An Authentication Protocol For Mobile Agents Using Bilinear Pairings

Amitabh Saxena, and Ben Soh
Dept. of Computer Science and Computer Engineering
La Trobe University, Bundoora, VIC, Australia 3086

August 16, 2005

Abstract

A mobile agent is a mobile program capable of maintaining its execution states as it migrates between different execution platforms. A key security problem in the mobile agent paradigm is that of trust: How to ensure that the past itinerary (of execution platforms) claimed by the agent is correct. This is necessary in order to establish a reasonable level of trust for the agent before granting execution privileges.

In this paper we describe a protocol using bilinear pairings that enables trust relationships to be formed between agent platforms in an ad-hoc manner without actively involving any trusted third party. This protocol can be used to authenticate agents before granting execution privileges. The main idea behind our approach is the concept of 'one-way' chaining.

1 Introduction

Mobile agents are agents that can physically travel across networks and perform tasks on machines that provide agent hosting capability. This allows processes to migrate from computer to computer, for processes to split into multiple instances that execute on different machines, and to return to their point of origin. A detailed discussion of mobile agents is beyond the scope of this paper and the reader is referred to [1]. Two foremost security challenges for mobile agents are (a) host protection and (b) agent protection. Our work on mobile agents is only focused only on host protection. For work on agent protection the reader is referred to [2, 3, 4, 5].

In contrast to approaches for host protection based on sandbox environments or other forms of code validation, our model aims to validate the itinerary of an agent. Our approach to security is based on a notion of trust which is summarized as follows: If all entities involved with the agent can be authenticated, a level of trust can be established, which can then be used for granting or denying execution privileges. Current solutions for host protection rely on tamper proof hardware, an on line trusted third party or a 'sandbox' model of execution [6, 7, 8]. Our method does not require any such measures. We use the concept of *one-way* signatures to connect arbitrary hosts in a chain of trust, thus enabling ad-hoc trust relationships to be formed.

The concept of one-way signature chaining was proposed in [9] and [10] where the authors constructed authentication protocols for mobile agents using hypothetical cryptographic primitives known as *strong non-commutative associative one-way functions*. The authors also asked if an equivalent protocol can be constructed using any existing cryptographic primitives. In this paper, we answer this question affirmatively and show that the mobile agent authentication protocol presented in [10] can be constructed using bilinear pairings, thus settling their open question.

Although the original concept of signature chaining presented in [10] is based on a standard certificate based Public Key Infrastructure (PKI), it can be shown that their model can be reduced directly to an Identity-Based Public Key Cryptosystem (ID-PKC) or a Certificate-Less Public Key Cryptosystem (CL-PKC) due to certain properties of the one-way function used.[1] In contrast to this, the protocol presented in this paper is based on a standard certificate based PKI and it is not known if a direct reduction to an ID-PKC or a CL-PKC exists.

2 Background

Any entity that runs a mobile agent platform server is called a *host*. We assume that all such hosts are identified by a public directory. Any host that initiated an agent into the system is called the *initiator* of the agent. Agents can migrate autonomously between different host platforms. This act of migration is called *agent transfer*. We assume that agent transfer is done over a secure channel using a standard agent transfer protocol. An *instance* of an agent is a snapshot of its state at any point of execution on some platform. An *itinerary* is the ordered list of hosts already visited by an agent.

2.1 Agent partitioning

Using the object oriented paradigm, we assume that any instance of a mobile agent can be split (or partitioned) into a *static* part (consisting of object methods) which is unchanging as the agent hops across platforms and a *dynamic* part (consisting of data and the state information of the interacting objects) that changes at each hop. Depending on the specific implementation, the partitioning schemes can differ. However, in this section we enumerate certain properties relevant in our context.

1. *Unique*: It may be possible that an instance of the agent can be partitioned in more than one ways. A partition scheme is *unique* if all instances of the agent have a unique static and dynamic part.

[1]The reader is referred to [11] for a discussion of an ID-PKC and to [12] for a discussion of a CL-PKC

2. *Identical*: A partition scheme is *identical* if all instances of the agent have at least one common static part.

3. *Mutually authenticating*: We further assume that some static and dynamic parts can be made mutually inseparable. This means that the agent's functionality is available if and only if both the static and dynamic parts correspond to the same agent. Mixing and matching between different agents is not possible. We say that the scheme is *mutually authenticating* if all instances of the agent have at least one mutually inseparable partition.

4. *Ideal*: A partitioning scheme is *ideal* if it is unique, identical and mutually authenticating.

2.2 Fixed Strings

Let L_1 and L_2 be any two languages. For some $x \in L_1$ and some $y \in L_2$, the ordered pair (x, y) is said to be *fixed* if and only if there exists a (polynomial-time computable) binary function $\sigma : L_1 \times L_2 \mapsto \{0, 1\}$ such that $\sigma(x, y) = 1$ and it is computationally intractable to find another string $\hat{y} \in L_2$ such that $\sigma(x, \hat{y}) = 1$.

