Journal of Information Security and Applications 42 (2018) 71–86

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Formal analysis of a security protocol for e-passports based on rewrite theory specifications☆

Manjukeshwar Reddy Mandadi[a], Varuneshwar Reddy Mandadi[b], Kazuhiro Ogata[c,*]

[a] Amrita Vishwa Vidyapeetham (Amrita University) Amritanagar, Ettimadai, Coimbatore, Tamil Nadu 641112, India
[b] Indian Institute of Science (IISc) Bangalore, CV Raman Rd, Devasandra Layout, Bengaluru, Karnataka 560012, India
[c] Japan Advanced Institute of Science and Technology (JAIST) 1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan

ARTICLE INFO

Article history:

2010 MSC:
00-01
99-00

Keywords:
Authentication
e-Passport
Key exchange
Maude
Model checking
Rewriting

ABSTRACT

We report on a case study in which Password Authentication Connection Establishment (PACE) protocol has been formally analyzed based on its rewrite theory specification with Maude, a rewriting logic-based computer language and system. Dominik Klein has formally verified with interactive theorem proving that PACE enjoys the key secrecy property under the condition that the password shared by a passport chip C and a terminal T would be never leaked to the third party. In contrast, our analysis supposes that the password is leaked to an intruder once it has been used in a session completed. Under the condition, the analysis unveils some security weakness that PACE does not enjoy the correspondence (or authentication or agreement) properties from both C and T points of view. Then, we propose that one-time password is used in PACE. We have formally analyzed that the revised version enjoys the correspondence properties under the latter condition. We have used the Maude `search` command that can be used to conduct reachability analysis because the correspondence properties can be formalized as invariant properties.

© 2018 Elsevier Ltd. All rights reserved.

1. Introduction

Passports are travelers' documents that have been issued by the passport holders' country government and certify the holders' identities. International travelers must have their passports to pass the immigration control each time they enter and leave each country. The immigration control carefully examines each passport to check if the passport has not been counterfeited, the passport holder is exactly the same as the person whose personal information is recorded in the passport, the passport holder is allowed to enter or leave the country, etc. To make it much more difficult to counterfeit passports, machine readable passports, or e-passports have been introduced. Terminals at the immigration control communicate e-passports to read the data stored in the e-passports. The communication must be secure. To this end, cryptographic protocols, or security protocols could be used.

Password Authentication Connection Establishment (PACE) protocol was invented by German Federal Office for Information Security (BSI)(TR-03110) for this aim, and adopted as the international standard by International Civil Aviation Organization (ICAO)(Doc 9303). A passport chip (C) and a terminal (T) at the immigration control securely share a secret password in advance somehow in PACE. Intermediate keys are exchanged twice in each PACE session, making C and T share some final keys. Diffie–Hellman (DH) key-exchange protocol [1] is supposed to be used in the second key exchange, while any other key-exchange could be used in the first key exchange. DH protocol is supposed to use elliptic curves instead of the multiplicative group of integers modulo a number due to the lower computation burden.

PACE is formalized as a state machine, which is described as a rewrite theory specification in Maude [2]. State transitions are described as rewrite rules in rewrite theory specifications. Maude is a rewriting logic-based computer language, a direct successor of OBJ3 [3], an algebraic specification language. Complex systems could be succinctly specified as state machines in Maude because associative and/or commutative binary operators are conveniently used to express systems states. The Maude system is equipped with many commands to analyze rewrite theory specifications, one of which is the `search` command. The `search` command searches the reachable states of a state machine in a breadth-first manner from a given (initial) state for states that match a given pattern and/or satisfy a given condition. The `search` command can be used to conduct invariant model checking for state machines described as rewrite theory specifications. The `search` command is used to model check that PACE (formalized as a state

☆ This work was partially supported by Japan Society for the Promotion of Science Kakenhi Grant Number 26240008.

