Published in Security and Privacy in Ad-Hoc and Sensor Networks ESAS06 LNCS 4357

Cryptographic Protocol to establish trusted history of interactions

Samuel Galice[1], Marine Minier[1], John Mullins[2], and Stéphane Ubéda[1]

[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
[2] Département de génie informatique, École Polytechnique de Montréal,
P.O. Box 6079, Station Centre-ville, Montréal (Québec), Canada, H3C 3A7
john.mullins@polymtl.ca

Abstract. In the context of ambient networks, this article describes a cryptographic protocol called Common History Extraction (CHE) protocol implementing a trust management framework. All the nodes are supposed to share the same cryptographic algorithms and protocols. An entity called imprinting station provides them with two pairs of public/private keys derived from their identities. Also, two strange nodes wanting to initiate an interaction have to build a seed of trust. The trust between two nodes is based on a mutual proof of previous common met nodes.
Keywords: cryptographic protocol, trust management framework, Identity based encryption.

Introduction

Nowadays, wireless communications are a critical aspect of computing devices, and offer open solutions for providing mobility and autonomous actions: a smart device as the center of a Personal Area Network is only one major device in an environment where every object will soon be able to communicate. Devices in radio range can potentially establish self-organized networks of two or more objects. In such a context, the peer-to-peer communication capabilities of smart objects will not be restricted simply to access fixed networks and mobility during the use of more complex services, addressed by means of ad hoc communication capabilities, will necessarily receive more attention.

However without centralized trusted agents, we are facing a risk management problem requiring a specific security model and associated cryptographic techniques. Also, we propose a trust decision based on the use of informations cryptographically proved, to reduce this risk. Roughly, smart devices record past interactions between autonomous nodes in a *history* (after a bootstrap phase); to interact, nodes first search previous common met nodes in their histories; then, they mutually authenticate; and finally, they prove, using a security protocol presented here, that these common interactions really took place. If the number

of such common interactions is sufficient that is, upper a certain threshold, then the interaction may occur [17].

The security protocol proposed here is based on the notion of cryptographic ID first introduced by A. Shamir [25], adapted to elliptic curves by D. Boneh and M. Franklin [5] for the cipher and used by Chen, Zhang and Kim [8] for a signature without a trusted PKG (Private Key Generator). The main advantages to use elliptic curve identity based cryptography is the gain in size and in computational time in adequacy with small devices used in ambient networks such as PDAs or smart phones. Moreover, user's public key being or being derived from his identity, there is no requirement of public key directories. Also, key distribution being far simplified, this make ID-based cryptosystems advantageous over the traditional Public Key Cryptosystems (PKCs).

This paper is organized as follows: section 1 presents relevant approaches concerning trust and trust management framework and specifies the proposed history based trust approach. Section 2 provides a detailed description of our protocol while section 3 performs the security analysis of our protocol against classical attacks. Section 4 precises some parameters required for the protocol.

1 Trust management framework and the Common History Extraction (CHE) protocol

Reliability trust, as the name suggest, can be interpreted as the reliability of something or somebody according to the Gambetta's definition [12]. This can be formulated as follows (Reliability Trust): "Trust is the subjective probability by which an individual, Alice, expects that another individual, Bob, performs a given action on which its welfare depends". This definition includes the concept of dependence on the trusted party, and the reliability (probability) of the trusted party, as seen by the trusting party. However, trust can be more complex than Gambetta's definition indicates. For example, Falcone and Castelfranchi [10] recognise that having high (reliability) trust in a person in general is not necessarily enough to decide to enter into a situation of dependence on that person. Therefore, we can also adopt the following definition (decision trust) by McKnight and Chervany [9]: "Trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible".

1.1 Related works

According to [27], 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 we establish and evaluate trust relationships between nodes. In credential and policy-based trust management system [2–4], a node uses credential verification to establish a trust relationship with other nodes. Their 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 [13]. 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 [15]. 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 a social network which represents the relationships existing within a community and form conclusions about nodes reputations based on different aspects of the social network. Examples of such trust management systems include Regret [23, 24] that identifies groups using the social network, and NodeRanking [22] that identifies experts using the social network.

