A trust protocol for community collaboration

Samuel Galice[1], Marine Minier[1], and Stéphane Ubéda[1]

CITI INSA-Lyon - ARES INRIA Project
CITI, INSA de Lyon, Bâtiment Léonard de Vinci
21 Avenue Jean Capelle, 69621 Villeurbanne Cedex
FirstName.Name@insa-lyon.fr

Abstract. In ambient environments, new security challenges that are not adequately addressed by existing security models appear. In such context, intelligent communication devices participate to spontaneous and self-organized networks where unexpected interactions with unknown devices take place. Without centralized organization, security turns in a risk management problem.
In this paper we propose and analyze a computational model of trust that captures trust dynamics of the human society. In our model, past experiences and recommendations are aggregated in the notion of *history of past interactions* which are protected by cryptographic material. To avoid the trust dissemination, each entity is viewed as an autonomous device and a trust level is computed based only upon selfish evaluation of common trustworthy nodes. Our proposal reduces the complexity of the decision-making process by providing *proved data* that can be the foundation of the final decision. The proposed trust model is described together with an overview of the cryptographic protocol and its security analysis. The trust function is analyzed through intensive simulations depending on the impact of the chosen parameters of the trust evaluation and on the dynamics of the studied groups.
Keywords: trust management framework, cryptographic protocol, Identity-based cryptosystems.

1 Introduction

Nowadays, smart devices such as mobile phones, personal digital assistants and the like, act in a more and more ubiquitous environment. Their wireless communications capabilities grow up very quickly like their computing capacities. New types of services come out from dynamic groups of objects which can act together cooperatively facing various interaction contexts.

Smart communications objects belong to group with long term relations of size scaling from very few (objects belonging to unique person), to hundred of devices. Those objects hosted by people are organized in a social group with common rules. Those communication devices participate to spontaneous and self-organized networks with encountered other mobiles devices and with an always more and more intelligent environment. Contexts of interaction range

Please use the following format when citing this chapter:

Galice, S., Minier, M. and Ubéda, S., 2007, in IFIP International Federation for Information Processing, Volume 238, Trust Management, eds. Etalle, S., Marsh, S., (Boston: Springer), pp. 169–184.

from access to authenticated servers to unexpected interactions with unknown devices.

Such an environment introduces new security challenges that are not adequately addressed by existing security models. Without centralized organization, security turns in a risk management problem where specific trust model and associated cryptographic techniques are required. Each device needs to carry self-contained information and methods to be able to make fully autonomous trust decisions.

In human interaction, trust is a *continuous variable* that is compared to the aim of the interaction to evaluate the risk of the operation. Our computational model of trust captures trust dynamics of the human society. As mentioned in [6], trust is subjective and individual as suggested according to Gambetta's definition [9]: "Trust is the subjective probability by which an individual, Alice, expects that another individual, Bob, performs a given action on which its welfare depends". Trust also depends on stable groups such as family, friends or colleagues at work defining subjective trusted communities.

The aim of this paper is to describe and analyze a complete trust model dedicated to smart mobile communicating devices. Our proposal holds all the desired properties for a distributed trust framework. First of all, the proposed framework *derives from human social system* in order to be socially accepted. Human evaluation of trust is a complex system and is difficult to mimic in a computational model. Therefore, we know that this evaluation is a combination of past experiences and external information that can be simplified as recommendation information. Human evaluation of trust is also depending on the *context of interaction.*

In our trust model, past experiences and recommendations are aggregated in the notion of *history of past interactions* (as proposed in [6]) which are protected by cryptographic material. Context may be derived by collecting past history of objects in the environment. In our model, the acting peer tries to forge a direct experience with the target party using the content of their own histories. We avoid the trust dissemination, each entity is viewed as an autonomous device and the trust evaluation is based only upon selfish evaluation of common trustworthy nodes. The trust level is then computed only after successful transactions corresponding with a positive reputation mechanism as described in [18]. Our proposal reduces the complexity of the decision-making process by providing *proved data* that can be the foundation of the final decision.

Besides already presented properties, our trust model is highly adaptable and parameters can be set to correspond to various model of communities, each with its own trust policy. Our model is also *robust to classical security attacks* like Sybil and man in the middle attacks. And last, our proposal can be fitted in a *light weight* decision module, both in term of required computing capability and bandwidth requirement.

This paper is organized as follows: section 2 presents relevant approaches concerning trust and trust management framework. Section 3 specifies the proposed history based trust approach and provides an overview of our protocol

(already described in [8]) with a dedicated security analysis. Section 4 gives the trust metric and explains the impact of the parameters associated with the metric. Section 5 shows using intensive simulations the effect of the parameters on the dynamics of groups using our model.

