IJICTR
International Journal of Information & Communication Technology Research

Volume 15- Number 3 – 2023 (10 -20)

Using Blockchain to Improve the Security Of The X3DH Key Exchange Protocol

Sayed Javad Mousavi
Department of Information Technology, Faculty of Industrial Engineering
Tarbiat Modares University
Tehran, Iran
sayedjavad.mousavi@modares.ac.ir

Kamal Chaharsooghi*
Department of Information Technology, Faculty of Industrial Engineering
Tarbiat Modares University
Tehran, Iran
skch@modares.ac.ir

Gholam Ali Montazer
Department of Information Technology, Faculty of Industrial Engineering
Tarbiat Modares University
Tehran, Iran
montazer@modares.ac.ir

Received: 5 May 2023 – Revised: 11 August 2023 - Accepted: 29 August 2023

Abstract—**First and most important step to making secure end-to-end encryption is key exchange. X3DH is one of the most used protocols to do that. It uses a trusted server to exchange keys. If the key exchange is secure then we have identification, authentication, integrity, non-repudiation, and confidentiality for messages. In X3DH, if the trusted server is compromised the entire end-to-end encrypted connection will be exposed. Transport Layer Security (TLS) is used for client-server communication. Therefore, the whole security is based on a certificate authority (CA) therefore there will be the single point of failure. In this paper, we proposed using blockchain as a trusted medium to exchange keys and identity authentication. The proposed method is based on the use of X3DH in instant messaging. This method improves the first step of the X3DH algorithm which includes authentication. This is the first time using blockchain directly to identify a user.**

Keywords: End-to-end encryption, X3DH, Asymmetric encryption, Blockchain, Identity key, Authentication

Article type: Research Article

© The Author(s).
Publisher: ICT Research Institute

I. INTRODUCTION

X3DH is widely used in instant messaging (IM) and nowadays IM applications became more popular. As an example, Whatsapp has 2 billion active users and 100 billion messages sent by it daily [1]. Furthermore, IM applications are used to send and receive patients' documents and information even in an emergency situation [2]. Therefore IM applications need to be more secure. Also, there are some reports on security leakage of IM applications like Phishing attacks on web-based versions [3], changing the name of the sender in a group chat to abuse famous names [4], and using mobile network vulnerabilities to access messages or even take control of an account [5]–[8]. Also, there may be other unreported flaws in IM applications and their security protocols that were used to create Pegasus spyware by NSO [9], [10]. Accordingly, our method is prepared to use in IM. But it can easily be modified for other purposes.

* Corresponding Author

[Downloaded from journal.itrc.ac.ir on 2024-12-22]

[DOI: 10.61186/itrc.15.3.11]

IM applications are using TLS to make a secure channel between clients and a server [11] then they build end-to-end encryption using X3DH [12] and Double Ratchet (DR) [13] algorithms. X3DH and DR are part of the Signal protocol. In theory, eavesdropping on a connection secured by this protocol is impossible for IM service providers, network operators, developers, and other eavesdroppers. But, if it is implemented flawlessly! An important feature of X3DH is to ensure that the current compromise of entities cannot impact the security of cryptographic primitives used in the past. It is because of mass storage capabilities and powerful infiltration technics that are used by adversaries. It is known as *forward secrecy* in the context of key exchange protocols. Also, the term *forward security* can be used as a generalization of forward secrecy in the key exchange context [14]. Since, key exchange protocol with forward secrecy automatically enables forward security of any desired security property, in this context we can use the terms *forward security* and *forward secrecy* interchangeably [14].

A. X3DH protocol

X3DH key agreement protocol was developed by Open Whisper Systems (OWS) [12]. Against Diffie–Hellman algorithm, this protocol doesn't need another party to be online. Because this protocol used a trusted third party to solve the problem. According to the X3DH protocol, each party must generate three types of key pairs and upload their public keys to a trusted server. These keys are as follows [15]:

1. Identity key (*IK*): long-lived public key generated using Curve25519 in the Diffie-Hellman key management algorithm. It generates after identification.
2. Signing Perkey (*SPK*): Middle-lived public key generated the same way as *IK*
3. Perkey signature (Sig(*IK*, Encode(*SPK*))): signature of *SPK* using IK's private key
4. A set of one-time Perkey (*OPK*): a set of public keys generated the same way as *IK*. But after one-time usage of each key, the key must be deleted to guarantee forward secrecy.

IK is generated at the installation of the IM application or it updates after a long period. *SPK* must be updated after a short period. As an example, consider client A wants to make an end-to-end encrypted connection to client B. Therefore, A must acquire three types of B's public keys from a trusted server. Then A can calculate the master key (*MK*) using B's three keys, A's *IK*, and A's ephemeral key (*EK*). Right after calculating *MK* A's ephemeral key and B's used *OPK* must be deleted to ensure Forward Secrecy. A's ephemeral key is equivalent to *OPK* but has not been sent to the server yet. Then A sends its *IK* to B and defines which *SPK* and *OPK* of B were used. Therefore B can calculate the same *MK* [16]. The equation (1) shows A's calculations.

$$\begin{aligned} DH1 &= DH\left(IK_A, SPK_B\right) \\ DH2 &= DH\left(EK_A, IK_B\right) \\ DH3 &= DH\left(EK_A, SPK_B\right) \\ DH4 &= DH\left(EK_A, OPK_B\right) \\ MK_{AB} &= \mathrm{KDF}\left(DH1 \| DH2 \| DH3 \| DH4\right) \end{aligned} \quad (1)$$

KDF function in the equation (1) is described in X3DH [12] protocol in detail. Therefore, by getting the pre-described three types of keys B doesn't need to be online in the first step. After calculating *MK* by A and B they can initiate the DR algorithm. This algorithm needs *SPK* of A and B and *MK* as the first input on A's side to be initiated [17]. DR needs a B's public key and a root key at first. B's SPK can be used as an input public key and MK can also be used as the first root key. Then DR protocol will generate these three keys.

1. Root key
2. Receive chain key
3. Send chain key

Also, there is a message key that calculates from the chain key. These three keys recalculate and change after each message send or receive. More details are in [17] and [13].