2.3 Authentication Requirements

In this section, we give the high-level authentication requirements for our model. we define the following two requirements:

1. *Initiator authentication*: Is the claimed initiator the same as the real initiator?

2. *Itinerary authentication*: Is the claimed itinerary the same as the real itinerary?

Our requirement for unconditional security is itinerary authentication. It is evident, however, that this will also always involve initiator authentication, since the initiator is the first host in the itinerary. We introduce the concept of *relative authentication* to imply that the first host (the initiator) in an itinerary is unknown. On the other hand, *absolute authentication* implies that the initiator can be authenticated.

2.4 One-way Chaining

Represent the host platforms as points of a acyclic directed graph. As the agent hops, a new arc directed from the receiver to the sender is added to the graph. The edges of such a graph will represent a hop-by-hop path of the agent in the reverse direction from the current host to the initiator. In this notation the statements "a passed the agent to b" and "There is a path of unit length from

b to a" are considered equivalent. We can consider this graph to describe the path by which trust is propagated in the system.[2]

1. We say that a *direct* path exists from b to a if and only if b can prove (in the context of the agent) something about a that no other host can. That is, b has some *extra* information about a that others cannot extract from b's proof.

2. Let $\{h_0, h_1, \dots h_n\}$ be a set of hosts for some $n \geq 1$. We say a *chained* path exists from h_n to h_0 if and only if there exists a direct path from h_x to h_{x-1} for each x from 1 to n.

3. We say that there is a *one-way* chained path from b to a if and only if there is a chained path from b to a and there is no (direct or chained) path from a to any other host.

Assume that i is the initiator of the agent, a is any sending host and b is the receiving host. Also, excepting the act of agent transfer no other interaction is allowed between any hosts. Using this scenario, authentication can be redefined in the context of b as follows:

(a) *Relative*: Determine that a chained path from a to i exists.

(b) *Absolute*: Determine that a one-way chained path from a to i exists.

3 Our Authentication Protocol

Our implementation of the protocol is based on bilinear pairings. Bilinear pairings were first used in cryptography by Boneh and Franklin in [11] where they presented the first short and secure IBE scheme. Although bilinear pairings are mostly known for their use in identity based cryptography, other non-identity based applications also have been proposed using bilinear pairings [13, 14, 15]. Our authentication protocol presented here is based on an ordinary certificate based PKI.

3.1 Bilinear Pairings

Let G_1 be a cyclic additive group generated by P, whose order is a prime q and G_2 be a cyclic multiplicative group of the same order. Assume that computing the discrete logarithm in both G_1 and G_2 is hard. A (non-invertible) bilinear pairing is a map $e : G_1 \times G_1 \mapsto G_2$ and satisfies the following properties:

1. *Bilinearity*: $e(aP, bQ) = e(P, Q)^{ab}$ For all $P, Q \in G_1$ and $a, b \in \mathbb{Z}_q$.

2. *Non-degeneracy*: $P \neq 0 \Rightarrow e(P, P) \neq 1$.

[2]We intuitively define trust to propagate in the reverse direction of the agent. If the agent moves from a to b, we are interested to know if b trusts a. That is, if there is path from b to a. Moreover we are only interested in those hosts that modified the dynamic part.

3. *Computability*: e is efficiently computable.

Typically, the map e will be derived from either the Weil or Tate pairing on an elliptic curve over a finite field. Despite the fairly complex mathematics involved in constructing such maps, cryptographic protocols based on pairings can be described entirely without ever referring to the actual implementation. Pairings and other parameters should be selected in proactive for efficiency and security.

3.2 Initial Setup

We assume that the mobile agent can be partitioned using an ideal scheme (see section 2.3). Represent by M, the static part and by D_i the dynamic part of the i_{th} instance. For any agent $\{M, D_i\}$, the sending, platform is I_i and the receiving platform is I_{i+1}. The initiator of the agent is I_0.

Let e be a bilinear map as defined in section 3.1. Define a cryptographic hash functions $H : \{0,1\}^* \mapsto G_1$. Let P be a generator of G_1. All these parameters are generated by a central controller. To participate in this protocol, each user must have a certified public key. We consider the process of certification outside the scope of our protocol:

> Each participant I_i generates a random $x_i \in \mathbb{Z}_q$ as the private key. The corresponding public key is $Y_i = x_i P$

3.3 Agent Initiation

To enable absolute authentication, we require that the pair (M, I_0) be fixed. A possible approach for this is to involve a Trusted Third Party (TTP) to certify the pairs. The TTP ensures that the same pair cannot be reused again for a certain period of time. We note, however, that it may also be possible to fix the pair (M, I_0) (without involving a TTP) using the methods for code obfuscation, undetachable signatures and watermarking described in [2, 4, 5, 16, 17, 18, 19, 9]. For simplicity, in this paper, we assume that I_0 uses a TTP is used to fix the pair (M, I_0). It asks the TTP to certify the ordered pair (M, I_0) using a standard signature scheme (like RSA).Denote by C, the certificate from the TTP. To avoid chosen cipher text attacks, a time stamp is included in the certificate. Users who created their public keys after this time are precluded from participating in this protocol.

