Computer Aided Verification of Accountability in Electronic Payment Protocol with CryptoVerif

Bo Meng

International Journal of Advancements in Computing Technology, Volume 3, Number 3, April 2011

Abstract

During the past few decades electronic payment protocols has been studied. A lot of electronic payment protocols, for example, 3KP, SET, have been proposed which claimed that have security properties. To our best knowledge, till now analysis of 3KP protocol has not with automatic tool in computational model. Recently owning to the contribution of Meng et al., 3KP protocol can be analyzed with automatic tool in computational model. In this study firstly the state-of-art of electronic payment protocol and the proof are presented. Then the term, process and correspondence assertion in Blanchet calculus are used to model accountability and 3KP protocol with Meng et al. mechanized framework of electronic payment protocols in computational model with active adversary. Finally, 3KP protocol is analyzed in Meng et al. framework with mechanized tool CryptoVerif. The result shows that 3KP protocol has money accountability and goods accountability, which is consistent with its claim. To our knowledge, we are conducting the first automatic analysis of 3KP protocol in computational model.

Keywords: Computational Model, Money Accountability, Goods Accountability, Mechanized Proof

1. Introduction

With the expansion of the Internet, many electronic commerce systems have been developed in the past few years. Electronic payment protocols play an important role in electronic commerce system. Hence the practical secure electronic payment protocol should have accountability, atomicity, anonymity, non-repudiation and fairness. Generally electronic commerce protocol uses the cryptographic technologies to confirm these securities in electronic commerce.

During the past few decades, many electronic payment protocols have been introduced, such as SOCPT [1], Virtual Credited Card, Secure Electronic Transaction (SET), Internet Key Protocol (iKP) [2], Net Pay [3].

In order to verify the security properties of security protocol including electronic payment protocols and increase the confidence of the people, two approaches have been developed for analyzing security protocols form the beginning of the 1980s. One approach relies on a symbolic model of protocol executions in which cryptographic primitives are treated as black boxes. Several logics used to analyze the accountability of electronic payment protocol have been introduced, such as Kailar logic [4, 5], Kessler and Neumann logic [6], Kungpisdan and Permpoontanalarp logic[7], Van Herreweghen logic[8], Meng and Zhang formal framework[9]. Many automatic tools in this approach have been developed, for example, SMV, NRL, Casper, Isabelle, Athena, Revere, SPIN, Brutus, ProVerif, Scyther, and AVISPA. Several electronic payment protocols have been analyzed with automatic tools, for example, SET protocol and its variants are analyzed by Isabelle [10], SMV [11] and AVISPA [12]. But the results of proof based on symbolic model are not quite clear.

The other approach relies on a computational model that base issues of complexity and probability. This approach use a strong notion of security, guaranteed against all probabilistic polynomial-time attacks. Backes and Dursmuth[13] present the first cryptographically sound Dolev-Yao-style security proof of iKP protocol by hand. The computation approach can be more realistic, but it is difficult to automatic proof until the introduction of mechanized tool CryptoVerif[14,15] which is the only automatic tool with computational model. To our best knowledge, it does not existing that analysis of
security properties and electronic payment protocols with automatic tool in computational model until Meng et al.[16] propose the first automatic framework of money accountability and goods accountability based on Blanchet calculus. Meng et al. mechanized framework can be used to automatically analyze the money accountability and goods accountability of electronic payment protocols with the mechanized tool CryptoVerif.

Owing to analysis of 3KP protocol is not clear, at the same time especially owing to the contribution of Meng et al.[16], in this study, based on Meng et al. mechanized framework[16], we use Blanchet calculus in the computational model to mechanized analyze 3KP protocol with mechanized tool CryptoVerif.

2. Contribution and overview

During the past few decades electronic payment protocols has been studied. A lot of electronic payment protocols have been proposed which claimed that have the security properties, for example, accountability, atomicity, anonymity, non-repudiation and fairness. In order to verify the security properties of security protocol include electronic payment protocols and increase the confidence of the people, two approaches have been developed for analyzing security protocols form the beginning of the 1980s. One approach relies on a symbolic model of protocol executions in which cryptographic primitives are treated as black boxes. The other approach relies on a computational model that base issues of complexity and probability. This approach use a strong notion of security, guaranteed against all probabilistic polynomial-time attacks. The computation approach can be more realistic. To our best knowledge, these security properties and electronic payment protocols are analyzed with informal method, or with symbolic method, or with computational model by hand, which depends on experts’ knowledge and skill and is prone to make mistakes. There does not existing that analysis of electronic payment protocols and its security properties with automatic tool in computational model until Meng et al.[16] propose the first automatic framework of money accountability and goods accountability based on Blanchet calculus in computational model. And the mechanized CryptoVerif is the only automatic tool with computational model.

So analysis of security properties and electronic payment protocols with automatic tool in computational model plays an important role in security protocol world and is a significant work.

Recently owning to the introduction of Blanchet calculus and CryptoVerif, at the same time owning to analysis of 3KP protocol is not clear, especially owning to the contribution of the contribution of Meng et al.[16], in this study, based on Meng et al. mechanized framework[16], we use Blanchet calculus in the computational model to mechanized analyze 3KP protocol with mechanized tool CryptoVerif.

The main contributions of this paper are summarized as follows in detail:

- The state-of-art of electronic payment protocol and the proof including in symbolic model and in computational model are presented. We find that electronic payment protocols and its security properties are analyzed with informal method, or with symbolic method, or with computational model by hand. There does not existing that analysis of electronic payment protocols and its security properties with automatic tool in computational model until Meng et al.[16] propose the first automatic framework of money accountability and goods accountability based on Blanchet calculus in computational model. And the mechanized CryptoVerif is the only automatic tool with computational model.
Applying Meng et al. automatic model [16] based on Blanchet calculus in computational model with active adversary for automatically analysis of 3KP protocol. So the term, process and correspondence assertion in Blanchet calculus is used to model security properties including accountability and 3KP protocol. The money accountability and goods accountability are expressed by non-injective or injective correspondence. The analysis itself is performed by automatic tool CryptoVerif developed by Blanchet.

The result shows that 3KP electronic payment protocol has money accountability and goods accountability, which is consistent with its claim. To our knowledge, we are conducting the first automatic analysis in computational model of 3KP electronic payment protocol in active adversary. Figure 1 shows the model of automatic verification of electronic payment protocols.

3. Related work

In this section the state-of-art of electronic payment protocol and its proof in symbolic model and in computational model are presented. We find that security properties and electronic payment protocols are analyzed with informal method, or with symbolic method, or with computational model by hand. There does not existing that analysis of electronic payment protocols and its security properties with automatic tool in computational model until Meng et al.[16] propose the first automatic framework of money accountability and goods accountability based on Blanchet calculus in computational model. And the mechanized CryptoVerif is the only automatic tool with computational model.

The practical secure electronic payment protocol should have the following properties: accountability, atomicity, anonymity, non-repudiation and fairness. These secure properties play important roles in implementation of secure transactions over the public Internet. Electronic commerce protocol uses the cryptographic technologies to confirm the security of parties in the electronic commerce. A lot of electronic payment protocols, for example, SOCPT, Virtual Credited Card, SET, I kp, VCPT, CyberCoin, DigiCash, eCoin, MilliCent, NetCash, NetBill, FSTC, CAFÉ,Agora,Mondex, MiniPay, NetCents, PayWord, LMCCPP, NetPay, are proposed.

Accountability[4,5] is the property whereby the association of a unique originator with an object or action can be proved to a third party, which concerned with the ability to show that particular parties who engage in electronic commerce protocols are responsible for some transactions. In particular, accountability involves the ability of a party to convince to another party. Generally, accountability is used to resolve the disputation among participants in the electronic commerce. In the practical aspects the accountability consists of money or payment accountability [8] and goods accountability [7, 17]. Money accountability is about the authorized transfer of money from customer’s account to merchant’s account. Goods accountability is about the authorized order of goods by a customer. Goods accountability can be used to resolve disputes on the mismatch between the goods that is ordered by a customer and the goods that is delivered by a merchant and it can also be used to deal with the goods atomicity and the certified delivery of electronic commerce protocol.

