Using Process Algebras to Formally Specify Mobile Agent Data Integrity Properties: a Case Study

Xavier Hannotin, Paolo Maggi and Riccardo Sisto
Dip. di Automatica e Informatica - Politecnico di Torino
Corso Duca degli Abruzzi 24, I-10129 Torino, ITALY
email:maggi@athena.polito.it, sisto@polito.it

Abstract

This paper shows how cryptographic protocols for mobile agent data integrity properties can be formally specified by using spi calculus, an extension of π calculus with cryptographic properties. In particular, by means of a case study, it is shown how a specification technique initially conceived only for classical cryptographic protocols can be used in the context of mobile agents as well. Our case study includes the spi calculus specification of a sample mobile agent data integrity protocol and of its security property.

1. Introduction

The use of software architectures based on mobile agents to develop distributed computing systems is gaining more and more attention because it seems to exhibit advantages over the traditional client-server paradigm in several applications. However, to make mobile agents become a widely used new paradigm for distributed computing, it is necessary to manage the various security problems arising when it is adopted [6, 3]. Such problems generally fall into two main categories. On one hand, it is necessary to protect hosts from malicious agents coming from the network, on the other hand it is necessary to protect agents from malicious hosts and network intruders. While the first kind of problem has been studied extensively, at the moment only some partial solutions are available for the second one (e.g. [9, 7, 4, 2, 8]).

In general, all the solutions adopted to ensure security properties in distributed systems are based on some kind of cryptographic protocol which, in turn, uses basic cryptographic operations such as encryption and digital signatures. Despite their apparent simplicity, such protocols have revealed themselves to be very error prone, especially because of the difficulty generally found in foreseeing all the possible attacks and all the possible behaviors of various parallel protocol sessions. For this reason, a lot of attention is being paid to formal methods which can help in developing error-free protocols or in analyzing the vulnerability of existing protocols. Up to now, such methods have been successfully employed to formally verify security properties of classical message-based protocols, such as authentication protocols, whereas they have not yet been applied to analyze the security of agent-based systems. It is worth noting that the specification and verification of security issues related to mobile agent systems involves new aspects not encountered in classical cryptographic protocols. In particular, in addition to the classical threats, such as those related to possible alteration of messages being transmitted, it is also needed to model the fact that the correct behavior of an agent is not guaranteed when the agent is being executed in an untrusted host. This means that the behavior of an agent potentially becomes unpredictable each time it visits an untrusted host.

This paper addresses such new aspects and explores the possibility of applying existing formal specification techniques to those cryptographic mechanisms specifically designed for the protection of agents from their environment, with a particular emphasis on agent data integrity, which is the most typical property of interest.

2. A sample protocol for mobile agents data integrity

A mobile agent (MA) is a program that can migrate from one network host to another one while executing. Mobile agents are executed by agent interpreters that run on each host and communicate by message passing. An agent migrating from one host to another host consists of a static part, typically including the agent code and, possibly, some static data, and a dynamic part, including all the agent elements that can change over time (program counter, stack, variables, etc.).

In this paper, we show how to model using process algebras a simple protocol which aims at the integrity of a

data gathering mobile agent that runs on several possibly untrusted hosts. This agent goes to several hosts and simply picks up pieces of data on its way. For example, the agent could be a shopping agent dispatched to visit different companies and find out the prices at which they sell a given product, so as to select the cheapest company. Data integrity of such an agent means that a host cannot tamper with the data already collected without being detected. This is a classical problem for which different protocols have been proposed [9, 7, 4, 2, 8].

The specific protocol we consider here was proposed in [2] by Corradi et al.. The idea is that agents carry along a cryptographic proof of the data they have already gathered. This proof prevents hosts from tampering with the data already collected without being detected.

For the description of the protocol we use the following notation. $hash()$ is a cryptographic hash function, i.e. a function which, theoretically, cannot be inverted (x cannot be deduced from $hash(x)$). Following the spi calculus notation, the private and public part of the key of host H are denoted H^- and H^+ respectively. Encryption of data x by private (public) key of host H is denoted $\{[x]\}_{H^-}$ ($\{[x]\}_{H^+}$).

