TELKOMNIKA, Vol.16, No.6, December 2018, pp.2668~2675
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v16i6.9811

Numerical Method for Evaluating E-cash Security

Dany Eka Saputra*[1], Sarwono Sutikno[2], Suhono Harso Supangkat[3]
[1]Departement of Informatics, STMIK "AMIKBANDUNG", Bandung, Indonesia
[1,2,3]School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, Indonesia
*Corresponding author, e-mail: dekastra@stmik-amikbandung.ac.id

Abstract

Security evaluations of electronic cash (e-cash) schemes usually produce an abstract result in the form of a logical proof. This paper proposes a new method of security evaluation that produces a quantitative result. The evaluation is done by analyzing the protocol in the scheme using the Markov chain technique. This method calculates the probability of an attack that could be executed perfectly in the scheme's protocol. As proof of the effectiveness of our evaluation method, we evaluated the security of Chaum's untraceable electronic cash scheme. The result of our evaluation was compared to the evaluation result from the pi-calculus method. Both methods produced comparable results; and thus, both could be used as alternative methods for evaluating e-cash security.

Keywords*: evaluation, markov chain, protocol, security.*

1. Introduction

Security is the most important aspect of an electronic cash (e-cash) scheme. A lack of security in an e-cash scheme presents users with the risk of financial loss. As such, a scheme must prove that it is able to mitigate or eliminate these risks. The security evaluation of an e-cash scheme provides the basis of whether a scheme can be implemented. The evaluation assesses the scheme's protection from attacks on its operation. To be implemented, a scheme does not necessarily need to meet all the security criteria. The operational environment and the scheme's objectives determine the security criteria that need to be met.

The common security criteria in e-cash are: double-spending, anonymity, forgery, and exculpability [1-3]. To fulfil the criterion on double spending, the e-cash scheme should not allow users to use the same e-cash data in more than one transaction. Anonymity requires that no one should be able to determine the owner of the e-cash from the data itself. This may be further classified into smaller categories depending on the level of secrecy required [2]. The e-cash scheme should also prevent forgery; i.e., the generating of e-cash data without involving the proper protocol or involving an illegal party. The last criterion, exculpability, requires that the e-cash issuer or manager should not be able to accuse an honest user of double spending. Also, the issuer should not be able to create e-cash data without the request of a legitimate user. The most common evaluation method of an e-cash scheme is the random oracle model, which evaluates the cryptographic scheme that formed the basis for the e-cash scheme [4-9]. The analysis is conducted via a simulation between an adversary (an entity with the intention to attack the scheme), and an oracle (a query-processing machine). The random oracle model provides logical proof to the security, but not a quantifiable number that measures the security of the system. The random oracle model assumes an idealized environment where the adversary is bound to a fully restricted environment. The adversary must follow the simulation rules to send a query to the oracle. In this simulation, no other external entity can intervene with the communication between the adversary and the oracle. Due to these assumptions, the security of a scheme regarding the random oracle model might differ significantly from the security of its actual implementation [10]. Some researchers also believe that the random oracle model is too strong as a proof. It creates a faction that reject the usage of random oracle model as a security proof for any cryptographic implementation [11].

The usage of random oracle model usually relies on some mathematical assumption [8, 12]. In this kind of security model, the security of the scheme is based on the difficulty to solve the mathematical problem used as the assumption. For example, many

Received May 6, 2018; Revised September 27, 2018; Accepted October 24, 2018

schemes rely on a variation of Diffie-Hellman problem as the assumption of its security model. Since the difficulty of Diffie-Hellman problem is quite well-known, then it is quite difficult to break the e-cash scheme that implement Diffie-Hellman problem as its main security mechanism. Another approach is to evaluate the security of cryptographic primitive used in the e-cash scheme, such as the digital signature scheme. As we can see in various works in digital signature [13-16], the process to evaluate the security is using logical proof. The result provides a strong basis to determine the security of the cryptographic primitive, and in the end ensure the security of the e-cash scheme that using the primitive. Dreier et al., [17] propose another method of e-cash security evaluation using pi-calculus. Rather than evaluating the cryptographic scheme using the random oracle model, their method evaluates the protocol, which represents the operational step-by-step procedure of the e-cash scheme. As such, the pi-calculus method can simulate the implementation condition more closely than the random oracle model can. In their paper, Dreier et al., use an implementation of this method, ProVerif, to analyze the implementation of Chaum's scheme [1].

