Proceedings of the Federated Conference on Computer Science and Information Systems pp. 793–795
DOI: 10.15439/2017F53
ISSN 2300-5963 ACSIS, Vol. 11

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 $\mathbb{Z}$ be a ring of integer numbers. For $n \in \mathbb{N}$ let's call $\mathbf{a} = (a_1, \ldots, a_n)$ *n-dimensional integer vector* if $\forall i \in \{1, \ldots, n\} a_i \in \mathbb{Z}$. For $n \in \mathbb{N}$ let $\mathbb{Z}^n$ 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 $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$ the following is true $\mathbf{x} + \mathbf{y} = (x_1 + y_1, \ldots, x_n + y_n)$.
2) For each $\mathbf{x} \in \mathbb{Z}^n, \alpha \in \mathbb{Z}$ the following is true $\alpha\mathbf{x} = (\alpha x_1, \ldots, \alpha x_n)$.

Moreover, for $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$ let's call $\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i$ a *dot product*.
Let χ be a probability distribution over $\mathbb{Z}$. Accordingly, $x \leftarrow \chi$ denotes sampling $x \in \mathbb{Z}$ according to χ. Moreover, for $a, b \in \mathbb{Z}, a < b$ let $x \leftarrow U_{a,b}$ denote sampling x uniformly from $\{a, a+1, \ldots, b\}$. Moreover, let $\mathbf{x} \leftarrow U^n_{a,b}$ denote sampling $\mathbf{x}$ in the following way: $\mathbf{x} = (x_1 \leftarrow U_{a,b}, \ldots, x_n \leftarrow U_{a,b})$.

Definition II.1. Homomorphic encryption Let $\mathbf{x} \in \mathbb{Z}^n$ be a fixed n-dimensional integer vector for some $n \in \mathbb{N}$. Moreover, let's consider that $\mathbf{x}$ has *at least* 2 coprime components.
For number $d \in \mathbb{Z}$ and vector $\mathbf{x}$ we will call a vector $\mathbf{a} \in \mathbb{Z}^n$ an *interpretation* if $\mathbf{x} \cdot \mathbf{a} = d$.
Therefore we have a mapping $\Phi_{\mathbf{x}} : \mathbb{Z} \to \mathbb{Z}^n$. Easy to notice that this mapping has the following properties:

1) $\Phi_{\mathbf{x}}(\mathbf{a}_1 + \mathbf{a}_2) = \Phi_{\mathbf{x}}(\mathbf{a}_1) + \Phi_{\mathbf{x}}(\mathbf{a}_2)$
2) $\Phi_{\mathbf{x}}(\alpha\mathbf{a}) = \alpha\Phi_{\mathbf{x}}(\mathbf{a})$

This mapping is called *homomorphic encryption*.

III. Key exchange based on homomorphic encryption

In this section we represent the key exchange algorithm. We suppose that 2 users – Alice (server) and Bob (client) decides to get a common key. Moreover, we suppose that both users trust each other (Authentication is completed) and both users know the value of $k \in \mathbb{N}$.

0) Alice and Bob, together, choose $z, z' \in \mathbb{Z}$ such that $z < z'$ and number $n \in \mathbb{N}$.
1) Alice chooses the number $n \in \mathbb{N}$ and the secret vector $\mathbf{x} \leftarrow U^n_{z,z'}$, the set of vectors $\mathbf{a}_1 \leftarrow U^n_{z,z'}, \ldots, \mathbf{a}_k \leftarrow U^n_{z,z'}$. Then Alice calculates $d_1 = \mathbf{a}_1 \cdot \mathbf{x}, \ldots, d_k = \mathbf{a}_k \cdot \mathbf{x}$. In addition, Alice chooses the number $p \in \mathbb{N}$ and finds the set of vectors $\mathbf{s}_1, \ldots, \mathbf{s}_p \in \mathbb{Z}^n : \forall i \in \{1, \ldots, p\} \mathbf{s}_i \cdot \mathbf{x} = 0$. Alice sends $\mathbf{a}_1, \ldots, \mathbf{a}_k$, $\mathbf{s}_1, \ldots, \mathbf{s}_p$ to Bob.
2) Bob chooses the number $m \in \mathbb{N}$ and the secret vector $\mathbf{y} \leftarrow U^m_{z,z'}$, the set of vectors $\mathbf{b}_1 \leftarrow U^m_{z,z'}, \ldots, \mathbf{b}_k \leftarrow U^m_{z,z'}$. Then Bob calculates $h_1 = \mathbf{b}_1 \cdot \mathbf{y}, \ldots, \mathbf{h}_k = \mathbf{b}_k \cdot \mathbf{y}$. In addition, Bob chooses the number $q \in \mathbb{N}$ and finds the set of vectors $\mathbf{r}_1, \ldots, \mathbf{r}_q \in \mathbb{Z}^m : \forall i \in \{1, \ldots, q\} \mathbf{r}_i \cdot \mathbf{y} = 0$. Bob sends $\mathbf{b}_1, \ldots, \mathbf{b}_k$, $\mathbf{r}_1, \ldots, \mathbf{r}_q$ to Alice.
3) Alice calculates $\mathbf{v} = d_1\mathbf{b}_1 + \cdots + d_k\mathbf{b}_k + \mu_1\mathbf{r}_1 + \cdots + \mu_q\mathbf{r}_q$, where $\mu_1 \leftarrow U_{z,z'}, \ldots, \mu_q \leftarrow U_{z,z'}$. Alice sends vector $\mathbf{v}$ to Bob.

