Mathematical Problems of Computer Science 45, 90--98, 2016.

Optimization Techniques for Generic Secure Two-party Computation Platform

Tigran V. Sokhakyan

Russian-Armenian (Slavonic) University
e-mail: tigran.sokhakyan@gmail.com

Abstract

In this article we present an implementation of general purpose secure two-party computation framework offering security against semi-honest threat model. Proposed framework implements Yao's garbled circuits protocol and incorporates novel oblivious transfer protocol based on white-box cryptography methods for the first time to avoid computationally expensive public key operations. Also experimental results illustrating the efficiency of our framework compared with previous implementations are provided.

Keywords: Secure two-party computation, Yao's garbled circuits protocol, White-box cryptography, Oblivious transfer.

1. Introduction

As the world is getting more and more connected in many real world scenarios, parties with different and potentially conflicting interests have to interact. Examples of this are citizens and governments (electronic passport and electronic ID), patients and health insurers or medical institutions (electronic health card, e-health services), or companies and service providers (cloud computing). In the mentioned context questions of paramount importance are the architecture and the ability of an underlying communication system to fulfill varied security and privacy requirements of the involved parties.

Protocols for secure computations allow two or more mutually distrustful parties communicate and jointly compute some commonly agreed function on each other's input without possessing a trusted party, having privacy and authenticity guarantees. Andrew Yao pointed out that secure two-party protocols can be constructed for computation of any computable function [1].

Yao's protocol remains one of the most actively studied methods for secure computations. Although Yao never published a precise protocol, the very first real world implementation of secure two-party computation (S2PC) [2] used Yao's basic garbled circuit approach, and it

remains the primary paradigm for the plenty of 2PC implementations that have been developed during the past eleven years [3 - 5].

Yao's protocol has a great practical significance. In many real-world situations, the inputs to a function may be too valuable or sensitive to share with other parties. Efficient S2PC algorithms enable a variety of electronic transactions, previously impossible due to mutual mistrust of participants. Bringer et al. presented recent advances in privacy-preserving biometric identification [6], when it is desirable for individual genetic data to be kept private but still checked against a specified list. The more general case of multiparty computation has already seen real-world use in computing market clearing prices in Denmark [7]. This is not so forth full list of applications: auctions [8], contract signing [9], etc.

Organization of the paper. In Section 2 necessary cryptographic background is covered. In Section 3 we give implementation overview of our framework. Section 4 provides high level description of framework usage and Section 5 provides experimental results.

2. Background

In subsequent sections we briefly introduce the main cryptographic tools we have used in our framework: garbled circuits, white-box cryptography based oblivious transfer (OT) protocol and optimization techniques for Yao's protocol.Yao's Garbled Circuits Protocol.

2.1. Yao's Garbled Circuits Protocol

Yao's garbled circuit protocol [1] allows two mutually distrustful parties holding inputs x and y to evaluate an arbitrary computable function $f(x, y)$ on their input values without leaking any side information about their inputs beyond what is explicitly implied by the function output.

The main idea is that one party (called garbled circuit *generator*) generates an "encrypted" version of the Boolean circuit C computing the function f, and the second party (called garbled circuit *evaluator*) obliviously computes the garbled circuit. Note that reverse engineering techniques are not applicable to garbled circuit, thus the *evaluator* does not learn any intermediate value.

Suppose the *generator* has a Boolean circuit C with 2 fan-in gates computing the function f. At the first step the *generator* fixes some integer k and assigns two random looking bit strings w^0 and w^1 to each wire of circuit C (label w^b conceptually encodes value $b \in \{0, 1\}$ for the wire w). Then for the gate g having output wire w_k and input wires w_i, w_j, the *generator* prepares the garbled table with the following entries:

$$Enc_{w_i^{b_i}, w_j^{b_j}} \quad (w_k^{g(b_j, b_j)}), \tag{1}$$

where *Enc* is an encryption scheme fixed by *generator*. The collection of all garbled gates is called a garbled circuit.

