Formal Verification of an Authentication Protocol in M-Commerce

Sara Motiee
University of British Columbia
motiee@ece.ubc.ca

Abstract—**M-commerce is the buying and selling of goods and services using wireless handheld devices such as cellular telephone and personal digital assistants. One of the main concerns in m-commerce is the lack of verified authentication and key establishment protocols that are secure against fraud, counterfeit, and theft in mobile electronic transactions. In this report, the current m-commerce authentication protocols are studied and one of them named NAETEA is verified formally using Murphi, which is a formal verification tool. The result of formal verification of this protocol shows that it is secure against the attacks in which attacker can replay the previously transferred messages or generate messages using some components of previously transferred messages.**

I. Introduction

IN the recent years e-commerce technology has developed rapidly. On the other hand, there is a growing demand for using mobile devices. The combination of these two phenomena has lead to emergence of m-commerce. M-commerce (mobile commerce) is the buying and selling of goods, services and information using wireless handheld devices such as cellular telephones or personal digital assistants (PDAs) which uses a wireless connection to establish a communication between all necessary parties in a financial transaction. The transaction can use the Internet as the medium; however, any other network can be used. M-commerce is the new mode of e-commerce, which is getting popular increasingly because of widespread use of wireless and mobile communication devices. Although m-commerce provides many new commercial opportunities, it presents many technical challenges. One of these challenges is securing the whole infrastructure, which supports m-commerce transactions. One of the most crucial security concerns in m-commerce is the mutual authentication and key establishment between wireless device of client and the wired service provider. However, the specific features of m-commerce infrastructure causes that the security mechanisms and protocols which are applied in wired networks can not be applied in m-commerce. The first feature is that m-commerce infrastructure consists of a wired network backbone such as Internet and a wireless access network. While the wired part of infrastructure has large amount of computational, storage and bandwidth resources, the wireless portion and mobile devices are limited in terms of such resources. Secondly, users of m-commerce applications can perform transactions while they are in move. However, in the wired infrastructure, all the involved parties in the transaction are fixed. The third distinct feature of m-commerce infrastructure is the openness of air interface, which is more vulnerable to snooping. The above-mentioned features decrease the security of m-commerce environment so that achieving secure protocols in m-commerce is more challenging.

There is an evident need to prove that the authentication protocol of an m-commerce application is secure against different types of attacks so that users can rely on the system to trade business or do online shopping. One approach to prove the security of an authentication protocol is the use of formal verification methods. Formal methods verify the correctness of systems that are too complicated and whose correct operation is of so high importance. To verify a system formally, first the system is specified in a formal specification language that has some mathematical basis. Then theorems are proved about the specification with the assistance of an automatic theorem-prover. The reason for using formal verification tools to analysis the security of an authentication protocol is that they can explore the whole state space that a protocol covers. Although the security protocol specifications are usually very small (not more than four or five message exchanges typically), they operate under complex environments. Therefore, to analysis a protocol for correctness, it is necessary to consider many roles, and their interactions with each other and the intruder behavior. Formal verification tools can assure that they examine all the possible interactions between all the involved parties in the protocol.

In this report one of the m-commerce authentication and key establishment protocols named NAETEA is verified formally using Murphi. Murphi is a general-purpose state enumeration tool, which has been used to analysis Needham Schroeder protocol, Kerberos and TMN protocol. It has been shown in [1] that Murphi is efficient for examining relatively short protocols and can detect replay attacks or errors resulting from confusion between independent executions of a protocol by independent parties.

The remainder of this report is structured as follows. Section 2 is related works in which the current m-commerce authentication protocols and the current verification tools are introduced. It also introduces the m-commerce protocols, which have been verified formally. Section 3 introduces Murphi and section 4 explains NAETEA authentication protocol, which has been analyzed in this report. Section 5 shows how this protocol is modeled and verified using Murphi. Section 6 presents verification results and section 7 is conclusion.

II. Related Works

In this section, the authentication protocols, which have been used in m-commerce, are introduced briefly. Then the existing formal verification tools and approaches are reviewed. Also the m-commerce authentication protocols, which have been analyzed using formal methods, are introduced.

