

SECURE DATA COLLECTION WITH UPDATES

SERGIO LOUREIRO, REFIK MOLVA and ALAIN PANNETRAT
*Institut Eurecom, 2229 route des cretes, B.P. 193
06904 Sophia-Antipolis Cedex, France
E-mail: {loureiro, molva, pannetra}@eurecom.fr*

This paper describes a protocol to protect data collected by mobile agents roaming through a set of potentially malicious hosts. This protocol is based on an original secure cryptographic technique that assures the integrity of a sequence of data segments regardless of the order of each segment in the sequence. The protocol allows each host to update the data it previously submitted in a way that is suitable for free competition scenarios like comparison shopping or distributed auction and for highly dynamic environments like stock markets. The set of hosts can be visited several times in random order and a short message digest allows for the integrity verification of all the collected data.

1 Introduction

Recent advances in software technology allow the design of mobile code that is capable of moving over the network and running independently at remote hosts. Best known examples of mobile code are Java applets, ActiveX programs, and software agents. The mobile code paradigm offers several advantages over the more traditional distributed computing approaches: flexibility in software design beyond the well established object oriented paradigm, autonomy of components, and bandwidth optimization, just to name a few of them. As usual, advantages come with a cost that is increased vulnerability in the face of malicious intrusion scenarios akin to Internet. Possible vulnerabilities with mobile code fall in one of two categories:

- Attacks performed by the mobile program against the remote execution environment and its resources like in the now common example of malicious Java applets.
- Subversion of the mobile code and unauthorized modification of its data by the remote execution environment.

This paper's focus is the second problem, i. e. the protection of mobile code from potentially malicious hosts, which has not yet attracted so much attention from software manufacturers. This category is also quite atypical since it does not rely on the security of the execution environment which has always been a basic assumption in classical reasoning about the security of cryptographic systems. Protecting mobile code from malicious remote execution environments can be rephrased as carrying out a trusted operation as defined by a mobile program using an execution environment on an untrusted host. Some authors [1] conjecture that mobile code cannot be effectively protected against a malicious execution environment that has full access to both its code and data segments. On the other hand, few researchers [6] [8] were able to show the possibility of obtaining a safe execution of secret functions on untrusted execution environments. The results obtained by the latter group, unfortunately, only apply to a limited set of functions. Solutions for securing mobile code against the execution environment are thus still in their infancy and no widespread practical implementations exist.

In this paper, we focus on a special case of mobile code security that is the protection of the data collected by mobile agents against potentially competing hosts. A typical illustration for such a scenario

is a software agent that bargains on behalf of a customer by visiting various merchant systems at different locations. Secure data collection schemes can also be used to carry out a distributed auction. Another example is the collection of data in stock markets, where the data is available at different locations. In these examples the emphasis of security is shifted from the mobile program to the data that is collected from the visited hosts. The collected data is subject to disclosure, modification, and repudiation by single hosts, a group of colluding hosts, or network intruders.

We suggest an original integrity scheme to protect the data submitted to the mobile agent by competing hosts visited by the agent. The integrity scheme ensures the protection of each piece of data against modification or tampering by parties other than the origin of the data (the host that submitted the data). The integrity of the entire set of data collected by the mobile agent is assured based on an original secure cryptographic mechanism using discrete exponentials. Thanks to the properties of our mechanism, the data collection protocol offers three advantages:

- Each host can update the data it previously submitted without additional complexity or memory increase.
- The integrity verification mechanism is independent of the sequence of individual data submissions by different hosts.
- The integrity verification is not computationally intensive and it does not depend on the number of updates. Moreover, the complexity of the verification does not increase significantly with the number of visited hosts.

The update facility is suitable for commercial competition as required during an auction and it allows for fast reaction in case of dynamic scenarios like stock exchange markets. The second property of the protocol allows for the collection of data from hosts without any constraint on the itinerary of the mobile agent.

The paper is organized as follows. Section 2 presents the data collection scenario and formalizes the security requirements. The generic cryptographic technique for data integrity is presented in section 3. The data collection protocol based on this technique is described in section 4. Section 5 is devoted to the evaluation of the security properties previously defined.

2 Data Collection System

The purpose of our data collection system is to allow mobile agents to travel among hosts of a network, to collect individual data segments from these hosts and to return the set of data segments to the originator of the agent. Each data segment collected by the agent can either be the result of some computation by the agent, based on some local input, or simply the input of some data by the visited host, without any processing by the agent.

