Modeling and Verification of Fiat-Shamir Zero Knowledge Authentication Protocol

Amit K. Maurya, Murari S. Choudhary, P. Ajeyaraj, and Sanjay Singh

Department of Information and Communication Technology
Manipal Institute of Technology, Manipal University, Manipal-576104, India
sanjay.singh@manipal.edu

Abstract. Model checking is a multi-purpose, automatic technique for verifying finite-state concurrent systems. Formal verification methods have quite recently become usable by industry. Presently model checking has been widely used in hardware, software validation and security protocol analysis. Fiat-Shamir is one of the many zero-knowledge authentication protocol which is used for security authentication purpose. In this paper, we have proposed a formal model of Fiat-Shamir authentication protocol using Finite State Machine (FSM). Security requirements are represented using Computation Tree Logic (CTL). These security requirements are verified and analyzed using symbolic model checker tool NuSMV. Based on our verification we have identified one of the security flaw of Fiat-Shamir protocol using the NuSMV model checker.

Keywords: Model Checking, Zero Knowledge Authentication Protocol, Fiat-Shamir Protocol, CTL, Finite State Machine (FSM).

1 Introduction

An authentication protocol is a type of cryptographic protocol with the purpose of authenticating entities wishing to communicate securely. In password authentication, the claimant needs to send her secret to the verifier. It leads to a problem of eavesdropping. In addition, a dishonest verifier could reveal the password to others or use it to impersonate the claimant.

In challenge-response entity authentication, the claimant's secret is not sent to the verifier. The claimant applies a function on the challenge sent by the verifier that includes her secret. In some challenge response-methods, the verifier actually knows the claimants secret, which could be misused by the dishonest verifier. In other methods, the verifier can extract some information about the secret from the claimant by choosing a preplanned set of challenges. In zero-knowledge authentication [1][2], the claimant does not reveal anything that might cause danger to the confidentiality of the secret. The claimant proves to the verifier that she knows a secret, without revealing it. The interactions are so designed that they can not lead to revealing or guessing the secret. After exchanging messages, the verifier only knows that the claimant does or does not have the secret, nothing more. The result is a yes or no situation.

N. Meghanathan et al. (Eds.): CCSIT 2012, Part II, LNICST 85, pp. 61–70, 2012.

To the best of our knowledge there is no work on modeling and verification of Fiat-Shamir zero knowledge authentication protocols has been reported in the literature. In this work the Fiat-Shamir Protocol based on the zero-knowledge authentication has been considered for the modeling and verification purpose. Many authentication protocols have been proposed and found to have flaws after the authentication. In order to avoid such problems arising in the design of protocols, several methods have been proposed to analyze them. Among the various methodologies proposed, model checking has been proved to be very useful for this purpose. Model checking [3] [4] is a general purpose, automatic technique for verifying finite-state concurrent systems. This technique provides a way to model a system using state-transition system and all the requirements to be checked is expressed using temporal logic. Given a model and requirements a model checker simulates to verify that whether a requirement is satisfied or not.

The aim of this work is to present a methodology for analyzing cryptographic protocol and to identify the security flaw of Fiat-Shamir authentication protocol using a symbolic model verifier NuSMV.

The rest of this paper is structured as follows. Section 2 gives the overview of the Fiat-Shamir protocol. Section 3 describes model of Fiat-Shamir protocol. In section 4 key properties of the system are verified. Section 5 provide conclusion of the work.

2 Overview of the Fiat-Shamir Protocol

In the Fiat-Shamir protocol [2], a trusted third party chooses two large prime numbers p and q to calculate the value of $n = p \times q$. The value of n is announced to the public. The values of p and q are kept secret. Alice the claimant, chooses a secret number s between 1 and n-1. She calculates v= s mod n. She keeps s as her private key and registers v as her public key with the third party. Verification of Alice by Bob can be done in four steps shown below:

