Signed and Weighted Trust Credentials in Fraglets

Fabio Martinelli
IIT-CNR
Pisa, Italy
fabio.martinelli@iit.cnr.it

Marinella Petrocchi
IIT-CNR
Pisa, Italy
marinella.petrocchi@iit.cnr.it

ABSTRACT

We continue our line of research by extending the Fraglets computation model with digital signature and security credentials dealing with a quantitative notion of trust. We also give a modeling example where trust credentials are managed at fraglets level.

1. INTRODUCTION

The fraglets communication and computational model has been introduced, and extended, in [14, 15, 17, 18]. This model resembles the chemical reactions in living organisms and it has been shown to be pretty suitable for applications of protocol resilience and genetic programming experiments.

In past work [11, 9], we started introducing security mechanisms to fraglets. In particular, we dealt with data encryption, by giving rules for symmetric cryptography, and with access control, by defining a threat model constraining communication and preventing misuse by malicious fraglets. The main intent has been to enrich the set of communication protocols that fraglets were originally able to model, by giving, contextually, a basement for a formal reasoning.

Along with basic security guarantees like confidentiality and access control, in this paper we contribute by defining digital signatures for fraglets, providing integrity and authentication of origin.

Also, it is worth noticing that the fraglets model natively deals with distributed and autonomic communication. From a security point of view, the management of distributed systems bring to consider *trust* aspects. A distributed architecture quests the need for an upgrade of standard information security mechanisms to the concept of *adaptive security*, that involves adaptation to changing environments. Examples of adaptation may be coming to better decisions when determining what kind of secure mechanisms should be used. Within this context, trust, in its notion of a subjective belief by which an entity expects that another entity performs a given action on which its welfare depends [4], plays a fundamental role, since security and resilience of networks and systems can be improved by exploiting trust information. As an example, a certain access control policy could state that data can be entered by user U not only if a behavioral sequence of actions have been executed by U (*e.g.,* after that critical files have been opened) but also only if the trust level of U is greater than a given value [2]. Also, trust models have been investigating within a *social* context. Recent proposals exploit trust management for setting up recommender systems, see, *e.g.,* [5, 6, 12].

This paper contributes to enrich the fraglets computational model by: 1) defining rules for digital signature; 2) presenting a trust management framework dealing with a quantitative notion of trust and defining operators able to combine trust credentials, both along and across opinion's paths.

The structure of the paper is as follows. Upon recalling basic operations for processing fraglets, the next section presents the work that has been done in the near past for securing communication between fraglets. In particular, symmetric encryption and access control mechanisms are presented. Also, the section contributes in introducing asymmetric cryptography for fraglets. In section 3, we present a fraglets trust management framework, based on the language RTML [8, 16], defining simple and composed credentials, highlighting the presence of trust levels, and the way through which combining them. Section 4 gives an example. Finally, we conclude with final remarks in section 5.

2. FRAGLETS AND SECURITY

A fraglet is denoted as $[s_1\ s_2\ \ldots\ tail]$, where s_i is a symbol and *tail* is a (possibly empty) sequence of symbols.

Nodes of a communication network may process fraglets as follows. Each node maintains a fraglet store to which incoming fraglets are added. Fraglets may be processed within a store, except for the rule that transfers a fraglet from a source store to a destination store.

Fraglets processing is through a simple prefix programming language. Each rule may involve a single fraglet, or two fraglets.

Here, we recall the instructions that are used throughout the paper. In particular, we illustrate rules $Send$ and $Match$, coming from the original set of instructions defined in [15]. Also, we remind the rules for symmetric encryption and decryption, presented in [9].

It is worth noticing that, originally, the fraglet processing engine has been thought to continuously execute tag matching operations on the fraglets in a store, in order to determine the actions that should be applied to them. Within a

Bionetics '08, November 25-28, 2008, Hyogo, Japan

computing security context, tag matching operations may be dangerous, since, when executed with no guards, they may cause security flaws, such as decryption of sensitive information without directly knowing the decryption key. This is the reason why we have introduced a specific threat model for fraglets, contextually to mechanisms for controlling the access to the stores (see [9] for details).

The interested reader can find the original set of rules (no security rules) in the tutorial available online on the fraglets website [3].

Rule *Send* is responsible for transferring a fraglet from a store A to another context (store) B. The rule takes as input the fraglet ${}_A[send\ B\ tail]$, located at store A, and returns the fraglet ${}_B[tail]$, located at store B. The name of the second store is given by the second symbol in the original fraglet. The *Send* operation is unreliable, *i.e.,* it is not certain the *tail* reaches the destination store.

