Authenticated commutator key-agreement protocol

Alexander Ushakov
Stevens Institute of Technology
sasha.ushakov@gmail.com

May 2, 2012

Commutators

$(G, \cdot)$ – group.

Definition

For $a, b \in G$ define:
$b^{-1}ab$ called the conjugate of a by b
$a^{-1}b^{-1}ab$ called the commutator of a and b.

Notation:
$[a, b] = a^{-1}b^{-1}ab$
$a^b = b^{-1}ab$.
We use vector notation $\overline{a} = (a_1, \ldots, a_n)$ for tuples of elements in G.
$\overline{a}^b$ denotes $(a_1^b, \ldots, a_n^b)$.

Commutator identity

Let $g \in G$. Then given

- $\overline{a} = (a_1, \ldots, a_n)$ and $\overline{a}' = (a'_1, \ldots, a'_n)$, where $a'_i = g^{-1} a_i g$,
- $a = a_{i_1}^{\varepsilon_1} \ldots a_{i_k}^{\varepsilon_k}$.

one can compute

$$[g, a] = \left(a_{i_1}'^{\varepsilon_1} \ldots a_{i_k}'^{\varepsilon_k}\right)^{-1} \cdot a_{i_1}^{\varepsilon_1} \ldots a_{i_k}^{\varepsilon_k}$$
$$[a, g] = [g, a]^{-1}.$$

Knowledge of g is not necessary to compute $[g, a]$!

Commutator key-agreement protocol (ephemeral CKA)

 I. Anshel, M. Anshel, and D. Goldefeld, *An algebraic method for public-key cryptography*, (1999).

Fix a group G and tuples $\overline{a} = (a_1, \ldots, a_n)$ and $\overline{b} = (b_1, \ldots, b_n)$.

	Alice	Bob
private key	$a = a_{i_1}^{\varepsilon_1} \ldots a_{i_k}^{\varepsilon_k}$	$b = b_{i_1}^{\varepsilon_1} \ldots a_{i_k}^{\varepsilon_k}$
public key	$\overline{b}^a$	$\overline{a}^b$

$$\Downarrow$$

$$K = [a, b]$$

- For Alice: $K = [a, b]$.
- For Bob: $K = [a, b]$.

The CKA-problem

The tuples $\overline{a}, \overline{b}, \overline{a}^b, \overline{b}^a$ uniquely define the commutator $K = [a, b]$. The problem of computing $K(\overline{a}, \overline{b}, \overline{a}^b, \overline{b}^a)$ is called the commutator key-agreement problem.

(Simultaneous conjugacy search problem) Given $(a_1, \ldots, a_n)$ *and* $(a'_1, \ldots, a'_n)$ *find any element* $y \in G$ *satisfying* $a'_i = y^{-1} a_i y$.

CKA-problem is treated as follows:

1. find a conjugator y for the tuples $\overline{a}$ and $\overline{a}'$;
2. find a conjugator x for the tuples $\overline{b}$ and $\overline{b}'$;
3. compute $K' = [x, y]$ and hope that $K' = K$.

Surprisingly this approach works in practice:

- length-based attacks,
- attacks using automatic structure of the braid group (braid group was initially proposed to be used as a platform group).

The CKA-problem

The tuples $\overline{a}, \overline{b}, \overline{a}^b, \overline{b}^a$ uniquely define the commutator $K = [a, b]$. The problem of computing $K(\overline{a}, \overline{b}, \overline{a}^b, \overline{b}^a)$ is called the commutator key-agreement problem.

> *(Simultaneous conjugacy search problem)* *Given* $(a_1, \ldots, a_n)$ *and* $(a'_1, \ldots, a'_n)$ *find any element* $y \in G$ *satisfying* $a'_i = y^{-1} a_i y$.

CKA-problem is treated as follows:

1. find a conjugator y for the tuples $\overline{a}$ and $\overline{a}'$;
2. find a conjugator x for the tuples $\overline{b}$ and $\overline{b}'$;
3. compute $K' = [x, y]$ and hope that $K' = K$.

Surprisingly this approach works in practice:

- length-based attacks,
- attacks using automatic structure of the braid group (braid group was initially proposed to be used as a platform group).

The CKA-problem

The tuples $\overline{a}, \overline{b}, \overline{a}^b, \overline{b}^a$ uniquely define the commutator $K = [a, b]$. The problem of computing $K(\overline{a}, \overline{b}, \overline{a}^b, \overline{b}^a)$ is called the commutator key-agreement problem.

> *(Simultaneous conjugacy search problem)* *Given* $(a_1, \ldots, a_n)$ *and* $(a'_1, \ldots, a'_n)$ *find any element* $y \in G$ *satisfying* $a'_i = y^{-1} a_i y$.

