Mutual Authentication for Low-Power Mobile Devices

joint work with Markus Jakobsson

David Pointcheval
Département d'Informatique
ENS - CNRS

David.Pointcheval@ens.fr http://www.di.ens.fr/~pointche

Overview

- Introduction
- Key-Agreement and Mutual Authentication
  - Definitions
  - Security Model
  - Example
- New Proposal
  - Security
  - Partial Forward-Secrecy
- Conclusion

Secret Communications

For many applications confidentiality of the communications is required

- financial transactions
- medical information
- industrial/commercial data
- intellectual property
- …

Encryption

Cryptography provides various solutions:

- symmetric encryption
  - both parties must initially share a secret
  - if the shared secret is corrupted all the communications are revealed
- public key encryption
  - it is very costly

  $\Rightarrow$ Key Agreement Protocol

Key Agreement Scheme

Two parties (a client-Alice and a server-Bob) each owns a pair of public/private keys

After a short communication, they both share a common secret data such that:

- semantic security

  no polynomial time adversary can learn any information about this data from the public data and the view of the communication

Further Properties

- mutual authentication

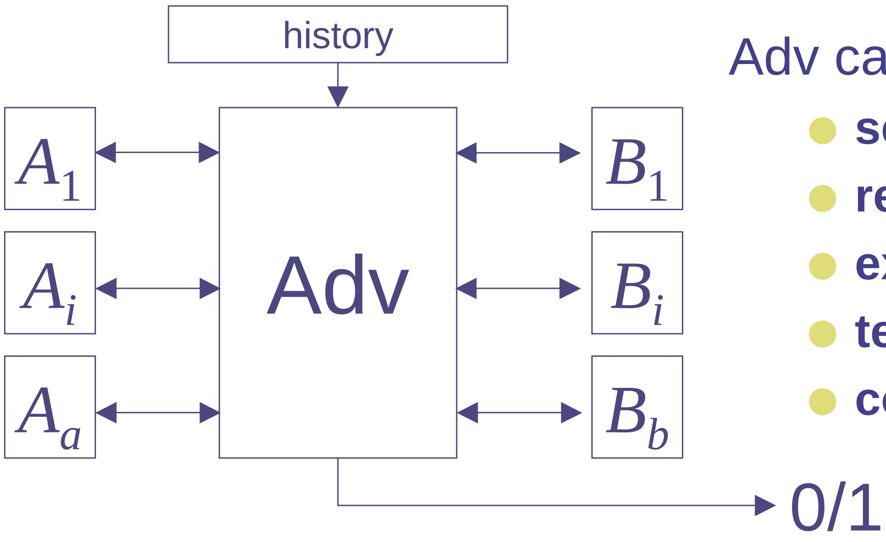
  they are both sure to share the secret with the people they think they do

- forward secrecy

  even if a long-term secret data is corrupted, previous shared secrets are still semantically secure

Formal Model

We use the BR-model revisited by Shoup



Adv can ask

- **send**-queries
- **reveal**-queries
- **execute**-queries
- **test**-query
- **corrupt**-queries

Formal Model (cont'd)

- The adversary has the entire control of the network with **send**-queries:
  - to send message to Alice or Bob (in place of Bob or Alice respectively)
  - to intercept, forward and/or modify messages
- The history can be built using the **execute**-query, but also simply forwarding messages using **send**-queries

Formal Model (cont'd)

- A misuse of the secret data is modeled by the **reveal**-query, which is answered by this secret data
- For the semantic security, the adversary asks one **test**-query which is answered, according to a bit b, by
  - $b=0$: the actual secret data
  - $b=1$: a random string

$\Rightarrow$ the adversary has to guess this bit b

Forward Secrecy

Forward secrecy means that the adversary cannot distinguish a session key established ***before*** any corruption of the long-term secret keys:

- the **corrupt**-query is answered by the long-term secret key of the corrupted party
- then the **test**-query must be asked on a session key established ***before*** any **corrupt**-query

Diffie-Hellman Key Exchange

The most classical key exchange scheme has been proposed by Diffie-Hellman:

G = <g>, cyclic group of prime order q

- Alice chooses a random $x \in \mathbf{Z}_q$, computes and sends $X=g^x$
- Bob chooses a random $y \in \mathbf{Z}_q$, computes and sends $Y=g^y$
- They each can compute the session key

$$K = Y^x = X^y$$

Properties

- It is well-known to provide the semantic security of the session key under the Decisional Diffie-Hellman Problem
- If one derives the session key as $k = \mathrm{H}(K)$, where H is assumed to behave like a random oracle, semantic security is relative to the Computational Diffie-Hellman Problem

