On the Automated Implementation of Modal Logics Used to Verify Security Protocols

Tom Coffey, Reiner Dojen, Tomas Flanagan
Data Communication Security Laboratory
Department of Electronic and Computer Engineering
University of Limerick, Ireland

tom.coffey@ul.ie, reiner.dojen@ul.ie, tomas.flanagan@ul.ie

Abstract: Formal verification provides a rigid and thorough means of evaluating the correctness of cryptographic protocols so that even subtle defects can be identified. As the application of formal techniques is highly involved, software has been developed in order to facilitate protocol verification. Protocol weaknesses or flaws can thus be identified and corrected during the design process. In this paper the verification process is illustrated by analysing the ASK protocol using the tool AAPA2. The virtues and limitations of the tool are discussed. Overall, the analysis shows that the use of automated tools in verifying security protocols offers significant advantages to the protocol designer.

Keywords: Security Protocols, Protocol Verification, Automated Verification, Authentication, Mobile Security

1. Introduction

Network security encompasses the cryptographic protocols and algorithms used to ensure secure communication in a hostile environment. Cryptographic security protocols find applications in many areas such as Internet and mobile communications security. Traditionally, cryptographic protocols have been designed and verified using informal and intuitive techniques. However, the absence of formal verification of these protocols can lead to flaws and weaknesses remaining undetected. Formal verification techniques provide a rigid and thorough means of evaluating the correctness of a cryptographic protocol so that even subtle defects can be discovered. Hence, they facilitate the design and testing of general-purpose cryptographic protocols. However, the application of formal verification techniques is highly complex, tedious and prone to errors. Software tools such as Athena[1], TABS [2], AAPA2 [3], Huima's Model Checker [4], FDR Model Checker [5], and the NRL Protocol Analyzer [6] have been developed in an attempt to automate the formal verification process.

One of the most effective formal method of protocol verification is the use of modal logics. This technique has successfully identified several previously unknown protocol flaws, e.g. [7], [8], [9]. AAPA2, an automated verification tool based on a variation of GNY, has been used to verify a variety of security protocols [3]. This paper presents the verification of the ASK protocol [10] using AAPA2. This verification helps to demonstrate the virtues and limitations of automated security protocol verification.

2. The Formal Verification Process

The generic approach to logic-based formal verification involves the following steps:

1. formal specification of the protocol steps in the language of the logic
2. formal specification of the protocol assumptions
3. formal specification of the protocol goals
4. application of the axioms and inference rules of the logic to assumptions and protocol steps to derive the goals

Once a protocol has been successfully verified it can be considered secure within the limitations of the used logic. (For example we are not aware of any logic which can be used to prove confidentiality.) Conversely, even the results of a failed verification are helpful, as these results point to missing assumptions or weaknesses in the protocol. This gives an opportunity to address these identified weaknesses.

3. Automatic Authentication Protocol Analyzer II (AAPA 2)

The Automatic Authentication Protocol Analyzer, version II [3] is a tool for analysing cryptographic protocols. It is based on an extension of the GNY logic [8] and uses the HOL proving engine [11] to automatically determine whether a protocol meets its specifications. The new features of the extended logic in AAPA2 are intended to facilitate automation and to address limitations of the GNY logic. An example of such changes is the introduction of automatic type checking which removes the requirement for GNY-style recognisability assumptions. However, as the GNY logic has been modified in AAPA2, a proof using the tool is not equivalent to a manual proof using GNY.

AAPA2 features an intermediate specification language (ISL2), intended to aid the protocol designer with steps one to three of the verification process. The user must input formal specifications of protocols and their security goals using ISL2 and the software automatically verifies in minutes whether these security goals are met. However, the software offers only limited assistance in creating the formal protocol model, as the designer must implicitly interpret the semantics of the messages exchanged.

Step four of the verification process is performed automatically. If the AAPA2 cannot prove a given goal, it produces a failure message. If the AAPA2 discovers a flaw in a protocol, it writes the flaw to an output file and continues with the analysis of the protocol. This allows the software to detect multiple flaws in a protocol, should they exist. The output produced by AAPA2 describes any failures that are encountered and all of the theorems that are proved successfully.