We also take some notation from [2], where MIC stands for "Message Integrity Code" and is the cryptographic proof we have just mentioned, C is a "cryptographic counter", which is incremented by successive applications of $hash$ by the agent and AD is the list of already collected data. The hosts where data have to be collected are decided by the agent dynamically, in such a way that each host is visited at most once. The hosts are denoted, in order of visit by the agent, H_0, which is the initiator, and H_i $(1 \leq i \leq n)$, which are the hosts where data must be collected. The initiator initially creates the agent, sends it out and, at the end of the computation, receives it with the collected data. Each host H_i $(1 \leq i \leq n)$ has a piece of data D_i that will be collected by the agent.

AD_i and MIC_i are respectively the collected data and the MIC value after the agent has left H_i. Similarly, successive values taken by the cryptographic counter are denoted C_i. CID is the (static) code of the agent. It is signed by a trusted party for authentication and it is carried along from host to host with the agent. The agent moving from host H_{i-1} to host H_i can be represented by a message containing CID, AD_{i-1}, MIC_{i-1} and C_i.

2.1. Informal protocol description

- Initialization: H_0 generates a secret number C_0. It creates the agent and passes $C_1 = hash(C_0)$ to it.
- First Hop: The agent encrypts C_1 with H_1^+ to let it be accessible only on H_1, and then moves to H_1, carrying with itself only the encrypted C_1 (i.e. $\{[C_1]\}_{H_1^+}$). The collected data list AD_0 and the initial MIC MIC_0 are empty.
- On host H_1: After the agent has reached H_1, it asks H_1 to decrypt $\{[C_1]\}_{H_1^+}$, thus obtaining C_1 and collects D_1, so having $AD_1 = \{D_1\}$. Then it computes $MIC_1 = hash(D_1, C_1)$, and it increments the cryptographic counter by computing $C_2 = hash(C_1)$.
- Hop i ($2 \leq i \leq n$): The agent encrypts C_i with H_i^+ and then moves to host H_i carrying with itself the already collected data AD_{i-1}, the cryptographic proof MIC_{i-1}, and the value of the cryptographic counter encrypted with H_i's public key: $\{[C_i]\}_{H_i^+}$.
- On host H_2 ($1 \leq i \leq n$): After having decrypted $\{[C_i]\}_{H_i^+}$, the agent collects D_i and appends it to the collected data list, so computing $AD_i = AD_{i-1} \cup \{D_i\}$. It then computes a new proof by:

  $$MIC_i = hash(D_i, C_i, MIC_{i-1})$$

  and increments the cryptographic counter.
- Last hop: The agent encrypts C_n with H_0^+ and then moves from H_n back to H_0 carrying with itself the whole data AD_n, the computed checksum MIC_n, and $\{[hash(C_n)]\}_{H_0^+}$.
- Termination: H_0 receives from the agent AD_n, MIC_n, and $\{[hash(C_n)]\}_{H_0^+}$. From the values of C_0 and AD_n, H_0 can compute MIC_n, and check that it is the same that has just been received from the agent. If any difference is found, the agent data is considered to be invalid. This guarantees the integrity of validly collected data.

A host cannot modify the already collected data, basically because it cannot retrieve C from $hash(C)$. Since H_i does not know C_j $(j < i)$, it cannot modify D_j $(j < i)$ and so it cannot reconstruct a valid MIC.

As also noted in [2], this solution does not work if the agent visits a host to collect data twice, or if malicious hosts cooperate. One advantage of the method is that it allows intermediate hosts to read the data already gathered (they are not encrypted).

3. Modeling a mobile agent system

Formal models of cryptographic protocols typically are composed of a set of principals which send messages to each other according to the protocol rules, and an intruder, representing the activity of possible attackers. The intruder can perform any kind of attack: it can not only overhear all the transmitted messages, learning their contents, but it

can also intercept messages and send new messages created using all the items it has already learned, as well as new nonces. So the intruder can fake messages and sessions.

