On the Security of Stickel's Key Exchange Scheme

Michal Sramka
Center for Cryptology and Information Security,
Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431
sramka@math.fau.edu

Abstract

E. Stickel proposed a variation of the Diffie-Hellman key exchange scheme based on non-abelian groups, claiming that the underlying problem is more secure than the traditional discrete logarithm problem in cyclic groups. We show that the proposed scheme does not provide a higher level of security in comparison to the traditional Diffie-Hellman scheme.

1 Introduction

E. Stickel in [3] proposed a variation of the Diffie-Hellman key exchange scheme [1] to non-abelian groups. In particular, he described a key exchange protocol that uses exponentiations of two non-commuting group elements. The proposal also contains an implementation detail for a specific subgroup of a general linear group of prime degree n. Although the proposal lacks a rigorous security analysis, the author claims that a brute-force attack of an instance would require searching through a space of size $(2^n - 1)^2$.

In this paper, we show that the proposed key exchange scheme can, in fact, be successfully attacked with a considerably smaller complexity. In particular, we show that the scheme can be broken by searching through a space of cardinality $2^n - 1$. Also, for the general case with two non-commuting elements of order n_1 and n_2 ($n_1 \geq n_2$), we show that the worst-case time complexity of breaking the scheme is $O(n_1)$ group operations while requiring storage of $O(n_2)$ group elements.

We start by presenting a description of Stickel's key exchange scheme and the implementation details, then we show our main cryptanalytic results. We conclude with a generalization of our proposed method for factoring group elements.

JCMCC **66** (2008), pp. 151-159

1.1 Proposal

We firstly provide a description of the E. Stickel's general scheme [3] for exchanging a secret key between two parties (Scheme 1) and then describe some implementation details that lead to Scheme 2, a variation of the first scheme.

Let G be a non-abelian finite group and $\alpha, \beta \in G$ be two non-commuting elements. Let n_1 denote the order of α and n_2 the order of β. Both elements α and β are public information.

The key exchange protocol between Alice and Bob can be described in the following steps:

Scheme 1 (Stickel's key exchange scheme).

1. Bob randomly chooses integers r and s with $0 < r < n_1$, $0 < s < n_2$. The integers r and s are kept secret. Bob forms $\gamma := \alpha^r \beta^s$ and sends γ to Alice.

2. In a similar way, Alice chooses integers v and w with $0 < v < n_1$ and $0 < w < n_2$, which she keeps secret. Alice forms $\delta := \alpha^v \beta^w$ and sends δ to Bob.

3. Alice computes the key $\kappa := \alpha^v \gamma \beta^w$, and similarly, Bob computes the key κ as $\alpha^r \delta \beta^s$.

Note that an arbitrary element $\tau \in G$ known to both parties can be placed in the middle of the products γ and δ, respectively, to obtain new $\gamma := \alpha^r \tau \beta^s$ and $\delta := \alpha^v \tau \beta^w$.

The correctness of the protocol is obvious in both cases.

1.1.1 Implementation details

An implementation of the general key exchange scheme over certain subgroups of the general linear matrix groups leads to a large family of concrete instances.

Let n be a prime such that $2^n - 1$ is also a prime. Primes of this form $2^n - 1$ are called *Mersenne primes*, and n in this case is called a *Mersenne exponent*, e.g., $n = 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, \ldots$.

Let $p(x)$ and $q(x)$ be two inequivalent irreducible polynomials of degree n over the finite field $GF(2)$. Let C and D be the companion matrices of $p(x)$ and $q(x)$, respectively. That is, if $p(x) := x^n + c_{n-1}x^{n-1} + \cdots + c_1 x +$

$c_0 \in GF(2)[x]$, then the corresponding $n \times n$ *companion matrix* is

$$C := \begin{pmatrix} 0 & 0 & \dots & \dots & c_0 \\ 1 & 0 & 0 & \ddots & c_1 \\ 0 & 1 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 & c_{n-2} \\ 0 & 0 & \dots & 1 & c_{n-1} \end{pmatrix}.$$

Finally, let $\mathbb{F}$ be an extension field of $GF(2)$, and let T_1 and T_2 be arbitrary invertible $n \times n$ matrices over $\mathbb{F}$; i.e., $T_1, T_2 \in GL_n(\mathbb{F})$. The purpose of these matrices is to render eigenvalue/eigenvector attacks infeasible. See the original proposal [3] for details. For our purposes it suffices to know that the field $\mathbb{F}$ is a finite extension of $GF(2)$ of degree at least 2.