3.4 Transfer Protocol

An arbitrary participant I_i will process the agent as follows: On receiving it from I_{i-1}, it first follows the verification procedure. Before passing the agent (after execution) to I_{i+1}, it follows the signing procedure. I_0, however, only follows the signing procedure. The certificate C must always accompany the message.

The following additional definitions will be useful:

1. Define $U_0 = x_0 H(M)$ and $U_i = x_i H(M) + U_{i-1}$ for $i > 0$
   Thus $U_i = (x_1 + x_2 + \ldots x_i)H(M)$
2. Define $V_0 = Y_0$ and $V_i = Y_i + V_{i-1}$ for $i > 0$
3. Define $W_i = x_i H(\text{“}I_{i-1}, I_i\text{”})$
4. Define $Z_i = x_i H(D_i)$.

Signing

The signature of I_i on the agent $\{M, D_i\}$ is the set: $\{C, U_i, \{W_1, W_2, \ldots W_i\}, Z_i\}$. The list of participants, “$I_0, I_1, \ldots I_i$” is also assumed to part of the signature.

Verification

For clarity, we describe the verification procedure to be followed by I_{i+1}. It consists of five tests and the message is rejected if any of them fail:

1. Verify certificate C and confirm first participant I_0.
2. Verify $e(U_i, P) \overset{?}{=} e(V_i, H(M))$.
3. Verify $e(W_j, P) \overset{?}{=} e(H(\text{“}I_{j-1}, I_j\text{”}), Y_j)$ for all j where $(1 \leq j \leq i)$.
4. Verify $e(Z_i, P) \overset{?}{=} e(H(D_i), Y_i)$.
5. Verify that M and D_i belong to the same agent (via the mutually authenticating property).

If all above succeed, accept the itinerary claimed by agent as valid. As a note, we would like to mention that the signatures verified in steps 3 and 4 of the verification process can be constructed using any ordinary signature scheme (like RSA). However it is necessary that the signatures of step 2 be based on pairings.

3.5 Overview of the protocol

The above protocol is an example of a one-way signature chaining scheme. To understand this, see that steps 2 and 3 of the verification process involve the public keys of all participating users (in the right order). Moreover, since M and I_0 cannot be un-linked due to the certificate C, it is ensured that a different initial user cannot be used for M.

We see that the signatures have an “additive” property, demonstrated by the fact that I_{i+1} can ‘add’ more information to the signature U_i of I_i by computing U_{i+1}. Note that computing any U_i just from U_{i+1} is considered infeasible due to the assumed properties of the bilinear map.[3]. User I_{i+1} sends U_{i+1} as the new proof while it keeps U_i, the old proof as its secret evidence in case of a dispute.

Assuming that all users are unique, a few points about this protocol are noteworthy :

1. Each I_i who passes the message must include its name in the signature and in the right sequence for validation to succeed.

2. Users cannot remove names of other users from the list in the signature without knowledge of their private keys, nor can they change the order or add new names.

3. Authentication is relative to I_0 who in turn authenticates with the TTP. If, however, it is possible to establish the originator of a message directly from its contents or by some other means, the TTP can be eliminated. For a discussion on this see [9].

4. The signing and verification procedures are completely non-interactive.

5. The dynamic part is only authenticated to the previous hop. The itinerary authentication is done entirely using the static part.

It is easily seen that the signing time is independent of the number of users. However, the signature length and the verification time increase linearly with the number of users in the list. This is not a problem unless the list becomes very large.

4 Security Analysis

In this section, we outline a rough security analysis of our protocol. We consider an attack to be successful if the ordered list of names in the signature contains false information and the verification procedure accepts. Assuming that I_i is the attacker, a combination of the following attacks are possible:

1. It does not include its name in the list.

2. It adds one or more names to the list.

3. It deletes one or more names from the list or changes the order of names.

We will consider each scenario seperately. We note that a detailed security analysis of the protocol is out of the scope of this paper but we also note that the simplicity of the protocol does not demand such analysis.

1. The first possibility is ruled out since otherwise steps 3 and 4 of the verification process will fail.

[3]Observe that U_i cannot be computed from U_{i+1} without knowledge of x_i but knowledge of U_i does not reveal x_i.

2. Arbitrary names cannot be added to the list because I_i cannot compute signatures M_i on behalf of other users. Thus, if a false user is added to the list, step 3 of the verification process will fail.