Speaking in terms of Protocol Messages, X3DH is a part of the Signal protocol proposed by Moxie Marlinspike and Trevor Perrin [12]. They provide an overview of X3DH which is defined over the TCP layer as an array of transferring bytes. Also, analyzing the protocol is out of the scope of this paper. But, there is an analysis of the Signal protocol in [18]. Therefore, we can describe X3DH protocol messages based on it [12]. This protocol has more details in terms of Protocol Messages. Here we present a simplified version of protocol message exchange. Again, assume that we have A, B, and a trusted Server. Also, A connects to the server and B also connects to the server over TCP. After they have established a TCP connection, there is a TLS handshake. Then, E.164 encoded telephone number verification is completed which is equal to the identity verification. Thereafter, the exchange of X3DH messages begins. In brief, the basic messaging of the protocol have shown in Fig. 1.

Most IM applications that use signal protocol, implement the above encryption mechanism. But, almost all of them have no blocking mechanism. Therefore, an attacker can use network vulnerabilities like a voice mail attack [5] to access a victim's account and read past messages. Also, there are other known vulnerabilities in the mobile network like redirect calls on SS7 using mimicking Home Location Register (HLR) operation [6]. Furthermore, there are no defined authentication mechanisms in the SS7 for a short message service center (SMSC) that forwards SMSs to recipients [6]. So, it can be used to eavesdrop on authentication SMS from the server.

[Downloaded from journal.itrc.ac.ir on 2024-12-22]

[DOI: 10.61186/itrc.15.3.11]

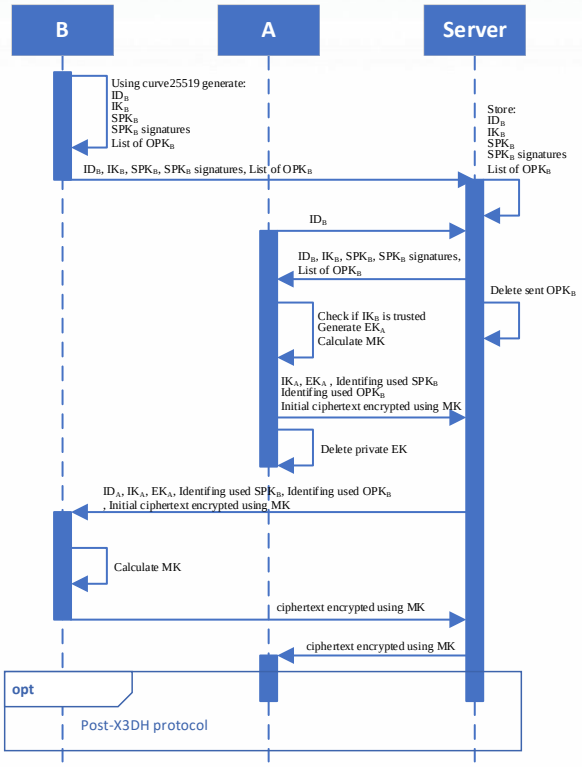

Figure 1. Basic messaging of X3DH protocol

B. *Literature review*

Recently, many researchers are working on how to use blockchain to solve problems that can't solve with traditional methods. In the information security area, researchers proposed some methods using blockchain to solve the problem of insouciance issuing certificates by CAs [19]–[21], using temper evident logging mechanism of blockchain to solve the problem of unreported security breaches [22], the same method for the issued backdated certificate to bypass new provisions [19], using distributed processing feature of blockchain along with its transparency for the single point of failure problem in the current PKI [23], [24], and transparency of blockchain can solve the problem of compelling CAs by government or other external power [25]. These methods are used to secure the internet of things (IoT) [26], increasing the security of wireless communications [27], and webservers security [28]. These researches are generally using the concept of asymmetric key cryptography. Also, there are some methods to secure the IM without blockchain like using a backup server to guarantee the integrity of messages [29], increasing the security of IM applications using Honey Encryption [30], protecting group chats using Vigenère encryption [31], and a mobile app for encrypting messages before send and decrypting them after it received [32]. There is research to improve the X3DH protocol using blockchain for IoT [33]. Their proposed method consists of a smart contract that simply stores *IK*, *SPK*, and *OPK* instead of a trusted server but there is no identification was introduced to tie the client's identity to *IK*. Furthermore, storing *SPK* and *OPK* on the blockchain can endanger Forward Secrecy. Because deleting a variable's value in a smart contract cannot remove its data from the blockchain. It can be extracted from previous blocks. Also, in another work, they proposed generating key pairs in the smart contract [34] which makes its security even worse. Because miner nodes can access all information inside asmart contract during its execution. Depending on which miner is most likly to be the winer of the block private key may expose to a malicious node. In this paper, we introduced a method for the identification and authentication of a client at the first step before the registration of *IK* in a distributed maner without the need for a central third party. Also, our method only stores *IK* which is the important key in X3DH protocol. *SPK* and *OPK* never stores on the blockchain to ensure Forward Secrecy.

II. SECURITY MODEL

Generally, the security of a connection with well-implemented end-to-end encryption consists of authentication, integrity, confidentiality, and non-repudiation. All of these features can be achieved at the first step when the trusted server ties one identity to a public key. In the traditional method, this identification step was possible by sending an SMS from a server to the new user or by a call from the server to the user. But in our proposed method trusted server is replaced with the smart contract therefore there is no human involved in the identification process inside the smart contract. Also, the smart contract is a distributed program over the blockchain so it removes the need for a single trusted point.

A. *Assumptions and notation*

In this research, we assume that sending and receiving of messages are asynchronous and that the blockchain network has enough users so the smart contract is always available. Each user has a unique identification that we denote with *Nbr*. It can be his/her mobile number. All important symbols are described in TABLE I. I.

TABLE I. DESCRIPTION OF ALL SYMBOLS

Variable	Description
Nbr	User unique ID or phone number
IK	Identity public key
IpK	Identity private key
SPK	Signed public Perkey
SpPK	Signed private perky
OPK	One-time public perky
EK	An ephemeral key similar to *OPK* but is used before sending to the server
SPKsig	Signed value of *SPK*
Adr	An Ethereum network Address
Crt	Smart contract
CrtAdr	Smart contract address (same format as *Adr*)
RndNbr	Randomly selected phone number
RndAdr	Randomly selected *Adr*
RndVal	Pseudo-Random value
Rgd	Registered in smart contract
C	Cost
SNbr	Source number of an SMS

All functions involved in the traditional registration and first authentication method are depicted in Fig. 2.

