Model-Based Analysis of Money Accountability in Electronic Purses*

Il-Gon Kim[1], Young-Joo Moon[1], Inhye Kang[2], Ji-Yeon Lee[1], Keun-Hee Han[1], and Jin-Young Choi[1]

[1] Dept. of Computer Science and Engineering, Korea University, Seoul, Korea
{igkim, yjmoon, jylee, khhan, choi}@formal.korea.ac.kr
[2] Dept. of Mechanical and Information Engineering, University of Seoul, Seoul, Korea
inhye@uos.ac.kr

Abstract. The Common Electronic Purse Specifications (CEPS) define requirements for all components needed by an organization to implement a globally interoperable electronic purse program. In this paper we describe how we model purchase transaction protcol in CEPS using formal specification language. We define and verify the money accountability property of the CEPS, and we address its violation scenario in the presence of communication network failures. Using model checking technique we find that transaction record stored in the trusted-third party plays a essential role in satisfying the accountability property.

Keywords: Formal specification and verification, security, e-commerce protocol, CEPS, model checking, money accountability, Casper, FDR.

1 Introduction

The use of smart cards as electronic purses and smart credit card/debit cards is increasing the market potentiality of electronic commerce. The technology for secure and stable electronic payment cards is being driven by big companies. Visa, Proton and a number of European financial institutions have collaborated to create the Common Electronic Purse Specifications (CEPS)[1], with the purpose of having some degree of international interoperability in all national purse schemes.

One of the most important requirements in electronic commerce protocols is to ensure the accountability on electronic transactions. *Money accountability* means that money is neither created nor destroyed in the process of an electronic transaction between customer and merchant. For example, a customer loads an e-money on smart card from the bank, and attempts to use the coin to pay a merchant. Unfortunately, communication networks fails in the transit of e-money from the customer to the merchant. In this situation, the consumer cannot be convinced that the e-money has been spent correctly. This is a critical

* This work was supported by Korea Research Foundation Grant (KRF-2003-003-D00397).

issues for money accountability, because the customer and the merchant can be susceptible to dispute between transacting parties. In other words, without adequate accountability assurance in CEPS, there would be no means to settle legal disputes against fraudulent individuals.

In this paper we address the accountability problem in the CEPS, and we verify it with model checking technique. First, we model accountability property and CEPS process in CSP (Communicating Sequential Processes) process algebra language, respectively. Second, the property model is expressed as *SPEC*, and the CEPS is modelled as *SYSTEM*. Lastly, we use the FDR (Failure Divergence Refinement) model checking tool. If *SYSTEM* is a refinement of *SPEC*, the set of behaviors generated by *SYSTEM* is a subset of those generated by *SPEC*. It means that the CEPS satisfies the accountability property.

The remainder of this paper is organized as follows. Section 2 describes some related work. Section 3 gives a brief overview of the CEPS. Section 4 briefly describes model checking technique, and describes how the CEPS and the accountability property can be modelled and verified using CSP and FDR. Finally, section 5 concludes this paper.

2 Related Work

There has been many research in using model checking to verify the security aspects of protocols. However, relatively little research has been carried out on formal analysis of accountability in e-commerce protocols.

Il-Gon Kim et al.[5] used the FDR model checker to verify the secrecy and authentication properties of a m-commerce protocol.

Heintze et al.[3] researched on non-security properties such as money atomicity and goods atomicity of two e-commerce protocols(NetBill[2] and Digicash[4]).

Indrakshi Ray et al.[7] proposed mechanisms that desirable properties addressed in [3] are still preserved in a fair-exchange protocol despite network failures.

Jan J*ü*rjens et al.[6] investigated the security aspect of the CEPS using AUTOFOCUS tool. The author modelled CEPS with several diagrams which supported by the AUTOFOCUS tool. Then he added intruder model in order to intercept messages and learn secrets in the messages. Finally, the author used the AUTOFOCUS connection to the model checker SMV.

In this paper we specify the behaviour of the CEPS purchase transaction protocol in presence of network failures, using model checking technique proposed in [3][7]. We focus on verifying the accountability property in the CEPS, not in the viewpoint of security properties such as confidentiality and authentication. So far, there is no research to specify and verify the accountability of the CEPS using model checking. In this regard our work is different from the work of Jan J*ü*rjens et al.

3 CEPS

In this section we give a brief overview of the Common Electronic Purse Specification and describe its purchase transaction protocol.