2 Related Work

According to [18], trust management systems are classified into three categories: credential and policy-based trust management, reputation-based trust management, and social network-based trust management. This approach depends on the way trust relationships between nodes are established and evaluated. In credential and policy-based trust management system [2–4], a node uses credential verification to establish a trust relationship with other nodes. The concept of trust management is limited to verifying credentials and restricting access to resources according to application-defined policies: they aim to enable access control [10]. A resource-owner provides a requesting node access to a restricted resource only if it can verify the credentials of the requesting node either directly or through a web of trust [11]. This is useful by itself only for those applications that assume implicit trust in the resource owner. Since these policy-based access control trust mechanisms do not incorporate the need of the requesting peer to establish trust in the resource-owner, they by themselves do not provide a complete generic trust management solution for all decentralized applications. Reputation-based trust management systems on the other hand provide a mechanism by which a node requesting a resource may evaluate its trust in the reliability of the resource and the node providing the resource. Trust value assigned to a trust relationship is a function of the combination of the nodes global reputation and the evaluating nodes perception of that node. The third kind of trust management systems, in addition, utilize social relationships between nodes when computing trust and reputation values. In particular, they analyze the social network which represents the relationships existing within a community and they form conclusions about nodes reputations based on different aspects of the social network. Examples of such trust management systems include Regret [16,17] that identifies groups using the social network, and NodeRanking [14] that identifies experts using the social network.

Ambient networks are environments where only a distributed reputation system is allowed [13]: there is neither centralized functions nor central location for submitting the ratings or for obtaining the reputation scores of nodes. Each participant simply records his opinion deduced from his own experience about another party. A node, in order to protect itself from potential malicious nodes, trusts only information which is obtained locally: a communication protocol allows all participants to obtain ratings from each other. The reputation of a target party is computed by a specific agent with the help of requested ratings and possibly from other sources of information. Of course, proper experiences with a target party carry a weight higher than the received ratings. But it is

not always the case and the main difficulty is thus to find the distributed stores which deliver these specific ratings considering that the trust data is disseminated in numerous stores. Nevertheless, this information is easily provided on request for a relying party.

3 Our trust management framework

3.1 Our model

General Overview Current trust management systems suppose that most of the encountered terminals are honest so that the number of malicious information is not enough important to mislead the trust decision process. If this assumption is true in some general context, it does no longer hold for personal communicating devices in ambient and intelligent environment. Except in the case of direct interactions between each mobile device such as in personal experiences, all knowledge comes from uncertified devices. Moreover, the availability of this information is limited because of the restricted size of storage of these mobiles.

Our trust management framework is thus designed for decentralized environment where only partial information is available. Trust is evaluated and derived from the following types of information: past personal experiences, encountered device recommendations, and contextual information such as the moment and the place where the interaction takes place. A history based approach (as in [6, 12]) is used in combination with some cryptographic materials: in case of a successful interaction, each involved node stores a *history element* signed by both parties. The number of interactions with a node called *intensity* of interaction, is also stored. The semantics of a history element is important but this is out of the scope of this paper (see [8]). Each node also carries a *blacklist* which takes into account the untrustworthy nodes. This situation may occur because these nodes were dishonest during several interactions or the service did not repeatedly proceed properly. The full management policy of this blacklist is also out of the scope of this paper.

Thus, a history implies that trust decision process is based on the validity of exchanged information since it not only relies on the honesty of the transmitted information, but also it depends on fully *certified* data: the mobiles are thus incited to be honest regarding transmitted information. In the case of multiple interactions with the same node, the device has to keep only the last proof of interaction to lower the volume of recorded data and the computing overhead.

To sum up our proposition, a node A evaluates the trustworthiness in a node B using only local information: its history H_A, the intensity $I_A(B)$ of the relation with B, its own blacklist BL_A, and the history H_B transmitted by B. With the help of cryptographic algorithms (see section 3.2), node A can check the validity of any history element in H_B as soon as it has the knowledge of the public identity of the involved nodes. As explained later, the verification

is restricted to $H_A \cap H_B$ which corresponds to the known common devices between A and B. Conserve a history element relating A and B means *node A recommends node B* but unlike classical recommendation framework, this assumption can be verified.

The lifetime approach In an environment where exists neither a central regulating entity nor authorizing accreditations or the revocation of objects, a possibility is to let make time: the data elements are automatically revoked after their lifespans expire [15]. A temporal semantics can easily be added to a history element if both nodes agree on a creation and an expiration date. This information is simply concatenated with existent data before the signature. Nevertheless, nothing guarantees that the both entities will choose correct values for this information: the reality may be different (dishonest nodes or malfunction). But there is no real benefit to cheat on these values. Indeed, each entity may filter a received history element according to its local trust policy: an element can be rejected if its creation date is too old, its validity period is considered to be abnormally long although being still valid or if its lifespan is of course expired. No information having an infinite lifespan in the system is guaranteed by this timestamp.