Our security scheme assures the integrity of data segments against tampering and deletion attacks that might originate from a host visited by the agent, a set of colluding hosts or an intruder on the network. The security of the process used to generate the data segments at each host is out of the scope of our scheme, based on the assumption that, even though each host might behave maliciously against other hosts, each host can be trusted with respect to the generation of its own data.

Migration is another important aspect of the data collection scheme with respect to the security of the collected data. By controlling the migration process, malicious hosts can have a significant impact on the set of data segments collected by the agent. Our data integrity scheme does not address the security of the agent's itinerary.

In a typical data collection scenario, the mobile agent is generated by a source host H_0 and visits intermediate hosts H_i based on certain migration decisions, which are beyond the scope of this paper. Each intermediate host H_i submits a piece of data o_i . In addition, each host computes as part of the secure data collection scheme an integrity proof value O_i , that will be integrated on the overall integrity proof value $\Gamma(S)$, included in the agent and computed by the previous host H_{i-1} . The new integrity value $\Gamma(S \cup \{O_i\})$ and the set of data segments collected from the previously visited hosts $\Sigma = \{o_0, o_1, \dots, o_i\}$ are transmitted to the next host. The data collection scheme allows the agent to visit hosts that were already visited and lets these hosts update the data pieces they previously submitted. When the data collection process terminates (or the agent is called back), the agent returns to its originator H_0 . At this point the agent returns to its originator the set Σ of data segments collected from all the visited hosts and the final integrity proof value $\Gamma(S)$. The originator can then verify the integrity of the data segments in Σ using $\Gamma(S)$. Table 1 summarizes the components involved in the secure data collection process.

Table 1: Data Collection Components

H_0	originator
$H_i, 1 \leq i \leq n$	visited hosts
$o_i, 1 \leq i \leq n$	data collected from H_i , i. e., output of the data collection algorithm
$O_i, 1 \leq i \leq n$	integrity proof associated with o_i
Σ	set of data collected, i. e. $\{o_0, o_1, \dots, o_i\}$
S	set of integrity proofs, i. e. $\{O_0, O_1, \dots, O_i\}$
$\Gamma(S)$	integrity proof associated with all the elements of set S

2.1 Security Requirements

The data collection process is exposed to a number of attacks from network intruders and legitimate hosts behaving maliciously with respect to competing parties as depicted in [3]. These attacks raise a number of security requirements as follows:

- **Data Integrity:** o_i cannot be modified or updated by parties other than H_i .
- **Truncation Resilience:** only the data segments $o_j, i < j < k$, submitted between the first malicious host H_i and another malicious host H_k can be truncated from the set of data pieces.
- **Insertion Resilience:** no data segment can be inserted unless explicitly allowed.
- **Data Confidentiality:** o_i cannot be disclosed to parties other than H_i and H_0 .

- **Non-Repudiation of Origin:** H_i cannot deny having submitted o_i once it was actually included in the set of collected data.

Our definition of the data integrity requirement expands the previous definitions that can be found in [3] and [9] in that a host can update the data it previously submitted. We believe that the update facility is required in free competition and dynamic commercial environments, like stock markets.

Data confidentiality and non-repudiation are not mandatory in all scenarios. In some real life scenarios like auction, the confidentiality service might even be conflicting with the free competition model. Like the protocols proposed in [3], our data collection scheme can be enhanced with data confidentiality based on public key encryption and with non-repudiation of origin based on digital signatures.

A limitation of data collection schemes based on data integrity mechanisms is the degree of truncation resilience that can be offered. As pointed out in [9], the collusion between two malicious hosts allows the deletion of the data collected on the path between them by substituting the data set with a copy recorded prior to the beginning of the truncated path. This seems to be an inherent limitation regardless of the way the integrity function is computed in each scheme. A solution to this problem consists of assuring the integrity of the migration path in addition to the integrity of the collected data. This problem is beyond the scope of this paper and calls for solutions aiming at the integrity of execution for mobile code like [7] and [4], additional constraints on the data collection process like a fixed number of hosts to be visited [9], or a pre-defined migration path.

2.2 Related Work

Prior work addressing the integrity of collected data [3][9] uses a technique called hash chaining as the basic mechanism to achieve the integrity of the data pieces submitted by visited hosts. The result of the chained hash computations is the proof of integrity for all the collected data. This technique was first introduced by [10] to create secure audit logs due to its efficiency in terms of computational complexity and size of the integrity proof. In the case of audit logs, it is important to keep a tamper-proof record of all the operations performed in a resilient way.

