Key Exchange Algorithm Based on Homomorphic Encryption

Sergei Krendelev
Novosibirsk State University
JetBrains Research Cryptographic Lab
s.f.krendelev@gmail.com

Ilya Kuzmin
Novosibirsk State University
JetBrains Research Cryptographic Lab
dargonaxxe@gmail.com

Abstract—Key exchange algorithm based on homomorphic encryption idea is reviewed in this article. This algorithm might be used for safe messaging using one-time pads. Since algorithm requires a low amount of computing resources, this method might be used in IoT to provide authentication.

Keywords—homomorphic encryption; key exchange; one time pad

I. INTRODUCTION

The expected evolution of quantum computers is causing the intensive development of cryptographic primitives called postquantum cryptography. February 6, 2016 was the day that the National Institute of Standards and Technology (NIST) offered to start the development of new postquantum cryptography standards which might be used in governmental needs. According to documents submitted by NIST, algorithms based on a discrete logarithmic problem are vulnerable to quantum attacks. Moreover, elliptical cryptography methods are considered to be vulnerable. Therefore, we need to replace the Diffie-Hellman key exchange algorithm in TLS protocol.

Nowadays, offered postquantum key exchange algorithms are based on lattice theory [4] (LWE, RLWE [2]), which is used in key exchange algorithms named New hope [1] and Frodo [3]. These algorithms are being supported by Google as TLS postquantum update.

We represent the key exchange algorithm based on basic homomorphic encryption properties and linear algebraic methods. The main purpose of this algorithm is to ensure secure messaging. It’s assumed that the one-time pad method will be used. To use the algorithm in TLS one should change it in accordance with specification.

II. NOTATION

In this section we will describe the notation that will be used.

Let be a ring of integer numbers. For let’s call a vector a = (a1, ..., an) n-dimensional integer vector if . For n ∈ ℕ let Zn denote the set of all possible n-dimensional integer vectors. Moreover, it’s assumed that n-dimensional vectors have the following properties:

1) For each pair x, y ∈ Zn the following is true

\[ \forall i \in \{1, \ldots, n\}, x_i + y_i \in \mathbb{Z} \]

2) For each x ∈ Zn, α ∈ ℤ the following is true

\[ \alpha x = (\alpha x_1, \ldots, \alpha x_n) \]

Moreover, for x, y ∈ Zn let’s call \( \langle x, y \rangle = \sum_{i=1}^{n} x_i y_i \) a dot product.

III. KEY EXCHANGE ALGORITHM BASED ON HOMOMORPHIC ENCRYPTION

In this section we represent the key exchange algorithm. We suppose that users Alice (server) and Bob (client) desires to get a common key. Moreover, we suppose that both users trust each other (Authentication is completed) and both users know the value of k ∈ ℕ.

0) Alice and Bob, together, choose z, z’ ∈ ℤ such that z < z’ and number n ∈ ℕ.

1) Alice chooses the number α ∈ ℤ and the secret vector x ← U_{Z^2}^n, the set of vectors a1 ← U_{Z^2}^{n−1}, ..., an ← U_{Z^2}^{n−1}. Then Alice calculates d1 = a1 · x, ..., dn = an · x.

In addition, Alice chooses the number p ∈ ℕ and finds the set of vectors s1, ..., sp ∈ Zn: \( \forall i \in \{1, \ldots, p\}, s_i · x = 0 \). Alice sends a1, ..., an, s1, ..., sp to Bob.

2) Bob chooses the number m ∈ ℕ and the secret vector y ← U_{Z^2}^m, the set of vectors b1 ← U_{Z^2}^{m−1}, ..., bm ← U_{Z^2}^{m−1}. Then Bob calculates b1 = b1 · y, ..., bm = bm · y.

In addition, Bob chooses the number q ∈ ℕ and finds the set of vectors r1, ..., rq ∈ Zn: \( \forall i \in \{1, \ldots, q\}, r_i · y = 0 \). Bob sends b1, ..., bm, r1, ..., rq to Alice.

3) Alice calculates v = d1b1 + ⋯ + dmbm + µ1r1 + ⋯ + µqrq, where µ1 ← U_{Z^2}, ..., µq ← U_{Z^2}. Alice sends vector v to Bob.

REFERENCES

[1] These algorithms are being supported by Google as TLS update.


[3] These algorithms are being supported by Google as TLS update.