SOCPT [1] is based on analysis of most existing online payment protocols. It is of security, accountability, atomicity, partial anonymity, non-repudiation and fairness. The technologies applied by SOCPT mainly include symmetric techniques, asymmetric techniques, hash function and digital signature and so on. symmetric techniques and asymmetric techniques is used to guarantee data confidentiality and use digital signature to implement message integrity, consistency and non-
repudiation, use dual signature to separate order information and personal finical information.

SET was developed by MasterCard and Visa corporations and is used widely in the world by credit card. SET use public key cryptography scheme proposed by two major credit card companies. There are three parties engaging in SET protocol: client, merchant, and acquirer. Client is an authorized party who has a credit card and wants to buy goods or services from merchant. Merchant is an authorized party who has goods or services to sell to customer. A merchant that accepts payment credit cards must have a relationship with an acquirer. Acquirer, behaving as a payment gateway between Customer, Merchant, and Banks, is a party who deducts the money from customer’s account and transfers the money to merchant’s account. SET is money atomicity protocol Goods atomic protocols are money atomic, and also affect an exact transfer of goods for money. NetBill protocol is certified delivery protocol. Digicash is not money atomic protocol.

iKP [2] is also credit-card based ecommerce payment protocol. The protocol step of iKP is similar to that of SET. iKP is a family of protocol in that it consists of three types of protocol, which depends on the number of certificate of the engaging party. 3KP protocol is one of the families of iKP electronic payment protocols and consists of customer who will make the payment, merchant who will receive the money and acquirer which will withdraw the money from the account of customer to account of merchant. The technologies applied by 3KP protocol mainly include symmetric encryption, asymmetric encryption, hash function and digital signature. It uses symmetric techniques and asymmetric techniques to guarantee data confidentiality and use digital signature to implement message integrity, consistency and accountability.

In order to verify the security properties of security protocol include electronic payment protocols and increase the confidence of the people, two approaches have been developed for analyzing security protocols form the beginning of the 1980s. The one approach relies on a symbolic model of protocol executions in which cryptographic primitives are treated as black boxes. Since the seminal work of Dolev and Yao, it has been realized that this approach enables significantly simpler and many automatic tools have been developed. But the result of proof is not quite clear. The other approach relies on a computational model that base issues of complexity and probability. This approach use a strong notion of security, guaranteed against all probabilistic polynomial-time attacks. The computation approach can be more realistic, but it is difficult to automatic proof.

In formal community Kailar [4, 5] is probably the first who propose a modal logic to reason about accountability. Kailar’s definition of accountability is concerned with the ability to prove the association of an originator with some action to a third party without revealing any private information to the third party. The party who can prove such a statement is called a prover whereas the third party who is convinced of the proof is called a verifier. Kailar employs the modal operator ‘CanProve’ to formalize the concept of accountability, i.e. Prover CanProve φ to Verifier where Prover and Verifier stand for prover and verifier, respectively, and φ stands for a general statement about some action. However, Kailar’s logic is not suitable for analyze the real-world e-commerce protocols because of the following two reasons: firstly, Kailar’s logic can analyze the signed plain message only. Messages in real-world ecommerce protocols are not just signed plain messages, but they often are multiply encrypted and/or hashed messages which are signed Secondly, Kailar’s logic does not reason about verifiers at all. Van Herrewegen[8] points out that reasoning about verifiers is essential for analyzing real-world e-commerce protocols. It should be noted also that Kailar’s definition for accountability is general in that the actions that are associated with an originator can be of any kinds.

Following Kailar [4, 5], Kessler and Neumann [6] employs a modal logic to reason about the accountability. However, Kessler and Neumann provide an alternative definition of the modal operator “CanProve” by means of sending messages. Its goal to show the accountability is to show “Prover believes Prover CanProve φ to Verifier”. One way to show that Prover believes Prover CanProve φ to Verifier holds is for Prover to believe that Prover can convince Verifier to believe φ by sending some messages that Prover has to Verifier. Thus, this logic offers reasoning about both prover’s beliefs and verifier’s beliefs, and in particular, prover’s beliefs about verifier’s beliefs.

Based on Kessler and Neumann’s logic [6], Kungpisdan and Permpong SLarp [7] provide a modal logic which is an extension and a simplification of Kessler and Neumann’s logic. It employs the concept of provable authorization in the present of private information. In order to solve disputes, a prover wants to send only the necessary information to prove some statements to a judge who acts as a
verifier without revealing the unnecessary private information. With this concept, prover can prove the
statement without revealing private information to verifier. They extend Kessler and Neumann’s logic
in two main aspects. With it they analyze SET and iKP protocol. They argue that the analysis of two
kinds of accountability shows that SET lacks of both kinds of accountability because of its message
format that combines Price and OD with in the same hash Hash (Price, OD). When proving money
accountability, prover is required to send both Price and OD, which are the inputs of hash, to verifier in
order to prove Price. Prover is also required to reveal Price to verifier in order to prove goods
accountability. Proving money accountability in iKP is successful because Price and OD are separated
with applying hash function. Verifier cannot infer OD because it is hashed.

Van Herreweghen[8] proposes informal description of authorization and gives an analysis of SET
and iKP. The analysis shows that the Customer in a SET transaction has no secure receipt of payment.
A comparison shows the equivalent version of iKP to provide more complete evidence than SET. The
analysis is not formal since it is done without using any formal logic. However, the analysis is
presented partly in rule-based styles.

Meng et al. [18] use the Kessler and Neumann logic to prove the soundness of the requirements and
analyze SOCPT protocol with its framework. Its results show that it has the properties of money
accountability and goods accountability. Meng and Zhang [9] also introduce generally formal
definition of accountability in electronic transaction based on Kessler and Neumann logic and the SET
protocol is analyzed with its framework. It results show that it has the properties of money
accountability and goods accountability. They also think that the analysis of SET by Kungpisdan and

Brackin[19] uses an automated theorem prover Automatic Authentication Protocol Analyzer based
on Higher Order Logic to analyze two large protocols for electronic commerce: the main-and coin-
sequence protocols developed by CyberCash. These protocols are similar to SET in their primary
respects. His result shows that the two large protocols have authentication properties. Meadows and
Syverson[20] present a formal specification of requirements for the payment portion of the SET
protocol by introducing transaction vectors, projections thereon, and the vector agreement. Their
specification is expressed by NRL language. But they do not analyze SET protocol with NRL. Bella et
al.[10] use Isabelle to analyze the complete Purchase protocols of SET and find that owning to the lack
of explicitness in the dual signature makes some agreement properties fail: it is impossible to prove
that the Cardholder meant to send his credit card details to the very payment gateway that receives
them. Lu et al.[11] use SMV to analyze the authentication, confidentiality and integrity of a variant of
SET. They also talked about its lacks, for example how to deal with transaction records and give their
suggestions. Shaikh and Devane[12] use AVISPA to analyze the authentication, confidentiality and
secrecy of the SET protocol. It is shown that these securities hold within the established security of PKI.
Panti et al.[21] propose a methodology for verifying security requirements of electronic payment
protocols by means of model checking. They extended correspondence property to not only use for
authentication but also confidentiality and integrity. At the same time they analyze a variant of SET
with NuSMV and discovered two attacks that allow a dishonest user to purchase a good debiting the
amount to another user. Meng et al. [23] use model of Backes et al. to automatically verify Meng
remote internet voting protocol with automatic tool ProVerif. The result is that Meng protocol has
coercion resistance. But it has not soundness because ProVerif found an attack on soundness.

In computational model Backes and Durmith[13] present the first cryptographically sound Dolev-
Yao-style security proof of iKP protocol by hand. The payment protocol is a slightly simplified variant
of the 3KP payment protocol and comprises a variety of different security requirements ranging from
basic ones like the impossibility of unauthorized payments to more sophisticated properties like
disputability. They show that the payment protocol is secure against arbitrary active attacks, including
arbitrary concurrent protocol runs and arbitrary manipulation of bitstrings within polynomial time if the
protocol is implemented using provably secure cryptographic primitives. Although they achieve
security under cryptographic definitions, their proof does not have to deal with probabilistic aspects of
cryptography and is hence within the scope of current proof tools. The reason is that they only exploit a
Dolev-Yao-style cryptographic library with a provably secure cryptographic implementation.