Since such models are meant to reveal possible security flaws in the protocols and not flaws in the cryptosystems used by the protocols, cryptography is modeled in a very abstract way and it is assumed to be "perfect". This means that:

- the only way to decrypt an encrypted message is to know the corresponding key;
- an encrypted message does not reveal the key that was used to encrypt it;
- there is sufficient redundancy in messages so that the decryption algorithm can detect whether a ciphertext was encrypted with the expected key.

Although such assumptions are obviously not completely true for real cryptosystems, they represent the properties of an ideal cryptosystem, so they are useful to isolate the flaws of the protocol itself. In other words, any flaw found with this model is a real protocol flaw, but it is possible that the model does not reveal other weaknesses due to the used cryptosystems.

In order to model a mobile agent system, we use a technique quite similar to the one just described, based on the same assumptions about perfect cryptography and intruders. Agents are not modeled as autonomous mobile principals, but the whole agent-based system is represented at a lower level of abstraction, closer to the real system. Principals represent hosts which, by their agent execution platform, can execute mobile agent code. The migration of an agent from host to host is represented by a message, exchanged by the principals that represent the involved hosts, containing the agent code and data.

Since the integrity of the static agent code and the static agent data is a problem shared by all mobile agents, it can be solved by the MA platform, independently on the particular functionality of the agent. Since we are not interested in validating this part of the protocol, we assume that this problem is already solved in a reliable way and we do not model code transmission explicitly, but we simply assume that trusted hosts always execute the right code. So each agent hop is represented by a message containing only the dynamic part of the agent data. For example, in modeling the sample protocol described in [2], messages would just contain the collected data followed by the MIC and C values.

The main new aspect in mobile agent cryptographic protocols with respect to classical authentication protocols is the possibility of having attacks due both to network intruders and to untrusted hosts that may alter the behavior of agents hosted by their execution platform in an unpredictable way.

Modeling agents by messages exchanged by the hosts helps us in taking all such issues into account. Let us assume that we have a single untrusted host H_u. Attacks due to H_u can be taken into account in the above model giving the intruder the possibility of totally replacing it. To obtain this, it is enough to give the intruder all the secrets known by H_u, which, in our example, coincide with H_u's private key. Having the private key of H_u, the intruder can totally replace H_u, i.e. it can intercept any message directed to H_u, decrypt it exactly as H_u could do and forge any message H_u could produce in response to it. In other words, the intruder incorporates the behaviors of all possible network intruders as well as those of all possible untrusted hosts. This models any kind of malicious behavior of H_u, including any modification in the execution of the mobile agent on H_u, as well as the case in which the agent is sent by H_u to a host different from the one where it should go.

This approach can be easily extended to model environments with several untrusted hosts: inserting all their private keys in the initial intruder knowledge. This corresponds to modeling several untrusted hosts that can cooperate. The intruder process knows the keys of all the untrusted hosts and so can replace each of them and use the total knowledge of all of them.

Modeling untrusted hosts that can cooperate is adequate for most applications. Nonetheless, a solution can be found even for cases in which it could be useful to model untrusted hosts that are unable to cooperate. These cases are difficult to model using the modeling approach explained above, because the specifier cannot control how the intruder is modeled, the only thing that can be specified about the intruder being its initial knowledge. However, the model with cooperating untrusted hosts includes, as a special case, the one with uncooperating hosts. Indeed, by analyzing the first model, we can find out all possible attacks against the protocol, including those that do not require any untrusted host cooperation. So a way out of this problem is to analyze all the attacks reported by the analysis tool and then filter out the ones involving cooperating untrusted host.

For what concerns the specification of data integrity properties, they can be expressed in the same way authenticity properties are specified in classical authentication protocols. Indeed, requiring that some data that is considered valid at a given site has actually been delivered by the expected one is really an authenticity property.

4. A spi calculus model of the sample protocol

4.1. Introduction to Spi

Spi-calculus [1] is an extension of π-calculus [5], which adds a few cryptographic primitives to π-calculus, namely hashing and shared, private and public key encryption. Like π-calculus, it is a first order process algebra, so it does not address mobility explicitly. Its syntax and semantics can be found in [1]. We give here only a brief overview of the constructs used in this paper.

