Formal Analysis of an Anonymous Fair Exchange E-Commerce Protocol

Weiqiang Kong[1], Kazuhiro Ogata[1,2], Jianwen Xiang[1], Kokichi Futatsugi[1]
[1]*Japan Advanced Institute of Science and Technology(JAIST)*
weiqiang, jxiang, kokichi@jaist.ac.jp
[2]*NEC Software Hokuriku, Ltd.*
ogatak@acm.org

Abstract

Fair exchange and anonymity are important requirements of e-commerce protocols. We have formally analyzed an e-commerce protocol, which is claimed to satisfy the two requirements. The protocol, together with the intruder, has been modeled as an OTS, a kind of transition system. Then the OTS has been written in CafeOBJ, an algebraic specification language. Although most part of the two requirements can be expressed as safety properties, liveness properties are needed to fully express them. We have expressed the safety part of the two requirements in CafeOBJ and partly verified that the OTS satisfies the safety part by writing proof scores in CafeOBJ.

1. Introduction

Fair exchange and anonymity are important requirements of e-commerce protocols. Fair exchange ensures that either each of a customer and a merchant involved in an e-commerce transaction obtains the other's item (either goods or money), or neither of them does; Customer's anonymity ensures that the real identity of any customer is not revealed during an e-commerce transaction.

Several e-commerce protocols have been proposed [1, 2], which are claimed to satisfy fair exchange and/or customer's anonymity. Among them is an anonymous fair exchange e-commerce protocol[1]. The designers analyze this protocol in an informal way without considering any intruders and conclude that the protocol satisfies both requirements. Informal analysis is not enough especially for security protocols among e-commerce protocols, which has been demonstrated by a number of researches[3, 7]. That is why we have formally analyzed the protocol with the OTS/CafeOBJ method[8, 9] to show that it does satisfy both requirements.

The anonymous fair exchange e-commerce protocol, together with the most general intruder à la Dolev and Yao[4], has been modeled as an OTS[9] (Observational Transition System), a kind of transition system. The OTS has then been written in CafeOBJ[5, 6], an algebraic specification language. Although most part of fair exchange and customer's anonymity can be expressed as safety properties, liveness properties are needed to fully express them. We have expressed the safety part of the two requirements in CafeOBJ and partly verified that the OTS satisfies the safety part by writing proofs or proof scores in CafeOBJ.

In this paper, we describe the case study, especially focusing on how to model the protocol, together with the intruder, as an OTS; and how to write the OTS in CafeOBJ; and how to express the safety part of the two requirements in CafeOBJ. We do not describe how to write proof scores in CafeOBJ in detail (refer to [9] for the details).

The rest of the paper is organized as follows. Section 2 outlines the OTS/CafeOBJ method. Section 3 describes an abstract version of the protocol. Section 4 describes the formal model of the protocol. Section 5 describes how to express the safety part of the two requirements and mentions their verification. Section 6 concludes the paper.

2. The OTS/CafeOBJ Method

CafeOBJ can be used to specify abstract machines as well as abstract data types. A visible sort denotes an abstract data type, while a hidden sort the state space of an abstract machine. There are two kinds of operations to hidden sorts: action and observation operations. An action operation can change the state of an abstract machine. Only observation operations can be used to observe the inside of an abstract machine. An action

0-7695-2216-5/04 $20.00 © 2004 IEEE

p1.	M	$\rightarrow$	TP	:	$\varepsilon_{K'}$(goods), $S_M(cc(\varepsilon_{K'}$(goods))), ε_{TP}(K'), $S_M(cc(\varepsilon_{TP}$(K')))
p2.	TP	$\rightarrow$	C	:	$\varepsilon_{K'}$(goods), $S_{TP}(cc(\varepsilon_{K'}$(goods)))
m1.	C	$\rightarrow$	M	:	PO, $S_{ic}(cc$(PO)), ε_M(Cipub)
m2.	M	$\rightarrow$	C	:	$S_M(cc$(PO)),ε_K(goods), $S_M(cc(\varepsilon_K$(goods))), ε_{MB}(Macct), $S_M(cc(\varepsilon_{MB}$(Macct)))
m3.	C	$\rightarrow$	CB	:	$\varepsilon_{CB}(S_C$(MTI))
m4.	CB	$\rightarrow$	C	:	$\varepsilon_C(S_B$(P))
m5.	C	$\rightarrow$	M	:	$\varepsilon_M(S_B$(P))
m6.	M	$\rightarrow$	MB	:	$\varepsilon_{MB}(S_B$(P))
m7.	MB	$\rightarrow$	M	:	S_{MB}(ack)
m8.	M	$\rightarrow$	C	:	ε_{ic}(K), $S_M(cc$(K))