The Common Electronic Purse Specification (CEPS) was proposed by Visa, Proton and a number of European financial institutions, in order to have a common standard for all national purse schemes. Currently, stored value smart cards(called "electronic purses") have been proposed to allow an electronic purse application which conforms to CEPS. It is designed as a standard for 72 different e-purse systems worldwide to work together.

Herein we address the central function of CEPS, the purchase transaction, which allows the cardholder to use e-money on a smart card to pay for goods. In this paper we assume that the communicating participants in the purchase transaction protocol should consist of three part; customer's smart card, PSAM in the merchant's POS (Point Of Sale) device, and bank. The POS device embeds Purchase Security Application Module (PSAM), which is used to store and process data transferred from cardholders. The bank plays a role of checking the validity of customer's card and account.

3.1 Purchase Transaction Protocol

The PSAM starts the protocol (see Fig.1) after the smart card(CARD) is inserted into the POS device, by sending a debit request to the card. The debit request message contains an e-money amount for a deal. In this paper we focus on addressing the accountability property, not security properties. For the detailed notation and meaning of each data and key used in messages, the reader can see [1]. At step 2, the CARD responds with the message containing e-money singed by the customer. The message of purchase response includes the e-money amount signed by the customer. Then the PSAM forwards the customer's electronic payment order (EPO) to the BANK. At step 4, the BANK checks the validity of this EPO and sends to the PSAM a receipt of the fund transfer. Finally, if the electronic transaction between the CARD and the PSAM has completed successfully, then the PSAM sends 'transaction completed' message to the CARD.

1. PSAM $\rightarrow$ CARD : debit request
2. CARD $\rightarrow$ PSAM : purchase response
3. PSAM $\rightarrow$ BANK : endorsed signed EPO
4. BANK $\rightarrow$ PSAM : signed receipt
5. PSAM $\rightarrow$ CARD : transaction completed

Fig. 1. Purchase transaction protocol of the CEPS

4 CSP Specification and FDR Model Checking of the Purchase Transaction Protocol

In this section we describe how we model and verify the purchase transaction protocol and the accountability property using CSP/FDR.

We use CSP process algebra language to encode communicating channel and each participant. A CSP process denotes a set of sequences of events, where an event represents a finite state transition. A CSP process may be composed of component processes which require synchronization on some events. In this paper we encode communicating channel, the customer, the merchant, and the bank processes in CSP.

In FDR model checking tool, the refinement model is described as a process, say SYSTEM, and the property model is specified as another process, say SPEC. If SYSTEM is a refinement of SPEC, it means that the set of the possible behaviors of SYSTEM is a subset of the set of possible behaviors of SPEC. This relationship is written SPEC $\sqsubseteq$ SYSTEM in FDR. If SYSTEM is not a refinement of SPEC, the FDR tool generates a counter-example that describes under what scenario the property is violated. In the following subsections, SYSTEM represents the purchase transaction protocol and SPEC means the accountability property.

4.1 Communication Environment Process

We model asynchronous communication channel which proposed in [3][7], instead of using synchronous channel, because the asynchronous communication is useful to consider a network failure in the CSP model.

Fig.2 shows logical communication channel in the purchase transaction protocol. For example, the customer uses two communication processes(*COMMcp* and *COMMcb*) in order to communicate with the merchant and the bank, respectively. For example, the process *COMMcp* reads a data from channel *coutp*(which is connected to the channel *coutp* in the CARD) and writes the data to channel *pinc* and the process *COMMpc* reads a data from channel *poutc* and writes it to channel *cinm*. In a failure network, the transaction data between the customer and the merchant may be lost. To reflect on this unreliable communication channels, we model that *COMMcp* and *COMMpc* could lose sending or receiving data non-deterministically.