Then the *generator* passes the garbled circuit and mapping for the labels for the output wires to the *evaluator*. Note that only the *generator* knows mapping between binary input bits for input wires and it can simply send the garbled circuit to the evaluator with label $w_i^{x_i}$ for input wire w_i, where x_i is the i-th bit of its input. To obtain wire labels for its input the *evaluator* runs oblivious transfer protocol described next with the *generator*.

The evaluation of garbled circuit is done in a hierarchical way. Given labels w_i and w_j of input wires of garbled gate g the *evaluator* decrypts the appropriate entry of garbled table using the keys w_i and w_j. When labels of all output wires are computed the *evaluator* sends the function output value to the *generator* using the provided mapping for output wires.

2.2. White-box Cryptography Based OT Protocol Extension

1-out-of-2 oblivious protocol (OT) is an essential part of Yao's garbled circuit protocol. It involves two parties: *sender* holding two strings w_0 and w_1, and *receiver* holding the selection bit b. OT protocol allows the *sender* to transmit exactly one input string w_b to *receiver*; the *receiver* learns nothing about $w_{b\oplus 1}$ and the *sender* does not learn selection bit b. Currently several OT protocols are available. For implementation of OT we use novel white-box cryptography based OT protocol which is secure in the semi-honest setting and is introduced in [10], which is order of magnitude faster compared with previous OT protocol implementations.

2.3. Efficient Garbling Schemes

To get more practical use of Yao's garbled circuits protocol many optimizations are developed during the past decade, some of which are compatible with each other and are incorporated in our framework. Kolesnikov and Schneider [11] introduced a technique eliminating the need to garble XOR gates (XOR gates become "free", involving no communication or cryptographic operations). Also, we use the technique proposed by Pinkas et al. [12] allowing to reduce the size of a garbled table from four to three ciphertexts, and saving 25% of network bandwidth for non-XOR gates. Another optimization to apply is the FleXOR technique by Kolesnikov et al. [13] combined with two garbled row reduction [12] instead of one. We have added this option, as two garbled row reduction and FreeXOR techniques are not compatible. The combination of optimization techniques is user configurable.

3. The Structure of Computation Framework

The framework implements Yao's garbled circuits protocol to perform the privacy-preserving computation of function $f(a, b)$, where the actual values of arguments are provided by mutually distrustful participants.

Our framework is implemented in C++ programming language. It contains a compiler generating an equivalent Boolean circuit from the algorithmic representation of given function $f(a, b)$. Popular $flex$ and $bison$ tools were used to generate the compiler. The details of compiler implementation and description of its' input language are given in [16].

The second major part of our framework is the implementation of Yao's garbled circuits protocol. The implementation employees various state of the art optimizations together with usage of novel oblivious transfer protocol based on white-box cryptography operations. White-box cryptography methods are used to avoid computationally expensive public key operations during oblivious transfer. In our implementation, we use the white-box implementation of SAFER+ [17].

Other optimizations, introduced earlier in this field, are incorporated for speedup of different aspects of practical Yao's garbled circuits protocol implementation. To avoid delays caused by transmission of small amounts of data blocks over the network, communication batching is employed. The usage of communication batching gives the opportunity of data compression sent over the network. This would be highly desirable especially for wide area networks, where the usage of the underlying network becomes prohibitively expensive. To the best of our knowledge, none of the previous implementations uses emphasized data compression tool.

During the construction and the evaluation of garbled circuit encryption function fixed by underlying garbling scheme is used extensively. For performing this operation faster, whenever it is possible, we have made use of AES encryption instructions supported on popular Intel processors. Although, these instructions are not portable, they provide additional speedup whenever they are available.

In the baseline Yao's garbled circuits protocol a garbled circuit is constructed and transmitted over the network for evaluation, which requires entire circuit to be stored in memory. We have incorporated pipelined construction and evaluation of the garbled circuit [16], to reduce the amount of memory needed during computations and enable the participants do the computations without delays.