Figure 1: The simplified protocol

operation is basically specified with equations by describing how the value of each observation operation changes. Declarations of observation and action operations start with **bop**, and those of other operations with **op** or **ops**. Declarations of equations start with **eq**, and those of conditional ones with **ceq**. The CafeOBJ system rewrites a given term by regarding equations as left-to-right rewrite rules.

We assume that there exists a universal state space called Υ. We also suppose that each data type used has been defined beforehand, including the equivalence between two data values v_1, v_2 denoted by $v_1 = v_2$. A system is modeled by observing only quantities that are relevant to the system and how to change the quantities by state transition from the outside of each state of Υ. An OTS S consists of $\langle O, I, T \rangle$:

O: A set of observers. Each $o \in O$ is a function o : $\Upsilon \rightarrow D$, where D is a data type. Given an OTS S and two states v_1, $v_2 \in \Upsilon$, the equivalence between two states, denoted by $v_1 =_S v_2$, with respect to S is defined as $\forall o \in O . o(v_1) = o(v_2)$.

I: The set of initial states such that $I \subset \Upsilon$.

T: A set of conditional transition rules. Each $t \in T$ is a function t: $\Upsilon/{=_S} \rightarrow \Upsilon/{=_S}$ on equivalence classes of Υ with respect to $=_S$. Let $t(v)$ be the representative element of $t([v])$ for each $v \in \Upsilon$ and it is called *the successor state* of v with respect to t. The condition c_t, for a transition rule $t \in T$ is called *the effective condition*. The effective condition is supposed to satisfy the following requirement: given a state $v \in \Upsilon$, if c_t is false in v, then $v =_S t(v)$.

An OTS is described in CafeOBJ. Observers are denoted by CafeOBJ observation operators, and transition rules by CafeOBJ action operators.

An execution of S is an infinite sequence $v_0, v_1, \ldots$ of states satisfying:

- *Initiation*: $v_0 \in I$.
- *Consecution*: For each $i \in \{0,1,\ldots\}$, $v_{i+1} =_S t(v_i)$ for some $t \in T$.

3. The Protocol to be Analyzed

The protocol we have formal analyzed is called an anonymous fair exchange e-commerce protocol[1]. There are a few properties possessed by this protocol, while the fair exchange and anonymity are the primary properties that we want to analyze, so we have simplified this protocol, but remained the essential parts of it.

The simplified protocol includes five parties: customer, merchant, customer's bank and merchant' bank and a third party. Each principal (party) is given a private/public key-pair; the private key is only available to the owner, while merchant and third party is known to every customer and the same of their public keys. Nobody else is possible to know or guess the relationship between a customer (merchant) and his/her bank except themselves, and they know each others' public key. We also assume that no information about the identity of the customer is gained by communicating with the merchant. Although, in practice, this may be a questionable assumption that supporting this assumption in an implementation may involve another party, an anonymizer, which the customer trusts not to reveal identity information[2]. But in this paper, we only put emphasis on the mechanism of this protocol and analyze the message flows of it.

We show the simplified protocol in Figure 1. *C*, *M*, *CB*, *MB* and *TP* stand for a customer, a merchant, the customer's bank, the merchant's bank and third party respectively. p1, p2 and m1 to m8 are tags of messages involved in this protocol, where p1 and p2 are considered as preparation phase.

Cryptographic primitives used in the protocol are as follows:

- $cc(.)$: Cryptographic checksum, which ensures the correctness of inside information.
- $\varepsilon_p(.)$: Encryption with principal P's (one-time) public key. An exception is for the encrypted

goods, where *P* equals the key *K* or *K'* used to encrypt the goods customer purchasing.

- S_p(.): Digital signature computed with principal *P*'s (one-time) private key.

