Preserving Privacy and Fairness for an Innovative E-Commerce Model: Penny M-lottery

Jung-San Lee, Ying-Chin Chen, Ya-Han Kang, Ren-Kai Yang

Department of Information Engineering and Computer Science, Feng Chia University, Taiwan
leejs@fcu.edu.tw, ycchen.blythe@gmail.com, kangkangannie@gmail.com, a9017100@gmail.com

Abstract

Lottery games are prevalent worldwide because they provide participants with the opportunity to win a big fortune. With the development of the Internet, the practicability of online lotteries has advanced, because players can purchase lottery tickets online rather than at physical stores. However, the price of participating in traditional e-lottery is high so that not most players can join the game. Therefore, a novel type of lottery called penny lottery has been introduced. The price of penny lottery participation is lower than that of a traditional lottery, but it still offers people the opportunity to be a millionaire; thus, leading to fascination for participants with budget deficits. In recent years, the technology of mobile devices has developed rapidly. Applications are increasingly available in mobile environments. Consequently, we aim to develop an efficient mobile penny lottery mechanism called penny M-lottery. This mechanism could realize the fairness and offer privacy to participants. In addition, the process accuracy is verified through BAN logic and the security is confirmed by AVISPA. Experimental results revealed that the mechanism can be implemented in mobile environments because of the low computational costs and communication overheads.

Keywords: e-commerce, Penny M-lottery, Mobile, Fairness, Privacy, BAN logic

1 Introduction

Lottery is a prevalent form of entertainment worldwide. Without loss of generality, lottery games are initiated by governments, and the profits go toward charitable donations. Furthermore, participants have the chance to make a big fortune. The probability of each player winning is equal, and predicting winners is impossible. Since many people hope to win vast sums of money, lotteries attract thousands of participants. Lotteries involve players, lottery agents, and a lottery center. According to the game rules, each participant purchases a ticket with a set of numbers within a predefined range. Lottery agents sell tickets to the public, while the lottery center generates the winning numbers and announces the result after the deadline for purchasing tickets has passed. Most crucially, fairness of the game must be upheld. Because of the birth of e-commerce in 2000, people can now shop and access various services at any places via the Internet. E-commerce has generated multiple business models, and merchants are increasingly engaging in e-commerce because of its convenience and practicality. Hence, the players are able to purchase lottery tickets anywhere and anytime. In other words, players need not visit a lottery merchant to purchase tickets. However, the challenge of e-lottery is achieving fairness and user anonymity when players purchase tickets and claim prizes.

Sako [1] presented a scheme for offering random lottery outcomes to enable the players to trust the validity and fairness of the winning results. In 2001, Zhou and Tan [2] advanced the mechanism to ensure user anonymity by protecting the personal information of users and protecting winners against theft. Goldschlag Stubblebine [3] proposed publicly verifiable lotteries, where players can examine the legality of tickets by using a verification function to check whether they have joined the game. As fairness is a crucial requirement in lottery game, each ticket shall equally contribute to the winning set. If people trust that the lottery is fair, more players are attracted to participate [4-5]. In 2009, Lee and Chang [6] suggested an innovative *t*-out-of-*n* lottery scheme and defined the applications of e-lottery in more detail. The scheme enables participants to select *t* out of *n* numbers when purchasing tickets. Specifically, participants are capable of purchasing *t* numbers once, as opposed to having to purchase *t* tickets to obtain *t* numbers as in previous schemes. The *t*-out-of-*n* scheme closely resembles actual lottery rules.

In 2016, Chen et al. [7] have proposed a novel mechanism for purchasing e-lottery tickets by using mobile devices; in this mechanism, participants incur only lightweight computational costs. Thus, this mechanism can be implemented in smartphones. In particualr, they have introduce the concept of joint purchase, in which a player could invite friends to

*Corresponding Author: Jung-San Lee; E-mail: leejs@fcu.edu.tw
DOI: 10.3966/160792642019092005007

share the cost of one single ticket. Even traditional lotteries have become more popular around the world, high ticket costs render participation difficult for some people.

So far, aonther novel type of lottery which is the so-called "penny lottery" has not been realized in electronic commerce. A penny lottery ticket costs only a few cents, which is lower than that the cost of a traditional lottery ticket. Such low costs can attract participants with budget deficits. In addition, a penny lottery provides people with the opportunity to win a substantial amount of money with little expense. The characteristics of a penny lottery are as follows: First, a lottery server (LS) launches a penny pot in which all players put their tickets. Second, one winning ticket is selected from all those purchased. Each round has a winner, and the jackpot is not split. Therefore, the players know that the money they dedicate goes toward a prize that must be won by one of them. By contrast, in traditional lotteries, players may contribute large sums of money without return or may receive a prize that is divided among several winners, which could result in players ceasing participation.

Inspired by the mechanism of penny lottery, we aim to design a mobile version (hereafter denoted as penny M-lottery) in the present study. The correctness of the mutual authentication between participants and the LS is verified by the BAN logic which is a mathematical model for checking the correctness of protocol. Furthermore, the security of proposed method is verified through the formal tool, automated validation of Internet security protocols and applications (AVISPA). In the experiments, our scheme is proven to be capable of realizing low computational costs and communication overheads. Thus, the scheme can be executed efficiently on smartphones. As the penny M-lottery provides convenience and low ticket costs, players can join the game whenever and wherever by using their smartphones; thus, leading to that players can win the jackpot by spending small amounts of money.

The rest of this thesis is organized as follows. In Section 2, we define the essential requirements of the new mechanism, followed by the description of our proposed method. In Section 4, we present discussions regarding analyses of requirements and security, while the performance simulations are illustrated in Section 5. Finally, we make conclusions in Section 6.

2 Preliminary

Here we introduce the scenario of how to proceed the penny lottery game and define the essential of penny M-lottery in subsection 2.1 and 2.2, respectively.

2.1 Scenario

The penny M-lottery scenario is shown in Figure 1. There are three participants in our scheme, including players, lottery server (LS), and lottery agent. Players can purchase tickets from the LS, serving as the online lottery merchant. The LS must generate the winning ticket after the deadline. Moreover, the lottery agent is the government, which is the lottery organizer. First, each player who wishes to participate must register with the LS. Subsequently, players are able to purchase lottery tickets with the tokens and choose the game period and quantity of tickets. The LS then selects a winner from all of participates at the deadline with a fair and public way. The winner can claim the prize via providing proof the ticket receipt.

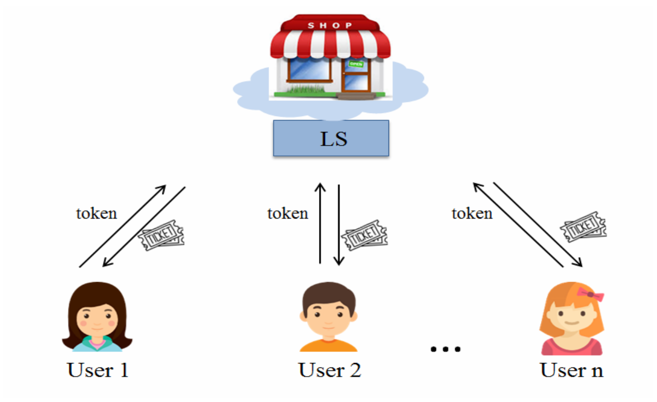


Figure 1. Scenario

2.2 Requirements

In the proposed scheme, Alice is a player who joins the mobile penny raffle, LS is a lottery server, and Eve is an attacker. Our scheme must confirm the requirements so that it could be applied in practice [4].

Fairness. The probability of the winning ticket in the penny M-lottery must be the same. It can attract more players to join the raffle if the game is fairness.

Unforgeability. An attacker Eve cannot forge the winning ticket and claim the prize successfully. This shall be guaranteed to protect the right of the winner.

Anonymity. The personal identity of ticket owner shall not be revealed from the ticket. This can protect the winner from being robbed.

Untraceability. If Eve intercepts a message, she cannot retrieve the information of the transmitter. The element can protect the privacy of the winner, including the period of penny lottery and the quantity of tickets.

Unlinkability. Even Eve is able to intercept messages in different sessions; she cannot link messages to know the relationship between tickets and players. It can avoid Eve from tracking the relevance of the messages. The element is to elevate the security of players.

Public verification. Alice can verify whether her ticket is included in this play or not via inputting ticket parameters through a confessed polynomial. Security: The proposed scheme shall be able to resist malicious attacks, including smart card stolen attack, perfect forward secrecy, impersonation attack, server spoofing attack, replay attack, and man-in-the-middle attack.

Mutual authentication. Alice and LS can verify each other. Alice verifies the legality of LS so that she can avoid linking a phishing server, while LS confirms the validity of Alice to ensure she is a legal user.

Lightweight. The computational cost and communicational cost in the authentication phase shall be light to perform the game on mobile devices and elevate the interest of players.

Convenience. Alice can join the raffle as long as she can connect to the Internet with the mobile devices. It can promote the sales of ticket.

3 Proposed Scheme

In this chapter, we describe the implementation of the penny M-lottery on mobile devices. The proposed scheme consists of the following four phases: registration phase, login phase, ticket purchasing phase, and claim the prize phase. The notations are defined in Table 1.

