The Diffie–Hellman Problem in Lie Algebras

Beata Rafalska

University of Warmia and Mazury
Olsztyn, Poland
winka@ols.vectranet.pl

Abstract. Cryptography in its present state relies increasingly on complex mathematical theories, e.g., elliptic curves, group theory, etc. We address in this article the problem of proxy signatures and we set this problem in the framework of Lie algebras. We show how to use a chosen maximal set of differentiable automorphisms in order to carry out the task of proxy signing. We also show possible attacks and the way to protect against them.

Keywords: cryptography, proxy signatures, attacks, Lie algebras, differentiable maps.

To the memory of Professor Zdzisław Pawlak

1 Introduction

1.1 Lie Algebras

Definition 1. *Let* $\mathbf{K}$ *be a commutative field. We say that a linear space* L *over* $\mathbf{K}$ *is a Lie algebra, if there is a bilinear operation*

$$L \times L \ni (a, b) \rightarrow [a, b] \in L,$$

called the Lie bracket (Lie product), satisfying the conditions:

(a) $[a, b] = -[b, a]$, *(anty-symmetry)*
(b) $\Big[a, [b, c]\Big] + \Big[b, [c, a]\Big] + \Big[c, [a, b]\Big] = 0.$

Condition (b) is called the Jacobi identity.

Obviously, condition (a) can be rewritten in an equivalent form: $[a, a] = 0$.

Fact 1. The set of all endomorphisms $End(X)$, where X is a linear space, is a Lie algebra with bracket defined as:

$$[f, g] = fg - gf,$$

for $f, g \in End(X)$.

M. Kryszkiewicz et al. (Eds.): RSEISP 2007, LNAI 4585, pp. 622–629, 2007.

Definition 2. *Let A be a linear algebra over* $\mathbf{K}$, *with an operation* $A \times A \ni (a, b) \to a \star b \in A$. *An endomorphism* $\alpha : A \to A$ *is called a differentiation, if for* $a, b \in A$

$$\alpha(a \star b) = \alpha(a) \star b + a \star \alpha(b).$$

The set of all differentiations over the algebra A *is denoted as* $Der(A)$.

Theorem 1. $Der(A)$ *is a Lie subalgebra of* $End(A)$.

Proof. Proof of this theorem is a simple algebraic computing.

Now, we consider some maximal set of pair-wise commuting differentiations and we denote it as $CDer(A)$, clearly, such a set is non-unique.

Fact 2. There exists a non-empty set $CDer(A)$.

Proof. Let $\alpha \in Der(A)$. Then α is commutative with α. Composite $\alpha\alpha \in Der(A)$, but α is commutative with $\alpha\alpha$ too, etc. So that $\alpha, \alpha\alpha, \ldots, \alpha\alpha \ldots \alpha$ are pair-wise commuting, and the set $\{\alpha^n : n = 1, 2, \ldots\}$ extends to a maximal set $CDer(A)$.

Fact 1.2 shows that a set $CDer(A)$ exists.

Theorem 2. $CDer(A)$ *is an algebra.*

Proof. Proof of this theorem is a simple algebraic computing.

1.2 The Diffie-Hellman Problem

Let us recall the definition of discrete logarithm from [2].

Let $\mathbb{F}_p^* = (\mathbb{Z}/p\mathbb{Z})^* = \{1, 2, \cdots, p-1\}$ be the multiplicative group of integer numbers modulo a prime number p. Let $g \in \mathbb{F}_p^*$ be a fixed element. *The discrete logarithm problem* in $\mathbb{F}_p^*$ at the base g is the problem of finding for the fixed $y \in \mathbb{F}_p^*$ of a natural number x, such that $y = g^x$ modulo p.