Quantities appearing in the protocol are as follows:

- PO: Purchase order, which contains name of the goods *C* is purchasing, the price *C* is paying, the pseudo identities of *C* and name of *M*.
- Cipub: The one-time public key *C* created for current purchasing.
- goods: The goods *C* is purchasing.
- Macct: Merchant *M*'s bank account in *MB*.
- MTI: Money transfer instruction, which contains the amount of money that is being transferred, *C*'s bank account in *CB* that is to be debited and *M*'s encrypted bank account that is to be credited ε_{MB}(Macct).
- P: The payment token, which contains the amount of money that is being credited and *M*'s encrypted account that will be credited ε_{MB}(Macct).
- ack: The acknowledgement *MB* sent to *M* after crediting the appropriate account.
- K and K': the keys used to both encrypt and decrypt goods. K has a mathematical relation with K'. By the theory of cross-validation[1], customer can verify that the product he is about to receive is the one he will be paying for.

The protocol description is as follows:

p1: *M* sends *TP* the goods encrypted with *K'* and the key *K'* encrypted with *TP*'s public key, and the signed cryptographic checksums of encrypted goods and encrypted Key.

p2: On receipt of P1, *TP* sends encrypted goods to customer *C*, who wants to buy this goods, and the signed cryptographic checksum of encrypted goods.

m1: *C* initiates the transaction by sending *M* the purchase order, signed cryptographic checksum of purchase order using *C*'s one-time private key and *C*' one-time public key used in this transaction, encrypted with *M*'s public key.

m2: On receipt of m1, *M* checks the correctness of PO and sends *C* the goods encrypted with K and *M*'s bank account encrypted with *MB*'s public key and signed cryptographic checksums of PO, encrypted goods and encrypted bank account using *M*'s private key.

m3: On receipt of m2, *C* checks the correctness of PO and the correctness of encrypted goods (by comparing the two encrypted goods). Then *C* sends *CB* the signed MTI, encrypted with *CB*'s public key.

m4: On receipt of m3, *CB* debits appropriate money from *C*'s bank account and creates equivalent payment token. Then *CB* sends *C* the signed P, encrypted with *C*'s public key. Here *CB* signs P using private key Bprv, which is able to be verified by all banks without knowing which bank has signed it. This idea is similar to using a group signature scheme[1].

m5: On receipt of m4, *C* sends *M* the signed P, encrypted with *M*'s public key.

m6: On receipt of m5, *M* sends *MB* the signed P, encrypted by *MB*'s public key.

m7: On receipt of m6, *MB* gets *M*'s bank account and credits the payment token into *M*'s bank account. Then *MB* sends an acknowledgement to *M*.

m8: On receipt of m7, *M* sends *C* the key K, encrypted by *C*'s one time public key, which can be used to decrypt the encrypted goods, and the signed cryptographic checksum of K using *M*'s private key.

4. Modeling the Protocol

4.1. Assumptions

We suppose that there exists one and only one legitimate third party and there exists one and only one legitimate bank for each customer and each merchant respectively. We also suppose that there exist multiple malicious (untrustworthy) principals that act as customers, merchants, banks and third party. The combination and cooperation of malicious principals is modeled as the most general intruder à la Dolev and Yao[4]. The intruder gleans as much information as he can, fakes messages based on the gleaned information and sends them to possible principals so as to attack the protocol. Since we suppose that the cryptosystem used is perfect, the intruder's action is limited. What the intruder can do are as follows:

- Eavesdrop any message flowing in the network.
- Glean any quantity from the message; however the intruder can decrypt a cipher text only if he/she knows the key to decrypt.
- Fake and send messages based on the gleaned information; however the intruder can sign something only if he/she knows the key to sign, and cannot predict unknown values such as Cacct.

4.2. Formalization of Messages

Before formalizing messages, we formalize data types that constitute messages by means of initial algebra. The following visible sorts and the corresponding data constructors for those data types are parts of the whole 43 data types we formalized:

Customer, **Merchant**, **CBank**, **MBank** and **Tparty** denote customers, merchants, customers' banks, merchants' banks and third party. Constants **ic**, **im**, **icb**, **imb** and **itp** denote the intruder acting as a customer, a merchant, a customer's bank, a merchant's bank and third party.

Random denotes the unguessable random numbers.

Cfname denotes the fake name of customer. Given a customer *c*, a merchant *m* and a random number *r*, the fake name customer *c* used to by goods from merchant *m* is denoted as cfn(*c*,*m*,*r*), where *r* makes the fake name globally unique.