Table 1. Notations

Notation	Definition
Alice	The user
LS	The lottery server
ID/*PW*	Identification/Password of user
SID	Identification of LS
x	The master key of server
$a_1 / a_2 / r_1 / r_2 / y$	Random numbers
$T_1 / T_2 / T_3 / T_4$	Timestamp
$h(.)$	One-way hash function
$E_k / D_k(.)$	Encryption/Decryption using key k
$\oplus$	XOR operation
$\|\|$	Concatenation operation
$\vartriangle T$	Valid time delay
token_num	Token number
num	Ticket number
information	The serial number of lottery

3.1 Registration Phase

If Alice wishes to participate in the raffle, she must register with the LS. The registration flowchart is shown in Figure 2.

Step 1. Alice selects a pair of (*ID*, *PW*). Then she computes *NPW* and sends *ID* to the LS through a secure channel. *NPW* is a security value for smart card to verify the player.

$$NPW = h(ID \oplus PW)$$

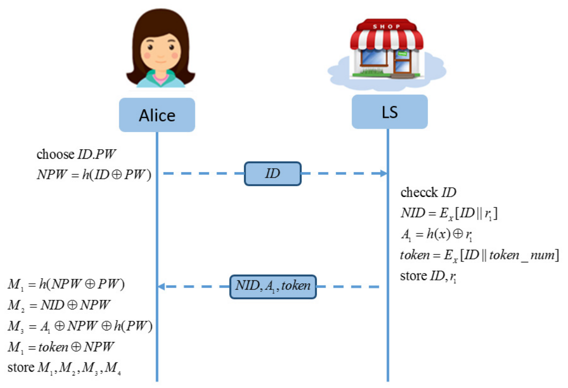


Figure 2. Registration phase

Step 2. The LS checks the validity of *ID*. If it is valid, it chooses r_1 and computes *NID*, A_1, and *token*. Then LS stores *ID* and r_1 in the database. Finally, LS sends {*NID*, A_1, *token*} to Alice through a secure channel.

$$NID = E_x[ID \| r_1]$$
$$A_1 = h(x) \oplus r_1$$
$$token = E_x[ID \| token\_num]$$

Step 3. Upon receiving the response, Alice computes and writes M_1, M_2, M_3, and M_4 into a smart card.

$$M_1 = h(NPW \oplus PW)$$
$$M_2 = NID \oplus NPW$$
$$M_3 = A_1 \oplus NPW \oplus h(PW)$$
$$M_4 = token \oplus NPW$$

3.2 Login Phase

After the registration phase, Alice can login the LS. Besides, Alice and the LS would compute the session key used to buy the raffle tickets. The login flowchart is illustrated in Figure 3.

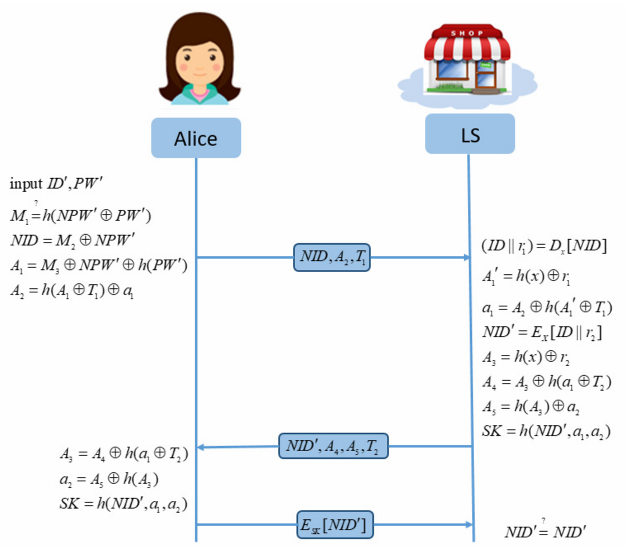

Alice
LS
input ID', PW'
$M_1 \overset{?}{=} h(NPW' \oplus PW')$
$NID = M_2 \oplus NPW'$
$A_1 = M_3 \oplus NPW' \oplus h(PW')$
$A_2 = h(A_1 \oplus T_1) \oplus a_1$
NID, A_2, T_1
$(ID \| r_1) = D_x[NID]$
$A_1' = h(x) \oplus r_1$
$a_1 = A_2 \oplus h(A_1' \oplus T_1)$
$NID' = E_x[ID \| r_2]$
$A_3 = h(x) \oplus r_2$
$A_4 = A_3 \oplus h(a_1 \oplus T_2)$
$A_5 = h(A_3) \oplus a_2$
$SK = h(NID', a_1, a_2)$
NID', A_4, A_5, T_2
$A_3 = A_4 \oplus h(a_1 \oplus T_2)$
$a_2 = A_5 \oplus h(A_3)$
$SK = h(NID', a_1, a_2)$
$E_{SK}[NID']$
$NID' \overset{?}{=} NID'$

Figure 3. Login phase

Step 1. Alice inputs *ID*', *PW*' into smartphone to compute and check whether $h(NPW' \oplus PW')? = M_1$ holds or not. If it holds, Alice can retrieve *NID* and A_1 as follows. Moreover, she chooses a_1 to calculate A_2 and sends $\{NID, A_2, T_1\}$ to LS.

$$NID = M_2 \oplus NPW'$$
$$A_1 = M_3 \oplus NPW' \oplus h(PW')$$
$$A_2 = h(A_1 \oplus T_1) \oplus a_1$$

Step 2. Subsequently, LS checks $T_1' - T_1 \le \triangle T$ and decrypts *NID* to verify the legality of *ID*. If the verification is successful, LS can obtain a_1 and select a random number r_2 to update *NID* and A_1 to *NID'*, A_3. Furthermore, it generates A_4 and selects a_2 to calculate A_5 and *SK*. Finally, LS renews r_2 in the database and delivers $\{NID, A_4, A_5\}$ to Alice.

$$A_1 = h(x) \oplus r_1$$
$$a_1 = A_2 \oplus h(A_1) \oplus T_1$$
$$NID = E_x[ID \| r_2]$$
$$A_3 = h(x) \oplus r_2$$
$$A_4 = A_3 \oplus a_1$$
$$A_5 = h(A_3) \oplus a_2$$
$$SK = h(NID, a_1, a_2)$$

Step 3. When Alice receives the reaction, she would check the LS for legality. If it holds, Alice can extract a_2 to calculate *SK*. Finally, Alice transfers the fresh identity *NID'* encrypted by the session key to LS.

$$SK = h(NID, a_1, a_2)$$
$$E_{SK}[NID']$$

Step 4. The LS decrypts the message to confirm the justifiability of *NID'*.

3.3 Ticket Purchasing Phase

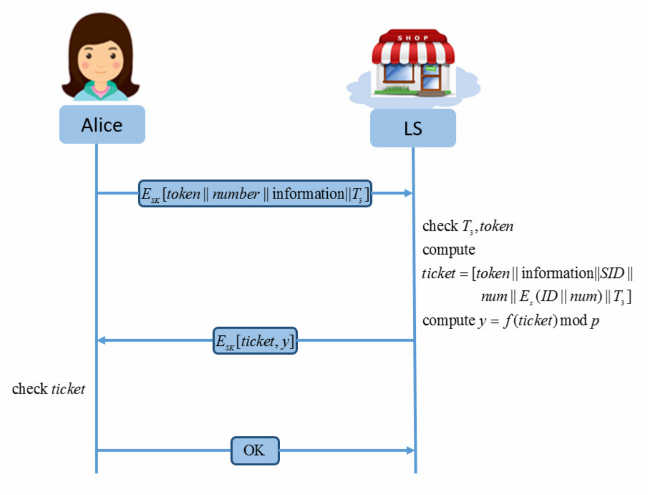
After the login phase, Alice can purchase tickets from LS. The flowchart of the ticket purchasing process is presented in Figure 4.

Step 1. Alice selects the desired ticket quantity along with the lottery period and sends $\{E_{SK}[token \| quantity \| \text{information}]\}$ to the LS.

Step 2. Upon receiving the message, LS examines the validity of *token* and computes the *ticket*. Then, it computes the function $y = f(ticket) \bmod p$ and transmits $\{E_{SK}[ticket, y]\}$ to Alice.

$$ticket = [token \| \text{information} \| SID \| num \| E_x(ID \| num) \| T_3)]$$

Step 3. Alice calculates $y = f(ticket) \bmod p$ to check for ticket validity and confirm her participation.

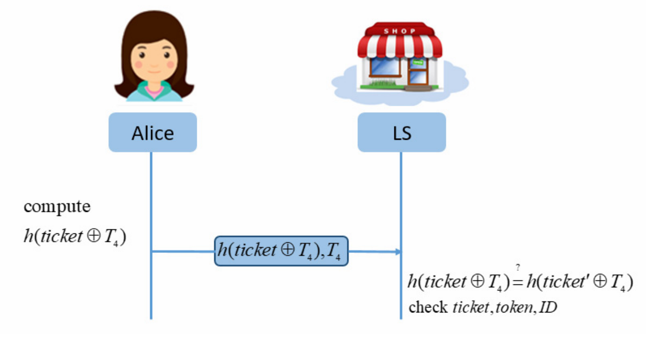


Alice
LS
$E_{SK}[token \| number \| \text{information} \| T_3]$
check T_3, *token*
compute
$ticket = [token \| \text{information} \| SID \| num \| E_x(ID \| num) \| T_3]$
compute $y = f(ticket) \bmod p$
$E_{SK}[ticket, y]$
check *ticket*
OK

Figure 4. Ticket purchasing phase

3.4 Claim Prize Phase

Suppose that Alice has won the play, she must claim her prize by presenting the winning ticket. The flowchart of the prize claiming process is shown in Figure 5.