1. Alice, the claimant, chooses a random number r between 0 and n-1. She then calculates the value of $x = r^2 \bmod n$; x is called as witness.
2. Alice sends x to Bob as the witness.
3. Bob the verifier, sends the challenge c to Alice,c is either 0 or 1.
4. Alice calculates the response $y = rs^c$, where r is a random number selected by Alice in the first step. s is her private key and c is the challenge (0 or 1).
5. Alice sends the response to Bob to show that she knows value of her private key, s. She claims to be Alice.
6. Bob calculates $y^2 \bmod n$ and xv^c. If these two values are equal then Alice either knows the value of s (she is honest) or she has calculated the value of y in some other way (dishonest) because we can easily prove that y^2 is the same as xv^c in the modulo n arithmetic as given below

$$y^2 = (rs^c)^2 = r^2 s^{2c} = r^2 (s^2)^c = xv^c$$

These six steps constitute a round; the verification is repeated several times with the value of c equal to 0 or 1. The claimant must pass the test in each round to be verified. If she fails one single round, the process is aborted and she is not authenticated. This entire process is shown in Fig.1.

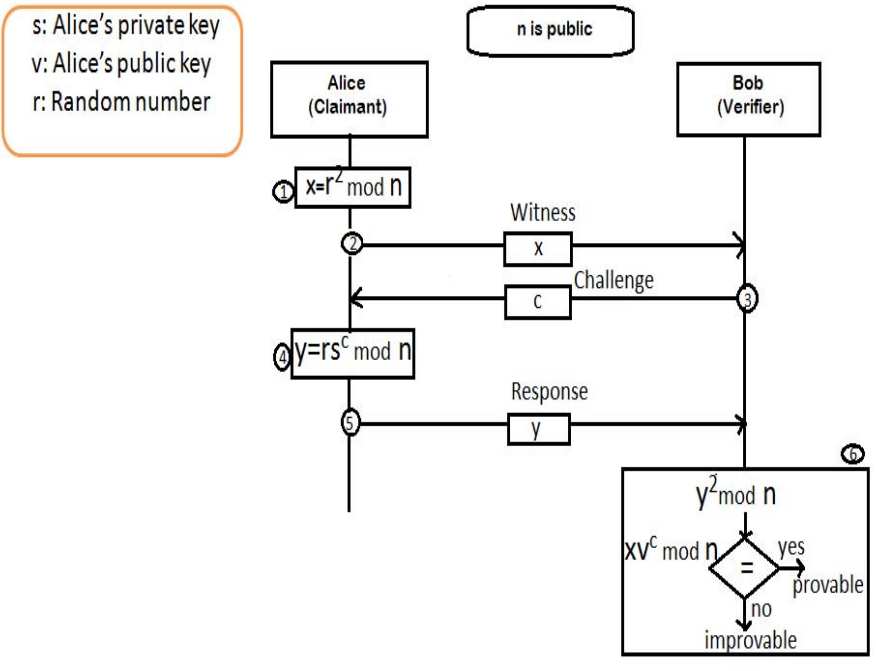

Fig. 1. Fiat-Shamir protocol

3 Model of the Protocol

Our methodology of analysis is based upon model checking, an automatic technique for verifying finite state concurrent systems. In the temporal logic model checking, the concurrent system is modeled as a state transition diagram, whereas properties are expressed in a temporal logic such as Linear Time Temporal Logic (LTL) or CTL [4]. The NuSMV model checker [5][6] following the above approach, adopts a structured input language to describe the model of system and the temporal logic CTL to express desired properties. The model of protocol has a modular structure. Each module is associated to an entity of the system and describes its behavior. The main modules used here are the *Alice*, *Bob* and *Third party* and they are considered to be honest principals. Model of the system is represented by a Kripke structure [4].

3.1 Kripke Structure

Kripke structure is a 4-tuple $M = (Q,I,\Delta,I)$ where:

- Q is a finite set of states.
- $I \subseteq Q$ is the set of initial states.
- $\Delta \subseteq Q \times Q$ is a transition relation which represents a state and its successor states.
- $L{:}Q \rightarrow 2^{AP}$ is a function which returns the set of atomic proposition that hold true in a state.

Kripke structure is used to represent the static topology and dynamic behavior of a system. Each state relates with a set of atomic propositions.

3.2 Specifying a Protocol as a Finite State Machine

Here the protocol is expressed as a state transition diagram. First an initial state is specified. Then an arc is drawn to another state for each message that can be sent or received at that point. Fig.2 describes the FSM model of Fiat-Shamir protocol.