Caccount and **Maccount** denote customer's bank account in CBank and merchant's bank account in MBank, respectively. Given a customer *c* and his fake name *cfn* used in current transaction, *c*'s bank account is denoted as ca(*c*,*cfn*). Here why we defined customer's fake name in the bank account is because that by doing this, we can relate customer's fake name with his bank account information, which is needed for us to verify the property we want to analyze. Given a merchant *m*, his bank account is denoted as ma(*m*).

Price denotes the price of the goods.

Po denotes the purchase order. Give a customer *c*'s fake name *cfn*, the name of the goods *g*, the price of the goods *pri* and merchant *m*, po(*g*,*pri*,*cfn*,*m*) denotes the purchase order a customer *c* sends to the merchant *m* using *c*'s fake name *cfn*.

Key denotes the keys created by the merchant, which are used to encrypt the goods sent to the third party and customer. For example, given a fake name of customer *cfn*, a merchant *m* and a random number *r*, k(*cfn*,*m*,*r*) denotes the key created by merchant *m* for *cfn*, and *r* make the key globally unique.

Encgoods denotes the encrypted goods. Given a goods *g* and key *k*, eg(*g*,*k*) denotes the goods encrypted by *k*.

Ack denotes the acknowledgement.

Sig*i* (*i* = p1, p2, p3, 1, 2, 3, 4, 5, 6, 7, 8) denotes digital signature Sig*i*(*i* = p1, p2, p3, 1, 2, 3, 4, 5, 6, 7, 8). Given a cryptographic checksum of encrypted goods, which is cc($\varepsilon_{K'}$(goods)) and the private key of merchant *m*, which is *mprv*, the corresponding Sigp1 is denoted as sigp1(cc($\varepsilon_{K'}$(goods)),*mprv*). The other ten digital signatures are defined likewise.

Cipher*i*(*i* = p1, p2, 1, 2, 3, 4, 5, 6, 7) denotes Cipher text*i*(*i* = p1, p2, 1, 2, 3, 4, 5, 6, 7). Given a goods *goods* and the key *k'*, the corresponding cipherp1 is denoted as encp1(*goods*,*k'*). The other eight cipher texts are defined likewise.

Mti denotes money transfer instruction. Given a price of goods *pri*, a customer's bank account *cacct* and cipher4 *enc4*, mti(*pri*,*cacct*,*enc4*) denotes the money transfer instruction created by customer.

Ptoken denotes the payment token. Given a price of goods *pri* and cipher4 *enc4*, ptoken(*pri*,*enc4*) denotes the payment token created by customer's bank.

For each data constructor, such as cipherp1, projection operators, such as content and key that return arguments' values, are also defined. For example, content(encp1(*goods*,*k'*)) = *goods* and key(encp1(*goods*,*k'*)) = *k'*.

After formalization of data types, we begin to formalize messages. According to the protocol, we have ten operators (p1, p2 and m1 to m8) to denote or construct the ten messages in the protocol, which can be divided into two categories. The result is shown respectively in Figure 2 and Figure 3.

Message is the visible sort denoting messages. For each data constructor, projection operators that return the corresponding arguments are also defined. Such as for the data constructor m1, we define projection operators like **creatorm1**, **senderm1**, **receiverm1**, **pom1**, **s1m1** and **c1m1**, which return the first argument, the second argument, the third argument, po, sig1 and cipher1 in message 1, respectively. Moreover predicates **p*i*?**(*i* = 1, 2) and **m*j*?**(*j* = 1, 2, 3, 4, 5, 6, 7, 8) are also defined, which check whether a given message is a *i* message, where *i* = 1, 2 or a *j* message, where *j* = 1, 2, 3, 4, 5, 6, 7, 8.

In the first category of messages, which describe the messages among the customer, merchant and third party, the first, second and third arguments of each constructor denote the actual creator (sender), the seeming sender and the receiver of the corresponding message. The first argument is meta-information that is only available to the outside observer of OTS and the principal that has sent the corresponding message, and this argument cannot be forged by the intruder, while the remaining arguments may be forged by the intruder. Therefore suppose that there exists a message in the network. It is true that the principal denoted by the first argument has sent the message; if the first argument is the intruder and the second one is not, then this message is a fake message created by the intruder.