Blanchet [14, 15] proposes a probabilistic polynomial calculus based on computational model. In
this calculus, messages are bitstrings and cryptographic primitives are functions operating on
bowstrings. Blanchet calculus is adapted from the pi calculus and its semantics is purely probabilistic. All processes run in polynomial time: polynomial number of copies of processes and length of messages on channels bounded by polynomials. Blanchet calculus has been carefully designed to make the automated proof security protocols. Blanchet calculus consists of terms and processes. At the same time they develop a mechanized tool CryptoVerif [14, 15] with computational model. CryptoVerif does not rely on soundness results for symbolic model but directly automate the proofs made in cryptography, based on sequences of games. It can directly prove security properties of cryptographic protocols in the computational model in which the cryptographic primitives are functions on bit-strings and the adversary is a polynomial-time Turing machine. It can prove secrecy properties and that events can be executed only with negligible probability, also it can handle various cryptographic primitives. Meng et al. [22] use Meng and Shao mechanized framework of deniable authentication protocols to analyze Fan et al. interactive deniable authentication protocol with CryptoVerif. The results show that Fan et al. protocol has weak deniability, but not strong deniability.

Recently Meng et al. [16] propose the first mechanized model of accountability in electronic payment protocol based on Blanchet probabilistic polynomial calculus. The money accountability and goods accountability are expressed by non-injective or injective correspondence. Meng et al. mechanized framework can be used to automatically analyze the money accountability and goods accountability of electronic payment protocols with the mechanized tool CryptoVerif.

To our best knowledge, there does not existing that analysis of electronic payment protocols and its security properties with automatic tool in computational model until Meng et al.[16] propose the first automatic framework of money accountability and goods accountability based on Blanchet calculus in computational model. And the mechanized CryptoVerif is the only automatic tool with computational model.

Inspired by the works of Blanchet and Meng et al. [16], in this study, we use Blanchet calculus to mechanized analyze the electronic payment protocols with CryptoVerif based on Meng et al. automatic framework.

4. Mechanized proof tool CryptoVerif

In this section we give a brief overview of the mechanized prover CryptoVerif, formalize deniable authentication protocol using it, and summarize authentication and secrecy properties proved by CryptoVerif (http://www.cryptoverif.ens.fr). In most cases, the prover succeeds in proving the desired properties when they hold, and obviously it always fails to prove them when they do not hold. In other words CryptoVerif is sound but not complete which means that it cannot prove are not necessarily invalid.

The mechanized prover CryptoVerif can directly prove security properties of cryptographic protocols in the computational model in which the cryptographic primitives are functions on bit-strings and the adversary is a polynomial-time Turing machine. It also can prove secrecy properties and events that can be executed only with negligible probability, also it can handles various cryptographic primitives, for example, MACs, stream and block ciphers, public-key encryption, signatures, hash functions. CryptoVerif works for N sessions with an active adversary. It can also give a bound on the probability of an attack (exact security). CryptoVerif runs either automatically or interactively, in which case it receives guidance from the user for selecting transformations. In a recent case study, CryptoVerif is used to verify several cryptographic primitive and security protocols.

Authentication is expressed as correspondences [18], much as in symbolic models. Computationally, correspondences assert that, if some event is executed, then other events must also have been executed, at least once, with matching parameters, at least with overwhelming probability. CryptoVerif can deal with more general properties expressed as logical formulas; also both injective and non-injective properties can be analyzed. A non-injective correspondence is a property of the form “if some events have been executed, then some other events have been executed at least once”. Injective correspondences are properties of the form “if some event has been executed n times, then some other events have been executed at least n times”. Injective correspondences are more difficult to check than non-injective ones, because they require distinguishing between several executions of the same event.

CryptoVerif operates in two modes: a fully automatic and an interactive mode. The interactive mode,
which is best suited for protocols using asymmetric cryptographic primitives, requires a CryptoVerif user to input commands that indicate the main game transformations the tool should perform. CryptoVerif is sound with respect to the security properties it shows in a proof, but properties it cannot prove are not necessarily invalid.

5. Meng et al. automatic model

In this section we review Meng et al. automatic model [16] of money accountability and goods accountability. Meng et al. automatic model uses Blanchet calculus to model the money accountability and goods accountability of electronic payment protocol.

5.1. Electronic payment protocol and active adversary model

In electronic payment protocol the principles consists of customer, merchant, and acquirer. Generally customer sends a payment order to the merchant and merchant sends it to acquirer, then acquirer gets the payment order which mean that it allows acquirer to remove money from customer’s account. After that merchant requests the acquirer to deposit money to his account. Finally acquirer sent the success of payment and deposit to customer and merchant. The communication channels consist of three channels. One is the channel \textit{channelCA} between the customer and acquirer. The other two channels are the channel \textit{channelCM} between the customer and the merchant and the channel \textit{channelMA} between the merchant and the acquirer.

In Meng et al. automatic model a probabilistic polynomial-time attacker has full control of the communications channels \textit{channelCA}, \textit{channelCM} and \textit{channelMA}: it can listen to all the transmitted information, decide what messages will reach their destination and when change these messages at will or inject its own generated messages. The formalism represents this ability of the attacker by letting the adversary be the one in charge of passing messages from one party to another. The attacker also controls the scheduling of all protocol events including the initiation of protocols and message delivery. The electronic payment protocols are in a context in which the honest participants are willing to run sessions with the adversary. That is mean the adversary is an active attacker in the channel \textit{channelCA}, \textit{channelCM} and \textit{channelMA}.

5.2. Formalization of electronic payment protocol

In electronic payment protocol \textit{EPP}, Meng et al. automatic model assumes that the first messages is sent by \textit{Merchant} to \textit{Customer}, then the information related to payment is sent to \textit{Merchant} and \textit{Acquirer}. After that the payment response information is sent to \textit{Customer} and \textit{Merchant} by \textit{Acquirer}. It also assumes that \textit{EPP} consists of odd number of rounds \textit{l} between \textit{Merchant} and \textit{Customer}, rounds \textit{m} between \textit{Merchant} and \textit{Acquirer}, rounds \textit{n} between \textit{Acquirer} and \textit{Customer}. It also assumes that the first message of rounds \textit{l} is from \textit{Merchant} to \textit{Customer}, the first message of rounds \textit{m} is from \textit{Merchant} to \textit{Acquirer} and the first message of rounds \textit{n} is from \textit{Customer} to \textit{Acquirer}. So that the 1-th message of \textit{EPP} is from \textit{Merchant} to \textit{Customer}, the \textit{n}-th message is from \textit{Merchant} to \textit{Acquirer}.

In the following we review the definition of secure electronic payment protocol \textit{SEPP} represented by a process in Blanchet calculus in Meng et al. automatic model.

**Definition SEPP**: A secure electronic payment protocol process \textit{SEPP} with session functions \textit{sessionid} and \textit{sessionid’} for any probabilistic polynomial-time adversary:

\[
\textit{SEPP} = \text{Initprocess} \left( \textit{sessionid} \right) \rightarrow \textit{CustomerProcess} \left( \textit{sessionid}^{\prime} \right) \rightarrow \textit{MerchantProcess} \left( \textit{sessionid}^{\prime} \right) \rightarrow \textit{AcquirerProcess} \left( \textit{sessionid}^{\prime} \right)
\]

Such that:

1. If the adversary just send \textit{Customer} to \textit{MerchantProcess'} as the first message and relays between \textit{MerchantProcess'} and \textit{CustomerProcess'}; between \textit{MerchantProcess'} and \textit{AcquirerProcess'}; \textit{AcquirerProcess'} and \textit{CustomerProcess'}, then \textit{CustomerProcess'} finishes with \textit{Merchant} and \textit{MerchantProcess'} finishes.
with Customer; AcquirerProcess' finishes with Merchant and MerchantProcess' finishes with Acquirer; CustomerProcess' finishes with Acquirer and AcquirerProcess' finishes with Customer.