The data collection scheme suggested by [9] and [3] differs from the solution presented in this paper regarding various aspects. One of the objectives of the data collection schemes presented in [9] and [3] is to prevent hosts from updating the data they previously submitted. These schemes therefore only allow closed bids and include data confidentiality and host anonymity as a basic requirement in order to assure fairness among competing hosts. Our solution in contrast addresses a dynamic scenario including several rounds of competing offers between bidders. As a result, unlike the solutions in [9] and [3], our scheme allows for multiple updates of each data piece by the submitting host. Moreover, data confidentiality and host anonymity are not mandatory requirements of the free competition scenario.

Furthermore, hash chaining as a basic data integrity mechanism does not meet the requirements of the our data collection scheme. Hash chaining is tied with an implicit sequence among the various data pieces that are protected and the knowledge of the sequence is mandatory for the verification process, i. e., the verification process has to compute the hash chain in the same order as the data collection process. Dynamic scenarios like auction whereby each data piece may be updated several times, hash chaining would require to keep track of all the past values for each data piece. Our scheme is thus based on a novel data integrity technique called set hashing that allows for the verification of the most recent value of each data piece without keeping track of past values. With set hashing, the computation of the

integrity check value and its verification can be performed in random order with respect to the data pieces.

Our scheme aims at a scenario that fosters competition by keeping the information from competing sources in cleartext and by authorizing frequent updates, whereas [3] and [9] assume a rigid scenario based on widespread confidentiality. While our scheme can be easily enhanced with classical confidentiality mechanisms, confidentiality can hardly be suppressed from the solutions in [3] and [9] because of the inherent dependency between confidentiality and integrity mechanisms on which relies the design of these solutions.

3 Set Hashing

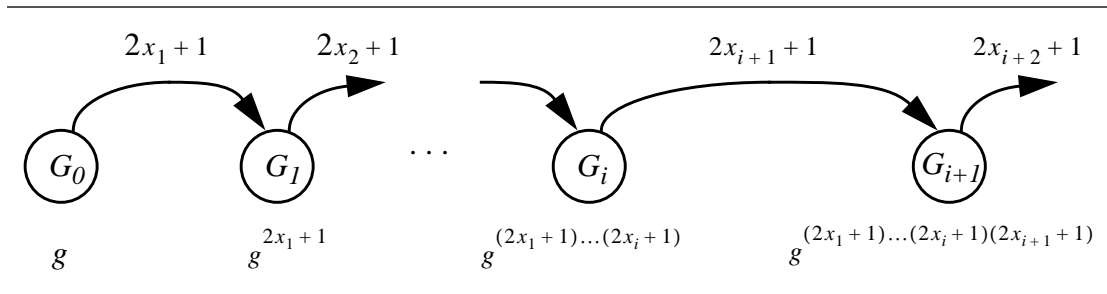
This section defines an original cryptographic mechanism at the core of the proposed data collection algorithm. We build a “set hash” using the difficulty of solving the discrete logarithm problem in a finite field, in combination with a classical cryptographic hash function. This mechanism provides a method to hash together a set of data blocks in an order-independent fashion.

3.1 Generator sequences

Let p be a large Sophie Germain prime¹, that is $p = 2q + 1$ where q is also prime. Define g as a generator of the cyclic group \mathbb{Z}_p^* . Then for all x in $\{1 \dots (q-3)/2\}$, $g' = g^{2x+1} \bmod p$ is also a generator of the cyclic group \mathbb{Z}_p^* . Hence the following sequence $(G_i)_{i \geq 0}$ is a sequence of generators in \mathbb{Z}_p^* :

- $G_0 = g$
- $G_{i+1} = (G_i)^{2x_{i+1}+1} \bmod p$

where $(x_i)_{i > 0}$ is a sequence in $\{1 \dots (q-3)/2\}$ [5].



In words, the advantage of this construction is that the result is always a generator, which has the properties referred in the next section.

1. Also called a “strong prime”.

3.2 Properties

- *Property 1.* (security) With the knowledge of any G_i and G_{i+1} , it is computationally infeasible to compute x_{i+1} . Solving for x_{i+1} would require an adversary to solve a discrete logarithm to the base G_i in \mathbb{Z}_p^* [2].

- *Property 2.* (commutativity) If $G_{i>0}$ is defined by the sequence $(x_j)_{0<j\leq i}$ then for any $G'_{i>0}$ defined from a permutation of $(x_j)_{0<j\leq i}$ we have $G_i = G'_i$. This second property is based on the commutativity of the exponent field \mathbb{Z}_{p-1}^* .