In π-calculus, a system is described as a set of concurrent communicating processes. The basic computational step and synchronization mechanism is interaction, in which a term M is communicated from an output process to an input process via a named channel m. A term can be basically a name (representing a data or a channel), a variable or a pair (N, L), where N and L are in turn terms. An output process $\overline{m}(M).P$ is ready to output M on channel m. If an interaction occurs, term M is communicated on m and then process P runs. An input process $m(x).Q$ is ready to input on channel m. If an interaction occurs in which term M is communicated on m, then process $Q[M/x]$ runs, where the post-fix operator $[M/x]$ indicates literal substitution of variable x by term M.

A composition $P|Q$ behaves as processes P and Q running in parallel. Each of them can interact with the other one on channels known to both, or with the outside world independently of the other one.

A restriction $(\nu n)\ P$ is a process that makes a new, private name n and then behaves as P. This operator is used to specify data that is not known to the intruder.

A pair splitting process $let\ (x, y) = M\ in\ P$ behaves as $P[N/x][L/y]$ if term M is the pair (N, L). Otherwise the process is stuck.

New kinds of terms have been introduced in spi-calculus to represent cryptographic operations. Term $\{[M]\}_{H^+}$ is the encryption of term M with the public key of H, denoted H^+. A term of this form can be decrypted by using the private key of H, H^-, by the process $case\ y\ of\{[x]\}_{H^-}\ in\ P$, which behaves as $P[M/x]$ if y is $\{[M]\}_{H^+}$ and is stuck otherwise.

The description of spi calculus in [1] introduces testing equivalence ($\simeq$), which is a kind of observational equivalence that can be used to express security properties. If a process P is ready to synchronize immediately on channel β, we say that P exhibits barb β, and we write $P \downarrow \beta$. If P can exhibit β, immediately or after a few internal reactions, we write $P \Downarrow \beta$. A test is a pair (R, β), where R is a testing process and β is a barb. A test (R, β) is passed by process P if $(P|R) \Downarrow \beta$. Testing equivalence $\simeq$ is formally defined by:

$$P \simeq Q \ \Leftrightarrow\ P \sqsubseteq Q\ and\ Q \sqsubseteq P$$

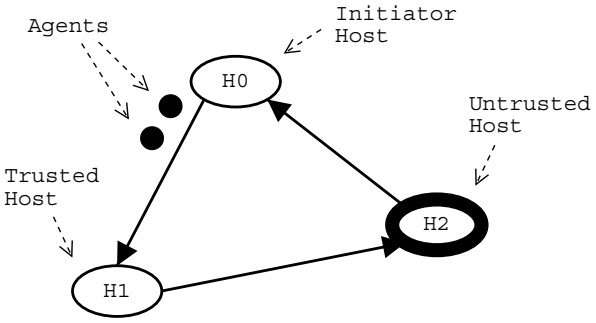

Figure 1. Modeled instance of the Corradi et al.'s protocol.

where the preorder $\sqsubseteq$ is defined as

$$P \sqsubseteq Q \ \Leftrightarrow \forall (R, \beta)\ (P|R) \Downarrow \beta \Rightarrow (Q|R) \Downarrow \beta$$

In other words, $P \sqsubseteq Q$ iff all the tests passed by P are passed also by Q. As a consequence, two processes are testing equivalent iff they pass the same tests, i.e. they are indistinguishable by testing.

4.2. The sample protocol model

In this section we show the modeling in spi of an instance of the protocol described in section 2. For simplicity, we deal with a fairly small instance of the protocol. In this instance (Fig. 1), an agent is sent out from the initiator H_0, visits two hosts, identified as H_1 and H_2, and comes back to H_0. D_1 and D_2 are the data gathered by the agent on H_1 and H_2 respectively. Hosts H_0 and H_1 are trusted, whereas H_2 is not. In this way we can express the integrity property asserting that the data collected at H_1 should not be modified when the agent passes through H_2. Of course, it is possible to extend the model to incorporate an arbitrary number of visited hosts. Moreover, we can have multiple sessions of the protocols, each one corresponding to a different agent.