Further Features

- But there is no authentication
- By simply signing both flows and adding key confirmation rounds, one easily gets

mutual authentication + forward secrecy

Alice (S_a, P_a)		Bob (S_b, P_b)
$x \in \mathbf{Z}_q$, $X = g^x$	Bob, X, $\sigma(S_a, X)$ →	$y \in \mathbf{Z}_q$, $Y = g^y$ $K = X^y$
k_1 correct?	← Alice, Y, $\sigma(S_b, X, Y)$, k_1	$k_1 = H(K\|1)$
$k_2 = H(K\|2)$	k_2 →	k_2 correct?
	$k = H(K\|0)$	

Properties

- It provides a high security level, relative to the Computational Diffie-Hellman Problem, in the random oracle model
- It requires high on-line computational cost:
  - at least one exponentiation
  - one signature (Schnorr = 0 exp. on-line)
  - one verification (Schnorr = 2 exp. on-line)

Discussion

- Schnorr's signature:
  - the on-line signing process is very low
  - the verification process requires two exp.
- What about encryption ?
  - One could replace signatures by public-key encryption
  - But no PK Encryption scheme with both efficient encryption and decryption processes

What about mixing encryption/signature ?

High Level Description

- Bob decrypts an El Gamal ciphertext to authenticate himself
- Alice (low-power) uses a Schnorr identification to authenticate herself
  - the server does not introduce any randomness
  - for a designated server, she can precompute everything

New Proposal

Alice $(x, y = g^x)$ — Bob $(X, Y = g^X)$

$a \in \mathbb{Z}_q$, $A = g^a$, $t \in \mathbb{Z}_q$, $T = g^t$

$K = Y^a$

$k = H_0(Y,A,K)$

$r = H_1(T,Y,A,K)$

$k_2 = H_2(Y,A,K)$ $\xrightarrow{\text{Bob}, A, r}$ $K = A^X$

$k_2 = H_2(Y,A,K)$

k_2 correct? $\xleftarrow{k_2, e}$ $0 \le e \le 2^\kappa$

$s = t - xe \bmod q$ $\xrightarrow{s}$ $H_1(g^s y^e, Y,A,K) = r$?

$k = H_0(Y,A,K)$

Security Result

Alice $(x, y = g^x)$ — Bob $(X, Y = g^X)$

$a \in \mathbb{Z}_q$, $A = g^a$, $t \in \mathbb{Z}_q$, $T = g^t$

$K = Y^a$

$k = H_0(Y,A,K)$

$r = H_1(T,Y,A,K)$

$k_2 = H_2(Y,A,K)$ $\xrightarrow{\text{Bob}, A, r}$ $K = A^X$

$k_2 = H_2(Y,A,K)$

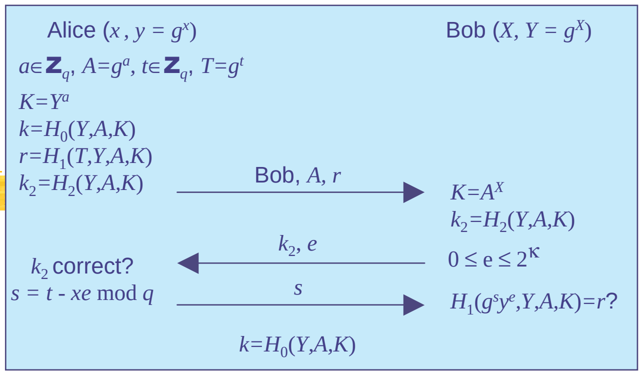
k_2 correct? $\xleftarrow{k_2, e}$ $0 \le e \le 2^\kappa$

$s = t - xe \bmod q$ $\xrightarrow{s}$ $H_1(g^s y^e, Y,A,K) = r$?

$k = H_0(Y,A,K)$

- Semantic Security:
  to get any information about k,
  one has to solve CDH(Y,A)

Security Result
simulation of Alice



Without x : thanks to
the random oracle H_1

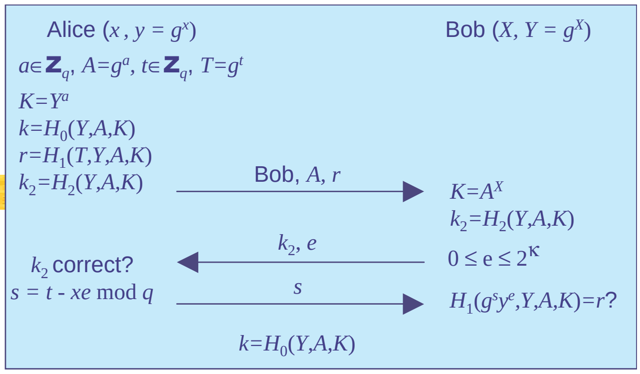
- **send**-queries

  for $a \in \mathbf{Z}_q$, $A=g^a$, and a random r,
  given e, one chooses a random s
  and defines $H_1(g^s y^e,Y,A,Y^a) \leftarrow r$