Identity and impregnation It is of course impossible in absolute to avoid the compromise of a node either by a technical action (hacking) or by a social engineering attack (stealing of password, ...). Consequently, an attacker who compromises a mobile has a direct access on the history elements present on this device. This problem is addressed in our model through the impregnation of a device with an identity.

Identity is set at the birth of the mobile device and is the result of a collaboration between the node (or the owner) and a special device called *imprinting station*. Although this imprinting station implements the trust model but it is not certified by any authority. Each imprinting station defines a domain of security which corresponds to a dedicated social group. A domain may contain a large group of mobile devices or just a single smart mobile device embedding its own imprinting station.

At the end, a node A can easily check that an encounter B either belong to the same community or not by verifying their respective imprinted signatures. Then, we could define the $C(A, B)$ parameter which is a Boolean value assigned to true if and only if A and B belong to the same community. We call this value the *community parameter*. Depending of the social model underlying a community, this parameters can be included or not in the trust function.

To mitigate the impact of compromised nodes, the identity has also a lifespan. Before its expiration time, a node needs to be re-initiated by its imprinting station in order to update its new identity lifespan. A cryptographic link is created between two consecutive identities ID_1 and ID_2 (as explained in the section 3.2). While there exists some elements that are non expired or signed with the older identity, this identity is yet presented to check previous history elements. The new identity is used to sign new history elements.

3.2 A detailed approach of our protocol

An entity of our model is equipped at least with a *cryptographic package* what will make it compatible de facto with any other entity of our model, i.e. other objects which implicitly accept the model. When an object received this package and the initial parameters, it can then initiate sessions of communication by the means of the CHE protocol (detailed in [8]). If a session is accepted, the two involved nodes estimate, considering their security policies, that they can trust each other during this interaction.

Starting from an empty history, a node records all the successful interactions made with other nodes in order to support the future spontaneous interactions. To prove the past interactions, it creates with each met node an element of history related to their respective identities and signed by the two parts. Before any interactions, nodes must build a *trust germ*, by counting the number of common nodes they have in their history, or by manually forcing the relation: this is the *bootstrap* phase of our model. If the number of common interactions is sufficient (greater than a threshold p which is a function of the size n of the community and the maximum size H_{max} of the history), they can then interact.

The initial seed of trust Each device receives an initial *trust germ* from its *imprinting station*. It is composed by the following information: an identifier ID_u chosen by the device owner (eMail address or IP address or just a simple name or pseudonym) supposed to be unique within the security domain built by this imprinting station, an identity which is obtained from this identifier by concatenating it with a date d and a lifespan T ($\mathrm{ID} = \mathrm{ID}_u||d||T$), a first pair of private/public key $(S_{\mathrm{ID}}, Q_{\mathrm{ID}})$ for cipher operations, a second pair of keys $(S^S_{\mathrm{ID}}, Q^S_{\mathrm{ID}})$ for the signature and a set representing all the public parameters of the elliptic curves required along computations:

$$\text{Params: } \Omega := \langle \mathbb{F}_p, a, b, P, h, G_1, G_2, e, H_1, H_2, H'_1, H'_2; P_{pub,\Omega} \rangle$$

where: a and b are the parameters of a particular elliptic curve $y^2 = x^3 + ax + b$ on $\mathbb{F}_p$; P, a particular point of this curve of prime order q; h, the cofactor defined as $h = \#E(\mathbb{F}_p)/q$; G_1, is a first additive cyclic group of prime order q built using the P point; G_2, a multiplicative cyclic group of the same order; e, a bi-linear pairing from $G_1 \times G_1$ to G_2; $H_1 : \{0,1\}^* \rightarrow G_1^*$ and $H_2 : G_2 \rightarrow \{0,1\}^n$, two map-to-point hash functions required for the Boneh-Franklin's Identity Based Encryption (BF-IBE) (see [5] for more details); and $H'_1 : \{0,1\}^* \times G_1 \rightarrow G_1$ and $H'_2 : \{0,1\}^* \times G_1 \rightarrow \mathbb{Z}_q$, two hash functions required for the Chen-Zhang-Kim IBS signature scheme (CZK-IBS) (see [7] for more details). Notice that the node public keys are directly derived from their identities due to the use of Identity-Based cryptosystems.