Ambient networks are environments where only a distributed reputation system, i.e. without any centralized functions, is allowed [20]. In a distributed system there is no central location for submitting ratings or obtaining reputation scores of others: each node must protect itself from potential malicious nodes using only self-contained informations and a local control. The trust data can be distributed stores where ratings can be submitted, or each participant simply records the opinion about each experience with other parties, and provides this information on request from relying parties. A relying party, who considers transacting with a given target party, must find the distributed stores, or try to obtain ratings from as many community members as possible who have had direct experience with that target party. The relying party computes the reputation score based on the received ratings. In case the relying party has had direct experience with the target party, the experience from that encounter can be taken into account as private information, possibly carrying a higher weight than the received ratings. The two fundamental aspects of distributed reputation systems are: a distributed communication protocol that allows participants to obtain ratings from other members in the community and a reputation computation method used by each individual agent to derive reputation scores of target parties based on received ratings, and possibly on other information.

In our model that uses a history based approach as the one proposed in [7], the acting peer tries to forge a direct experience with the target party using the content of their own histories. The trust level is then computed only after successful transactions corresponding with a positive reputation mechanism as described in [27].

1.2 Overview of our protocol

Let us first briefly describe the general architecture where trusted histories take place. All smart devices participating to this trust management architecture have to carry common specific cryptographic algorithms and protocols. This is obtained through an *imprinting phase* previous to any other interactions. Special fixed secure functional units called *imprinting stations* are supposed [26]. A device belongs to a domain associated to a specific station and receives from this station an *initial seed of trust* constructed from a secret master key s unique for each station.

Just after each node has received its initial trust germ, history is obviously empty of any interaction. The number of common nodes is of course insufficient to permit an autonomous running and thus, it is necessary a bootstrap phase. So, two persons that want to exchange some services or some informations initiate an interaction by forcing by the hand this particular meeting - as in a Bluetooth like model [14], this gives the desired history element. After this bootstrap period, the nodes use the content of their histories to accept or reject a new interaction, the human intervention is then obvious and no more forcing are needed.

Then, it starts recording a history based upon the knowledge of its interactions with encountered nodes. When two strange nodes interact for the first time, they exchange the concatenation of the public keys of their respective histories and search their common elements. The interaction takes place if the number of common nodes is upper a given threshold. Of course, they need to prove one to each other the common history that could be trusted. The cryptographic protocol described in this paper ensures that any recorded element of history cannot be used by any other node. This issue will be discussed in section 3. This mutually proved history is used to create and enforce the trust relation needed to establish service interactions. At the end of any interaction, a provable value created and signed by the other party constitutes an element of history to parties. This common value also proves the identities of the nodes in presence. The core of the cryptographic method used to extract elements of history common to interacting nodes is called *Common History Extraction* protocol. In our model, there are no trust notation or reputation principles as proposed for example in [18].

This model could be compared to a non transitive version of the one used in the "Web of trust" defined by GnuPG [11]: in our model, we (weakly) authenticate nodes and previous interactions based on successful previous meetings but only at distance one between nodes. We does not consider here a conditionally transitive trust (i.e. a contextual trust). The identity itself is proved also by the use of identity-based cryptography and moreover we use elliptic curves cryptography as basic blocks to design our protocol.

Our proposal is an alternative to pairing model requiring intervention of users and not relevant in the case of short term association between devices. Moreover, in the pairing model, authentication and encryption are made using a symmetric key derived from a PIN information physically entered on each device, making the model prone to simple off-line attack due to this shared

key [14]. Although a distributed n to n pairing model with removal or banishment of devices procedures and rules to solve potential conflicts is proposed in [21] but this model only takes into account a long term virtual private network with a secure long term community. In [6] a history based trust model is also presented but dedicated to group signature and using trusty environment to generate elements of history.

2 A detailed description of the CHE protocol

2.1 The initial seed of trust

Each device receives a *trust germ* from its *imprinting station*. It is composed by the following initial informations: ID an identity (eMail adress or IP address or just a simple name) supposed to be unique in the domain of the imprinting station chosen by the node, $(S_{\mathrm{ID}}, Q_{\mathrm{ID}})$ a first pair of private/public key 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:

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

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 bilinear 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 Encrytion (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 [8] and annex A for more details).

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 the station. This value depends on each station and must be absolutely kept secret by it.