Initially the agent moves from H_0 to H_1 on channel c_{H_1}. Since, at this time, the first value of the cryptographic counter (C_1) is the only data contained in the agent, the movement of the agent from H_0 to H_1 is simply modeled by H_0 sending C_1 encrypted with ${H_1}^-$to H_1. Then H_1 passes the agent to H_2 on channel c_{H_2}. This time the agent data is D_1, the value of MIC is $hash(D_1, C_1)$ and the value of the cryptographic counter is $hash(C_1)$. The complete description of the behavior of H_1 is then

$$H_1(D_1) =$$
$$c_{H_1}(y).$$
$$case\ y\ of\{[C_1]\}_{H_1^-}\ in$$

$$\overline{c_{H_2}}(D_1, hash(D_1, C_1), \{[hash(C_1)]\}_{H_2^+})$$

The encrypted cryptographic counter is received at H_1 in variable y, ($c_{H_1}(y)$), it is decrypted with H_1's private key to obtain C_1 ($case\ y\ of\{[C_1]\}_{H_1^-}$), which is then used by the agent to build the new AD, MIC and C components.

H_2 receives the agent and executes the agent code as described in section 2. Then H_2 sends the agent back to H_0 on channel c_{H_0}:

$$\begin{aligned}
&H_2(D_2) = \\
&c_{H_2}(x). \\
&let(D_1, MIC_1, y) = x\ in \\
&case\ y\ of\{[C_2]\}_{H_2^-}\ in \\
&\overline{c_{H_0}}(D_1, D_2, hash(D_2, C_2, MIC_1), \{[hash(C_2)]\}_{H_0^+})
\end{aligned}$$

When H_0 receives the agent, it checks that the value of MIC_2 corresponds to the initial secret and collected data. So the complete description of H_0 is:

$$\begin{aligned}
&H_0() = \\
&(\nu\ C_0) \\
&\overline{c_{H_1}}(\{[hash(C_0)]\}_{H_1^+}). \\
&c_{H_0}(x).let(x_1, D_2, MIC_2, y) = x\ in \\
&case\ y\ of\{[C_3]\}_{H_0^-}in \\
&case\ C_3\ of\ hash^3(C_0)in \\
&case\ MIC_2\ of \\
&\ hash(D_2, hash^2(C_0), hash(x_1, hash(C_0))))in \\
&\overline{c_{H_0}}(x_1)
\end{aligned} \tag{1}$$

The notation $hash^n(x)$ is used to denote that $hash$ function is applied n times on x. It is to be noted that if the computed MIC_2 matches, H_0 tries to send D_1 on channel c_{H_0}. This last step has been added for testing the protocol (see testing equivalence, section 4.1).

The whole protocol instance is:

$$\begin{aligned}
&P(D_1, D_2) = \\
&(\nu\ H_0{}^-)H_0()|(\nu\ H_1{}^-)H_1(D_1)|H_2(D_2)
\end{aligned} \tag{2}$$

where $H_2{}^-$ is not private, since H_2 is an untrusted host.

4.3. Security Properties

We are interested in agent integrity, i.e. we want any tampering of D_1 by H_2 or by network intruders to be detected. Therefore, let us write the following specification of H_0:

$$\begin{aligned}
&H_{0Spec}(D_1) = \\
&(\nu\ C_0) \\
&\overline{c_{H_0H_1}}(\{[hash(C_0)]\}_{H_1{}^+}). \\
&c_{H_0}(x).let(x_1, D_2, MIC_2, y) = x\ in \\
&case\ y\ of\{[C_3]\}_{H_0{}^-}in \\
&case\ C_3\ of\ hash^3(C_0)in \\
&case\ MIC_2\ of \\
&\ hash(D_2, hash^2(C_0), hash(x_1, hash(C_0))))in \\
&\overline{c_{H_0}}(D_1)
\end{aligned} \tag{3}$$

and the following whole protocol specification:

$$\begin{aligned}
&P_{Spec}(D_1, D_2) = \\
&(\nu\ H_0{}^-)H_{0Spec}(D_1)|(\nu\ H_1{}^-)H_1(D_1)|H_2(D_2)
\end{aligned} \tag{4}$$