[Downloaded from journal.itrc.ac.ir on 2024-12-22]

[DOI: 10.61186/itrc.15.3.11]

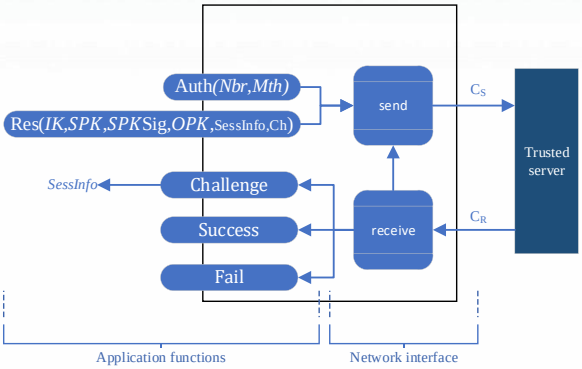

Figure 2. All functions involved in traditional authentication

According to Fig 2, at first, *Nbr* and authentication method must be sent to the server by function Auth. Then the session information *SessInfo* is received by function Challenge. At the same time, a code is sent to the user's phone number. In the next step, function Res will send all *IK*, *SPK*, sign of *SPK*, *OPK*, *SessInfo*, and the code received by phone number *ch* to the server. The result will receive by function Fail or Success and depending on the process failure or success the related operation will perform. C_R and C_S are encrypted information in the TLS layer.

B. *Threat model*

In this research we considered three types of attacks that can perform using:

Mobile network vulnerabilities: as we mentioned before some known flaws in the mobile network can be exploited to get the victim's one-time password (OTP) sent from the trusted server in the traditional method. Also, there may be other unknown vulnerabilities that can be used to access an OTP.

Compromised trusted server: if the server platform is contaminated by a virus it can be used by an attacker to access someone's account. Also, server administrators may be forced by the government or other authorities to hack an account. In this situation, the legitimate public key will be replaced by the attacker's public key at the new connection session and then the attacker can replace subsequent keys and re-encrypt the communication. Therefore, the entire communication will be exposed.

Network service provider vulnerabilities: since network traffic of the server can be detected by service providers. Therefore, if any device or network component in the infrastructure is compromised or hacked the traffic will be rerouted by an attacker. The traffic is encrypted through SSL/TLS but as we mentioned before there are many vulnerabilities in the PKI model that can be exploited to access the protected data and also change them. In this situation, a fake public key can be replaced for the victim's *IK* at the initiation of a session, then the subsequent key will be replaced and the attacker can re-encrypt the communication. Therefore, the entire communication can be manipulated and eavesdropped on.

C. *Security goals*

The main security goal of this research is to be sure of *IK*'s owner identity. Since availability is one of the security features. Therefore, the availability of the authentication process at any time and situation is also important. Furthermore, the proposed method must avoid the single point of failure problem by distributing its processes over the entire network.

III. BLOCKCHAIN

Blockchain is big data consisting of many small parts each called a Block. These blocks are chained together using cryptographic methods. Also, there is a network of nodes that agreed on a set of rules called Consensus Rules one of them is the Consensus algorithm an example of a consensus algorithm is Proof of Work (PoW). A node is a member of a blockchain until it obeys Consensus rules. The Consensus algorithm is necessary to add a new block at the end of the blockchain. The process of adding a new block is called mining and the miners can earn cryptocurrency for their work. Any new transaction must add to a block along with some other new transaction and the block must seal by a miner. After the completion of the mining process, all transactions inside the mined block will be validated and no one can change it. Each block has a header. The hash value of the previous block's header must be added to the current block header. Blocks are chained in this way. Each transaction data has information about the sender address, receiver address, and the transferred amount. Also, a transaction can contain other information like a note. Each node in the network can have many wallets and each wallet has a unique address in the entire network. [35]

Smart contract was developed based on blockchain. The smart contract is a way to process data in a distributed manner. Miner machines are responsible to execute a smart contract and store the results into the mined block and receive crypto instead. There are two types of processing data: 1) processing results to change the memory, and 2) processing doesn't need to change the memory. In the first type, the processing must be done by miners and the results must be stored in the mined block. But the second type of processing data can be done by each node that has the complete file of the blockchain. In the first type, the cost of processing data must pay using Gas. Gas is exchangeable with Ether but only during the call of a smart contract. Gas price varies and depends on the Ether price in the real market and the bargaining power of a miner. [36]

IV. IMPLEMENTATION

We used the following steps to evaluate the proposed method.

Building test environment: we made an Ethereum private network with several nodes. The consensus algorithm was proof of work (PoW). The network had two miners, arbitrary numbers of ordinary nodes, and a boot node. The smart contract was deployed over this network through one of the ordinary nodes.

Implementing the new authentication method: a smart contract was developed by solidity language. Also, a DApp was developed using the web3.js library to communicate with the smart contract and do the tests.

[Downloaded from journal.itrc.ac.ir on 2024-12-22]

[DOI: 10.61186/itrc.15.3.11]

Proof of concept: to realize the proposed method for improving authentication in the X3DH protocol, all functions needed for decentralized authentication were developed in a smart contract and its mandatory DApp. This DApp can be integrated with the application using the X3DH protocol and then connect to the smart contract. All DApp functions can be used along with other functions that were built into the application to use the X3DH protocol without changes.

V. THE PROPOSED PROTOCOL

First, we should describe the authentication process. In the traditional authentication process, the trusted server sends an SMS to a mobile number then the client application sends session information, a code from the received SMS, and a set of public keys. After evaluation by the server and successful verification, the server will distribute client public keys. But in our proposed protocol, when client authentication is needed for the first time the following steps are needed. We assume each mobile number is equivalent to a unique wallet address from the smart contract perspective.

1) First, a client application that has an address *Adr* of the Ethereum network sends its number *Nbr* and a registration request to the smart contract *Crt*. The client Ethereum address *Adr* is detectable from its request by the smart contract.

2) Smart contract randomly selects *h* registered phone number from the set of previously registered phone numbers $\{Nbr_i \mid \text{Status}(Nbr_i) = Rgd\}$. Then generates a random value $RndVal_i$ for each randomly selected number $RndNbr_i$. Each $RndNbr_i$ has previously registered the address $RndAdr_i$ on the Ethereum network.

3) The client application will receive the set of randomly selected numbers $\{RndNbr_i \mid 1 \le i \le h\}$ and the set of unique codes $\{RndVal_i \mid 1 \le i \le h\}$ that must send to that numbers. These two sets will receive in the form of two ordered arrays.

4) When the client sent its first request to the smart contract it must simultaneously spend an amount of Ether C_{reg} to let the smart contract do the process. The C_{reg} consists of smart contract processing cost c_g and an amount of Ether C_w to guarantee the whole process. Each $RndAdr_i$ is equivalent to a registered client that must spend an amount of Ether ct_i in this process. Accordingly $ct = \sum_{i=1}^{h} ct_i$ and $C_w = ct + ct \times r$. C_{reg} must be updated after each registration process. More about these costs are in the next steps.