Alice
LS
compute
$h(ticket \oplus T_4)$
$h(ticket \oplus T_4), T_4$
$h(ticket \oplus T_4) \overset{?}{=} h(ticket' \oplus T_4)$
check *ticket*, *token*, *ID*

Figure 5. Claim prize phase

Step 1. If Alice's *num* of ticket matches the winning numbers, she computes $h(ticket \oplus T_4)$ and sends $\{h(ticket \oplus T_4), T_4\}$ to the LS.

Step 2. Upon receiving the message, LS computes $h(ticket' \oplus T_4)$ and checks whether $h(ticket' \oplus T_4)? = h(ticket \oplus T_4)$ and verifies the validity of *token*, *ID*, *num*.

4 Security Analysis

In this section, we first define the assumption of cryptographic technology and analyze security in relation to the requirements outlined in subsection 4.2. Continuously, we explain how the mechanism resists the various attacks, as described in subsection 4.3. The BAN logic security verification is shown in subsection 4.4. Finally, examination under the AVISPA is displayed in subsection 4.5.

4.1 Assumption

The security of mechanism is based on

cryptographic assumptions: AES cryptosystem, one-way hash function, and exclusive-or operation [8-9].

4.1.1 Advanced Encryption Standard (AES)

The plaintext *M* is encrypted by the key *k* with the size 128 bits to obtain the ciphertext $C = [M]_k$. It is computationally infeasible to derive the *M* without the *k*.

4.1.2 One-way Hash Function

SHA-256 takes an input *x* to obtain the result $y = h(x)$ with the size of 256 bits. The security definition is described as follows [8].

(1) Preimage resistance: Given the output *y*, it is computationally infeasible to get the plaintext *x*.

(2) The second preimage resistance: It is computationally infeasible to find out *x* and *x'* such that $x \neq x'$, but $h(x) = h(x')$.

(3) Collision resistance: Given *x* and *h*(*x*), it is computationally infeasible to find the other plaintext *x'* and satisfy $h(x) = h(x')$.

4.1.3 Exclusive-or Operation

The exclusive-or operation $\oplus$ cannot be compromised in the polynomial time. Given the message m_1 and m_2, it is easy to compute $C = m_1 \oplus m_2$, but it is computationally infeasible to retrieve the message m_1 from *C* without knowing the message m_2.

4.2 Requirement Analysis

Here, we analyze how the penny M-lottery scheme can guarantee the following requirements: fairness, unforgeability, anonymity, untraceability, unlinkability, public verification, security, mutual authentication, lightweight, and convenience.

4.2.1 Fairness

After the ticket purchasing deadline has passed, LS generates a winner from the polynomial $f(x)$. It computes the output of function $f(x)$ by utilizing all purchased tickets $ticket = [token \,||\, \text{information} \,||\, SID \,||\, num \,||\, E_x(ID \,||\, num) \,||\, T_3)]$. This has guaranteed that all the tickets are included to contribute to the results. Thus, no one is able to interfere in yielding a winning ticket, so that the requirement of fairness can be confirmed in the play.

4.2.2 Unforgeability

If Eve wants to replicate the winning ticket, she must fail. In the ticket purchasing phase, Eve attempts to copy the winning ticket by adjusting *ID* and *num*. However, *ID* and *num* are encrypted into $ticket = [token \,||\, \text{information} \,||\, SID \,||\, num \,||\, E_x(ID \,||\, num) \,||\, T_3)]$. Eve has to decrypt $E_x(ID \,||\, num)$ by using *x* to gain *ID* and *num*; however, *x* is secret in the server so that Eve cannot obtain the master key under the assumption of AES encryption. Thus, she will fail to copy the *ticket*.

4.2.3 Anonymity

When the malicious attacker Eve intercepts the message $NID = E_x[ID \,||\, r_1]$, she cannot learn the real identity of Alice who transmitted the message. It is due to that *ID* is encrypted by the master key *x*, and the master key is only known to LS. Based on AES assumption, the attacker cannot obtain the real identity of Alice from message *NID* without the secret information. Therefore, the requirement of anonymity is achieved.

4.2.4 Untraceability

When Eve intercepts a message, she cannot learn the information of transmitter according to the interception. In the login phase, Eve obtains the information of Alice by retrieving the parameters *NID* and A_2. Note that *NID* is encrypted by the master key *x*. Eve cannot access the master key since it is confidential in the LS. Based on AES assumption, Eve is unable to gain the intelligence of *NID*. In addition, A_2 is protected by the hash function. Replying on the definition of hash function, Eve is unable to retrieve the information A_1 from the A_2. Eve cannot acquire the confidential *ID* of *NID* and A_2; consequently, she cannot know the information of Alice. Accordingly, the requirement of untraceability could be confirmed in the game.

4.2.5 Unlinkability

If Eve continually intercepts the login messages in each session, she cannot recognize whether these messages are from the same transmitter or not. In other words, even if Eve snatches every login message $\{NID, A_2, T_1\}$ in each round, she is incapable of knowing who sent the requests. Moreover, Eve cannot learn the identity of the real server with which Alice wants to connect. Eve receives the message NID, A_2, T_1 in a different session. However, the message $NID = E_x[ID \,||\, r_1]$ would be changed in each session owing to it is composed by random number r_1. Furthermore, r_1 is encrypted by the master key *x*, which is only known to the LS. Based on AES, Eve cannot decrypt *NID* without the master key. In other words, Eve has no way to surmise the proper r_1 to calculate *NID*. Therefore, Eve cannot find out the association of each message through *NID*.

4.2.6 Public Verification

After the purchase process, Alice can check the legality of her ticket. Alice would gain the $token = E_x[ID \| token\_num]$ in the registration phase. No one can obtain the *token* from the message $token = M_4 \oplus NPW$ based on the assumption of exclusive-or operation. After the ticket purchasing phase, Alice's *token* is encrypted and added to her ticket. Therefore, Alice retrieves the $token = M_4 \oplus NPW$ in the smart card by utilizing *NPW*, which is known only to Alice. Subsequently, Alice is able to compare the *token* stored in the smart card and encrypted in her ticket. If it holds, the validity of Alice's ticket can be confirmed. Moreover, Alice can check whether she has participated in the raffle or not. LS provides the polynomial $f(x)$ by utilizing the *ticket* of players. The polynomial $f(x)$ is subsequently used for public verification. That is to say, the polynomial would be changed according to who purchases tickets. When Alice receives the message $\{E_{SK}[ticket, y]\}$ from the LS, she can obtain $(ticket, y)$ by using the session key *SK*. Alice regards $(ticket, y)$ as a coordinate. Thus, Alice calculates whether $f(ticket) = y$ is valid. If the equation holds, Alice knows that she has successfully participated.

4.2.7 Security

Our proposed scheme can resist various attacks, such as stolen smart card attack, perfect forward secrecy, impersonation attack, server spoofing attack, replay attack, and man-in-the-middle attack. The details of how to resist such attacks are analyzed in subsection 4.3.

4.2.8 Mutual Authentication

In the proposed scheme, Alice and LS can verify each other's legality. Alice sends the login request $\{NID, A_2, T_1\}$ to LS. The message contains *NID* and A_2. The parameter $NID = E_x[ID \| r_1]$ is encrypted by the master key, which is known only to LS. When the LS obtains the parameter *NID*, it can learn the real identity of Alice by using the master key *x*. Thus, the LS can confirm whether Alice is an authorized user. If the verification succeeds, the LS can believe in that Alice is a valid user. When Alice receives the response from LS, she verifies the legality of LS by utilizing the random number a_1. Assume that Alice successfully retrieves A_3 from $A_4 = A_3 \oplus h(a_1 \oplus T_2)$, she can confirm that LS is a legality server. Due to the assumption of exclusive-or operation, LS is able to compute the proper A_4 by the random numbe a_1. In addition, a_1 is calculated as $a_1 = A_2 \oplus h(A_1^{'} \oplus T_1)$. Based on the characteristic of hash function, LS can obtain a_1 by using the correct A_1'. Furthermore, LS generates the proper a_1 by utilizing the master key *x*. In other words, Alice can check the validity of LS by using the random number a_1, because only the LS can gain a_1 with the master key *x*. Moreover, LS and Alice have to generate the session key together. LS chooses the random number a_2, which is protected by the parameter A_3. Based on the exclusive-or operation, Alice owns the correct a_1 to compute A_3 from the message $A_3 = A_4 \oplus h(a_1 \oplus T_2)$. Alice continually obtains a_2 by the proper A_3 from the message $a_2 = A_5 \oplus h(A_3)$ under the assumption of exclusive-or operation. Thus, Alice and LS can extract the random number selected by the other party to generate the session key *SK*. Thus, the player and LS can agree to mutual authentication and negotiate a common Session key. The formal security verification based on BAN logic model is shown in subsection 4.4.