2. With overwhelming probability, there exists an injective function that maps each index $i$ of a process MerchandProcess' that finished with Customer to the index $i'$ of a process CustomerProcess' with intended principle Merchant such that

$$\text{sessioner}'[zm_1[i], zm_2[i], \ldots, zm_n[i]] = \text{sessioner}'[y_1[i'], y_2[i'], \ldots, y_n[i']].$$

3. With overwhelming probability, there exists an injective function that maps each index $i$ of a process CustomerProcess' that finished with Merchant to the index $i'$ of a process MerchandProcess' that finished with Customer such that

$$\text{sessioner}'[y_1[i], y_2[i], \ldots, y_n[i]] = \text{sessioner}'[zm_1[i'], zm_2[i'], \ldots, zm_n[i']].$$

4. With overwhelming probability, there exists an injective function that maps each index $i$ of a process MerchandProcess' that finished with Acquirer to the index $i'$ of a process AcquirerProcess' with intended principle Merchant such that

$$\text{sessioner}'[zm_1[i], zm_2[i], \ldots, zm_n[i]] = \text{sessioner}'[y_1[i'], y_2[i'], \ldots, y_n[i']].$$

5. With overwhelming probability, there exists an injective function that maps each index $i$ of a process AcquirerProcess' that finished with Merchant to the index $i'$ of a process MerchandProcess' that finished with Acquirer such that

$$\text{sessioner}'[y_1[i], y_2[i], \ldots, y_n[i]] = \text{sessioner}'[zm_1[i'], zm_2[i'], \ldots, zm_n[i']].$$

6. With overwhelming probability, there exists an injective function that maps each index $i$ of a process CustomerProcess' that finished with Acquirer to the index $i'$ of a process AcquirerProcess' with intended principle Customer such that

$$\text{sessioner}'[z_1[i], z_2[i], \ldots, z_n[i]] = \text{sessioner}'[x_1[i'], x_2[i'], \ldots, x_n[i']].$$

7. With overwhelming probability, there exists an injective function that maps each index $i$ of a process AcquirerProcess' that finished with Customer to the index $i'$ of a process CustomerProcess' that finished with Acquirer such that

$$\text{sessioner}'[x_1[i], x_2[i], \ldots, x_n[i]] = \text{sessioner}'[z_1[i'], z_2[i'], \ldots, z_n[i']].$$

In the above definition of SEPP the injective correspondence can be instead by non-injective correspondence.

**Definition SEPP with events:** If EPP$'$ satisfies the condition one in definition SEPP and EPP$'$ satisfies the following correspondences:

1. $i$-event(whole$_\text{Customer}(\text{Merchant}, x)) \Rightarrow nj$-event(whole$_\text{Customer}(\text{Customer}, x))$
2. $i$-event(whole$_\text{Acquirer}(\text{Merchant}, x)) \Rightarrow nj$-event(whole$_\text{Acquirer}(\text{Acquirer}, x))$
3. $i$-event(whole$_\text{Customer}(\text{Acquirer}, x)) \Rightarrow nj$-event(whole$_\text{Customer}(\text{Customer}, x))$
4. $i$-event(whole$_\text{Customer}(\text{Customer}, x)) \Rightarrow nj$-event(whole$_\text{Customer}(\text{Customer}, x))$
5. $i$-event(whole$_\text{Customer}(\text{Acquirer}, x)) \Rightarrow nj$-event(whole$_\text{Customer}(\text{Customer}, x))$
6. $i$-event(whole$_\text{Customer}(\text{Customer}, x)) \Rightarrow nj$-event(whole$_\text{Customer}(\text{Customer}, x))$

with public variables $V = \emptyset$, then EPP' is a SEPP with session functions (sessionid and sessionid'). EPP$'$ process can be obtained from EPP by adding the following events in the following Table 1.
Table 1. events in SEPP

<table>
<thead>
<tr>
<th>event</th>
<th>Just before</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin\textsubscript{\textit{od}} (Customer-Merchant)</td>
<td>Customer sends order message</td>
</tr>
<tr>
<td>end\textsubscript{\textit{od}} (Merchant-Customer)</td>
<td>Merchant sends receipt message</td>
</tr>
<tr>
<td>begin\textsubscript{\textit{od}} (Merchant-Customer)</td>
<td>Merchant sends capture message</td>
</tr>
<tr>
<td>begin\textsubscript{\textit{od}} (Merchant-Acquirer)</td>
<td>Customer sends payment message</td>
</tr>
<tr>
<td>begin\textsubscript{\textit{od}} (Customer-Acquirer)</td>
<td>Acquirer sends receipt message</td>
</tr>
<tr>
<td>end\textsubscript{\textit{od}} (Acquirer-Merchant)</td>
<td>Acquirer sends receipt message</td>
</tr>
<tr>
<td>101</td>
<td>Customer sends (l)-th message</td>
</tr>
<tr>
<td>PI\textsubscript{\textit{end}} (Merchant-Customer)</td>
<td>Merchant sends OK\textsubscript{\textit{block}} (Customer) in additional message CM message</td>
</tr>
<tr>
<td>PI\textsubscript{\textit{end}} (Acquirer-Customer)</td>
<td>Acquirer sends (m)-th message</td>
</tr>
<tr>
<td>PI\textsubscript{\textit{end}} (Acquirer-Merchant)</td>
<td>Merchant sends (m)-th message, OK\textsubscript{\textit{block}} (Acquirer)</td>
</tr>
<tr>
<td>PI\textsubscript{\textit{end}} (Acquirer-Merchant)</td>
<td>Acquirer sends OK\textsubscript{\textit{block}} (Merchant) in additional message AM message</td>
</tr>
<tr>
<td>PI\textsubscript{\textit{end}} (Acquirer-Merchant)</td>
<td>Customer sends (n)-th message, OK\textsubscript{\textit{block}} (Acquirer)</td>
</tr>
<tr>
<td>PI\textsubscript{\textit{end}} (Customer-Merchant)</td>
<td>Customer sends OK\textsubscript{\textit{block}} (Customer) in additional message AC message</td>
</tr>
</tbody>
</table>

5.3. Definition of money accountability

Generally in electronic payment protocols one type is that the customer first pays the money then the merchant sends the goods to customer. The other is that the merchant first sends the goods to customer, and then the customer pays the money. Meng et al. automatic model is based on the first category.

If \(\textit{EPP}'\) is a SEPP and \(\textit{EPP}''\) satisfies that: \textit{end\textsubscript{\textit{od}}} (Merchant-Customer), \textit{end\textsubscript{\textit{od}}} (Acquirer-Customer) and \textit{end\textsubscript{\textit{od}}} (Acquirer-Merchant) are true; at the same time \(\textit{EPP}''\) also satisfies the following correspondence:

1. \textit{end\textsubscript{\textit{od}}} (Merchant-Customer) \(\Rightarrow\) begin\textsubscript{\textit{od}} (Customer-Merchant)
2. \textit{end\textsubscript{\textit{od}}} (Acquirer-Customer) \(\Rightarrow\) begin\textsubscript{\textit{od}} (Customer-Acquirer)
3. \textit{end\textsubscript{\textit{od}}} (Acquirer-Merchant) \(\Rightarrow\) begin\textsubscript{\textit{od}} (Merchant-Acquirer)

With public variables \(V = \emptyset\), then \(\textit{EPP}\) is SEPP with session functions (sessionid and sessionid') with money accountability. \(\textit{EPP}''\) process can be obtained from \(\textit{EPP}\) by adding the following events in the following Table 1.
5.4. Definition of goods accountability

In order to use the correspondence to model the goods accountability, we classify the electronic payment protocols into two categories: one is that the customer agrees on order description, then the merchant agrees on it; the other is that merchant agrees on order description, then the customer the order description. Meng et al. automatic model is based on the first category.

If \(EPP'\) is a SEPP and \(EPP'\) satisfies that \(\text{endevent}_{\text{Merchant-Customer}}\) is true ,at the same time \(EPP'\) satisfies the correspondence:

\[
\text{endevent}_{\text{Merchant-Customer}} \Rightarrow \text{enginevent}_{\text{Customer-Merchant}}
\]

with public variables \(V = \emptyset\), then \(EPP\) is a SEPP with session functions (sessionid and sessionid') with goods accountability. \(EPP'\) process can be obtained from \(EPP\) by adding the following events in the following Table 1.

6. Modeling 3KP protocol in Blanchet calculus

6.1. 3KP protocol

iKP [2] is credit-card based ecommerce payment protocol. The protocol step of iKP is similar to that of SET. iKP is a family of protocol in that it consists of three types of protocol, which depends on the number of certificate of the engaging party.3KP protocol is one of the families of iKP electronic payment protocols.3KP protocol consists of customer who will make the payment, merchant who will receive the money and acquirer which will withdraw the money from the account of customer to account of merchant. They claim that 3KP protocol has accountability, privacy, anonymity. The technologies applied by 3KP protocol mainly include symmetric encryption, asymmetric encryption, hash function and digital signature. It uses symmetric techniques and asymmetric techniques to guarantee data confidentiality and use digital signature to implement message integrity, consistency and accountability.