Rule $Match$ concatenates two fraglets with matching tags. With the following two fraglets, $[match\ s\ \ldots tail_1]$ and $[s\ tail_2]$, the rule *match* returns the fraglet: $[tail_1\ tail_2]$, and the matching tag is s.

Symmetric encryption. *Enc* and *Dec* are the rules for symmetric encryption and decryption, that use the same key K. With two fraglets ${}_A[enc\ s\ K]$ and ${}_A[s\ tail]$, respectively representing the symmetric key and the tail to be encrypted, the rule *Enc* returns the fraglet ${}_A[s : tail_K]$. Similarly, fraglet ${}_B[t\ tail]$ is the result of the application of rule *Dec* to the fraglets inputs: ${}_B[dec\ t\ tail_K]$ and ${}_B[t\ K]$.

Access control. The introduction of guarded operations prevents the occurrence of security flaws. In the threat model proposed for fraglets in [9], assumptions are that:

- Stores are of two kinds: trustworthy, untrustworthy. An untrustworthy store is a store where fraglets can be maliciously processed.
- The adversary can:
  - eavesdrop during transmission;
  - process fraglets into untrustworthy stores by means of all usual transformation and reaction rules
  - maliciously deviate fraglets into untrustworthy stores where she can operate on those.
- As usual in computer security, the adversary should not:
  - guess private information that are not in her original knowledge, nor eavesdropped, nor known during processing (*e.g.,* private keys, secret keys);
  - have capability of processing fraglets within trustworthy stores, *e.g.,* by not entering trustworthy stores.

Concerning encrypted fraglets, trustworthy stores contain right secret keys K. Preventing malicious fraglets to enter trustworthy stores, means: 1) not to automatically reveal secret keys; 2) not to have sensitive information decrypted by unintended recipients, by simply (and even not voluntarily!) operating a tag matching reaction.

In order to avoid security flaws of that kind, we have introduced access control mechanisms for fraglets.

In the original syntax, instruction ${}_A[send\ \ B\ \ tail]$ is executed with no guards, *i.e.,* unless unreliability, *tail* may enter B with no control by B itself. In [9], We have introduced so called guarded synchronizations for entering the store.

Guarded synchronizations. The idea is to let trustworthy stores allow a fraglet to enter if and only if a guarded synchronization occurs.

Broadly speaking, a synchronization occurs when two complementary instructions are executed, *e.g.,* send/ receive instructions. Then, a guarded synchronization occurs when two complementary actions are executed, depending on the outcome of some operation.

Within the fraglets world, one may think to allow the reception of *tail* from A iff *tail* satisfies some kind of policy P, for some fraglet populating store B. At conceptual level we have:

$${}_A[\text{send B tail}],\ {}_B[\text{receive A P(tail)}] \rightarrow {}_B[\ \text{tail}]$$

An example policy is the following: access may be granted iff, on receiving from A something encrypted, B has the right decryption key K, *e.g.,* :

$${}_A[\text{send B dec t } tail_K],\ {}_B[\text{receive A K}] \rightarrow\ {}_B[\text{dec t } tail_K]$$

where the outcome ${}_B[dec : t : tail_K]$ is subject to an implicit capability of performing decryption with a certain key.

Note that the intuition behind this is similar to the one of *coactions*, whose introduction leads to the theory of Safe Ambients SA [7], starting from the theory of Mobile Ambients MA [1]. Born with the intent of reasoning about properties of mobile processes, including security, MA defines an ambient as $n[P]$, with n name of the ambient, and P a process running inside. Processes within the same ambient may exchange messages. There are 3 primitives for movement: IN, an ambient enters another ambient; OUT, an ambient exits another ambient; OPEN, that dissolves an ambient boundary, giving access to its contents. SA were proposed for fighting some form of interferences in MA, leading possibly to write *incorrect* programs. Indeed, programs could not behave in the expected way in all contexts, *e.g.,* an ambient n may be opened, but may also jump into ambient m. The introduction of coactions made any movement IN, OUT, OPEN possible only if both participants agree.

Digital signature. A digital signature, the cryptographic construct aiming at assuring authentication of origin and integrity to a message, notoriously exploits mathematical properties of a pair of keys, a public key pk that is publicly available, and the corresponding private key sk, that is supposed to be a secret of its owner.

