Aspect Oriented Modeling of Impersonation Attack for A Secure Account based Protocol for Mobile Payment

Devendra Mani Tripathi
Department of Computer Engineering
Army Institute of Technology
Pune, India

Nikita Gupta
Department of Computer Engineering
Army Institute of Technology
Pune, India

ABSTRACT

In this paper we propose aspect oriented modeling and verification of a secure account based protocol for mobile payment when the application is under impersonation attack during the purchase phase of the protocol. The proposed work addresses the attack aspect and attack mitigation aspects that are woven inside secure account based protocol for mobile payment. The model proposed in this work gives a better understanding to implementer of secure account based protocol. Eventually the proposed model is verified by Alloy Analyzer to check the security concern at issuer's end. The model is checked for reasonably large scope of Alloy Analyzer without finding any counter example; this verifies the correctness of proposed model.

General Terms

Software engineering, M-Commerce.

Keywords

Impersonation Attack Modeling, Mobile Payment, Aspect Oriented Model.

1. INTRODUCTION

The scope of mobile commerce is constantly expanding so as to reach the un-reached. The exponential growth in mobile and wireless communication has brought forward incredible opportunities for mobile commerce. However the limited processing power, memory, power consumption of mobile devices and security aspects of wireless network are hindering the widespread growth of mobile payment.

Aspect oriented modeling is way to model functional and non-functional requirement of system. This methodology has opened new perceptions for software architects in addressing security concerns as aspects in a software oriented system. Aspect oriented modeling has the advantage that cross cutting concerns can be better handled during design and development phase of software development.

In the present work we used aspect oriented modeling to show the behavior of the secure account based mobile payment protocol [6] when the protocol is under impersonation attack.

The paper is organized as follows. Section II gives a brief overview of the related work on the topic. Section III covers the proposed model. Section IV covers the technique for conversion of simplified class diagram to Alloy model. Finally, Section V presents the results and concludes the paper..

2. RELATED WORK

This section explores the existing payment protocols and their aspect oriented models that are available in literature.

Georg *et al.* [2] have proposed a methodology based on AOM that incorporates the security mechanisms in an online payment application. In the proposed methodology, functionality of the application is described using a primary model, attacks are treated using aspects. Then attack aspect is incorporated into the primary model to obtain the misuse model. Finally security-treated model is analyzed to give assurance that it is resilient to attacks. The authors have considered MITM attack to analyze the performance of the system.

Xu *et al.* [3] have proposed an aspect-oriented approach for the separation and composition of security and functional requirements. It provides a structured way to handle the crosscutting nature of security threats and threat mitigations by incorporating aspect-orientation for requirements analysis. Based on the Use Case driven development, they specify security threats and threat mitigation as aspects that encapsulate pointcuts and advice.

An approach to aspect-oriented modeling and verification with finite state machines was introduced by Xu *et al.* [4]. The approach provides explicit notations for capturing crosscutting concerns and incremental modification requirements with respect to class state models. Aspect models and class models are then woven into the composite aspect-oriented model through a weaving mechanism and transformed into finite state process.

An aspect-oriented framework for building intrusion-aware software systems was proposed by Zhu and Zulkernine [5]. They begin with identifying the vulnerabilities in the target system and then specify the attacks that exploit the vulnerabilities, Attack scenarios and intrusion detection aspects are modeled using an aspect-oriented UML profile..

3. ASPECT ORIENTED MODEL

In this section we will discuss the detailed behavior of the protocol when it is subjected to the impersonation attack. It is assumed that the secure account based protocol [6] is used for mobile payment. UMLIntr [7] is used for the development of aspect oriented diagram. The model also shows the attack mitigation aspect at the issuer's end. This section basically describes the dynamic behavior of the model.

The sequence diagram for mobile payment system under impersonation attack is shown in Figure 1. We are considering a scenario when some illegitimate customer tried to make a payment with using the actual secret key.

The aspect oriented sequence diagram clearly shows that the around advice requestingValidation assists the Issuer to decide whether an authentication should be granted or not. The aspect model show validation aspect only at issuer but it can be used at merchant end also, since we are considering the merchant as honest so validation aspect is shown only at issuer.