Both the random oracle model and the pi-calculus method produce qualitative results as proofs of security. The results of these evaluation methods do not provide a number which could be used as a basis of measurement or comparison of security levels. Instead, they only provide logical reasoning that states why the scheme is secure. These logical proofs can be difficult to understand for someone without a background in computer science or mathematics. This paper proposes a method for evaluating the security of e-cash which produces a numerical result. This method uses a similar approach to the method in [17], by evaluating the protocol. Our method calculates the probability of a security risk using a Markov chain technique. This method provides a quantitative result that is indicative of the implementation conditions. In this paper, we also provide a sample calculation for this evaluation method by measuring the security of Chaum's untraceable e-cash scheme [1]. The evaluation result is then compared to the results of Dreier et al.'s pi-calculus evaluation of the same e-cash scheme.

2. Proposed Method

The proposed method of evaluation consists of four phases: redefining the protocol into its formal description, constructing the attack scenario, constructing the transition matrix, and calculating the Markov chain probability. This method assumes that it is possible to calculate the probability of an attack on the cryptographic scheme used in an e-cash scheme. The phases of our evaluation method can be seen in Figure 1.

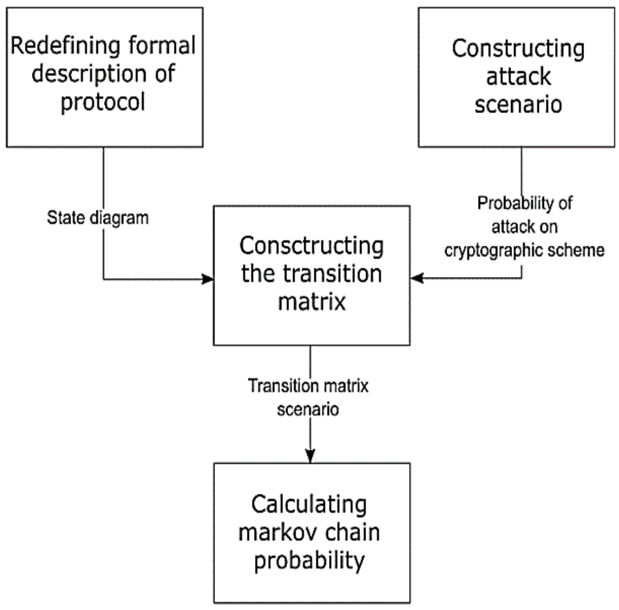
Redefining formal description of protocol
Constructing attack scenario
State diagram
Probability of attack on cryptographic scheme
Consctructing the transition matrix
Transition matrix scenario
Calculating markov chain probability

Figure 1. Proposed method of evaluation

2.1. Redefinition of Formal Description

The first phase of the evaluation is to redefine the protocol in the e-cash scheme to make it suitable for evaluation using Markov chain techniques. The Markov chain evaluates the probability of transition between states, while e-cash protocols are usually in the form of step-by-step algorithms which do not describe the protocol state. Therefore, the common description of e-cash protocol cannot be used as a basis for protocol evaluation using Markov chain techniques. For our method, we use a finite state model as the formal description model. The protocol is modelled as a state diagram which shows states and the transitions between them. The state diagram provides the basis for forming the transition matrix. Since e-cash protocols are usually not state-oriented, determining the states from the protocol is often not a straightforward process. One possible approach is to evaluate the purpose of each data transaction in a protocol to determine the state involved and its transitions.

2.2. Constructing Attack Scenarios

This phase aims to generate the success probability of an attack on the cryptographic scheme used in an e-cash scheme. Most e-cash schemes rely on one or more cryptographic schemes for their basic security. The strength of a cryptographic scheme depends on the kind of adversary it is protected from. The success probability of an attack might differ depending on which party involved in the process is honest. Even if the calculation shows that the probability of a successful attack on a cryptographic scheme is unacceptable, the e-cash scheme can remain secure. It may have some other mechanism in its protocol to reduce or mitigate the risk associated with the attack.