As the reader can note, equation (4) differs from equation (2) only in the last line. The process $H_{0Spec}(D_1)$ is a "magical" process that is ready to output the right value of D_1 on c_{H_0} for any incoming agent that passes the validity tests. If we prove that specification (5) is testing equivalent to model (3), we have proved that the protocol ensures integrity of the agent data. Indeed, in that case, we are sure that the value of D_1 that H_0 has received has not been tampered with. Otherwise, the specification and the model would be willing to output different values through channel c_{H_0}, and they would not be testing equivalent.

So, the formal expression of integrity for the above agent is :

$$P_{Spec}(D_1, D_2) \simeq P(D_1, D_2) \tag{5}$$

Equation (5) is only an integrity specification. To specify additional properties other equations are needed.

5. Conclusions

In this paper we explore how some formal specification techniques, originally developed to model various security properties of cryptographic protocols, can be applied to model integrity properties of mobile agents as well.

Mobile agent systems can be modeled by means of the messages exchanged by the hosts where they are executed, in a way quite similar to the one used for authentication protocols. In this way, untrustedness of hosts can be modeled simply making their secrets known to the intruder.

A clear formal specification is very important to unambiguously and precisely describe a mobile agent security mechanism and its properties. It expresses not only the contents of the exchanged messages, but all the aspects that are relevant to guarantee the security properties of interest, including the checks that must be done when messages are received. So a formal specification of this kind is a valid and practical basis for a correct implementation.

References

[1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. Research Report 149, Digital Equipment Corporation Systems Research Center, Jan. 1998. A shortened version of this report appeared in Information and Computation 148(1999):1-70.

[2] A. Corradi, R. Montanari, and C. Stefanelli. Mobile agents integrity in E-commerce applications. In *of the 19th IEEE International Conference on Distributed Computing Systems Workshop (ICDCS'99)*, pages 59 – 64, Austin, Texas, May 31 – June 5 1999. IEEE Computer Society Press.

[3] W. Farmer, J. Guttman, and V. Swarup. Security for mobile agents: Issues and requirements. In *Proceedings of the 19th National Information Systems Security Conference*, pages 591–597, Baltimore, Md., October 1996.

[4] G. Karjoth, N. Asokan, and C. Gülcü. Protecting the Computation Results of Free-Roaming Agents. In K. Rothermel and F. Hohl, editors, *Proceedings of the 2nd International Workshop on Mobile Agents*, volume 1477 of *Lecture Notes in Computer Science*, pages 195–207. Springer-Verlag: Heidelberg, Germany, 1998.

[5] R. Milner, J. Parrow, and D. Walker. A calculus for mobile processes. Part I and II. *Information and Computation*, 100(1), 1992.

[6] C. Tschudin. Mobile agent security. In M. Klusch, editor, *Intelligent Information Agents: Cooperative, Rational and Adaptive Information Gathering on the Internet*, Lecture Notes in Computer Science, pages 431–446. Springer-Verlag, Berlin, Germany, 1999.

[7] G. Vigna. Cryptographic Traces for Mobile Agents. In G. Vigna, editor, *Mobile Agent Security*, Lecture Notes in Computer Science No. 1419, pages 137–153. Springer-Verlag: Heidelberg, Germany, 1998.

[8] X. F. Wang, X. Yi, K. Y. Lam, and E. Okamoto. Secure information gathering agent for internet trading. In C. Zhang and D. Lukose, editors, *Proceedings of the 4th Australian Workshop on Distributed Artificial Intelligence on Multi-Agent Systems : Theories, Languages, and Applications (DAI-98)*, volume 1544 of *LNAI*, pages 183–193, Berlin, Germany, July 1998. Springer.

[9] B. S. Yee. A sanctuary for mobile agents. Technical Report CS97-537, UC San Diego, Department of Computer Science and Engineering, Apr. 1997.