5) After receiving sets of $RndVal_i$ and $RndNbr_i$ the client must send each $RndVal_i$ to its equivalent $RndNbr_i$.

6) Each client with registered $RndNbr_i$ after receiving its $RndVal_i$ by SMS has a client application. The client application will send the $RndVal_i$ and the number sent it $SNbr_i$ to the smart contract by its registered address $RndAdr_i$. If the user that requested registration was an honest user $SNbr_i$ would be the *Nbr*. Also, the costs related to sending these values to smart contract ct_i plus *r* percent as a motivation will pay back to each $RndAdr_i$.

7) The contract calculates the ct_i during the receiving values from $RndAdr_i$. Then the contract stores ct_i for $RndAdr_i$.

8) The smart contract can find $RndNbr_i$ the corresponding number of $RndAdr_i$ and has the $SNbr_i$ received from $RndAdr_i$. If the $RndVal_i$ of $RndNbr_i$ is correct and the $SNbr_i$ is equal to *Nbr* then the smart contract considers *Nbr* as a registered number by setting a boolean variable to 1. Also, the contract will publish its *IK*.

9) When the last $RndVal_i$ receives by contract and it's correct the contract will send $ct_i + ct_i \times r$ Ether to each $RndAdr_i$.

10) If $C_w \le (C_{reg} - c_g)$ the remaining Ether will send back to *Adr*. But in the situation that $C_w > (C_{reg} - c_g)$ the *Adr* must send the amount of deficit value before the contract can send $ct_i + ct_i \times r$ and set the *Nbr* as a registered number.

11) If in the time *t* the $RndAdr_i$ didn't send $RndVal_i$ the client *Adr* can request for replacing the $RndNbr_i$. This request can cost C_{rp} Ether for client *Adr* which $C_{rp} = c_g$.

The basic messaging of the new protocol according to the above steps is depicted in Fig. 3. It shows the registration process without considering some error cases like $SNbr_i \neq Nbr$, validating $RndVal_i$ corresponding $RndAdr_i$, $C_w \neq (C_{reg} - c_g)$, and responding time of $RndAdr_i$. It sheds light on the basic concept of the new protocol. In Fig. 3 the Ether is sent like a message which is true and even each arbitrary message must be inside a transaction. A transaction is mandatory for a message. Therefore, messages and transactions are together.

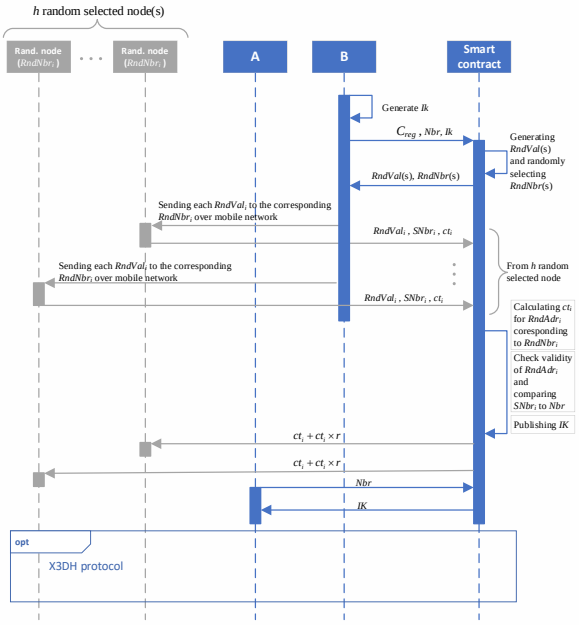

Figure 3. Basic messaging of the new protocol

[Downloaded from journal.itrc.ac.ir on 2024-12-22]

[DOI: 10.61186/itrc.15.3.11]

These are the overall steps but more details will be here. The most important variable is *h*. It defines the number of randomly selected addresses (equivalent to the number of randomly selected registered numbers). We call it the hardness of registration. By increasing the *h* the attacks will be defeated effectively. As an example, assume there are *A* registered addresses in the contract, and also somehow a malicious user could register *m* addresses. The malicious user wants to get control over a victim's phone number's *IK* using its *m* registered phone numbers or addresses. The probability of selecting one phone number from *m* malicious phone numbers can calculate by equation (2) .

$$p = \frac{m}{A} \tag{2}$$

Therefore, the possibility of randomly selecting all *h* phone numbers from the *m* malicious phone numbers can calculate by equation (3).

$$P = p^h \tag{3}$$

Now assume the malicious user wants to repeat the operation k_t times. So the possibility of randomly selecting *h* phone numbers from the *m* malicious phone numbers by k_t times repeating can calculate by equation (4).

$$P_k = \binom{k_t}{1} P(1-P)^{k_t-1} + \ldots + \binom{k_t}{1} P^{k_t-1}(1-P) + \binom{k_t}{k_t} P^{k_t} \tag{4}$$

The equation (4) can be abstracted by some statistical calculations. So, the overall success probability of a malicious user with k_t times repeating can calculate by equation (5).

$$P_k = 1 - \left(1 - \left(\frac{m}{A}\right)^h\right)^{k_t} \tag{5}$$

Therefore, for the success rate of P_k how much the malicious user must pay? If for each *i* we have $ct_i = c_c$ then registration cost can be calculated by equation (6) .

$$C_{reg} = h \times \left(c_c + \left(c_c \times r\right)\right) + c_g \tag{6}$$

Malicious users must spend C_T Ether according to equation (7) to achieve a success probability of P_k.

$$C_T = \left(k_t + m\right) \times C_{reg} \tag{7}$$

If Ether to the USD exchange rate is *e* and the overall cost in USD is *E*, then the *E* can be calculated by equation (8).

$$E = C_T \times e \tag{8}$$

Therefore the malicious user must pay E USD to achieve the success probability of P_k as the equation (9).

$$E = \left(k_t + m\right) \times \left(h \times \left(c_c + \left(c_c \times r\right)\right) + c_g\right) \times e \tag{9}$$

But the honest user must pay E_r as calculated in the equation (10).

$$E_r = \left(h \times \left(c_c + \left(c_c \times r\right)\right) + c_g\right) \times e \tag{10}$$

Now we should perceive it by numbers. If *h*=5 and the number of malicious nodes are *m*=50 also all registered phone number is *A*=10000. According to our evaluation at least c_c=0.002 in Ether and c_g=0.003. Also, we define a reasonable motivation as *r*=0.05. Ether to the USD exchange rate is 1,971 on 18 May 2022. If a malicious user's number of try be k_t=1000 times to get all *h* phone numbers from his/her *m* registered phone number, his/her chance of success will be P_k=0.000000003 which is a very small possibility. The malicious user must pay at least *E*=27,938.92 USD for all k_t=1000 times try and m=50 registered phone number. Instead, the cost of registration for an honest user will be E_r=26.60 USD. Therefore, the malicious user must even pay more for more tries to get more chances of success. According to this example, costs can defeat any attack on this protocol which is tunable by *h* and we evaluate it more here.

