Golden Linear Group Key Agreement Protocol

7

Mohammad Vahidalizadehdizaj and Avery Leider

Abstract

Security in group communication has been a significant field of research for decades. Because of the emerging technologies like cloud computing, mobile computing, ad-hoc networks, and wireless sensor networks, researchers are considering high group dynamics in communication security. Group key agreement protocol is for providing enough security for the group communication. Group key agreement protocol is a way to establish a shared cryptographic key between groups of users over public networks. Group key agreement protocol enables more than two users to agree on a shared secret key for their communications. In this paper we want to introduce a new key agreement protocol (GKA) and a new linear group key agreement protocol (GLGKA). These protocols are proper for the emerging dynamic networks. In GLGKA each member of the group can participate in key generation and management process. Our goal is to provide more efficient key generation approach for group members. This protocol can be used in cloud-based data storage sharing, social network services, smart phone applications, interactive chatting, video conferencing, and etc.

Keywords

Cyber security • Diffie-Hellman • Key agreement protocol • Group key agreement protocol

7.1 Introduction

Security and reliability are two significant factors in modern computing. In this environment most of the services can be provided as shared services. These shared services should have the essential cryptographic factors like data confidentiality, data integrity, authentication and access control to be a secure shared service. It is really important for us to communicate securely over an insecure communication channel. One way to provide this secure channel is using the key agreement protocol method [5].

Note that, digital signature, encryption, and key agreement are three important topics in the field of cryptography. Group key agreement protocol can provide a secure communication channel between more than 2 users over an open network. The idea is having a shared session key between all the members of a group and no one else. Because of the growth of the group-oriented applications like video conferencing, the role of the shared session key between members of a group became critical. A lot of research have been focused on this area so far, but still we need a more efficient and secure solution [3].

Group key agreement protocol is very useful in interactive chatting applications that are really popular these days. Most of the people launch these applications in their cellphones, tablets, notebooks, or their PCs. These applications should provide secure communication channel for their users over an untrusted network. Most of the users

M. Vahidalizadehdizaj (✉) • A. Leider
Department of Computer Science, Pace University, 1 Pace Plaza, New York, NY, USA
e-mail: m.vahidalizadeh@gmail.com; aleider@pace.edu

S. Latifi (ed.), *Information Technology – New Generations*, Advances in Intelligent Systems and Computing 558, DOI 10.1007/978-3-319-54978-1_7

have online profiles and the privacy of these profiles is really important for these users. The application should share a session key between group members. We are going to propose our own group key agreement protocol (GLGKA). In our protocol all users contribute in generation and management of the shared session key [9].

If we want to define a shared session key between two users, we should use a key agreement protocol. One of the most important key agreement protocols is Diffie-Hellman protocol. This protocol was introduced in 1976 by Whitfield Diffie and Martin Hellman. It is a method for defining a shared session key between two users. It uses the public key exchange implementation that was introduced by Ralph Merkle [5].

If we want to define a shared session key for more that two parties, we should use a group key agreement protocol. When the communication channel is not trusted, group key agreement protocols will be useful. In this paper we will introduce a new linear group key agreement protocol named GLGKA. In our protocol, all group members contribute in generation of the shared key. In our protocol there are two sub-protocols for the people who join or leave the group [4].

Defining a shared session key for a group is a complicated task. The challenges are lack of third parties, expensive communication, and limited capabilities of the portable devices. There are also some other challenges for this area that will be discussed later in this paper. For example, in adhoc networks there is not enough trust in the network for communication [2, 4].

In our proposed protocol (GGKA) each member should establish a secure channel with the leader by our proposed secure group communication sub-protocol (SGC). SGC protocol should be used later to transfer the shared session key to the group after a user leaves or joins the group. Our protocol does defines a shared session key for the group with less computation and communication costs. Members may have different devices with different capabilities. Our computations should be fast enough for all kinds of devices. To put it in a nutshell, in our protocol the amount of communication for defining and changing a shared session key for a group is optimized [1, 9].