4.2.9 Lightweight

The experiments of computational cost and communicational cost demonstrate that the proposed scheme has high efficiency, which are shown in Section 5.

4.2.10 Convenience

A person can join the lottery through the Internet connection and by using an uncomplicated mobile device. That is to say, Alice can participate from any place where she can use a smartphone and connect to the Internet.

4.3 Security Analysis

In this section, we analyze how to resist various potential attacks, including smart card stolen attack, perfect forward secrecy, impersonation attack, server spoofing attack, and replay attack.

4.3.1 Smart Card Stolen Attack

Suppose that the attacker Eve steals Alice's smart card with the intention of impersonating Alice, she will fail. It is due to that $M_1 = h(NPW \oplus PW)$ is the hash output of *PW*. Based on the characteristic of hash function, Eve is unable to retrieve the secret *PW*. Besides, Eve cannot generate *NID* from the parameter $M_2 = NID \oplus NPW$ without *PW* and *ID* to compute *NPW* because of the assumption of exclusive-or operation. A_1 is continually retrieved from

$M_3 = A_1 \oplus NPW \oplus h(PW)$. According to the exclusive-or operation, the authorized parameter A_1 cannot be obtained by Eve without the *NPW* and *PW*. Therefore, Eve is unable to retrieve *token* from the message $M_4 = token \oplus NPW$ under the assumption of exclusive-or. In the proposed scheme, all the parameters stored in the smart card are ciphertext; thus, the malicious attacker cannot dig the desired secure information.

4.3.2 Perfect Forward Secrecy

If a malicious attacker Eve compromises the session key $SK = h(NID', a_1, a_2)$, she is incapable of computing the previous and future keys. Because each session key contains two random numbers a_1, a_2 and different NID', all the session keys are distinct. Each NID' is encrypted by the LS, which utilizes the master key, and contains the random number r_2. The random number r_2 is distinct and protected in the parameter $NID' = E_x[ID \| r_2]$. According to the AES assumption, it is computationally infeasible for Eve to resume the previous session key and compute the future key using the session key *SK*. As a result, our scheme can achieve the perfect forward secrecy.

4.3.3 Impersonation Attack

Assume that the adversary Eve intends to simulate a legal user Alice to log in to the LS, she must imitate the login message $\{NID, A_2, T_1\}$. Nevertheless, Eve has no way to acquire the correct *NPW*, because she cannot forge $NID = M_2 \oplus NPW$ under the assumption of exclusive-or operation. When the LS receives the request, it decrypts the forged message NID', thereby detecting that Eve is a malicious attacker based on her inability to acquire the meaning message. Hence, the proposed scheme can resist impersonation attack.

4.3.4 Server Spoofing Attack

If Eve attempts to imitate the LS, she has to send the correct response message $\{NID', A_4, A_5\}$ to Alice. However, Eve fails to forge the correct message since she is unable to generate $A_1' = h(x) \oplus r_1$ without the master key *x* under the assumption of hash function. Furthermore, Eve cannot retrieve a_1 from $a_1 = A_2 \oplus h(A_1' \oplus T_1)$ due to the assumption of exclusive-or operation. Continuously, based on the exclusive-or operation, she is continually unable to compute the response message A_4 without a_1 . Therefore, the proposed scheme can resist server spoofing attack.

4.3.5 Replay Attack

Suppose that the malicious attacker Eve attempts to intercept Alice's login message during the transmission of the message and execute the replay attack. She delivers the request $\{NID, A_2, T_1\}$ to the LS. Regardless of whether the login message is fresh, the request is authenticated through examination of the timestamp T_1. As the message $A_2 = h(A_1 \oplus T_1) \oplus a_1$ contains the timestamp T_1, which is protected by the hash function, Eve is unable to adjust the timestamp T_1 protected in A_2 according to the security assumption of hash function. Once LS contracts both timestamps T_1 which contains in the message $\{NID, A_2, T_1\}$ and parameter $A_2 = h(A_1 \oplus T_1) \oplus a_1$, it would learn that they are unequal. Therefore, the LS can confirm that the login request is not a fresh message. Hence, Eve fails to replay the login request.

4.3.6 Man-in-the-middle Attack

In the man-in-the-middle attack, the attacker Eve may attempt to impersonate a legal user Alice and server LS utilizing intercepted the message. In our scheme, Alice and LS can verify each other's identities based on the mutual authentication. Hence, the proposed scheme is secure against man-in-the-middle attack.

4.4 BAN Logic

We employ the BAN logic model to prove the correctness of mutual authentication. The notations used in the BAN logic are defined in Table 2. Beside, *P* and *Q* range over participants; *X* and *Y* refer to statements; *k* denotes the encryption and decryption key [10].

Table 2. BAN logic notations

Notation	Definition
$P \mid\equiv X$	*P* believes in *X*
$P \triangleleft X$	*P* sees *X* (receive)
$P \mid\sim X$	*P* once said *X* (send)
$P \mid\Rightarrow X$	*P* has jurisdiction over *X*
$\#(X)$	The formula *X* is fresh
$\{X\}_k$	The formula *X* is encrypted under the *k*
$P \overset{k}{\leftrightarrow} Q$	*P* and *Q* may use the shared key *k* to communicate
$< X >_Y$	The formula *X* is combined with the Formula *Y*

We are able to prove the mechanism by three logical rules: *message-meaning*, *nonce-verification*, and *jurisdiction.*

R1. *The message-meaning*

$$\frac{P \mid\equiv Q \leftrightarrow P, P \lhd \{X\}_k}{P \mid\equiv Q \mid\sim X}$$

If *P* believes that the key *k* is shared with *Q* and sees the message *X* encrypted by *k*, *P* can believe in that *Q* once said the message *X*.

R2. *The nonce-verification*

$$\frac{P \mid\equiv \#(X), P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X}$$

If *P* believes the message *X* is fresh and that *Q* once said the message *X*, then *P* can believe that *Q* believes in the message *X*.

R3. *The jurisdiction*

$$\frac{P \mid\equiv Q \mid\Rightarrow X, P \mid\equiv Q \mid\equiv X}{P \mid\equiv X}$$

If *P* believes that *Q* has jurisdiction over *X* and that *Q* believes the message *X*, then *P* believes in the message *X*.

Since Alice is a legitimate player, we only need to certificate that Alice and LS can authenticate to each other. The following goals shall be confirmed in the novel penny M-lottery mechanism.

G1. $Alice \mid\equiv SK$

G2. $LS \mid\equiv SK$

G3. $Alice \mid\equiv LS \mid\equiv SK$

G4. $LS \mid\equiv Alice \mid\equiv SK$

According to the sequence of BAN logic, the messages of the communication must be transferred to the idealized form. First, we simplify the messages to the generic type as follows.

M1. $Alice \rightarrow LS \quad NID, A_2, T_1$

M2. $LS \rightarrow Alice \quad NID', A_4, A_5, T_2$

M3. $Alice \rightarrow LS \quad E_{SK}[NID']$

Subsequently, we transfer the generic messages into the idealized form.

I1. $Alice \rightarrow LS \quad \{ID\}_x, \{a_1, T_1\}_{ID}$

I2. $LS \rightarrow Alice \quad <\mathrm{A}_3, \mathrm{T}_2>_{a_1}, <a_2>_{A_3}$

I3. $Alice \rightarrow LS \quad \{ID\}_{SK}$

The followings are the assumptions according to the authorized messages in the registration phase.

A1. $Alice \mid\equiv ID$

A2. $LS \mid\equiv ID$

A3. $Alice \mid\equiv LS \mid\equiv ID$

A4. $LS \mid\equiv Alice \mid\equiv ID$

A5. $Alice \mid\equiv a_1$

A6. $LS \mid\equiv a_2$

A7. $LS \mid\equiv x$

A8. $LS \mid\equiv Alice \mid\Rightarrow ID$

A9. $LS \mid\equiv Alice \overset{ID}{\leftrightarrow} LS$

Proof of G1-G6:

From I1, we have

D1. $LS \lhd \{ID\}_x, \{a_1, T_1\}_{ID}$

D2. $LS \mid\equiv \#(T_1)$

According to *the message-meaning rule*, A9, and D1, we obtain the following

D3. $LS \mid\equiv Alice \mid\sim \{a_1, T_1\}$

Combining I1 and D3, we can derive D4 as

D4. $LS \mid\equiv \#(a_1)$

Using t*he nonce-verification rule*, D2, and D4, we can infer that that

D5. $LS \mid\equiv Alice \mid\equiv a_1$

Based on *the jurisdiction rule*, A8, and D5, we can obtain

D6. $LS \mid\equiv a_1$

Here LS is able to compute $SK = h(NID, a_1, a_2)$. This means that LS must believe in the session key. Since LS can trust all the parameter of session key.

We have the following from A2, A6, and D6.

D7. $LS \mid\equiv SK$

From I2, we have

D8. $Alice \lhd < A_3, T_2 >_{a_1}, < a_2 >_{A_3}$

According A5, D8, Alice believes in a_1 to decrypt the ciphertext. Thus, Alice could trust the legality of the message to infer D9.