- *Property 3.* (cancellation) With the knowledge of G_{i+1} and x_{i+1} it is possible to compute G_i as:

$$G_i = (G_{i+1})^{\frac{1}{2x_{i+1}+1}} \bmod p$$

- *Property 4.* (computational complexity) With the knowledge of the set $(x_i)_{0<i\leq n}$, the computation of G_i requires $2n$ multiplications, n additions and only 1 exponentiation since:

$$G_i = g^{(2x_1+1)(2x_2+1)\dots(2x_n+1)} \bmod p$$

3.3 Integrity Function

The combination of *property 2* and *3* allows us to work on sets instead of sequences, thus we define an integrity function $\Gamma: \left\{ \mathbb{Z}_{(q-3)/2}^+ \right\} \rightarrow \mathbb{Z}_p^*$ which takes an unordered set of elements in $\mathbb{Z}_{(q-3)/2}^+$ and produces an element in \mathbb{Z}_p^* :

- $\Gamma(\emptyset) = g$
- $\Gamma(S \cup \{x\}) = (\Gamma(S))^{2x+1} \bmod p$
- $\Gamma(S - \{x\}) = \Gamma(S)^{\frac{1}{2x-1}} \bmod p$ (*property 3*)

where $S \subset \left\{ \mathbb{Z}_{(q-3)/2}^+ \right\}^*$ and $x \in \mathbb{Z}_{(q-3)/2}^+$.

4 Data Collection Protocol

In this section we describe our data collection protocol using the set hashing mechanism. Unless otherwise indicated, this section uses the notation of the previous section, whereby $h()$ denotes a collision-free hash function (for example MD5), p is a public prime and “|” denotes concatenation.

Our protocol does not rely on a public key infrastructure, however a shared key between the source and each of the participant hosts is needed. The generation of the individual integrity proof for each data segment by a visited host requires the knowledge of a secret shared between the source and the visited host. In order to perform the verification of the global integrity proof, the source has to know all the individual secrets shared with the hosts visited by the agent.

4.1 Setup

Each host $H_{i>0}$ exchanges a secret shared key K_i , $i < 0 \leq n$ with the source H_0 . For example, K_i can be exchanged using the Diffie-Hellman protocol [2].

The source H_0 sends an agent to visit a set of hosts $\{H_1, \dots, H_n\}$ with an initial *set hash* value Γ and an empty data collection list Σ :

- $\Gamma(S) = \Gamma(\emptyset) = g \text{ mod } p$
- $\Sigma = \emptyset$

4.2 First visit

Each host H_i visited by the agent for the first time, receives:

- $\Gamma(S) = \Gamma(\{O_1, \dots, O_{i-1}\})$
- $\Sigma = \{o_1, o_2, \dots, o_{i-1}\}$

It computes $O_i = h(o_i|K_i)$ and then sends:

- $\Gamma(S) = \Gamma(\{O_1, O_2, \dots, O_{i-1}, O_i\})$
- $\Sigma = \{o_1, o_2, \dots, o_{i-1}, o_i\}$

The submission of an offer is not mandatory for each visited host.

4.3 Update

An offer o_j is updated by host H_j to a new value o'_j in 3 steps:

5. The old offer o_j is replaced by the new offer o'_j in Σ .
6. An intermediate set hash $\Gamma(S')$ is derived from $\Gamma(S)$ by cancelling out O_j
7. The new set hash $\Gamma(S'')$ is computed taking into account O'_j .

The first step is straightforward. In the second step, *property 3* is used to compute a new set hash $\Gamma(S')$ that does not include $O_j = h(o_j|K_j)$:

$$\Gamma(S') = \Gamma(S - \{O_j\}) = \Gamma(S)^{\frac{1}{2^{O_j-1}}} \text{ mod } p$$

In the third step, we use the new value $\Gamma(S')$ and update it with $O'_j = h(o'_j|K_j)$:

$$\Gamma(S'') = \Gamma(S' \cup \{O'_j\}) = \Gamma(S')^{O'_j} \bmod p$$

4.4 Verification

Once the agent goes back to H_0 the source can verify the integrity of the set of offers by using *property 4* to check that:

$$\Gamma(S) = g^{(2O_1 + 1)(2O_2 + 1)\dots(2O_n + 1)}$$

where $\Gamma(S)$ is the integrity check value received by the source.

If this condition fails, none of the offers are considered as valid. It should be noted that the cost of verification is much lower than the cost of generation of $\Gamma(\{O_i | (0 < i \leq n)\})$. During the data collection process, the submission of each data piece o_i requires the computation of a discrete exponentiation whereas the verification of the integrity value for the entire set of data requires only a single exponentiation.