Within fraglets, we associate to each store a private key, resident in the store, *e.g.,* at store A, we will have fraglet ${}_A[\text{sign s } sk_A]$, with sk_A representing the private key. The corresponding public key is spread to the other stores. Sending of the public key is subject to a guarded synchronisation, to avoid the reception of any unexpected data but the right public key:

$${}_A[\text{send B t } pk_A],\ {}_B[\text{receive A P(t } pk_A)] \rightarrow {}_B[\text{t } pk_A],$$

The rules *Sign* and *Ver*, respectively for applying and verifying a digital signature, are as follows:

With two fraglets, ${}_A[sign\ s\ sk_A]$ and ${}_A[s\ tail]$, respec-

tively representing the signing private key and the tail to be signed, rule *Sign* returns the signature ${}_A[s\ tail_{sk_A}]$. Similarly, ${}_B[t\ tail]$ is the result of verifying the signature ${}_B[s\ tail_{sk_A}]$ by applying ${}_B[t\ pk_A]$.

With a little abuse of notation, we assume that each tail encrypted by a private key, like $tail_{sk}$, is understandable as tail by the recipient. Indeed, being the digital signature a cryptographic mechanism intended to authentication of origin and integrity, but not confidentiality, it is common to send both the plaintext *tail* and the signature $tail_{sk}$.

Note that, within this framework, the capability of signing some piece of data is associated to the store, and not to a specific fraglet inside the store. The non disclosure of the store's private key to malicious fraglets, as well as the capability to forge signatures, is prevented by exploiting the access control mechanism and the guarded synchronization recalled above. Thus, leakage of private keys is prevented by let untrusted fraglets entering trusted stores.

3. FRAGLETS AND TRUST

We consider the Role-based Trust Management Language RTML [16, 8], that defines credentials through roles and attributes, possibly parameterised. As a basic example for credentials, $A.r(p) \leftarrow D$ means that A assigns to D the role (or attribute) r with parameter p.

In this paper, we concentrate on credentials expressing trust towards the capability of performing a functionality f, or that give a recommendation rf regarding a third party able to perform f. Credentials of this kind can also be weighted, *i.e.,* they can specify the degree, called also level, or weight, of the trust assignment.

In [10, 2], we proposed an extension of RTML with trust weights, that we partly recall here:

- (**functional trust**) $A.f(v) \leftarrow D$. A trusts D for performing functionality f with degree v.
- (**recommendation**) $A.rf(v) \leftarrow D$. A trusts D as a recommender with degree v. The recommender is able to suggest someone else for performing f.
- (**indirect functional trust**) $A.f \leftarrow A.rf.f$. This statement says that: 1) if $A.rf(v_1) \leftarrow D$, *i.e., D* is a recommender for A with weight v_1, and 2) $D.f(v_2) \leftarrow C$, *i.e.,* according to D, C is able to do f with weight v_2, then $A.f(v) \leftarrow C$, where $v = v_1 \otimes v_2$, *i.e., C* is *indirectly* trusted by A to perform f with a new weight $v = v_1 \otimes v_2$.
- (**intersection**) $A.f \leftarrow A_1.f_1 \cap A_2.f_2$. This statement defines that if D is trusted both by A_1 to perform f_1 with weight v_1 and by A_2 to perform f_2 with weight v_2, then D is trusted by A to perform f with a new weight $v = v_1 \odot v_2$.

We do not explicitly express weights in the indirect functional trust and linking intersection statements. Indeed, these statements combine basic credentials (the simple member ones) and they determine how weights presented in the basic credentials must be combined too.

Operators $\otimes$ and $\odot$ are introduced to combine the trust weights. Generally speaking, $\otimes$ combines opinions along a path, *i.e.,* A's opinion for B is combined with B's opinion for C into one indirect opinion that A should have for C, based on what B thinks about C. The latter, $\odot$, combines opinions across paths, *i.e.,* A's indirect opinion for X through path p1 is combined with A's indirect opinion for X through path p2 into one aggregate opinion that reconciles both. To work properly, these operators must form an algebraic structure called a c-semiring, [13].

The above credentials can be instantiated at fraglets level:

FUNCTIONAL:
$[cred_f\ (A, f(v), D)\ tail]$
RECOMMENDATION:
$[cred_{rf}\ (A, rf(v), D)\ tail]$
INDIRECT FUNCTIONAL:
$[cred_{if}\ (A, f(v), A, rf(v_1), f(v_2))\ tail]$
INTERSECTION:
$[cred_i\ (A, f(v), A_1, f_1(v_1), A_2, f_2(v_2))\ tail]$

The first two credentials are direct ones.

