Modeling and Verification of Payment System in E-Banking

Iqra Obaid
Department of Computer Sciences
COMSATS Institute of Information Technology, (COMSATS) Lahore, Pakistan

Syed Asad Raza Kazmi
Department of Computer Science
Government College University (GCU), Lahore,
Pakistan

Awais Qasim
Department of Computer Science
Government College University (GCU), Lahore,
Pakistan

***Abstract*—Formal modeling and verification techniques have been used to ensure the reliability and accuracy of multiple systems to be verified. In contrast to ordinary testing techniques which exhibit the presence of flaws and errors in a system, formal methods prove their absence. Electronic banking (e-banking) services have become very popular with the escalating development in the information and communication technology. Due to the presence of complexity, an e-banking system requires an efficient security model. One important approach to ensure the reliability and security of the e-banking system is through the use of formal methodologies. This study explores the opportunity of modeling interbank payment system through a case study of 1-link Automated Teller Machine (ATM). A generic verification system SPIN (Simple Promela Interpreter) is, therefore, employed to model and then to verify the integrity and security of payment system in e-banking. Linear temporal logic formulas are further summarized to assure the security of the e-banking system. The principal conclusion of the work includes a complete procedure of verification and modeling of the payment system in 1-link ATMs.**

***Keywords*—E-banking; model checking; Simple Promela Interpreter (SPIN); formal methods; Linear Temporal Logic (LTL) formula; Promela introduction**

I. INTRODUCTION

Software usage is increasing rapidly in all aspects of life and the reliability of these software has become a prime challenge, especially when the safety-critical software [17] are involved where failures often lead to a sudden loss of life, money or valuables. While using an e-banking payment system to make a transaction, it is vital that the software handling the complete process must guarantee the secure end to end transaction as well as the privacy of data to avoid its misuse. Such software is critical and not easy to be handled and developed.

During the earlier few decades, a number of languages [16] have been suggested for the specification and modeling of software oriented problems. The main aim of these languages is to render the behavior of software at the highest level of abstraction than merely as a code. Model checking verifies the correctness properties of finite-state space, where the properties of the current system are often expressed as formulas of temporal logic (TL). Later, efficient algorithms are adopted that traverse the whole model of the system and identify whether the system holds those properties or not. Similarly, testing of payment system over the internet is being conducted from a past few years. A number of models [9] are proposed and various formulas are expressed to verify the integrity and security of the payment system, but there is no considerable and definite work to verify the payment system between multiple banks i.e. 1-Link e-banking. The most promising approach to ensure the security of an e-banking system is based on formal methods and model checking [18]. This model checking approach usually involves following steps: firstly, the payment system is modeled including all the main features, secondly, property oriented language is used to specify the reliability properties, and finally, a reachability graph with all the execution paths is drawn to verify that these paths verify the properties.

The immense challenges are: provision of authentic secure services to the banking customers as well as assurance of veracity and confidentiality of all the information that is exchanged during the process. Therefore, an efficient security model is required which should provide the banking customers with a sense of security in data usage and transactions. It should, also, be responsible for ensuring the security of the overall information or data exchanged/used in the end to end transaction. For this purpose, Simple Promela Interpreter (SPIN) [8], a standard verification tool, is employed in this study to model the system. The language used as input by the SPIN allows creating a high-level system model of many distributed systems using three components: processes, objects, message channels.

In this paper, model verification in e-banking is presented through a case study of verifying 1-Link ATMs using SPIN model checker. The results of verification clearly show that method of model checking is feasible to verify the 1-link ATMs. Furthermore, this paper explains how to employ a model checker to verify and analyze the integrity and security of payment system in e-banking system.

This paper is further structured as follows: Section II explicates the previous studies. The preliminaries required in the rest of paper are described in Section III. Then, in Section IV, 1-link ATM system model, and system properties are presented using EFSM. Section V presents the experiments of verification and discusses the results. The paper finishes with some conclusion and future work.

II. BACKGROUND

Current researches are directed towards the identification of malevolent activities and attacks in e-banking systems. These researchers have introduced the attack techniques in which, currently, only vulnerabilities are focused. In [1] an e-learning model has been implemented for secure exchange of e-content over the network. Later, the model has been formally verified using SPIN which shows that no unreachability state exists, thus, the system is viable. In [2] a protocol is proposed to identify the legitimate user. But it lacks a technique to authenticate already built e-banking systems.