In 3KP protocol, an acquirer Acquirer has a private key, \(SK_a\), which is used to generate signature and to recover the plaintext from the ciphertext, and public key \(PK_a\) which is used to verify the signature and to encrypt the plaintext. At the same time the customer Customer and merchant Merchant has the private key \(SK_c\), \(SK_m\) and public key \(PK_c\), \(PK_m\), respectively. \(H(\bullet)\) is a strong collision-resistant one-way hash function. \(\text{encrypt}(PK, \bullet)\) is a public-key encryption function with using \(PK\). \(\text{Sign}(SK, m)\) is a signature function with \(SK\), and including the signed information \(m\). Before the execution of 3KP protocol Customer has the knowledge \(\{\text{PIN,SK}_c,PK_c,PK_m,\text{CERT}_m\}\), Merchant has the knowledge \(\{\text{ID}_m,\text{CERT}_m,SK_m,PK_m,PK_a,\text{CERT}_a\}\), Acquirer has the knowledge \(\{SK_a,PK_a,PK_c,\text{CERT}_a\}\).

1. When Customer make an order Merchant replay it with InitMessage= \(\{\text{CERT}_m,\text{Offer,CERT}_m\}\). InitMessage mainly includes certificate \(\text{CERT}_m\) of Merchant, the offer information Offer and certificate \(\text{CERT}_a\) of Acquirer. Offer= \(\{\text{Desc,Amnt,Cur,Dat,ID}_a\}\) consist of offer description Desc, amount Amnt, currency Cur, date Dat and merchant of identification ID_a.

2. InitMessage activates Customer. Customer checks certificate \(\text{CERT}_a\) of Merchant. Then he constructs \(\text{SLIP}=\{\text{ODesc,OCur,ODat,ID}_a,\text{PIN CardIn,H(Order)}\}\) which is the payment information and consist of order description \(\text{ODesc}\), amount \(\text{OAmnt}\), currency \(\text{OCur}\), date \(\text{ODat}\), merchant of identification \(\text{ID}_a\), card information CardIn and hash code of \(\text{Order}\). After that Customer encrypt \(\text{SLIP}\) with the public key of \(\text{Acquirer}\) and gets the ciphertext \(\text{encrypt}(PK_a, \text{SLIP})\) of \(\text{SLIP}\). Customer froms \(\text{Order}=\{\text{ODesc, OAmnt,OCur,ODat,ID}_a,\text{Addr}\}\) which consists order description \(\text{ODesc}\), amount \(\text{OAmnt}\), currency \(\text{OCur}\), date \(\text{ODat}\) and merchant of identification \(\text{ID}_a\), and delivery address \(\text{Addr}\). After that he generates the hash code \(H(\text{Order})\) of Order. AAfter that Customer generates the digital signature
Finally, Customer forms order payment request 

\[ \text{OPreq} = \{\text{CERT}, \text{Order}, \text{Sign}[\text{SK}_c, \text{encrypt}[\text{PK}_a, \text{SLIP}], \text{H} (\text{Order})] \} \]

and sends it to Merchant.

3. Merchant checks the Order matches his Offer. Then he generates the digital signature

\[ \text{Sign}[\text{SK}_m, \text{Sign}[\text{SK}_c, \text{encrypt}[\text{PK}_a, \text{SLIP}], \text{H} (\text{Order})]] \]

of \( \text{Sign}[\text{SK}_c, \text{encrypt}[\text{PK}_a, \text{SLIP}], \text{H} (\text{Order})] \). After that Merchant forms authorization capture request message

\[ \text{AuthCapReq} = \{\text{CERT}^2, \text{CERT}, \text{Sign}[\text{SK}_m, \text{Sign}[\text{SK}_c, \text{encrypt}[\text{PK}_a, \text{SLIP}], \text{H} (\text{Order})] \} \]

and sends it to Acquirer.

4. Acquirer receives \( \text{AuthCapReq} \) and extracts the digital signature

\[ \text{Sign}[\text{SK}_e, \text{Sign}[\text{SK}_c, \text{encrypt}[\text{PK}_a, \text{SLIP}], \text{H} (\text{Order})]] \]

with public key \( \text{PK}_a \) of Customer. Then he decrypts \( \text{encrypt}[\text{PK}_a, \text{SLIP}] \) with private key \( \text{SK}_e \) of Acquirer and gets \( \text{SLIP} \). After that Acquirer gets \( \text{H} (\text{Order}) \) and ID \( \text{ID}_m \) of merchant from \( \text{SLIP} \) and checks that these match the value \( h \) and the ID sent by the merchant. Using amount \( \text{OAmt} \), currency \( \text{OCur} \), date \( \text{ODat} \), credit card number, expiration date, and PIN, Acquirer makes the payment. After success of the transaction he computes ciphertext \( \text{encrypt}[\text{PK}_a, \text{SLIP}] \) of \( \text{SLIP} \) with public key \( \text{PK}_a \) of Acquirer. Then he forms

\[ \text{AUTH} = \{\text{Result}, \text{H} (\text{OAmt}, \text{OCur}, \text{ODat}, \text{ID}_a), \text{H} (\text{Order})\} \]

consist of result \( \text{Result} \), hash code \( \text{H} (\text{OAmt}, \text{OCur}, \text{ODat}, \text{ID}_a) \) of \( \text{OAmt}, \text{OCur}, \text{ODat}, \text{ID}_a \) and hash code of Order. Finally, Acquirer generates the authorization payment response

\[ \text{AuthCapRes} = \{\text{Sign}[\text{SK}_e, \text{AUTH}, \text{encrypt}[\text{PK}_a, \text{SLIP}]] \] with private key \( \text{SK}_e \) and sends it to Merchant.

Merchant receives \( \text{AuthCapRes} \). He checks the validity of the digital signature

\[ \text{Sign}[\text{SK}_m, \text{AUTH}, \text{encrypt}[\text{PK}_a, \text{SLIP}]] \], based on the already known data AUTH and encrypt[PK_a, SLIP]. If the signature is valid and the payment is success then he transmits

\[ \{\text{Sign}[\text{SK}_m, \text{AUTH}, \text{encrypt}[\text{PK}_a, \text{SLIP}]]) \] to Customer. At the same time, Merchant generates the digital signature

\[ \text{Sign}[\text{SK}_m, \text{Sign}[\text{SK}_c, \text{encrypt}[\text{PK}_a, \text{SLIP}], \text{H} (\text{Order})]] \]

of information got from Customer with private key \( \text{SK}_m \) of Merchant and sends it to Customer

\[ \text{AuthPayCon} = \{\text{Sign}[\text{SK}_m, \text{AUTH}, \text{encrypt}[\text{PK}_a, \text{SLIP}]] \} || \{\text{Sign}[\text{SK}_c, \text{encrypt}[\text{PK}_a, \text{SLIP}], \text{H} (\text{Order})]] \}

Customer gets the authorization payment confirmation information \( \text{AuthPayCon} \), then he verifies the digital signatures

\[ \text{Sign}[\text{SK}_a, \text{AUTH}, \text{encrypt}[\text{PK}_a, \text{SLIP}]] \] with public key \( \text{PK}_a \) of Acquirer and

\[ \text{Sign}[\text{SK}_a, \text{Sign}[\text{SK}_c, \text{encrypt}[\text{PK}_a, \text{SLIP}], \text{H} (\text{Order})]] \] with public key \( \text{PK}_a \) of Merchant. If the verification is valid then Customer confirms that the Acquirer withdraws his money from his account and deposits it to account of Merchant. The payment ends. Merchant sends the goods to Customer.

6.2. Cryptographic assumptions

- expand UP_CMA_signature(keyseed, pkey, skey, value, value, seed, skgen, sign, check, Psign, Psigncoll).
- expand IND_CCA2_public_key_enc(keyseed, pkey, skey, value, value, seed, skgen1, pkeygen1, enc, dec, injbot, Z, Penc, Pencoll).
- expand ROM_hash(value, value, H, hashoracle, qH).