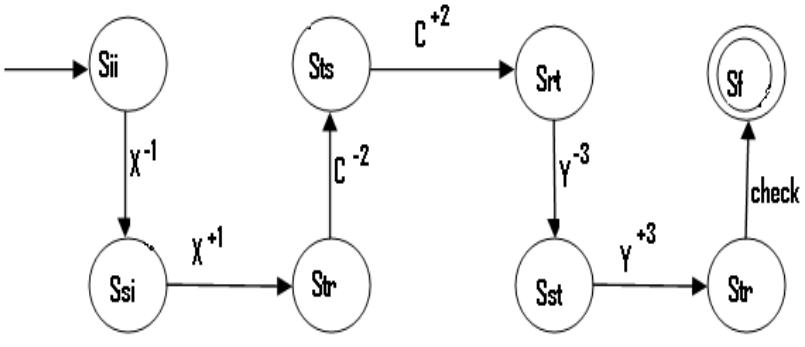

Fig. 2. FSM for Fiat-Shamir protocol

Table 1. Notations Used in the States of FSM

i	idle
s	sending
t	sent
r	received

Table.1 gives the different notations used in the states of FSM.

Table.2 gives the messages that are passing between different states of FSM.

Table.3 shows, what different states are representing and also what will be the state of Bob (verifier) and Alice (claimant) at different input values.

Table.4 gives different states of the FSM and their corresponding labels.

Table 2. Different Messages Exchanged

X^{-1}	First Message X is sent by Alice.
X^{+1}	First Message X is received by Bob.
C^{-1}	Second Message C is sent by Bob.
C^{+1}	Second message C is received by Alice.
Y^{-1}	Third message Y is sent by Alice.
Y^{+1}	Third message Y is received by Bob.
$check$	Event to enter final state.

Table 3. Description of Different States of FSM

S_{ii}	Alice and Bob in Idle State.
S_{si}	Alice is in Sending state and Bob is in Idle state.
S_{tr}	Alice is in Sent state and Bob is in Received state.
S_{ts}	Alice is in Sent state and Bob is in Sending state.
S_{rt}	Alice is in Received state and Sent state.
S_{st}	Alice is in Sending state and Bob is in Sent state.
S_{tr}	Alice is in Sent state and Bob is in Received state.
S_f	Alice and Bob will be in final state.

Table 4. States and their Corresponding Labels

State	**State Label**
S_{ii}	$s,\ r,\ n$
S_{si}	x, s, r, n
S_{tr}	x, s, r, n
S_{ts}	x, s, r, n
S_{rt}	x, s, r, n, c
S_{st}	x, s, r, n, c, y
S_{tr}	x, s, r, n, c, y
S_f	$value1, value2$

4 Verification and Analysis

4.1 Variables Used

Following are the list of variables maintained by the third party.

- p and q: These are the two prime numbers used for calculating private and public key.
- $nvalue.n$: Here $nvalue.n$ is equal to $p*q$ and is calculated by the third party and announced to the public.

Following are the list of variables maintained by the principal Alice.

- s: A secret key chosen between 1 and n - 1.
- v: Calculated as,
  $(alice.s)^2 \, mod \, (thirdparty.nvalue.n)$. This is the public key. Alice keeps s as her private key and registers v as her public key with the third party.
- r: Random number chosen between 0 and n - 1.
- $i.info.xx$: Temporary variable used by Alice and further assigned to x the witness. $i.info.xx$ is calculated as $(alice.r)^2 \, mod \, (thirdparty.nvalue.n)$
- $challenge : boolean$: The variable and its value is equal to the challenge that is sent by Bob, if he is the honest principal otherwise Alice can guess the value of challenge randomly (0 or 1).
- $i.info.yy\_c1$: Temporary variable calculated as $(alice.r) \times (alice.s) \bmod (thirdparty.nvalue.n)$ when the value of challenge is 1. This value will be assigned to y and sent to Bob for further calculation.
- $i.info.yyy\_c0$: Temporary variable calculated as $(alice.r) \, mod \, (thirdparty.nvalue.n)$ when the value of challenge is 0. This value will be assigned to y and sent to Bob for further calculation.

Following are the list of variables maintained by the principal Bob.