In one of the worst scenarios, the smart contract was publicly available after initial 5000 phone number registrations and instantly after that, an attacker was able to register other 5000 phone numbers equal to 50 percent of all registered phone numbers. So, *A*=10000 and *m*=5000 the attacker spends 159,651 USD for registration also he/she wants one time try to get all *h* randomly selected phone numbers from its *m* registered phone numbers. Now we evaluate the results by changing *h*. As depicted in Fig. 4 by increasing *h* the success probability of the attacker will decrease significantly even when he/she has 50 percent of all registered phone numbers. In the *h*=20 his/her success probability is 0.00000095 which is very small for having 50 percent of all registered phone numbers. Also, for high values of *h* registration costs will increase but it mostly affects the attacker. In this case for h=20 attacker must pay 1005411.04 USD to register 5000 phone numbers. It's just the amount he/she must pay in the blockchain but there are also other costs outside of the blockchain.

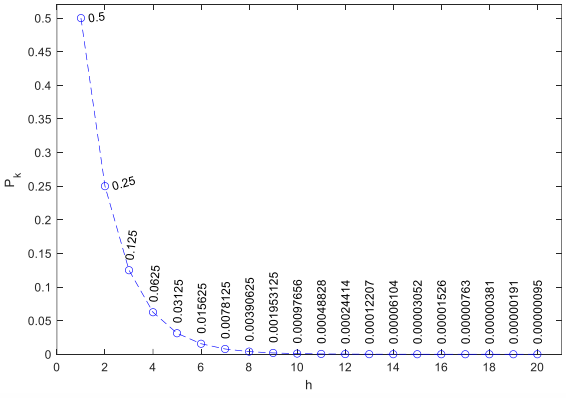

Figure 4. Attacker success probability by changing *h*

A. *Nodes collaboration*

There is a question: in regards to the verification cost ct_i for randomly selected phone numbers, how do

[Downloaded from journal.itrc.ac.ir on 2024-12-22]

[DOI: 10.61186/itrc.15.3.11]

motivate nodes to verify a new phone number? As mentioned in step 6 the smart contract must pay back ct_i and $ct_i \times r$ to the address that spends ct_i Ether for verification. The value $ct_i \times r$ must add to the payback to encourage randomly selected nodes to verify the new phone number. In the test implementation $r = 0.05$. Also, a maximum interval t must be defined for randomly selected nodes to verify the new node. Because the randomly selected node may be out of the grid or the node may be offline. In the test implementation $t = 10$ Minutes. After this interval new node can request to replace randomly selected nodes.

If h is 3 and two out of the three randomly selected phone numbers don't verify the new phone number, the amount of money spent by nodes that verified the new number will not pay back. Also, it is happening when $h \geq 4$ and less than 25% of randomly selected phone numbers don't verify the new phone number. Thereafter, they will be replaced by other randomly selected phone numbers. Not verifying a new number can happen rarely by mobile network errors. But, verifying a wrong number is a malicious activity. Therefore, it can be called punishment. Also, if less than 1% of consecutively cumulative randomly selected phone numbers don't verify a new number, it considers a verifying action and its money will pay back without motivation present. This can be caused by an error in the mobile network and defining an effective percentage is a part of our future study.

B. *The smart contract evaluation*

An important dilemma in this smart contract and any other smart contract is how to generate random values. Theoretically, in computer science generating a genuine random value in a finite time is impossible. Therefore, any so-called random values in programming languages are pseudo-random values. In our method, pseudo-random values are generated using a function depicted in the equation (12).

$$\text{rand}(a) = \text{UInt}\left(\text{k256}\left(bD \,||\, bT \,||\, Ad \,||\, c \,||\, \text{sha256}(i)\right)\right) \quad (11)$$

In (11) the function UInt() gives an unsigned integer. k256 is an abbreviation of keccak256 which is a function in Solidity language to generate a hash value from its input. The *bD* gets the hardness of the current block. Variable bT contains the timestamp of the current block. Also, the variable *Ad* contains the address of the requesting node. Variables i and c are counter to generate more random values. This function in the equation (11) tries to use unmanipulable sources outside the smart contract to generate random values.

Before deploying the smart contract the value of c_t is set by default. But after the first registration and after each new registration it must be updated. Therefore, after deploying the smart contract and before the publishing of its address, it must have at least one more registration after h registered numbers. The first h phone number doesn't need to be verified. Because the number of registered phone numbers is less than h. It must be done first after deployment and before publicly publishing. Also, according to our calculation before the public publishing of the smart contract address, it's better to register at least 10000 phone numbers. It makes it hard enough for an attacker to register a victim's phone number.

C. *Backward compatibility*

In the current X3DH protocol, all three types of public keys must generate by the client. Therefore the proposed method can be used without the need to modify the current trusted server and a small modification in the current client. On the client side after generating three types of public keys *IK*, *SPK*, and *OPK* first the client identity must be verified by the smart contract (i.e its processing is on the blockchain) and its *IK* being published by the smart contract, then the client can send those public keys to the trusted server. As depicted in Fig. 5 the function Auth_B (inside the bottom rectangular shape that can assume as a blockchain-related module) takes the phone number *Nbr* and public identity key *IK*. Then, its output will be *IK* if the process is successful. Then the old function Auth takes the *IK* and sends it to the server. Other client software after receiving *IK* and Nbr_O can compare them with the *IK* in the smart contract. The client sends Nbr_O using the function Check_B to the smart contract and gets the IK_O then compares *IK* and IK_O. If the process was successful the client would use all three keys to establish a secure end-to-end connection. As we mentioned before there is no interference between the two protocols logically. Therefore, these new functions are independent of the old functions and all of the new functions can be added as a module to the application. Since the *IK* is the important key in the X3DH protocol it is the only key publishing on the smart contract. Also, publishing other keys on the smart contract can endanger future security. Because after storing a value on a smart contract it can never be deleted from blockchain data even if it is deleted from smart contract variables it exists in the previous blocks. The W_b is data communicated by the smart contract through the Ethereum Wire Protocol (EWP) [37]. EWP is based on the RLPx protocol [38] which encrypts data over the transfer layer. The R_b is read data from the smart contract without changing its memory. If the client node is a full node and the blockchain file is up to date then the client node read this data from its local memory. But if the client node is a light node it needs to use EWP and RLPx to get the information from other full nodes.