We remind now the Diffie-Hellman key exchange system (see [2]). Assume, that Alice and Bob want to agree on a secret key in any cryptosystem with private keys. Keys exchange occurs over an insecure communication channel, so that an adversary Charlie knows the substance of all communicates, which are sent between Alice and Bob. Alice and Bob agree at first on a large prime number p and a base g. Then Alice in secret picks a random natural number $k_A < p$ (of the same order as p) and computes the remainder from division of g^{k_A} by p and the result is sent to Bob. Bob proceeds in a similar manner and sends to Alice $g^{k_B} \in \mathbb{F}_p^*$ keeping k_B secret. The key agreed upon will be the number $g^{k_A k_B}$. The problem which Charlie is facing, is the *Diffie-Hellman problem:* having the data $g, g^{k_A}, g^{k_B} \in \mathbb{F}_p^*$, compute $g^{k_A k_B}$. It is worth to notice, that everyone who can solve the discrete logarithm problem, can solve the Diffie-Hellman problem, too.

In [1], the author generalizes the discrete logarithm problem and the Diffie-Hellman problem to cyclic groups. We define the *general discrete logarithm problem* as follows: Let $G =< a_1, a_2, \cdots, a_n >$ be a cyclic group and $f : G \to G$

be a non identity automorphism. General discrete logarithm problem is to find $f(b)$ for any $b \in G$ having given $f(a)$ for some $a \in G$. In other words the general discrete logarithm problem is to find the automorphism f knowing its action on only one element.

Suppose now, that we have two non identity automorphisms $\varphi, \psi : G \to G$ and that we know $a, \varphi(a)$ i $\psi(a)$. Then, the *general Diffie-Hellman problem* is to find $\varphi(\psi(a))$.

2 Diffie-Hellman Problem in Lie Algebras

2.1 Key Exchange System

Alice and Bob want to agree on a private key for exchange of information over an insecure channel. They agree on a Lie algebra L, a set $CDer(A)$, and an element $g \in L$. Alice picks randomly a differentiation $\alpha \in CDer(L)$, and an element $a \in L$. She sends Bob the value $\alpha([g, a])$. Bob picks at random a differentiation $\beta \in CDer(L)$, and he sends to Alice the value $\beta(\alpha([g, a]))$. Alice determines α^{-1} and computes $\beta([g, a])$:

$$\alpha^{-1}(\beta(\alpha([g, a]))) = \alpha^{-1}(\alpha(\beta([g, a])))) = \alpha^{-1}\alpha(\beta([g, a])) = \beta([g, a])$$

Now, Alice randomly chooses a next differentiation $\gamma \in CDer(L)$, and computes $\gamma(\beta([g, a]))$ and then the result is sent to Bob. Alice can compute $\gamma([g, a])$ too, and Bob, knowing the differentiation β, computes β^{-1} and finds $\gamma([g, a])$, (in analogy to Alice's computation). The value $\gamma([g, a])$ is their fixed key.

2.2 System Analysis

Notice, that Alice doesn't show a, so the adversary Charlie knows $L, g, \alpha([g, a])$, $\beta(\alpha([g, a]))$ and $\gamma(\beta([g, a]))$. To find $\gamma([g, a])$ is a problem which incorporates the general discrete logarithm problem with the general Diffie-Hellman problem described in [2]. We can restrict the problem to finding a differentiation β (exactly β^{-1}) having as information $L, g, \alpha([g, a]), \beta(\alpha([g, a]))$ and $\gamma(\beta([g, a]))$. The task of finding β, can be reduced to computation α. Finally, the problem of finding the key can be reduced to computing of the differentiation α knowing only the action of α on one element $[g, a]$. An additional impediment for Charlie is the fact, that he doesn't know the element a, which Alice doesn't show.

2.3 Sending Information

For simplifying of notation, we mark the earlier fixed key as $x = \gamma([g, a])$. Suppose, that Alice wants to send to Bob an information m already converted to an element from the Lie algebra L. Alice sends to Bob the element $y = [m, x]$. Bob knows the Lie algebra L, so he knows the Lie bracket and key x and he can compute the value m. The difficulty of this computation will depend on the specified Lie bracket and the internal multiplication in the algebra. For an

example, for Lie algebra with the standard commutator and anti-commutative internal multiplication, we have:

$$[m, x] = y$$

$$mx - xm = y$$

$$2mx = y$$

$$m = 2^{-1} y x^{-1}.$$

Adversary Charlie doesn't know the key x, so he can't decode the information m. Further, if Alice computes the information hash $H(m)$, and she sends to Bob the algebra element $z = [H(m), x]$, too, then Bob will be able to verify, whether he gets the information in an unspoiled form. Analogically, to decipher the information m, Bob will decifer the hashed message $H(m)$, and he compares whether what he has got is the same as the element $H(m)$, which he got from Alice. So, for increased security, Alice sends to Bob the pair (y, z).