4. Formal Verification of the ASK Protocol

Several new protocols have been proposed for wireless communications [12]. Among these is the ASK protocol [10], designed by Aydos, Sunar and Koc, which was the first to use elliptic curve

cryptography for authentication and key agreement. The following notation is used to describe this protocol:

U	an identifier of the user
V	an identifier of the service provider
S	an identifier of the certification authority
KK	a key encrypting key
SK	a session key
Kx	public key of X
^Kx	private key of X
Rx	a random nonce generated by X
CertX	certified public key for principal X
(X,Y)	Conjunction of two formulae
{X}E (K)	Symmetric encryption and decryption
{X}E (Kx)	Public key encryption/decryption
{X}E (^Kx)	Private Key encryption/decryption

4.1 Protocol Description

The message exchanges of the ASK protocol are as follows:

V → U: Kv
U computes KK={Kv}E (^Ku)
U → V: Ku
V computes KK = {Ku}E (^Kv), SK = KK + Rv
V → U: {CertV, Rv}E (KK)
U computes SK = KK + Rv
U → V: {CertU, Rv}E (KK)

During the first two steps of the ASK protocol, the principals exchange public keys. At this stage, each principal can compute the key encryption key, KK, using the Diffie-Hellman technique. Next, the service provider, V, generates a nonce, Rv, which is added to KK to create the session key. V sends a message to the user, containing the nonce Rv and his certificate, encrypted under KK. The certificate contains the identity and public key of principal V along with an expiration time, all signed by the certification authority. Upon receipt of Rv the user can calculate the session key. U can also verify that the public key, received earlier, belongs to the service provider by examining the certificate. The protocol concludes with a message from the user to the service provider containing CertU and Rv to provide authentication of the user.

4.2 Modelling ASK using AAPA2

Difficulties arose in modelling the certificates used to share public keys. To address these issues, each principal assumes that the validity interval on the other's time stamp assures freshness. Further, the protocol assumes that each principal has obtained its certificate from the certification authority prior to the protocol run. In order for the software model to function correctly, it was necessary to include the messages in which each principal obtains its certificate.

4.2.1 Assumptions:

The basic assumptions for the protocols are similar to those required when using the GNY logic directly. However, the software applies automatic type-checking and as a result does not require the assumption that formulae are recognisable. Also, the logic assumes that all encryption functions and principal identities are publicly available. The AAPA2 software requires such assumptions to be made explicit.

It is assumed that the certification server, S, possesses the identities and certificates of the protocol participants, U and V. Further, S believes that the appropriate public key belongs to U and V respectively. Both U and V possess the identities U, V and S and the encryption function E. They also possess their own private keys and the public key of S. Both principals believe Ks to be S's public key and that S is trustworthy. It is further assumed that U and V can check the validity intervals on the certificates they obtain. Finally, V possesses a nonce Rv which it believes to be fresh.

4.2.2 Protocol Goals:

The ASK protocol is aimed at providing mutual authentication and key agreement in low-cost, low-power portable devices such as mobile phones. At the end of a successful protocol run each principal should possess the other's public key and believe that this key belongs to the other principal. In addition, the participants of the protocol should be satisfied that the session key is a suitable key for communication and that it is fresh. Each principal should also be able to confirm that the other protocol participant possesses the session key and believe it to be a shared secret.

4.3 Results

A summary of the achieved goals and detected failures is given below:

Failures:

F1: Goal statement: U Believes (Fresh Ku_);
F2: U Believes (V Conveyed Rv);

Goals Proved:

V Possesses Kv_;
U Possesses Ku_;
U Believes SharedSecret U V KKu;
U Believes SharedSecret U V Ku_;
V Believes SharedSecret U V KKv;
V Believes SharedSecret V U Kv_;
V Believes Fresh Kv_;
V Believes U Conveyed {Rv,CertU}E1(KKv);

Warnings:

Principals U believes term ValIntKv is fresh, but did not create this term.
Principals V believes term ValIntKu is fresh, but did not create this term.