This model also shows

1. Customer is working as an attacker because he is not a genuine customer.

2. In addition to this the model also shows aspects, joinpoint, pointcut and advice.

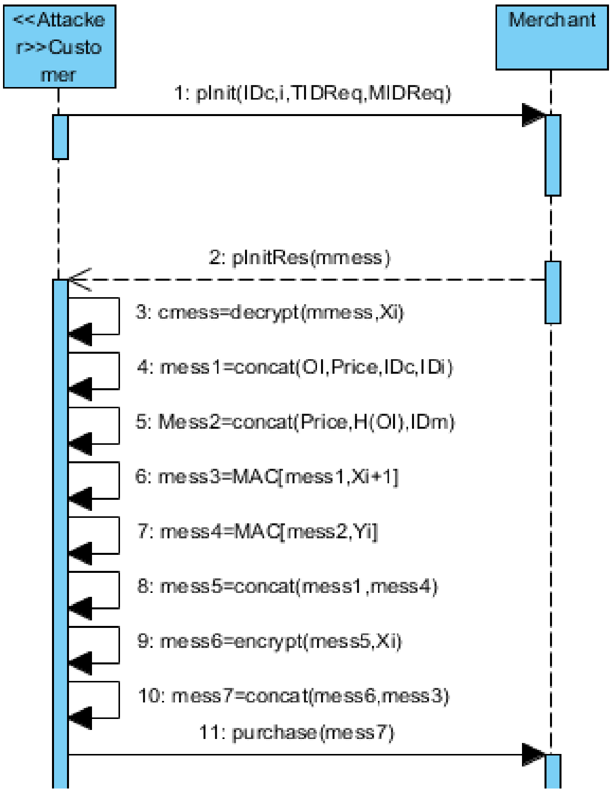
<<Attacker>>Customer
Merchant
1: pInit(IDc,i,TIDReq,MIDReq)
2: pInitRes(mmess)
3: cmess=decrypt(mmess,Xi)
4: mess1=concat(OI,Price,IDc,IDi)
5: Mess2=concat(Price,H(OI),IDm)
6: mess3=MAC[mess1,Xi+1]
7: mess4=MAC[mess2,Yi]
8: mess5=concat(mess1,mess4)
9: mess6=encrypt(mess5,Xi)
10: mess7=concat(mess6,mess3)
11: purchase(mess7)

Figure 1: Sequence diagram for mobile payment system under impersonation attack

Next we model the sequence diagram (Figure 2) of the attack mitigation aspect at the issuer. This sequence diagram demonstrates how the attack mitigation aspect interacts with other entities in the system. The interaction between the two objects is represented as a message denoted by an arrow. A joinpoint is represented as a special message with the <<joinpoint>> stereotype. An advice is represented as a message to the <<aspect>> class with the <<advice>> stereotype. The sequence begins with the merchant sending an authorization request message, pgRequestValidation to the payment gateway. The payment gateway forwards the message as issuerRequestValidation after some processing at his end. Issuer after receiving the issuerRequestValidation, executes an advice requestingValidation, which basically checks for impersonation attack. The advice matches the secret that it receives from customer and verifies its authenticity. If the secret key sent by customer matches with the issuer key then the advice returns res = 0 which means attack is not present, else it return res = 1, that indicates presence of impersonation attack.

If issuer receives res = 0 it processes the request normally by first checking the customer's account balance and based on that he sends yes or no message to the merchant and the customer. If the issuer receives res = 1 from the aspect then it sends an abort message to the customer and the merchant via payment gateway. The abort message means that the transaction has been declined because of some undesirable cause. If the transaction has been aborted then the one time secret key produced during the transaction will not be saved in the issuer and customer database so that it can be used later for some other transaction.

We shall use Alloy Analyzer [8] to verify the security property. For the purpose, we present a simplified class diagram that contains only those components that are essential to verify the security property. This simplified class diagram is derived from the aspect oriented sequence diagrams 1 and 2.

The process of deriving simplified class diagram from aspect oriented sequence diagram cannot be fully automated, but we can follow some guideline to make it partially automated as mentioned below.