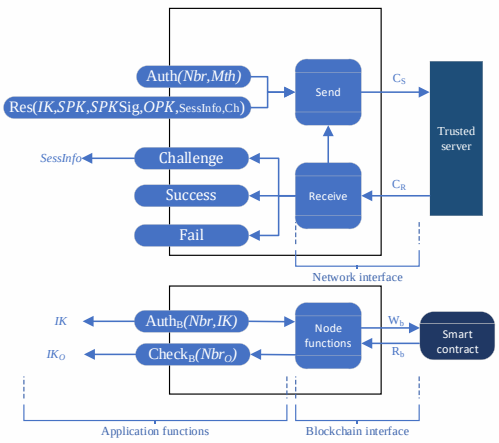

Figure 5. functions of the new authentication method in combination with the old method. The upper rectangular shape contains basic functions needed for traditional X3DH and the bottom rectangular shape contains basic functions for the new protocol.

[Downloaded from journal.itrc.ac.ir on 2024-12-22]

[DOI: 10.61186/itrc.15.3.11]

VI. Security analysis

Authentication: this paper's new method of authentication can significantly improve the process of authentication in the X3DH protocol. This method can defeat all eavesdropping attacks from the mobile network. Also, this method is using distributed processing and doesn't need a central server. If an eavesdropping attack happened on the mobile network it's unusable to have access control over a specific identity. Because each registered phone number in the smart contract has a unique Ethereum address which is also registered in the smart contract and it's not possible to send the codes from other addresses.

MitM attack: in this method, *IK*s are exchanged over the blockchain network with a robust authentication mechanism. In this way even if the trusted server is compromised it couldn't change the keys with fake keys to perform a MitM attack. Also, if an attacker hacks into the TLS layer using PKI vulnerabilities an attacker can't change the keys and perform a MitM attack. Because all *IK*s must check with their copy on the blockchain.

Denial of service (DoS) attack: in the traditional method there is a trusted server that is vulnerable to DoS attack. But the smart contract is a distributed processing mechanism, a DoS attack on it needs to take down all nodes on the blockchain network which is impossible to do in practice.

Single point of failure: in the traditional method trust in a server was based on SSL/TLS secure communication. In SSL/TLS protocol the authentication of the server is based on its certificate signature. The certificate was signed by a certificate authority (CA) and its root certificate is in the root store along with other root certificates. In case of a fake root certificate in the root store or a compromised CA [23], [24] all communication over SSL/TLS will be exposed. But in our method using the root store is not necessary.

Transparency problem: transparency in a blockchain network is an advantage, but sometimes it may make a security problem. In the proposed protocol *RndNbr*(s) and *RndVal*(s) can be read by anyone. So, it comes to mind that any of those possible attacks on the mobile network can be used here too. Here is a big difference. Those attacks on an unregistered recipient are successful because that recipient has no anchor between known entities. But, in our method, the recipient is registered before and has a valid address on the blockchain network. Therefore, he/she is the only one that can send to or receive from the smart contract by that address until the associated private key to that address remains secure. It's because smart contract only accepts *RndNbr*(s) from its associated address.

Owning certain phone numbers: as we mentioned in subsection B of section V, if we assume a situation with 10000 registration that an attacker has control over 5000 registered numbers by registering them or colluding with them. If h=1, then it is obvious that the probability of a successful attack is 50% for each try. But, even in this situation by setting h=20 the probability of a successful attack is 0.00000095. Also, the related cost only inside the blockchain is more than 1 million USD (according to the exchange rate of Ether to USD 1,971 on 18 May 2022). This amount of money can encourage any staff of a central authority to compromise in a traditional single point of trust, which gives the attacker a 100% probability of a successful attack. Like other protocols, using this protocol needs some tuning to be more secure. The value of h and the time of the public release of the smart contract address are two important parameters. Public release of the smart contract address must be after the registration of a certain amount of honest members in combination with the selected value of h, resulting in less than the 0.000001 (one over million) probability of a successful attack. When a member validates a new number with the wrong sender, he/she individually has to be honest due to the punishment mechanism. Malicious activity mostly can happen if a group of registered numbers is controlled by an attacker. Also, designing an internal method for the smart contract to decrease the value of h by increasing the number of registrations is super easy. Defining the value of h by voting after the public release of the smart contract address is our future work.

Other security features like integrity, confidentiality, and non-repudiation can be achieved by the X3DH and DR protocols if the authentication is secured.

VII. Performance

In this research, we evaluate the time of response from the smart contract concerning the number of miners. Response time is necessary to be low for user satisfaction. In our private test blockchain, there were 15 nodes. The network begins to work with 2 miners. Step by step number of miners was increased until there were 10 miner nodes and 5 full nodes. To find the response time for each network configuration, 30 registration requests were sent to the smart contract, and its average was recorded. There was no correlation between the number of miners and the response time. It depends on the average block time which is mostly around 0.22 minutes from the beginning of the main net [39]. This time is calculated from when a request is sent by a client until the smart contract's response is received by the client and not includes the time of sending and receiving SMSs which depends on many parameters outside the blockchain network. Therefore, it only includes the processing time of the smart contract.

VIII. Conclusion

The traditional X3DH protocol has a weakness in its authentication mechanism by using a server. We improve this protocol using a distributed authentication mechanism using blockchain. This new method has a tunable parameter h. By defining h it can resist all attacks which were described in the Security Analysis section. Also, the parameter h must define in a way that doesn't expand the registration time too much. Because it can affect user satisfaction with the application. In general, tuning of h is a tradeoff that must select between more security and user satisfaction. In brief, the advantages of this method are a strong authentication mechanism, resistance against MitM and DoS attacks, and no single point of failure.

[Downloaded from journal.itrc.ac.ir on 2024-12-22]

[DOI: 10.61186/itrc.15.3.11]

But, some questions may come to mind which we write as follows:

The proposed protocol can be similarly implemented by a trusted third party sitting in the place of a smart contract. Then, why we must use blockchain? The answer is simple: because by using blockchain there is no single or a cluster of servers so there is no single point of failure and authority control. The storage and part of the data processing are distributed over the entire network and the other part of data processing which is needed to change the storage is distributed over all miner nodes with a set of strict rules.