- **reveal/test**-queries

  with a, one can compute $(Y,A,K)=(Y,A,Y^a)$
  and then k and k_2

Security Result
simulation of Bob



Without X : thanks to
the random oracles, … but not enough

- **send**-queries

  with A, one answers a random k_2
  - it defines $H_2(Y,A,\mathrm{CDH}(Y,A)) \leftarrow k_2$

- **reveal/test**-queries
  - the same way, one answers a random k
  - it defines $H_0(Y,A,\mathrm{CDH}(Y,A)) \leftarrow k$

Security Result
random oracles

Alice (x, $y = g^x$)		Bob (X, $Y = g^X$)
$a \in \mathbf{Z}_q$, $A=g^a$, $t \in \mathbf{Z}_q$, $T=g^t$		
$K=Y^a$		
$k=H_0(Y,A,K)$		
$r=H_1(T,Y,A,K)$		
$k_2=H_2(Y,A,K)$	$\xrightarrow{\text{Bob}, A, r}$	$K=A^X$
		$k_2=H_2(Y,A,K)$
k_2 correct?	$\xleftarrow{k_2, e}$	$0 \le e \le 2^\kappa$
$s = t - xe \bmod q$	$\xrightarrow{s}$	$H_1(g^s y^e, Y,A,K)=r$?
	$k=H_0(Y,A,K)$	

- Implicitly
  - $H_2(Y,A,\mathrm{CDH}(Y,A)) \leftarrow k_2$
  - $H_0(Y,A,\mathrm{CDH}(Y,A)) \leftarrow k$

the simulation of the random oracles requires an access to a DDH-oracle:

to a query (Y,A,V)

- one first checks whether $V = \mathrm{CDH}(Y,A)$
- and then can give a consistent answer

The Diffie-Hellman Problems

- Computational

  Given $A=g^a$ and $B=g^b$
  Compute $\mathrm{DH}(A,B) = C=g^{ab}$

- Decisional

  Given A, B and C in $<g>$
  Decide whether $C = \mathrm{DH}(A,B)$

- Gap

  Solve the computational problem,
  with access to a decisional oracle

Security Result
equivalence

Alice $(x, y = g^x)$		Bob $(X, Y = g^X)$
$a \in \mathbf{Z}_q$, $A=g^a$, $t \in \mathbf{Z}_q$, $T=g^t$		
$K=Y^a$		
$k=H_0(Y,A,K)$		
$r=H_1(T,Y,A,K)$		
$k_2=H_2(Y,A,K)$	Bob, A, r $\longrightarrow$	$K=A^X$
		$k_2=H_2(Y,A,K)$
k_2 correct?	$\longleftarrow$ k_2, e	$0 \leq e \leq 2^\kappa$
$s = t - xe \bmod q$	s $\longrightarrow$	$H_1(g^s y^e,Y,A,K)=r$?
	$k=H_0(Y,A,K)$	

- We can prove that with a DDH oracle, one can perform all the simulations

$\Rightarrow$ answering the **test**-query is harder than solving the GDH problem

- Bob is actually a DDH oracle:

  given an instance (U,Y,W) to the DDH
  - one sends U to Bob, getting back k_2
  - one checks whether $k_2=H_2(Y,U,W)$

Advanced Properties

- Mutual authentication

  the key confirmation flows +
  - the Schnorr's identification provides the Client-to-Server authentication
  - the El Gamal decryption provides the Server-to-Client authentication
- Partial Forward Secrecy:
  - corrupted server: any session key derived from the communication: $k_2=H_2(Y,A, A^X)$
  - corrupted client: forward-secrecy guaranteed

Efficiency

Alice $(x, y = g^x)$		Bob $(X, Y = g^X)$
$a \in \mathbf{Z}_q$, $A = g^a$, $t \in \mathbf{Z}_q$, $T = g^t$		
$K = Y^a$		
$k = H_0(Y,A,K)$		
$r = H_1(T,Y,A,K)$		
$k_2 = H_2(Y,A,K)$	Bob, A, r $\longrightarrow$	$K = A^X$
		$k_2 = H_2(Y,A,K)$
k_2 correct?	$\longleftarrow$ k_2, e	$0 \le e \le 2^\kappa$
$s = t - xe \bmod q$	s $\longrightarrow$	$H_1(g^s y^e, Y,A,K) = r$?
	$k = H_0(Y,A,K)$	

- computational cost
  - client:
    - off-line: 2 exponentiations
    - off-line (known server): 1 exp. + 3 hashing
    - on-line: 1 hashing + 1 modular add-mult
    - Improvement: using GPS, instead of Schnorr
  - server: 3 exp. + 3 hashing
- communication cost

$|A|+|r|+|k_2|+|e|+|s| = |G| + 3 \times 80 + |q|$ bits

$\Rightarrow$ about 70 Bytes using elliptic curves

Conclusion

New Key-Agreement scheme
which provides

- semantic security of the session key
- mutual authentication
- partial forward-secrecy
  w.r.t. the corruption of the client
- low-power client
  - only one on-line add-multiplication
  - less than 70 bytes of communication