- The designer must decide what assertions will be tested using Alloy Analyzer. In our case we need to ensure that if the issuer proceeds with sending an issuerResponseCode to the payment gateway without aborting the authorization request then this implies that the customer is genuine. The formulation of this assertion is influenced by Alloy Analyzer since the tool works by attempting to find a counterexample to the assertion.

- Every message to a different object lifeline in Fig. 1 and Fig. 2 has the potential to become a method of the receiving object, if the object performs some computation of interest as a result of receiving the message. If the receiving object just passes the message through to another object lifeline, the method will exist in the final receiving object. Since the message always exists between two life lines so this list can be created automatically.

- The complete class diagram should show the types of all the attributes and the return types of methods. Since Alloy has no primitive types everything must be declared as a separate type, which will be a set in the Alloy Model. Various enumerated types are defined in the model such as OrderType with the values OI and aOI(aOI is ordered information tempered by the attacker).

- Classes are specified with method named for the message received by the class. For example, Attacker class has a method called pInitfromCustomer which means that the attacker receives a pInit message from customer.

The class diagram obtained from the aspect oriented sequence diagram using the above assumption is shown in Fig. 3. This class diagram acts as the basis of generating an Alloy model, which is shown in the next section.

4. ALLOY MODEL

The next step in the analysis is to convert the simplified class diagram to Alloy model. The analysis is necessary to verify the model's robustness towards attack. The process of conversion of UML to Alloy is mentioned below.

- UML classes and aspects are directly translated to signatures in Alloy.

