Mobile Agent Platform based Wallet for Preventing Double Spending in Offline e-Cash

Irwan*[1], Armein Z. R. Langi[2], Emir Husni[3]
School of Electrical Engineering and Informatics
Institut Teknologi Bandung,
Bandung, Indonesia

***Abstract*—Electronic cash (or e-cash) research has been going on for more than three decades since it was first proposed. Various schemes and methods are proposed to improve privacy and security in e-cash, but there is one security issue that less discussed mainly in offline e-cash, namely, double-spending. Generally, the mechanism to deal with double-spending in offline e-cash is performing double-spending identification when depositing the coin. Even though the mechanism is successful in identifying double-spender, but it cannot prevent double-spending. This paper proposes the Mobile Agent Platform based Wallet (MAPW) to overcome the double-spending issue in offline e-cash. MAPW uses the autonomy and cooperation of agents to give protection against malicious agent, counterfeit coin and duplicate coin. This model has been verified using Colored Petri Nets (CPN) and has proven to be successful in preventing double-spending, and overcoming malicious agent, and counterfeit coins.**

***Keywords*—e-Cash; double-spending; MAPW; CPN**

I. Introduction

Nowadays, electronic payment has proliferated along with the use of the Internet. The electronic payments can be classified into four categories: online credit card, electronic check, smart cards based electronic payment, and e-cash [1]. E-cash is the only electronic payment that provides not only security but also the privacy of its users. E-cash generally consists of three types of entities: bank, user, and merchant. The user withdraws coins from the bank and spends it on the merchant who then deposits the coins to the bank. The security aspect of e-cash should cover some main properties (1) *unforgeability*: no user can create coin other than the authorized party; (2) *no framing*: no one except the owner of a coin can spend it; and (3) *double-spending prevention*: coin can only be spent once. The privacy aspect covers the *anonymity of users*, which mean no one can discover the true identity of a user correlated with withdrawal or spending transaction.

Following the first e-cash scheme introduced by Chaum [2], [3], many e-cash schemes [4], [5], [6], [7] have been proposed, and most of them focus on improving privacy and security (unforgeability and no framing). These proposed e-cash schemes only provide double-spending identification to overcome double-spending. Double-spending is a security issue in which the same coin can be spent more than once since an e-cash's coin is a set of digital data that can be duplicated easily. Double-spending identification is a mechanism to identify whether a coin is a duplicated or not. If the coin is identified as duplicated coin, e-cash system revoke anonymity and discover the identity of the duplicated coin owner. The double-spending identification successfully discovers the identity of double-spender, but it cannot prevent double-spending in advance, especially for offline e-cash scheme.

Several existing proposed methods to fulfill the property of double-spending prevention such as blockchain [8], smartcard [9], and mobile agent [10]. Nakamoto proposed the use of blockchain in a peer-to-peer e-cash system to prevent double-spending [8]. Blockchain is a global ledger where all transaction recorded. Every transaction must be broadcasted to all nodes and added to the ledger. Because all nodes keep this ledger, it is impossible to perform double-spending. The blockchain-based method is comparatively slow in transaction speed and confirmation. As reported by Statista, average confirmation time of bitcoin, which is one of e-cash scheme that implements the blockchain method, is 9.47 minutes[1].

In order to manage the double-spending problem in offline e-cash scheme, Liu proposed a method that uses a smartcard that records a pair of all withdrawn coins [9]. When a customer spends a coin, a merchant requests the pair of the spent coin to the smartcard. Smartcard searches the pair to prove that the coin has not been spent yet. If the pair exists in the smartcard, merchant accepts the coin while smartcard deletes the pair of the coin. The customer cannot spend the same coin because a pair of the coin no longer exist in the smartcard. However, this method does not provide any mechanism to prevent a pair of the spent coin rewritten in the smartcard.

Furthermore, Salama proposed a more advanced method in against the double-spending problem by using Optical Memory Card (OMC) and mobile agent [10]. OMC is a write-only card that used for recording the serial number of the spent coin. The mobile agent is used as a coin that can identify the spent coin which its serial number has been recorded in OMC. Hence, customer cannot spend the same coin if the serial number of the coin recorded in OMC. This method can prevent double-spending in advance but has limitation in OMC memory. When OMC has no memory space, the double-spending prevention capability cannot be performed.

From the above analysis, the existing methods have not been able to prevent double-spending optimally. These methods still have open issues in the slow confirmation time, the inability to counter data of spent coin to be copied on the smartcard and limited data storage space on OMC. Double-spending causes financial losses, so this issue is paramount to

[1]Average confirmation time of Bitcoin transactions from June 2017 to June 2018 according to https://www.statista.com/statistics/793539/bitcoin-transaction-confirmation-time / accessed 31 July 2019

resolve. Thus, we need to construct a method that meets the double-spending prevention property and prove the security of the method. This paper proposes the Mobile Agent Platform based Wallet (MAPW) model that is not only able to deter counterfeit coin and prevent double-spending but also protect the mobile agent platform from malicious agents.