3. Finally deleting names or changing order is not possible either. If the order of participants is changed, the verification process in step 3 will fail with a very high probability. To see this, we enumerate the following strong security characterstics of our scheme:

   (a) Signature Unforgeability: It is not possible for any participant to generate signatures for other participants without knowledge of their private keys assuming the hardness of the Bilinear Diffie-Hellman problem (BDH). Similarly computing any private keys from the public information is will be equivalent to solving the Discrete Logarithm (DL) problem in G_1 (and consequently G_2).

   (b) Chained Signature Unforgeability: Similarly it is hard to add arbitrary participants in the chained signatures without knowledge of their public key due to the difficulty of the DL problem.

5 Conclusion

In this paper, we used the concept of signature chaining to propose an authentication protocol for mobile agents based on bilinear pairings. Our method is based on the notion of additive zero knowledge [9] which enables trust to propagate between different provers. We demonstrated that signature chaining can be used to form ad-hoc trust relationships between multiple participants in a dynamic and non-interactive manner. Our protocol can be used to authenticate the itinerary of mobile agents without any active involvement of a Trusted Third Party(TTP). We also note that it may be possible to completely eliminate the TTP using methods of code obfuscation, watermarking and undetachable signatures. We note that the size of signatures and the verification time increase linearly with the number of users. A further improvement would be to try to find schemes where the verification time and signature size is constant. Finally, it is worth researching if a certificate-less or an identity based scheme can be derived from the certificate based one presented in this paper.

References

[1] David Kotz and Robert S. Gray. Mobile agents and the future of the internet. *SIGOPS Oper. Syst. Rev.*, 33(3):7–13, 1999.

[2] Tomas Sander and Christian F. Tschudin. Protecting mobile agents against malicious hosts. *Lecture Notes in Computer Science*, 1419:44–60, 1998.

[3] Joy Algesheimer, Christian Cachin, Jan Camenisch, and Günter Karjoth. Cryptographic security for mobile code. In *SP '01: Proceedings of the IEEE Symposium on Security and Privacy*, pages 2–11. IEEE Computer Society, 2001.

[4] Panayiotis Kotzanikolaou, Mike Burmester, and Vassilios Chrissikopoulos. Secure transactions with mobile agents in hostile environments. In *Australasian Conference on Information Security and Privacy*, pages 289–297, 2000.

[5] Joris Claessens, Bart Preneel, and Joos Vandewalle. (how) can mobile agents do secure electronic transactions on untrusted hosts? a survey of the security issues and the current solutions. *ACM Trans. Inter. Tech.*, 3(1):28–48, 2003.

[6] Bennet S. Yee. A sanctuary for mobile agents. In *Secure Internet Programming*, pages 261–273, 1999.

[7] U. G. Wilhelm, S. Staamann, and L. Buttyán. Introducing trusted third parties to the mobile agent paradigm. In J. Vitek and C. Jensen, editors, *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, volume 1603, pages 471–491. Springer-Verlag, New York, NY, USA, 1999.

[8] G. Karjoth, D.B. Lange, and M. Oshima. A security model for aglets. *IEEE Internet Computing*, 1(4):68–77, 1997.

[9] Amitabh Saxena and Ben Soh. Authenticating mobile agent platforms using signature chaining without trusted third parties. In *Proceedings of The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-05)*, pages 282–285, 2005.

[10] Amitabh Saxena and Ben Soh. A novel method for authenticating mobile agents with one-way signature chaining. In *Proceedings of The 7th International Symposium on Autonomous Decentralized Systems (ISADS 05)*, pages 187–193, 2005.

[11] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. pages 213–229, 2001.

[12] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. Cryptology ePrint Archive, Report 2003/126, 2003.

[13] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532, London, UK, 2001. Springer-Verlag.

[14] Z. Cheng, L. Vasiu, and R. Comley. Pairing-based one-round tripartite key agreement protocols, 2004.

[15] Ratna Dutta, Rana Barua, and Palash Sarkar. Pairing-based cryptographic protocols : A survey. Cryptology ePrint Archive, Report 2004/064, 2004.

[16] Christian S. Collberg and Clark Thomborson. Watermarking, tamper-proofing, and obfuscation - tools for software protection. In *IEEE Transactions on Software Engineering*, volume 28, pages 735–746, August 2002.

[17] Chenxi Wang, Jonathan Hill, John Knight, and Jack Davidson. Software tamper resistance: Obstructing static analysis of programs. Technical report, University of Virginia, 2000.

[18] Julien P. Stern, Gael Hachez, Francois Koeune, and Jean-Jacques Quisquater. Robust object watermarking: Application to code. In *Information Hiding*, pages 368–378, 1999.

[19] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. Cryptology ePrint Archive, Report 2001/069, 2001.