Another optimization is employed to minimize the amount of memory needed for storing the intermediate wire labels during construction and evaluation of the garbled circuit. The compiler generates usage count for each gate, and it is decremented each time the value of a wire is accessed. The intermediate results are swept off the memory as the counter hits zero.

The above-mentioned optimizations or their alternatives are used in the previous implementations of S2PC and none of the previous implementations incorporates all of them. The use of these techniques enabled us to automatically construct and evaluate circuits having more gates than previous implementations are capable.

4. Usage of Two-party Computation Framework

For convenience suppose there are two parties called *Alice* and *Bob* wishing to compute the function $f(a,b)$ on their private values a and b. The higher level view of framework usage is presented in Fig. *1*, and the flow is described below.

First the parties need to describe the function f in using the source language of the compiler which is a part of the framework [16]. Then an equivalent Boolean circuit is generated implementing function f. This circuit is then shared among *Alice* and *Bob*. Note, that resource-intensive generation of the Boolean circuit is only once and the circuit can be used multiple times.

Having circuit structure *Alice* determines the number of input bits corresponding to *Bob*'s input wires and generates n garbled labels for these wires. After this *Alice* and *Bob* initiate 1-out-of-2 white-box oblivious transfer protocol extension where *Alice* inputs generated garbled labels and *Bob* inputs bits of his private input y. After this step *Bob* obliviously gets wire labels corresponding to his input value. Then *Alice* and *Bob* perform pipelined evaluation (see Section 4.1) of the circuit. After this step *Alice* and *Bob* communicate to get their respective private outputs without leaking anything. Note, that *Alice* knows the correspondence between plain Boolean values and garbled ones for all wires. Thus, she can restore her private output bits when actually garbled values are known to her. Also, Alice cannot reveal *Bob*'s private output because she does not know actually garbled values for wires corresponding to *Bob*'s private output. For these reasons *Bob* sends to *Alice* garbled labels corresponding to her private output wires and *Alice* sends to *Bob* garbling scheme for his private output wires.