The rest of the paper is organized as follows. Section 7.2 will review related works. Section 7.3 is about our proposed key agreement and group key agreement protocols. Section 7.4 is our join protocol. Section 7.5 is our leave protocol. Section 7.6 explains about the complexity of our proposed protocols. Section 7.7 is comparison section. Section 7.8 shows our experiments. Finally, Sect. 7.9 concludes this paper.

7.2 Related Works

In this section we are going to review Diffie-Hellman key agreement protocol, which is the most popular protocol in this field. We are also going to review DL08 and KON08 protocols, which are the most efficient group key agreement protocols [8].

7.2.1 Key Agreement Protocol

In this section we review Diffie-Hellmand key agreement protocol, which is the most popular key agreement protocol. Diffie-Hellman is one of the earliest implementations of the public key cryptography, which generates a shared session key between two users. We can use this key later for encrypting or decrypting our information just like a symmetric key. Most of the key agreement protocols use Diffie-Hellman as their basis. RSA also followed Diffie-Hellman method to implement its public key cryptography method [4].

This protocol is based on mathematics. Its fundamental math includes algebra of exponents and modulus arithmetic. To explain how this protocol works, we use Alice and Bob in our example. The goal of this protocol is to agree on a shared secret key between Alice and Bob. Note that, an eavesdropper should not be able to determine the shared session key by observing transferred data between Alice and Bob. Alice and Bob independently generate these keys for themselves in two sides. These symmetric keys will be used to encrypt and decrypt data stream between Alice and Bob. Note that, this key do not travel over the network. The steps of this protocol are shown in Table 7.1. We plan to compare our key agreement protocol (GKA) with this popular key agreement protocol later [8].

Table 7.1 Diffie-Hellman key agreement protocol

Step	Action	Description
1	Alice and Bob agree on two numbers p and g	p is a large prime number and g is called base or generator
2	Alice picks a secret number a	Alice's secret number = a
3	Bob picks a secret number b	Bob's secret number = b
4	Alice computes her public number $X = g^a \ mod \ p$	Alice's public number = X
5	Bob computes his public number $Y = g^b \ mod \ p$	Bob's public number = Y
6	Alice and Bob exchange their public numbers	Alice knows p, g, a, X, Y Bob knows p, g, b, X, Y
7	Alice computes $k_a = Y^a mod \ p$	$k_a = (g^b mod \ p)^a \ mod \ p$ $k_a = (g^b)^a \ mod \ p$ $k_a = (g^{ba}) \ mod \ p$
8	Bob computes $k_b = X^b mod \ p$	$k_b = (g^a mod \ p)^b \ mod \ p$ $k_b = (g^a)^b \ mod \ p$ $k_b = (g^{ab}) \ modp$
9	By the law of algebra, Alice's k_a is the same as Bob's k_b, or $k_a = k_b = k$	Alice and Bob both know the secret value k

7.2.2 Group Key Agreement Protocols

The first group key agreement protocol that we are going to review is DL08. Desmedt and Lange proposed a three-round group key agreement protocol in 2007. This protocol is based on pairings and it is proper for groups of parties with different computational capabilities. In this protocol, a balanced group of n parties should have approximately $n/2$ more powerful parties [9].

The number of computations for calculations of signatures and verifications are important factors in computing the complexity of this protocol. We assume that a Digital Signature algorithm is used by the signing scheme. We can assume that a signature generation has the cost of one exponentiation and a signature verification has the cost of two exponentiation [9].

According to this assumptions, the complexity of this protocol includes total number of $(9n/2) + 2n\lg_3[n/3]$ multiplications, $3n/2$ pairings and $3n/2$ exponentiation. The parties will have to transmit $7n/2$ messages and also receive $3n + n\lg_4[n]$ messages. DL08 is a three-round protocol based on the Burmester-Desmedt scheme that achieves the best performance from aspect of cost [9, 10].