Figure 2. Cryptographic assumptions
In our computational analysis model, we assume that; public-key encryption is assumed to be indistinguishability under adaptive chosen ciphertext attacks. Public key signature is assumed to be unforgeable against chosen-message attack. Hash function is in the random oracle model. Figure 2 describes the cryptographic assumptions in our analysis of 3KP protocol.

### 6.3. Processes

![Figure 3. EPPof3KP process](image)

The complete formal model of 3KP protocol in Blanchet calculus is given in Figures. Figure 3 to 7 report the basic process include EPPof3KPprocess, Initprocess, CustomerProcess, MerchantProcess and AcquirerProcess in money accountability and goods accountability forming our model of 3KP protocol. The process EPPof3KPprocess in Fig.4 is assumed to run in interaction with an adversary, which also models the network.

Initprocess generates customer’s public key \( pkC \) and private key \( skC \), merchant’s public key \( pkM \) and private key \( skM \), acquirer’s public key \( pkA \) and private key \( skA \) which are used to encrypt and customer’s public key \( pkC1 \) and private key \( skC1 \), merchant’s public key \( pkM1 \) and private key \( skM1 \), acquirer’s public key \( pkA1 \) and private key \( skA1 \) which are used to provide digital signature.

Initprocess in Figure 4 firstly generates the customer’s public key \( pkC \) and private key \( skC \) in the following procedure: Initprocess receives an empty message on channel \( start \), sent by the adversary. Then, it chooses randomly with uniform probability a bitstring \( rc \) in the type \( keyseed \), by the construction \( new \ rc : keyseed \). A type \( T \), such as \( keyseed \), aims at denoting a set of bitstrings. However, the considered set of bitstrings depends on the security parameter \( h \), which determines the length of keys. Then, Initprocess generates the customer’s public key \( pkC \) corresponding to the coins \( rc \), by calling the public-key generation algorithm \( pkgen(rc) \). Similarly, Initprocess generates the secret key \( skC \) by calling \( skgen(rc) \). It outputs the public key \( pkC \) on channel \( start \), so that the adversary has customer’s public key \( pkC \).

![Figure 4. Initprocess](image)

After that Initprocess generates the merchant’s public key \( pkM \) and private key \( skM \) in the following
procedure: \texttt{Initprosess} chooses randomly with uniform probability a bitstring \( rm \) in the type \texttt{keyseed} by the construction new \( rm : \texttt{keyseed} \). Then, \texttt{Initprosess} generates the merchant’s public key \( pk_M \) corresponding to the coins \( rm \), by calling the public-key generation algorithm \( \texttt{pkgen}(rm) \). Similarly, \texttt{Initprosess} generates the secret key \( sk_M \) by calling \( \texttt{skgen}(rm) \). It outputs the public key \( pk_M \) on channel \texttt{start} , so that the adversary has customer’s public key \( pk_A \).

Then, that \texttt{Initprosess} generates the acquirer’s public key \( pk_A \) and private key \( sk_A \) in the following procedure: \texttt{Initprosess} chooses randomly with uniform probability a bitstring \( ra \) in the type \texttt{keyseed} by the construction new \( ra : \texttt{keyseed} \). Then, \texttt{Initprosess} generates the acquirer’s public key \( pk_A \) corresponding to the coins \( ra \), by calling the public-key generation algorithm \( \texttt{pkgen}(ra) \). Similarly, \texttt{Initprosess} generates the secret key \( sk_A \) by calling \( \texttt{skgen}(ra) \). It outputs the public key \( pk_A \) on channel \texttt{start} , so that the adversary has customer’s public key \( pk_A \).

After outputting this message, the control passes to the receiving process, which is part of the adversary. Several processes are then made available, which represent the roles of customer, merchant and acquirer in the protocol: the process \( \{ \texttt{CustomerProcess} \} \{ \texttt{MerchantProcess} \} \{ \texttt{AcquirerProcess} \} \). The replication \( \{ \texttt{CustomerProcess} \} \{ \texttt{MerchantProcess} \} \{ \texttt{AcquirerProcess} \} \) is the parallel composition of \( \{ \texttt{CustomerProcess} \} \{ \texttt{MerchantProcess} \} \{ \texttt{AcquirerProcess} \} \) and \( \{ \texttt{AcquirerProcess} \} \). The replication \( \{ \texttt{CustomerProcess} \} \) represents \( n \) copies of the process \texttt{CustomerProcess} indexed by the replication index \texttt{customer} ; The replication \( \{ \texttt{MerchantProcess} \} \) represents \( n \) copies of the process \texttt{MerchantProcess} indexed by the replication index \texttt{merchant} ; The replication \( \{ \texttt{AcquirerProcess} \} \) represents \( n \) copies of the process \texttt{AcquirerProcess} indexed by the replication index \texttt{acquirer}.

The process \texttt{CustomerProcess} in Figure 5 begins with an input on channel \texttt{c1} which is abbreviation of \( c1 \), \texttt{Customer} ; the channel is indexed with \texttt{customer} so that the adversary can choose which copy of the process \texttt{CustomerProcess} receives the message by sending it on channel \texttt{c1} for the appropriate value of \texttt{customer}.

\texttt{CustomerProcess} receives payment initiation message \texttt{Customer} and \texttt{C\_Offer} sent by merchant through channel \texttt{c1} and check it. If the check is successes \texttt{CustomerProcess} randomly chooses with uniform probability a value \texttt{CardIn} in type \texttt{value} by construct new \texttt{CardIn} value . At the same time creates \texttt{BIN} in type \texttt{value} by construct new \texttt{BIN} value and \texttt{Addr} in type \texttt{value} by construct new \texttt{Addr} value. After that it forms order \texttt{Order} by construct \texttt{Order} :value=\texttt{concat}2\texttt{ODesc},\texttt{OAmnt},\texttt{OCur},\texttt{ODat},\texttt{Merchant},\texttt{Addr}) and payment information \texttt{SLIP} by construct \texttt{SLIP} :value=\texttt{concat}4\texttt{OAmnt},\texttt{OCur},\texttt{ODat},\texttt{Merchant},\texttt{PIN},\texttt{CardIn},\texttt{H(Order)} . Order :value=\texttt{concat}2\texttt{ODesc},\texttt{OAmnt},\texttt{OCur},\texttt{ODat},\texttt{Merchant},\texttt{Addr} consists of offer description \texttt{ODesc} , amount \texttt{OAmnt} , currency \texttt{OCur} , date \texttt{ODat} , merchant \texttt{Merchant} and address \texttt{Addr} . \texttt{SLIP} :value=\texttt{concat}4\texttt{OAmnt},\texttt{OCur},\texttt{ODat},\texttt{Merchant},\texttt{PIN},\texttt{CardIn},\texttt{H(Order)} consists of order amount \texttt{OAmnt} , currency \texttt{OCur} , date \texttt{ODat} , merchant \texttt{Merchant} , card information \texttt{CardIn} and hash code of \texttt{Order} . \texttt{CustomerProcess} uses the public key \texttt{pk_A} of acquirer to encrypt \texttt{SLIP} and gets the ciphertext \texttt{enc(SLIP, }pk_{A}\texttt{, }A_{r1}\texttt{)} . He also form the hash code \texttt{H(Order)} of \texttt{Order}. Then \texttt{CustomerProcess} construct \texttt{ENCASL\_HO} by construct \texttt{ENCASL\_HO} :value=\texttt{concat}3\texttt{enc(SLIP, pk_{A}\texttt{, }A_{r1}\texttt{, }H(Order))} . At the same time it generates the digital signature \texttt{sign(ENCASL\_HO, sk_C, r2)} of \texttt{ENCASL\_HO} with his private key \texttt{sk_C} . Finally he forms \texttt{OPreq} by the construct \texttt{OPreq} :value=\texttt{concat}3\texttt{ENCASL\_HO, SIGNC\_ENCASL\_HO} . Then \texttt{CustomerProcess} sent \{ \texttt{Merchant, Order, OPreq} \} by channel \texttt{c2}.
let CustomerProcess =
  (*3*)\(c1(=\text{Customer}, C_\text{Offer}:\text{value})\);
  let concat1(ODesc: value, OAmt: value, OCurr: value, ODate: value, = Merchant) = C_\text{Offer} in
  new CardIn: \text{value};
  new Addr: \text{value};
  new PIN: \text{value};
  let Order : \text{value} = concat2(ODesc, OAmt, OCurr, ODate, Merchant, Addr) in
  let SLIP: \text{value} = concat4(OAmt, OCurr, ODate, Merchant, PIN, CardIn, H(Order)) in
  new r1: seed;
  let ENC ASL_HO: value = concat3(enc(SLIP, pk_A, r1), H(Order)) in
  new r2: seed;
  let SIGNC ENC ASL_HO: value = sign(ENC ASL_HO, skC, r2) in
  let OPreq: \text{value} = concat3(ENC ASL_HO, SIGNC ENC ASL_HO) in
  event Begin OD Customer(Merchant);
  event Potion Customer(Merchant);
  event Potion Customer(Acquirer);
  event Begin PI Customer(Merchant);
  event Begin PI Customer(Acquirer);
  (*4*)\(c2<\text{Merchant}, \text{Order}, \text{OPreq}>;\)