- $challenge : boolean$: variable called the challenge is chosen by Bob and the value is either 0 or 1. This challenge will be sent to Alice.
- $value1$: If the value of challenge is 1, the variable calculated as,
  $((alice.r \times alice.s) \, mod \, (thirdparty.nvalue.n))^2 \, mod \, (thirdparty.nvalue.n)$
  Otherwise if the value of challenge is 0, the variable is calculated as,
  $(alice.r \, mod \, (thirdparty.nvalue.n))^2 \, mod \, (thirdparty.nvalue.n)$.
  this value of $value1$ will be used for comparison with $value2$.
- $value2$: If the value of challenge is 1 then the variable will be calculated as,
  $(((alice.r)^2 mod \, (thirdparty.nvalue.n)) \times v.alice) mod \, (thirdparty.nvalue.n)$
  Otherwise, if the value of challenge is 0, the value is calculated as,
  $(alice.r)^2 mod \, (thirdparty.nvalue.n)$
  After calculating $value1$ and $value2$, these values are compared. If these two values are matching then Alice is authenticated.

4.2 Syntax and Semantic of CTL

Security requirements (properties) are expressed with a temporal logic that is Computation Tree Logic (CTL). The BNF of CTL syntax is as follows [4]:

$\Phi ::= \bot | \top | (\neg \Phi) | (\Phi \wedge \Phi) | (\Phi \vee \Phi) | (\Phi \rightarrow \Phi) |$ AXΦ| EXΦ| AFΦ| EFΦ | EGΦ| A[ΦUΦ]| E[ΦUΦ]

The first symbol of the pair is either "A" or "E" where "A" represents "all path" and "E" represents "exist a path". "A" and "E" are path operator.

The second symbol of the pair is possibly "X", "F", "G" and "U". "X" represents "next state", "F" represents "some future state", "G" represents "all cases in future", "U" represents "until".

A path operator must accompany with a temporal operator, and vice versa.

4.3 Verification Results

Given the model of the system to analyze, NuSMV the model checker simulates all its possible behaviors in order to verify whether requirement is satisfied. In this model authentication is the major requirement. As specified, in Fiat-Shamir protocol the claimant is authenticated if and only if the values of $y^2 \ mod \ n$ and $xv^c \ mod$ are matching. This is possible when the claimant Alice is honest and she must get the correct challenge from Bob. There is a possibility that a dishonest user can guess the correct value of challenge and can pass the test. This problem can be fixed by repeating this authentication process many times. Claimant is authenticated if and only if she passes all the tests assuming that it is not possible to guess the value of challenge c properly all the time. There are two possible cases under consideration:

- Now we will consider the first case where claimant is considered to be honest. If she is honest user then she must receive the correct value of challenge either 0 or 1. In NuSMV this can be checked using CTL specification, $AF(alice.challenge = bob.challenge- > AF(value1 = value2))$. In simple English it means that in all future states when *alice.challenge* is equal to *bob.challenge* then in all the future states $value1$ and $value2$ which are calculated by Bob should be same.
- Considering the second case, where claimant is considered to be a dishonest principal. In this case challenge sent by Bob is not equal to the challenge that is assumed by the Alice. Then the values of $value1$ and $value2$ are going to change because of different interpretation of the value of c by Alice and Bob. This can be verified by the CTL specification $AF(alice.challenge! = bob.challenge- > AF(value1! = value2))$. It means that in all future states when *alice.challenge* is not equal to *bob.challenge* then in all the future states $value1$ and $value2$ which are calculated by Bob should not be same.

Though it is said that in the case of dishonest principal when the value of challenge sent by Bob and value of challenge assumed by the Alice are different, protocol is going to give different values for $value1$ and $value2$, it is not the case always. In such situation authentication is going to fail. This is one of the major flaw noticed in the Fiat-Shamir protocol by this work. It is proved by running the second CTL specification $AF(alice.challenge! = bob.challenge- > AF(value1! = value2))$. While running it is noticed that NuSMV gives the result as FALSE. It makes sure that in some cases for different values of c it is not at all possible that $value1$ and $value2$ takes the different values.