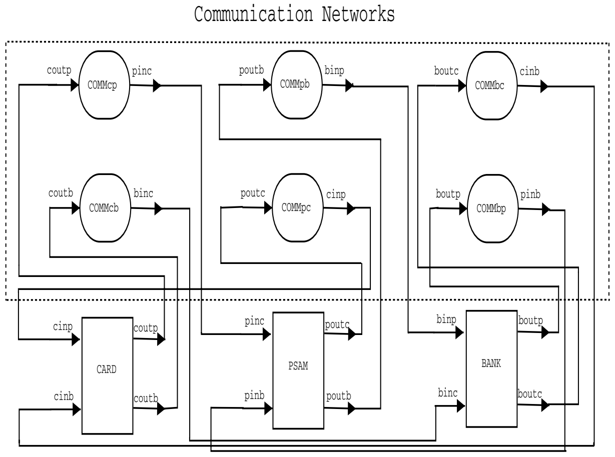

Fig. 2. Logical communication channel in purchase communication protocol

4.2 The CARD Process

We model the CARD process could load e-money from the BANK process if the customer's account holds enough balance. Then the CARD waits for the debit request(*DebitA* or *DebitB*) from the PSAM. If the CARD receives *DebitA* or *DebitB* correctly, it sends electronic payment order(*epo_TokenA* or*epo_TokenB*) to the PSAM. In the model we allow the CARD to send two different electronic payment orders for one purchase transaction, because this will be considered evidence to detect fradulent double deposit of the merchant. Because the CEPS provides on-line and off-line communication between the CARD and the PSAM, the following scenarios could be possible. As the customer's response to the debit request is in transit, the communication network fails. Thus the customer is left in an uncertain situation whether the e-money has been spent correctly or not. This is a critical issue for money accountability, because the merchant could be accused of double deposit. To confirm the accountability, we allow the customer to go to the bank and wait for the arbitration decision based on the transaction record. To reflect on this scenario, we model that the card process sends second

```
CARD = STOP |~| LOAD_CARD
LOAD_CARD = coutb!loadtoken ->
            cinb?x ->
            if x==token then loaded_token -> (USE_TOKEN [] KEEP_TOKEN)
            else STOP

USE_TOKEN = cinp?x -> DEBIT_CARD(x)
KEEP_TOKEN = cKeepsToken -> STOP

DEBIT_CARD(x) = if (x==DebitA) then
                (coutp!epo_TokenA -> EPO_TOKEN_SENT)
                else if (x==DebitB) then
                (coutp!epo_TokenB -> EPO_TOKEN_SENT)
                else RETURN_TOKEN

RETURN_TOKEN = coutb!paymentAlready ->
             cinb?x ->
             (if (x==refundSlip) then REFUND_RECEIVED
             else if (x==depositSlip) then epo_tokenSpent -> ARBITRATION
             else ERROR_DEBIT)

REFUND_RECEIVED = STOP
EPO_TOKEN_SENT = (cinp?y ->
                 (if (y==ok) then epo_tokenSpent -> (END |~| USE_TOKEN)
                 else if (y==no) then STOP
                 else RETURN_TOKEN))
               [] USE_TOKEN
               [] (timeoutEvent -> RETURN_TOKEN)
```

Fig. 3. The CARD process

EPO(*epo_TokenB*) again, if the response from the merchant is not received in a specified time after sending first EPO(*epo_TokenA*) to the merchant.

The CARD process's main behaviours could be summarized as follows:

– use token(*USE_TOKEN*) : the customer may use a token for purchasing a good
– load token(*LOAD_CARD*) : the customer may load a token from the bank
– token holding(*KEEP_TOKEN*) : the customer may keep a token for future use, not spending it to purchase goods
– token return(*RETURN_TOKEN*) : the consumer may return token to the bank for refund, in case of incorrect transaction.

4.3 The PSAM Process

The PSAM process starts with sending the debit request message(*DebitA* or *DebitB*) to the CARD; the *SEND_DEBIT_A* and *SEDN_DEBIT_B* states represent this step. Then the PSAM waits for the response(*epo_TokenA* or *epo_TokenB*)

```
PSAM = STOP |~| REPEATED_DEBIT_REQUEST(none)

REPEATED_DEBIT_REQUEST(previousDebitRequest) =
if (previousDebitRequest == none) then (SEND_DEBIT_A [] SEND_DEBIT_B)
else if (previousDebitRequest == DebitA) then SEND_DEBIT_B
else SEND_DEBIT_A

SEND_DEBIT_A =  (poutc!DebitA -> WAIT_FOR_RESPONSE(epo_TokenA))
SEND_DEBIT_B =  (poutc!DebitB -> WAIT_FOR_RESPONSE(epo_TokenB))

WAIT_FOR_RESPONSE(epo_Token) = pinc?x ->
if (x==epo_Token) then (mGets_epoToken->FORWARD_EPO_TO_BANK(epo_Token))
else poutc!bad_epoToken -> NO_TRANSACTION

NO_TRANSACTION = STOP
FORWARD_EPO_TO_BANK(epo_Token)=poutb!epo_Token -> WAIT_FOR_BANK(epo_Token)

WAIT_FOR_BANK(epo_Token) =
(pinb?x -> (if x==depositSlip then M_MAY_BE_FRAUD(epo_Token)
           else if x==refundSlip then (mGetsRefundSlip -> STOP)
           else if x==alreadyDeposited then FRAUD_DISCOVERED
           else if x==badBalance then poutc!no -> STOP
           else STOP))

FRAUD_DISCOVERED = STOP
M_MAY_BE_FRAUD(epo_Token) = END |~| FORWARD_EPO_TO_BANK(epo_Token)|~|
                            REPEATED_DEBIT_REQUEST(epo_Token)  |~|
                            poutc!ok -> STOP
```