In [3] PIN based ATM authentication method is evaluated and shows how contextual factors like distraction, trust, memorability influence the ATM use. Later on the basis of the findings, several implications are drawn to design an alternative secure ATM authentication system.

In [4], DHCP is presented according to modeling and verification concepts. In [5], a formal method of e-commerce system based on ebXML for the verification is presented which highlights some weaknesses of that protocol due to lack of any complete and clear specifications. In [6], a model-checking approach is applied to examine the features of ad-hoc networks. Therefore, it demonstrates, how model checking and SPIN are appropriate to study the ad-hoc networks system properties. In [7] a case study of web services is presented, and a verification technique based on model checking and SPIN [8] is proposed. The problem in adopting this checking approach is a state-space explosion. On the other hand, multiple approaches are available to combat the problem, which could be categorized as either simplifying the investigating model of the system under consideration by a higher level of abstraction, or reduction of resources consumption in the model-checking process.

In [9] an approach is proposed to verify retail banking system, which was verified in SPIN model checker. It later verifies that the model checking and SPIN are applicable for inspecting a banking system. In [10]-[15] multiple systems are presented which are verified using model checking approaches.

III. PRELIMINARIES

The model checking technique principally depends on modeling a finite state model (FSM) of a current system and then finally checking whether the desired property holds in the system or not. This approach is primarily used in the verification of protocol and hardware verification, but currently, the technique is also used in software systems. For model checking two approaches are often used, firstly, temporal model checking where the finite transition system is used to model the system and temporal logic is used to express the specifications. On the other hand, in the later approach, the automaton is used to present specifications as well as the system. Further, this system is compared to specifications in order to determine whether the behavior confirms specifications or not.

A wide-ranging model checking tools are available and in use, such as, NuSMV2 [16], SPIN, FDR, JAVA Pathfinder and Maria. Among all, the model checker SPIN provides a user-friendly interface and it groups multiple process executions in respective equivalence classes using the theory of partial order reduction and accepts PROMELA for model specification. PROMELA language models the verification models which represent a system abstract, where only those properties are presented that are needed to be verified. PROMELA language consists of three types of objects: asynchronous message channels, processes, and data objects. Variables and message channels can be declared both globally as well as locally in a process whereas processes can only be defined globally. Processes specify the system behavior while variables and channels define the environment where processes occur.

Two basic ways can be used to verify the system using model checker SPIN. The foremost method is to take any current system and on the basis of that system, verification models are built that includes all the behaviors of the system which needs to be verified. The next approach is to construct a verification model that shows all the necessary specifications of the system. Such system models serve as a high-level description of the system under consideration.

The temporal logics used in model checking can be classified into two types: Computational Tree Logic and Linear Temporal Logic. Computational Tree Logic (CTL) is also identified as branching-time logic i.e. its model is a tree structure which is suitable mostly for hardware verification applications; while Linear Temporal Logic (LTL) is known as linear time logic basically used for software verification applications. The model checker SPIN supports LTL for the specification of system properties, which have natural language like statements. Linear temporal logic consists of a few operators, such as “O” (next state), “U” (until), “<>” (eventually), “W” (weak until or unless) and “[]” (always/square). By combining these with Boolean operators, Linear Temporal Logic can be used to define many important properties of a software system under consideration.

IV. FORMAL MODELING OF INTERNET PAYMENT SYSTEM

A. Extended Finite State Machine

ATMs, nowadays, are the most rapidly emerging sensation of the internet banking technology. With the passage of time, the ordinary ATM systems are replaced by 1-link ATM that is linking multiple banks across the countries. Thus, it is not only helping the banks to handle their clients but also the clients to access their bank accounts from anywhere in the world. The main operations of these ATMs, similar to ordinary ATMs, include transaction inquiries, cash withdrawal, cash deposits, account transfers, bills payment and many others. In this section, a model will be presented (including both Promela and EFSM model) of 1-link ATM system. A simple model is designed so that a reduced number of states can be acquired which could be easily managed during formal verification. Particularly, how the PIN or ATM card number or other related details are encrypted or decrypted at various stages during the process, have been ignored.