A. M-Commerce Authentication Protocols

The current authentication protocols between wireless devices and wired service providers can be categorized to two groups. The first group applies a trusted third party to do the authentication between two parties and the second group does not use any third party. Each group approaches have some advantages and disadvantages. The first group approaches minimize the overhead on mobile devices but the third party is able to access the session key and communication traffic. Also the third party will be a security bottleneck. The second group approaches may use public-key infrastructures, which are not cost-effective for mobile devices.

Some examples of the first category include the protocol, which is introduced in [2] and applies entity authentication method based on self-updating hash chains scheme. The other protocol [3] is based on combining the KryptoKnight protocol [4] and the X.509 protocol [5]. It also decreases the security risks implied by trusted third party because the third party does not keep the consumers' important payment information, such as credit card or debit card information. Also Zhang [6] has proposed an asymmetric authentication protocol named NAETEA (Network Assisted End-To-End Authentication) in which the wireless access home network of a mobile station assists in authentication of mobile station and service provider.

The second category of m-commerce authentication protocols does not use any third party. One example is Simoes et al' s protocol, which is based on symmetric cryptography but requires a previous agreed internal key between service provider and mobile device [7]. AuthenLink [8] is another authentication protocol, which operates using a microprocessor chip (ChipTag) implanted under human skin. The ChipTag is able to authenticate user's access to systems and connects them wirelessly through the Radio Frequency Identification technology. One more example is ASPeCT [9], which has tried to solve the performance problem. However, it is assumed that the service provider has a reliable identifier which mobile device knows it before starting the transaction.

Among the current m-commerce authentication protocols, two of them have been verified formally. The first one is Chen et al.'s protocols which is based on hash chain [10]. Three entities exist in this protocol: mobile user, network information service provider and trusted third party. The authors have proposed an extended form of BAN Logic [11] for analysis of their protocol. They have shown using BAN Logic that the goals of authentication of these entities are achieved. However, no intruder is modeled in their analysis.

The other formally verified protocol is Song et. al's protocol [3] in which a third party takes care of authentication between buyer and seller. Authentication between the sellers and third party is achieved using symmetric key cryptography and authentication between seller and third party is done via PKI system. This protocol is verified by using CSP/FDR [12] and guarantees that the three parties can authenticate to each other. The authors have also modeled the intruder who can overhear all network messages; prevent messages from their intended recipients; and transmit fake messages to any other party.

The protocol, which is studied in this report, is named NAETEA [6], which uses home network of mobile station in authentication process and will be explained in section 4.

B. Formal Verification Methods

Authentication protocols are well suited for being analyzed using formal verification methods. They are usually well defined so that they can be modeled accurately and also they are complex so that their analysis using manual approaches is prone to errors. Different formal verification tools have been applied on security protocols [12]. Each of these tools has some limitations, which make the verifying of protocols a challenging job. Some of the verification tools are general-purpose model checkers such as FDR [13] and Murphi [1]; and some are special-purpose model checkers such as Interrogator [14], Brutus [15] and NRL Protocol Analyzer [16]. FDR is a model checker for CSP (Communicating Sequential Processes). Every role of a protocol is translated to a CSP process; also there is an intruder process in the model. Then a concurrent composition of finitely many instantiations of the role processes and the intruder process is considered, and checked against different properties. Murphi is an automatic verification tool, which searches for insecure points within the state space using a model checker. The general-purpose model checkers suffer from the state space explosion problem. So they can be used for verifying the protocols which have a small number of participants, e.g. three or five, and send and receive a small number of messages.

Interrogator and Brutus start with an initial state of a protocol execution and search through all possible sequences of actions to see whether an attack could happen. NRL Protocol Analyzer starts from an insecure state and performs a backward search trying to prove that this insecure state is unreachable. It can prove a protocol to be correct for arbitrary number of participants. However, it requires high level of expertise, and its running time is considerable.

A different approach, based on formal verification, is to use model logics. The best example of this category is BAN logic [11]. For using BAN Logic, an initial set of beliefs is adopted, and then another set of beliefs is adopted when a message is received in a protocol. If the resulting set of beliefs is acceptable, then the protocol is proven to be correct. The logic cannot be used to prove secrecy, only authenticity, because the logic does not attempt to model knowledge.

In this report, Murphi has been chosen for analyzing the m-commerce authentication protocol. Murphi will be introduced in the next section.