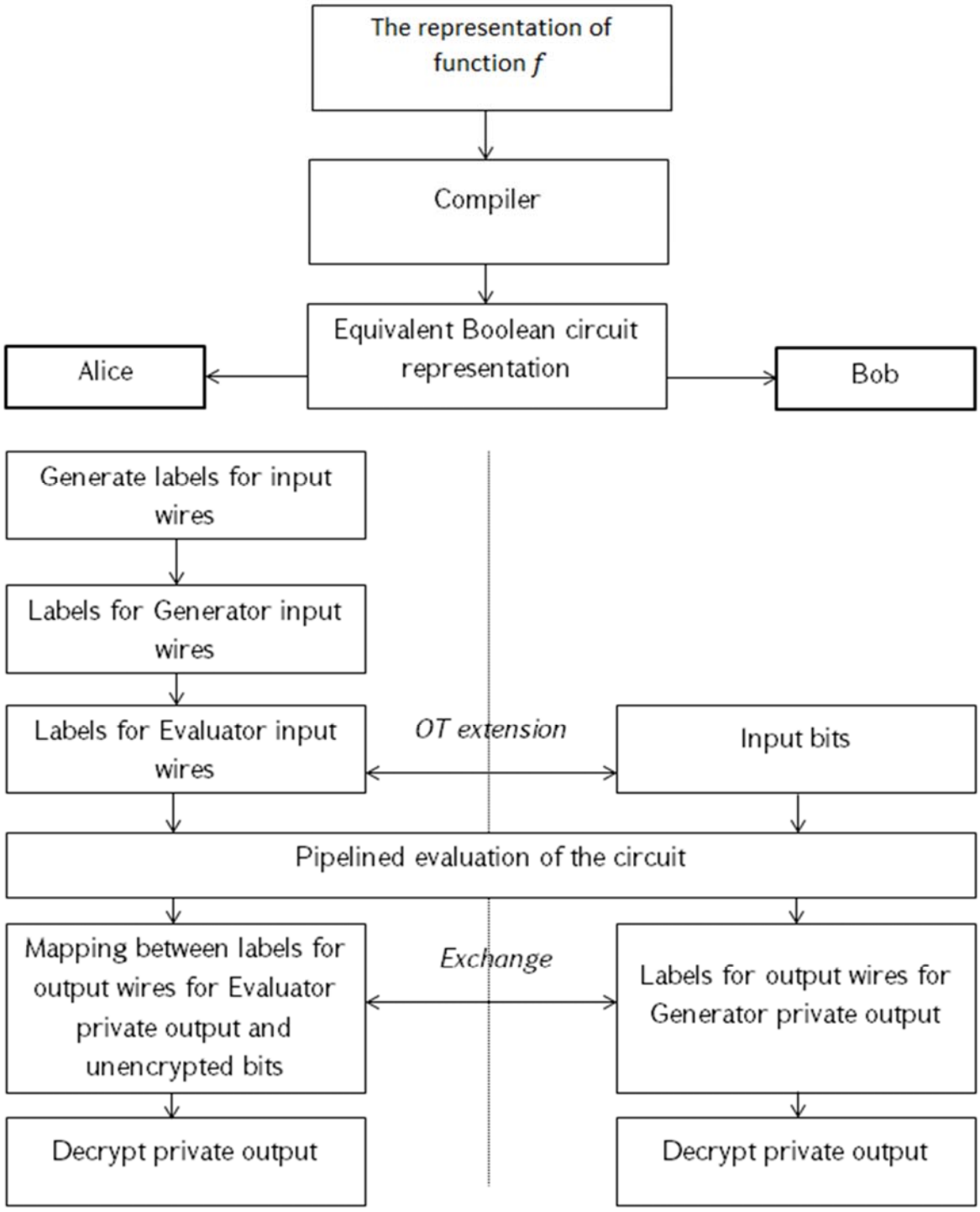

Fig. 1. The higher level view of steps performed by participants during computation of function f(a, b) using our framework.

5. Experimental Results

This section presents comparison of running times between our framework and a previous implementation implementing Yao's garbled circuits protocol in semi-hones adversarial model

[18]. For this, we considered privacy-preserving evaluation of two popular functions, namely, secure computation of Hamming and Levenshtein distances between provided arguments.

For convenience we call the participants of computations *the client* and *the server*. This naming convention comes from the practice, where these problems are considered in client-server architecture.

Formally, the Hamming distance between given pair of n-bit binary strings c and s, denoted $H(c,s)$, is defined the total number of correspondingly different bits between c and s. In a privacy-preserving setting, c is the secret input of the client and s is the secret input of the server. In this setting, the client and the server wish to jointly compute $H(c,s)$, or use it as an intermediate result during subsequent computations, without revealing respective private inputs to another party.

Levenshtein distance problem between two strings has important applications in computational biology, comparing text files and various fields. The problem statement as follows; given a set O consisting of basic operations applicable on individual characters of some string and two strings x and y, find the edit distance between them. The edit distance between strings x and y is denoted by *Levenstein(x, y)* and is defined as the minimum number of basic operations from O needed for transformation of string x into y. We considered the typical case, when the set O consists of insertion, deletion and replacement of single character in certain position of the source string.

Fig. *2* presents comparison of overall running times of privacy-preserving Hamming distance computation between the previous implementation and the implementation presented in this work.