The next protocol that we want to review is KON08. It is a cluster-based GKA protocol proposed by Konstantinou in 2010. It is based on Joux's tripartite key agreement Protocol. It has two variants: contributory and non-contributory. This protocol assumes that nodes are clusters with two or three members [9].

In the lower levels nodes belong to only one cluster. In upper levels nodes belong to two clusters. Authentication can be provided by the use of an authenticated version of Joux protocol. Authentication method does not influence the number of rounds or the communication cost of this protocol. In particular, the protocol has $log_2 n/3$ rounds and $4n$ messages should have been transmitted. In the authenticated version, the group has to perform no more than $5n$ scalar multiplications and $11n/2$ pairing computations [2, 6, 9].

7.3 Golden Linear Group Key Agreement Protocol

We divided our main protocol into two sub-protocols. The first one is secure channel sub-protocol. The second one is initiation sub-protocol. Our goal is to define a shared session key for a group of users by these two sub-protocols. In the secure channel sub-protocol, we will use our own key agreement protocol named GKA (Golden Key Agreement Protocol). Based on GKA protocol, all members and leader of the group should share three numbers a, p, and q at the beginning [4, 6, 9].

7.3.1 Golden Key Agreement Protocol (GKA)

This protocol is based on mathematics. The fundamental math includes algebra of logarithms and modulus arithmetic.

Table 7.2 Golden key agreement protocol

Step	Action	Description
1	Alice and Bob agree on three numbers a, p, and q	q is a large prime number $a = p^n$ p = 2, 3, . . ., n n, u, and v = 1, 2, 3, . . ., n
2	Alice picks a secret number u	Alice's secret number = u u mod q is not zero
3	Bob picks a secret number v	Bob's secret number = v v mod q is not zero
4	Alice computes her public number $A = ((u \bmod q) \times log_p a) \bmod q$	Alice's public number = A $= (log_p a^{(u \bmod q)}) \bmod q$
5	Bob computes his public number $B = ((v \bmod q) \times log_p a) \bmod q$	Bob's public number = B $= (log_p a^{(v \bmod q)}) \bmod q$
6	Alice and Bob exchange their public numbers	Alice knows u, a, p, q, A, and B Bob knows v, a, p, q, A, and B
7	Alice computes $k_a = ((u \bmod q) \times B) \bmod q$	$k_a = ((u \bmod q) \times (log_p a^{(v \bmod q)})) \bmod q$ $k_a = (log_p a^{(v \bmod q) \times (u \bmod q)}) \bmod q$ $k_a = (log_p a^{((u \times v) \bmod q)}) \bmod q$
8	Bob computes $k_b = ((v \bmod q) \times A) \bmod q$	$k_b = ((v \bmod q) \times (log_p a^{(u \bmod q)})) \bmod q$ $k_b = (log_p a^{(u \bmod q) \times (v \bmod q)}) \bmod q$ $k_b = (log_p a^{((v \times u) \bmod q)}) \bmod q$
9	By the law of algebra, Alice's k_a is the same as Bob's k_b, or $k_a = k_b = k$	Alice and Bob both know the secret value k

For this discussion we will use Alice and Bob as an example. The goal of this process is agreeing on a shared secret session key between Alice and Bob. In this process eavesdroppers should not be able to determine the shared session key. This shared session key will be used by Alice and Bob to independently generate the keys for each side. These keys will be used symmetrically to encrypt and decrypt data stream between Alice and Bob. Note that, our shared session key should not travel over the network. This new key agreement protocol is described in Table 7.2.

Table 7.3 is an example of generating a shared session key between Alice and Bob based on GKA protocol. In this example we choose n = 2, p = 2, and q = 7($a = p^n = 4 = 2^2$).