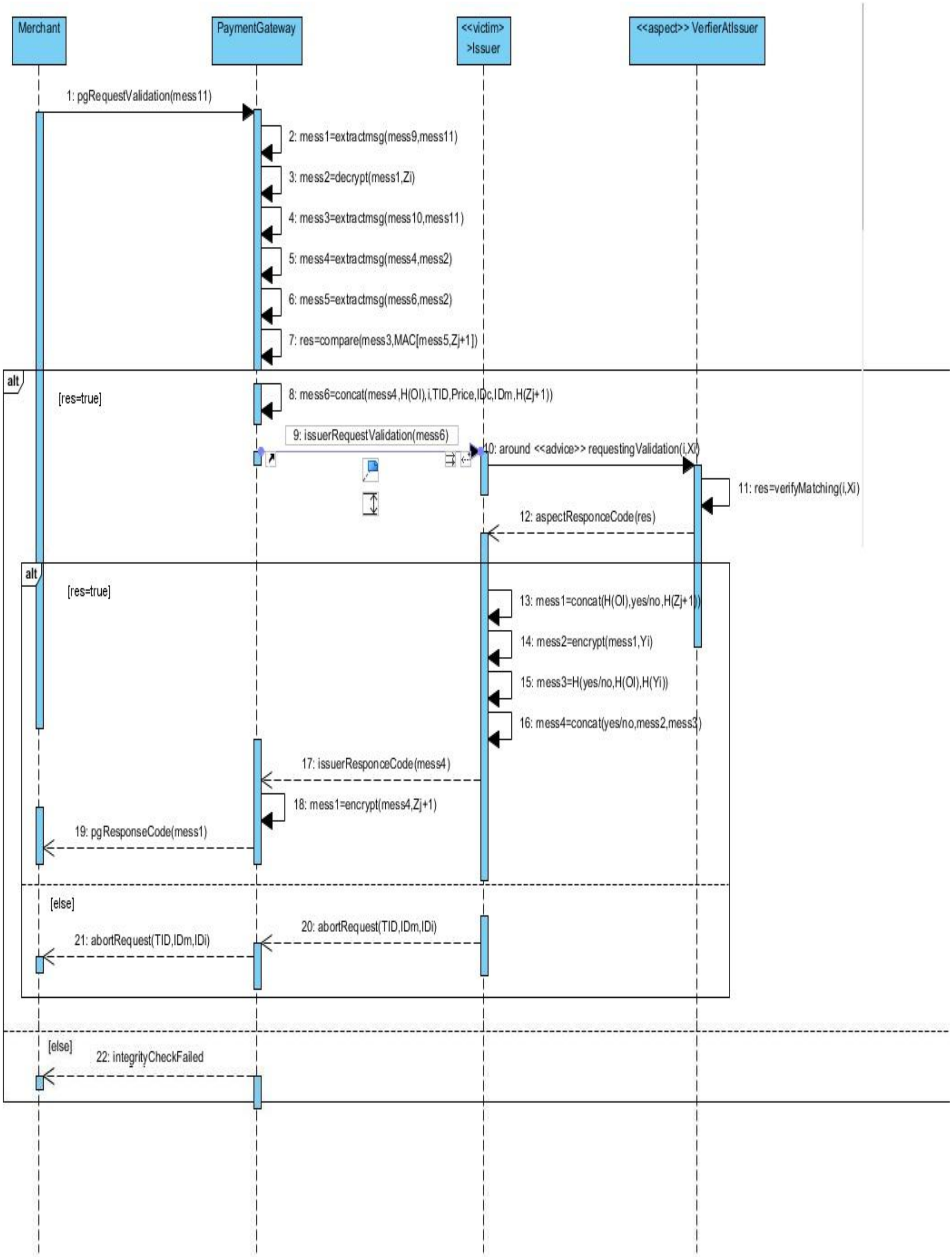
Merchant
PaymentGateway
<<victim>
>Issuer
<<aspect>> VerfierAtIssuer
1: pgRequestValidation(mess11)
2: mess1=extractmsg(mess9,mess11)
3: mess2=decrypt(mess1,Zi)
4: mess3=extractmsg(mess10,mess11)
5: mess4=extractmsg(mess4,mess2)
6: mess5=extractmsg(mess6,mess2)
7: res=compare(mess3,MAC[mess5,Zj+1])
alt
[res=true]
8: mess6=concat(mess4,H(OI),i,TID,Price,IDc,IDm,H(Zj+1))
9: issuerRequestValidation(mess6)
10: around <<advice>> requestingValidation(i,Xi)
11: res=verifyMatching(i,Xi)
12: aspectResponceCode(res)
alt
[res=true]
13: mess1=concat(H(OI),yes/no,H(Zj+1))
14: mess2=encrypt(mess1,Yi)
15: mess3=H(yes/no,H(OI),H(Yi))
16: mess4=concat(yes/no,mess2,mess3)
17: issuerResponceCode(mess4)
18: mess1=encrypt(mess4,Zj+1)
19: pgResponseCode(mess1)
[else]
20: abortRequest(TID,IDm,IDi)
21: abortRequest(TID,IDm,IDi)
[else]
22: integrityCheckFailed

Figure 2: Sequence diagram of the attack mitigation aspect at the issuer

- UML associations are translated to fields of signatures.
- Class attributes are translated to signature fields.
- UML types and enumerators are translated to signatures.
- Methods and advices are translated to Alloy predicates.