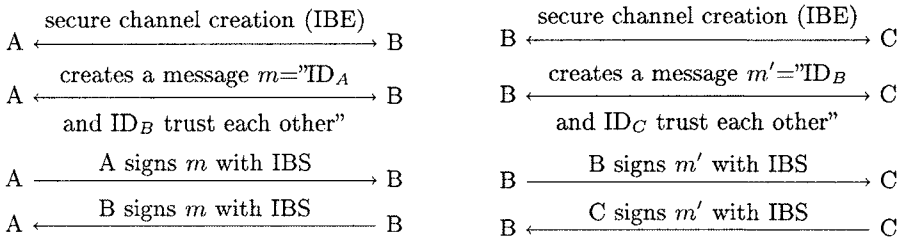
Another important point is that each smart device shares the same following cryptographic algorithms and protocols downloaded from the imprinting station: a fingerprint algorithm, a signature algorithm, a zero-knowledge protocol, a protocol to construct secure channel and the public parameters.

Ω-values are the domain identifier values provided to each node imprinted by the same imprinting station. Every imprinting station possesses the same Ω-values except $P_{pub,\Omega} = sP$ varying along the parameter s, the master key of a station. This value depends on each station and must be absolutely kept secret by it. None of these imprinting stations is supposed to be certified by any authority. Moreover, an independent mobile imprinting itself may be its own standalone security domain. The only values that each smart device has to keep secret is S_{ID} and S^S_{ID} as usually in cryptosystems.

Notice that if a first identity is $\mathrm{ID}_1 = (\mathrm{ID}_u||d_1||T_1)$ where ID_u represents the name or a pseudonym, d_1 a date and T_1 a lifespan, this identity allows to generate the first corresponding key pairs. Then, the updated identity ID_2 is equal to $\mathrm{ID}_2 = ((\mathrm{ID}_u||d_2||T_2)||MAC((\mathrm{ID}_1||\mathrm{ID}_u||d_2||T_2), P_{pub,\Omega}))$ where d_2 represents a second date, T_2 another lifespan and MAC is a MAC algorithm. And so on, the next identities are created using the same operations, generating a MAC chain.

The reciprocal trust Once the initialization phase is done, a node may interact with other nodes without any contacts with its imprinting station. This forms a second phase in the protocol.

The first step of our protocol supposes that both entities Alice and Bob have already interacted at least once and have built a trust bond: this is a message m signed by Bob that Alice publishes in the public part of her history $(m, sign_B(m))$ while Bob publishes $(m, sign_A(m))$ in its own history. This bond could be created by forcing by the hand the beginning interaction as in a Bluetooth like system if the number of common elements of their history were insufficient. Let us note that if Alice and Bob have already met and if this new interaction is successful, they just have to modify the respective values of the intensity and to rebuild a new history element to replace the old one because it contains a timestamp. Suppose now that in the same way Bob and Charlie have built a secure channel to exchange a common message of mutual trust m'.



The second step of our protocol describes a trust bond establishment using history contents between two entities (here Alice and Charlie) that have never met. Thus, when Alice meets Charlie for the first time, they exchange the concatenation of all the public keys Q_{ID} contained in their history. Once this first exchange carried out, Alice and Charlie realize that they have both met

before Bob and want to mutually prove this common meeting. Charlie, first, proves to Alice that Bob trusts him using the message m'. Alice could verify the contents of m' because she knows Bob's public keys from her own previous meeting.

$$\text{A} \xrightarrow{\text{did you meet Bob before ?}} \text{C}$$

$$\text{A} \xleftarrow{(m', sign_{SS_B}(m'))} \text{C}$$

verifies m'

The reciprocal process will be then repeated by Alice.

3.3 Security analysis

Security requirements The following traditional cryptographic properties are guaranteed by our protocol: an offline authentication (users performs each other a weak authentication using the IBE scheme and as Charlie knows the Bob's public keys, he could authenticate his signature), integrity is guaranteed by the hash function used in the IBS scheme as in the classical case of a certificate, confidentiality is guaranteed by the use of the cryptographic IDs. Those IDs also permit to guarantee that the first phase of our protocol was correctly done. The secure channel built at the beginning of the exchange in the first phase also prevents a man-in-the-middle attack.

The user could preserve its anonymity because he is free to choose his own pseudonyms according the context and could have several pseudonyms distributed by different imprinting stations. Those pseudonyms are certified through the used identity-based schemes and they preserve the real identity of their owner, even if his meetings when he acts in the network are known with other peers with pseudonyms. Moreover, each identity defines its own history and all the pseudonyms are certified, thus tackling "Sybil attacks". Our model also guarantees the non-repudiation: each user is preventing from denying previous meetings or actions. Revocation is also possible using the timestamp linked with an ID and included in the key pairs (as previously described in 3.2).

Classical attacks As mentioned in [8] and due to the use of the IBE-scheme, the well known key escrow drawback is inherently present in our protocol. We then suppose that all the imprinting stations must be trusted entities. Otherwise, they can read and send messages instead of nodes. However, the signature scheme used here prevents such an attack from happening because the signature key pair generated is unknown from the imprinting station.