D9. $Alice \mid\equiv A_3, T_2$

Considering D8, we have

D10. $Alice \mid\equiv a_2$

Combining A1, A5, and D10, Alice believes the parameters of session key and she must trust the session key to have D11.

D11. $Alice \mid\equiv SK$

With A5, I1, and D8, we can derive that

D12. $Alice \mid\equiv Alice \overset{a_1}{\leftrightarrow} LS$

D13. $Alice \mid\equiv LS \mid\equiv a_1$

Applying D8 and D12 to *the message-meaning rule*, we can get

$$\text{D14. } Alice \mid\equiv LS \mid\sim A_3$$

From I2, we can further deduce that

$$\text{D15. } Alice \mid \#(T_2)$$

According I2 and D15, we can derive

$$\text{D16. } Alice \mid \#(A_3)$$

With *the nonce-verification rule*, D14, and D16, we get the following

$$\text{D17. } Alice \mid\equiv LS \mid\equiv A_3$$

Then we can employ D9 and D17 to derive

$$\text{D18. } Alice \mid\equiv Alice \overset{A_3}{\leftrightarrow} LS$$

Based on D8, we have

$$\text{D19. } Alice \lhd\!< a_2 >_{A_3}$$

With *the message-meaning rule*, D18, and D19, we can infer the result as

$$\text{D20. } Alice \mid\equiv LS \mid\sim a_2$$

Combining D8 and D16, we can conclude that

$$\text{D21. } Alice \mid\equiv \#(a_2)$$

Bringing D20 and D21 into *the nonce-verification rule*, we can have

$$\text{D22. } Alice \mid\equiv LS \mid\equiv a_2$$

According A3, D13, and D22, we can deduce

$$\text{D23. } Alice \mid\equiv LS \mid\equiv SK$$

Using I3, we have

$$\text{D24. } LS \lhd \{ID\}_{SK}$$

Finally, we can have the result from A2, D11, and D24 as

$$\text{D25. } LS \mid\equiv Alice \mid\equiv SK$$

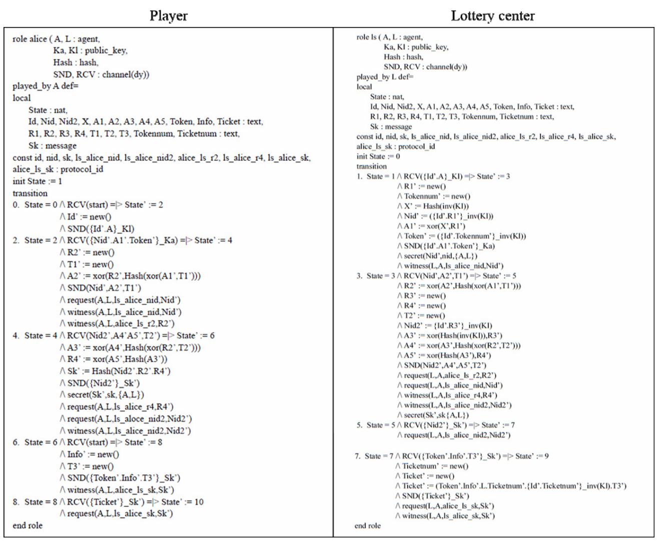
Afterwards, we have beliefs of all the goals from D7, D11, D23, and D25.

4.5 Simulation Using AVISPA

In this section, the formal tool: AVISPA, is utilized to validate our scheme [12]. There are four back-end validators, including On-the-Fly-Model-Checker (OFMC), Constraint-Logic-based Attack Searcher (CL-ATSE), Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP), and SAT-based Model-Checker (SATMC). The tool is used to examine whether the proposed method has the vulnerability or suffers from network attacks. The version of AVISPA for simulation is Security Protocol Animator version 1.6 (SPAN 1.6) installed on an Ubuntu 10.10 Light workstation with an Intel® Core™ i5-5200U CPU running at 2.20 GHz with 1.00 GB of RAM.

4.5.1 Protocol Specification

AVISPA is described by High Level Protocol Specification Language (HLPSL). According to the syntax of AVISPA and the used parameters, there are two roles in the proposed protocol: players and lottery server. They are clearly defined in the protocol using HLPSL as shown in Figure 6. Under the HLPSL description, we replace the random number a_1 and a_2 with R_2 and R_4 since the values a_1 and a_2 are conflict with the security parameters A_1 and A_2.



Player

```
role alice ( A, L : agent,
            Ka, Kl : public_key,
            Hash : hash,
            SND, RCV : channel(dy))
played_by A def=
local
      State : nat,
      Id, Nid, Nid2, X, A1, A2, A3, A4, A5, Token, Info, Ticket : text,
      R1, R2, R3, R4, T1, T2, T3, Tokennum, Ticketnum : text,
      Sk : message
const id, nid, sk, ls_alice_nid, ls_alice_nid2, alice_ls_r2, ls_alice_r4, ls_alice_sk,
alice_ls_sk : protocol_id
init State := 1
transition
0.  State = 0 /\ RCV(start) =|> State' := 2
              /\ Id' := new()
              /\ SND({Id'.A}_Kl)
2.  State = 2 /\ RCV({Nid'.A1'.Token'}_Ka) =|> State' := 4
              /\ R2' := new()
              /\ T1' := new()
              /\ A2' := xor(R2',Hash(xor(A1',T1')))
              /\ SND(Nid',A2',T1')
              /\ request(A,L,ls_alice_nid,Nid')
              /\ witness(A,L,ls_alice_nid,Nid')
              /\ witness(A,L,alice_ls_r2,R2')
4.  State = 4 /\ RCV(Nid2',A4'A5',T2') =|> State' := 6
              /\ A3' := xor(A4',Hash(xor(R2',T2')))
              /\ R4' := xor(A5',Hash(A3'))
              /\ Sk' := Hash(Nid2'.R2'.R4')
              /\ SND({Nid2'}_Sk')
              /\ secret(Sk',sk,{A,L})
              /\ request(A,L,ls_alice_r4,R4')
              /\ request(A,L,ls_aloce_nid2,Nid2')
              /\ witness(A,L,ls_alice_nid2,Nid2')
6.  State = 6 /\ RCV(start) =|> State' := 8
              /\ Info' := new()
              /\ T3' := new()
              /\ SND({Token'.Info'.T3'}_Sk')
              /\ witness(A,L,alice_ls_sk,Sk')
8.  State = 8 /\ RCV({Ticket'}_Sk') =|> State' := 10
              /\ request(A,L,ls_alice_sk,Sk')
end role
```

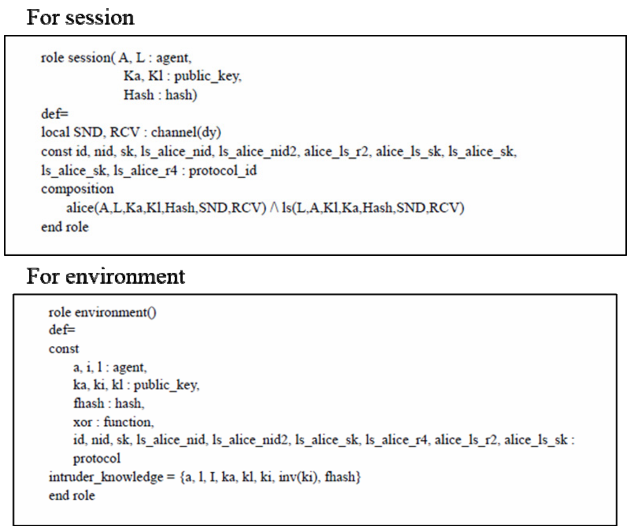
Lottery center

```
role ls ( A, L : agent,
          Ka, Kl : public_key,
          Hash : hash,
          SND, RCV : channel(dy))
played_by L def=
local
     State : nat,
     Id, Nid, Nid2, X, A1, A2, A3, A4, A5, Token, Info, Ticket : text,
     R1, R2, R3, R4, T1, T2, T3, Tokennum, Ticketnum : text,
     Sk : message
const id, nid, sk, ls_alice_nid, ls_alice_nid2, alice_ls_r2, ls_alice_r4, ls_alice_sk,
alice_ls_sk : protocol_id
init State := 0
transition
1.  State = 1 /\ RCV({Id'.A}_Kl) =|> State' := 3
              /\ R1' := new()
              /\ Tokennum' := new()
              /\ X' := Hash(inv(Kl))
              /\ Nid' := ({Id'.R1'}_inv(Kl))
              /\ A1' := xor(X',R1')
              /\ Token' := ({Id'.Tokennum'}_inv(Kl))
              /\ SND({Id'.A1'.Token'}_Ka)
              /\ secret(Nid',nid,{A,L})
              /\ witness(L,A,ls_alice_nid,Nid')
3.  State = 3 /\ RCV(Nid',A2',T1') =|> State' := 5
              /\ R2' := xor(A2',Hash(xor(A1',T1')))
              /\ R3' := new()
              /\ R4' := new()
              /\ T2' := new()
              /\ Nid2' := {Id'.R3'}_inv(Kl)
              /\ A3' := xor(Hash(inv(Kl)),R3')
              /\ A4' := xor(A3',Hash(xor(R2',T2')))
              /\ A5' := xor(Hash(A3'),R4')
              /\ SND(Nid2',A4',A5',T2')
              /\ request(L,A,alice_ls_r2,R2')
              /\ request(L,A,ls_alice_nid,Nid')
              /\ witness(L,A,ls_alice_r4,R4')
              /\ witness(L,A,ls_alice_nid2,Nid2')
              /\ secret(Sk',sk{A,L})
5.  State = 5 /\ RCV({Nid2'}_Sk') =|> State' := 7
              /\ request(L,A,ls_alice_nid2,Nid2')

7.  State = 7 /\ RCV({Token'.Info'.T3'}_Sk') =|> State' := 9
              /\ Ticketnum' := new()
              /\ Ticket' := new()
              /\ Ticket' := (Token'.Info'.L.Ticketnum'.{Id'.Ticketnum'}_inv(Kl).T3')
              /\ SND({Ticket'}_Sk')
              /\ request(L,A,alice_ls_sk,Sk')
              /\ witness(L,A,ls_alice_sk,Sk')
end role
```

Figure 6. HLPSL for roles

In order to imitate the attacker, Dolev-Yao model is applied in experiments to simulate the intruder knowledge [13]. The syntax of session and environment are depicted in HLPSL as shown in Figure 7. According to Figure 8, we particularly verify the perfect forward secrecy, impersonation attack, and server spoofing attack.