CKA-problem is treated as follows:

1. find a conjugator y for the tuples $\overline{a}$ and $\overline{a}'$;
2. find a conjugator x for the tuples $\overline{b}$ and $\overline{b}'$;
3. compute $K' = [x, y]$ and hope that $K' = K$.

Surprisingly this approach works in practice:

- length-based attacks,
- attacks using automatic structure of the braid group (braid group was initially proposed to be used as a platform group).

Large subgroup attack

V. Shpilrain and A. Ushakov, The conjugacy search problem in public key cryptography: unnecessary and insufficient, *2004.*

In general $K' \neq K$. To ensure that $K' = K$ one has to solve a (presumably) harder problem:

> *(SCSP relative to a subgroup) Given $\overline{a}$ and $\overline{a}'$ find any $y \in \langle b_1, \ldots, b_n \rangle$ satisfying $\overline{a}' = \overline{a}^y$.*

The decision version of this problem was recently proved to be unsolvable for braid groups.

For the original parameter values it was shown that:

- *Elements of public tuples often generate the whole group (large subgroup attack);*
- *SCSP and SCSP relative to a subgroup are equivalent;*
- *An evil party, say Bob, can choose his tuple $\overline{b}$ so that it will be easy to recover Alice's private key.*

A. Miasnikov, V. Shpilrain and A. Ushakov, Random subgroups of braid groups: an approach to cryptanalysis of a braid group based cryptographic protocol, *(2006).*

Authenticated CKA protocol

Fix a group G and tuples $\overline{a} = (a_1, \ldots, a_n)$ and $\overline{b} = (b_1, \ldots, b_n)$.

	Alice	Bob
static private key	$a \in F(\overline{a})$	$b \in F(\overline{b})$
static public key	$\overline{b}^a$	$\overline{a}^b$
ephemeral private key	$x \in F(\overline{a})$	$y \in F(\overline{b})$
ephemeral public key	$\overline{b}^x$	$\overline{a}^y$
shared key	$a^{-1}b^{-1}[xa^{-1}, yb^{-1}]ab$	

$$\begin{aligned} K =& a^{-1}b^{-1}[xa^{-1}, yb^{-1}]ab \\ =& [a,b]\cdot[b,x]\cdot[x,y]\cdot[y,a]\cdot[a,b] \\ \text{For Alice:}\quad & [a,b]\cdot[b,x]\cdot[x,y]\cdot[y,a]\cdot[a,b] \\ \text{For Bob:}\quad & [a,b]\cdot[b,x]\cdot[x,y]\cdot[y,a]\cdot[a,b]. \end{aligned}$$

To impersonate Bob, Eve should be able to solve the following computational problem:

(The authentication problem)

For any Alice's ephemeral public key $\overline{b}^x$

- *reply with some tuple $\overline{a}^y$;*
- *compute the shared key $a^{-1}b^{-1}[xa^{-1}, yb^{-1}]ab$.*

Authentication problem

$$\begin{aligned} K =& a^{-1}b^{-1}[xa^{-1}, yb^{-1}]ab \\ =& [a,b]\cdot[b,x]\cdot[x,y]\cdot[y,a]\cdot[a,b]. \end{aligned}$$

Theorem

The CKA-problem for G can be efficiently solved if and only if the authentication problem for G can be efficiently solved.

Proof.

"$\Rightarrow$" Assume we can efficiently solve CKA-problem. To solve authentication problem

- Eve generates random y,
- sends $\overline{a}^y$ to Alice,
- computes all the commutators in the product $K = [a,b]\cdot[b,x]\cdot[x,y]\cdot[y,a]\cdot[a,b]$.

"$\Leftarrow$" Assume we can solve the authentication problem. Then for a particular tuple $\overline{a}^y = \overline{a}^b$ we have $K = [a,b]$. Hence, we can solve the CKA-problem. $\square$

Full-forward secrecy

Assume that a passive adversary Eve learned the static private keys $a \in F(\bar{a})$ and $b \in F(\bar{b})$.

Theorem

If Eve can compute the shared key K for a new session between Alice and Bob with (unknown) ephemeral keys x and y, then she can compute $[x, y]$, i.e., can solve an instance of the CKA-problem.

Eve can compute $[x, y]$ as follows:

$$[x, y] = b^{-1}[b^{-1}, x]a^{-1}ba \cdot K \cdot b^{-1}a^{-1}b[y, a^{-1}]a,$$

where $K = a^{-1}b^{-1}[xa^{-1}, yb^{-1}]ab$.