Our trust management framework is a cross-domain protocol: two nodes, not belonging to the same domain (or to the same imprinting station) could nevertheless interact by comparing the contents of their respective histories once

they exchange the public key of their security domains (we suppose here that all the other parameters are the same).

Our protocol also guarantees the non-transferability of the history because only the knowledge of the secret keys allows to use the content of the history (the secure channel initially built prevents the use of the history elements). Then, stolen identities or pseudonyms or histories could not be useful.

Sybil-like attacks A node or a user could using our protocol forges several identities or pseudonyms from the same or different imprinting stations and then uses them in Sybil-like attacks. However, in our model, one identity or pseudonym could only be linked with a particular history.

For example, suppose that an attacker (Eve) wants to attack a single entity Alice, then she creates first several pseudonyms $S_1, \cdots, S_n$. Alice asks her a particular service, they realize that they have enough common history elements to interact. Suppose now that Eve does not provide the corresponding service to Alice with her S_1 pseudonym, Alice then decides to blacklist the S_1 Eve's pseudonym. So, Eve must use an other pseudonym, S_2 for example, if she wants to interact and attack Alice again. To manage this operation, she must build an other time a sufficient number of history elements common with Alice. Even if, she knows the pseudonyms of nodes to meet again with her second pseudonym, she must play an active and positive role inside the "Alice's friends". The attack using several pseudonyms is then very expensive in our case and requires lots of social engineering.

Clone attacks As mentioned in [8], a major attack against our model is the clone one where Alice clones herself with some other terminals. Those clones with exactly the same keys could build a very strong history and have lots of recorded elements and could interact more easily than the others. Therefore, Alice cloned devices could be carried by different persons visiting different places in order to have different histories. This is not considered by us as a major risk since it is a social engineering attack which is difficult to conduct as well as difficult to surround by cryptographic methods.

4 General Context of our analysis

Having presented the basic block of our trust management framework and having discussed its security requirements, we then describe the general context of our framework main processes: how a node A really computes the trust value concerning the node B, supposing that the node A is the service provider - the trusty - whereas the node B is the trustor.

First, we give a general overview of our notations and then we introduce a function which rates the trustfulness between each node.

4.1 General context

For sake of simplicity, we consider a unique community which is characterized by its *size* n. The community dynamics depends of the *interaction rate* of each individual node, i.e. the average number of interactions by unit of time. The other parameters are H_{max} the maximal size of a history which is the same for all nodes and BL_{max} the maximal size of the blacklist which is also the same for all nodes. The node A stores its trusted nodes in its history H_A and reciprocally, it stores untrustworthy nodes in its blacklist BL_A. Hence, to a trusted node B corresponds an element $h_A(B)$ in the history H_A. In addition, each element is tagged with two fields: the first one, denoted by $I_A(B)$, represents the intensity of the relation with B, the second, denoted by $U_A(B)$, represents the *utility* of B, i.e. the usefulness of the node B with respect to the node A. This last notion is related to the number of times this element contributes in the computation of common elements.

In a more general framework, with different communities, trust policy could be different according to either an interaction takes place with a member of its community or not, taking into account the $C(A, B)$ *community parameter.* More deeply, the internal structure of a community could also modify the trust policy: for instance, through the social degree of the community, initial trust may be total: each node is valid and active in the community (for example for objects belonging to a same family). On the contrary, the initial trust may be partial or even non-existent if the social degree of these communities is loose (for example for objects belonging to members of a national sporting federation with several thousand of members). In this case, the weight given to the *community parameter* could no more be the same and the behavior of a mobile in such a community depends essentially of its own experiences through its history.

4.2 Trust function

Direct trust value We first introduce the main element of our trust function. Suppose now that A and B are two nodes belonging or not to the same community. The main element of trust in our model is the threshold of common history elements. To compute this value, we need to introduce the direct trust value:

$$d(A, B) = \alpha|H_A \cap H_B| + (\alpha - 1)|BL_A \cap H_B|$$

where α varies in the interval $[0, 1]$.

This coefficient indicates the weight of the number of common elements $|H_A \cap H_B|$ versus the number of untrustworthy nodes of A that B considers trustfulness. This value obviously admits negative value for some values of $|BL_A \cap H_B|$, but we consider that if its value exceeds a positive threshold p, then the associated value $T(A, B)$, representing the direct trust level, is equal to one, otherwise it is equal to 0. The parameter p defines thus the threshold for a direct trust.