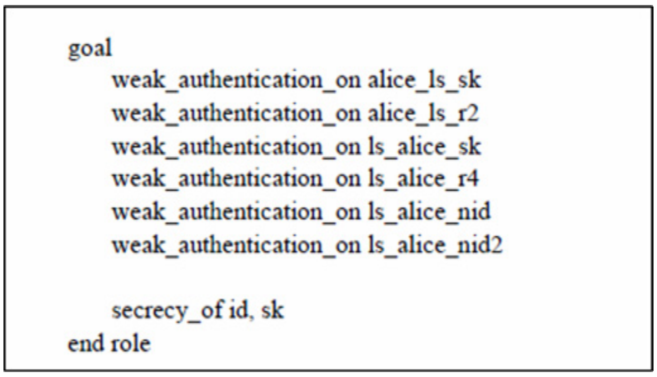
For session

```
role session( A, L : agent,
              Ka, Kl : public_key,
              Hash : hash)
def=
local SND, RCV : channel(dy)
const id, nid, sk, ls_alice_nid, ls_alice_nid2, alice_ls_r2, alice_ls_sk, ls_alice_sk,
ls_alice_sk, ls_alice_r4 : protocol_id
composition
     alice(A,L,Ka,Kl,Hash,SND,RCV) /\ ls(L,A,Kl,Ka,Hash,SND,RCV)
end role
```

For environment

```
role environment()
def=
const
     a, i, l : agent,
     ka, ki, kl : public_key,
     fhash : hash,
     xor : function,
     id, nid, sk, ls_alice_nid, ls_alice_nid2, ls_alice_sk, ls_alice_r4, alice_ls_r2, alice_ls_sk :
     protocol
intruder_knowledge = {a, l, I, ka, kl, ki, inv(ki), fhash}
end role
```

Figure 7. HLPSL for composed roles



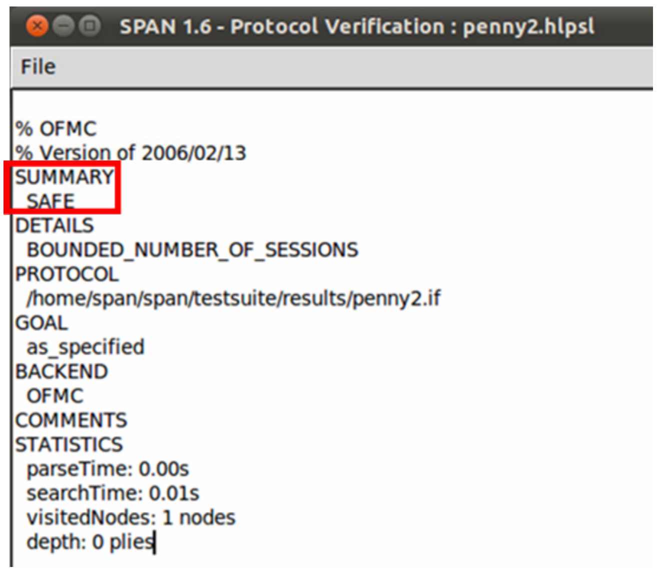
```
goal
    weak_authentication_on alice_ls_sk
    weak_authentication_on alice_ls_r2
    weak_authentication_on ls_alice_sk
    weak_authentication_on ls_alice_r4
    weak_authentication_on ls_alice_nid
    weak_authentication_on ls_alice_nid2

    secrecy_of id, sk
end role
```

Figure 8. HLPSL for goal

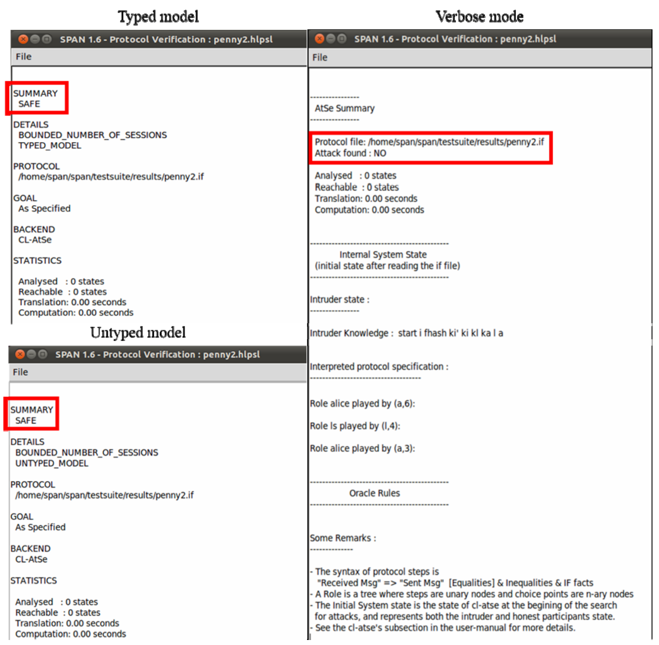
4.5.2 Simulation and Analysis

Here, the security of our mechanism is validated. Since the back-end validators, TA4SP and SATMC, do not support the xor operation, we apply CL-ATSE and OFMC to analyze our method. The major difference between two back-ends is the approach applying to intermediate format (IF). The optimized Baader & Schulz unification algorithm is directly utilized in CL-ATSE to deal with the IF's [14]. On the other hand, the modularization is adopted in OFMC to rewrite IF's. The outcome of OFMC is safe as shown in Figure 9. Regarding to CL-ATSE, our mechanism is simulated in typed model, untyped model, and verbose mode, respectively. In typed model, all parameter types are considered in a program. On the contrary, all variables are regards as generic type in untyped model. As to the verbose mode, the whole intruder trace is illustrated once the protocol is insecure. As shown in Figure 10, the result of CL-ATSE has demonstrated that the proposed protocol is secure against any potential risk.



```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/penny2.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.01s
  visitedNodes: 1 nodes
  depth: 0 plies
```

Figure 9. The output format of OFMC



Typed model

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /home/span/span/testsuite/results/penny2.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 0 states
  Reachable  : 0 states
  Translation: 0.00 seconds
  Computation: 0.00 seconds
```

Untyped model

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  UNTYPED_MODEL

PROTOCOL
  /home/span/span/testsuite/results/penny2.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 0 states
  Reachable  : 0 states
  Translation: 0.00 seconds
  Computation: 0.00 seconds
```

Verbose mode

```
----------------
 AtSe Summary
----------------

 Protocol file: /home/span/span/testsuite/results/penny2.if
 Attack found : NO

 Analysed   : 0 states
 Reachable  : 0 states
 Translation: 0.00 seconds
 Computation: 0.00 seconds


------------------------------------------
         Internal System State
 (initial state after reading the if file)
------------------------------------------

Intruder state :
----------------

Intruder Knowledge :  start i fhash ki' ki kl ka l a


Interpreted protocol specification :
----------------------------------

Role alice played by (a,6):

Role ls played by (l,4):

Role alice played by (a,3):


------------------------------------------
             Oracle Rules
------------------------------------------


Some Remarks :
--------------

- The syntax of protocol steps is
   "Received Msg" => "Sent Msg"  [Equalities] & Inequalities & IF facts
- A Role is a tree where steps are unary nodes and choice points are n-ary nodes
- The Initial System state is the state of cl-atse at the begining of the search
  for attacks, and represents both the intruder and honest participants state.
- See the cl-atse's subsection in the user-manual for more details.
```

Figure 10. The outcome format of CL-ATSE

5 Experiments

In this chapter, we present the analysis of the study questionnaire, computational cost, and communication overhead. We implemented the operations on a personal computer (Pentium Dual-Core with E6700 3.20 GHz processor, 4 GB bytes memory, and Windows 7 operating system) and smartphone (ASUS_Z00ED with Quad-core 1.2 GHz processor, 2 GB memory, and Google Android 5.0 operating system).