7.3.2 Secure Channel Sub-protocol

In this part the leader will establish a secure connection to each one of users in the group. Firstly, each user will choose a private key (user i will choose private key p_i). The leader also will choose a private key (p_c). Then, each user will send $A_i = ((p_i \bmod q) \times log_p a) \bmod q$ to the leader. Also the

Table 7.3 GKA example

Step	Action
1	Alice and Bob agree on n = 2, p = 2, a = 4 and q = 7
2	Alice picks a secret number u = 3
3	Bob picks a secret number v = 4
4	Alice computes her public number $A = ((3\ mod\ 7) \times log_2 2^2)\ mod 7 = 3$
5	Bob computes his public number Bob's public number = B $A = ((4\ mod\ 7) \times log_2 2^2)\ mod 7 = 1$
6	Alice and Bob exchange their public numbers Alice knows u, a, p, q, A, and B Bob knows v, a, p, q, A, and B
7	Alice computes $k_a = ((u\ mod\ q) \times (log_p a^{(v\ mod\ q)}))\ mod q$ $k_a = ((3\ mod\ 7) \times (1)\ mod 7 = 3$
8	Bob computes $k_b = ((v\ mod\ q) \times (log_p a^{(u\ mod\ q)}))\ mod q$ $k_a = ((4\ mod\ 7) \times (6)\ mod 7 = 3$
9	By the law of algebra, Alice's Alice and Bob both know the k_a is the same as Bob's k_b secret value k or $k_a = k_b = k = 3$

leader will broadcast $A_c = ((p_c\ mod\ q) \times log_p a) mod\ q$ to all the group members [6, 7].

Then, each one of the users will calculate $k_i = ((p_i\ mod\ q) \times A_c) mod\ q$. Then, the leader will calculate $k_{c_i} = ((p_c\ mod\ q) \times A_i) mod\ q$. As a result of this sub-protocol, the leader will have a secure line with each one of the members [8].

7.3.3 Initiation Sub-protocol

In this part, the leader will generate a shared session for the users. Firstly, each user chooses his own private key p_i and send it to the leader through the secure channel. After that, the leader will generate a shared session key for the group based on the equation below and will send it to the members through their secure channels.

$$k = (log_p a^{(p_1 p_2 \cdots p_n)})\ mod\ q$$
$$a = p^n \Rightarrow k = (log_p p^{[n(p_1 p_2 \cdots p_n)]})\ mod\ q$$
$$\Rightarrow k = ((p_1 p_2 \ldots p_n) \times log_p p^n)\ mod\ q$$

Note that, this shared session key is only for current users of the group. if anyone leaves or joins the group, the shared group key should be updated. In the next sections we are going to review our join and leave sub-protocols. Join protocol is useful when a new member joins the group and leave protocol is useful when a member leaves the group.

7.4 Join Protocol

When a new user ($user_{n+1}$) want to join a group, he should establish a secure channel with leader of the group. He uses our secure channel protocol (GKA) for this step. Then, he will send his private key with positive sign to the leader through his secure channel. Then, the leader will update his shared session key by multiplying it to the received number $((p_{n+1}\ mod\ q))$, since the sign is positive. After that, the leader will broadcast the updated shared session key to the group members. The key will be updated for the leader, members of the group, and the new member Based on the equation below [5].

$$Current\ k = (log_p a^{(p_1 p_2 \cdots p_n)})\ mod\ q$$
$$New\ k = (p_{n+1}\ mod\ q) \times (log_p a^{(p_1 p_2 \cdots p_n)})\ mod\ q$$
$$= (p_{n+1} \times log_p a^{(p_1 p_2 \cdots p_n)})\ mod\ q$$
$$= (log_p a^{(p_1 p_2 \cdots p_n p_{n+1})})\ mod\ q$$

7.5 Leave Protocol

When an existing user (for example $user_n$) want to leave the group, he should inform the leader before he leaves. He should send his private key with negative sign $(-1 \times p_n)$ to the leader through his secure channel. Then, the leader will update his private key by dividing it by the received number. Note that, the leader ignores the negative sign. This negative sign is only symbol of leaving. After that, the leader will broadcast the new shared group key. After that, the leader and the group members will have the same updated key based on the equation below [2, 6, 7].