General trust function A trust value is also context-dependent, so we need to combine the direct trust value with other values given by the context of the interaction, limited here to the *intensity* of the relation between the two involved nodes and to the *community parameter*. We then compute the general trust function about the node B viewed by the node A using the following formula:

$$TF_A(B) = \frac{\beta_A T(A,B) + \gamma_A \frac{I_A(B)}{I_{max}(A)} + \delta_A C(A,B)}{\beta_A + \gamma_A + \delta_A}$$

with $T(A,B) = 1$ if $d(A,B) \geq p$, 0 otherwise and with $C(A,B) = 1$ if $\Omega_A = \Omega_B$, 0 otherwise; where β_A, γ_A and δ_A are three parameters that belong to $[0,1]$; and where $I_{max}(A)$ represents the maximal intensity admitted by A. Then, The general trust function gives a trust notation that belongs to $[0,1]$. According this value and the trust threshold t_{ID} defined by each node, the involved nodes could decide to interact or not.

The β, γ and δ values represent the weights of each parameter we want to take into account. They depend on the local trust policy of the node. For example, a node will prefer, if it has never met a node C (then, the corresponding $I_A(C)$ value is equal to 0), to take into account the number of encountered nodes represented by the β parameter than the community one (they belong to the same tennis club). More precisely, the δ parameter represents the degree of structure of a community and will depend on the type of the community.

5 Experiments and Simulation results

We aim here to propose a set of rules for the management of a group by evaluating the various parameters in order to make the mobiles as autonomous as possible: we seek for instance to lower the duration from which the dynamic process takes the top compared to the bootstrap phase (Fig. 1) by adjusting the different parameters (the maximum history size H_{max}, the metric threshold p,...).

In this section, the presented simulations only compute the $d(A,B)$ parameter, the most important one, considering the other ones as some bonus of interactions. An evaluation including the blacklist process efect is also presented. Let us recall also that two nodes having interacted jointly several times keep only one element of history: this element is updated each time as necessary. For needs of the performed simulations, two types of network were considered: the first type was built upon a uniform random distribution which selects the pairs of interacting nodes, while for the second type, the pairs of nodes are picked with respect to a power law distribution.

The bootstrap phase The bootstrap phase of a node is very important in our proposition and requires the intervention of its owner. At initial step, the history of the node A is empty of trusted elements. And thus, the metric above is useless since no terms can be evaluate. Hence, each trusted element $h_A(B)$

(resp. each blacklisted element B) which is added by the user in the history (resp. in the blacklist) of A has a great impact in the dynamics of this node.

It is also important to notice that this phase implicitly has an impact on the defense mechanism of a group: a group may protect itself by imposing a strong *social activity* to users belonging to another groups. A malevolent user which has compromised an object, must act inside the group in order to avoid losing the benefit of his situation. The situation is quite the same for a benevolent user who leaves for a long time his group: if he does not anymore maintain his relations and wants to be reintegrated, he must undertake one more time the *bootstrap* phase. This fact can be seen as a disadvantage of our protocol, nevertheless, initiate a *bootstrap* phase is easier for an authorized user than for an attacker.

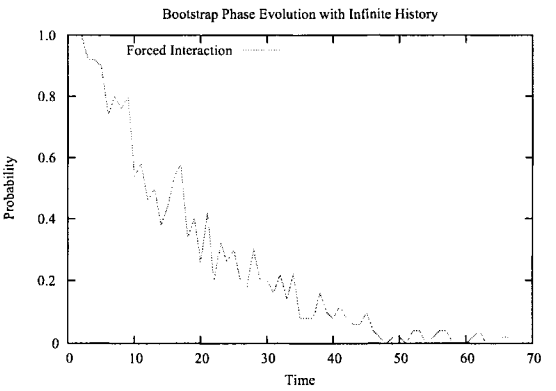


Fig. 1. Evolution of forced interactions by section of 50 steps of time for a community of size $n = 100$ nodes considering a history with an infinite size.

Eviction policy of the history A major element of the policy management of a community is the rule to apply for incorporating a new element in a saturated history. Indeed, the size of the history is necessarily limited for a mobile with small resources. We plan here to set up two modes of replacement. The first mode, which is denoted by FIFO (*First In, First Out*), removes the *oldest* element out of a considered node history: the first withdrawn element has the oldest date. Such a policy allows thus to make disappear the mobiles which are no longer active in the community. The second mode, which is denoted by LFU (*Least Frequently Used*), withdraws the useless elements which appear in the computation of common elements. To measure the importance of each history element, we take into account the number of times this element is used to compute the number of common elements: each common element between the two parts i and j is credited with one point, this corresponds to the value $U_i(j)$ that represents the *utility* of an element. A history element having the

lowest points account is purged and replaced by the partner of the interaction if the current interaction succeeds. We focus on the study of the probability that two distinct nodes have a sufficient number of common elements in their respective histories at time t. This probability is: $P(t) = \frac{2}{n(n-1)} \sum_{i \neq j} T(i,j)$ with $n(n-1)/2$ corresponding to the total number of distinct nodes pairs and $T(i,j)$ is the Boolean value previously defined without taking into account the blacklist process.