Registration requires paying in real money, isn't it a big turn-off for any social app that can fail eventually? We all know that "*there is no free lunch!*" and that's why nowadays many people become more cautious about sending sensitive information using social apps. Therefore, it is a tradeoff between the ways someone wants to pay off.

The registration process requires referral(s). This means that one cannot register on its own. Does it make a social network hard to enter? We can compare it with a VIP party and of course, entering a VIP party is a bit hard. Both of these need referrals, except for the proposed protocol referrals are selected randomly and no one has control over it. Also, it is part of the tradeoff between the ways someone wants to pay off, between having a secure connection e bit hard or establishing a connection easily but less secure. Therefore, more high value for *h* is having more possible secure connections and vice versa.

Is blockchain a trusted medium? One of the important feature of blockchain is being a tamper-proof log of timestamped data which means no one can change it. Also, due to its transparency feature, all data in the blockchain can be read by anyone. All data is accessible from all over the blockchain network because it is distributed over the entire network of full nodes. Therefore, it provides enough trust as a medium to use for identity-related data exchange. So, no one can change identity except its owner, all changes in identity keys are visible to anyone, there is no single place for identity data to be controlled, and identity data is accessible from all nodes of the blockchain network.

In future work, we will implement it on the large scale to get more practical analytical data. Also, we will adapt this method to other security protocols like SSL/TLS and IoT environments.

REFERENCES

[1] B. Dean, "WhatsApp 2022 User Statistics: How Many People Use WhatsApp?," Backlinko, Jan. 05, 2022. https://backlinko.com/whatsapp-users (accessed May 07, 2022).

[2] U. Gulacti and U. Lok, "Comparison of secure messaging application (WhatsApp) and standard telephone usage for consultations on Length of Stay in the ED," Applied clinical informatics, vol. 8, no. 03, pp. 742–753, 2017.

[3] D. Cuddeford, "WhatsApp: Mobile Phishing's Newest Attack Target," Dark Reading, Aug. 28, 2018. https://www.darkreading.com/endpoint/whatsapp-mobile-phishing-s-newest-attack-target (accessed May 07, 2022).

[4] D. Barda, R. Zaikin, and O. Vanunu, "FakesApp: A Vulnerability in WhatsApp," Check Point Research, Aug. 07, 2018. https://research.checkpoint.com/2018/fakesapp-a-vulnerability-in-whatsapp/ (accessed May 07, 2022).

[5] M. Vigo, "Compromising online accounts by cracking voicemail systems," ERSONAL HACKING PROJECTS, WRITEUPS AND TOOLS, Aug. 14, 2018. https://www.martinvigo.com/voicemailcracker/ (accessed May 12, 2022).

[6] K. Ullah, I. Rashid, H. Afzal, M. M. W. Iqbal, Y. A. Bangash, and H. Abbas, "Ss7 vulnerabilities—a survey and implementation of machine learning vs rule based filtering for detection of ss7 network attacks," IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 1337–1371, 2020.

[7] J. Botha, W. C. Vant, and L. Leenen, "A comparison of chat applications in terms of security and privacy," in Proc. 18th Eur. Conf. Cyber Warfare Secur., 2019, p. 55.

[8] R. Abu-Salma et al., "The security blanket of the chat world: An analytic evaluation and a user study of telegram," 2017.

[9] J. Cox, "NSO Group Pitched Phone Hacking Tech to American Police," Vice, May 12, 2020. https://www.vice.com/en/article/8899nz/nso-group-pitched-phone-hacking-tech-american-police (accessed Sep. 20, 2022).

[10] amnesty.org, "Forensic Methodology Report: How to catch NSO Group's Pegasus," Amnesty International, Jul. 18, 2021. https://www.amnesty.org/en/latest/research/2021/07/forensic-methodology-report-how-to-catch-nso-groups-pegasus/ (accessed Sep. 20, 2022).

[11] T. Perrin, "The Noise Protocol Framework," noiseprotocol, Protocol Revision 34, Jul. 2018. Accessed: May 09, 2022. [Online]. Available: http://noiseprotocol.org/noise.pdf

[12] M. Marlinspike and T. Perrin, "The X3DH Key Agreement Protocol," Signal, Protocol Revision 1, Nov. 2016. Accessed: May 10, 2022. [Online]. Available: https://signal.org/docs/specifications/x3dh/x3dh.pdf

[13] T. Perrin and M. Marlinspike, "The Double Ratchet Algorithm," Signal, Algorithm Revision 1, Nov. 2016. Accessed: May 10, 2022. [Online]. Available: https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf

[14] C. Boyd and K. Gellert, "A Modern View on Forward Security," The Computer Journal, vol. 64, no. 4, pp. 639–652, Apr. 2021, doi: 10.1093/comjnl/bxaa104.

[15] N. Rastogi and J. Hendler, "WhatsApp security and role of metadata in preserving privacy," arXiv Prepr. arXiv1701, vol. 6817, pp. 269–275, 2017.

[16] T. Carpay and P. Lontorfos, "WhatsApp End-to-End Encryption: Are Our Messages Private?," Retrieved, vol. 2, no. 05, p. 2020, 2019.

[17] M. Bolli and P. Kofmel, "WhatsApp End-to-End Encryption," Seminar, Bern University of Applied Sciences, Bern, Switzerland, 2017.

[18] D. Van Dam, "Analysing the signal protocol," URL: ru. nl/publish/pages/769526/z00b_2019_thesis_dion_van_dam_2019_eerder. pdf, 2019.

[19] Mozilla, "CA/WoSign Issues," MozillaWiki. https://wiki.mozilla.org/CA/WoSign_Issues (accessed Feb. 11, 2022).

[20] sslshopper, "SSL Certificate for Mozilla.com Issued Without Validation," sslshopper, Jan. 01, 2008. https://www.sslshopper.com/article-ssl-certificate-for-mozilla.com-issued-without-validation.html (accessed Feb. 11, 2022).

[21] microsoft, "Microsoft Security Bulletin MS01-017 - Critical," Microsoft Security Bulletins, Jun. 23, 2003. https://docs.microsoft.com/en-us/security-updates/securitybulletins/2001/ms01-017 (accessed Feb. 11, 2022).

[22] A. Arnbak and N. A. van Eijk, "Certificate Authority collapse: regulating systemic vulnerabilities in the HTTPS value chain," 2012.