Fig. 4. The PSAM process

from the CARD. After receiving the EPO (Electronic Payment Order) token, the PSAM sends it to the BANK in order to check the validity and the balance of the cardholder. If the payment token from the customer has been settled successfully, the merchant will get a deposit slip; denoted by *depositSlip*. When a merchant attempts to deposit a coin twice, the PSAM process receives *alreadyDeposited* data from the bank.

Herein the *M_MAY_BE_FRAUD* state is used to trace the fraud when a merchant attempts to deposit a coin twice for one purchase transaction.

4.4 The Bank Process

The most important function in the BANK process plays a role in recording logs about an electronic transaction in order to guarantee the accountability. If the BANK decides that an EPO token of the customer is valid, then it debits the

```
 BANK =  binc?x -> (if (x==loadtoken) then
                  (debitC -> boutc!token -> RECORD_LOG(0,0,0))
                  else STOP)

 RECORD_LOG(Flag, A, B) =
  binc?x -> (if (x==paymentAlready) then
           (if (Flag==0) then
              (creditC -> boutc!refundSlip -> RECORD_LOG(1,0,0))
            else if (A==1 or B==1) then
             (arbitration -> boutc!depositSlip -> RECORD_LOG(Flag, A, B))
            else RECORD_LOG(Flag, A, B))
           else RECORD_LOG(Flag, A, B))
[] binp?x ->(if (x==epo_TokenA) then
           (if (Flag==0) then
              (creditM -> boutp!depositSlip -> RECORD_LOG(1,1, B))
            else if (B==0 and A==1) then
              (boutp!alreadyDeposited -> mFraud -> RECORD_LOG(Flag, A, B))
            else if (B==1 and A==1) then
              (creditM -> boutp!depositSlip -> RECORD_LOG(Flag, A, B))
            else if (A==1 and B==1) then STOP
            else (arbitration -> boutp!refundSlip -> RECORD_LOG(Flag, A, B)))
          else if (x==epo_TokenB) then
            if (Flag==0) then
              (creditM -> boutp!depositSlip -> RECORD_LOG(1,A,1))
            else if (A==0 and B==1) then
              (boutp!alreadyDeposited -> mFraud -> RECORD_LOG(Flag, A, B))
            else if (A==1 and B==0) then
              (creditM -> boutp!depositSlip -> RECORD_LOG(Flag, A, B))
            else if (A==1 and B==1) then STOP
            else arbitration -> boutp!refundSlip -> RECORD_LOG(Flag, A, B)
         else RECORD_LOG(Flag, A, B))
```

Fig. 5. The BANK process

balance of the card and credits the account of the merchant(denoted by events *debitC* and *creditM*). In addition, the BANK process can settle out arbitration state caused by fraud of a customer or a merchant. For example, after a merchant gets a deposit slip by finishing a successful transaction with a customer, he/she may try to deposit an e-money twice. In this situation, the BANK process warns that the merchant has already deposited and it may be fraud by the merchant(shown by *boutp!alreadyDeposited* and *mFraud* events).

4.5 Failure Analysis of Money Accountability Property

Accountability is one of the most critical requirements to electronic commerce protocols and it can be divided into two categories; money accountability and goods accountability.

Money accountability property means that money is neither created nor destroyed in the steps of an electronic commerce transaction[3]. *Goods accountability* property represents that a merchant receives payment if and only if the customer receives the goods[3].