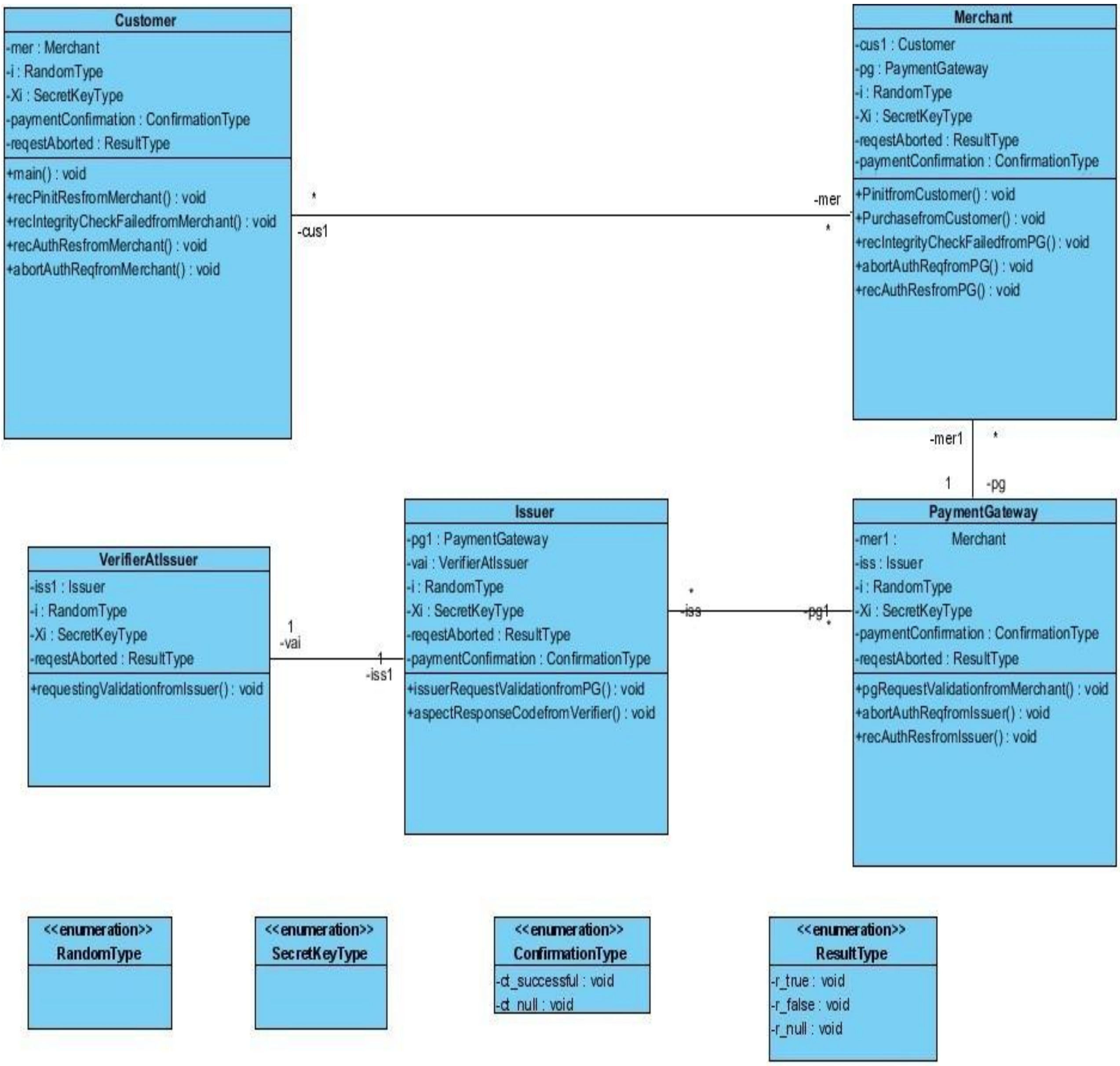
Customer
-mer : Merchant
-i : RandomType
-Xi : SecretKeyType
-paymentConfirmation : ConfirmationType
-reqestAborted : ResultType
+main() : void
+recPinitResfromMerchant() : void
+recIntegrityCheckFailedfromMerchant() : void
+recAuthResfromMerchant() : void
+abortAuthReqfromMerchant() : void
*
-cus1
-mer
*
Merchant
-cus1 : Customer
-pg : PaymentGateway
-i : RandomType
-Xi : SecretKeyType
-reqestAborted : ResultType
-paymentConfirmation : ConfirmationType
+PinitfromCustomer() : void
+PurchasefromCustomer() : void
+recIntegrityCheckFailedfromPG() : void
+abortAuthReqfromPG() : void
+recAuthResfromPG() : void
-mer1
*
1
-pg
VerifierAtIssuer
-iss1 : Issuer
-i : RandomType
-Xi : SecretKeyType
-reqestAborted : ResultType
+requestingValidationfromIssuer() : void
1
-vai
1
-iss1
Issuer
-pg1 : PaymentGateway
-vai : VerifierAtIssuer
-i : RandomType
-Xi : SecretKeyType
-reqestAborted : ResultType
-paymentConfirmation : ConfirmationType
+issuerRequestValidationfromPG() : void
+aspectResponseCodefromVerifier() : void
*
-iss
-pg1
*
PaymentGateway
-mer1 : Merchant
-iss : Issuer
-i : RandomType
-Xi : SecretKeyType
-paymentConfirmation : ConfirmationType
-reqestAborted : ResultType
+pgRequestValidationfromMerchant() : void
+abortAuthReqfromIssuer() : void
+recAuthResfromIssuer() : void
<<enumeration>>
RandomType
<<enumeration>>
SecretKeyType
<<enumeration>>
ConfirmationType
-ct_successful : void
-ct_null : void
<<enumeration>>
ResultType
-r_true : void
-r_false : void
-r_null : void