5 Security Evaluation

The data collection protocol presented in this paper fulfills the following security requirements:

- **Data Integrity:** each segment of collected data can only be modified by its originator, tampering with a data segment or unauthorized modification thereof by intruders will be detected by the source. Data modification attempts by an intruder may consist of the update of a data segment o_i with a new value o'_i generated by the intruder or of the replacement of the current data segment with an old value that was previously submitted by its legitimate origin. Both types of modification attempts would require the intruder to first cancel out the integrity proof O_i of the current data segment from the global integrity proof $\Gamma(S)$. Past values of O_i computed by the origin of the data segment are kept secret and computing O_i from the actual data segment o_i requires the knowledge of the secret K_i shared between the origin of the data segment and the source. Appending a new data segment computed by the intruder similarly would require the knowledge of the shared secret K_i . The intruder is thus unable to either get an old value of O_i or to compute a new one that is valid without knowing the shared secret. Another direction for a possible modification attack would consist in deriving O_i from $\Gamma(S)$ and $\Gamma(S \cup \{O_i\})$. That would require the computation of a discrete logarithm which is known as computationally infeasible.
- **Truncation Resilience:** truncation of one or several data segments from a valid offer by a single intruder is infeasible since this would be equivalent to multiple data integrity attacks. As already discussed in 2.1, collusion can result in the truncation of data submitted by all the hosts visited on the path between two malicious hosts. To prevent this type of attacks, the source can publish the received data in order to allow visited hosts to verify and complain if needed, as suggested in [3]. Alternatively a pre-defined list of hosts with a mandatory submission scheme, be it with empty offers, could also alleviate the truncation problem.

- **Insertion Resilience:** no data can be inserted by unauthorized parties because only hosts sharing a secret key with the source can generate a valid integrity proof O_i .

As explained in section 2.1, data confidentiality and non-repudiation are not considered part of mandatory requirements since the main purpose of our security scheme is data integrity in a free competition environment. Nonetheless, these missing features can easily be retrofitted in the data collection scheme using classical data encryption and digital signature mechanisms. For example, data confidentiality can easily be ensured, encrypting the data with the shared key.

6 Conclusion

We described an original protocol to protect dynamic data collected by mobile agents when roaming through a set of hosts. We only considered perfectly autonomous agents, i. e., without any communication with the source or with some kind of trusted party.

Unlike prior work, our protocol allows hosts to update their own submissions without keeping track of past values, and to submit data in a random order thanks to the original set hashing technique. This technique also allows the source to verify the integrity of all the collected data segments without knowing the sequence of data submissions by each host. Moreover, the size of the integrity proof is small and independent of the number of hosts or updates, and the verification is not computationally intensive.

7 References

- [1] David Chess, Colin Harrison, and Aaron Kershenbaum. Mobile agents: Are they a good idea? – update. In *Mobile Object Systems: Towards the Programmable Internet*, pages 46–48. Springer-Verlag, April 1997. Lecture Notes in Computer Science No. 1222.
- [2] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transaction on Information Theory*, 22(6):644–654, November 1976.
- [3] N. Asokan G. Karjoth and C. Gulcu. Protecting the computation results of free-roaming agents. In Kurt Rothermel and Fritz Hohl, editors, *Proc. of the Second International Workshop, Mobile Agents 98*, pages 195–207, 1998. Springer-Verlag Lecture Notes in Computer Science No. 1477.
- [4] Bernd Meyer Ingrid Biehl and Susanne Wetzel. Ensuring the integrity of agent-based computations by short proofs. In Kurt Rothermel and Fritz Hohl, editors, *Proc. of the Second International Workshop, Mobile Agents 98*, pages 183–194, 1998. Springer-Verlag Lecture Notes in Computer Science No. 1477.
- [5] Neal Koblitz. *A course in number theory*. Springer-Verlag, 1994.
- [6] Sergio Loureiro and Refik Molva. Function hiding based on error correcting codes. In Manuel Blum and C. H. Lee, editors, *Proceedings of Cryptec'99 - International Workshop on Cryptographic Techniques and Electronic Commerce*, pages 92–98. City University of Hong-Kong, July 1999.
- [7] George C. Necula. Proof-carrying code. In *Proceedings of the 24th ACM Symposium on Principles of Programming Languages*, Paris, France, January 1997.

- [8] Tomas Sander and Christian Tschudin. Towards mobile cryptography. In *Proceeding of the 1998 IEEE Symposium on Security and Privacy*, Oakland, California, May 1998.
- [9] Bennet Yee. A sanctuary for mobile agents. Technical Report CS97-537, UC at San Diego, Dept. of Computer Science and Engineering, apr 1997.
- [10] Mihir Bellare Bennet Yee. Forward integrity for secure audit logs. Technical report, UC at San Diego, Dept. of Computer Science and Engineering, nov 1997.