In the second category of messages, which describe the messages between customer (merchant) and the customer's (merchant's) bank, the first and second argument denotes the actual creator (sender) and receiver of corresponding message, respectively. Because, as we have supposed before, nobody else (such as the intruder) is possible to know or guess the relationship between a customer (merchant) and his/her bank. So the only possibility that an intruder can fake this kind of messages, for example message m3, is a customer acting as an intruder fakes message m3 and sends the fake message to his own bank. Therefore, the value of the first argument in the second category of messages must be equal to the actual sender of corresponding messages. Other parts of messages in category two are defined similarly to the messages in

```
op p1  : Merchant Merchant  Tparty   Cipherp1  Sigp1 Cipherp2 Sigp2     -> Message
op p2  : Tparty     Tparty       Cfname  Cipherp1  Sigp3                -> Message
op m1 : Customer Cfname   Merchant   Po Sig1 Cipher1                    -> Message
op m2 : Merchant Merchant Cfname     Sig2 Encgoods Sig3 Cipher2 Sig4    -> Message
op m5 : Customer Cfname   Merchant   Cipher5                            -> Message
op m8 : Merchant Merchant Cfname     Cipher7 Sig8                       -> Message
```

Figure 2: Data constructors denoting the six messages in the first category

```
op m3 : Customer  CBank      Cipher3                                    -> Message
op m4 : CBank      Customer  Cipher4                                    -> Message
op m6 : Merchant  MBank      Cipher6                                    -> Message
op m7 : MBank      Merchant  Sig7                                       -> Message
```

Figure 3: Data constructors denoting the four messages in the second category

the first category, there are also projection operators that can return corresponding arguments.

4.3. Formalization of the Network

The network is modeled as a bag of messages, which is used as the storage that the intruder can use by means of gleaning quantities from the network and faking messages based on these gleaned quantities. The network is also used as each principal's private memory that reminds the principal to send messages, of which first arguments are the principal. The emptiness of the network means that no message has been sent.

The intruder tries to glean 39 kinds of quantities from the network as much as possible, which include keys that are used to encrypt goods and other information, customer's bank account, money transfer instruction, encrypted goods, payment token, purchase order, acknowledgement, cryptographic checksums of corresponding information, digital signatures ranging from sigp1 to sig8 and cipher texts ranging from cipherp1 to cipher7, etc.

The collections of above quantities gleaned by the intruder from the network are denoted by CafeOBJ operators, parts of which are listed as follows:

```
op  ccipub    : Network -> ColCipub
op  ckey      : Network -> ColKey
op  ccacct    : Network -> ColCacct
op  cmti      : Network -> ColMti
op  ceg       : Network -> ColEncgoods
op  cack      : Network -> ColAck
op  csigpi    : Network -> ColSigpi  (i = 1, 2, 3)
op  csigi     : Network -> ColSigi   (i = 1, ... , 8)
op  cenci     : Network -> ColEncpi (i = 1, 2)
op  cencpi    : Network -> ColEnci   (i = 1, ... , 7)
```

Network is the visible sort denoting networks. Col*X* is the visible sort denoting collections of quantities denoted by visible sort *X*, in which *X* denotes the quantities introduced above. For example, given a snapshot *nw* denotes the network, ccipub(*nw*) and ckey(*nw*) denote the collection of Cipub and Keys gleaned by the intruder from *nw*.

Those operators are defined with equations. We give an example of ccipub in Figure 4.

In Figure 4, constant **void** denotes the empty bag, which means no message in the network. Operator _**\in**_ is the membership predicate of collections. The comma in "MSG,NW" is the data constructor of bags, and MSG,NW denotes the network obtained by adding message MSG to network NW. The first equation says that initially the only CIPUB that is available to the intruder is the CIPUB created by intruder himself. Since CIPUB is only transferred in message m1, so if there exists message m1 in the network, then if the Cipher1 of message m1 is encrypted with the intruder's public key and the content of Cipher1 is equal to CIPUB, then CIPUB is available to the intruder, which is denoted by the second conditional equation. Otherwise, if these conditions are not satisfied, then the intruder cannot get CIPUB from current message and will check network recursively, which is denoted by the last equation.

Equations defining the remaining data constructors are written in a similar way.