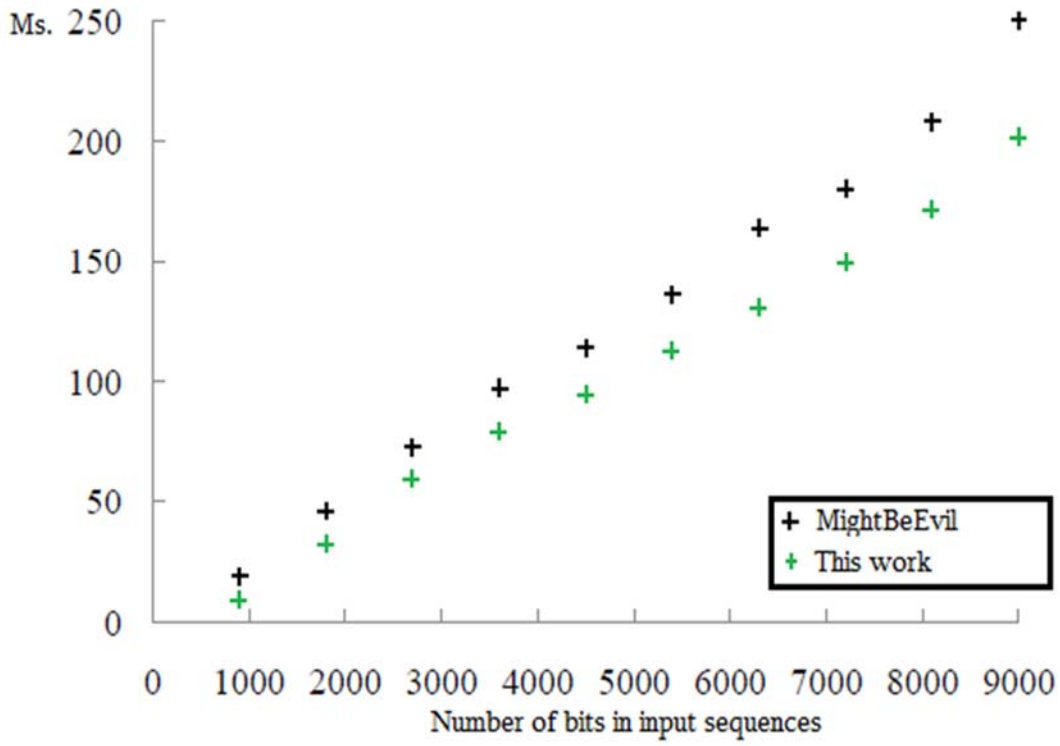

Fig. 2. Comparison of running times of secure Hammaing distance computation between MightBeEvil and the presented framework.

Fig. *3* presents comparison of overall running times of privacy-preserving edit distance computation between the previous implementation and the implementation presented in this paper.