An ATM is used to login into the bank account using the ATM card and the PIN, to perform the desired operation

against an account (withdraw cash, deposit cash, bill payment, inquire balance, etc.), and finally log off after performing the desired operation. A user always gets three chances to login into the account using some valid PIN, afterwards, if the client/user fails the ATM card is locked by the ATM till it is reset by the bank. But this function is also performed by a regular or an ordinary ATM. The basic difference between an ordinary and a 1-link ATM lies in the use of any ATM card in the ATM i.e. in 1-link ATMs one could perform the transaction from his account using any 1-link ATM whether it belongs to the respective bank or not.

1-link ATM involves four parties, the client or the cardholder, the cash dispenser (terminal), the ATM network (consortium) and the host ATM server. The client interacts with the bank through the cash dispenser or terminal to perform any kind of operation. The cash dispenser first receives a request from the client to perform a specific operation, and the cash dispenser, then, forwards the request to the ATM network or consortium. Consortium after checking the bank details forwards the request to the respective host bank server of which the client holds the account. The host bank server after receiving the request from the ATM Network again gives the response to the ATM network in the form of approval or rejection. So after the response from the bank server, ATM network then forwards this response to the cash dispenser to make an appropriate response to the client.

Some specifications in PROMELA language need to be described of 1-link ATM in order to use SPIN model checker. But before going into the description of properties in PROMELA, we need to model the specifications of 1-link ATM in EFSM (Extended Finite State Machine). Fig. 1 shows the basic specifications of 1-link ATM using EFSM. This model will be further expressed in PROMELA. The variable *loginAttempts* in the EFSM describes the total number of unsuccessful attempts by the client to enter a valid PIN of the ATM card and when the variable is greater or equal to 3 the card is locked by the ATM.

In Fig. 1 the EFSM of the 1-link ATM is presented, which consists of nine states. The label of a transition *RequestWithdrawal/PINOk/LoginAttempts<3* shows that when the client requests for the cash withdrawal then two conditions should be satisfied, i.e. the PIN should be valid and the login attempts should be less than or equal to 3. In this case, only, the state will be changed from "card valid" to "withdraw balance". Similarly, the state transition *Logon/PINInvalid/LoginAttempts>=3* shows that even if the client again tries to login and the Login Attempts become greater or equal to 3 the ATM card will be locked by the ATM, therefore, the state will change from "Re-Logon" to "Card Locked".

On the other hand, the label of transition *WithdrawAmount<AccountBalance/PINOk* shows that when the cardholder/client requests to withdraw amount and the PIN is verified, than the ATM server will check whether the amount to be withdrawn is smaller than the account balance of the client. If so, then the state will be changed to "Transaction Ok" otherwise if *WithdrawAmount>AccountBalance/PINOk* then the state will be changed to "Transaction Invalid".

B. PROMELA Model

A PROMELA program comprises of 3 basic types of objects: asynchronous message channels, processes, and data types. Processes define the behavior of processes, variables are used to store information of the system being modeled and message channels are basically used for modeling the communication between the processes. The syntax of PROMELA allows the creation of multiple processes dynamically which could be synchronized through message channels as PROMELA language is similar to that of C language.