III. Murphi

Murphi is a protocol verification tool that has been used to verify some protocols in the area of multiprocessor cache coherence protocols and multiprocessor memory models [1][17][18]. It is an explicit state protocol verifier that consists of a Murphi Compiler and a Murphi Verifier. To verify a protocol, first it should be modeled in Murphi language and some desired properties about the protocol should be defined. Then the Murphi compiler is applied on the modeled protocol and generates a special-purpose verifier for this particular Murphi description. The generated verifier is a collection of C++ include files and contains the core state enumeration algorithms. This verifier automatically checks by explicit state enumeration if all reachable states of the model satisfy the defined properties. For the state enumeration either breadth first or depth first search can be selected.

To model a protocol in Murphi language, some global variables, data types, transition rules and a set of invariants are defined. The state of the model is the value of all global variables and transition rules specify how one state is evolved to the next state. Each rule has a condition and action. If the condition is satisfied, the action will be executed.

The correctness of protocol can be checked in three ways. First some invariants can be defined which are Boolean conditions that have to be true in every reachable state. The second approach is to using explicit "assert" and "error" statement that can be called within an action. If one of these conditions occurs, the verifier halts and prints a sequence of states that leads from the initial state to the error state. The third approach is to check protocol for deadlock state in which no other state than the current state can be reached.

Murphi has been used to analysis three security protocols so far and has succeeded to find some security vulnerabilities in them [1]. These protocols include Needham Schroeder protocol, TMN and Kerberos protocol.

Similar to other general-purpose model checkers, Murphi may encounter state explosion problem and also modeling the intruder has difficulties. However, Murphi implements a richer set of methods such as symmetry reduction, hash compaction, reversible rules and repetition constructors which increase the size of the protocols which can be verified and reduce the memory and runtime requirements during the state enumeration. Also it has implemented improvements for analyzing security protocols so that the modeling of intruder will be easier [19].

In this report, Murphi 3.1 is used which include the above features [18].

IV. NAETEA Protocol

In [6], a novel protocol for mutual authentication and key establishment between a wireless device and wired service provider is proposed. This protocol is based on asymmetric cryptography but the heavy cryptographic operations are shifted toward service provider and home network of wireless device. Also the home network does not have access to session key or any plain-text messages transferred between two end entities and the home network is accountable for every operation it performs.

The end-to-end authentication between a MS (mobile station) and a SP (service provider) can be fulfilled using three authentication processes: MS-HN authentication (HN is the home network of mobile station), HN-SP authentication and MS-SP authentication. It is assumed MS and HN are already authenticated using home networks standards and they share a secret session key $K_{MS\text{-}HN}$ and a secret temporary identity TMUI which is used as the identity of the MS during end to end authentication. Also since HN and SP are connected to wired network, their authentication can be performed using different protocols and it is assumed they have been authenticated to each other and share the session key $K_{HN\text{-}SP}$. So the authentication between MS and SP can be done through the following steps:

1. An MS initiates the authentication process by sending to HN its identity idMS, the SP's identity idSP and a random number encrypted with the public key of the SP (EpuSP(x)). All these fields are encrypted using the session key $K_{MS\text{-}HN}$. Also the integrity of message is provided using a keyed hash value of the whole message.
2. The HN forwards EpuSP(x) to the SP together with a hash value h(TMUI) and its signature sigHN (= EpvHN(h(h(TMUI), EpuSP(x))), where pvHN is the private key of the HN. The message is encrypted using session key $K_{HN\text{-}SP}$.
3. The SP replies to the MS's request with a random number y and its signature sigSP (= EpvSP(h(y)), where pvSP is the private key of the SP). It then computes the secret session key $K_{MS\text{-}SP}$ (= h(x, y)).
4. The HN forwards y to the MS with a hashed value h($K_{HN\text{-}SP}$). h($K_{HN\text{-}SP}$) will be used to authenticate the SP to the MS.
5. The MS computes the session key $K_{MS\text{-}SP}$ (= h(x, y)) and sends a verifiable authenticator h(h(TMUI), $K_{MS\text{-}SP}$) to the SP.
6. The SP computes hash value h(h(TMUI), $K_{MS\text{-}SP}$), where the value h(TMUI) is received in message 2 and $K_{MS\text{-}SP}$ is computed by the SP. If this value equals the value received from MS, the MS is authenticated. Then the SP computes h(h($K_{HN\text{-}SP}$), $K_{MS\text{-}SP}$) and sends it to the MS.
7. MS uses h($K_{HN\text{-}SP}$) received in transaction 4 to compute value h(h($K_{HN\text{-}SP}$), $K_{MS\text{-}SP}$), and compares it with the one received. If they are equal, the SP is authenticated and the authentication process is successfully completed.