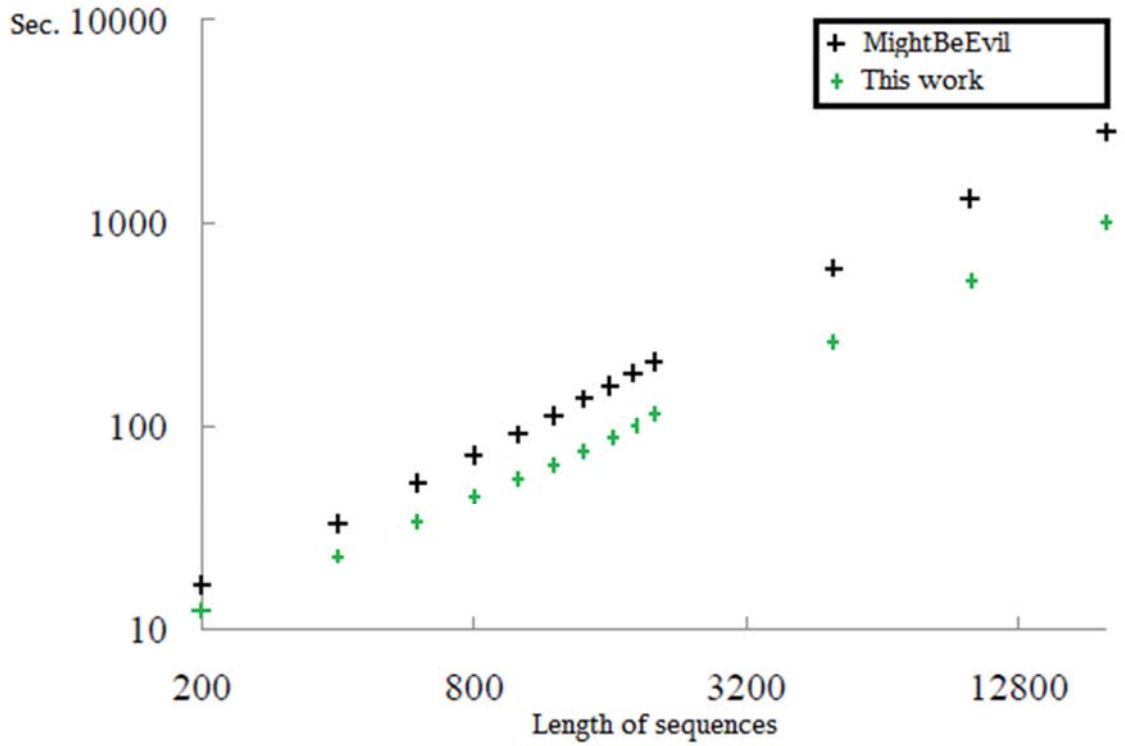

Fig. 3. Comparison of running times of secure Levenshtein distance computation between MightBeEvil and the presented framework.

As it was expectable, our framework outperforms the previous implementation. The difference of running times grows with the number of input bits of the function. The performance of previous implementations of S2PC suffers highly from extensive usage of public key operations during OT protocol execution, which is used to obliviously exchange the garbled values corresponding to the input bits of the server (for details see Section 3). As the number of input bits grows, previous implementation spends more time on OT execution step compared with our implementation, which replaces the usage of computationally expensive public key operations with order of magnitude faster white-box cryptography counterparts.

6. Conclusion

In this article, we have presented a framework for secure two-party computation offering security in the semi-honest adversarial model. The framework incorporates various optimizations with usage of white-box cryptography methods, which are order of magnitude faster compared with public key operations, during oblivious transfer protocol execution step. Experimental results given in the article prove that these optimizations really enhance the overall running time of the computations. The benefit of white-box cryptography methods usage is highly expressed when the length of the input arguments grows.

We plan to make further improvements in our system to enable safety against malicious adversaries, with comparable performance and experimentally measure the impact of white-box cryptography methods on used network bandwidth.

References

[1] A. C.-C. Yao, "How to Generate and Exchange Secrets (Extended Abstract)," *27th Annual Symposium on Foundations of Computer Science, Toronto*, pp. 162 - 167, October 27-29, 1986.

[2] D. Malkhi, N. Nisan, B. Pinkas and Y. Sella, "Fairplay - Secure Two-Party Computation System," *Proceedings of the 13th USENIX Security Symposium*, pp. 287-302, 2004.

[3] T. K. Frederiksen, T. P. Jakobsen, J. B. Nielsen, P. S. Nordholt and C. Orlandi, "MiniLEGO: Efficient Secure Two-Party Computation from General Assumptions," *EUROCRYPT*, pp. 537-556, 2013.

[4] Y. Huang, J. Katz and D. Evans, "Quid-Pro-Quo-tocols: Strengthening Semi-honest Protocols with Dual Execution," *IEEE Symposium on Security and Privacy, SP 2012, May,* pp. 21-23, 2012.

[5] Y. Lindell, B. Pinkas and N. P. Smart, "Implementing two-party computation efficiently with security against malicious adversaries," *Security and Cryptography for Networks*, Springer, pp. 2-20, 2008.

[6] J. Bringer, H. Chabanne and A. Patey, "Privacy-preserving biometric identification using secure multiparty computation: An overview and recent trends," *Signal Processing Magazine, IEEE,* pp. 42-52, 2013.