Known-key security

Let $K_1, \ldots, K_n$ be session keys for Alice and Bob. Assume that there exists a polynomial time algorithm which solves the following problem:

$$\begin{cases} K_1, \ldots, K_n \\ \text{public info for } n+1 \text{ run} \end{cases} \quad \Rightarrow \quad K_{n+1}. \tag{1}$$

Then, using the identity

$$[x, y] = b^{-1}[b^{-1}, x]a^{-1}ba \cdot K \cdot b^{-1}a^{-1}b[y, a^{-1}]a.$$

we can solve the following problem

$$\begin{cases} [x_1, y_1], \ldots, [x_n, y_n] \\ \text{public info} \end{cases} \quad \Rightarrow \quad [x_{n+1}, y_{n+1}],$$

The ephemeral keys x_{n+1} and y_{n+1} are chosen randomly and independently from the key space. Hence, there exists a polynomial time algorithm solving the problem

$$\text{public info} \Rightarrow [x_{n+1}, y_{n+1}]$$

Thus, there exists a polynomial time algorithm solving the CKA-problem.

Key-compromise impersonation

	Alice	Bob
static private key	$a \in F(\overline{a})$	$b \in F(\overline{b})$
static public key	$\overline{b}^a$	$\overline{a}^b$
ephemeral private key	$x \in F(\overline{a})$	$y \in F(\overline{b})$
ephemeral public key	$\overline{b}^x$	$\overline{a}^y$
shared key	$a^{-1}b^{-1}[xa^{-1}, yb^{-1}]ab$	

If Eve learned Alice's static private key, then she can authenticate to Bob as Alice. The situation when knowledge of Alice's static private key allows Eve authenticate to Alice as Bob is called key-compromise impersonation.

The following scenario leads to key-compromise impersonation in our protocol.

- Eve sends $\overline{a}^b$ (known publicly) as an ephemeral public key to Alice.
- This makes $K = [a, b]$.
- Eve does not know b, but she knows a and, hence, can compute $[a, b]$.

Thus, Eve can impersonate Bob to Alice.

Ad-hoc fix: Alice should not accept the shared key $K = [a, b]$.

Key-compromise impersonation

	Alice	Bob
static private key	$a \in F(\overline{a})$	$b \in F(\overline{b})$
static public key	$\overline{b}^a$	$\overline{a}^b$
ephemeral private key	$x \in F(\overline{a})$	$y \in F(\overline{b})$
ephemeral public key	$\overline{b}^x$	$\overline{a}^y$
shared key	$a^{-1}b^{-1}[xa^{-1}, yb^{-1}]ab$	

If Eve learned Alice's static private key, then she can authenticate to Bob as Alice. The situation when knowledge of Alice's static private key allows Eve authenticate to Alice as Bob is called key-compromise impersonation.

The following scenario leads to key-compromise impersonation in our protocol.

- Eve sends $\overline{a}^b$ (known publicly) as an ephemeral public key to Alice.
- This makes $K = [a, b]$.
- Eve does not know b, but she knows a and, hence, can compute $[a, b]$.

Thus, Eve can impersonate Bob to Alice.

Ad-hoc fix: Alice should not accept the shared key $K = [a, b]$.

Key-compromise impersonation

	Alice	Bob
static private key	$a \in F(\overline{a})$	$b \in F(\overline{b})$
static public key	$\overline{b}^a$	$\overline{a}^b$
ephemeral private key	$x \in F(\overline{a})$	$y \in F(\overline{b})$
ephemeral public key	$\overline{b}^x$	$\overline{a}^y$
shared key	$a^{-1}b^{-1}[xa^{-1}, yb^{-1}]ab$	

If Eve learned Alice's static private key, then she can authenticate to Bob as Alice. The situation when knowledge of Alice's static private key allows Eve authenticate to Alice as Bob is called key-compromise impersonation.

The following scenario leads to key-compromise impersonation in our protocol.

- Eve sends $\overline{a}^b$ (known publicly) as an ephemeral public key to Alice.
- This makes $K = [a, b]$.
- Eve does not know b, but she knows a and, hence, can compute $[a, b]$.

Thus, Eve can impersonate Bob to Alice.

Ad-hoc fix: Alice should not accept the shared key $K = [a, b]$.

Zero-knowledge authentication protocol

	Alice the prover	Bob the verifier
static private key	$a \in F(\overline{a})$	$b \in F(\overline{b})$
static public key	$\overline{\alpha} = \overline{b}^{a}$	$\overline{\beta} = \overline{a}^{b}$
commitment	$\overline{\gamma} = \overline{\alpha}^{x}$	

A single round of the protocol is performed as follows:

- Alice chooses a random $x \in F(\overline{a})$ and sends the commitment $\overline{\gamma} = \overline{\alpha}^{x}$ to Bob.
- Bob replies with a random value $c \in \{0, 1\}$ called the challenge.
- If $c = 0$, then Alice sends the element $z = x \in F(\overline{a})$ in which case Bob checks if the equality $\overline{\gamma} = \overline{\alpha}^{z}$ is satisfied.
- If $c = 1$, then Alice sends the commutator $K = [b, ax]$ in which case Bob checks if $K = [b, ax]$ is correct.