5.1 Computational Cost

In this subsection, we discuss the computational cost of players and LS in the registration phase, login phase, purchasing phase, and claim prize phase. Let T_{XOR} , T_h , $T_{E_{AES}}$, and $T_{D_{AES}}$ denote the time spent running an exclusive-or operation, a one-way hash function, an AES encryption, an AES decryption, Rivest–Shamir–Adleman (RSA) encryption, and RSA decryption, respectively. The definite running time is shown in Table 3. In addition, we executed each operation 10,000 times and calculated the average to evaluate the performance.

Table 3. Execution time

	T_{XOR}	T_h	$T_{E_{AES}}$	$T_{D_{AES}}$
User	0.000436 ms	0.06 ms	0.084 ms	0.075 ms
Server	0.000034 ms	0.008 ms	0.028 ms	0.027 ms

In the registration phase, our scheme must execute six exclusive-or operations and three one-way hash functions. Therefore, the user's running time could be $3T_h + 6T_{XOR} \approx 0.055$ ms . As LS in the registration phase must execute two AES encryptions, a one-way hash function, and an exclusive-or operation. Hence, the execution time is $2T_{E_{AES}} + T_h + T_{XOR} \approx 0.052$ ms. Calculating the computational cost of the user requires executing one AES encryption, seven one-way hash functions, and one exclusive-or operation in the login phase. The running time is $T_{E_{AES}} + 7T_h + 10T_{XOR} \approx$ 0.5 ms. On the other hand, the server must perform two AES decryptions, one AES encryption, six one-way hash functions, and seven exclusive-or operations. The execution time can thus be $2T_{D_{AES}} + T_{E_{AES}} + 6T_h + 7T_{XOR}$ ≈ 0.13 ms. In the ticket purchasing phase, each user has to execute one RSA decryption, one AES encryption, and one exclusive-or operation. Therefore, the execution time is $T_{D_{AES}} + T_{E_{AES}} + T_{XOR} \approx$ 0.159 ms. Moreover, calculating the computational cost of the server in the ticket purchasing phase requires playing two AES decryptions and two AES encryptions. The execution time is $2T_{D_{AES}} + 2T_{E_{AES}} \approx 0.11$ ms. Calculating the computation costs of the user and LS in the prize claiming phase requires executing two hash operations, two exclusive-or operations, and two AES decryptions. The running time of the user is approximate to $T_h + T_{XOR} \approx$ 0.06 ms, while that of LS approaches $T_h + T_{XOR} + 2T_{D_{AES}} \approx 0.06$ ms. The computational cost results are illustrated in Table 4.

Table 4. Computational cost results

	Registration	Login	Purchasing	Claim prize
User	$3T_h + 6T_{XOR}$	$T_{E_{AES}} + 7T_h + 10T_{XOR}$	$T_{D_{AES}} + T_{E_{AES}} + T_{XOR}$	$T_h + T_{XOR}$
Server	$2T_{E_{AES}} + T_h + T_{XOR}$	$2T_{D_{AES}} + T_{E_{AES}} + 6T_h + 7T_{XOR}$	$2T_{D_{AES}} + 2T_{E_{AES}}$	$T_h + T_{XOR} + 2T_{D_{AES}}$

Table 4 shows the computational costs of users and the LS in each session. We estimate the computational costs of users via a smartphone. If a user decides to participate, he or she must undergo the login and ticket purchasing phases. The experimental results reveal the time spent on the user side to be 0.66 ms. Generally, the computational capacity of a smartphone is burdened by two iterations of asymmetric encryption. In other words, the validity of the proposed scheme could be confirmed by applying it on mobile devices. Even players purchasing many tickets for each lottery round would not increase the cost of time in the mobile environment. Table 4 shows that our mechanism performs lightweight operations such as exclusive-or operations and one-way hash functions, both of which can reduce the computational cost. Based on the descriptions in this subsection, our scheme can achieve the lightweight assumption.

5.2 Communication Overhead

In our scheme, we assume that the length of the user's identity and timestamp is 32 bits, the block size of the AES encryption and decryption is 128 bits, and the output of one-way hash function is 256 bits.

The communication cost of the registration phase for sending the identity *ID* and receiving the parameters *NID*, A_1, *token* is described as follows. *NID* and A_1 are the results of exclusive-or operations, and *token* is the AES encryption result. Therefore, the communication cost is $32 + 128 + 256 + 128 = 544$ bits. In the login phase, each user sends NID, A_2, T_1 to the LS. Then, LS responses parameters of *NID*, A_4, A_5, and T_1, and the user transmits a parameter protected by the AES. Here, A_4 and A_5 are the outcomes of exclusive-or operations. Consequently, the communication cost is $32 \times 2 + 128 \times 3 + 256 \times 3 = 1216$ bits. In the ticket purchasing phase, each user sends a purchasing request, and LS responds to user a lottery ticket. These messages are the results of AES encryption. In other words, the communication cost is $256 + 384 = 640$ bits. Each user continually transmits the winning ticket to the LS. This message is a consequence of one-way hash and timestamp. The communication cost is 288 bits. The communication costs are shown in Table 5.

Table 5. Communication overheads

Phase	Length
Registration	544 bits
Login	1216 bits
Purchasing	640 bits
Claim prize	288 bits

The penny M-lottery is implemented in mobile environments. Hence, the messages are delivered through the 4G network. If the bandwidth of the 4G network is 20 MHz and the data rate is 300 Mbit/s [11], it costs 0.0016 ms to send messages during the registration phase. Furthermore, the transmission time is 0.0035 ms in the login phase and 0.0018 ms in purchasing phase. In the prize claiming phase, conveying the messages costs 0.0007 ms. Moreover, each user must undergo the login and purchasing phases in each lottery round. Overall, the communication cost of each round is 0.007 ms. Therefore, players need not spend excessive amounts of time purchasing tickets. Although each participant purchases a specific quantity of tickets in one round, this does not increase the data transmission and communication costs. Therefore, the communications costs in the proposed scheme are considered low.

5.3 Questionnaire

To understand the relationship between ticket sales volume and the size of the jackpot bonus, we developed a questionnaire. Moreover, we analyze the total cost associated with the time spent by players purchasing various ticket quantities through their smartphones. We enrolled 1000 sample subjects and investigated their lottery participation intentions. The sample consisted of 553 men and 447 women, 672 of whom were office workers and 328 were students. First, we introduced the lottery rules of this examination as follows: If the jackpot is NT$100 and each ticket costs NT$5, 20 tickets are sold. In other words, the quantity of tickets to be sold is adjusted according to the jackpot in each round. We defined three cases to investigate how many tickets each subject would purchase with jackpots of NT$3500, NT$30,000, and NT$80,000. The answers are discussed as follows.

Case 1: NT$3500 jackpot

How many tickets would you purchase?

When the jackpot was NT$3,500 (Figure 11), approximately 50% of the players wanted to purchase fewer than five tickets. When the jackpot was NT$30,000 (Figure 12), the proportion of players who wanted to purchase more than 20 tickets was 25%; however, the proportion of those who wanted to purchase fewer than five tickets declined. Once the jackpot was NT$80,000 (Figure 13), approximately 50% of the players wanted to purchase more than 20 tickets. Based on Figures 11-13, the study participants were willing to spend more money to participate in the lottery when the jackpot is higher

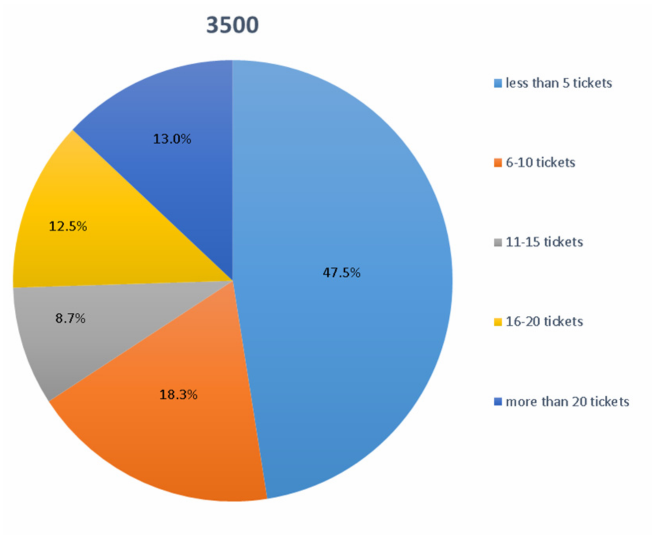


Figure 11. Analytic data 1

Case 2: NT$30,000 jackpot

How many tickets would you purchase?

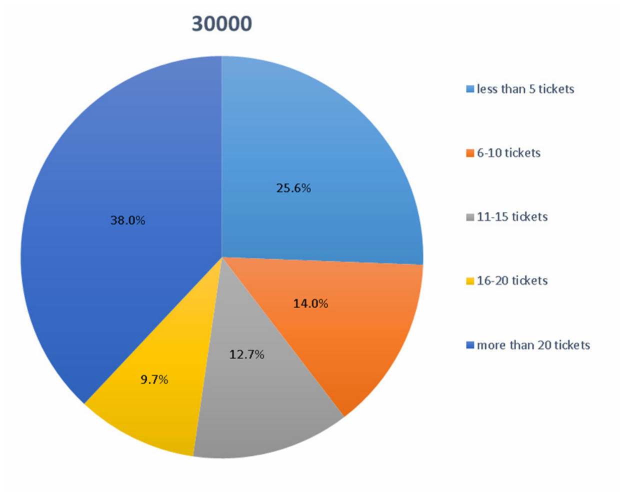


Figure 12. Analytic data 2

Case 3: NT$80,000 jackpot

How many tickets would you purchase?

