2)

Alice

$A, g^x$

$B, g^y, \text{sign}_B(g^y, g^x)$

$A, \text{sign}_A(g^x, g^y)$

random $x$, check sign

Bob

random $y$, check sign

Best of two worlds:

- avoid man-in-the-middle attacks: impossible for an attacker to forge $\text{sign}_B(g^y, g^x)$
- avoid replay: the DH value of the correspondent is also signed

But ...
Identity Impersonation

Alice

- $A$, $g^x$
- $B$, $g^y$, $\text{sign}_B(g^x, g^y)$
- $A$, $\text{sign}_A(g^x, g^y)$

Bob

- $C$, $g^x$
- $B$, $g^y$, $\text{sign}_B(g^x, g^y)$
- $C$, $\text{sign}_C(g^x, g^y)$

$K = g^{xy}$

- All exchanged messages sent by Alice are seen by Bob as coming from Charlie.
- Charlie does not know the key $(g^{xy})$: Unknown Key-Share (UKS) attack.
Identity impersonation can have dramatic effects

- Bob is a bank ...
- Alice and Charlie are two customers
- Alice put money on her account with digital cash
- the bank puts money on Charlie account
Signed Diffie-Hellman version 3

Security Proof:

- forward secrecy: key integrity, protection of the long-term secrets (signature keys), ...
- ISO/IEC 9798-3 Standard
SSL Protocol (Secure Socket Layer)

- developed by Netscape Inc. to secure communications
- OSI application level
- started deployed (1994) with SSL v2 ... (56 bits DES, MD5)
- 1996 : SSL v3 (SSL v2 is not secure and no more used)
- TLS 1.0 (≈ SSL v3) in 1999
- TLS 1.2 (January 2008), RFC 4346
- Nowadays TLS 1.3 (August 2018), RFC 8446
Develop a secure API using objects that can be viewed as sockets by the applications using them

**Compatibility effort:**

- redploy existing applications
- intermediate layer under existing protocols: HTTP, POP, IMAP, ...
The scheduling of a session protected by TLS is composed of 2 stages:

- **Handshake Phase**: Negotiation of a cipher suite (cryptographic algorithms to encrypt, authenticate, ...) and key exchange.
- **Record Phase**: Encryption and authentication of the communications.

The authentication is unilateral (server authenticate only):

- The mutual authentication is heavy to deploy in practice.
- Variants (TLS-PSK, TLS-SRP, ...)

---

Pierre-Alain Fouque

Authentication & Key Exchange
Scheduling of the protocol

Preferred cipher

Verify

HMAC

Client

Serveur

ClientHello
Ciphersuite, Random

ServerHello
Certificate
ServerKeyExchange
ServerHelloDone

X509 (+ CertificateRequest)

ClientKeyExchange
ChangeCipherSpec
Finished

encrypted and MAC

ChangeCipherSpec
Finished

encrypted + MAC