Theorem

Completeness property holds.

Proof.

Both, Alice and Bob can compute $[b, ax]$. □

Zero-knowledge authentication protocol

	Alice the prover	Bob the verifier
static private key	$a \in F(\overline{a})$	$b \in F(\overline{b})$
static public key	$\overline{\alpha} = \overline{b}^{a}$	$\overline{\beta} = \overline{a}^{b}$
commitment	$\overline{\gamma} = \overline{\alpha}^{x}$	

A single round of the protocol is performed as follows:

- Alice chooses a random $x \in F(\overline{a})$ and sends the commitment $\overline{\gamma} = \overline{\alpha}^{x}$ to Bob.
- Bob replies with a random value $c \in \{0, 1\}$ called the challenge.
- If $c = 0$, then Alice sends the element $z = x \in F(\overline{a})$ in which case Bob checks if the equality $\overline{\gamma} = \overline{\alpha}^{z}$ is satisfied.
- If $c = 1$, then Alice sends the commutator $K = [b, ax]$ in which case Bob checks if $K = [b, ax]$ is correct.

Theorem

Zero-knowledge property holds.

Proof.

Bob can simulate runs of this protocol. Hence, information sent to Bob reveals no additional knowledge of a to Bob. □

Soundness property

Assume that CKA-problem is hard. Hence, SCSP in G is hard too.

	Alice the prover	Bob the verifier
static private key	$a \in F(\overline{a})$	$b \in F(\overline{b})$
static public key	$\overline{\alpha} = \overline{b}^{a}$	$\overline{\beta} = \overline{a}^{b}$
commitment	$\overline{\gamma} = \overline{\alpha}^{x} = \overline{b}^{ax}$	

Response is $z = x$ if $c = 0$ and $K = [b, ax]$ if $c = 1$.

If Eve can guess the value of c, then she can correctly respond with a proper answer to Bob's challenge.

On the other hand if she improperly guesses c, then:

- $(c = 0)$ Eve has to solve the SCSP for $(\overline{\alpha}, \overline{\gamma})$ which is hard.
- $(c = 1)$ Eve has to find the commutator $[b, ax]$. If she would be able to do that, then she would be able to compute $[b, a] = x \cdot [x, b] \cdot K \cdot x^{-1}$, i.e., to solve an instance of CKA-problem

Hence, the soundness error of a single round is $1/2$ and repeating one round 100 times we can make it 2^{-100}.

Soundness property

Assume that CKA-problem is hard. Hence, SCSP in G is hard too.

	Alice the prover	Bob the verifier
static private key	$a \in F(\overline{a})$	$b \in F(\overline{b})$
static public key	$\overline{\alpha} = \overline{b}^{a}$	$\overline{\beta} = \overline{a}^{b}$
commitment	$\overline{\gamma} = \overline{\alpha}^{x} = \overline{b}^{ax}$	

Response is $z = x$ if $c = 0$ and $K = [b, ax]$ if $c = 1$.

If Eve can guess the value of c, then she can correctly respond with a proper answer to Bob's challenge.

On the other hand if she improperly guesses c, then:

- $(c = 0)$ Eve has to solve the SCSP for $(\overline{\alpha}, \overline{\gamma})$ which is hard.
- $(c = 1)$ Eve has to find the commutator $[b, ax]$. If she would be able to do that, then she would be able to compute $[b, a] = x \cdot [x, b] \cdot K \cdot x^{-1}$, i.e., to solve an instance of CKA-problem

Hence, the soundness error of a single round is $1/2$ and repeating one round 100 times we can make it 2^{-100}.

Soundness property

Assume that CKA-problem is hard. Hence, SCSP in G is hard too.

	Alice the prover	Bob the verifier
static private key	$a \in F(\overline{a})$	$b \in F(\overline{b})$
static public key	$\overline{\alpha} = \overline{b}^{a}$	$\overline{\beta} = \overline{a}^{b}$
commitment	$\overline{\gamma} = \overline{\alpha}^{x} = \overline{b}^{ax}$	

Response is $z = x$ if $c = 0$ and $K = [b, ax]$ if $c = 1$.

If Eve can guess the value of c, then she can correctly respond with a proper answer to Bob's challenge.
On the other hand if she improperly guesses c, then:

- $(c = 0)$ Eve has to solve the SCSP for $(\overline{\alpha}, \overline{\gamma})$ which is hard.
- $(c = 1)$ Eve has to find the commutator $[b, ax]$. If she would be able to do that, then she would be able to compute $[b, a] = x \cdot [x, b] \cdot K \cdot x^{-1}$, i.e., to solve an instance of CKA-problem

Hence, the soundness error of a single round is $1/2$ and repeating one round 100 times we can make it 2^{-100}.

Homework

Study computational properties of the CKA-problem in different groups.

Thank you