This paper is organized as follows. In Section 2, we describe the preliminaries on offline e-cash and agent's technology in e-cash. Overview of the proposed MAPW model is presented in Section 3. In Section 4, the Colored Petri Nets (CPN) model of MAPW is described, and the analysis of security is presented in Section 5. Finally, Section 6 concludes the proposed model and Section 7 gives pointers to future work.

II. Preliminaries

This section introduces some concepts related to offline e-cash system and the use of agent's technology in the e-cash.

A. Overview of Offline e-Cash

The prevalent model of the e-cash schemes involved three different parties, namely a bank, customers, and merchants. The life cycle of e-cash coin involves these three parties as given in Fig. 1. This life cycle begins when a customer withdraws the coin from the bank (withdrawal protocol). Then, the customer spends the coin by sending it to a merchant in trading for some goods or services (payment protocol). Finally, the merchant ends the cycle by depositing coin (deposit protocol). There are two types of e-cash schemes, namely, online and offline.

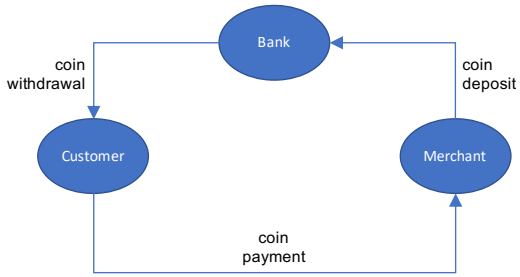
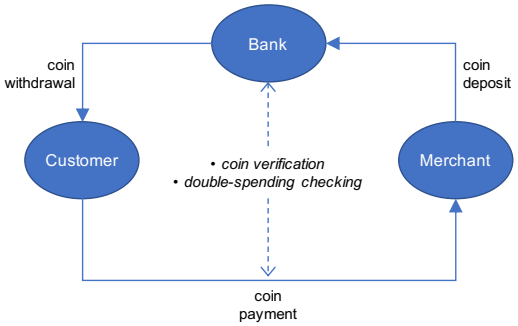


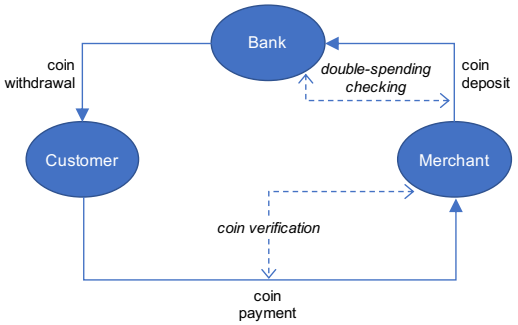
Fig. 1. The circulation of e-cash coin.

Both online and offline e-cash schemes perform two major verifications, namely coin's validity verification and identification of the coin's double-spending. In an online e-cash scheme [11], as illustrated in Fig. 2a, the merchant has to be online with the bank when performing both verifications on the payment protocol. The coin from the payment is accepted if the coin is valid and never used before. While in offline e-cash scheme [12], [13], [14], as illustrated in Fig. 2b, the merchant accepts a coin on payment protocol and subsequently verifies the validity of the coin without involving the bank. The bank identifies double-spending of a coin on deposit protocol.

The potential for dishonest customers to double-spend coins is higher in offline e-cash since the coins are not verified at the coin payment protocol. Therefore, an offline e-cash scheme must have a mechanism to prevent double-spending. Besides, e-cash also must be resistant to counterfeit coin and adversary users.



(a) Online e-cash



(b) Offline e-cash

Fig. 2. Coin verification in traditional e-cash

B. Agent's Technology in e-Cash

An agent is a computer program that is able to take independent action on behalf of its user or owner. Generally, the agent is categorised into two types: static and mobile agent. The static agent always stays in one place and performs the operations according to its intended purposes. The mobile agent is the one with mobility capability that allows it to migrate from one host to another to perform the operations for its owner.

Within the last decade, the paradigm of the agent system was discussed broadly. Many suggestions for future fields of application of the agent system have been made in a distributed system such as distributed database [15] and digital library [16]. In the e-cash system, the paradigm of the agent system address the challenges of communication bottleneck and resource ability of a user. There are two main research fields about e-cash and agent. The first field concern of agent is an e-commerce framework, including the notion of payment [17]. Furthermore, the possibility of an agent to carry and spend e-cash is the second field [10], [18].

The agent technology adoption in many fields led to sophisticated design and security threats. Since mobile agent migrates from one host to another, mobile agent more vulnerable than the static agent. Thus, the mobile agent platform protection is vital because it is an environment where a mobile agent gets executed. Mobile agent platform suffers security threats from a foreign agent that performs denial of service attack

and unauthorized access. Protection of mobile agent platform from a malicious agent can adopt various techniques. The first is sandbox technique which isolates untrusted agent so cannot alter the platform or agent in it [19]. Simple Malicious Identification Police (MIP) model [20] is the second technique that can be adopted to protect the mobile agent platform. The concept of this technique is identifying malicious agent by scanning the byte code of the agent.

III. Proposed Model

The proposed model is the MAPW model for preventing double-spending in offline e-cash scheme. This section gives an overview and description of the proposed model in detail.