In this paper, we do not deal with failure analysis of goods accountability property because goods delivery is not included in the CEPS. Money accountability in the CEPS could be considered in the viewpoint of a customer and a merchant. Customer's money accountability property may be defined as the following trace specification.

```
SPECc = STOP |~| (loaded_token ->
                               ((epo_tokenSpent -> STOP) |~|
                               (cKeepsToken -> STOP) |~|
                               (creditC -> STOP)))
```

Customer's money accountability written in CSP means that once e-money is loaded into a smart card, the customer may choose keep it for future use, spend it for purchasing goods, or return it for refund. After using FDR tool, we found that customer's money accountability property is satisfied in the spite of unreliable communication networks and merchant's fraudulent behaviour.

```
SPECm = STOP |~| (mGets_epoToken ->
                               ((creditM -> STOP) |~|
                               (mGetsRefundSlip -> STOP)
```

Merchant's money accountability described in CSP represents that once e-money is transferred into the PSAM, the merchant's account balance may be incremented and the merchant may get a refund slip for incorrect transaction. When we run FDR tool, it shows the following counterexample which represents that merchant's money accountability property(*SPECm* process) may be violated due to a merchant fraud.

```
coutb.loadtoken, binc.loadtoken, debitC, boutc.token, cinb.token,
loaded_token, poutc.DebitB, cinp.DebitB, coutp.epo_TokenB,
pinc.epo_TokenB, mGets_epoToken, poutb.epo_TokenB, binp.epo_TokenB,
creditM, boutp.depositSlip, pinb.depositSlip, poutc.DebitA,
cinp.DebitA, coutp.epo_TokenA, pinc.epo_TokenA, mGets_epoToken,
boutp.alreadyDeposited, mFraud
```

This sequence of CSP events may show the scenario where a merchant attempts to deposit an e-money twice for a deal with a customer. After the customer sends an e-money for a good price to the merchant(shown by *coutp.epo_TokenB* event), communication network fails between the customer and the merchant. The merchant finishes a transaction with a bank and the account balance of the merchant is incremented(*poutb.epo_TokenB*, *creditM*, and *pinb.depositSlip*). Then the customer sends another e-money again (*coutp.epo_TokenA*) to the merchant because the customer can't confirm the transaction result due to the network failure. At this moment a malicious merchant attempts to use the customer's e-money again by sending it to the bank.

However, the merchant's fraud behaviour can be detected by the transaction record in the bank. Therefore, above counterexample doesn't mean a violation of the merchant's money accountability property. When we modify the *SPECm* process to contain at least one fraud event *mFraud*, we confirm that the CEPS model satisfies the merchant accountability property. This result means that the merchant fraud may happen in the e-commerce transaction based on the CEPS, by tampering of a POS device and man-in-the-middle attack. However the risk of the merchant accountability violation against the merchant fraud could be solved through comparing the signed transaction log information stored in the card and the bank.

5 Conclusion

In this paper we analyzed the money accountability properties of the Common Electronic Purse Specification (CEPS) using model checking approach. We have also shown how model checking using CSP and FDR can be used to describe e-commerce transaction in the CEPS including communication failure networks and detect the violation scenario of the accountability properties.

In our modelling of the CEPS we have abstracted away the purchase transaction of the CEPS by focusing on the non-security aspect. We have found that the risk of customer security against the merchant may bring about the cause to violate the money accountability. In addition, we have also identified that the transaction record in the trusted-third party such as the bank could provide the most effective solution to guarantee the accountability in the e-commerce protocol.

In the future we plan to analyze accountability properties of the Load Security Application Module (LSAM) which plays a role in loading e-money into the card.

References

1. CEPSCO, Common Electronic Purse Specification, Business Requirements vers. 7.0, Functional Requirements vers. 6.3, Technical Specification vers.2.2, available from http://www.cepsco.com, 2000.
2. B. Cox, J.D.Tygar, and M. Sirbu, "NetBill Security and Transaction Protocol", In *Proceedings of the First USENIX Workshop in Electronic Commerce*, pp.77-88, July 1995.
3. N. Heintze, J. Tygar, J. Wing, and H. Wong, "Model Checking Electronic Commerce Protocols", In *Proceedings of the 2nd USENIX Workshop in Electronic Commerce*, pp.146-164, November 1996.
4. A. Flat, D. Chaaum, and M. Naor, "Untraceable Electronic Cash", In Advances in Cryptography -*Proceedings of CRYPTO'88*, pp.200-212, Springer-Verlag, 1990.
5. I.G. Kim, H.S. Kim, J.Y. Lee, and J.Y. Choi, "Analysis and Modification of ASK Mobile Security Protocol", Proceedings of the Second IEEE International Workshop on Mobile Commerce and Services, pp.79-83, July 2005.
6. Jan Jürjens, Guido Wimmel, "Security Modelling for Electronic Commerce: The Common Electronic Purse Specifications", *Proceedings of the IFIP conference on towards the E-society*, pp.489-506, 2001.
7. I. Ray, I. Ray, "Failure Analysis and E-commerce Protocol using Model Checking", Proceedings of the Second International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems, pp.176-183, June 2000.