$$Current\ k = (log_p a^{(p_1 p_2 \cdots p_n)})\ mod\ q$$
$$New\ k = (log_p a^{(p_1 p_2 \cdots p_n)})\ mod\ q\ /\ (p_n\ mod\ q)$$
$$= (log_p a^{(p_1 p_2 \cdots p_n)/p_n})\ mod\ q$$
$$= (log_p a^{(p_1 p_2 \cdots p_{n-1})})\ mod\ q$$

7.6 Complexity

In this section, we are going to analyze the cost of our protocol. We start from secure channel sub-protocol. In the secure channel sub-protocol, the leader should establish a secure connection with each one of the members based on the private key of the member. The leader will have two calculations and one broadcast. Each member will have two calculations and will send one message and receive one

message. In the initiation sub-protocol, the leader will have one calculation and one broadcast. He will also receive n messages from members of the group. Each member will send one message to the leader and receive one message from the leader in this sub-protocol.

In the join protocol, the leader will have three calculations (two for secure channel sub-protocol and one for updating the shared group key), and will send one message to the new user and one broadcast to all members of the group. Each user will only receive one message. In the leave protocol, the leader will have one calculation and one broadcast to members of the group. The leaving member should send one message to the leader and each member except the leaving member should receive one message. In conclusion, our protocol will send $3n$ messages and receive $3n$ messages. Our protocol should do $2n + 2$ calculations for a group of n parties.

7.7 Comparison

In this part we are going to compare our group key agreement protocol with top two group key agreement protocols based on [9]. These protocols are DL08 and KON08. We compare these protocols from aspect of number of rounds, total number of sent messages, total number of received messages, and computations cost. You can see the result of these comparisons in Table 7.4.

Table 7.4 Group key agreement protocols comparison result

Protocol	Rounds	Sent Messages	Received Messages	Computations
DL08	3	$7n/2$	$3n + nlog_4n$	$15n/2 + 2n$ $[log_4n]$
KON08	$log_2n/3$	$4n$	$4n$	$21n/2$
GLGKA	2	$3n$	$3n$	$2n + 2$

As you see in Table 7.4, the comparison result shows that our group key agreement protocol is better than two other protocols from aspect of total number of sent and received messages. Our protocol is also better from them from aspect of computation cost.

7.8 Experiment

In this part we are going to compare Golden Key Agreement Protocol with Diffie-Hellman, which is the most popular key agreement protocol. We implemented these two algorithms and tested them with different parameters. We did all the experiments with a laptop with core-i7 CPU (2670QM 2.20 GHz), 8 GB of DDR3 Ram, and windows seven 64-bit operating system.

The first experiment was with q(GKA) = 15, 485, 863, p(DH) = 15, 485, 863, a = Random(2, 10), p(GKA) = Random(2, 10), g(DH) = Random(2, 10), and n(GKA) = 1. You can see the result of these experiments in Fig. 7.1. We had a loop in this experiment. You can see the number of iterations in each experiment. The second experiment was with q(GKA) = 982, 451, 653, p(DH) = 982, 451, 653, a = Random(2, 10), p(GKA) = Random (2, 10), g(DH) = Random(2, 10), and n(GKA) = 1. You can see the result of these experiments in Fig. 7.2. We had a loop in this experiment. You can see the number of iterations in each experiment.

In the third and last experiment, we ran our protocol against Diffie-Hellman 16 times. Each time we recorded the running time of both protocols. You can see result of these experiments in Fig. 7.3. As you see int Fig. 7.3, our protocol is faster than Diffie-Hellman. In these sixteen experiments the average running time of Diffie-Hellman was 801,268.8125 nanoseconds and Average running time of our protocol in these experiments was 86,662.5625 nanoseconds.

