Towards a Framework for Autonomic Security Protocols

Simon N. Foley and Hongbin Zhou

Boole Centre for Research in Informatics,
Department of Computer Science,
University College, Cork, Ireland.
{s.foley,zhou}@cs.ucc.ie

Abstract. This paper proposes a belief logic based approach that allows principals to negotiate and on-the-fly generate security protocols. When principals wish to interact then, rather than offering each other a fixed menu of 'known' protocols, they negotiate and generate a new protocol that is tailored specifically to their current security environment and requirements. This approach provides a basis for autonomic security protocols. Such protocols are self-configuring since only principal assumptions and protocol goals need to be a-priori configured. The approach has the potential to survive security compromises that can be modelled as changes in the beliefs of the principals. A compromise of a key or a change in the trust relationships between principals can result in a principal self-healing and synthesising a new protocol to survive the event.

1 Introduction

Networked services and applications are typically commissioned with a fixed repertoire of security protocols that provide for necessary authentication, key-exchange, non-repudiation, delegation, and so forth. Applications and services are expected to negotiate with each other and agree on appropriate security protocols that both can (and are willing to) use. A simple example is a web-service that requires clients to establish SSL based connections. When negotiating a connection, the client and server agree on the version of the protocol to use. While protocols may be designed to support a range of different underlying authentication protocols, and so forth, it is not feasible to expect principals to be, a priori, conversant in all possible protocols. Protocol agnostic approaches such as Jini [8] allow resource providers to register the protocol, that its clients should use, with a Jini Server. While more flexible, the provider's protocol is fixed and is not generally suitable for security protocols.

We are investigating the use of protocol synthesis techniques [5, 7, 11, 13] to allow principals negotiate and on-the-fly generate security protocols. When principals wish to interact then, rather than offering each other a fixed menu of 'known' protocols, the protocol negotiation process generates a new protocol that is tailored specifically to their current security environment and requirements. A principal's security environment reflects the keys that it knows, the trust relationships with other principals and any other assumptions it holds.

We conjecture that applications that use such *Self-configuring* protocols would not require configuration to provide a fixed number of 'known' protocols. Instead, the application designer specifies the necessary security requirements for valid interaction. For example, a server may be willing to accept any connection from an authenticated principal; the security requirements and current environment of the server (and client) are used to synthesis a protocol that meets these goals.

Protocols could be generated on the basis of the security environment of the principals. A change in the security environment of a principal may result in the re-negotiation of a new security protocol. This provides a basis for *survivable* security protocols that have the potential to, in effect, *self-heal* and adapt to recover from changes in the security environment.

These characteristics—self-configuring, self-healing and survivability—are are properties that form part of the autonomic computing manifesto [2]. In the next section we outline our current research on this area.

2 Autonomic Security Protocols

The Basic Protocol Synthesis Protocol (BPSP) is a bootstrapping protocol that is used by principals to negotiate and generate a new protocol specification. It is used when an Initiator requests connection to a Responder. Principals specify their protocol goals and assumptions using the Simple Logic [4].

The Simple Logic is a BAN-style belief logic that uses abstract channels similar to the Spi Calculus [1] to represent keyed communication between principals. This results in a very simple, yet expressive, logical system. A synthesis technique is also proposed [4] that can be used to guide the (manual) systematic calculation of a protocol from its goals.

As part of our current research we are exploring automated protocol generation techniques such as [5, 11, 7] can be used within our framework. We have developed an automatic verification tool [10] for the Simple Logic and implemented using Theory Generation [9]. This tool has been

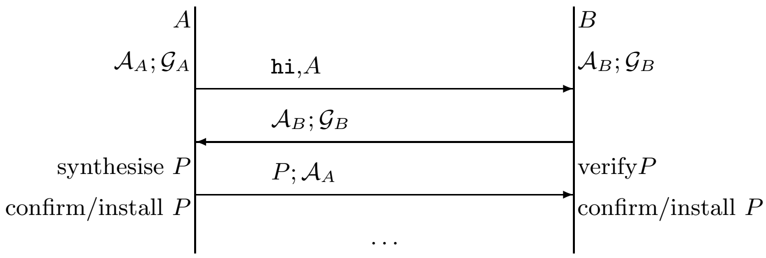

Fig. 1. Basic Protocol Synthesis Protocol (BPSP)

extended [13] to support the automated synthesis/generation of protocols. The approach in [13] combines and automates the manual synthesis rules from the Simple Logic with Guttman's [7] manual design process. We adapt the synthesis rules of the Simple Logic to guide an automatic backwards search for a sub-protocol from a single goal. Given a number of individual goals, an automated technique is described to combine synthesised sub-protocols into final candidate protocols.