Now, both matrices C and D have prime order $2^n - 1$, $CD \neq DC$ (as long as $p(x) \neq q(x)$), and so the cardinality of the set $\{C^i T_1 T_2 D^j \mid i, j \in \mathbb{Z}\}$ is $(2^n - 1)^2$.

An adaptation of Scheme 1 to these settings is straightforward: C and D play the role of α and β, resp. (with $n_1 = n_2 = 2^n - 1$), and $T_1 T_2$ plays the role of τ. However, a slightly different variant of Scheme 1 was proposed in [3] by E. Stickel. Namely, two additional secret scalars a and b were added:

Scheme 2 (Stickel's key exchange scheme (2)).

1. Bob randomly chooses integers r and s with $0 < r, s < 2^n - 1$, and a scalar $b \in \mathbb{F}$. The parameters r, s, and b are kept secret. Bob computes $F := bC^r T_1 T_2 D^s$ and sends this matrix F to Alice.

2. Similarly, Alice randomly chooses secret integers v and w with $0 < v, w < 2^n - 1$, and a secret scalar $a \in \mathbb{F}$. Alice forms $H := aC^v T_1 T_2 D^w$ and sends H to Bob.

3. Alice computes the key (matrix) $K := aC^v F D^w$, and similarly, Bob computes the key K as $bC^r H D^s$.

1.2 Cryptanalysis

The security of Scheme 1, that is, the security of the final exchanged key κ, is based on the fact that an opponent is unable to factorize γ or δ into α's and β's. Here we are using notation as in Section 1.1.

However, a factorization of γ or δ can be successfully obtained by knowing only one out of the four secret parameters r, s, v, or w: Without loss of generality, suppose an opponent learns the value of r. Then, the opponent

can compute the key κ from r and the public values as $\alpha^r \delta \alpha^{-r} \gamma$. Similarly, knowledge of any one of the remaining three yields the key. Therefore the security of the scheme depends on the knowledge of just one of the four secret parameters.

Let $Q := \langle \alpha \rangle \cap \langle \beta \rangle$ be a subgroup of G. Then, from elementary theory of cyclic groups [2], it follows that $t := n_1/|Q|$ is the smallest positive exponent such that $\alpha^t \in Q$, or equivalently t is the smallest positive exponent such that $\alpha^t \in \langle \beta \rangle$. Now, consider the element $\mu := \alpha^{-i}\gamma = \alpha^{r-i}\beta^s$ for some integer i. It is easy to see that $\mu \in \langle \beta \rangle$ if and only if $r - i \equiv 0 \pmod{t}$. And so, r is one of the $|Q|$ numbers $i + kt$, where $k = 0, 1, 2, \ldots, |Q| - 1$. The correct r can be then obtained by constructing a key κ' for each possibility $i + kt$ and verifying whether κ' is the correct key.

The following algorithm implements these ideas:

Algorithm 1.
Input: α, β, γ and n_1
Output: (i, t, m) such that $r = i + kt$ for some $k \in \{0, 1, \ldots, m-1\}$

1. Set $m := |\langle \alpha \rangle \cap \langle \beta \rangle|$ and $t := n_1/m$.
2. For $i = 1, 2, \ldots, t$ do
   (a) If $\alpha^{-i}\gamma \in \langle \beta \rangle$ then output (i, t, m) and stop.

1.2.1 The worst-case complexity analysis

The worst case complexity of Algorithm 1 occurs when the subgroups generated by α and β intersect trivially (i.e., $|\langle \alpha \rangle \cap \langle \beta \rangle| = 1$) or if it is too costly to compute the order of $\langle \alpha \rangle \cap \langle \beta \rangle$. In either case, the value of t in Algorithm 1 becomes n_1.

On the other hand, group membership testing can be a hard problem. For the worst-case analysis, we will assume that the subgroup $\langle \beta \rangle$ must be stored as a list – possibly as a sorted list, so that we can test for membership using the binary search method.

Consider the following algorithm, equivalent to Algorithm 1, but rewritten for the purpose of complexity estimation.

Algorithm 2 (The worst-case complexity of Algorithm 1).
Input: $\alpha, \beta, \gamma, n_1$, and n_2
Output: r

1. Set $\sigma := \beta$ and table T to be empty.
2. For $i = 1, 2, \ldots, n_2$ do

(a) Using binary search, determine where to insert element σ into table T, such that after the insertion table T remains sorted, and perform the insertion.

(b) Set $\sigma := \sigma\beta$.

3. Set $\omega := \alpha^{n_1-1}$ (i.e. $\omega = \alpha^{-1}$).

4. Set $\sigma := \gamma$.

5. For $i = 1, 2, \ldots, n_1$ do

(a) Set $\sigma := \omega\sigma$ (i.e. $\sigma = \alpha^{-i}\gamma$).