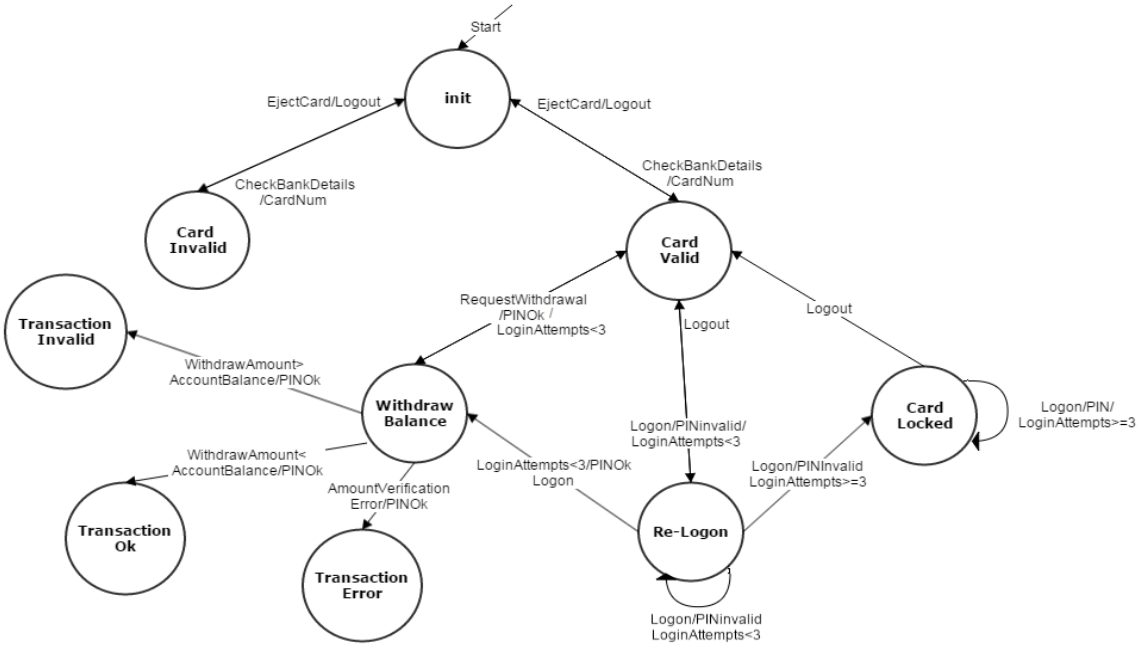

Fig. 1. EFSM of internet payment system.

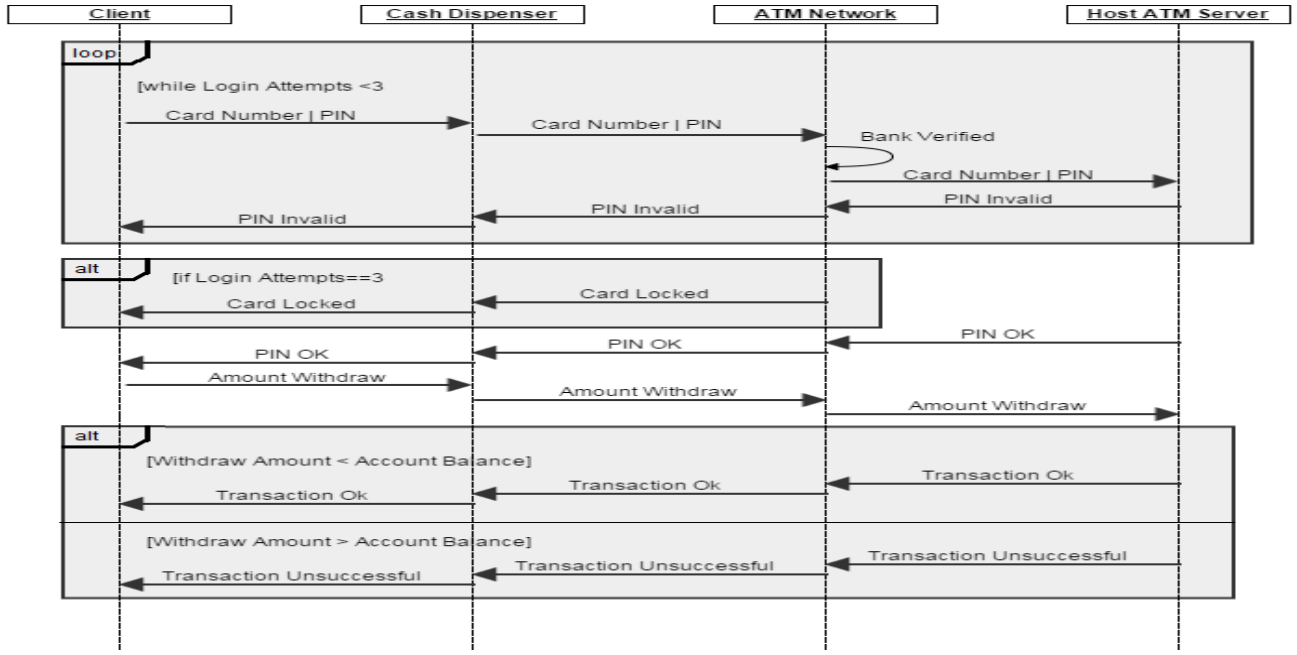

Fig. 2. Sequence diagram of an ATM system.