Figure 1 depicts the basic Protocol Synthesis Protocol. A protocol initiator A requests connection to B. B responds by passing details of its protocol goal $\mathcal{G}_B$ and the assumptions $\mathcal{A}_B$ that it currently holds. Principal A uses its own assumptions and those presented by B to synthesise a new protocol P that meets their respective goals. This protocol is returned to B, which attempts to validate using the validation tool. If validation is successful then A and B install and engage in the protocol P.

Example 1 Consider a service B that expects connections to be authenticated. The protocol requirement for the server is that a connecting principal Q should be authenticated. This is expressed in the logic as the goal

$$\mathcal{G}_B(Q) \stackrel{\Delta}{=} B \models Q \Vert\!\sim (B, Nb) \tag{1}$$

where N_B is a nonce. For reasons of space we do not describe the logic in detail, however, the connectives '$\models$' and '$\Vert\!\sim$' have the usual interpretation [6] of 'believes' and 'recently said' in this run of the protocol.

The assumptions of B reflect its belief that it knows A's public key K_A:

$$\begin{aligned} \mathcal{A}_B \stackrel{\Delta}{=} & B \models w(C_A) = \{A\}; B \in r(C_A); \\ & B \models \# N_B; \end{aligned}$$

where C_A is the channel established by the public/private key pair K_A, K_A^{-1}, and may be written only by A and read by B (and possibly others).

For simplicity, we assume that A has no goals, and for the moment its assumptions are not relevant. On receipt of B's protocol goal $\mathcal{G}_B(A)$ and assumptions $\mathcal{A}_B$, the connecting client A can immediately synthesise an (idealised) protocol P that corresponds to

$$\begin{aligned} P \triangleq\ & B \to A : N_b; \\ & A \to B : \{B, N_b\}_{K_A^{-1}}; \end{aligned}$$

which, in turn can be verified by B. Assuming that it is possible for the principals to automatically engineer a protocol implementation from P, then the principals install their protocol components and execute it. $\triangle$

In the above example it is possible for A and B to negotiate a new protocol without the participation of other principals. The protocol synthesis protocol will have to handle situations where the participants need the assistance of third parties, both in terms of deriving protocols and in implementing protocols.

Example 2 Suppose that C shares a secret key K_{AC} (securing channel C_{AC} and requests a connection with A. Suppose further that B trusts A in the sense that $\mathcal{A}_B$ includes

$$B \models (A \Vdash\!\sim \phi_1) \Rightarrow (A \models \phi_1) \qquad (2)$$

$$B \models (A \Vdash\!\sim (C \mid\!\sim \phi_2)) \Rightarrow (C \mid\!\sim \phi_2) \qquad (3)$$

for arbitrary ϕ_1, ϕ_2. These formulae reflect B's belief that A is honest and that A is competent in deciding whether C at some time in the past said ('$\mid\!\sim$') some message. Given these beliefs, then C (given $\mathcal{A}_B$) can synthesise the protocol

$$\begin{aligned} P_2 \triangleq\ & B \to C : N_b \\ & C \to A : \{B, N_b\}_{K_{AB}} \\ & A \to B : \{C, B, N_b\}_{K_A^{-1}} \end{aligned}$$

The basic protocol synthesis protocol (Figure 1) must be extended to include participation of A in the setting up and execution of the final protocol. $\triangle$

Note that the Basic Protocol Synthesis protocol as described is potentially vulnerable to attack. In this case an attacker tricks one of (or both) the principals into using a different protocol. Having generated a protocol P, techniques such as [3] may prove useful in making the protocol robust against such protocol-identity attacks. This requirement may also be expressed as additional goals that should be verified by the principles B and A, respectively,

$$B \models A \mid\!\sim P$$
$$A \models B \models A \mid\!\sim P$$

We assumes that the assumptions and goals of the principals are included as part of the specification of P (this ensures that $B \models A \mid\!\sim \mathcal{A}_A$, and so forth).