4.4. Formalization of Trustable Principals

Before modeling the behavior of trustable principals, we describe the values observable from the outside of the system. We suppose that the set of used random numbers and the network are observable. The observers are denoted by CafeOBJ observation

```
eq   CIPUB \in ccipub(void)  =  (cocfn(cfnoci(CIPUB)) = ic) .
ceq  CIPUB \in ccipub(MSG,NW)  =  true
     if m1?(MSG) and key(c1m1(MSG))  =  im and content(c1m1(MSG))  =  CIPUB .
ceq  CIPUB \in ccipub(MSG,NW)  =  CIPUB \in ccipub(NW)
     if not(m1?(MSG) and key(c1m1(MSG))  =  im and content(c1m1(MSG)) = CIPUB) .
```

Figure 4: Equations defining ccipub

```
bop sdp1 :  System Merchant Tparty Encgoods Key                                 -> System
bop sdp2 :  System Customer Tparty Encgoods Random Message                      -> System
bop sdm1 :  System Customer Merchant  Price      Random Message                 -> System
bop sdm2 :  System Merchant Random    MBank  Message                            -> System
bop sdm3 :  System Customer CBank     Random Merchant Message Message           -> System
bop sdm4 :  System CBank     MBank    Message                                   -> System
bop sdm5 :  System Customer Merchant  Random Message Message Message Message    -> System
bop sdm6 :  System Merchant MBank     Message Message Message                   -> System
bop sdm7 :  System MBank    Ack       Message                                   -> System
bop sdm8 :  System Merchant Message   Message Message Message Message           -> System
```

Figure 5: Action operators denoting the behavior of trustable principals

```
op   c-sdm1 : System Customer Merchant Price Random Message -> Bool
eq   c-sdm1(S,C,M,PRI,R,MSG)  =  (MSG \in nw(S) and p2?( MSG) and cocfn(receiverp2(MSG)) = C
                                  and cccep1(content(sep3op2(MSG))) = ep1op2(MSG)
                                  and tpotpprv(key(sep3op2(MSG))) = senderp2(MSG)) .
ceq ur(sdm1(S,C,M,PRI,R,MSG))  =  ur(S) .
ceq nw(sdm1(S,C,M,PRI,R,MSG))
     =  m1(C,cfn(C,M,R),M, po(g(M,cfn(C,M,R),R),PRI,cfn(C,M,R),M),
           sig1(ccpo(po(g(M,cfn(C,M,R),R),PRI,cfn(C,M,R),M)),ciprv(cfn(C,M,R),M,R)),
           enc1(cipub(cfn(C,M,R),M,R),M))  , nw(S)
   if c-sdm1(S,C,M,PRI,R,MSG) .
ceq sdm1(S,C,M,PRI,R,MSG)     =  S   if not c-sdm1(S,C,M,PRI,R,MSG) .
```

Figure 6: Equations defining sdm1

operators **ur** and **nw**, respectively, which are declared as follows:

```
bop ur  : System -> URand
bop nw  : System -> Network
```

where **URand** is the visible sort denoting sets of random numbers. The set of used random numbers is used to generate really fresh random numbers.

The behavior of trustable principals is sending ten kinds of messages according to the protocol, which is denoted by ten CafeOBJ action operators. The action operators are shown in Figure 5.

For each action operators, let c-sdp*i* (*i*=1, 2) and c-sdm*j* (*j*=1,…,8) be the operators denoting the condition on which sdp*i* and sdm*j* can be effectively executed. Given *s*, *c*, *m*, *pri*, *r* and *msg* denote a state of the system, a customer, a merchant, the price of goods, a random number and a message, respectively. Let's see sdm1 for example in Figure 6. c-sdm1(s,c,m,pri,r,msg) denotes the condition that customer c has received message p2 and the content of message p2 is correct. If the condition is satisfied, then action sdm1 will add message m1 into the network. Otherwise, this action does not change the state of the system.

The remaining action operators are defined in a similar way.

4.5. Formalization of the Intruder

We have defined what information the intruder can glean from the network, then we describe what messages the intruder fakes based on the gleaned information. We suppose that the intruder can fake any message if the message can be made from the quantities gleaned by the intruder. However we do not consider meaningless messages faked by intruder, which will not attack the protocol.