As EFSM of the system is modeled, the next stage involves the modeling of the system using PROMELA. We need to choose how the system communicates with the four parties. The simple message flow is presented using sequence diagram in Fig. 2. Messages which need to be transferred between the channels are defined as mtype={card,PIN,insertCard, requestLogon, PINvalid, PINinvalid, cardLocked, requestWithdrawal, transactionOk, checkBankDetails, amountInvalid, transactionUnsuccessful, verifyPIN, endTransaction, removeCard, cardInvalid, cardValid, transactionError}. Message channels are used to represent the inter-process communication. The channels are declared using buffer size and data types as shown in Fig. 3.

```
chan cashDispenser_client = [MAX] of {byte,mtype,mtype,byte};
        //Message from ATM to card holder
chan client_cashDispenser = [MAX] of {byte,mtype,mtype,byte};
        //Message from card holder to ATM
chan cashDispenser_atmNetwork = [MAX] of {byte,mtype,mtype,byte};
        //Message from ATM to consortium
chan atmNetwork_cashDispenser = [MAX] of {byte,mtype,mtype,byte};
        //Message from consortium to ATM
chan atmNetwork_hostAtmServer = [MAX] of {byte,mtype,mtype,byte};
        //Message from consortium to ATM server
chan hostAtmServer_atmNetwork = [MAX] of {byte,mtype,mtype,byte};
        //Message from ATM server to consortium
```

Fig. 3. Definitions of channels in PROMELA.

Here the keyword MAX is used to represent the terminal numbers or simply the number of clients or card holders. The channel *client_cashDispenser* sends the messages from the card holder to the cash dispenser or ATM and each of the message have four parts: the first part is of type byte that shows the card number of the card inserted by the client in the ATM, the second field is of type mtype that represents the operations carried out by the card holder to the cash dispenser, the third field represents the amount that relates to the operation and is also of type byte and the last field represents the data send or the response received. Similarly, *cashDispenser_client* sends messages from the cash dispenser/ATM to the client. In this way *cashDispenser_atmNetwork* send messages from the ATM to the ATM Network or the consortium, *atmNetwork_hostAtmServer* send messages from the consortium to the host ATM network, i.e. the network of that bank to which the card belongs and similarly vice versa.

As the model checker SPIN does not allow the user to participate when the process is running, i.e. SPIN runs in closed conditions. Therefore, to traverse all the cases, i.e. in case the PIN is entered incorrect, PIN is entered correct or if the PIN is entered wrong three times and many others, the variables for the amount in the account, the amount withdrawn, PIN is whether OK or not, need to be designed carefully.

For example, if the balance of the account is 4000 and the withdrawal amount is 1000, the SPIN will only traverse the path of transactionOk, while rest of the states, transactionError and transactionUnsuccessful, are not reachable. So just on the base of this case, the result can't be assumed that the system model under consideration does not hold the properties. In addition, if the user enters the wrong PIN 3 times than the path labeled cardLocked will only be traversed by the model checker while the rest of the paths PINOK and PINinvalid are again not reachable. So to model a realistic system, these variables need to be changed. So during the verification of the system, the model is verified against multiple different sets of values. The processes of the card holder, cash dispenser (ATM), consortium and host ATM server respectively are modeled in PROMELA using SPIN. Major functions of each process are represented below. A process of client gets ATM card, check its validity, gets PIN form user.

```
Request_Logon:
if
::atomic{cashDispenser_client?eval(cardNum),requestLogon,
0,PINvalid->
        PINOK=true;
        goto selectOperation}

::atomic{cashDispenser_client?eval(cardNum),requestLogon,
0,PINinvalid->
        PINOK=false;
        printf("You entered an invalid PIN");
        goto Request_Logon}

::atomic{cashDispenser_client?eval(cardNum),requestLogon,
0,cardLocked->
        PINOK=false;
        printf("Card is Locked");
        cardLock=true;
        goto CardLocked}
fi;
```