A. Overview of Proposed Model

The MAPW model is intentionally designed as a coin's wallet with protection against malicious agent, counterfeit coin, and double-spent coin. Simple MIP model is adopted to protect MAPW against malicious agent. In order to overcome the counterfeit and double-spent coin, MAPW applies the autonomy and cooperation capabilities of the software agent.

The main idea of MAPW model is to append double-spending identification when receiving a coin thus can prevent double-spending in advance. Fig. 3 illustrates the proposed offline e-cash cycle used by MPAW. There are three parties: bank $\mathcal{B}$ that able to issue coins and accept deposited coins; customer $\mathcal{C}$ that can withdraw and spend coins; and merchant $\mathcal{M}$ that can accept spending coins and deposit coin. Our proposed model is composed of withdrawal protocol, payment protocol, and deposit protocol.

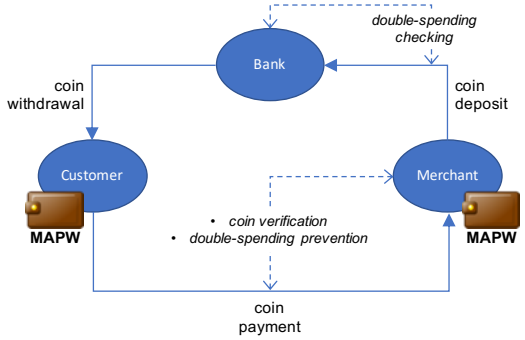


Fig. 3. The proposed offline e-cash cycle.

The simple description of these protocols is given as follows:

1) The withdrawal protocol: The customer $\mathcal{C}$ withdraws a coin c_i from $\mathcal{B}$. If $\mathcal{B}$ and $\mathcal{C}$ pass the challenge-response, $\mathcal{B}$ sends c_i to $\mathcal{C}$. Then, $\mathcal{C}$'s wallet verifies the signature of c_i and performs the synchronization checking.

2) The payment checking: The customer $\mathcal{C}$ spends a coin c_i to $\mathcal{M}$. At first, both of them perform challenge-response and c_i will be sent to $\mathcal{M}$ if the challenge-response is performed successfully. The $\mathcal{M}$'s wallet subsequently performs the verification of c_i's signature and the synchronization checking.

3) The deposit checking: The merchant $\mathcal{M}$ can deposit coin c_i to the bank $\mathcal{B}$. Before depositing c_i, $\mathcal{M}$ and $\mathcal{B}$ perform challenge-response. $\mathcal{B}$ allows $\mathcal{M}$ to send c_i if they pass the challenge-response. First, $\mathcal{B}$ verifies the signature of c_i and checks whether c_i has been previously deposited. If the signature of c_i is valid and c_i is never deposited before, $\mathcal{B}$ accepts c_i.

4) The synchronization checking: The purpose of the synchronization checking is preventing the wallet from receiving duplicate coin. The synchronization is performed when the wallet of $\mathcal{C}$ or $\mathcal{M}$ receives or sends a coin. The wallet checks the identity of the new coin whenever it receives a new coin. The new coin is accepted if there are no coins in the wallet that has the same identity as the new coin. Otherwise, the wallet refuses the new coin. Before the wallet sends a coin, the wallet checks the existence of coin's identity. If coin's identity exists, it allows the coin to migrate to another wallet.

B. Model Description

MAPW model, as shown in Fig. 4, has four static agents (like user, bank, identifier, and killer agent) and one mobile agent (coin agent). The static agents, with their respective duties and responsibilities, protect the mobile agent platform, and ensuring no counterfeit and double-spent coin. User agent performs three e-cash protocols (withdrawal, payment, and deposit), and is responsible for incoming and outgoing checking. Bank agent is an agent of the bank's representative that stores the identity of all coin agent in the wallet and identifies a duplicate agent. Identifier agent determines whether the foreign agent is a malicious agent or not. Killer agent kills any malicious agent, counterfeit coin, and double-spent coin.

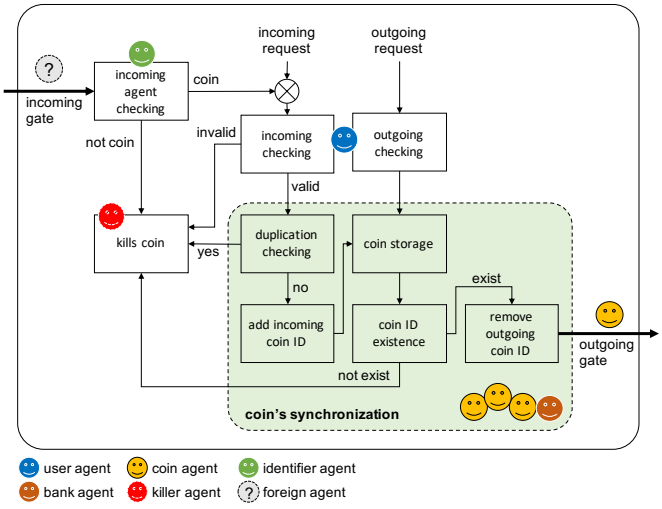


Fig. 4. The model of MAPW.