None of those stations is supposed to be certified by any authority. Moreover, an independent mobile, imprinting itself, may be its own standalone security domain. Another important point is that each smart device shares the same 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. The only values that each smart device has to keep secret is S_{ID} and S^S_{ID} as usually in cryptosystems.

In the context of mobile objects with low capacity, cryptography based on elliptic curves (ECC) leads to many advantages. In particular, its use makes

possible to develop algorithms and protocols whose the robustness and the computational and space cost are more advantageous than usual cryptography (as RSA).

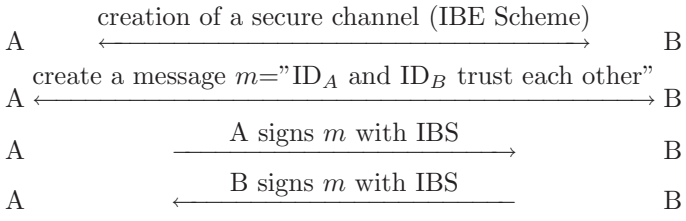
2.2 The Common History Extraction (CHE) protocol: How the mechanism enhances the reciprocal trust

Once the initialization phase done, a node may then interact with other nodes without connexion with its imprinting station. This is the second phase of our protocol.

The "Common History Extraction" protocol extracts the common acquaintances contained in the nodes' history. We have followed the Boneh and Franklin proposition [5] to construct the secret/public key pair of each node, to cipher some messages and also to build a secure channel with a weak authentication. We also have made use of the Chen-Zhang-Kim's Identity Based signature scheme as defined in [8] to sign the required elements. Thus, each node has received two pairs of public/secret keys during the imprinting phase, one pair $(S_{\mathrm{ID}}, Q_{\mathrm{ID}})$ for the cipher operation and one pair $(S^S_{\mathrm{ID}}, Q^S_{\mathrm{ID}})$ for a signature purpose.

The first step of our protocol takes place once Alice and Bob have interacted yet. In this case, they already have built a trust bond using one of the three following possible methods: they could have already met and just have to rebuild a trust nonce; they could also have built a trust bond constructed on the previous common interactions; or during a bootstrap phase and then, the corresponding users have forced by the hand the beginning of the interaction.

At the end of their interaction, Alice and Bob build, in a secure channel using the IBE scheme a message m of reciprocal trust. They sign it. Alice stores in her history $(m, Q_{\mathrm{B}}, Q^S_{\mathrm{B}}, sign_{S^S_{\mathrm{B}}}(m))$ while Bob stores $(m, Q_{\mathrm{A}}, Q^S_{\mathrm{A}}, sign_{S^S_{\mathrm{A}}}(m))$ in his history.



Suppose now that in the same way Bob and Charlie have built a secure channel to exchange a common message m'.

$$\begin{array}{lcl} \text{B} & \overset{\text{creation of a secure channel (IBE Scheme)}}{\longleftrightarrow} & \text{C} \\ \text{B} & \overset{\text{create a message } m'=\text{"ID}_B \text{ and ID}_C \text{ trust each other"}}{\longleftrightarrow} & \text{C} \\ \text{B} & \overset{\text{B signs } m' \text{ with IBS}}{\longrightarrow} & \text{C} \\ \text{B} & \overset{\text{C signs } m' \text{ with IBS}}{\longleftarrow} & \text{C} \end{array}$$

Following that, if Alice meets Charlie, to mutually prove that they have respectively met Bob previously, they will exchange a public part of their histories and Charlie, first, will prove to Alice that Bob trust him using m'.

$$\begin{array}{lcl} \text{A} & \overset{\text{did you meet Bob before ?}}{\longrightarrow} & \text{C} \\ \text{A} & \overset{(m', sign_{S_B^S}(m'))}{\longleftarrow} & \text{C} \\ \text{verifies } m' & & \end{array}$$

The reciprocal process will be repeated by Alice.

Notice also that due to the particular structure of the built message, we could easily add some semantic notions in this message and prove the associated keywords used.

3 Security analysis

3.1 Classical attacks

We sketch in this section the security analysis of our protocol against classical attacks. The security of each cryptographic primitives of [5] and [8] has been clearly established in the initial articles describing those primitives.