A process of dispenser dispense cash only when the PIN and the amount entered is valid

```
  cashDispenser_atmNetwork!cardNum,requestLogon,0,PIN;
if
:: atomic{
atmNetwork_cashDispenser?eval(cardNum),requestLogon,0,P
INvalid->
cashDispenser_client!cardNum,requestLogon,0,PINvalid;
        goto Cash_Withdrawal;}
:: atomic{
atmNetwork_cashDispenser?eval(cardNum),requestLogon,0,P
ININvalid->
cashDispenser_client!cardNum,requestLogon,0,PINinvalid;
        goto start;}
:: atomic{
atmNetwork_cashDispenser?eval(cardNum),requestLogon,0,c
ardLocked->
cashDispenser_client!cardNum,requestLogon,0,cardLocked;
        cardLock=true;
       goto start;
}
fi;
```

The process of ATM network refers to the network of the ATM owner bank which checks bank details of the client and then sends the details to the host bank network for PIN verification and other account details verification.

```
Check_Bank_Details:
if
::atmNetwork_cashDispenser?eval(cardNum),requestLogon,0,
PIN->
if
   ::(loginAttempts<3)->
   atmNetwork_hostAtmServer!cardNum,verifyPIN,0,PIN->
  if
        ::atomic{hostAtmServer_atmNetwork?eval(cardNum
),verifyPIN,0,PINvalid->
        atmNetwork_cashDispenser!cardNum,requestLogon,
0,PINvalid;
        goto CashWithdrawal;}
        ::atomic{hostAtmServer_atmNetwork?eval(cardNum
),verifyPIN,0,PINinvalid->
        atmNetwork_cashDispenser!cardNum,requestLogon,
0,PINinvalid;
        loginAttempts=loginAttempts+1;}
      fi;
::atomic{atmNetwork_cashDispenser!cardNum,requestLogon,
cardLocked->
        loginAttempts=loginAttempts+1;
        goto Check_Bank_Details}
fi;
```

The process of host ATM network refers to the network of client host bank which verifies the PIN and other account details.

```
server_start:
atmNetwork_hostAtmServer?eval(cardNum),verifyPIN,0,PIN
valid->
if
::atomic{hostAtmServer_atmNetwork!cardNum,verifyPIN,0,P
INvalid->
        PINOK=true;
        goto Cash_Withdrawal;}
::atomic{hostAtmServer_atmNetwork!cardNum,verifyPIN,0,P
ININvalid->
        PINOK=false;
        goto server_start;}
fi;
Cash_Withdrawal:
        atmNetwork_hostAtmServer?cardNum,requestWithd
rawal,withdrawAmount,PIN->
if
::atomic{(withdrawAmount<=accountBalance)->
      if
      ::atomic{hostAtmServer_atmNetwork!cardNum,request
      Withdrawal,withdrawAmount,transactionOk->
          accountBalance=accountBalance-withdrawAmount;
          transactionOK=true;
          goto server_start;}
     ::atomic{hostAtmServer_atmNetwork!cardNum,request
     Withdrawal,withdrawAmount,transactionError->
          transactionOK=false;
          goto server_start;}
      fi;
}
::atomic{(withdrawAmount>accountBalance)->
        hostAtmServer_atmNetwork!cardNum,requestWithd
rawal,withdrawAmount,transactionUnsuccessful->
          transactionOK=false;
          goto server_start;}
fi;
```

V. RESULTS

For the model specifications given in PROMELA, SPIN helps the users to identify the deadlocks or unreachable code

in the model. In addition, SPIN can verify multiple claims on the execution of model by verifying the LTL properties inserted in SPIN.

TABLE I. LTL FORMULAS OF 1-LINK ATM MODEL

	LTL formulas
1	[](PINOK && transactionOK -> <>cashDispensed)
2	[](ejectCard -> <>printReciept)
3	[](cashDispensed && !continueTransaction) -> <>(printReciept && ejectCard))
4	[]!(cardLock && !ejectCard)

In this work a 1-link ATM system is modeled in PROMELA and its various properties are analyzed in the above sections. In this section, several properties of the 1- link ATM system presented above are verified using SPIN. For example, the size of the model and the time for the verification of the model is measured.