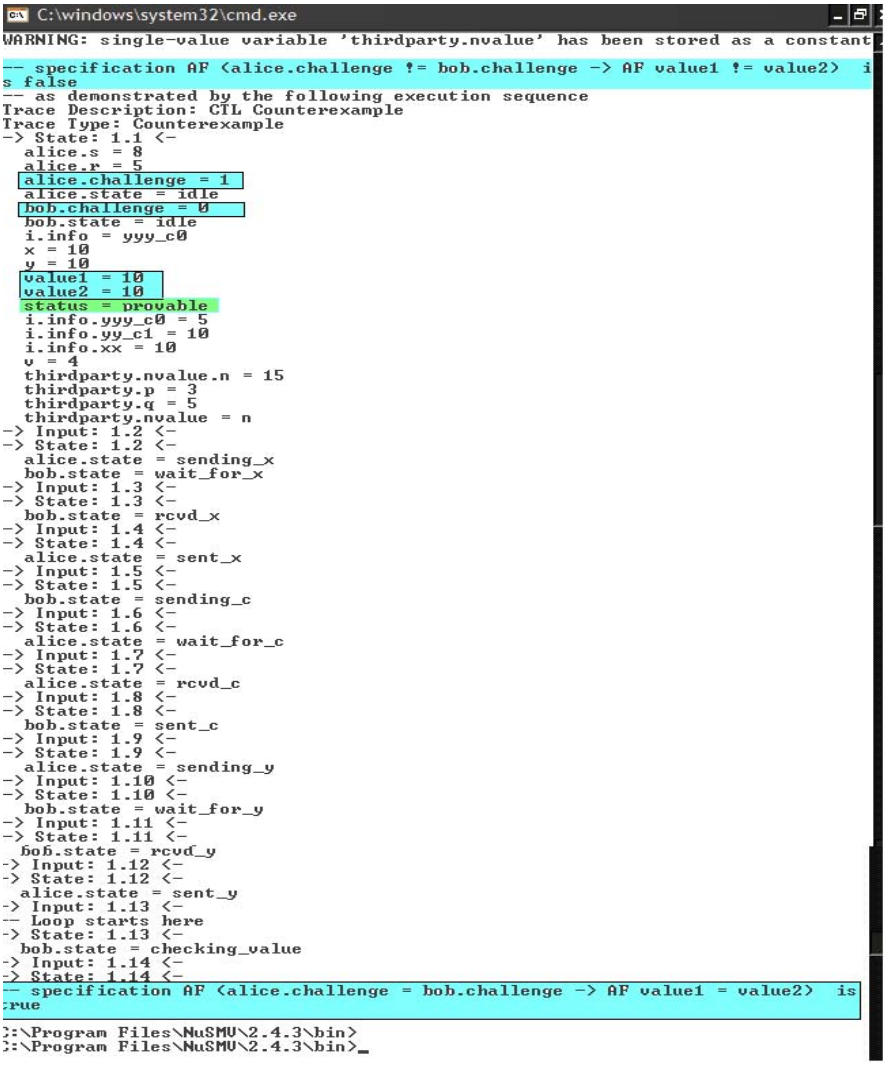

Fig. 3. Observation of Protocol's Flaw

The Fig.3 shows the verification output of Fiat-Shamir protocol where the protocol has failed. As we see that the values of challenge of claimant (Alice) and verifier (Bob) are not equal (i.e 1, 0 respectively) but the $value1$ and $value2$ (i.e 10) are still equal, that means if claimant does not have the secret key even though she could pass the test. It also shows that for some values of challenge the protocol passes the test that means Fiat-Shamir protocol could identify that whether claimant has the secret key or not.