The Indirect Functional credential contains the rule through which combining two direct credentials, one of functional trust and one of recommendation, to give as output a third credential of functional trust. Thus, input premises are two direct credentials fraglets and the output is the fraglet consisting of the inferred credential.

Indirect Functional

In	*Out*
$[cred_{rf}\ (A, rf(v_1), D)\ tail]$ $[cred_f\ (D, f(v_2), B)\ tail]$	$[cred_f\ (A, f(v), B)\ tail]$

Weight v is obtained combining v_1 and v_2 by $\otimes$.

Similarly, by deconstructing the Intersection credentials as input and output fraglets, we have:

Intersection

In	*Out*
$[cred_f\ (A_1, f(v_1), D)\ tail]$ $[cred_f\ (A_2, f(v_2), D)\ tail]$	$[cred_f\ (A, f(v), D)\ tail]$

Weight v is obtained combining v_1 and v_2 by $\odot$.

4. APPLICATION EXAMPLE

Recommender systems play an important role in dynamic environments when interacting entities are often unknown one to each other. In this case, the reputation level of the unknown entity, built up by direct experience, and reinforced by (trusted) third parties, is significant in strengthening social relationships. Rummble (http://www.rummble.com) is an example of a successful recommender system on the web, based on trust.

We use fraglets to formalize a simple example describing how one can exploit recommendations from (trusted) third parties, to strengthen personal beliefs. Consider the following scenario. An employer E starts the interviews for engaging a new clerk. The criterion for the evaluation is as follows. A candidate C must pass through an interview through which the employer evaluates C's capabilities in doing the work f. Thus, at the end of the interview, E will trust C for performing f with a certain weight v_E. This direct opinion of E is expressed by the following credential: ${}_E[cred_f\ (E, f(v_E), C)]$. Also, E requires a letter of reference coming from a previous employer PE. The

letter could be sent by e-mail directly from PE, digitally signed, in order to prove its authenticity, and expressed by ${}_{PE}[(cred_f\ (PE, f(v_{PE}), C))_{sk_{PE}}]$. The credential says that PE trusts C for doing f with weight v_{PE}. By accepting a letter of reference from PE, E is making the following assertion: ${}_{E}[cred_{rf}\ (E, f(1), PE)\ tail]$, *i.e.,* E completely trusts PE for recommending someone able to perform f. Conclusion to premises ${}_{E}[cred_{rf}\ (E, f(1), PE)\ tail]$ and ${}_{PE}[cred_f\ (PE, f(v_{PE}), C)\ tail]$ is, according to indirect functional, the new credential ${}_{E}[cred_f\ (E, f(v_{PE}), C)\ tail]$, *i.e.,* E has an indirect opinion about trusting C for doing f via PE. In particular, here $\otimes$ is the product operator, and 1 is the neutral element, thus $v_{PE} \otimes 1$ is v_{PE}. Currently, E has two possible weights of trust towards C, *i.e.,* v_{PE} and v_E. The intersection credential contains the rule that allows E to compare the two measures by obtaining a final level of trust $v_{PE} \odot v_E$. Here, the criterion for the comparison is not explicitly defined, since it is up to E's belief instantiating some particular operator for $\odot$. As an example, $\odot$ could be the operator that takes the maximum between v_{PE} and v_E. Also, E may decide to engage C if $v_{PE} \odot v_E$ is greater (or equal) than a certain threshold v_t.

We show fragments of the procedure by using fraglets, to give a flavor of the formalization. Throughout the formalization, we assume that guarded synchronizations occur when *Send* operations are up to be consumed. For the sake of readability, we omit to explicitly represent the guards. Tags t and s are used as convenient matching tags.