[7] P. Bogetoft, D. L. Christensen, I. Damgard, M. Geisler et al., "Secure multiparty computation goes live," *Financial Cryptography and Data Security*, Springer, pp. 325-343, 2009.

[8] G. Di Crescenzo, "Private selective payment protocols," *Financial Cryptography*, pp. 72-89, 2001.

[9] S. Even, O. Goldreich and A. Lempel, "A Randomized Protocol for Signing Contracts," *Commun. ACM,* vol. 28, no. 6, pp. 637-647, 1985.

[10] A. Jivanyan and G. Khachatryan, "Efficient Oblivious Transfer Protocols Based on White-Box Cryptography," *AUA Internal reports,* 2013.

[11] V. Kolesnikov and T. Schneider, "Improved Garbled Circuit: Free XOR Gates and Applications," *Automata, Languages and Programming, 35th International Colloquium*, Springer, pp. 486 - 498, 2008.

[12] B. Pinkas, T. Schneider, N. P. Smart and S. C. Williams, "Secure Two-Party Computation Is Practical," *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference*, pp. 250-267, 2009.

[13] V. Kolesnikov, P. Mohassel and M. Rosulek, "FleXOR: Flexible garbling for XOR gates that beats free-XOR," *Advances in Cryptology--CRYPTO 2014*, Springer, pp. 440-457, 2014.

[14] D. Kozen , "Lower bounds for natural proof systems," 18th Annual Symposium on Foundations of Computer Science, IEEE, pp. 254-266, Sep 30, 1977.

[15] C. S. Geol, J. Katz, R. Kumaresan and H.-S. Zhou. "On the security of the “free-XOR” technique," *Theory of Cryptography*, Springer Berlin Heidelberg, pp. 39-53, 2012.

[16] D. Danoyan and T. Sokhakyan, "A Generic Framework For Secure Computations", Proceedings of Russian-Armenian (Slavonic) University 2015 (Physical, mathematical and natural sciences), vol. 2, pp. 14-21, 2015.

[17] J. Massey, G. Khachatrian and M. Kuregian. "Nomination of SAFER+ as a Candidate Algorithm for Advanced Encryption Standard (AES)," Represented at the 1st AES conference, Ventura, USA, August 20-25, 1998

[18] Y. Huang, D. Evans, J. Katz and L. Malka, "Faster Secure Two-Party Computation Using Garbled Circuits," In *USENIX Security Symposium,* vol. 201, no. 1, August 8, 2011.

Submitted 10.10.2015, accepted 25.01.2016

Երկու մասնակցով ընդհանուր օգտագործման անվտանգ հաշվարկների համակարգի լավարկման մեթոդներ

S. Սոխակյան

Ամփոփում

Այս հոդվածում ներկայացված է երկու մասնակցով անվտանգ հաշվարկների համակարգի իրականացում: Առաջարկված համակարգը իրականացնում է Յաոյի հաղորդակարգը և նմանատիպ համակարգերի իրականացման համար առաջին անգամ օգտագործում է սպիտակ արկղի գաղտնագրության մեթոդների վրա հիմնված անտեղյակ փոխանցման նոր հաղորդակարգը, որը խուսափում է հաշվողական տեսանկյունից դանդաղ բաց բանալիով գործողություններից: Բերված են փորձնական արդյունքներ, որոնք ապացուցում են առաջարկված իրականացման արդյունավետությունը:

Методы оптимизации для обобщённой платформы конфиденциальных вычислений с двумя участниками

Т. Сохакян

Аннотация

В этой статье представлена реализация платформы конфиденциальных вычислений общего назначения с двумя участниками. Предложенная платформа реализует протокол Яо с искажёнными схемами, впервые используя протокол забывчивой передачи основанной на методах криптографии белого ящика, который не использует дорогостоящие операции с открытым ключом. Также приводятся экспериментальные данные, показывающие эффективность реализованной системы.