* Corresponding author.
E-mail address: ogata@jaist.ac.jp (K. Ogata).

https://doi.org/10.1016/j.jisa.2018.08.005
2214-2126/© 2018 Elsevier Ltd. All rights reserved.

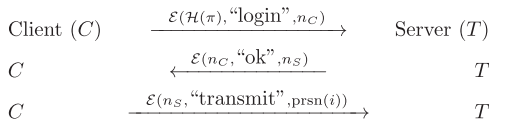

Fig. 1. Messages exchanged by a naive password login protocol at the ith login session.

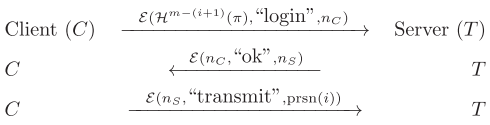

Fig. 2. Messages exchanged by a one-time password protocol at the ith login session for $i = 0, \ldots, m-1$.

machine and described as a rewrite theory specification) enjoys some security properties. The analysis takes into account the existence of an intruder that tries to glean information from messages in the network and fakes messages based on the gleaned information. Maude is also equipped with an LTL model checker and an Inductive Theorem Prover (ITP). Because the security properties we are interested in can be formalized as invariant properties, it is unnecessary to use the LTL model checker. One main goal of the research described in the present paper is to unveil some security weakness owned by PACE under some condition. This is why we have not used ITP but used the `search` command. Note that it is necessary to specify protocols as equational theory specifications so as to use ITP, while it is necessary to specify protocols as rewrite theory specifications so as to use the `search` command and the LTL model checker.

Klein [4] has formally verified with interactive theorem proving that PACE enjoys the key secrecy property under the condition that the password shared by C and T would be never leaked to the third party. In the analysis described in the present paper, in contrast, we suppose that once the password shared by C and T has been used in a session completed, it happens to be leaked to the intruder. Under this assumption, the analysis reveals that PACE does not enjoy the correspondence properties from both the T point of view and the C point of view, implying that keys exchanged by C and T with PACE may be leaked to the intruder. The correspondence properties are called the authentication properties or the agreement properties [5]. The correspondence property from the T (or C) point of view is that whenever T (or C) has completed a session to authenticate C (or T), C (or T) has also completed the same session to authenticate T (or C). After that, a revision of PACE such that one-time password is adapted is introduced. The analysis says that the revised version enjoys the properties when there are one passport chip, one terminal and one intruder, at most two sessions are made, and a password is leaked to the intruder once it has been used in a session completed. All experiments reported in the paper were conducted on a computer with 3.4 GHz processor and 32GB memory.

The rest of the paper is organized as follows. Section 2 describes some preliminaries: security protocols, state machines and Maude. Section 3 describes PACE. Section 4 reports on the first case study in which the original version is analyzed. Section 5 proposes a revised version and reports on the second case study in which the revised version is analyzed. Section 6 mentions some related work, and finally Section 7 concludes the paper.