(b) Using binary search, determine if σ is in table T. If it is, set $r := i$ and stop.

Steps 1 and 4 are assignments and are negligible from the complexity point of view. The computation of the inverse of α, in Step 3, is an exponentiation which can be performed using the square-and-multiply method by doing at most $2n_1\lceil\log_2(n_1)\rceil$ group multiplications. Each iteration of Step 2 consists of a binary search which requires at most $2\lceil\log_2(n_2)\rceil$ element comparisons, one group multiplication, and some negligible assignments. In total, Step 2 performs $2n_2\lceil\log_2(n_2)\rceil$ element comparisons and n_2 group multiplications. Finally, each iteration of Step 5 consists of one group multiplication, the binary search requiring at most $2\lceil\log_2(n_2)\rceil$ element comparisons, and some negligible assignments. In total, Step 5 performs at most $2n_1\lceil\log_2(n_2)\rceil$ element comparisons and n_1 group multiplications.

Concerning the space complexity, table T requires storage of n_2 group elements. The other steps require negligible (constant) storage.

Without loss of generality we can assume that $n_1 \geq n_2$. We then see that Algorithm 2 is dominated by Step 5, which requires at most $n_1(2\lceil\log_2(n_2)\rceil + 1) \leq n_1(2\lceil\log_2(n_1)\rceil + 1)$ group multiplications and element comparisons. Hence the complexity of Algorithm 2 is $O(n_1 \cdot \log n_1)$ group multiplications and element comparisons. However, in the case of abstract/generic groups, it is common to count only the group multiplications, i.e., the number of calls to the "black box" that realizes the group. Therefore the worst-case time complexity of Algorithm 2, and hence also of Algorithm 1, is $O(n_1)$ group multiplications and the space complexity is $O(n_2)$ group elements.

1.2.2 The case of Scheme 2

Of course, the generic attack described in the previous section applies to any implementation of Scheme 1. However, Scheme 2 is a slight variation of Scheme 1, and the structures used in the implementation allow for further

reduction in the complexity of the attack. In particular, the need for storage space is minimal, because group membership testing is easy.

In terms of the notation of Section 1.1.1, an opponent needs to obtain one of the four pairs (b, r), (b, s), (a, v), or (a, w) to factor F or H, that is, to obtain the key K. Again, without loss of generality, suppose the opponent has obtained the scalar $b \in \mathbb{F}$ and the integer r. Then the opponent can obtain the key K from b, r, and the public values as $K = C^r H T_2^{-1} T_1^{-1} C^{-r} b^{-1} F$.

Consider the matrix $M := T_2^{-1} T_1^{-1} C^{-i} F = T_2^{-1} T_1^{-1} C^{-i} C^r b T_1 T_2 D^s$ for any integer $0 < i < 2^n$. If $i = r$, then $M = bD^s$, and since $D \in GL_n(GF(2))$, the entries of M will consist of elements b and 0 only. On the other hand, if $i \neq r$ and since C and D are non-commuting elements of prime order, from elementary group theory [2], it follows that $C^{r-i} \notin \langle D \rangle$, and so $M \neq bD^j$ for any j. Moreover, because of statistical reasons, for the vast majority of fixed matrices T_1 and T_2 over $\mathbb{F}$, the entries of matrix M would consist of more than two elements from $\mathbb{F}$.

We have made some implicit assumptions here:

1. Although Scheme 2 did not specify it, the scalars $a, b \in \mathbb{F}$ should be chosen as non-zero elements. We assumed that $b \neq 0$.

2. For practical implementations, the field $\mathbb{F}$ is an extension over $GF(2)$ of degree more than 1. This follows from the fact that T_1 and T_2 are matrices that are supposed to make eigenvalue/eigenvector attacks infeasible [3]. We assumed that $|\mathbb{F}| > 2$.

3. Finally, we have assumed that the matrices T_1 and T_2 are known to the opponent. It is not clear from the original specification [3] whether these matrices (or the element τ in the case of Scheme 1) are pre-shared secrets. The other reason we believe that T_1 and T_2 are public is that they are not chosen randomly in $GL_n(\mathbb{F})$ but are constructed in a specific way to help make eigenvalue/eigenvector attacks infeasible.

The following algorithm implements these ideas:

Algorithm 3.
Input: $n, C, T_1 T_2$, and F
Output: candidates for b and r

1. For $i = 1, 2, \ldots, 2^n - 1$ do
   (a) Compute $M := (T_1 T_2)^{-1} C^{-i} F$.
   (b) If M consists of just two elements 0 and m,
   output $b := m$ and $r := i$ (as candidates) and continue.