V. Apply Murphi on NAETEA Protocol

To model the NAETEA protocol using Murphi, the below steps have been followed:

1- Model the protocol in Murphi: In this step the required data structures and rules for modeling the protocol is defined in Murphi language. For this purpose, MS, HN and SP are modeled by separate data types, which include their state and identity of the other two parties which they communicate with. Each of these entities can be in 3 states:

a) Sleep: The entity has not initiated any message.

b) Wait: The entity has generated a message and is waiting for a response.

c) Commit: The entity has received the desired response and is not going to reply back.

Also another data type is defined for a message and seven message types are specified. Other data types are defined for pair session keys between each two parties, HN and SP signatures, encrypted random numbers and the keyed hashed values that are used for integrity checking. The network is modeled by an array of messages and the total number of allowed messages is configurable.

For modeling the message transfer, seven different rules are defined, which show when each party is activated and how it responds to a received message.

Another part of the model is called “StartState” in which the MS, HN, SP and intruder are initialized by setting their states and their other data fields.

As an example, the data structures that have been defined for modeling the MS are as follows:

```
const:
      NumMS:   1;   -- number of MS
type:
      MSIdentity:   scalarset (NumMS);
      States : enum {
            SLEEP,
            WAIT,
            COMMIT
       };
      MS : record
        state:     States;
        homenet:  HNIdentity;
        servicepro: SPIdentity;
      end;
var mob: array[MSIdentity] of MS;
```

NumMS defines the number of MS in the network. MSIdentity represents the Id of MS and is a sub range of NumMS. Each MS is modeled by a record named MS, which consists of its state, the identifier of home network and identifier of service provider which mobile station initiates the communication. Finally, ‘mob’ is an array of MS record, which keeps the data of each mobile station. In the ‘StartState’ section of the model, the state of all MS is initialized to Sleep and the identifier of their home network and service provider are determined.

The behavior of mobile station is modeled with 3 rules. In the first rule, MS initiates the authentication by sending a message to HN and changes its own local state from SLEEP to WAIT. The second rule models the reception and checking of the reply from HN and sending the message to SP. The third rule checks the response from SP and in case the message is correct, MS’s state changes to COMMIT. Similar rules are defined for HN and SP. As an example, the following rule, shows how MS initiates the protocol:

```
ruleset i: MSIdentity do
 ruleset j: AgentId do
  rule 20 "MS starts protocol (step 1)"
    mob[i].state = I_SLEEP &
    !ismember(j,MSIdentity) &    -- only HN, SP and intruders
    multisetcount (l:net, true) < NetworkSize     ==>
  var
       outM: Message;   -- outgoing message
       MS_HN_key: PairKeyMS_HN;
       enX: EncryptedNounce;
       intChek: IntegrityCheckerM1;
       homenetworkid: HNIdentity;
       serviceproid : SPIdentity;
  begin
       undefine outM;
       outM.source  := i;
       outM.dest    := j;
       undefine MS_HN_key;
       MS_HN_key.party1 := i;
       MS_HN_key.party2 := homenetworkid;
       outM.keyM1 := MS_HN_key;
       outM.mType   := M_1;
       outM.mobileId := mobileid;
       outM.serviceId := serviceproid;
       undefine enX;
       enX.key := serviceproid;
       enX.nounce:= i;
       undefine intChek;
       intChek.id1 := i;
       intChek.id2 := serviceproid;
       intChek.encryptedX := enX;
       intChek.key := MS_HN_key;
       outM.encryptedX  :=   enX;
       outM.checker1 := intChek;
       multisetadd (outM,net);
       mob[i].state     := I_WAIT;
       mob[i].homenet := homenetwork;
       mob[i].servicepro := serviceproid;
  end;
 end;
end;
```

Since the rule is defined using ruleset, it will be instantiated for each MS. So when the value of NumMS is changed, the number of rules will be changed as well. The condition of the rule states that if MS is in SLEEP state and the destination of message is an entity other than MS and also the number of current messages of the network is less than the maximum allowed messages, this rule can be activated. When the rule is activated, the first message of the protocol will be composed and added to network. Also the state of MS is updated to WAIT.