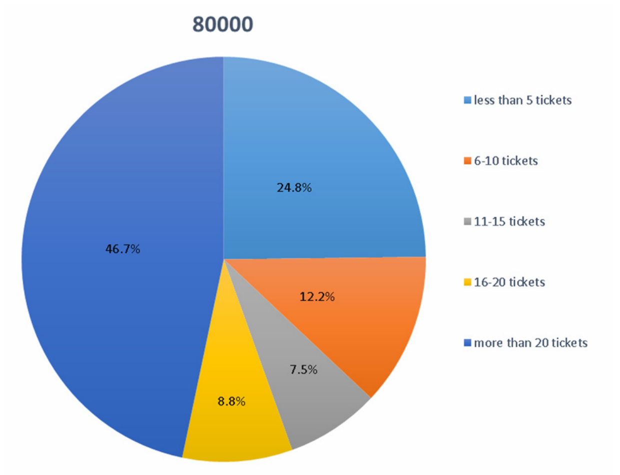


Figure 13. Analytic data 3

As revealed by the statistical data, most players would purchase more than one ticket regardless of the jackpot amount. Hence, we analyzed the execution time spent purchasing tickets to determine the relationship between execution time and number of tickets purchased based on information from the mobile device and LS. Because the questionnaire options for ticket purchase quantity were 1-5, 6-10, 11-15, 16-20, and more than 20, we selected 1, 6, 11, 16, and 21 tickets to represent the options. The definitions for execution time are shown in Table 6 and Table 7, while the analysis results are illustrated in Figure 14 and Figure 15.

Table 6. Definitions of execution time on the user side

Time	Definition
T1	The run time of the login phase by mobile device
T2	The run time of encrypting the purchasing information
T3	The time of decrypting tickets
T4	Total execution time of user

Table 7. Definitions of execution time on the server side

Time	Definition
T5	The run time of the login phase on server side
T6	The run time of generating tickets
T7	The time of encrypting tickets
T8	Total execution time of server

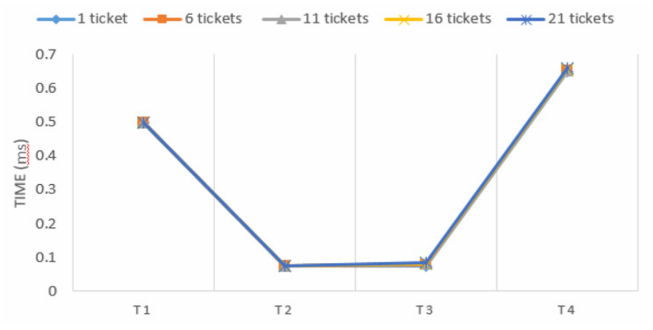


Figure 14. User execution time

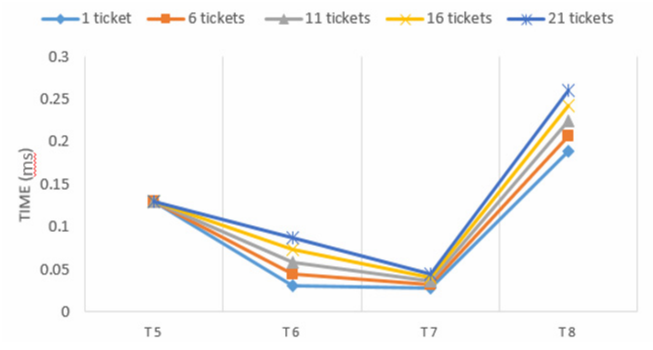


Figure 15. Server execution time

Regardless of the number of tickets a player purchases, the execution time in the login phase T1 and encrypted time of purchasing information T2 would be equal. When players purchase different ticket quantities, the lengths of encrypted information are distinct. Thus, T3 is not equal in each case. Therefore, the total execution time exerts little influence when various ticket quantities are purchased on the user side.

T6 is the total cost time spent by LS for generating tickets. However, the ticket quantity is not identical in each case. Therefore, the time spent producing tickets diverges on the server side. The experimental results show the computational time of generating a ticket to be 0.0028 ms. From the statistics analysis of this questionnaire, it is clear that most of players are willing to purchase multiple tickets for each play. Thus, we have to examine the performance of mobile device and server node under the scenario that many players buy multiple tickets at the same time. According to the execution time of user device, only the decryption time is influenced by the number of tickets. In fact, the entropy of ciphertext would not be significantly increased with the growth number of ticket. This has led to a small amount of time spent for purchasing multiple tickets on the mobile device. As to the server node, the time for each play depends on the generation of ticket. Nevertheless, the ticket is computed without the usage of asymmetric cryptosystem. LS would not be burdened even if it has to generate lots of the ticket for each player in each round.

6 Conclusions

In this paper, we have proposed a mobile penny lottery mechanism called penny M-lottery, in which participants can use mobile devices to join the play to have the opportunity of winning a large amount of money with low-cost tickets. The experimental results show that our scheme is lightweight so that participants can use mobile devices to purchase many tickets for a single lottery round without burdening their devices. In particular, we have demonstrated the mutual authentication between LS and players through BAN logic model. Moreover, our mechanism can resist malicious attacks to guarantee the privacy and right of player.

References

[1] K. Sako, Implementation of a Digital Lottery Server on WWW, *Proceedings of the International Exhibition and Congress on Secure Networking*, Germany, 1999, pp. 101-108.

[2] J. Zhou, C. Tan, Playing Lottery on the Internet, *Proceedings of the Third International Conference on Information and Communications Security*, Xian, China, 2001, pp. 189-201.

[3] D. M. Goldschlag, S. G. Stubblebine, Publicly Verifiable Lotteries: Applications of Delaying Functions, *Proceedings of the Second International Conference on Financial Cryptography*, Anguilla, British West Indies, 2006, pp. 214-226.

[4] C.-L. Chen, Y.-H. Liao, W.-J. Tsaur, A Secure and Fair Joint e-Lottery Protocol, *The Scientific World Journal*, Vol. 2014, No. 139435, April, 2014.

[5] E. Kushilevitz, T. Rabin, Fair e-Lotteries and e-Casinos, *Proceedings of the Cryptographers' Track at the RSA Conference 2001*, San Francisco, CA, USA, 2001, pp. 100-109.

[6] J.-S. Lee, C.-C. Chang, Design of Electronic t-out-of-n Lotteries on the Internet, *Computer Standards and Interfaces*, Vol. 31, No. 2, pp. 395-400, February, 2009.

[7] C.-L. Chen, M.-L. Chiang, W.-C. Lin, D.-K. Li, A Novel Lottery Protocol for Mobile Environments, *Computers and Electrical Engineering*, Vol. 49, pp. 146-160, January, 2016.

[8] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 4th ed., Prentice Hall, 2003.

[9] A. J. Menezes, J. Katz, P. C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[10] M. Burrows, M. Abadi, R. Needham, A Logic of Authentication, *ACM Transactions on Computer Systems*, Vol. 8, No. 1, pp.

18-36, February, 1990.

[11] P. Skocir, D. Katusic, I. Novotni, I. Bojic, G. Jezic, Data Rate Fluctuations from User Perspective in 4G Mobile Networks, *2014 22nd International Conference on Software, Telecommunications and Computer Networks*, Split, Croatia, 2014, pp. 180-185.

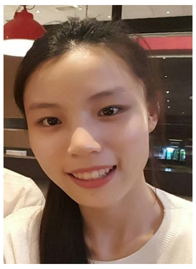
[12] L. Viganò, Automated Security Protocol Analysis with the AVISPA Tool, *Electronic Notes in Theoretical Computer Science*, Vol. 155, pp. 61-86, May, 2006.

[13] D. Dolev, A. C. Yao, On the Security of Public Key Protocols, *IEEE Transactions on Information Theory*, Vol. 29, No. 2, pp. 198-208, March, 1983.

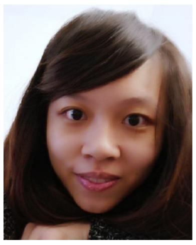
[14] F. Baader, K. U. Schulz, Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures, *Journal of Symbolic Computation*, Vol. 21, No. 2, pp. 211-243, February, 1996.

Biographies

Jung-San Lee has worked as a professor in the Department of Information Engineering and Computer Science at Feng Chia University, Taichung, Taiwan. His current research interests include image processing, information security, and network management.



Ying-Chin Chen has received her M.S. degree in information engineering and computer science in 2018, in Feng Chia University, Taichung, Taiwan. Her current research interests include information security and visual secret sharing.



Ya-Han Kang has received her M.S. degree in information engineering and computer science in 2017, in Feng Chia University, Taichung, Taiwan. Her current research interests include information security and e-commerce.

Ren-Kai Yang is pursuing his M.S. degree in information engineering and computer science in Feng Chia University, Taichung, Taiwan. His current research interests include information security and network management.