4) Bob calculates $\mathbf{w} = h_1\mathbf{a}_1+\cdots+h_k\mathbf{a}_k+\lambda_1\mathbf{s}_1+\cdots+\lambda_p\mathbf{s}_p$, where $\lambda_1 \leftarrow U_{z,z'}, \ldots, \lambda_p \leftarrow U_{z,z'}$. Bob sends vector $\mathbf{w}$ to Alice.
5) Alice calculates $l = \mathbf{w} \cdot \mathbf{x}$.
6) Bob calculates $t = \mathbf{v} \cdot \mathbf{y}$.

Let's prove that $l = t$.

$$l = \mathbf{w} \cdot \mathbf{x} = (h_1\mathbf{a}_1 + \cdots + h_k\mathbf{a}_k + \lambda_1\mathbf{s}_1 + \cdots + \lambda_p\mathbf{s}_p) \cdot \mathbf{x} = (h_1\mathbf{a}_1 \cdot \mathbf{x} + \cdots + h_k\mathbf{a}_k \cdot \mathbf{x}) + (\lambda_1\mathbf{s}_1 \cdot \mathbf{x} + \cdots + \lambda_p\mathbf{s}_p \cdot \mathbf{x}) = h_1\mathbf{a}_1 \cdot \mathbf{x} + \cdots + h_k\mathbf{a}_k \cdot \mathbf{x} = h_1d_1 + \cdots + h_kd_k$$

$$t = \mathbf{v} \cdot \mathbf{y} = (d_1\mathbf{b}_1 + \cdots + d_k\mathbf{b}_k + \mu_1\mathbf{r}_1 + \cdots + \mu_q\mathbf{r}_q) \cdot \mathbf{y} = (d_1\mathbf{b}_1 \cdot \mathbf{y} + \cdots + d_k\mathbf{b}_k \cdot \mathbf{y}) + (\mu_1\mathbf{r}_1 \cdot \mathbf{y} + \cdots + \mu_q\mathbf{r}_q \cdot \mathbf{y}) = d_1\mathbf{b}_1 \cdot \mathbf{y} + \cdots + d_k\mathbf{b}_k \cdot \mathbf{y} = d_1h_1 + \cdots + d_kh_k$$

$l = t$. Therefore key exchange is accomplished.

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 $\mathbf{a}_1, \ldots, \mathbf{a}_k, \mathbf{b}_1, \ldots, \mathbf{b}_k$, $\mathbf{s}_1, \ldots, \mathbf{s}_p, \mathbf{r}_1, \ldots, \mathbf{r}_q$. Also, the following system of equations is known by adversary:

$$\mathbf{w} = (\mathbf{b}_1 \cdot \mathbf{y})\mathbf{a}_1 + \cdots + (\mathbf{b}_k \cdot \mathbf{y})\mathbf{a}_k + \lambda_1\mathbf{s}_1 + \cdots + \lambda_p\mathbf{s}_p$$
$$\mathbf{v} = (\mathbf{a}_1 \cdot \mathbf{x})\mathbf{b}_1 + \cdots + (\mathbf{a}_k \cdot \mathbf{x})\mathbf{b}_k + \mu_1\mathbf{r}_1 + \cdots + \mu_q\mathbf{r}_q$$
$$\mathbf{s}_1 \cdot \mathbf{x} = 0, \ldots, \mathbf{s}_p \cdot \mathbf{x} = 0$$
$$\mathbf{r}_1 \cdot \mathbf{y} = 0, \ldots, \mathbf{r}_q \cdot \mathbf{y} = 0$$