This protocol permits to guarantee the following traditional cryptographic properties: weak authenticity (as Charlie knows the Bob's public key, 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 Key Escrow drawback. The use of the IBE scheme introduces the well known key escrow drawback: the PKG (here the imprinting station) could read all the messages exchanged between the nodes imprinted by itself and also cipher some due to its knowledge of each created key pair. That's why we decide

to use the IBE scheme only for the creation of the secure channel, the main step of our protocol is protected by the CZK-IBS scheme where the key pair generated is unknown from the imprinting station.

Man in the middle attack. Due to the use of an IBE scheme for creating the secure channel, each node could verify the validity of an ID by testing if a message could be decipher by this ID. So, a man in the middle attack could be easily discarded between the nodes. However, due to the key escrow drawback of the IBE protocol, a man in the middle attack could be performed by the imprinting station at the creation of the secure channel but this imprinting station could not sign the corresponding message. It could forge false messages but could not prove them. So, in all cases, a man in the middle attack can not be performed.

Denial of service attack. To discard the denial of service attack from a particular node, we suppose here that the first node performing the verification of the common elements of history is the one asking for a service. Our protocol does not prevent a distributed denial of service attack originating from a coalition of nodes against a single one.

3.2 Other attacks

A cross-domain protocol. One of the main advantage of our protocol is that it is a cross-domain protocol: two nodes, not belonging to the same security 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). The main problem in this case is the usurpation of security domains: suppose that an attacker forges an imprinting station with a name that already exists and that it generates exactly the same IDs than the ones of another security domain. At this point two ways are possible: or the attacker also steals the public key of this domain and gives it as its own public key or the attacker gives another public key. In the first case, a node of the attacker could not decipher any message send to it due to the lack of the corresponding secret key. In the second case, the two domains differ from the values of their public keys and are not exactly the same even if they have the same name and the same nodes IDs. So, the computations performed in our protocol will be different and two nodes with exactly the same complete ID do not decipher and sign in the same way.

However, our protocol could not completely discard this problem: two nodes will possess exactly the same ID and the same public key not the same signature key (upon which we perform the last challenge). So, we could differentiate `Alice1` from `Alice2` using their signature private keys.

Non-transferability of History. Suppose now that an attacker steals Alice's identity and all her history and that an attacker could not steal the secret key (in this case the attacker would become an Alice's clone!). So, this attacker, knowing all the Alice' history, could never prove the previous interactions because he does not know the secret S_{ID_A}. However, Alice could always clone herself with some other terminals but the benefit of such an attack is very low:

two nodes *Alice* in an ambient network could hardly construct exactly the same history. However, those nodes 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's 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.

Anonymity. Our protocol does not guarantee the anonymity of the nodes IDs and of the nodes previous meetings as done in [3]. As reported in [27], there is an inherent trade-off between trust and anonymity and our aim is to propose a trust management framework . Thus, we consider here the trust weather of the network: a minimal trust level could only be reached if the nodes reveal some informations about them. We could improve this aspect by changing the transmission of the list of the encounter nodes public keys according the weather of the network (by ciphering the list using the public key of the node with a nonce for example). We also could imagine a mechanism where the nodes broadcast the list of the public keys of the nodes they have met to increase the general trust level of the network: "who meets who ?" will become in this case a trust indicator.

4 Some implementation results

The main parameters in our model are the size of the common history a node requires to accept an interaction and the size of the history itself. Note that, there exists an asymmetry in interactions and the size of the common knowledge required to be receiver or to be provider do not need to be the same. Also the fact that the corresponding node belongs or does not belong to the same security domain may also influence. For inner domain relationship, the size of the common history required is clearly related to the size of the community.

4.1 Probabilistic approach

We consider here that the size of the history is k (using a least-recently-used (LRU) eviction policy) and depends on the total number n of nodes for a given imprinting station. We then want to estimate the required number p of common nodes in the history to permit access to some services. We suppose in this subsection that the nodes meetings are random and does not depend on some laws of proximity.

We then deduce the probability that A and B belonging to the n nodes group have at most p common knowledges (excluding p): $\frac{1}{\left(\binom{n}{k}\right)^2} \cdot \sum_{i=0}^{p-1} \binom{n}{i} \cdot \binom{n-i}{k-i} \cdot \binom{n-k}{k-i}$.