AAPA2 identifies two legitimate failures in the ASK protocol. The user cannot derive the belief that the session key is fresh, as the key does not contain anything which the user believes to be fresh. As a result, U does not believe that the V sent the message containing Rv and CertV. The third protocol message could be a replay from an old protocol run [12].

The software successfully proves that both principals possesses the key-derivation key and the session key and believe these keys to be suitable shared secrets for communication. V also believes the session key to be fresh and that U conveyed the final protocol message. The warnings occur because the software has difficulties modelling the validity intervals marked on the certificates.

5. Discussion and Conclusions

Our analysis of the ASK protocol highlighted both the merits and limitations of the software. The analysis successfully identifies a known weakness in the protocol. Some warnings were given due to difficulties in modelling the certificates used to distribute the public keys. On the whole, our results show that the use of software, such as AAPA2, is advantageous to the protocol designer.

Many protocol designers are discouraged from using formal verification techniques due to their complexity. Thus, the development of software, which facilitates the protocol designer in performing such verification, is significant. As the process becomes easier to carry out, greater numbers of protocols will be formally verified. This will reduce the number of insecure protocols in the public domain.

The automatic application of the axioms removes the requirement that the protocol designer be familiar with all of these axioms. Further, the possibility of errors due to poor understanding of the axioms is removed. Such errors could lead to the successful verification of insecure protocols.

The core purpose of tools such as AAPA2 is to allow non-experts in the area of modal logics to formally verify protocols. Ideally, the system would be intuitive and easy to use. However, in order to use the tool, a good understanding of the underlying logic is required. Further, the considerable experience is required to use the tools effectively. As a result, the tool is most beneficial if several protocols must be verified. Major improvements in user-friendliness will be required in order to encourage increased use of the software.

The main aim of this paper was to evaluate the suitability of tools based on formal logics for the purpose of protocol verification. Overall, the analysis shows that the use of software to verify protocols offers significant advantages to the protocol designer. However, limitations of current tools must be addressed before such tools will find widespread use.

References

[1] Song, D., Berezin, S. and Perrig, A., "Athena: A Novel Approach to Efficient Automatic Security Protocol Analysis", *Journal of Computer Security*, Vol.9, No.1/2, 2001, pp.47-74

[2] Cohen, E., "TAPS: A first order verifier for cryptographic protocols", *IEEE Computer Security Foundations Workshop*, 2000, pp.144-158

[3] Brackin, S., "Automatically Detecting Most Vulnerabilities in Cryptographic Protocols." *DARPA Information Survivability Conference and Exposition*, IEEE CS Press, Vol. I, 2000, pp.222-236

[4] Huima, A., "Efficient Infinite-State Analysis of Security Protocols", *Workshop on Formal Methods and Security Protocols FLOC'99*, 1999

[5] Lowe, G., "Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR", *Software Concepts and Tools*, Vol.17, No.2, 1996, pp.93-102

[6] Meadows, C., "The NRL Protocol Analyzer: An Overview", *Journal of Logic Programming"*, Vol.26, No.2, 1996, pp.113-131

[7] Burrows, M., Abadi, M. and Needham, R. "A logic of authentication", *DEC System Research Centre Report* No. 39, February 1989

[8] Gong, L., Needham, R. and Yahalom, R., "Reasoning about belief in Cryptographic Protocols", *1990 IEEE Computer Society Synopsis on Research in Security and Privacy,* 1990, pp.234-248

[9] Coffey, T. and Saidha, P., "A Logic for Verifying Public Key Cryptographic Protocols." *IEE Journal Proceedings-Computers and Digital Techniques*, Vol.144, No.1, 1997, pp.28-32

[10] Aydos, M., Sunar, B., and Koc, C. K., "An elliptic curve cryptography based authentication and key agreement protocol for wireless communication", *International Workshop on Discrete Algorithms and Methods for Mobility (DIAL M 98)*, 1998

[11] Gordon, M. HOL: "A proof generating system for higher order logic", *Technical Report 103, Cambridge University,* 1987.

[12] Horn, G., Martin, K. and Mitchell, C., "Authentication Protocols for Mobile Network Environment Value-added Services", *IEEE Transactions on Vehicular Technology*, Vol.51, No.2, 2002, pp.383-392