2- Add an intruder to the model: It is assumed that intruder is a participant in the protocol that can initiate communication with other participants of the system. Also intruder can perform the following actions:

a) Overhear every message: Remember all parts of each message.

b) Replay the intercepted messages.

c) Generate messages by using the components of intercepted messages.

The intruder is modeled by an array of messages and components of messages he knows. These components include random numbers and hashed values, which are transferred in protocol messages. Also three rules are defined for implementing the above three actions of intruder. As an example, the following rule shows how intruder intercepts a message:

```
ruleset i: IntruderId do
 choose j: net do
   rule 10 "intruder intercepts messages"
    !ismember (net[j].source, IntruderId)   ==>
      var temp: Message;
      begin
        alias msg: net[j] do   -- message to intercept
        temp := msg;
        undefine temp.source;   -- delete useless information
        undefine temp.dest;
        multisetadd (temp, int[i].messages);
        end;
      end;
     multisetremove (j,net);
    end;
 end;
end;
```

3- Defining the desired properties of the protocol: For verifying the correctness of protocol, two invariants are defined. These two invariants state that MS and SP should be authenticated correctly. One of the invariants is as follows:

```
invariant "SP correctly authenticated"
     forall i: MSIdentity do
        mob[i].state = COMMIT &
        ismember(mob[i].servicepro, SPIdentity)
        ->
        serpro[mob[i].servicepro].mobile = i &
        serpro[mob[i].servicepro].state = COMMIT
      end;
```

This invariant basically states that for each MS_i, if it is committed to a session with a SP, this SP (whose identifier is stored in the SP field of MS_i) must have started the protocol with MS_i (i is stored in the MS field of this SP) and must be in Commit state.

The other invariant is defined similarly to verify that MS is authenticated correctly.

VI. Verification Results

After modeling the protocol in Murphi language, the Murphi execution environment is setup. Then the modeled protocol is translated to a C++ file using Murphi compiler. The C++ file will also be compiled using a C++ compiler and an automatic verifier will be generated which is an executable file and verifies the correctness of protocol.

By changing the value of constant parameters of the model, different results can be obtained. Table 1 shows the result of verification for different number of MS, SP, HP, intruder and network size (The maximum number of messages which is allowed in the network.). The results which include the number of explored states and the time of verification are obtained by running the Murphi on a PC station with a 2.33 GHz Intel Dual Core CPU and 2 GB RAM.

TABLE I
Number of Reachable States in Different Execution Condition

Number of				Network Size	States	Time
MS	HN	SP	Intruder			
1	1	1	1	1	18000	1.25
1	1	1	1	2	87021	17.45
2	1	1	1	1	54304	8.32
1	2	1	1	1	57902	9.01
1	1	2	1	1	60982	9.32
2	2	1	1	1	716001	405.09
2	2	2	1	1	State Explosion	*

By increasing the number of involved parties in the protocol, Murphi encounters state explosion problem, which is shown in Table 1 as well.

By running the verifier in the above conditions, no error is found the protocol. So the result of verification proves that NAETEA protocol is able to authenticate the mobile station and service provider correctly and is not vulnerable to attacks in which intruder is able to replay the previously transferred messages or generate messages using some components of these messages.

VII. Conclusion

The success of m-commerce application is highly dependent on the secure and correct authentication of mobile user and service provider. However, most of the current m-commerce authentication protocols do not guarantee the correct authentication. One way to prove the correctness and security of a protocol is formal verification. In this report, after reviewing the current m-commerce authentication protocols and current formal verification tools, one of the m-commerce authentication protocols named NAETEA is verified using Murphi, which is an explicit state protocol verifier. In comparison to other model checker, Murphi has richer set of operations and repetition constructors, which enhance the size of verifiable protocols. Also because of use of symmetry reduction and hash compaction techniques, the verification time has decreased. For modeling the NAETEA, 7 transition rules, 3 states, two invariants and other required data structures for modeling the mobile station, service provider and mobile home network are defined. Also the behavior of intruder is represented by 3 rules, which can intercept the message, replay old messages or generate new messages using the components of old messages. The verification results for different configuration of this model proves that NAETEA can achieve the authentication of mobile user and service provider correctly and is secure against such an intruder behavior.