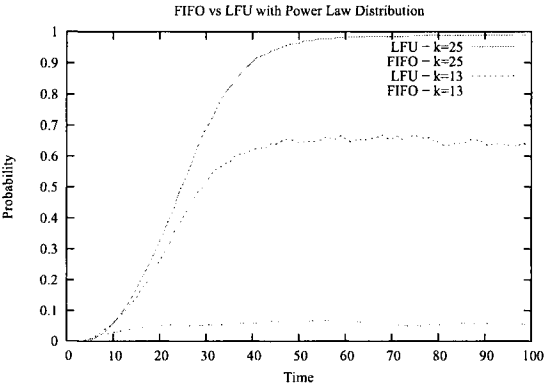


Fig. 2. Evolution of the P probability during time t for several k values (history size) according the eviction mode used (LFU or FIFO). Threshold: $p = 3$ for $n = 100$ nodes).

We have also computed such a probability by using the birthday paradox for several values of H_{max} ranging from $0,5 \times n/\ln(n)$ to $1,2 \times n/\ln(n)$, whereas the threshold p ranging from $0,5 \times \sqrt{n/\ln(n)}$ to $1,5 \times \sqrt{n/\ln(n)}$. On the one hand, a quick analysis shows that the obtained computation results are not really different as well as the case of a random distribution as the case of a power law distribution (however, with a light profit for this last). On the other hand, there is a great difference in behavior of the model according to the mode of replacement used (LFU or FIFO) as shown in Figure 2.

In conclusion, this analysis shows as results that the LFU mode is more efficient to keep the cohesion of a regular interacting nodes group than the FIFO mode. Indeed, the FIFO policy does not take into account the importance of the history elements. On the contrary, if we only keep the most active and useful elements, the chances to find them in other histories are increased. Their number is thus often greater than the threshold p. In consequence, the choice of an eviction policy is very clear: the LFU mode using the $U_i(j)$ value is opted. In addition, fixing a threshold at 3 or 4 is reasonable for communities of 100 or 200 nodes. Beyond, the protocol would require too many user interventions to be viable. Another information from this analysis is the choice of a power law

distribution versus a uniform random distribution to describe the behavior of the nodes activities in the community is negligible.

Impact of blacklist As we announced in the description of the model, the use of a blacklist is highly desirable. In a fully distributed environment, adding a node in a blacklist is the only possible sanction if its behavior is considered to be incorrect. It corresponds to a perfectly identified social behavior. In a risk evaluation approach, it can appear logical to penalize the nodes which present recommendation coming from nodes which are blacklisted.

The disadvantage of the blacklist policy is to prohibit some interaction with honest nodes only because some element of their history have been locally blacklisted. There is thus a balance to find between a very strict policy which will have a very negative impact on the whole of the community and permissive policy which will imply a too important taking risk.

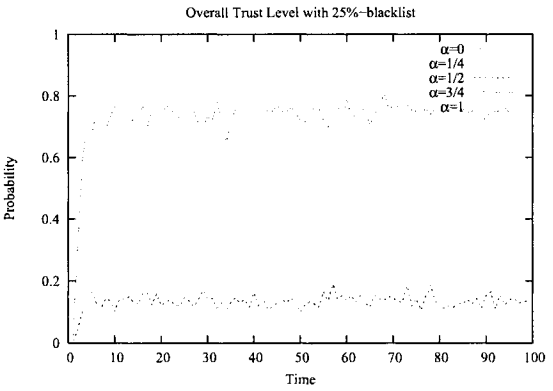


Fig. 3. This figure describes overall trust level of a 25-nodes community according to whether the blacklist elements are considered or not. Each node has a fixed history size $BL_{max} + H_{max} = 10$.

The figure 3 shows evolution of the overall trust along the time for a 25-nodes community. We observe that as the coefficient α decreases as does overall trust. Such a behavior is what expected. This observation could be extended for all values of α: for its small values, the dynamics of the system is stopped due to the higher importance of blacklisted elements than common elements in the history of each node. In contrast, for values around one, the dynamics of the system is not impacted by the presence of these blacklisted elements.

context awareness As we announced at the beginning of this paper, our model allows the introduction of context awareness evaluation. A node A can carry out easily a passive listening of the elements of the histories H_B and H_C exchanged by two nodes which apply our protocol in its radio range. The

node can compute the cardinality of both $H_A \cap H_B$ and $H_A \cap H_C$. By carrying out regular listening, the node can evaluate the social situation in which it is embedded. If the cardinality of intersected histories is always high the node could consider that it acts in a well known environment and may adapt its security policy to this context. The use of our protocol confers on the model two major advantages. First of all, context awareness evaluation is made of data that can by checked cryptographically. Secondly, with only some listening nodes obtain a large list of history elements what enables us to acquire this information very quickly. This second point is important to reduce the cost of such a listing and of the evaluation process.