[4] These algorithms are being supported by Google as TLS update.
IV. MITM PASSIVE ATTACK

In this section we estimate how successful a Man In The Middle (MITM) passive attack can be. Passive means that an adversary can’t edit the data transmitted by Alice and Bob. An adversary has vectors $a_1, \ldots, a_l, b_1, \ldots, b_k, s_1, \ldots, s_p, r_1, \ldots, r_q$. Also, the following system of equations is known by adversary:

$$w = (b_1 \cdot y)a_1 + \ldots + (b_k \cdot y)a_k + \lambda_1 s_1 + \ldots + \lambda_p s_p$$
$$v = (a_1 \cdot x)b_1 + \ldots + (a_k \cdot x)b_k + \mu_1 r_1 + \ldots + \mu_q r_q$$

Choosing proper values for $p, q, k, m, n$, users can make the system underdetermined. Hence the needed solution can’t be found by adversary. To improve the algorithm users can substantially increase dimension using sparse vectors.

V. TOY EXAMPLE

In this section we reduce the number of dimensions to show the way algorithm works.

Let $k = 4, n = 3, m = 2, p = 2, q = 2, z = 0, z' = 7$.

0) Alice and Bob chooses $z = 0, z' = 7, k = 4$.

1) Alice chooses $n = 3, x = (2 \ 3 \ 4)$,
   $$a_1 = (4 \ 3 \ 7),$$
   $$a_2 = (3 \ 0 \ 1),$$
   $$a_3 = (3 \ 5 \ 3),$$
   $$a_4 = (1 \ 3 \ 7).$$
   $$d_1 = 45, d_2 = 10, d_3 = 33, d_4 = 39.$$  
   $$s_1 = (-3 \ 2 \ 0),$$
   $$s_2 = (0 \ -4 \ 3).$$

2) Bob chooses $m = 2, y = (1 \ 5)$,
   $$b_1 = (6 \ 5),$$
   $$b_2 = (6 \ 6),$$
   $$b_3 = (5 \ 7),$$
   $$b_4 = (5 \ 4).$$
   $$h_1 = 31, h_2 = 36, h_3 = 40, h_4 = 25.$$  
   $$r_1 = (-5 \ -1),$$
   $$r_2 = (10 \ -2).$$

3) Alice chooses $\mu_1 = 6, \mu_2 = 5$, then calculates $v = 45b_1 + 10b_2 + 33b_3 + 39b_4 + 6r_1 + 5r_2 = (710 \ 668)$.

4) Bob chooses $\lambda_1 = 7, \lambda_2 = 3$, then calculates
   $$w = 31a_1 + 36a_2 + 40a_3 + 25a_4 + 7s_1 + 3s_2 = (356 \ 370 \ 557).$$

5) Alice calculates $x \cdot w = 4050$

6) Bob calculates $y \cdot v = 4050$

VI. IMPLEMENTATION AND PERFORMANCE

We implemented this algorithm using the C++ programming language. Implementation uses open source long arithmetic library GNU MP (GMP). The values for $k, n, m, p, q$ are fixed: $k = 60, n = 45, m = 40, p = 30, q = 35$. Table 1 represents the average results of 400 tests being executed on the single PC with CPU Intel Core i7-640M Processor with 4M Cache, 2.80 GHz. Here what represents each row:

1) Key size – amount of bits required to contain the key in memory.

2) Time spent on initialization – time taken by Alice to perform part 1 of algorithm (part 2 for Bob respectively).

3) Time spent on calculation – time taken by Alice to perform calculations from part 3 and 5 (4 and 6 for Bob respectively).

4) Time spent on data transmission – this time is a theoretical value. We calculated it assuming that both users have stable Internet connection of 50 Mbps.

5) Time spent on key exchange with preparation – time taken by user to perform the key exchange assuming that user already generated the data from part 1-2.

6) Time spent on key exchange without preparation – the opposite, time taken by user to perform key exchange with data generation.

7) Amount of transmitted data – size of messages sent by users.

VII. CONCLUSION

The reviewed algorithm is a promising cryptographic primitive that is believed to be resistant to quantum attacks. The implementation results are presented in Table 1. The benchmarking results are measured on Intel Core i7-640M

<table>
<thead>
<tr>
<th>Performance</th>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key size</td>
<td>2290 bits</td>
<td></td>
</tr>
<tr>
<td>Time spent on initialization</td>
<td>130 ms</td>
<td>108.5 ms</td>
</tr>
<tr>
<td>Time spent on calculation</td>
<td>5.2 ms</td>
<td>5.6 ms</td>
</tr>
<tr>
<td>Time spent on data transmission</td>
<td>2.9 ms</td>
<td>2.9 ms</td>
</tr>
<tr>
<td>Time spent on key exchange with preparation</td>
<td>8.1 ms</td>
<td>8.5 ms</td>
</tr>
<tr>
<td>Time spent on key exchange without preparation</td>
<td>138.2 ms</td>
<td>117.1 ms</td>
</tr>
<tr>
<td>Amount of transmitted data</td>
<td>19208 bytes</td>
<td>19016 bytes</td>
</tr>
</tbody>
</table>
with 2 cores running at 2.8 GHz. The implementation details are shown in GitHub repository.

REFERENCES


1github.com/dargonaxxe/homomorphic-encryption-key-exchange