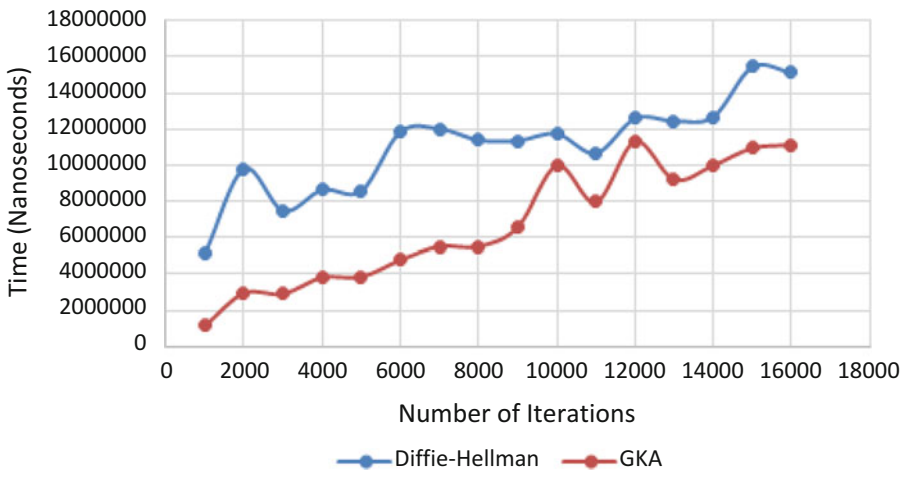

Fig. 7.1 Comparison of golden key agreement protocol with Diffie-Hellman (more than one iteration)

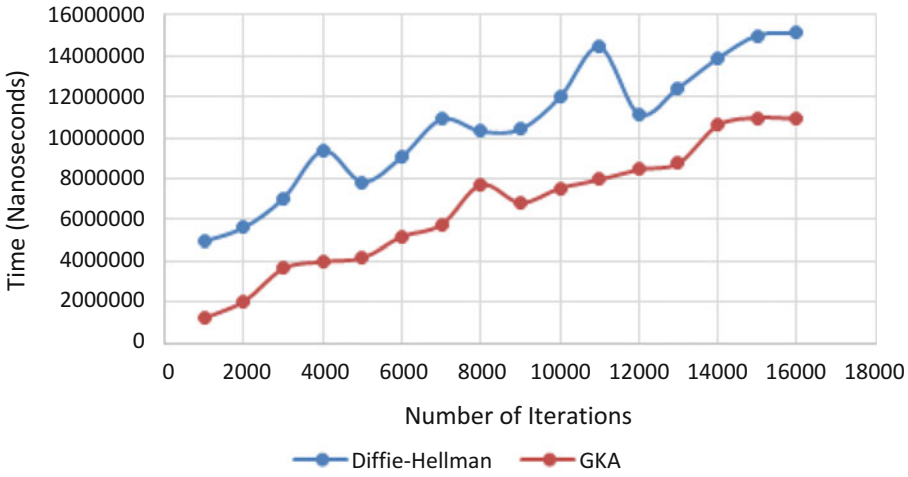

Fig. 7.2 Comparison of golden key agreement protocol with Diffie-Hellman (more than one iteration)

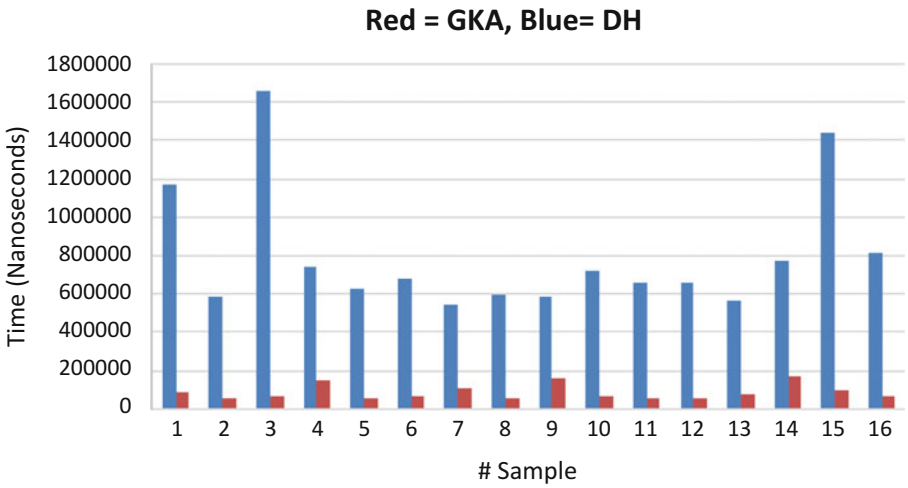

Fig. 7.3 Comparison of GKA with Diffie-Hellman (one iteration)