2.3. Constructing the Transition Matrix

In this phase, we build transition matrices based on the results of the first and the second phase. A transition matrix is built for each attack scenario generated from the second phase. This matrix will be used as the base tool for calculations in the next phase. A state in an e-cash protocol can usually only change into two other states. If the output of the state complies with the rule of the protocol, it changes into the next correct state. If the output does not comply with the rule, it changes into the "ABORT" state and ends the protocol. As such, the transition matrix can be represented as follows:

$$P = \begin{array}{c} \\ S_1 \\ S_2 \\ A \end{array} \begin{array}{c} \begin{array}{ccc} S_1 & S_2 & A \end{array} \\ \begin{pmatrix} 0 & a & 1-a \\ b & 0 & 1-b \\ 1 & 0 & 0 \end{pmatrix} \end{array} \tag{1}$$

The equation above involves three states: S_1, S_2, and A. States S_1, S_2 are the proper state that the protocol should follow. State A is the "ABORT" state which becomes the end state if the proper condition is not met during the protocol run. The variables a, b represent the probabilities that a state will change into the next state following the proper protocol. The values of a and b are the same as the probabilities of an attack on the cryptographic scheme calculated in the previous phase. This also depends on which attack scenario is being evaluated. For example, if all entities involved in this protocol are honest, then the $a = 1, b = 1$. If there is a dishonest entity, then the values of a, b become the probability of that entity breaking the cryptographic scheme.

2.4. Calculation of Markov Chain Probability

The Markov chain probability determines the security of an e-cash scheme against certain attack scenarios. The security relates to the probability of an e-cash scheme completing its protocol (not ending in "ABORT" state) in every attack scenario. The probability is calculated using a general Markov chain, as follows:

$$p_{ij}^n = p'_{ij} \in P^n \tag{2}$$

where n denotes the number of steps needed to complete the protocol normally. The calculation only sees the probability of a transition from the protocol's start state (i) to its end state (j) in n steps. The resulting probability is used as a quantitative basis to determine the security of the scheme.

3. Application in Evaluating Chaum's E-cash Scheme

In this section, we evaluate the security of Chaum's offline e-cash scheme [1] using our method. The results of this evaluation are then compared to the evaluation done by Dreier et al., using the pi-calculus method [17] to gauge the effectiveness of our method.

The analysis focuses only on the payment protocol of Chaum's e-cash scheme. The scheme has another protocol, the withdrawal protocol, which is used in creating new e-cash data. The security of the withdrawal protocol does not contribute directly to the security of the scheme since the presence of a dishonest entity in this protocol results in neither gain nor loss.

3.1. Redefining the Formal Description

Although the payment protocol involves three entities (user, merchant, and bank), our evaluation sees the whole protocol as a single system, as the state diagram describes how the whole protocol works rather than how each entity behaves.

The flow of the payment protocol in Chaum's scheme is as follows:

a. User sends e-cash data C to Merchant.
b. Merchant selects and sends random binary strings $z_1, z_2, \dots, z_{k/2}$ ($z_i \in \{0,1\}$)
c. For each binary string sent by Merchant, User sends back the appropriate proof-of-ownership.
d. If the proof does not match C, Merchant rejects the transaction and ends the protocol. Otherwise, the protocol proceeds to next step.
e. Merchant sends C and User's proof to Bank.
f. If Bank verifies the correctness of C, it settles the transaction. Otherwise, it aborts and traces the User.

The result of redefinition of this protocol can be seen in Figure 2; it has five states and five transitions. Step (a) is described as the start of the protocol in the form of the REQUEST state. Steps (b to d) verify the validity of C. These steps are represented as the CHALLENGE state of the protocol. Step (e) describes the state that verifies the signature of Bank and checks for double spending.

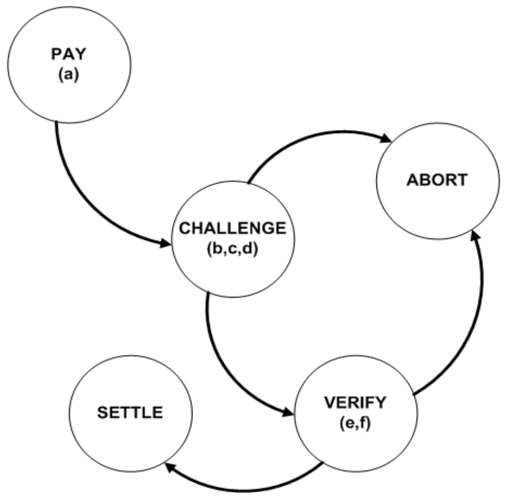

Figure 2. State diagram of payment protocol

3.2. Constructing Attack Scenarios

To construct the attack scenario, we needed to choose some implementation details for the e-cash scheme. In Chaum's scheme, a digital signature is needed to create the e-cash data. This digital signature protects the scheme from forgery. For our evaluation, we chose to use the RSA signature scheme with modulus N of order 10^{80} and security parameter $k = 50$. Both values are secure enough to be chosen as implementation values. This determines the length of e-cash data and the number of random binary string needed during the CHALLENGE state.