Like bank agent, coin agent stores the identity of all coin in the wallet and identifies the duplicate agent. The coin agent, as shown in Fig. 5, carries coin data like serial number, signature, origin, and sync ID. *Serial number* is a unique number that represents the identity of the coin. *Signature* is a bank's digital signature for verifying the authenticity of the coin. A flag that indicates whether the coin comes from a valid protocol or not is called *origin*. *Sync ID* is a memory space to store the identity

of the bank and other coin agents that are in the same wallet at the same time.

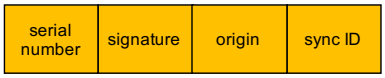


Fig. 5. The block data of coin.

C. The Function of Mobile Agent Platform based Wallet Model

The functional process algorithm of MAPW model involves two algorithms: the arrival and leaving of a coin that is respectively given in Algorithm 1 and Algorithm 2. In Algorithm 1, the arrival of a new agent triggers the identifier agent to identify the new agent. If the new agent is not identified as a coin agent, the identifier agent will trigger the killer agent to kill the new agent. Otherwise, the new agent will be considered as a new coin and forwarded to the user agent for verifying the new coin's signature. The new coin is allowed to broadcast its arrival to the bank agent and stored coins if its signature is valid, but the killer agent will kill it if its signature is invalid. After the bank agent and stored coins receive the broadcast message, they check the existence of the new coin ID in their memory of the stored coins' ID and send a kill command to the killer agent if its ID is a duplicate. However, the bank agent and stored coins save the new coin ID if the new coin is not a duplicate. The new coin also saves the bank agent and all stored coins' ID.

Algorithm 1 Algorithm for the arrival of agent/coin

```
if the agent is not a coin or request is not a valid incoming request then
    killer agent kills the agent and exit;
else
    the agent is considered as a new coin;
    user agent verifies the new coin's signature;
    if the signature of the new coin is invalid then
        sends a command to killer agent to kill the new coin;
    else
        the new coin broadcasts its arrival to stored coin and bank agent;
        the bank agent and stored coins check the new coin ID whether its already exist or not;
        if the ID of new coin is a duplicate of stored ID coin then
            the bank agent or stored coins send a kill command to killer agent;
        else
            the bank agent and stored coins save the new coin ID;
            the new coin ID saves all stored coin ID and bank agent;
        end if
    end if
end if
```

The leaving of a coin, as described in Algorithm 2, begins whenever the user agent accepts a valid outgoing request. The user agent sends a request to a coin for migrating to another wallet. The coin that accepts this request then broadcasts its migration to the bank agent and other coins. They delete the coin's ID from their memory and allow the coin's migration if the coin's ID is in their memory. Otherwise, they consider the coin as an invalid coin and trigger the killer agent to kill the coin.

Algorithm 2 Algorithm for the leaving of a coin

```
if request is a valid outgoing coin request then
    the user agent sends a request to the coin for migrating to another wallet;
    the coin broadcasts its migration;
    if the bank agent and other coins know the ID of the coin then
        other coins agent and bank agent delete ID of the coin;
        the coin migrates and deletes the ID of other coins;
    else
        killer agent kills the coin agent and exit;
    end if
else
    ignores request;
end if
```

IV. Colored Petri Nets Model of Mobile Agent Platform based Wallet

MAPW is the proposed model of double-spending prevention in offline e-cash. In order to determine the correctness and eliminating or minimizing the security of MAPW, it must be verified by using a formal method. There are various formal methods, but the most commonly used for the agent is Petri nets. For example, Petri nets can be used for modeling interaction protocol in multiagent system [21] and for verifying agent-based architecture [22]. This paper uses CPN, that is a combination of the capabilities of Petri nets and a high-level programming language, for the design, development, and analysis of MAPW [23].

TABLE I. Tested scenarios for MAPW model

Case	Agent type	Condition of coin			
		Signature	Legitimate origin	Duplicate	Known ID
malicious agent	not coin	-	-	-	-
counterfeit coin	coin	invalid ($sign$=0)	-	-	-
double spending	coin	valid ($sign$=1)	invalid ($orig$=0)	-	-
		valid ($sign$=1)	valid ($orig$=1)	yes	-
		valid ($sign$=1)	valid ($orig$=1)	no	no
normal spending	coin	valid ($sign$=1)	valid ($orig$=1)	no	yes

Table I shows a set of the tested scenario for MAPW model in proving the MAPW's protection against malicious agent, counterfeit coin, and double-spending. MAPW also should able to do normal spending. A malicious agent is an agent that its type is not a coin agent. A counterfeit coin is a coin with an invalid signature ($sign$=0). There are three possibilities of double-spending. First, a coin with a valid signature but not came from the legitimate origin (valid withdrawal or valid

payment). Second, coin with valid signature and came from legitimate origin but has a duplicate in the wallet. Third, a coin with a valid signature, came from legitimate and does not have a duplicate in the wallet, but its identity is not recognized. The last scenario is normal spending that is a coin with a valid signature, came from the legitimate origin, does not have any duplicate, and its identity is recognized by bank agent and coin agent.

The CPN model of MAPW consists of one main MAPW model page and four subpages for coin's generation, incoming coin, outgoing coin, and synchronization coin's ID. Fig. 6 illustrates the main MAPW model page in which accepts the incoming and outgoing request. Every time MAPW accepts incoming coin request, it will trigger coin generation to generate random coin, and this random coin will be checked by incoming coin checking. If the request is an outgoing coin request, it will be checked by outgoing coin checking.