References

[1] J. C. Mitchell, M. Mitchell, and U. Stern, "Automated Analysis of Cryptographic Protocols Using Murphi", *In Proc. of IEEE Symposium on Security and Privacy*, pp. 141-151, 1997.

[2] L. Chen, H. Zhang, and N. Liu, "Authentication and Micropayment Protocols based on Self-Updating Hash Chains", *In Proc of Sixth International Conference on Grid and Cooperative Computing (GCC 2007)*, pp. 467-472, 2007.

[3] M. Song, X. Hu, J. Li, and G. Deng, "An Authentication Model Involving Trusted Third Party for M-Commerce", *In Proc. of the Sixth International Conference on the Management of Mobile Business*, pp. 53-59, 2007.

[4] P. Janson, and G. Tsudik, "Secure and Minimal Protocols for Authenticated Key Distribution", *Computer Communications Journal*, Vol. 18, pp. 645-653, 1995.

[5] "The directory authentication framework", CCITT Recommendation X.509. CCITT (1998).

[6] L. He , and N. Zhang, "An Asymmetric Authentication Protocol for M-Commerce Applications", *In Proc of the Eighth IEEE International Symposium on Computers and Communications*, pp. 244-250, 2003.

[7] P. Simões, P. Alves, J. Rogado, and P. Ferreira, "An Authentication Protocol for Mobile Devices", *International Workshop on Internet 2000 (integrated in the Internatioanl Conference on Distributed Computing Systems-ICDCS'2000)*, Taiwan, April 10-13, 2000.

[8] C. Braz, and E. Aïmeur, "AuthenLink: A User-Centred Authentication System for a Secure Mobile Commerce", *In the Proc. of the 3rd International Workshop on Wireless Information System*, pp. 1-11, 2004.

[9] G. Horn, and B. Preneel, "Authentication and payment in future mobile systems", *In the Proceedings of European Symposium on Research in Computer Security (ESORICS '98), Springer-Verlag*, pp. 277-293, 1998.

[10] L. Chen, G. Zhang, and X. Li, "Efficient Identity Authentication Protocol and Its Formal Analysis", Computational Intelligence and Security Workshops, pp. 712-716, 2007.

[11] M. Burrows, M. Abadi, and R. Needham, "A Logic of Authentication", *ACM Transactions on Computer Systems*, Vol. 8, No. 1, pp. 16-36, 1990.

[12] C. Meadows, "Formal verification of cryptographic protocols: A survey", *In Advances in Cryptography – Asiacrypt 94*, pp. 135-150, 1995.

[13] A. W. Roscoe, "Modeling and verifying key-exchange protocols using CSP and FDR", *In Proc. of 8th IEEE Computer Security Foundations Workshop*, pp. 98-107, 1995.

[14] J. Millen, "The Interrogator model", *In Proc. of the 1995 IEEE Symposium on Security and Privacy*, pp. 251–260, 1995.

[15] E.M. Clarke, S. Jha, and W. Marrero. "Using state space exploration and a natural deduction style message derivation engine to verify security protocols", *In Proceedings of the IFIP Working Conference on Programming Concepts and Methods (PROCOMET)*, pp. 87-106, 1998.

[16] C. Meadows, "A model of computation for the NRL protocol analyzer", *In Proc. of the 1994 Computer Security Foundations Workshop, IEEE Computer Society Press*, pp. 84-89, June 1994.

[17] D. L. Dill, A. J. Drexler, A. J. Hu, and C. H. Yang. "Protocol verification as a hardware design aid. In 1992 IEEE International Conference on Computer Design, pp. 522–525, 1992.

[18] http://www.cs.utah.edu/formal_verification/software/murphi

[19] V. Shmatikov and U. Stern, "Efficient Finite-State Analysis for Large Security Protocols", *11th IEEE Computer Security Foundations Workshop*, pp. 106-15, 1998.