And then, the probability P they have at least p common knowledges is given by:

$$P = Pr(A \cap B \geq p) = 1 - \frac{1}{\left(\binom{n}{k}\right)^2} \cdot \sum_{i=0}^{p-1} \binom{n}{i} \cdot \binom{n-i}{k-i} \cdot \binom{n-k}{k-i} \qquad (1)$$

We have computed the corresponding probability of success in a such case and, inspired from the birthday paradox, have observed that for a given group of size n, if the size k of the history is $n/\ln(n)$ and the threshold number p of common knowledges is about $\sqrt{n/\ln(n)}$ then the probability of success (here to create a trust link) is greater than 50%. So, as an example, if $n = 100$, $k = 22$ and $p = 5$, the success probability is about $56,6\%$ (for the same parameters and $p = 3$, this probability reachs 92%). In this case, we see that the size k of the history is reasonable and could be easily carried by each node and that the number of verifications to perform, given by the p value is also not excessive.

We have summed up in the following table some results concerning the parameters n, k and p and the P probability:

n	$k = n/\ln(n)$	$p = \sqrt{n/(\ln(n)}$	P
100	22	5	56,6 %
200	38	7	61,9 %
500	81	9	94,1 %
1000	145	13	98,9 %
2000	264	17	99,99 %
5000	588	25	100 %
10000	1086	33	100 %

4.2 Simulation results

Our model is dedicated to smart devices, therefore such devices are belonging to a person and so resulting interaction graph is a social graph. Social graphs have been studied for a long time, first by sociologists and more recently by mathematicians [19].

The first property of social graph is the *small world* effect. This property means that even in social graph strongly geographical (so with insular part or social barriers) there exists short connecting path. More recently, some works emphasize recurrent clustering organization which can also affect the way social graph should be studied. The last property, which is very important for simulation, is the skewed degree distribution.

In order to study the p parameter of the trust model we use random graph with skewed distribution [16]. The sequence of degree is obtained through an exponential and continuous power-law distribution generator.

The aim of our simulation is to provide basic idea to verify the correct choice of the p parameter for a given community. The goal is to choose the right p parameter that give large probability of spontaneous interaction between nodes of the community and low probability of interaction between a node not belonging to the same community. This empirical approach need the knowledge of the community, in term of degree distribution. Most of community specification are arbitrary. This is a first step to automatic - or semi automatic configuration for our model and a specific community.

Let us suppose a community denoted C of 30 nodes interacting inside a social group G of 100 nodes, including the C community. We suppose that interactions are more frequent between nodes of C than between a node of C and a node of $G \backslash C$. Therefore, G also constitutes a social group and has same general properties. In our simulations, we define a community with 4 parameters: s the size of the community, d_{min} and d_{max} corresponding to the range of possible degrees of nodes, and α the exponent of the power-law distribution function. Here the parameters both of C and G:

	Nodes	d_{min}	d_{min}	α
C	30	6	12	2.4
G	100	5	10	2.4

We then study, according the p parameter the number of nodes of $G \backslash C$ that will be directly included in the C community considering a history with an infinite size. The p parameter must be chosen very carefully to prevent the community C from being drowned into the group G. We obtain the following results according the p parameter:

p	3	4	5	6
number of nodes included in the community	15-20	3-5	0-1	0
Inside C probability of spontaneous interaction	99.9%	99.98%	98.66%	92.84%

In the previous table one can see the number of nodes from $G \backslash C$ spontaneously included in the community C with respect to the value of p. Depending of parameters, the community can be relatively open - choosing a small value for p means that outsider nodes are easily included in C, or closed - choosing a high value for p makes the spontaneous inclusion of outsiders nodes difficult. The second line of the table shows that whatever the value of p, the community itself works fine with a probability of spontaneous interaction greater than 90% (computed with a history of size 15).

4.3 Implementation aspects

Moreover, to initiate an interaction, the node that provides a service to an other sends it the concatenation of all the public keys Q_{ID} that it has in its history. Each public key is 160 bits length, so in the most popular case with a history containing 30 elements, it must send a chain of 600 bytes, that is very reasonable.

In addition, with a threshold equal to 3, the number of verifications that must be done is very low. We have tested our protocol on a PC powered by a 3 Ghz Pentium IV and have found that the duration for ciphering and signing using our protocol is about 0.78 and 0.9 ms. This is also the duration for the verification of an element.