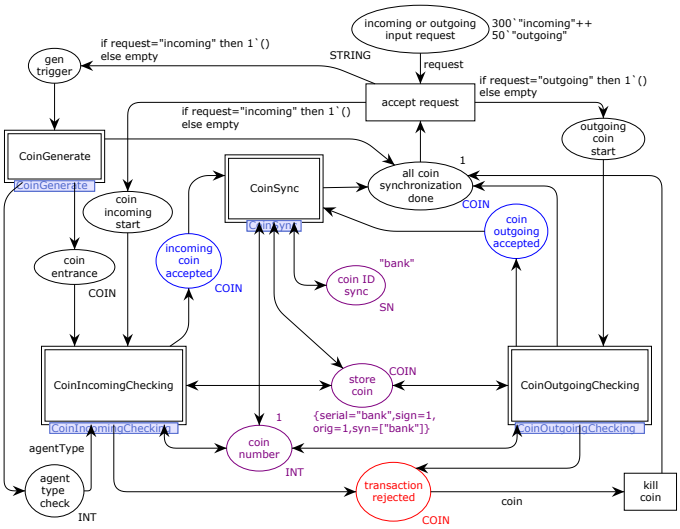


Fig. 6. Main CPN model of MAPW

There are four customized color types and five customized functions in the CPN model of MAPW. Fig. 7 shows the declaration of the four customized color types, namely, $FLAG$, SN, SYN, and $COIN$. $FLAG$ is an integer color type with a value of 0 or 1. SN is a string color type that represents serial number of a coin. SYN is a color type of the list of SN that represents the memory of a coin. $COIN$ is a record color type that consists of $serial$, $sign$, $orig$, and syn.

```
colset FLAG=int with 0..1;
colset SN=STRING;
colset SYN=list SN;
colset COIN=
   record serial:SN*sign:FLAG*orig:FLAG*syn:SYN;
```

Fig. 7. Declaration of CPN model color types

The five customized functions, as shown in Fig. 8, are $count()$, $notin()$, $serialVal()$, $boolVal()$, and $serialLabel()$. The $count()$ function is used for counting data in a list. The $notin()$ function is used for searching serial number of a coin in a list. The $serialVal()$ function returns a random integer value from 1 to 500. The $boolVal()$ function returns a random number of 0 or 1. The $serialLabel()$ function returns a random serial number of a coin.

```
fun count(synCoin:SYN)=
   if synCoin=[] then 0
   else 1+count(tl(synCoin));
fun notin(sn:SN,syn:SYN)=
   if syn=[] then true
   else if sn=hd(syn) then false
   else notin(sn,tl(syn));
fun serialVal()=
   discrete(1,500);
fun boolVal()=
   discrete(0,1);
fun serialLabel()=
   "serial"^Int.toString(serialVal());
```

Fig. 8. Declaration of CPN model function

A. CoinGenerate Subpage

The *CoinGenerate* subpage illustrated in Fig. 9 is the first subpage of MAPW's top-level CPN model and the CPN model of the identifier agent that is responsible for checking all incoming agent. This subpage performs the generation of a random incoming agent (coin and non-coin agent) that enters MAPW through the incoming gate, which is triggered by *gen trigger* place. Coin agent is represented by 1, while a non-coin agent is represented by 0. Random value 0 or 1 is generated by $boolVal()$ function. If a non-coin agent enters MAPW, it will be killed, and the model will return to wait for a request. However, if the incoming agent is a coin agent, it will generate a coin agent and send the coin agent to coin entrance.

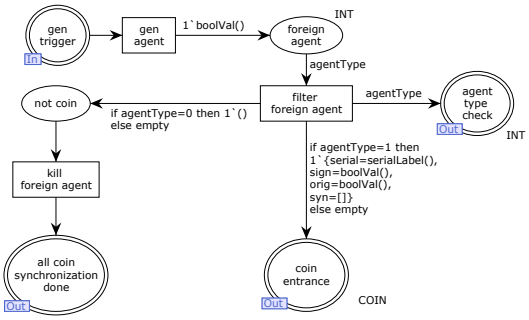


Fig. 9. CPN model of the *CoinGenerate* subpage

Coin data on this model uses $COIN$ color type $\{serial, sign, orig, syn\}$ that consists of $serial$ as the serial number or identity of the coin, $sign$ as the signature of the coin, $orig$ as the origin of the coins, and syn as a storage memory of all coins in the wallet. The value of $serial$, $sign$, and $orig$ are generated randomly in order to allow all conditions of the coin that enter MAPW either valid or invalid. The invalid coin is a duplicate, false signature or the entrance of coin without incoming request. $serialLabel()$ and $boolVal()$ function subsequently generates a random serial number of a coin, and random value of 0 or 1. The signature of the coin is valid if the value is 1 and invalid if the is 0.