3 Information Signature

In our algorithm, we can use the signature scheme with proxy signers, analogical to scheme described in [4].

3.1 Notation

We mark original signer as P_0 and proxy signers as $\{P_1, P_2, \cdots, P_m\}$. We suppose, that all signers P_i have private keys a_i and the corresponding public keys $A_i = [a_i, g]$, where g is Lie algebra's element, certified by the central authority for $i = 0, \cdots, m$. Let w be a message created by the original signer P_0. Moreover, we will assume that H i H_1 are some suitably chosen collision -free hash functions.

3.2 Group Secret Key Generation

P_0 prepares the information w, and chooses randomly an algebra element r and computes $R = [r, g]$. Next, he determines the value $H = H(w, R)$ of the collision-free hash function H. Having this data, P_0 computes the group secret key,

$$d = [a_o, H] + r.$$

We notice, that $d = d(R)$ and $d = d(a_0)$, where a_0 is a private key of P_0, thus only P_0 can compute d, moreover, the private key a_0 of signer P_0 is well protected by randomly behaving hash function H. The public verification that the signature is true is not difficult, too, because we have publicly known R and A_0 :

$$d = [a_0, H] + r \longrightarrow [d, g] = [[a_0, H], g] + [r, g] = [A_0, H] + R$$

$[d, g] = [[a_0, H] + r, g] = [a_0 H - H a_0, g] + [r, g] = [a_0 H, g] - [H a_0, g] + R$
$= [a_0, g]H - H[a_0, g] + R = [[a_0, g], H] + R = [A_0, H] + R.$

3.3 Group Secret Key Share

The original signer P_0 selects a polynomial

$$f_0(x) = c_{0(t-1)}x^{(t-1)} + \cdots + c_{01}x + d,$$

where each c_i for $i = 1, \cdots, t-1$ is a random algebra element. We see, that Lie multiplication of any element by itself results in 0, so we specify power as internal multiplication in algebra. Next, P_0 computes $C_{0i} = [c_{0i}, g]$ for $i = 1, \cdots, t-1$, and he sends it to proxy signers P_i.

$$Transfer : (\{C_{0i} = [c_{0i}, g] : i = 1, \cdots, t-1\}),$$

so that,

$$\begin{array}{lllll} f_0(x) = c_{0(t-1)}x^{(t-1} + \cdots + & c_{01}x + & d & & \\ \downarrow & \downarrow & \downarrow = a_0H + r & & \\ C_{0(t-1)} & C_{01} & HA_0 + R & & \end{array}$$

Let x_i be the public identity of P_i. Now, P_0 distributes the secret key $d_0 = f_0(0)$ distributing the values $y_{i0} = f_0(x_i)$ for each $P_i \in P$, and he sends them by secret channels.

$$P_0 \longmapsto f_0(x) = \sum_{i=1}^{t-1} c_{0i}x^i + d \qquad d = a_0H + r.$$

Each proxy signer can verify y_{i0} by the equation

$$[y_{i0}, g] = \underbrace{A_0H + R}_{[d,g]} + \sum_{j=1}^{t-1} x_i^j C_{0j}.$$

3.4 The Proxy Signature Generation

Now, each proxy signer P_i selects a secret polynomial

$$f_i(x) = c_{i(t-1)}x^{t-1} + \cdots + c_{i1}x + c_{i0} + a_i,$$

where c_{ik} for $k = 1, \cdots, t-1$ is a random Lie algebra's element and a_i is a secret key of P_i. Next, P_i computes and broadcasts $C_{ik} = [c_{ik}, g]$, for $k = 0, \cdots, t-1$.