The intruder's faking messages are denoted by CafeOBJ action operators, which are divided into ten classes, each of which corresponds to faking each type of message. In this paper, we give an example that

```
bop fkm11 : System Customer Merchant Random Po Sig1 Cipher1        -> System
bop fkm12 : System Merchant Customer Po  Random Random             -> System
bop fkm13 : System Customer Merchant Random Price Random           -> System
```

Figure 7: Action operators denoting the intruder's faking message m1

```
op c-fkm12 : System Merchant Customer Po Random Random -> Bool
eq c-fkm12(S,M,C,PO,R1,R)
     = not(R1 \in ur(S)) and PO \in cpo(nw(S)) and cfn(C,M,R) \in ccfn(nw(S)) .
ceq ur(fkm12(S,M,C,PO,R1,R)) = R1 ur(S) if c-fkm12(S,M,C,PO,R1,R) .
ceq nw(fkm12(S,M,C,PO,R1,R))
      = m1(ic,cfn(C,M,R),M,PO, sig1(ccpo(PO),  ciprv(cfn(ic,M,R1),M,R1)),
         enc1(cipub(cfn(ic,M,R1),M,R),M)) , nw(S)
     if c-fkm12(S,M,C,PO,R1,R) .
ceq fkm12(S,M,C,PO,R1,R) = S      if not c-fkm12(S,M,C,PO,R1,R) .
```

Figure 8: Equations defining fkm12

INV1.1 For any reachable state *s*, any merchant *m*,*m*1,*m*2, any customer *c*,*c*1, any third party *tp*1, any random number *r*, some *sigp*3, *sig*2, *sig*3,*sig*4, *sig*7, *sig*8, *encp*1,*enc*2, *enc*5, *enc*6, *enc*7 and any *eg*,

not(*m*= **im** and *c* = **ic**) and p2(*tp*1,*tp*,cfn(*c*,*m*,*r*),*encp*1,*sigp*3) \in nw(*s*) and
m2(*m*1,*m*,cfn(c,*m*,r),*sig*2,*eg*,*sig*3,*enc*2,*sig*4) \in nw(*s*) and m8(*m*2,*m*,cfn(*c*,*m*,*r*),*enc*7,*sig*8) \in nw(*s*)
implies
m5(*c*1,cfn(*c*,*m*,*r*),*m*,*enc*5) \in nw(*s*) and m6(*m*,mb(*m*),*enc*6) \in nw(*s*) and
m7(mb(*m*),*m*,*sig*7) \in nw(*s*)

INV1.2 For any reachable state *s*, any merchant *m*,*m*1, any customer *c*,*c*1, any third party *tp*1, any random number *r*, some *sigp*3, *sig*2, *sig*3,*sig*4, *sig*7, *encp*1, *enc2*, *enc*5, *enc*6 and any *eg*,

not(*m*= **im** and *c* = **ic**) and m5(*c*1,cfn(*c*,*m*,*r*),*m*,*enc*5) \in nw(*s*) and
m6(*m*,mb(*m*),*enc*6) \in nw(*s*) and m7(mb(*m*),*m*,*sig*7) \in nw(*s*)
implies
p2(*tp*1,*tp*,cfn(*c*,*m*,*r*),*encp*1,*sigp*3) \in nw(*s*) and m2(*m*1,*m*,cfn(c,*m*,r),*sig*2,*eg*,*sig*3,*enc*2,*sig*4) \in nw(*s*)

INV2 For any reachable state *s*, any merchant *m*, any customer *c*, any random number *r*,

ca(*c*,cfn(*c*,*m*,*r*)) \in cacct(nw(*s*)) implies (*c* = **ic**)

Figure 9: Formal definitions of properties

describes the three action operators for faking message m1, which are shown in Figure 7. The action operators are defined with equations, and the equations defining fkm12 are shown in Figure 8. The condition of fkm12 is denoted as c-fkm12, which demands that the random number used by intruder is never used before, purchase order po and customer's fake name cfn are gleaned by the intruder. If c-fkm12 is satisfied, then the action fkm12 will add the random number intruder used into the set of random number and add a faking m1 message into the network. Otherwise, this action will not change the state of the system. The meaning of this faking message is that: a customer *ic*, who is acting as an intruder, sends to merchant *m* his own (the intruder *ic*') one-time public key and signed cryptographic checksum of the purchase order using intruder *ic*'s one-time private key, while this message seems to be sent from another customer *c*'s fake name.

The remaining action operators are defined with equations in a similar way.

5. Verification

The main properties focused on in this paper are safety properties (precisely invariants). Proofs of invariants often need induction, especially induction on the number of applied transition rules (or action operators).

Let *c* be a customer who performs a transaction using the protocol with a merchant *m*, and we suppose that *c* and *m* are trustable. The informal descriptions of fair exchange property are as follows:

- **FX1.** If *c* receives the goods *c* has ordered, then *m* has already been paid or will be eventually paid for the goods.
- **FX2.** If *m* is paid for the goods *c* has ordered, then *c* has already received or will eventually receive the goods.