An alternative algorithm for the same task, but described by means of generic group membership testing is:

Algorithm 4.
Input: n, C, D, T_1, T_2, and F
Output: b and r

1. For $i = 1, 2, \ldots, 2^n - 2$ do
   (a) Compute $M := T_2^{-1} T_1^{-1} C^{-i} F$.
   (b) For each $m \in \mathbb{F} \setminus \{0\}$ do
      i. If $m^{-1} M \in \langle D \rangle$ then output $b := m$, $r := i$, and stop.

For $n = 31$, which was considered a safe security parameter [3], an opponent had to search through the set $\{C^i T_1 T_2 D^j \mid i, j \in \mathbb{Z}\}$ that is known to have cardinality of $(2^n - 1)^2 \approx 2^{62}$ (infeasible with current technology). However, if an opponent uses Algorithm 3, he/she will need to perform only $2^n - 1 \approx 2^{31}$ operations to break the scheme, and this can be performed on present day personal computers in a reasonable time.

In addition, all algorithms mentioned are highly parallelizable (linear parallelization). This follows from the fact that each iteration can be run independently. In particular, if n processors are used, then the speedup is by a factor of n.

1.2.3 Experimental results

Assume the following scenario: Alice and Bob will be using Scheme 2 with parameters as in Section 1.1.1. Let $n = 31$, C the companion matrix for $p(x) = x^{31} + x + 1 \in GF(2)[x]$, D the companion matrix for $q(x) = x^{31} + x^{30} + x^{28} + x^3 + 1 \in GF(2)[x]$, and $T_1, T_2 \in GL_{31}(\mathbb{F})$ chosen at random, where $\mathbb{F} = GF(2)[x]/(x^8 + x^4 + x^3 + x^2 + 1)$. An attacker was able to obtain the value of H sent over a public network from Alice to Bob, and the value of F sent from Bob to Alice.

We implemented Algorithm 3 in the C-language. Each of the field operations was performed as a table lookup, and ordinary "textbook" matrix multiplication was used. No further speedups or optimizations were used, as opposed to the proposition in [3]. Finally we used the Intel C-compiler v9.0.

A single Intel Pentium-IV, 2.5 GHz computer running Linux OS could perform approximately 750 iterations of Algorithm 3 in 1 second. Forty-four such computers (the current BOCA4 beowulf supercomputer cluster) finished the search in less than 31 hours, while some of the nodes were running other scientific computations at the same time. The search resulted

in a single possibility for b and r which was correct, and was computed after 8 hours and 23 minutes.

The computed values together with H and F can be used directly to obtain the key exchanged between Alice and Bob, as described in the Section 1.2.2.

The program returned only one candidate for b and r. Hence, this experiment also shows that even for a relatively small field $\mathbb{F}$ consisting of 256 elements, it is unlikely that in the case $i \neq r$ the matrix M would consist of only 2 distinct elements.

1.3 Summary

We have shown that there is a difference between obtaining the integers r and s from $\alpha^r\beta^s$ using a brute-force attack and the computational effort to obtain *either* r or s, which is needed in order to break the scheme and obtain the exchanged key.

We argued why only one out of the four secret exponents is needed in order to completely break the key exchange scheme. We proposed algorithms to obtain one of the exponents r or s and estimated their complexity. The time and space complexities of such algorithms can be directly used for the estimation of security parameters in the case of Stickel's scheme as well as any other cryptographic scheme (not necessary a key exchange scheme) based on similar security assumptions.

It should also be noted that once we have obtained one exponent using our proposed algorithms, we can use the known methods for solving traditional discrete logarithms in cyclic groups to obtain the other exponent if we wish to do so.

Finally, Algorithm 1 can be naturally extended to factorize a group element β into more than two predefined "basis" elements $\alpha_1, \alpha_2, \ldots, \alpha_\ell$ such that

$$\beta = \alpha_1^{x_1}\alpha_2^{x_2}\cdots\alpha_\ell^{x_\ell},$$

for some integers x_i's, provided that such factorization of β is possible.

Acknowledgments

I would like to thank Spyros S. Magliveras and Rainer Steinwandt for their valuable suggestions and for providing helpful comments during the research.

References

[1] W. Diffie and M. E. Hellman, New directions in cryptography. *IEEE Transactions on Information Theory* 22(1976), 644-654.

[2] J. J. Rotman, *Advanced Modern Algebra.* Prentice-Hall, 2002.

[3] E. Stickel, A New Method for Exchanging Secret Keys. In. *Proc. of the Third International Conference on Information Technology and Applications (ICITA '05)* 2(2005), 426-430.