$$Transfer : (C_{ik} : \{k = 0, \cdots, t-1\}, A_i)$$

P_i computes the value of the hash function $H_1 = H_1(w, R, M, B)$ too, where M is a message, which P_i wants to sign on behalf of the original signer P_0 and B is any subset of t (or more) proxy signers, and he computes the value $f_i(x_j)$ for $i \neq j$ and sends to P_j by the secret channel his part

$$y_{ji} := H_1 f_i(x_j), \qquad j = 1, \cdots t$$

$$P_i : y_{ji} = H_1 f_i(x_j) \longrightarrow P_j \qquad \forall j \neq i, P_j \in B.$$

Next, each signer P_j verifies the received values $f_i(x_j)$ from other $t-1$ proxy signers by the equation

$$[y_{ji}, g] = H_1(A_i + \sum_{k=0}^{t-1} x_j^k C_{ik}), \qquad \forall j \neq i, P_i \in B.$$

If all of the above equation hold, then each P_j computes his partial proxy signature from the received values as $s_j = \sum_{i=1}^{t} y_{ji}$.

$$P_1 : s_1 = \sum_{i=1}^{t} y_{1i} = \sum_{i=1}^{t} H_1 f_i(x_1) = H_1 f_1(x_1) + H_1 f_2(x_1) + \ldots + H_1 f_t(x_1) = H_1 f(x_1)$$

$$P_2 : s_2 = \sum_{i=1}^{t} y_{2i} = \sum_{i=1}^{t} H_1 f_i(x_2) = H_1 f_1(x_2) + H_1 f_2(x_2) + \ldots + H_1 f_t(x_2) = H_1 f(x_2)$$

$$\vdots$$

$$P_t : s_t = \sum_{i=1}^{t} y_{ti} = \sum_{i=1}^{t} H_1 f_i(x_t) = H_1 f_1(x_t) + H_1 f_2(x_t) + \ldots + H_1 f_t(x_t) = H_1 f(x_t)$$

This share has a value $H_1 f(x_j)$, where $f(x)$ is the virtual polynomial

$$\begin{array}{lccccc} f(x) = & x^{t-1}(\sum_{i=1}^{t} c_{i(t-1)}) + & \cdots + (\sum_{i=1}^{t} c_{i0}) + & (\sum_{i=0}^{t} a_{i0}) \\ \downarrow & \downarrow & \downarrow & \downarrow \\ F(x) = & x^{t-1} \underbrace{(\sum_{i=1}^{t} C_{i(t-1)})}_{:= C'_{t-1}} + \cdots + \underbrace{(\sum_{i=1}^{t} C_{i0})}_{:= C'_0} + & \underbrace{(\sum_{i=1}^{t})A_i}_{:= A'} \end{array}$$

The public obligations of signers group B are

$$Transfer : (\{C'_k : k = 0, 1, \cdots, t-1\}, A')$$

Next, each proxy signer P_j computes the threshold proxy signature on M as follows:

$$\sigma_j = \sigma_j(M, B, w, R) = y_{j0} + \sum_{i=1}^{t} y_{ji} = y_{j0} + s_j.$$

P_j sends by the secret channel the calculated σ_j for each $P_i \in B$. Now, each P_j does the test of the received shares:

$$[\sigma_j, g] = [y_{j0} + s_j, g] = [y_{j0} + \sum_{i=1}^{t} y_{ji}, g] = [y_{j0}, g] + [\sum_{i=1}^{t} y_{ji}, g]$$

$$= HA_0 + R + \sum_{k=1}^{t-1} x_j^k C_{0k} + \sum_{j=1}^{t} H_1(A_j + \sum_{k=0}^{t-1} x_j^k C_{jk}).$$

Finally, if all was correct, then the threshold proxy signature is the following: $(M, C'_0, A', \sigma, w, B)$, where $\sigma = d + H_1 f(0)$.

3.5 Verification of the Proxy Signature

Addressee of the message in the first step does verify correctness of the threshold (M, C_0', A', w, B) by the verifying equation:

$$[\sigma, g] = [d + H_1 f(0), g] = [d, g] + [H_1 f(0), g] = [f_0(0), g] + [H_1 f(0), g].$$

If this equation is true, then the addressee infers, that the proxy signature $(M, C_0', A', \sigma, w, B)$ is the proper proxy signature obtained from the delegation key of the original signer and that the set B consists of the actual proxy signers.