First of all, we have run SPIN for three cardholders to check for the errors, elapsed time and memory usage. Fig. 4 represents the results of verification in SPIN, the first line of the results represent the version of the SPIN verifier used in the verification of the model. In the results, the "+" sign in the second line indicates that the default algorithm is adopted. In Line 3 the search type is represented. The "-" sign in the next line represents that it is not using LTL formulas. In Line 5 it is shown that the process doesn't violate any of user defined conditions. Line 6 represents that acceptable cycles are also detected by the process. Later the next line represents invalid end states which indicate the absence of any deadlock. All the later results represent the information about the model about the states, memory usage, etc.

```
(Spin Version 6.1.0 -- 4 May 2011)
        + Partial Order Reduction

Full statespace search for:
        never claim             - (not selected)
        assertion violations    +
        acceptance  cycles      + (fairness disabled)
        invalid end states      +

State-vector 444 byte, depth reached 21, errors: 0
    342 states, stored
    571 states, matched
    913 transitions (= stored+matched)
      0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
   0.150        equivalent memory usage for states (stored*(State-vector + overhead))
   0.363        actual memory usage for states (unsuccessful compression: 242.23%)
        state-vector as stored = 1098 byte + 16 byte overhead
   2.000        memory used for hash table (-w19)
   3.433        memory used for DFS stack (-m100000)
   5.726        total actual memory usage

pan: elapsed time 0.002 seconds
```

Fig. 4. Verification results of 3 card holders.

In addition to this SPIN also helps the user to verify the model using LTL properties. LTL allows the user to express the behavior of the system using temporal properties that system must conform. Table 1 represents LTL formulas applied on the system, the first and most important property about the ATMs is that only when the cardholder enters the correct PIN and he has enough balance in his account, i.e. account balance should be greater than the amount to be withdrawn, then cash could be dispensed from the ATM. This property is expressed as LTL formula as: [] (PINOK && transactionOK -> <>cashDispensed).

The next LTL formula to be verified confirms that the ATM prints the receipt whenever the ATM ejects the card after the cash is dispensed. The property can be stated as: [] (ejectCard-> <>printReciept).

The next property which needs to be verified is that only when the cash is dispensed by the ATM and the user doesn't wish to continue transaction then the ATM will eject the ATM card and print the receipt. This property of the ATM can be verified by the LTL formula stated as: [] ((cashDispensed && !continueTransaction) -> <>(printReciept && ejectCard))

```
(Spin Version 6.1.0 -- 4 May 2011)
        + Partial Order Reduction

Full statespace search for:
        never claim             + (l0)
        assertion violations    + (if within scope of claim)
        non-progress cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 440 byte, depth reached 57, errors: 0
    181 states, stored
    267 states, matched
    448 transitions (= stored+matched)
      0 atomic steps
hash conflicts:         0 (resolved)

2.539   total actual memory usage

unreached in proctype Client
        (12 of 52 states)
unreached in proctype Cash_Dispenser
        (18 of 53 states)
unreached in proctype ATM_Network
        (13 of 49 states)
unreached in proctype Host_ATM_Server
        (5 of 33 states)
unreached in init
        (0 of 23 states)
unreached in claim l0
        _spin_nvr.tmp:10, state 11, "-end-"
        (1 of 11 states)

pan: elapsed time 0.001 seconds
```

Fig. 5. Result of Verification of LTL formula 3.

Above stated LTL properties should be verified by our system, i.e. 1-link ATM system. Now we will present certain results after applying those LTL formulas on the PROMELA model. After performing multiple experiments we came to the conclusion that the LTL formulas verify our PROMELA model. Fig. 5 presents the result, when SPIN performs a full state space search using the LTL formula [] ((cashDispensed && !continueTransaction) -> <>(printReciept && ejectCard)) on the PROMELA model. The first few lines of the result are already explained before but as compared to the last results, there are some differences like unreached in proctype. The term represents that there are few unreachable states in this

case because to verify the case we have defined different values to the variables like cashDispensed, continueTransaction, etc. As for the above case, the ATM should have dispensed the cash and the user didn't ask to continue transaction so in that case the ATM will now eject the card and print the receipt. This means the conditions like the user wishes to continue the transaction, or the user enters some invalid PIN, etc. will never arise. So in this way, the result that there are some states that could never be reached in this case is acceptable.

