A New Key-Agreement-Protocol

Björn Grohmann
Universität Karlsruhe, Fakultät für Informatik,
76128 Karlsruhe, Germany
nn@mhorg.de

Abstract

A new 4-pass Key-Agreement-Protocol is presented. The security of the protocol mainly relies on the existence of a (polynomial-time computable) One-Way-Function and the supposed computational hardness of solving a specific system of equations.

Keywords: Key-Agreement, ultra-high density Knapsack, One-Way-Function.

1 Introduction

At the end of a Key-Agreement-Protocol two parties, say Alice and Bob, share a common bit string s. During the protocol they are allowed to exchange a fixed number of messages m_i, $i = 1, \ldots, r$, over a public channel. The protocol is called secure, if no algorithm exist that computes the string s from the m_i's in a polynomial number of steps. Whether secure Key-Agreement-Protocols exist is still an open issue, although quite a few have been proposed – maybe the most popular being the Diffie-Hellman-Protocol [2], where the security is linked to the task of computing the element γ^{ab} of a given cyclic group from the elements γ^a and γ^b.

In this article, we present a new Key-Agreement-Protocol that uses four rounds of message exchange. Its security mainly relies on the existence of a (polynomial-time computable) One-Way-Function and the supposed computational hardness of solving a specific system of equations.

2 The Protocol

Public data: Suppose Alice and Bob want to exchange a secret key. They start by agreeing on a positive integer n and a prime p of size $\sim 2^{\sqrt{n \log n}}$. They further agree on a random matrix $C := (c_{i,j})_{i,j} \in \mathbb{F}_p^{n \times n}$, with $i, j \in \{1, \ldots, n\}$, and an injective (polynomial-time computable) One-Way-Function $h : \mathbb{F}_p \longrightarrow \{0,1\}^m$, where $\mathbb{F}_p$ denotes the finite field with p elements.

Private data: Next, Alice (resp. Bob) chooses a random element $\alpha \in \mathbb{F}_p$ (resp. β), n random bits $t_1, \dots, t_n$ (resp. $s_1, \dots, s_n$) and a random permutation σ on the set $\{1, \dots, n\}$ (resp. ρ), all of which she (resp. he) keeps secret.

The computations that follow are all taking place in the finite field $\mathbb{F}_p$.

First round: Alice computes for $j = 1, \dots, n$:

$$\mu_j := \sum_{i=1}^{n} t_i c_{i,j} + \sigma(j)\alpha \tag{1}$$

and sends $(\mu_j)_j$ to Bob.

Second round: Bob computes for $i = 1, \dots, n$:

$$\nu_i := \sum_{j=1}^{n} s_j c_{i,j} + \rho(i)\beta \quad \text{and} \quad \tau_A := \sum_{j=1}^{n} s_j \mu_j \tag{2}$$

and sends $((\nu_i)_i, \tau_A)$ to Alice.

Third round: Alice computes for $k = 1, \dots, \frac{n(n-1)}{2}$:

$$h(\tau_A - k\alpha) \quad \text{and} \quad \tau_B := \sum_{i=1}^{n} t_i \nu_i \tag{3}$$

and sends $((h(\tau_A - k\alpha))_k, \tau_B)$ to Bob.

Final round: Bob computes for $l = 1, \dots, \frac{n(n-1)}{2}$ the list $(h(\tau_B - l\beta))_l$ until he finds k_0 and l_0, such that

$$h(\tau_A - k_0\alpha) = h(\tau_B - l_0\beta) \tag{4}$$

and sends k_0 to Alice.

Alice and Bob now share a common element $g := \tau_A - k_0\alpha = \tau_B - l_0\beta$.

3 Analysis

We start by showing the correctness of the protcol and calculate the computational cost:

Theorem 1 *After the final step both parties share a common element* g*. The number of computational steps on both sides equals* $\mathbf{O}(n^2 \cdot \text{cost of evaluation of } h)$*.*

Proof. The correctness of the protocol follows from the easy observation that

$$\tau_A = \sum_{i,j=1}^{n} t_i s_j c_{i,j} + \alpha \sum_{j=1}^{n} s_j \sigma(j) = g' + \alpha k', \tag{5}$$

and respectively

$$\tau_B = \sum_{i,j=1}^{n} t_i s_j c_{i,j} + \beta \sum_{i=1}^{n} t_i \rho(i) = g' + \beta l', \tag{6}$$

and the fact that $1 \leqslant k', l' \leqslant n(n-1)/2$, which means that at least one pair of integers (k_0, l_0) within the given range exists, such that $g := \tau_A - k_0\alpha = \tau_B - l_0\beta$. The number of computational steps is also clear, since Bob can sort the list $(h(\tau_A - k\alpha))_k$ in $\mathbf{O}(n^2 \log n)$ steps, while the evaluation of the injective function h requires $\mathbf{\Omega}(\log p)$ operations. $\square$