The informal description of customer's anonymity is as follows:

- The bank account of a customer, who uses a fake name to purchase goods from a merchant, is never leaked.

The formal definitions of the safety part of the first two properties and the formal definition of the last property are shown in Figure 9.

Property INV1.1 in Figure 9 claims that if *c* receives the goods *c* has ordered, then *m* has already been paid. That is to say, INV1.1 can imply FX1. In INV1.1, *tp*1 may be different from trustable third party *tp*, which implies that *tp*1 might be an intruder acting as a third party. The meanings of *m*1, *m*2 and *c*1 are the same as *tp*1(so do the same constants in INV1.2)

Property INV1.2 in Figure 9 claims that if *m* is paid for the goods *c* has ordered, then *c* has already received the encrypted goods. To deduce FX2, we also need to prove another liveness property that claims if *m* is paid for the goods *c* has ordered, then *c* will eventually receive the key that can decrypt the encrypted goods.

Property INV2 in Figure 9 claims that any bank account gleaned by the intruder is the intruder's own bank account. In other words, the bank account of a trustable customer *c* who uses the fake name cfn(*c*,*m*,*r*) to purchase goods from a merchant *m* will not be gleaned by the intruder. In this paper, we suppose that from the bank account of a customer, others can recognize the identity of the customer. So the second property proves the customer's anonymity, although it can also be considered as safety property.

We have formally verified that INV2 holds for the protocol and partly verified that INV1.1 and INV1.2 hold for the protocol. The verification has been done by writing proof scores showing that the protocol satisfies these requirements in CafeOBJ and executing the proof scores with the CafeOBJ system. In this paper, we do not describe the detail that how to verify properties with rewriting as described in section1.

6. Conclusions

We have described a case study that we have formally analyzed an anonymous fair exchange e-commerce protocol that is claimed to satisfy fair exchange and customer's anonymity. Concretely, the protocol has been modeled as an OTS, the OTS has been written in CafeOBJ, and it has been partly verified that the OTS satisfies the safety part of the two requirements.

Our future work includes the completion of the verification of INV1.1 and INV1.2 in Figure 9. We should verify that the OTS has the liveness part of FX1 and FX2 as well. With respect to customer's anonymity, we assume that the real identity of a customer is equivalent to his/her bank account. We should also relax this assumption. There exists another important property with respect to customer's anonymity: no single principles have enough information to link a customer to a merchant. It seems like that the current way of modeling security protocols may have to be modified so as to deal with this property.

Acknowledgement

This research is conducted as a program for the "Fostering Talent in Emergent Research Fields" in Special Coordination Funds for Promoting Science and Technology by Ministry of Education, Culture, Sports, Science and Technology.

References

[1] Indrakshi Ray and Indrajit Ray, "An anonymous fair exchange e-commerce protocol", *Proceedings of the First International Workshop on Internet Computing and E-Commerce*, San Francisco, CA, April 2001.

[2] Jean Camp and J.D. Tygar, "Anonymous atomic transactions", *2nd Annual USENIX Workshop on Electronic Commerce Proceedings*, Oakland, Nov. 1996, pages 123-134.

[3] G. Lowe, "An attack on the Needham-Schroeder public-key authentication protocol", *Information Processing Letters,* 1995, 56:131-133.

[4] D. Dolev and A. C. Yao, "On the security of public key protocols", *IEEE Transactions on Information Theory,* IT-29:198-208, 1983

[5] CafeOBJ. CafeOBJ web page.
http://www.ldl.jaist.ac.jp/cafeobj/, 2001

[6] R. Diaconescu and K. Futatsugi, "CafeOBJ Report", *AMAST Series in computing,* 6. World Scientific, Singapore, 1998

[7] K. Ogata and K. Futatsugi, "Flaw and modification of the *i*KP electronic payment protocols", *Information Processing Letters,* 2003, 86:57-62.

[8] K. Ogata and K. Futatsugi, "Formal analysis of the iKP electronic payment protocols", In *International symposium on Software Security*, volume2609 of LNCS, pages 441-460. Springer, 2003

[9] K. Ogata and K. Futatsugi, "Proof scores in the OTS/CafeOBJ method", In *6th IFIP WG6.1 Iternational Conference on Formal Methods for Open Object-Based Distributed Systmes*, LNCS 2884, pages 170-184, 2003