[23] J. Braun and G. Rynkowski, "The Potential of an Individualized Set of Trusted CAs: Defending against CA

[Downloaded from journal.itrc.ac.ir on 2024-12-22]

[DOI: 10.61186/itrc.15.3.11]

Failures in the Web PKI," in 2013 International Conference on Social Computing, Sep. 2013, pp. 600–605. doi: 10.1109/SocialCom.2013.90.

[24] J. Braun, F. Volk, J. Buchmann, and M. Mühlhäuser, "Trust views for the web PKI," in European Public Key Infrastructure Workshop, 2013, pp. 134–151.

[25] C. Soghoian and S. Stamm, "Certified lies: Detecting and defeating government interception attacks against SSL (short paper)," in International Conference on Financial Cryptography and Data Security, 2011, pp. 250–259.

[26] M. Shen et al., "Blockchain-assisted secure device authentication for cross-domain industrial IoT," IEEE Journal on Selected Areas in Communications, vol. 38, no. 5, pp. 942–954, 2020.

[27] C. Li, Q. Wu, H. Li, and J. Liu, "Trustroam: A novel blockchain-based cross-domain authentication scheme for Wi-Fi access," in International Conference on Wireless Algorithms, Systems, and Applications, 2019, pp. 149–161.

[28] Z. Wang, J. Lin, Q. Cai, Q. Wang, D. Zha, and J. Jing, "Blockchain-based certificate transparency and revocation transparency," IEEE Transactions on Dependable and Secure Computing, 2020.

[29] R. Johari, S. Kalra, S. Dahiya, and K. Gupta, "S2NOW: Secure social network ontology using whatsApp," Security and Communication Networks, vol. 2021, 2021.

[30] E. O. Abiodun, A. Jantan, O. I. Abiodun, and H. Arshad, "Reinforcing the security of instant messaging systems using an enhanced honey encryption scheme: the case of WhatsApp," Wireless Personal Communications, vol. 112, no. 4, pp. 2533–2556, 2020.

[31] P. Rösler, C. Mainka, and J. Schwenk, "More is Less: On the End-to-End Security of Group Chats in Signal, WhatsApp, and Threema," in 2018 IEEE European Symposium on Security and Privacy (EuroS P), Apr. 2018, pp. 415–429. doi: 10.1109/EuroSP.2018.00036.

[32] H. Hamdani et al., "The Proposed Development of Prototype with Secret Messages Model in Whatsapp Chat.," International Journal of Electrical & Computer Engineering (2088-8708), vol. 8, no. 5, 2018.

[33] A. Ruggeri, A. Celesti, M. Fazio, A. Galletta, and M. Villari, "Bcb-x3dh: a blockchain based improved version of the extended triple diffie-hellman protocol," in 2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), 2020, pp. 73–78.

[34] A. Ruggeri, A. Galletta, A. Celesti, M. Fazio, and M. Villari, "An Innovative Blockchain Based Application of the Extended Triple Diffie-Hellman Protocol for IoT," in 2021 8th International Conference on Future Internet of Things and Cloud (FiCloud), Aug. 2021, pp. 278–284. doi: 10.1109/FiCloud49777.2021.00047.

[35] btcinformation, "Bitcoin Developer Guide," Bitcoin information. https://btcinformation.org/en/developer-guide#block-chain (accessed May 20, 2022).

[36] thereum.org, "Ethereum development documentation," ethereum.org, May 20, 2022. https://ethereum.org (accessed May 21, 2022).

[37] F. Lange, A. Toulme, and G. Ballet, "Ethereum Wire Protocol (ETH)." ethereum, 2015. Accessed: May 20, 2022. [Online]. Available: https://github.com/ethereum/devp2p/blob/6b0abc3d956a626c28dce1307ee9f546db17b6bd/caps/eth.md

[38] subtly et al., "The RLPx Transport Protocol." ethereum, 2015. Accessed: May 20, 2022. [Online]. Available: https://github.com/ethereum/devp2p/blob/6b0abc3d956a626c28dce1307ee9f546db17b6bd/rlpx.md

[39] bitinfocharts, "Ethereum Block Time Chart," BitInfoCharts, 2022. https://bitinfocharts.com/comparison/ethereum-confirmationtime.html (accessed Sep. 24, 2022).

Sayed Javad Mousavi received his M.Sc. degree in Information Technology Engineering from Tarbiat Modares University, Tehran, Iran and he is currently Ph.D. candidate in Information Technology Engineering at Tarbiat Modares University, Tehran, Iran. His research interests include Robotics Path Planning, Optimization, Information Security, Blockchain, Fuzzy System, Deep LSTM Network and Quantum Neural Network. He authored scientific articles in national and international peer-reviewed conference proceedings and journals.

Kamal Chaharsooghi is Full Professor of Industrial Engineering at the Faculty of Industrial & Systems Engineering, Tarbiat Modares University, Tehran, Iran. Professor Chaharsooghi was graduated from Southampton University and has obtained his Ph.D. degree from Hull University, England. Prof. Chaharsooghi's research interests include: Manufacturing Systems, Supply Chain Management, Information Systems, Systems Engineering; Strategic Management, International Marketing Strategy and Systems Theory. Professor Chaharsooghi's work has appeared in European Journal of Operational Research, International Journal of Advanced Manufacturing Technology, International Journal of Computers & Operations Research, International Journal of Information Technology & Decision Making, Journal of American Science, Computers and Industrial Engineering; Elsevier-Computers in Industry, International Journal of Mechatronics and Manufacturing Systems, International Journal of Production Economics, International Journal of Business Performance and Supply Chain Modelling, Scientia Iranica, Modares Journal of Engineering, Amirkabir Journal of Science and Technology, International Journal of Engineering Science, etc.

Gholam Ali Montazer received his B.Sc. degree in Electrical Engineering from the K.N. Toosi University of Technology, Tehran, Iran in 1993 and his M.Sc. and Ph.D. degrees in Electrical Engineering from the Tarbiat Modares University, Tehran, Iran in 1995 and 1999, respectively. Currently, he is a Full Professor of Information Technology at Tarbiat Modares University (TMU). His areas of research include, soft-computing approaches such as Artificial Neural Networks (ANN), Fuzzy Set Theory (FST), Rough Set Theory (RST), Evolutionary Computation (EC) and their applications in E-Learning and E-Commerce Systems and other cyber environment. The second area of his interests consists of Science, Technology and Innovation Policy Making including: Higher education planning, Research and Technology Policy and Technology Foresight.

[Downloaded from journal.itrc.ac.ir on 2024-12-22]

[DOI: 10.61186/itrc.15.3.11]