The above protocol gives rise to the following

Challenge 1 *Given* n*,* p*,* h*,* C*,* $(\nu_i)_i$*,* $(\mu_j)_j$*,* τ_A*,* τ_B*,* $(h(\tau_A - k\alpha))_k$ *and* k_0*, compute an element* g*, such that* $h(g) = h(\tau_A - k_0\alpha)$*.*

We (i.e. the author of this article) are not aware of any lower bound for the number of steps it takes to compute the element g from Challenge 1.

In what follows, we will present an algorithm that conjecturally requires $\mathbf{\Omega}(2^{\varepsilon\sqrt{n \log n}})$ operations, for some constant $\varepsilon > 0$.

We will try to compute the secrect bits $t_1, \ldots, t_n$ of Alice. As is easily seen, the knowledge of these bits will lead in a polynomial number of steps to the secret key. At the beginning there is only one equation for these bits, that is

$$x_1\nu_1 + \ldots + x_n\nu_n = \tau_B. \tag{7}$$

Now, heuristically speaking, while there are 2^n ways to select the values of the x_i's but only $p \sim 2^{\sqrt{n \log n}}$ possible values for τ_B, there are approximately $2^{n - \log p} \sim 2^{n(1-\sqrt{\log n/n})}$ solutions to equation (7) (in the language of Knapsack-Cryptography, we could speak of an ultra-high density Knapsack, since the density of this Knapsack tends to infinity [4]).

The other equations from (1) involving the t_i's can not be used immediately, since the permutation σ and the element α are both secret, but we can try to get rid of α by guessing r values of the permutation σ, say $\sigma'(1), \ldots, \sigma'(r)$, which gives us $r-1$ additional equations:

$$\begin{aligned}
\sum x_i(\sigma'(2)c_{i,1} - \sigma'(1)c_{i,2}) &= \sigma'(2)\mu_1 - \sigma'(1)\mu_2 \\
\sum x_i(\sigma'(3)c_{i,1} - \sigma'(1)c_{i,3}) &= \sigma'(3)\mu_1 - \sigma'(1)\mu_3 \\
&\vdots \\
\sum x_i(\sigma'(r)c_{i,1} - \sigma'(1)c_{i,r}) &= \sigma'(r)\mu_1 - \sigma'(1)\mu_r.
\end{aligned}$$

Again, by the same heuristic argument, the system of these equations together with equation (7) has approximately $2^{n - r\log p} \sim 2^{n(1-r\sqrt{\log n/n})}$ solutions, which means that we can not even be sure whether our guess was right, unless $n - r\log p \sim \log^\kappa n$, for some constant κ.

To summarize the discussion, the probability of guessing enough equations to compute the t_i (where we did not even talk about the computational cost of really solving these equations) is about $n^{-\varepsilon n/\log p} \sim 2^{-\varepsilon\sqrt{n\log n}}$, for some constant $\varepsilon > 0$, which is, at least from a theoretical point of view not too far away from the probability of guessing the secret α (resp. the secret key g) directly.

It is almost superfluous to say that these heuristic considerations do not prove anything about the security of the stated protocol. Nevertheless, in the author's opinion, Challenge 1 seems worth further investigation.

References

[1] Chor, M.J., Rivest, R.L.: A Knapsack-type Public Key Cryptosystem based on Arithmetik in Finite Fields. In: Blakely, G.R., Chaum, D (eds.) CRYPTO 1984. LNCS vol. 196, pp. 54-65. Springer, Heidelberg (1985)

[2] Diffie, W., Hellman, E.: New directions in cryptography. In: IEEE Trans. Inform. Theory 22(6), pp. 644-654 (1976)

[3] Lagarias, J.C., Odlyzko, A.M.: Solving Low-Density Subset Sum Problems. In: Journal of the Association for Computing Machiniery 32(1), pp. 229-246 (1985)

[4] Nguyen, P., Stern, J.: Adapting Density Attacks to Low-Weight Knapsacks. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol 3788, pp. 41-58. Springer, Heidelberg (2005)