We considered three attack scenarios: double spending, forgery, and exculpability. When calculating the success probability of each scenario, we also considered the entities that perform the attack.

3.2.1. Double Spending

The success probability of double spending depends on two things. First, the time elapsed between two VERIFY state that uses the same e-cash data. The longer the time, the smaller the success probability. Second, the probability of the merchants in both transactions using the same permutation of binary string in the CHALLENGE state. If this happens, the Bank still can determine that the data has been used during the VERIFY state in the second transaction occurs. However, since the Bank only possesses half of the proof-of-ownership for each z_i due the nature of the blind signature used in the Withdrawal protocol, the Bank cannot determine the perpetrator of the double spending. Steps e and f in the payment protocol can be conducted much later than the actual transaction (due to the scheme being offline), so the user could still get away with double spending.

The probability of this attack can be formulated as follow:

$$P_{db} \approx P\left(z_i = z_j\right).P(T_e) \tag{3}$$

with i, j denoting the two-different payment protocol involved in double spending and T_e denoting the time elapsed between the two protocols. The first part of equation ($P\left(z_i = z_j\right)$) represents the probability of combination of both proof i, j are the same. The second part ($P(T_e)$) represents the probability due to elapsed time.

In our evaluation, we did not consider the scenario where the User and Merchant collude to cheat the honest Bank by creating two different transactions and processes, both VERIFY states at the same time. In this scenario, although the probability of success is $P_{db} \approx 1$ and the Bank would not be able to determine which User is involved in this double spending, the Bank could still detect the existence of double spending. It might not credit the Merchant account with settlement, leaving only the User with the benefit. As such, this scenario is unlikely.

As such, we consider only the attack scenario where only the User is dishonest. The probability of both Merchants using the same binary string array corresponds to $\frac{k}{2} = 25$, which results in $P(z_i = z_j) = 2.98 \times 10^{-7}$. The time elapsed component of (3) can be approximated as $P(T_e) \approx 0$, since the time needed to change Merchant is likely to be relatively large. However, for our calculation we will use $P(T_e) = 10^{-6}$, to represent a number that is practically zero but still large enough to be used in the calculation. The success probability of double spending by the User then becomes $P_{db} = 2.98 \times 10^{-13}$.

3.2.2. Forgery

Forgery can only be initiated by the User. To forge the e-cash data, the User needs to falsify the Bank signature. The probability of breaking the RSA signature is the same as the probability of factoring N [18]. Given sufficiently long amount of time, the factoring theoretically can always succeed [19]. However, if we ignore the time needed to factor the number, the probability of breaking RSA is:

$$P_f = \frac{1}{\sqrt{\frac{N}{2}}} \approx 10^{-40} \tag{4}$$

The process of verifying the signature happens in the VERIFY state by the Bank. In the CHALLENGE state, the Merchant only checks for the proper form of C.

3.2.3. Exculpability

The exculpability scenario could happen if the Bank was dishonest. To accuse an honest User for double spending, the Bank needs to break the blind signature. In common double-spending scenarios, the Bank could determine the User from the User's response in the CHALLENGE state, which is transferred to the Bank in the VERIFY state. The proof-of-ownership for any e-cash data in [1] consists of two parts: a_i and $a_i \oplus \left(u \parallel (v + i)\right)$, where a_i is a random string/number chosen in withdrawal by User. The identity of the User is stored in u as an account number, and v denotes the counter associated with it. If the Bank has both pieces of information, it can extract u to identify the User. To accuse an honest user of double spending, the Bank must acquire *u* from the second part of the proof without knowing a_i. The probability of successfully doing such an operation is proportional to the bit length of a_i. For the purpose of this evaluation, let $0 \leq a_i \leq 1024$, which makes the length of $a_i = 2^{10}$. The probability of successfully extracting the User identity u without having a_i is:

$$P_e = \frac{1}{l} = 2^{-10} \tag{5}$$

where l denotes the length of a_i. Having the Merchant collude with the Bank does not change this probability. To provide both parts of proof to the Bank, the Merchant needs to force the User to double spend the e-cash data, which is unlikely to be done by the user under normal circumstances.