3 Discussion

We have sketched an architecture that is proposed to support on the fly synthesis of protocols. This approach provides a basis for autonomic security protocols. Such protocols are self-configuring since only principal assumptions and protocol goals need to be a-priori configured. The architecture has the potential to survive security compromises that can be modelled as changes in the beliefs of the principals. A compromise of a key or a change in the trust relationships between principals can result in a principal self-healing and synthesising a new protocol to survive the event. For example, if B no longer trusted A on what C said in the past then (Formula (3)) then a new protocol must be derived.

Much work remains to be done on this architecture on a number of fronts. We have slightly extended the Simple Logic and synthesis framework to better suit the task at hand. In tests, the protocol generation tool has performed well [13]: given mutual authentication/key exchange goals and assumptions that indicate a trusted third party, then the tool can generate approximately 500 valid (4/5 message) mutual authentication protocols within 40 seconds. On inspection, many of these candidate protocols are similar containing minor textual and redundant variations; we estimate that in this set there are 24 reasonable distinct four-message protocols and 76 reasonably distinct five-message protocols. Whether there is sufficient variety in such protocols for them to be used as "session protocols" is a topic for future research.

Currently, the architecture expects the protocol initiator to synthesise the final protocol. One challenge is to determine how assumptions held by other principals can best be discovered by initiating principals, and whether it would be better to hand-off sub-goals to other principals to synthesise.

Other future research includes exploring how to automatically translate an idealised protocol into executable software components. Some previous work [12] exists on this problem that we hope to draw on.

Acknowledgments

This work is supported by the Boole Centre for Research in Informatics, University College Cork under the HEA-PRTLI scheme.

References

1. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.
2. A.G.Ganek and T.A.Corbi. The dawning of the autonomic computing era. *IBM Systems Journal*, 42(1):5–18, January 2003.
3. Tuomas Aura. Strategies against replay attacks. In *Computer Security Foundations Workshop*, pages 29–68, 1997.
4. L. Buttyán, S. Staamann, and U. Wilhelm. A simple logic for authentication protocol design. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop (CSFW '98)*, pages 153–163, Washington - Brussels - Tokyo, June 1998. IEEE.
5. John A Clark and Jeremy L Jacob. Searching for a solution: Engineering tradeoffs and the evolution of provably secure protocols. In *proceedings of 2000 IEEE Symposium on Security and Privacy (SP 2000)*, pages 82–95. IEEE Computer Society, 2000.
6. Li Gong, Roger Needham, and Raphael Yahalom. Reasoning About Belief in Cryptographic Protocols. In Deborah Cooper and Teresa Lunt, editors, *Proceedings 1990 IEEE Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society, 1990.
7. Joshua D Guttman. Security protocol design via authentication tests. In *proceedings of 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 92–103. IEEE Computer Society, 2002.
8. Sun Microsystem Inc. Jini technology core platform specification version 1.2. www.jini.org, November 2001.
9. D. Kindred and J.M. Wing. Theory generation for security protocols. *ACM TOPLAS*, July 1999.
10. D. O'Crualaoich and S.N. Foley. Theory generation for the simple logic. Technical report, University College Cork. In preparation.
11. Adrian Perrig and Dawn Song. Looking for diamonds in the desert — extending automatic protocol generation to three-party authentication and key agreement protocols. In *proceedings of the 13th IEEE Computer Security Foundations Workshop*, pages 64–76. IEEE Computer Society, 2000.

12. D. Song, A. Perrig, and D. Phan. Agvi–automatic generation, verification, and implementation of security protocols. In *proceedings of 13th conference on computer aided verification CAV 2001*, pages 241–245, July 2001.
13. H. Zhou and S.N. Foley. Fast automatic synthesis of security protocols using backward search. In *ACM Workshop on Formal Methods for Security Engineering*, Washington, DC, USA, 2003.