  (*11*)\(c5(=\text{Customer}, C_\text{AuthPayCon}:\text{value})\);
  let concat3(C_M_\text{AuthCapRes: value}, A_\text{AuthCapReq: value}) = C_\text{AuthPayCon} in
  let concat3(C_AUTH ENCA SLIP: value, C_SIGNA AUTH ENCA SLIP: value) = C_M_\text{AuthCapRes} in
  if check(C_AUTH ENCA SLIP, pkA, C_SIGNA AUTH ENCA SLIP) then (*verify signature of A*)
    event Whole Customer(Acquirer);
  let concat3(C_OPreq: value, C_SIGNM OPreq: value) = A_\text{AuthCapReq} in
  find j1 <= N suchthat defined(AuthPayCon[j1]) &&\ (AuthPayCon[j1] = C_\text{AuthPayCon})
  (*verify signature of M*)
  event Whole Customer(Merchant);
  (*12*)\(c6<>.

  Figure 5. CustomerProcess

CustomerProcess receives authorization payment confirmation information Customer and C_\text{AuthPayCon}, then he constructs the message C_\text{AuthPayCon} by construct concat3(C_M_\text{AuthCapRes: value}, A_\text{AuthCapReq: value}) = C_\text{AuthPayCon}. At the same time he also forms C_M_\text{AuthCapRes} by construct concat3(C_AUTH ENCA SLIP: value, C_SIGNA AUTH ENCA SLIP: value) = C_M_\text{AuthCapRes}. Then he verifies the digital signatures C_AUTH ENCA SLIP with public key PK_A of Acquirer. Then he forms A_\text{AuthCapReq} by the construct concat3(C_OPreq: value, C_SIGNM OPreq: value) = A_\text{AuthCapReq}. Finally he verifies the digital signature of Merchant by find j1 <= N suchthat defined(AuthPayCon[j1]) &&\ (AuthPayCon[j1] = C_\text{AuthPayCon}) If the verification is valid then Customer confirm that the Acquirer withdraws his money from his account and deposits it to account of Merchant.
The process \texttt{MerchantProcess} in Figure 6 begins with an input on channel \texttt{c0} which is abbreviation of \texttt{c0[i]}; the channel is indexed with \texttt{i} so that the adversary can choose which copy of the process \texttt{MerchantProcess} receives the message by sending it on channel \texttt{c0[i]} for the appropriate value of \texttt{i}. The situation is similar for \texttt{CustomerProcess} and \texttt{AcquirerProcess}, which expects a message on channel \texttt{c0} which is abbreviation of \texttt{c1[Customer]} . At the same time for \texttt{AcquirerProcess}, which expects a message on channel \texttt{c5} which is abbreviation of \texttt{c5[Acquirer]} . The adversary can then run each copy of \texttt{CustomerProcess} or \texttt{AcquirerProcess}.

Then it generates Desc in the type value , by the construction new Desc:value. At the same time it also generates \texttt{Amt} in the type value , by the construction new Amt:value. At the same time he also generates \texttt{Cur} in the type value , by the construct new Cur:value and \texttt{Dat} in the type value by the construct new Dat:value. Finally \texttt{MerchantProcess} forms the offer Offer by Offer : value = concat1(Desc,Amt,Cur,Dat,Merchant) and sends Customer,Offer by channel \texttt{c1}.

\texttt{MerchantProcess} gets Merchant , M_Order and M_OPreq by channel \texttt{c2} and checks it. Then \texttt{MerchantProcess} forms \texttt{M_Order} by construct concat2(=Desc,=Amt,=Cur,=Dat,=Merchant,Addr:value)=M_Order . \texttt{M_Order} consists of offer description Desc amount Amt , currency Cur , date Dat ,merchant Merchant and address Addr . He also forms \texttt{M_OPreq} by construct concat3(M_ENCASL_HO:value,M_SIGNC_ENCASL_HO:value)=M_OPreq . \texttt{M_OPreq} consists
of \text{M\_ENCASL\_HO} and \text{M\_SIGNC\_ENCASL\_HO}. Then he verifies the digital signature \text{M\_ENCASL\_HO} with the public key \text{pkC} of customer by construct check(\text{M\_ENCASL\_HO},\text{pkC} \text{M\_SIGNC\_ENCASL\_HO}). If the verification is true then he generates the digital signature \text{SIGNM\_M\_OPreq} of \text{M\_OPreq} by construct \text{sign(M\_OPreq,skM,r3)}, then he generates the authorization capture request \text{AuthCapReq} by construct concat3(\text{M\_OPreq},\text{SIGNM\_M\_OPreq}) and send it by channel \text{c3}.

\text{MerchantProcess} receives \text{Merchant} and \text{M\_AuthCapRes} by channel \text{c4} and checks it. Then he also himself generates authorization payment response \text{M\_AuthCapRes} by construct concat3(\text{M\_AUTH\_ENCA\_SLIP:value},\text{M\_SIGNA\_AUTH\_ENCA\_SLIP:value})=\text{M\_AuthCapRes}. \text{M\_AuthCapRes} consists of \text{M\_AUTH\_ENCA\_SLIP} and \text{M\_SIGNA\_AUTH\_ENCA\_SLIP}. Then he verifies \text{M\_AUTH\_ENCA\_SLIP} with the public key \text{pkA} of acquirer by check(\text{M\_AUTH\_ENCA\_SLIP}, \text{pkA} \text{M\_SIGNA\_AUTH\_ENCA\_SLIP}). If the verification is success then he forms authorization payment confirmation information \text{AuthPayCon} by construct concat3(\text{M\_AuthCapRes},\text{AuthCapReq}) and send it by channel \text{c5}.

\begin{verbatim}
let AcquirerProcess =
  (**7**)c3(\text{Acquirer},\text{A\_AuthCapReq:value});
let concat3(A\_M\_OPreq:value,A\_SIGNM\_M\_OPreq:value)=A\_AuthCapReq in
if check(A\_M\_OPreq,\text{pkM} A\_SIGNM\_M\_OPreq) then
  (*verify signature of M*)
  event Potion_Acquirer(\text{Merchant});
  event end_PI_Acquirer(\text{Merchant});
let concat3(A\_M\_ENCASL\_HO:value,A\_M\_SIGNC\_ENCASL\_HO:value)=A\_M\_OPreq in
if check(A\_M\_ENCASL\_HO, \text{pkC} A\_M\_SIGNC\_ENCASL\_HO) then
  (*verify signature of C*)
  event Potion_Acquirer(\text{Customer});
  event end_PI_Acquirer(\text{Customer});
let concat3(A\_ENCASL:value,A\_HO:value)=A\_M\_ENCASL\_HO in
let injbot(A\_SLIP:value)=dec(A\_ENCASL,sk\_A) in
let concat4(A\_OAmt:value,A\_OCur:value,A\_ODat:value,=\text{Merchant},A\_PIN:value,A\_CardIn:value,=\text{A\_HO})=A\_SLIP in
new Result:bool;
let AUTH=concat5(Result, H(concat6(A\_OAmt,A\_OCur,A\_ODat,\text{Merchant})), A\_HO) in
let AUTH\_ENCA\_SLIP:value=concat3(AUTH, A\_ENCASL) in
new r4:seed;
let AuthCapReq:value=concat3(AUTH\_ENCA\_SLIP,\text{sign(AUTH\_ENCA\_SLIP,skA,r4)}) in
event Whole_Acquirer(\text{Merchant});
  event Whole_Acquirer(\text{Customer});
(*8*)c4<\text{Merchant},\text{AuthCapRes}>. (*profuce AUTH, and be signed by skA*)