3.3. Constructing the Transition Matrix

The transition matrix for Chaum's payment protocol can be generated using the state diagram in Figure 2, together with the attack scenarios and their probabilities in section 4.2 the general transition matrix can be expressed in Table 1.

Table 1. Transition Matrix of Chaum's Payment Protocol

Origin State	PAYMENT	CHALLENGE	VERIFY	SETTLE	ABORT
PAYMENT	0	1	0	0	0
CHALLENGE	0	0	a	0	1-a
VERIFY	0	0	0	b	1-b
SETTLE	0	0	0	0	0
ABORT	0	0	0	0	0

As stated in (1), the values of a and b depend on the attack scenarios. The probability of each scenario becoming the value of either a or b, depends on where the cryptographic attack is evaluated. The value of a and b for this evaluation is shown in Table 2.

Table 2. The values of Transition Matrix Variables per Scenario

Scenario	Value
Double spending	$a = 1; b = 2.98 \times 10^{-13}$
Forgery	$a = 1; b = 10^{-40}$
Exculpability	$a = 1; b = 2^{-10}$

3.4. Calculation of the Markov Chain

For each attack scenario in Table 1, we calculated the probability that the protocol will end in the proper state using several normal steps. The start state of the evaluated protocol is "PAYMENT" state, as can be seen in Figure 2. The protocol should end its process in "SETTLE" state after three steps in the normal condition. Should an attack occur, the probability of this protocol ending in "SETTLE" state should be low; and the probability that it will end in the "ABORT" state should be high.

By using (2), the probability of success for each attack can be seen in Table 3. Double spending has the probability of 2.646×10^{-38} to succeed. Forgery has lower probability $p_{ij} \approx 10^{-120}$. The exculpability scenario has the highest probability among the three scenarios, but it is still low. It can be concluded that it is improbable that any of the three attack scenarios would be carried out successfully.

Table 3. The Success Probability in Each Attack Scenario

Scenario	Probability
Double spending	$p_{ij} \approx 2.646 \times 10^{-38}$
Forgery	$p_{ij} \approx 10^{-120}$
Exculpability	$p_{ij} \approx 9.313 \times 10^{-10}$

3.5. Comparison of Result with Pi-calculus Method

Table 4 shows a comparison of results from the method used by Drier et al., and our method. Dreier et al., evaluated Chaum's scheme with security properties defined [20]; whereas, our evaluation does not strictly follow those definitions. However, the results of the two evaluation methods should still be comparable. The results from Drier et al. are shown as "hold" (where the scheme meets the security criterion for that particular scenario) and "fail" (where the scheme does not meet the security criterion). Our results state whether the success of the attack scenario is probable, based on the probabilities calculated earlier.

Table 4. Comparison of Results

Property	Dreier et al.,	Our Method
Forgery	Fail	Improbable
Double Spending	Hold	Improbable
Exculpability	Hold	Improbable
Weak Anonymity	Hold	Not applicable
Strong Anonymity	Hold	Not applicable

The evaluation results of double spending and exculpability are similar in both methods. It is useful to note that Dreier et al., evaluate the process of double spending only at the interaction between User and Merchant. The Merchant cannot determine whether e-cash data has been spent before. In our method, we evaluate the probability of double spending not only at the interaction of User and Merchant, but also at the interaction of Merchant and Bank.

It can be seen from Table 4, that the two methods seem to differ in their results for forgery. Our method indicates that forgery is improbable, but Dreier et al., fails the scheme in this criterion. This is due to the different definitions of forgery used in these two evaluations. Dreier et al., define forgery as the double spending of e-cash data, while we use the standard definition of forgery: the creation of e-cash data by the User without using the Withdrawal protocol. The definition used by Dreier et al., only covers the first half of the process that is defined in our definition of forgery.

Our method cannot evaluate the anonymity properties of an e-cash scheme. Our method only evaluates the scheme's protocol, as the anonymity attack is usually conducted outside the protocol of e-cash by evaluating the receipt data or the e-cash data itself.