Native fraglets at store E:
${}_{E}[cred_{rf}\ (E, rf(1), PE)]$
${}_{E}[cred_f\ (E, f(v_E), C)]$

Bootstrapping phase.
Sending the public key from PE to E.
${}_{PE}[send\ E\ t\ pk_{PE}]$ ${}_{E}[t\ pk_{PE}]$

At store PE:
Signing functional trust credential. Inputs:
${}_{PE}[sign\ s\ sk_{PE}]$
${}_{PE}[s\ cred_f\ (PE, f(v_{PE}), C)]$
Output:
${}_{PE}[s\ (cred_f\ (PE, f(v_{PE}), C))_{sk_{PE}}]$
Match operation to create a send operation. Inputs:
${}_{PE}[match\ s\ send\ E\ ver\ t]$
${}_{PE}[s\ (cred_f\ (PE, f(v_{PE}), C))_{sk_{PE}}]$
Output:
${}_{PE}[send\ E\ ver\ t\ (cred_f\ (PE, f(v_{PE}), C))_{sk_{PE}}]$
Send credential to E:
${}_{E}[ver\ t\ (cred_f\ (PE, f(v_{PE}), C))_{sk_{PE}}]$

At store E:
Verification of signature. Inputs:
${}_{E}[ver\ t\ (cred_f\ (PE, f(v_{PE}), C))_{sk_{PE}}]$
${}_{E}[t\ pk_{PE}]$
Output:
${}_{E}[cred_f\ (PE, f(v_{PE}), C)]$

Indirect functional rule. Inputs:
${}_{E}[cred_f\ (PE, f(v_{PE}), C)]$
${}_{E}[cred_{rf}\ (E, rf(1), PE)]$
Output:
${}_{E}[cred_f\ (E, f(v_{PE}), C)]$

Intersection rule. Inputs:
${}_{E}[cred_f\ (E, f(v_{PE}), C)]$
${}_{E}[cred_f\ (E, f(v_E), C)]$
Output:
${}_{E}[cred_f\ (E, f(v_{PE} \odot v_E), C)]$

5. CONCLUSIONS

In this paper, we have further enriched our security framework for the fraglets computational model (adopted in the BIONETS project), by adding digital signatures, trust and reputation credentials, and a way of managing them. We are able to deal with a quantitative notions of trust by using parametric algebraic operators. We are currently working on the implementation of the security and trust primitives.

6. ACKNOWLEDGMENT

Work partially supported by the EU project FP6-027748 BIONETS (*BIOlogically inspired NETwork and Services*).

7. REFERENCES

[1] L. Cardelli and A. D. Gordon. Mobile ambients. In *FoSSaCS*, pages 140–155, 1998.
[2] M. Colombo, F. Martinelli, P. Mori, M. Petrocchi, and A. Vaccarelli. Fine grained access control with trust and reputation management for globus. In *OTM Conferences (2)*, pages 1505–1515, 2007.
[3] FRAGLETS website. http://www.fraglets.net.
[4] Audun Jøsang. Trust and reputation systems. In *FOSAD*, pages 209–245, 2007.
[5] N. Lathia, S. Hailes, and L. Capra. Trust-based collaborative filtering. In *IFIPTM*, 2008.
[6] G. Lenzini, N. Sahli, and H. Eertink. Trust model for high-quality recommendation. In *Secrypt*, 2008.
[7] F. Levi and D. Sangiorgi. Mobile safe ambients. *ACM Trans. Program. Lang. Syst.*, 25(1):1–69, 2003.
[8] N. Li, J.C. Mitchell, and W. H. Winsborough. Design of a role-based trust management framework. In *S&P*, pages 114–130. IEEE, 2002.
[9] F. Martinelli and M. Petrocchi. Access control mechanisms for fraglets. In *BIONETICS*, 2007.
[10] F. Martinelli and M. Petrocchi. On relating and integrating two trust management frameworks. In *VODCA, ENTCS 168*. Elsevier, 2007.
[11] M. Petrocchi. Crypto-fraglets. In *BIONETICS*, 2006.
[12] D. Quercia, L. Capra, and V. Zanardi. Selecting trustworthy content using tags. In *Secrypt*, 2008.
[13] G. Rote. Path problems in graphs. *Computing Supplementum*, 7:155–189, 1990.
[14] C. Tschudin. Fraglets - a metabolistic execution model for communication protocols. In *Proc. AINS'03*, 2003.
[15] C. Tschudin and L. Yamamoto. A metabolic approach to protocol resilience. In *WAC, LNCS 3457*, 2004.
[16] W. H. Winsborough and J.C. Mitchell. Distributed credential chain discovery in trust management. *JCS*, 11(1):35–86, 2003.
[17] L. Yamamoto and C. Tschudin. Experiments on the automatic evolution of protocols using genetic programming. In *WAC, LNCS 3854*. Springer, 2005.
[18] L. Yamamoto and C. Tschudin. Genetic evolution of protocol implementations and configurations. In *SelfMan'05*, 2005.