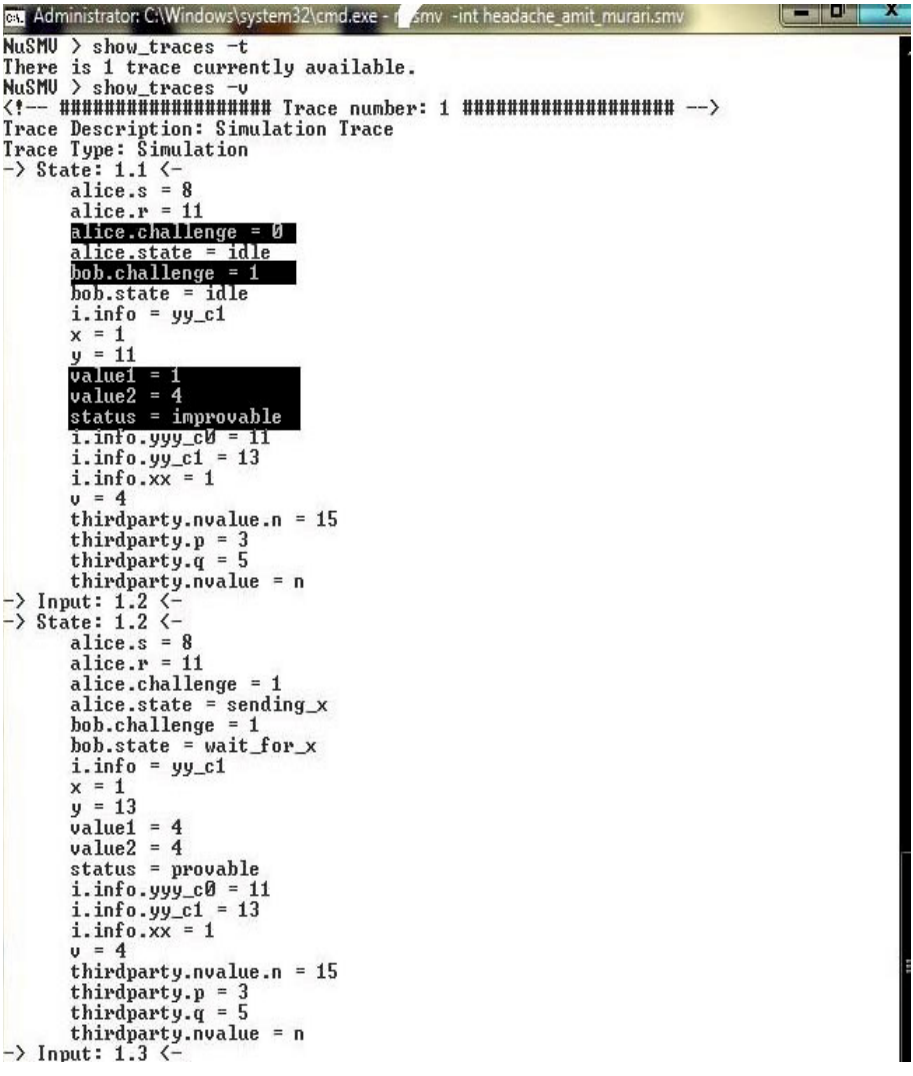

Fig. 4. Observation of Protocol

Fig.4 shows one of the observation of the verification where challenge guessed by Alice and challenge sent by Bob are different (0 and 1). *value*1 and *value*2 calculated by Bob are different (1 and 4) and hence the claimant (Alice) is get disqualified.

5 Conclusion

It has been observed that among the various methodologies proposed, model checking has been proved to be very useful to avoid the misconceptions arising

in the design of protocols. In this paper we have tried to model check Fiat-Shamir authentication protocol using NuSMV as the symbolic model verifier. We could able to check the initial configuration of the protocol. Also we could find out the limitation of the Fiat-Shamir protocol. This major flaw has been proved using CTL specification over the model.

References

1. Wikipedia: Zero knowledge proof (2011), http://en.wikipedia.org/wiki/Zero-knowledge_proof/
2. Forouzan, B.A.: Cryptography & Network Security, 1st edn. McGraw-Hill Press, United Kingdom (2008)
3. Cimatti, A.: Industrial Applications of Model Checking. In: Cassez, F., Jard, C., Rozoy, B., Dermot, M. (eds.) MOVEP 2000. LNCS, vol. 2067, pp. 153–168. Springer, Heidelberg (2001)
4. Huth, M., Ryan, M.: Logic In computer science, 2nd edn. Cambridge University Press, New Delhi (2004)
5. Cavada, R., et al.: Nusmv 2.5 user manual (2010), http://nusmv.fbk.eu/NuSMV/userman/v25/nusmv.pdf
6. Cavada, R., et al.: Nusmv 2.5 tutorial (2010), http://nusmv.fbk.eu/NuSMV/tutorial/v25/tutorial.pdf