Let us consider a node A belonging to a community C and 3 contexts where the proportion of C nodes surrounding A are respectively 80%, 50% and 10%. The objective of a context awareness evaluation for the node A is to detect as quick as possible in which A is really embedded. As we explained at the beginning of this section, with a bounded history, even while being in its own community, the intersection of history is not always sufficient for spontaneous interaction. This detection can be established by the gap in the ratio of known nodes over unknown nodes. Known nodes mean here those stored in its history. This ratio may be accurate with few samples. This could be proved analytically using again the birthday paradox.

6 Conclusion

We have proposed a distributed framework that produces trust assessments based on proved direct experience. Our cross-domain scheme supports a weak authentication process, user anonymity and resists to lots of attacks, especially the Sybil-like one. From this basis, we have designed a trust notation that takes into account a local blacklist process and that is context awareness. Finally, we have conducted experiments which show that this framework is suitable for large communities, the bootstrap phase being not an obstacle and that the blacklist process well prevents trusted nodes from the malicious behavior of some peers. As part of future work, we will investigate the dynamics of our model behavior for some special social cases.

References

1. *The First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002, July 15-19, 2002, Bologna, Italy, Proceedings.* ACM, 2002.
2. Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. The KeyNote Trust-Management System Version 2 - RFC 2704. RFC 2704, Available from `http://www.faqs.org/rfcs/rfc2704.html`, September 1999.
3. Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. The role of trust management in distributed systems security. In Jan Vitek and Christian Damsgaard

Jensen, editors, *Secure Internet Programming*, volume 1603 of *Lecture Notes in Computer Science*, pages 185–210. Springer, 1999.
4. Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society, 1996.
5. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - Crypto'2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.
6. Licia Capra. Engineering human trust in mobile system collaborations. In Richard N. Taylor and Matthew B. Dwyer, editors, *SIGSOFT FSE*, pages 107–116. ACM, 2004.
7. Xiofeng Chen, Fangguo Zhang, and Kwandjo Kim. A new ID-based group signature scheme from bilinear pairings. In *Information Security Applications, 4th International Workshop - WISA'03*, volume 2908 of *Lecture Notes in Computer Science*, pages 585–592. Springer-Verlag, 2003.
8. Samuel Galice, Marine Minier, John Mullins, and Stéphane Ubéda. Cryptographic protocol to establish trusted history of interactions. In *Third European Workshop on Security and Privacy in Ad hoc and Sensor Networks*, page LNCS 4357, september 2006.
9. Diego Gambetta. Can we trust trust? In Diego Gambetta, editor, *Trust: Making and Breaking Cooperative Relatioins*, chapter 13, pages 213–237. Published Online, 2000.
10. Tyrone Grandison and Morris Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 3(4), 2000.
11. Rohit Khare and Adam Rifkin. Weaving a Web of trust. issue of the World Wide Web Journal (Volume 2, Number 3, Pages 77-112), Summer 1997.
12. Véronique Legrand, Dana Hooshmand, and Stéphane Ubéda. Trusted ambient community for self-securing hybrid networks. Research Report 5027, INRIA, 2003.
13. Filip Perich, Jeffrey Undercoffer, Lalana Kagal, Anupam Joshi, Timothy Finin, and Yelena Yesha. In reputation we believe: Query processing in mobile ad-hoc networks. *mobiquitous*, 00:326–334, 2004.
14. Josep M. Pujol, Ramon Sangüesa, and Jordi Delgado. Extracting reputation in multi agent systems by means of social network topology. In *AAMAS* [1], pages 467–474.
15. Daniele Quercia, Stephen Hailes, and Licia Capra. Tata: Towards anonymous trusted authentication. In Ketil Stølen, William H. Winsborough, Fabio Martinelli, and Fabio Massacci, editors, *iTrust*, volume 3986 of *Lecture Notes in Computer Science*, pages 313–323. Springer, 2006.
16. Jordi Sabater and Carles Sierra. Regret: reputation in gregarious societies. In *Agents*, pages 194–195, 2001.
17. Jordi Sabater and Carles Sierra. Reputation and social network analysis in multiagent systems. In *AAMAS* [1], pages 475–482.
18. Girish Suryanarayana and Richard N. Taylor. A survey of trust management and resource discovery technologies in peer-to-peer applications.