Conclusion

This paper introduces a new cryptographic scheme to be included in a trust management framework dedicated to *ambient networks*. In such a context, mobile devices need to carry self-contained informations and methods to be able to make fully autonomous security decisions. Some verifications have to be done off line to replace a trusted third party. According the chosen trust policy, the validity of any trust bond can be moderate either by a timeout or by the renewal with fixed intervals of its period of validity contrary to the Bluetooth model where trust is acquired only once.

Our protocol makes use of the notion of cryptographic ID on elliptic curves combined with a history based approach to enforce the trust bond created between the nodes. We think that this approach, even if the size of the history could be a limiting factor when the community of nodes is huge, permits to prevent our framework from all the usual drawbacks as the ones that could appear in reputation models: coalition of nodes for destroying the reputation of a single node, transitivity of the trust,... Our framework takes only into account the past interactions that really happened and nothing else. So, the notion of trust is local and limited to a single node judgment.

In a near future, we want to extend the described protocol for group associations in ad-hoc networks. We also want to provide an anonymous mechanism to protect the privacy of nodes and we want to study other particular network attacks.

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. Laurent Bussard, Refik Molva, and Yves Roudier. History-based signature or how to trust anonymous documents. In Oxford T. Dimitrakos, editor, *Second International Conference on Trust Management*, pages 78–92, April 2004.
7. 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.

8. 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.
9. Norman L. Chervany and D. Harrison Mac Knight. What trust means in e-commerce customer relationships: an interdisciplinary conceptual typology. International Journal of Electronic Commerce, 6, 2, pp. 35-59, 2002.
10. Rino Falcone and Cristiano Castelfranchi. The socio-cognitive dynamics of trust: Does trust create trust? In Rino Falcone, Munindar P. Singh, and Yao-Hua Tan, editors, *Trust in Cyber-societies*, volume 2246 of *Lecture Notes in Computer Science*, pages 55–72. Springer, 2000.
11. The Free Software Foundation. The gnu privacy handbook. `http://www.gnupg.org/gph/en/manual.html`, 1999.
12. Diego Gambetta. Can we trust trust? In Diego Gambetta, editor, *Trust: Making and Breaking Cooperative Relatioins*, chapter 13, pages 213–237. Published Online, 2000.
13. Tyrone Grandison and Morris Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 3(4), 2000.
14. Markus Jakobsson and Susanne Wetzel. Security weaknesses in bluetooth. In David Naccache, editor, *CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 176–191. Springer, 2001.
15. 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.
16. Matthieu Latapy and Pascal Pons. Computing communities in large networks using random walks. *LNCS*, 3733:284, 2005.
17. Véronique Legrand, Dana Hooshmand, and Stéphane Ubéda. Trusted ambient community for self-securing hybrid networks. Research Report 5027, INRIA, 2003.
18. Jinshan Liu and Valérie Issarny. Enhanced reputation mechanism for mobile ad hoc networks. In Christian D. Jensen, Stefan Poslad, and Theodosis Dimitrakos, editors, *iTrust*, volume 2995 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2004.
19. S. Milgram. The small world problem. *Psychology Today*, 1:61–67, 1967.
20. 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.
21. Nicolas Prigent, Christophe Bidan, Jean-Pierre Andreaux, and Olivier Heen. Secure long term communities in ad hoc networks. In Sanjeev Setia and Vipin Swarup, editors, *SASN*, pages 115–124. ACM, 2003.
22. 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.
23. Jordi Sabater and Carles Sierra. Regret: reputation in gregarious societies. In *Agents*, pages 194–195, 2001.
24. Jordi Sabater and Carles Sierra. Reputation and social network analysis in multiagent systems. In *AAMAS* [1], pages 475–482.
25. Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.
26. Frank Stajano and Ross J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Security Protocols Workshop*, volume 1796 of *Lecture Notes in Computer Science*, pages 172–194. Springer, 1999.

27. Girish Suryanarayana and Richard N. Taylor. A survey of trust management and resource discovery technologies in peer-to-peer applications.
28. Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In *Advances in Cryptology - Eurocrypt'2001*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.