B. CoinIncomingChecking Subpage

Fig. 10 illustrates the *CoinIncomingChecking* subpage in more detail. *Agent type check*, *coin incoming start*, and *coin entrance* are provided as input. *CoinIncomingChecking* will be executed if a coin agent enters MAPW that is marked by value 1 of *agentType*. The data of coin agent that enters *CoinIncomingChecking* is generated by *CoinGenerate* subpage. Then this model verifies signature and flag of the coin agent whether valid or not. If the signature and flag are valid, the coin will be considered as an incoming coin and start the scanning process. Otherwise, the transaction is rejected, and the incoming coin agent is going to be killed.

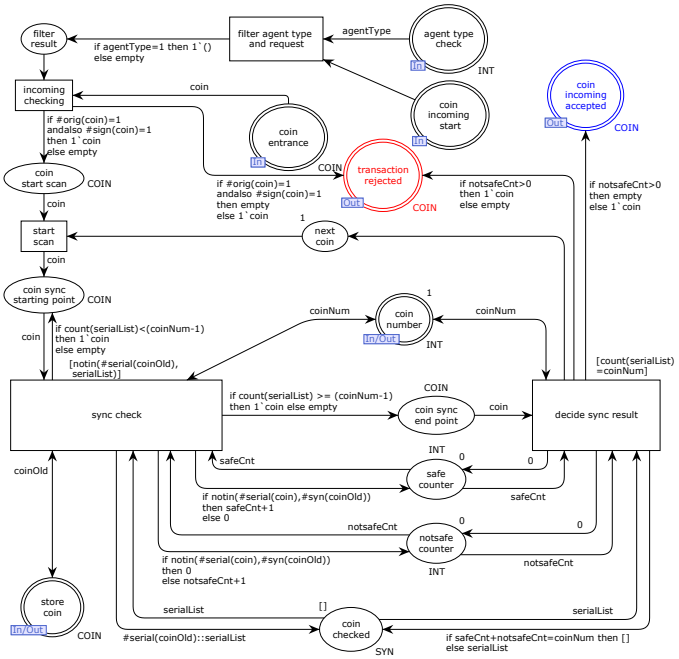


Fig. 10. CPN model of the *CoinIncomingChecking* subpage

The purpose of the scanning process in *CoinIncomingChecking* is as duplicate coin checker. All stored coin agent and bank agent in MAPW check the serial number of the new coin, and if they already have the serial number, then the new coin is rejected. Otherwise, the serial number of the new coin is fresh, the new coin is accepted and forwarded to *CoinSync* transition.

C. CoinSync Subpage

The function of *CoinSync* subpage is synchronizing the serial number of incoming and outgoing coin, as illustrated in Fig. 11. This function is triggered when *incoming coin accepted* or *outgoing coin accepted* fire a COIN token.

The synchronization of an accepted incoming coin begins with broadcasting the serial number of the incoming coin agent to bank agent and all stored coin agents. Bank agent and all stored coin agent add the serial number of the incoming coin agent to their *sync* variable. The incoming coin agent also adds the serial number of bank agent and all stored coin agent to its *sync* variable. Then incoming coin agent is accepted and adds the value of *coin number* place with 1.

The purpose of synchronization of an outgoing coin is removing the outgoing coin agent's serial number and decreasing *coin number* place value by 1. This synchronization starts by broadcasting the outgoing coin to all stored coins agent and bank agent to remove the serial number of the outgoing coin agent. Terminate synchronization and return to wait for a request.

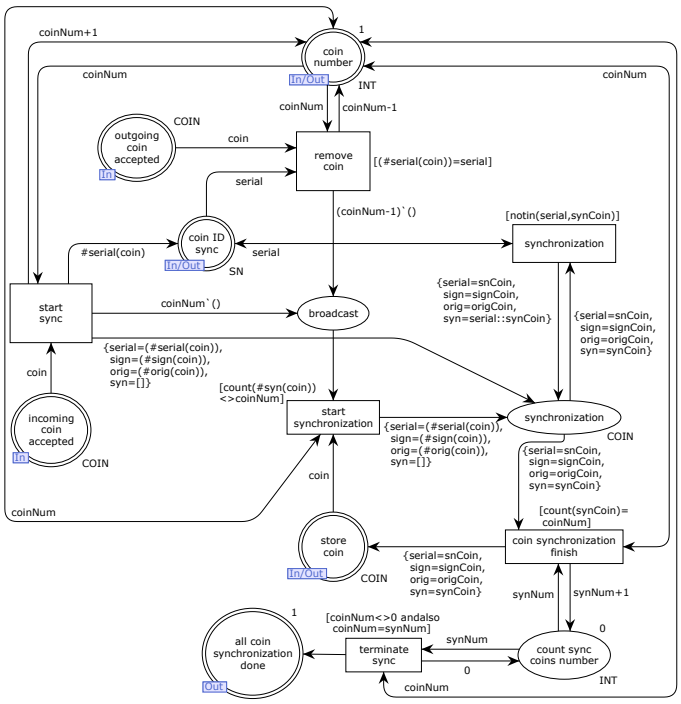


Fig. 11. CPN model of the *SynchronizationProcess* subpage

D. CoinOutgoingChecking Subpage

The *CoinOutgoingChecking* subpage, as illustrated in Fig. 12, serves outgoing request which asks for sending a coin agent to a new wallet. This subpage checks the availability and validity of an outgoing coin. The first checking, availability of coins checking, is performed to determine if there are coins stored in the wallet when receiving an outgoing request. If there are no coins, the outgoing request will be ignored. Oppositely, a coin will be called in order to send it to a new wallet.