Figure 3: Class Diagram

The following code shows signatures to represent Customer and VerifierAtIssuer. The signature declaration begins with the keyword ‘sig’ and it consists of various attributes and their types separated by a ‘:’ inside them.

```
sig VerifierAtIssuer{

iss1 : Issuer,

sharedsecret: Mapping,

reqestaborted : ResultType

}

abstract sig  Mapping{

map: RandomType lone->lone SecretType

}

abstract sig  RandomType{}

abstract sig  SecretType{}
```

The desired property that need to be checked is matching the secret key and random number i. Since each random number i generate a unique secret key so this can be chosen for checking the impersonation attack.

```
assert assert1

{

all  cus:Customer  |  cus.paymentconfirmation  =
ct_successful                                   =>
cus.mer.pg.iss.vai.sharedsecret in Mapping

}
```

5. RESULT AND CONCLUSION

Model verifications for all the models were performed on a personal computer with Intel Core 2 Quad CPU 2.83 GHz Q9950 with 3.25 GB of RAM. Table I shows the time taken in the analysis of secure account based protocol for mobile payment system model under impersonation attack.

Table 1. Alloy analysis result

Scope	**Time required For verification**
1 for all model elements	0 ms
5 for all model elements	125 ms
10 for all model elements	1594 ms
20 for all model elements	7203 ms
30 for all model elements	214812 ms

We started analysis with a scope of 1, the result was returned in less than a millisecond and no counter example was found. To increase our confidence the scope of the analysis was increased continuously up to 30, still no counter example was found. Hence we concluded that the model under consideration is robust to impersonation attack.

6. REFERENCES

[1] Juniper Research Forecasts Total Mobile Payments to Grow Nearly Ten Fold by 2013. Available from: www.juniperresearch.com.

[2] G. Georg, I. Ray, K. Anastasakis, B. Bordbar, M. Toahchoodee, and S. H. Houmb, "An aspect-oriented methodology for designing secure applications," Information and Software Technology, vol. 51, no. 5, pp. 846–864 , May 2009.

[3] D. X. Xu, V. Goel, K. E. Nygard, and W. E. Wong, "Aspect-oriented specification of threat-driven security requirements," International Journal of Computer Applications in Technology, vol. 31, no. 1/2, pp. 131-140, March 2008.

[4] D. Xu, O. El-Ariss, W. Xu, and L. Wang, "Aspect oriented modeling and verification with finite state machines," Journal of Computer Science and Technology, vol. 24, no. 5 , pp. 949-961, September 2009.

[5] Z. J. Zhu and M. Zulkernine, "A model-based aspect-oriented framework for building intrusion-aware software systems," Information and Software Technology, vol. 51, no. 5 , pp. 865–875, May 2009.

[6] S. Kungpisdan, B. Srinivasan and P. D. Le, "A secure account based mobile payment protocol," in Proceedings International Conference on Information Technology: Coding and Computing, pp. 35-39, April 5-7 2004.

[7] M. Hussein and M. Zulkernine, "UMLIntr: A UML profile for specifying intrusions," in Proceedings IEEE International Symposium and Workshop on Engineering of Computer Based Systems, pp. 279–286, March 27-30 2006.

[8] Alloy Analyzer. Available from http://alloy.mit.edu/alloy/

[9] D. Xu, M. Tu, M. Sanford, L. Thomas, D. Woodraska, and W. Xu, " Automated Security Test Generation with Formal Threat Models," IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 4, pp. 526-540, July-August 2012.