A Chen-Zhang-Kim's Identity Based Signature without trusted PKG signature scheme (CZK-IBS)

We review here Chen-Zhang-Kim's IBS without Trusted PKG signature scheme along with their security and computational efficiency in signing and verification phases. Note that whenever we say point, it represents a point on the underlying elliptic curve on which the bilinear pairings are realized [8].

Suppose that there exists an admissible cryptographic bilinear pairing e from $G_1 \times G_1$ to G_2 where G_1 is an additive cyclic group of prime order q, G_2 is a multiplicative cyclic group of the same order and P is an arbitrary generator of G_1. Suppose also that there exists two hash functions H_1' and H_2' defined as follows $H_1' : \{0,1\}^* \times G_1 \rightarrow G_1$ and $H_2' : \{0,1\}^* \times G_1 \rightarrow \mathbb{Z}_q$. Due to the recent results concerning collisions in hash functions as MD4, MD5 and also SHA-0 published in [28], we recommend to use at least SHA-1 and its derivatives as the used hash functions in the proposed protocols.

A.1 Description.

The CZK-IBS scheme is composed of four phases: the first one Extract provides a pair of signature keys built upon an identity ID, the second one Sign describes the signature process, the third one Verify checks the validity of a signature and the last one detects impersonation attacks done by the PKG.

- Extract: Each node receives from its own single identity ID a pair of secret/public keys $(S_{\mathrm{ID}}^S, Q_{\mathrm{ID}}^S)$ for the signature purpose:
  1. The node selects a random $r \in \mathbb{Z}_q^*$ as his long term secret key and sends rP to the imprinting station.
  2. The imprinting station computes $S_{\mathrm{ID}}^S = sQ_{\mathrm{ID}}^S = sH_1'(\mathrm{ID}\|T, rP)$ and sends it to the user via a secure channel, where T is the life span of the secret key s and where $Q_{\mathrm{ID}}^S = H_1'(\mathrm{ID}\|T, rP)$ is the public key linked with ID.
  3. The secret key of the user is the pair (S_{ID}^S, r) and the public key is directly derived from the identity ID.
- Sign: To sign a message m using the secret key (S_{ID}^S, r) corresponding to the identity (public key) ID the following steps are performed by the signer:
  1. Choose randomly $a \in \mathbb{Z}_q^*$ and compute $U = aQ_{\mathrm{ID}}^S$
  2. Compute $V = rH_1'(m, U)$
  3. Compute $h = H_2'(m, U + V)$
  4. Compute $W = (a + h)S_{\mathrm{ID}}^S$.

– Signature : $\sigma = \langle U, V, W, T, rP \rangle \in G_1 \times G_1 \times G_1 \times \{0,1\}^* \times G_1$.

– Verify : To verify a signature $\sigma = \langle U, V, W, T, rP \rangle$ of an identity ID on the message m the verifier does the following:
  1. Compute $Q_{\mathrm{ID}}^S = H_1'(\mathrm{ID} \| T, rP)$
  2. Compute $H_1'(m, U)$ and $h = H_2'(m, U + V)$
  3. Accept the signature if and only if the following equations hold:

$$e(W, P) = e(U + hQ_{\mathrm{ID}}^S, P_{pub}) \tag{2}$$
$$e(V, P) = e(H_1'(m, U), rP) \tag{3}$$

– Tracing: This phase is executed to detect impersonation attacks done by the PKG. The PKG can impersonate a signature for an identity ID as follows:
  1. The PKG chooses a random $r' \in \mathbb{Z}_q^*$ and let $Q_{\mathrm{ID}'}^S = H_2'(\mathrm{ID} \| T, r'P)$.
  2. He then performs the above described signing on a message m to produce $\langle U', V', W', r', P' \rangle$.

  The signature passes the verification test. However, he dishonesty of the PKG can be proved by the user by providing a "knowledge proof" of his secret key to an arbiter.

This scheme is secure against existential forgery under adaptively chosen message and ID attacks in the random oracle model assuming the hardness of CDHP. The scheme eliminates the inherent Key Escrow problem.

Moreover, the signing phase requires 2 map-to-point hash, 3 scalar multiplications and 1 point addition in G_1, 1 cryptographic hash (H_2') operation and 1 addition in $\mathbb{Z}_q$. The verification requires 4 pairing operations, 2 map-to-point hash, 1 scalar multiplication and 2 point additions in G_1 and 1 cryptographic hash operations.