The subsequent checking is the validity of coin checking, which verifies whether other coins agent and bank agent know the coin's serial number. If other coins agent and bank agent save the serial number of the coin in their sync variable, the coin is accepted as an outgoing coin and forwarded to CoinSync subpage. Otherwise, the outgoing request is rejected, and the coin is forwarded to *kill coin* transition in main CPN model.

V. Discussion

The CPN model of MAPW is verified by performing the state-space analysis that calculates all reducible states and state changes in order to observe the behavior of MAPW model, such as the nonexistence of loops and deadlocks; the

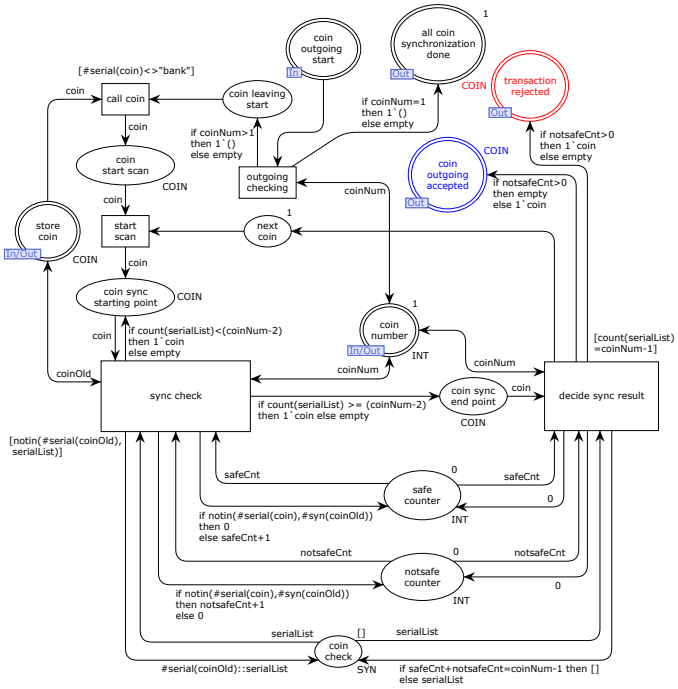


Fig. 12. CPN model of the *CoinOutgoingChecking* subpage

possibility of always being able to reach a certain state; and the delivery guarantee of a provided service. The state-space analysis results in Table II shows that there is no infinite occurrence sequence that indicates there is no loop in MAPW's model so that the termination of each module is guaranteed. There is no home marking, which is mean the impossibility to have an occurrence sequence that cannot be extended to reach the home marking. The state-space analysis detects the marking that has no enabled transition, which is called dead marking. Dead marking exists because not every coin leaves the wallet. Furthermore, the state-space analysis shows the absence of dead transitions and live transitions. The absence of dead transitions means that each transition has the possibility of occurring at least once while the absence of live transition means the transitions always occur in any condition.

TABLE II. STATE-SPACE ANALYSIS RESULTS OF MAPW MODEL

Property	Result
No infinite occurence sequence	None
Home markings	None
Dead markings	Yes
Dead transitions	None
Live transitions	None

In addition to state-space analysis, the CPN model of MAPW is also tested for its security (malicious agent, counterfeit coin, and double-spending) and functionality (normal-spending) by referring to scenarios in Table I. The result of tested scenarios for MAPW model (Tabel III) shows that the CPN model of MAPW passes all of the tested scenarios.

The properties of MAPW model is compared with two related works [9], [10] and the comparison can be seen in Table IV. Liu [9] and Salama [10] have forgery and double-spending

TABLE III. THE RESULT OF TESTED SCENARIOS FOR MAPW MODEL

Scenario	Result
malicious agent	detected and killed
counterfeit coin	detected and killed
double-spending	detected and killed
normal spending	pass

prevention, but they do not have any wallet protection. In order to perform double-spending prevention, Liu [9] uses smartcard and Salama [10] uses OMC and mobile agent. The MAPW model has both forgery and double-spending prevention. Double-spending prevention is performed by mobile agent without depending on the specific hardware. The MAPW model also has wallet protection that protects the wallet from malicious agents.

TABLE IV. THE COMPARISON OF PROPERTIES BETWEEN RELATED WORK AND PROPOSED MODEL

Property	Proposed model	Liu [9]	Salama [10]
wallet protection	yes	no	no
forgery prevention	yes	yes	yes
double-spending prevention	yes	yes	yes
card based	no	yes	yes
mobile agent based	yes	no	yes

VI. CONCLUSION

The offline e-cash is vulnerable of double-spending because the bank stay offline while the merchant accepts a coin anonymously from the customer. The merchant only checks the validity of the coin's signature when accepts the coin, but the merchant cannot determine whether the coin is a double-spent coin or not. The bank verifies double-spending after the transaction. This paper proposes the MAPW for offline e-cash that has been modeled, analyzed, verified, and tested by using CPN and tested scenarios. The result shows that the MAPW is able to prevent double-spending, protect wallet against the malicious agent and counterfeit coin.