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 $n = 3$, $\mathbf{x} = \begin{pmatrix}2 & 3 & 4\end{pmatrix}$,
   $\mathbf{a}_1 = \begin{pmatrix}4 & 3 & 7\end{pmatrix}$,
   $\mathbf{a}_2 = \begin{pmatrix}3 & 0 & 1\end{pmatrix}$,
   $\mathbf{a}_3 = \begin{pmatrix}3 & 5 & 3\end{pmatrix}$,
   $\mathbf{a}_4 = \begin{pmatrix}1 & 3 & 7\end{pmatrix}$.
   $d_1 = 45, d_2 = 10, d_3 = 33, d_4 = 39$.
   $\mathbf{s}_1 = \begin{pmatrix}-3 & 2 & 0\end{pmatrix}$
   $\mathbf{s}_2 = \begin{pmatrix}0 & -4 & 3\end{pmatrix}$.
2) Bob chooses $m = 2$, $\mathbf{y} = \begin{pmatrix}1 & 5\end{pmatrix}$,
   $\mathbf{b}_1 = \begin{pmatrix}6 & 5\end{pmatrix}$,
   $\mathbf{b}_2 = \begin{pmatrix}6 & 6\end{pmatrix}$,
   $\mathbf{b}_3 = \begin{pmatrix}5 & 7\end{pmatrix}$,
   $\mathbf{b}_4 = \begin{pmatrix}5 & 4\end{pmatrix}$.
   $h_1 = 31, h_2 = 36, h_3 = 40, h_4 = 25$.
   $\mathbf{r}_1 = \begin{pmatrix}-5 & 1\end{pmatrix}$,
   $\mathbf{r}_2 = \begin{pmatrix}10 & -2\end{pmatrix}$.

TABLE I
PERFORMANCE

	Alice	Bob
Key size	2290 bits	
Time spent on initialization	130 ms	108.5 ms
Time spent on calculation	5.2 ms	5.6 ms
Time spent on data transmission	2.9 ms	2.9 ms
Time spent on key exchange with preparation	8.1 ms	8.5 ms
Time spent on key exchange without preparation	138.2 ms	117.1 ms
Amount of transmitted data	19208 bytes	19016 bytes

3) Alice chooses $\mu_1 = 6, \mu_2 = 5$, then calculates $\mathbf{v} = 45\mathbf{b}_1 + 10\mathbf{b}_2 + 33\mathbf{b}_3 + 39\mathbf{b}_4 + 6\mathbf{r}_1 + 5\mathbf{r}_2 = \begin{pmatrix}710 & 668\end{pmatrix}$.
4) Bob chooses $\lambda_1 = 7, \lambda_2 = 3$, then calculates $\mathbf{w} = 31\mathbf{a}_1 + 36\mathbf{a}_2 + 40\mathbf{a}_3 + 25\mathbf{a}_4 + 7\mathbf{s}_1 + 3\mathbf{s}_2 = \begin{pmatrix}356 & 370 & 557\end{pmatrix}$.
5) Alice calculates $\mathbf{x} \cdot \mathbf{w} = 4050$
6) Bob calculates $\mathbf{y} \cdot \mathbf{v} = 4050$

VI. IMPLEMENTATION AND PERFORMANCE

We implemented this algorithm using the C++ programming language. Implementation uses open source long arithmetics 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

with 2 cores running at 2.8 GHz. The implementation details are shown in GitHub repository[1]

References

[1] Erdem Alkim, Leo Ducas, Thomas Poppelmann, and Peter Schwabe. Post-quantum key exchange - a new hope. Cryptology ePrint Archive, Report 2015/1092, 2015. http://eprint.iacr.org/2015/1092.

[2] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post-quantum key exchange for the tls protocol from the ring learning with errors problem. In 2015 IEEE Symposium on Security and Privacy, pages 553–570, May 2015.

[3] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. Cryptology ePrint Archive, Report 2016/659, 2016. http://eprint.iacr.org/2016/659 .

[4] Chris Peikert. Lattice Cryptography for the Internet, pages 197–219. Springer International Publishing, Cham, 2014.

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