2. Preliminaries

2.1. Security protocols

Let us consider a naive password login protocol. We suppose that a client (C) and a server (S) have securely shared something secret denoted π somehow in advance and use π multiple times for a long period of time. Fig. 1 shows messages exchanged in this protocol. When C wants to login S at the ith time, C computes the hashed value $\mathcal{H}(\pi)$ of π, generates a fresh nonce n_C, a cryptographically secure pseudo-random number, makes the cipher $\mathcal{E}(\mathcal{H}(\pi), \text{“login”}, n_C)$ by using $\mathcal{H}(\pi)$ as a symmetric key, where $\mathcal{E}$ is a symmetric encryption function, and sends the cipher to S. On receipt of the cipher by S, S tries to decrypt it with $\mathcal{H}(\pi)$ because S knows π and can compute $\mathcal{H}(\pi)$. If successfully decrypted, S finds the string “login” and a nonce n. S then generates a fresh nonce n_S, makes the cipher $\mathcal{E}(n, \text{“ok”}, n_S)$ with n as a symmetric key, and sends it to C. On receipt of the cipher by C, C tries to decrypt it with n_C generated by C for the ith session. If successfully decrypted, C finds the string “ok” and a nonce n'. C then makes the cipher $\mathcal{E}(n', \text{“transmit”}, \text{prsn}(i))$ with n' as a symmetric key, where prsn(i) is some personal information to be transmitted to S at the ith session, and sends it to S. On receipt of the cipher by S, S tries to decrypt it with n_S generated by S for the ith session. If successfully decrypted, S finds the string “transmit” and some data that must be prsn(i). As prsn(i) has reached S, the ith session is over. We suppose that there exists an intruder and the intruder happens to know the password π somehow as the first session is over. We also suppose that if $\mathcal{E}(\mathcal{H}(\pi), \text{“login”}, n_C)$ is in available in the network and the intruder knows π, the intruder may fake the cipher $\mathcal{E}(n_C, \text{“ok”}, n_{intr})$ and sends it to C by impersonating S, and if $\mathcal{E}(n', \text{“transmit”}, \text{prsn}(i))$ is available in the network and the intruder knows n', the intruder can glean prsn(i). Then, the question is whether the intruder could know prsn(i) in the ith session. We will answer the question later.

Let us next consider a one-time password login protocol based on what has been invented by Lamport [6]. We suppose that a client (C) and a server (S) have securely shared something secret denoted π somehow in advance and agreed on how many times (m) C would login S. Note that π must be protected but there is no problem if m would be known by someone else. When no session has been made by C and S, the intruder does not know π and then cannot compute $\mathcal{H}^{m-1}(\pi)$ even if the intruder happens to know m. Even when $\mathcal{H}^{m-1}(\pi)$ is leaked to the intruder after the completion of the first session, the intruder can compute $\mathcal{H}^{m}(\pi)$ but can never compute $\mathcal{H}^{m-2}(\pi)$. S computes a series of hashed values based on π. Let *hvseq* be the sequence $\mathcal{H}^{m-1}(\pi); \ldots; \mathcal{H}(\pi); \pi$, where $\mathcal{H}$ is a cryptographically secure hash function. *hvseq* must be securely stored in S, while there is no problem even if $\mathcal{H}$ would be disclosed. Fig. 2 shows messages exchanged in the protocol. When C wants to login S at the ith time, where $i = 0, \ldots, m-1$, C computes $\mathcal{H}^{m-(i+1)}(\pi)$, generates a fresh nonce n_C, makes the cipher $\mathcal{E}(\mathcal{H}^{m-(i+1)}(\pi), \text{“login”}, n_C)$ by using $\mathcal{H}^{m-(i+1)}(\pi)$ as a symmetric key, and sends the cipher to S. On receipt of the cipher by S, S tries to decrypt it with the top of *hvseq* as a symmetric key. If successfully decrypted, S finds the string “login” and a nonce n. S then generates a fresh nonce n_S, makes a cipher $\mathcal{E}(n, \text{“ok”}, n_S)$, and send it to C. On receipt of the cipher by C, C tries to decrypt it with n_C generated by C for the ith session. If successfully decrypted, C finds the string “ok” and a nonce n'. C then makes the cipher $\mathcal{E}(n', \text{“transmit”}, \text{prsn}(i))$ with n' as a symmetric key and sends it to S. On receipt of the cipher by S, S tries to decrypt it with n_S generated by S for the ith session. If successfully decrypted, S finds the string “transmit” and some data that must be prsn(i). As prsn(i) has reached S, the ith session is over and S throws away the top from *hvseq*. We suppose that there exists an intruder and the intruder happens to know the password used for the ith session every time the ith session is over. The intruder would do the same things as what have been described for the naive password login protocol. Then, the question is whether the intruder could

Download English Version:

https://daneshyari.com/en/article/9952281

Download Persian Version:

https://daneshyari.com/article/9952281

Daneshyari.com