7.9 Conclusion

We recommended two protocols in this paper. The first one was golden key agreement protocol. We used algebra of logarithms and modulus arithmetic in this protocol to improve the performance. We compared this protocol with Diffie-Hellman (using exponentiation) and we observed that our protocol is faster. We selected Diffie-Hellman for our comparison, since it is the most popular key agreement protocol. We also suggested a group key agreement protocol named golden linear group key agreement protocol. We compared our group key agreement protocol with DL08 and KON08 from aspect of the number of rounds, total number of sent messages, total number of received messages, and computation cost. DL08 and KON08 are two of the best group key agreement protocols available based on. As you seen in Table 7.4, our group key agreement protocol was better than other protocols.

References

1. Cormen, T. H., Stein, C., Rivest, R. L., & Leiserson, C. E. (2001). *Introduction to algorithms* (2nd ed.). Boston/Burr Ridge/Montréal/ McGraw-Hill Higher Education.
2. Dizaj, M. V. A. (2011). New mobile payment protocol: Mobile pay center protocol 4 (MPCP4) by using new key agreement protocol: VAC2. In *IEEE 3rd International Conference on Electronics Computer Technology (ICECT)*, April 8–10, 2011 (Vol. 2, pp. 67–73).
3. Dizaj, M. V. A., & Geranmaye, M. (2011). New mobile payment protocol: mobile pay center protocol 5 (MPCP5) by using new key agreement protocol: VG1. In *IEEE 3rd International Conference on Computer Modeling and Simulation (ICCMS)* (Vol. 2, pp. 246–252).
4. Dizaj, M. V. A., & Geranmaye, M. (2011). New mobile payment protocol: mobile pay center protocol 6 (MPCP6) by using new key agreement protocol: VGC3. In *IEEE 3rd International Conference on Computer Modeling and Simulation (ICCMS)* (Vol. 2, pp. 253–259).
5. Dizaj, M. V. A., Moghaddam, R. A., & Momenebellah, S. (2011). New mobile payment protocol: Mobile pay center protocol (MPCP). In *IEEE 3rd International Conference on Electronics Computer Technology (ICECT)*, April 8–10, 2011 (Vol. 2, pp. 74–78).

6. Dizaj, M. V. A., Moghaddam, R. A., & Momenebellah, S. (2011). New mobile payment protocol: Mobile pay center protocol 2 (MPCP2) by using new key agreement protocol: VAM. In *IEEE Pacific Rim Conference on Computers and Signal Processing (PacRim)*, August 23–26, 2011 (pp. 12–18).
7. Lee, E. -J., Lee, S. -E., & Yoo, K. -Y. (2008). A certificateless authenticated group key agreement protocol providing forward secrecy. In *2008 International Symposium on Ubiquitous Multimedia Computing* (pp. 124–129).
8. Vahidalizadehdizaj, M., & Tao, L. (2015). A new mobile payment protocol (GMPCP) by using a new key agreement protocol (GC). In *IEEE International Conference on Intelligence and Security Informatics (ISI)*, May 27–29, 2015 (pp. 169–172).
9. Vahidalizadehdizaj, M., & Tao, L. (2015). A new mobile payment protocol (GMPCP) By using a new group key agreement protocol (VTGKA). In *IEEE 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*.
10. Yao, G., & Feng, D. (2010). A complete anonymous group key agreement protocol. In: *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing* (pp. 717–720).