VI. Conclusion

In this paper model checking approach is introduced to verify 1-link ATM Systems. Firstly we consider 1-link ATM as an extended finite machine that is further presented in PROMELA. Further different properties of the system are expressed using LTL formulas and then the properties are verified using SPIN model checker. Finally, it proves that the model checking technology and SPIN model checker are both appropriate for verifying the business flows of 1-link ATM systems.

For our future research, we will try to modify the model by verifying more security related properties that include cash deposit, bill payment, and cash transfer using SPIN. Moreover the research can be expanded in other related domains of banking, as mobile banking, also.

Refrences

[1] Al Obisat, Farhan M., and Hazim S. AlRawashdeh. "Formal Verification of a Secure Model for Building E-Learning Systems." *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*7.6 (2016): 377-380.

[2] Osama Dandash, Phu Dung Le, and Bala Srinivasan. Security analysis for internet banking models. In Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on, volume 3, pages 1141–1146. IEEE, 2007.

[3] Alexander De Luca, Marc Langheinrich, and Heinrich Hussmann. Towards understanding atm security: a field study of real world atm use. In Proceedings of the sixth symposium on usable privacy and security, page 16. ACM, 2010.

[4] Syed MS Islam, Mohammed H Sqalli, and Sohel Khan. Modeling and formal verification of dhcp using spin. IJCSA, 3(2):145–159, 2006.

[5] Marina Mongiello. Finite-state verification of the ebxml protocol. Electronic Commerce Research and Applications, 5(2):147–169, 2006.

[6] Vladimir A Oleshchuk. Modeling, specification and verification of ad-hoc sensor networks using spin. Computer Standards & Interfaces, 28(2):159–165, 2005.

[7] Raman Kazhamiakin, Marco Pistore, and Marco Roveri. Formal verification of requirements using spin: A case study on web services. In Software Engineering and Formal Methods, 2004. SEFM 2004. Proceedings of the Second International Conference on, pages 406–415. IEEE, 2004.

[8] Gerard J Holzmann. The SPIN model checker: Primer and reference manual, volume 1003. Addison-Wesley Reading, 2004.

[9] Huiling Shi, Wenke Ma, Meihong Yang, and Xinchang Zhang. A case study of model checking retail banking system with spin. Journal of computers, 7(10):2503–2510, 2012.

[10] Wei Zhang. Model checking and verification of the internet payment system with spin. Journal of Software, pages 235–257, 2012.

[11] Tarek MI El-Sakka and M Zaki. Using predicate-based model checker for verifying e-commerce protocols. IJ Network Security, 16(2), 2014.

[12] Both, Andreas, Wolf Zimmermann, and René Franke. "Model checking of component protocol conformance–optimizations by reducing false negatives." *Electronic Notes in Theoretical Computer Science* 263 (2010): 67-94.

[13] Bernardo M. David Flavio G. Deus Rafael Timoteo de Sousa Jr. Laerte Peotta, Marcelo D. Holtz. A formal classification of internet banking attacks and vulnerabilities. International Journal of Computer Science & Information Technology (IJCSIT), 3, Feb 2011.

[14] Haiping Xu and Yi-Tsung Cheng. Model checking bidding behaviors in internet concurrent auctions. International Journal of Computer Systems Science & Engineering, 22(4):179–191, 2007.

[15] Tuan, Luu Anh, Man Chun Zheng, and Quan Thanh Tho. "Modeling and verification of safety critical systems: A case study on pacemaker." *Secure Software Integration and Reliability Improvement (SSIRI), 2010 Fourth International Conference on*. IEEE, 2010.

[16] Cimatti, Alessandro, et al. "Nusmv 2: An opensource tool for symbolic model checking." *International Conference on Computer Aided Verification*. Springer, Berlin, Heidelberg, 2002.

[17] Lockhart, Jonathan, Carla Purdy, and Philip Wilsey. "Formal methods for safety critical system specification." *Circuits and Systems (MWSCAS), 2014 IEEE 57th International Midwest Symposium on*. IEEE, 2014.

[18] Bozzano, Marco, and Adolfo Villafiorita. "Improving system reliability via model checking: The FSAP/NuSMV-SA safety analysis platform." *SAFECOMP*. Vol. 2788. 2003.