4. Conclusion

We have presented a numerical method for the security evaluation of an e-cash scheme. We have shown that this method can produce a quantitative result to evaluate the security level. The usage of a Markov chain for evaluation is effective as it can be used to evaluate the security of the e-cash scheme. From the comparison made with the pi-calculus method in evaluating Chaum's e-cash scheme, the results are similar. However, there is a difference in the result of forgery evaluation between our method and the pi-calculus method. The difference comes from the different definitions of forgery used by the two methods. Intrinsically, the results do not contradict each other.

Yet, our method is unable to evaluate the anonymity property of the e-cash scheme. This limitation comes from the fact that our method only evaluates the protocol and the events directly related to it. Attacks on anonymity lie outside the boundary of the protocol. Anonymity is dependent on how the scheme stores its user identities; thus, it is only related to the cryptographic scheme and not to the protocol that is used in the e-cash scheme.

References

[1]. Chaum D, Fiat A, Naor M. *Untraceable electronic cash*. In Proceedings on Advances in Cryptology. 1990. New York.

[2]. Blazy O, Canard S, Fuchsbauer G, Gouget A, Sibert H, Traoré J. *Achieving optimal anonymity in transferable e-cash with a judge*. In International Conference on Cryptology in Africa. 2011: 206-223.

[3]. Canard S, Pointcheval D, Sanders O, Traoré J. *Divisible e-cash made practical*. In IACR International Workshop on Public Key Cryptography. Springer. 2015: 77-100.

[4]. Bellare M, Rogaway P. *Random oracles are practical: A paradigm for designing efficient protocols*. In Proceeding of the 1st ACM conference on Computer and Communications Security. 1993: 62-73.
[5]. Hess F. *Efficient identity based signature schemes based on pairings*. In International Workshop on Selected Areas in Cryptography Springer. 2002: 310-324.
[6]. Fan CI, Sun WZ, Hau HT. Date attachable offline electronic cash scheme. *The Scientific World Journal*. 2014: 1-19.
[7]. Kang B, Xu D. Secure electronic cash scheme with anonymity revocation. *Mobile Information System*. 2016.
[8]. Cheng L, Wen Q. Cryptanalysis and improvement of a certificateless partially blind signature. *IET Information Security*. 2015; 9(6): 380-386.
[9]. Abouelseoud Y. *New blind signcryption schemes with application to e-cash systems. In Computing.* Communication and Networking Technologies (ICCCNT), 2014 International Conference on. IEEE. 2014: 1-6.
[10]. Canetti R, Goldreich O, Halevi S. The random oracle methodology, revisited. *Journal of the ACM (JACM).* 2004; 51(4): 557-594.
[11]. Koblitz N, Menezes AJ. The random oracle model: a twenty-year retrospective. *Design, Codes and Cryptography*. 2015; 77(2-3): 587-610.
[12]. Zhang J, Huo L, Liu X, Sui C, Li Z, Ma J. *Transferable optimal-size fair e-cash with optimal anonymity*. In Theoretical Aspects of Software Engineering (TASE), 2015 International Symposium on. 2015: 138-142.
[13]. Tan DN, Nam HN, Hieu MN, Van HN, Thi LT. New Blind Multi-signature Schemes based on ECDLP. *International Journal of Electrical & Computer Engineering (IJECE)*. 2018; 8(2).
[14]. Cai Z, Zhang Q, Li M, Gan Y, Zhang J. Multi-Domain Authentication Protocol Based on Dual-Signature. *TELKOMNIKA Telecommunication Computing Electronics and Control*. 2014; 13(1): 290-298.
[15]. Koppula S, Muthukuru J. Secure digital signature scheme based on elliptic curves for internet of things. *International Journal of Electrical and Computer Engineering (IJECE)*. 2016; 6(3): 1002-1010.
[16]. Zhang P, Jiang H, Zheng Z, Hu P, Xu Q. A new post-quantum blind signature from lattice assumptions. *IEEE Access*. 2018; 6: 27251-27258.
[17]. Dreier J, Kassem A, Lafourcade P. *Formal analysis of e-cash protocol. In e-Business and Telecommunications (ICETE)*, 2015 12th International Joint Conference on. IEEE 2015: 65-75.
[18]. Aggrawal D, Maurer U. Breaking RSA Generally is Equivalent to Factoring. *IEEE Transaction on Information Theory*. 2016; 62(11): 6251-6259.
[19]. Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*. 1978; 21(2): 120-126.
[20]. Canard S, Gouget A. *Anonymity in transferable e-cash*. In International Conference on Applied Cryptography and Network Security. 2008: 207-223.