VII. FUTURE WORK

There are various issues in offline e-cash which must be addressed in the future, such as the protection of coin against malicious host problem and the application of MAPW to offline transferable e-cash. The transferability of e-cash is a challenging problem because it has more aspect to consider including more user, and the coin grows in size.

REFERENCES

[1] S. Singh, "Emergence of payment systems in the age of electronic commerce: The state of art," in *2009 First Asian Himalayas International Conference on Internet*, November 2009, pp. 1–18.

[2] D. Chaum, "Blind signature for untraceable payment," in *CRYPTO '83 Proceedings on Advance in Cryptology*. New York: Plenum Press, 1983, pp. 199–203.

[3] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," in *CRYPTO '88 Proceedings on Advances in Cryptology*. New York: Springer-Verlag, 1988, pp. 319–327.

[4] S. Canard and A. Gouget, "Anonymity in transferable e-cash," in *Applied Cryptography and Network Security*, S. M. Bellovin, R. Gennaro, A. Keromytis, and M. Yung, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 207–223.

[5] C. I. Fan and V. S. M. Huang, "Provably secure integrated on/off-line electronic cash for flexible and efficient payment," *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 5, pp. 567–579, September 2010.

[6] O. Blazy, S. Canard, G. Fuchsbauer, A. Gouget, H. Sibert, and J. Traore, "Achieving optimal anonymity in transferable e-cash with a judge," *AFRICACRYPT*, pp. 206–223, July 2011.

[7] J. Zhang, H. Guo, Z. Li, and C. Xu, "Transferable conditional e-cash with optimal anonymity in the standard model," *IET Information Security*, vol. 9, no. 1, pp. 59–72, December 2015.

[8] S. Nakamoto, "A peer-to-peer electronic cash system," http://www.bitcoin.org/bitcoin.pdf, 2009.

[9] W. Y. Liu, Y. A. Luo, and Y. L. Si, "A security multi-bank e-cash protocol based on smart card," in *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics*. IEEE, August 2007, pp. 3244–3248.

[10] M. A. Salama, N. El-Bendary, and A. E. Hassanien, "Towards secure mobile agent based e-cash system," in *Proceedings of the First International Workshop on Security and Privacy Preserving in e-Societies*, New York, 2011, pp. 1–6.

[11] S. H. Islam, R. Amin, G. P. Biswas, M. S. Obaidat, and M. K. Kan, "Provably secure pairing-free identity-based partially blind signature scheme and its application in online e-cash system," *Arabian Journal for Science and Engineering*, vol. 41, no. 8, pp. 3163–3176, August 2016.

[12] X. Zhou, "Threshold cryptosystem based fair off-line e-cash," in *Second International Symposium on Intelligent Information Technology Application*, vol. 3. Shanghai: IEEE, 2008, pp. 692–696.

[13] W.-S. Juang, "An efficient and practical fair buyer-anonymity exchange scheme using bilinear pairing," in *2013 Eight Asia Joint Conference on Information Security*, 2013, pp. 19–26.

[14] C. Wang, H. Sun, H. Zhang, and Z. Jin, "An improved off-line electronic cash scheme," in *International Conference on Computational and Information Sciences*. IEEE, 2013, pp. 438–441.

[15] F. U. Ogban and U. Udoh, "A mobile agent-based distributed information retrieval system," *International Journal of Natural and Applied Sciences*, vol. 10, pp. 72–77, 01 2015.

[16] G. Liu, "The application of intelligent agents in libraries: a survey," *Program: Electronic Library & Information Systems*, vol. 45, no. 1, pp. 78–97, 2011.

[17] S. U. Guan, S. L. Tan, and F. Hua, "A modularized electronic payment system for agent-based e-commerce," *Journal of Research and Practice in Information Technology*, vol. 36, no. 2, pp. 67–87, May 2004.

[18] C. Anhalt and S. Kirn, "Towards payment systems for mobile agents," in *Proceedings of the 4th European Workshop on Multi-Agent Systems*, B. Dunin-Keplicz, A. Omicini, and J. Padget, Eds., vol. 223. CEUR, December 2006.

[19] R. Wahbe, S. Lucco, T. Anderson, and S. Graham, "Efficient software-based fault isolation," *ACM SIGOPS Operationg Systems Review*, vol. 27, no. 5, pp. 203–216, 1993.

[20] S. Venkatesan and C. Chellappan, "Protection of mobile agent platform through attack identification scanner (ais) by malicious identification police (mip)," in *2008 First International Conference on Emerging Trends in Engineering and Technology*, July 2008, pp. 1228–1231.

[21] B. Marzougui and K. Barkaoui, "Interaction protocols in multi-agent systems based on agent petri nets model," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 7, pp. 166–173, 2013.

[22] N. A. Mian and F. Ahmad, "Agent based architecture for modeling and analysis of self adaptive systems using formal methods," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 1, pp. 563–567, 2018.

[23] K. Jensen and L. M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*, 1st ed. Springer Publishing Company, Incorporated, 2009.