Next, the addressee computes:

$$f_0(0) = d = a_0 H + r,$$

$$[f_0(0), g] = [d, g] = A_0 H + R,$$

and

$$[H_1 f(0), g] = [H_1(w, R, M, B) f(0), g] = [H_1(\sum_{i=1}^{t} f_i)(0), g]$$

$$= [H_1 \sum_{i=1}^{t} c_{i0}, g] = H_1 \sum_{i=1}^{t} [c_{i0}, g] = H_1 \sum_{i=1}^{t} C_{i0} = H_1 F(0).$$

4 The Analysis of the Insider Attack

Suppose, that one from the pair proxy signer - insider attacker (without the loss of the generality, we agree that this is P_1) wants to get a threshold proxy signature on message M. While generating the proxy signature, P_1 does not broadcast his data C_{1k}, but he waits until will receives from remaining proxy signers their data C_{i1}. Now, P_1 computes the hash function $H_1 = H_1(w, R, M, B)$ and assign $f_1(x_j)$ for $i \neq j$. P_1 can compute $y_{j1} = H_1 f_1(x_j)$, but he can't compute s_1 which is indispensable to falsify the threshold proxy signature, because he does not know all y_{ji}. So, this attack isn't practical, let us suppose that proxy signers don't continue the broadcast of the data until they receive earlier obligations from all signers. Let's see now, what happens,when the insider attacks on the later transfer of the data, i.e. just during sending y_{ji}. Then the scheme generating the proxy signature would look as follows:

Each proxy signer P_i selects the secret polynomial $f_i(x) = c_{i(t-1)} x^{t-1} + \cdots + c_{i1} x + c_{i0} + a_i$, and they compute and broadcast $C_{ik} = [c_{ik}, g]$ for $k = 0, \cdots, t-1$. Later P_i computes the value of the hash function $H_1 = H_1(w, R, M, B)$ and determines $f_i(x_j)$ for $i \neq j$. At this moment, P_1 attacks and he doesn't broadcast his value y_{ji} but waits for values from remaining signers. In this way P_1 receives all values y_{1i} for $i \neq 1$ and he computes y_{11}. In this situation, after the verification of the data, P_1 can compute his part of the threshold proxy signature

$$s_1 = \sum_{i=1}^{t} y_{1i}.$$

So now P_1 computed his part of the threshold proxy signature s_1, but he has not broadcasted his data to remaining signers. Theoretically P_1 can privately compute y_1' and for this value the likely value s_1', so the threshold proxy signature is correct for y_1', however the counterfeited value y_1' will not pass verification conducts by remaining signers, because P_1 can not alter the sent earlier C_{1k}.

Finally, we see, that the scheme of the proxy signature is resistant to the attack by any insider signer if we suppose that proxy signers will not send data, until they not receive earlier obligations.

5 Conclusion

We have presented an approach to the Diffie-Hellman problem in Lie algebras, by exploiting sets of commutative differentiations. Our results generalize in a sense the approach in [1].

References

1. Mahalanobis, A.: Diffie-Hellman key exchange protocol and non-abelian nilpotent groups (2005) http://eprint.iacr.org/2005/110.ps
2. Koblitz, N.: Algebraiczne aspekty kryptografii, WNT, Warszawa (2000)
3. Wojtyński, W.: Grupy i algebry Liego. PWN, Warszawa (1986)
4. Pomykaa, J., Barabasz, S.: Eliptic Curve Based Threshold Proxy Signature Scheme with Known Signers. Fundamenta Informaticae 69(4), 411–425 (2006)
5. Goldwasser, S., Bellare, M.: Lecture Notes on Cryptography (1996) http://www-cse.ucsd.edu/~mihir/papers/gb.html
6. Desmedt, I.: Some Recent Research Aspect of Threshold Cryptography, Departament of Mathematics Royal Holloway University of London (1997) http://citeseer.ist.psu.edu/desmedt97some.html
7. Zhang, K.: Threshold Proxy Signature Schemes, Cambridge University Computer Laboratory (1997) http://citeseer.ist.psu.edu/zhang97threshold.html