\end{verbatim}

\textbf{Figure 7. AcquirerProcess}

\text{AcquirerProcess} in \textbf{Figure 7} receives \text{Acquirer} and \text{A\_AuthCapReq} by channel \text{c3}. Then he himself generates authorization payment request \text{A\_AuthCapReq} by construct concat3(\text{A\_M\_OPreq:value},A\_SIGNM\_M\_OPreq:value). \text{A\_AuthCapReq} consist of \text{A\_M\_OPreq} and \text{A\_SIGNM\_M\_OPreq}. \text{AcquirerProcess} verifies the digital signature \text{A\_M\_OPreq} with public key \text{pkM} of merchant by construct check(\text{A\_M\_OPreq}, \text{pkM} A\_SIGNM\_M\_OPreq) . If the verification is success then he verifies the digital signature \text{A\_M\_OPreq} with public key \text{pkC} of customer by construct check(\text{A\_M\_ENCASL\_HO, pkC A\_M\_SIGNC\_ENCASL\_HO}) . \text{A\_M\_OPreq} consists of \text{A\_M\_ENCASL\_HO} and \text{A\_M\_SIGNC\_ENCASL\_HO}. If the verification is true then he generates the authorization payment response \text{AuthCapRes} by construct concat3(AUTH\_ENCA\_SLIP,\text{sign(AUTH\_ENCA\_SLIP,skA,r4)}) . \text{AUTH\_ENCA\_SLIP} is generated by concat3(AUTH, A\_ENCASL) . \text{sign(AUTH\_ENCA\_SLIP,skA,r4)} is the digital signature of
AUTH_ENCA_SLIP with private key $sk_A$ of acquirer. AUTH consists of the result Result of payment, hash code of $\text{concat}(A_{OAmt}, A_{OCur}, A_{ODat}, \text{Merchant})$ and $A_{HO}$. A_SLIP consists of amount $A_{OAmt}$, currency $A_{OCur}$, date $A_{ODat}$, merchant $\text{Merchant}$, personal identification number $A_{PIN}$, card information $A_{CardIn}$, hash code $A_{HO}$ of order. $A_{M\_ENCASL\_HO}$ consists of $A_{ENCASL}$ and $A_{HO}$.

7. Automatic verification of accountability in 3KP protocol with CryptoVerif

CryptoVerif can take two formats as input. The first one is in the form of channels Front-end. The second one is in the form of oracles Front-end. In both cases, the output of the system is essentially the same. The main difference between the two front-ends is that the oracles front-end uses oracles while the channels front-end uses channels. In the channels front-end, channels must be declared by a channel declaration. There is no such declaration in the oracles front-end. Some constructs use a different syntax in the oracles front-end, to be closer to the syntax of cryptographic games.

In this paper we use the form of channels Front-end as the input of CryptoVerif. In order to prove the in 3KP protocol the Blanchet calculus model are needed to be translated into the syntax of CryptoVerif and generated the CryptoVerif inputs in the form of channels Front-end.

Table 2. The correspondences in 3KP have money accountability and goods accountability

<table>
<thead>
<tr>
<th>Correspondences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Potion_Merchant(Customer) =&gt; Potion_Customer(Merchant)</td>
</tr>
<tr>
<td>Whole_Customer(Merchant) =&gt; Whole_Merchant(Customer)</td>
</tr>
<tr>
<td>Potion_Acquirer(Merchant) =&gt; Potion_Merchant(Acquirer)</td>
</tr>
<tr>
<td>Whole_Merchant(Acquirer) =&gt; Whole_Acquirer(Merchant)</td>
</tr>
<tr>
<td>Potion_Acquirer(Customer) =&gt; Potion_Customer(Acquirer)</td>
</tr>
<tr>
<td>Whole_Customer(Acquirer) =&gt; Whole_Acquirer(Customer)</td>
</tr>
</tbody>
</table>

In order to verify the money accountability and goods accountability, according to Meng et al. model, firstly we need to prove 3KP is a SEPP which means that need to prove that authentication between Customer and Merchant, authentication between Acquirer and Merchant, authentication between Acquirer and Customer. Then we need to prove that money accountability and goods accountability. Above all 3KP must satisfy the conditions in Table 2.

Figure 8 gives the code of verification of money accountability and goods accountability in CryptoVerif. The analysis was performed by CryptoVerif and succeeded. The results are showed in Figure 9. According to Meng et al. model of money accountability and goods accountability 3KP protocol is proved to guarantee money accountability and goods accountability in computation model.
let CustomerProcess = channel start,c0,c1,c2,c3,c4,c5,c6.
query event end_PI_Acquirer(Merchant) ==> Begin_PI_Merchant(Acquirer).
query event end_PI_Acquirer(Customer) ==> Begin_PI_Customer(Acquirer).
query event end_PI_Merchant(Customer) ==> Begin_PI_Customer(Merchant).
query event end_PI_Acquirer(Merchant) ==> true.
query event end_PI_Acquirer(Customer) ==> true.
query event end_PI_Merchant(Customer) ==> true.
query event end_OD_Merchant(Customer) ==> true.
query event Potion_Acquirer(Customer) ==> Potion_Customer(Acquirer).
query event Whole_Merchant(Acquirer) ==> Whole_Acquirer(Merchant).
query event Potion_Merchant(Customer) ==> Potion_Customer(Merchant).

let AcquirerProcess = channel start,c0,c1,c2,c3,c4,c5,c6.
query event begin_PI_Acquirer(host).
query event Begin_PI_Acquirer(host).
query event end_PI_Acquirer(host).
query event Potion_Acquirer(Customer).
query event Whole_Acquirer(Merchant).
query event Whole_Acquirer(Customer).

let MerchantProcess = channel start,c0,c1,c2,c3,c4,c5,c6.
query event begin_PI_Customer(host).
query event Begin_PI_Customer(host).
query event end_PI_Customer(host).
query event Potion_Acquirer(Merchant).
query event Whole_Acquirer(Merchant).
query event Whole_Acquirer(Customer).
8. Acknowledgement

This study was supported in part by Natural Science Foundation of The state Ethnic Affairs Commission of PRC under the grants No: 10ZN09.

9. Conclusion

During the past few decades electronic payment protocols has been studied. A lot of electronic payment protocols have been proposed which claimed that have the security properties, for example, accountability, atomicity, anonymity, non-repudiation and fairness. In order to verify the security properties of security protocol include electronic payment protocols and increases the confidence of the people, two approaches: symbolic approach and computational approach have been developed for analyzing security protocols form the beginning of the 1980s. To our best knowledge, there does not existing that analysis of electronic payment protocols and its security properties with automatic tool in computational model until Meng et al. propose the first automatic framework of money accountability and goods accountability based on Blanchet calculus in computational model. In this study, based on Meng et al. mechanized framework we use Blanchet calculus in the computational model to mechanized analyze 3KP protocol with mechanized tool CryptoVerif.

First, the state-of-art of electronic payment protocol and the proof including in symbolic model and in computational model are presented. We find that there does not existing that analysis of electronic payment protocols and its security properties with automatic tool in computational model until Meng et al. propose the first automatic framework of money accountability and goods accountability based on Blanchet calculus in computational model. Second, we apply Meng et al. automatic model for automatically analysis of 3KP protocol. So the term, process and correspondence assertion in Blanchet calculus is used to model money accountability and goods accountability and 3KP protocol. The money accountability and goods accountability are expressed by non-injective or injective correspondence. The analysis itself is performed by automatic tool CryptoVerif developed by Blanchet. Third, the result shows that 3KP electronic payment protocol has money accountability and goods accountability, which is consistent with its claim. To our knowledge, we are conducting the first automatic analysis in computational model of 3KP electronic payment protocol in active adversary.

Further studies concentrate on formal analysis of other electronic payment protocols.

10. References


[22] Bo Meng, Fei Shao, Lin Li, Wei Huang, Dejun Wang, “Automatic Proofs of Deniable Authentication Protocols with a Probabilistic Polynomial Calculus in Computational Model”,

- 87 -