TATA: Towards Anonymous Trusted Authentication

Daniele Quercia, Stephen Hailes, Licia Capra

Department of Computer Science, University College London, London, WC1E 6BT, UK
{D.Quercia, S.Hailes, L.Capra}@cs.ucl.ac.uk

Abstract. Mobile devices may share resources even in the presence of untrustworthy devices. To do so, each device may use a computational model that on input of reputation information produces trust assessments. Based on such assessments, the device then decides with whom to share: it will likely end up sharing only with the most trustworthy devices, thus isolating the untrustworthy ones. All of this is, however, theoretical in the absence of a general and distributed authentication mechanism. Currently, distributed trust frameworks do not offer an authentication mechanism that supports user privacy, whilst being resistant to "Sybil attacks". To fill the gap, we first analyze the general attack space that relates to anonymous authentication as it applies to distributed trust models. We then put forward a scheme that is based on blinded threshold signature: collections of devices certify pseudonyms without seeing them and without relying on a central authority. We finally discuss how the scheme tackles the authentication attacks.

1 Introduction

To produce reliable assessments, distributed trust frameworks must be able *uniquely* to authenticate their users. To see why, consider the following example. Samantha's and Cathy's devices exchange recommendations about shops in their local area. After the exchange, as they know (have authenticated) each other, Samantha's device values Cathy's recommendations based on Cathy's reputation as recommender (i.e., whether her past recommendations have been useful), and vice versa. If it was able to easily generate a new pseudonym, Cathy's device could produce fake recommendations without being traceable. In general, to trace past misbehavior, users should not be able easily to change their pseudonyms - ideally, each user should have one and only one pseudonym.

On the other hand, to protect their privacy, users should *anonymously* authenticate each other, i.e., authenticate without revealing real identities. For example, Samantha may wish to buy kinky boots. She thus uses her mobile device to collect the most useful recommendations from the most trustworthy sources. The recommendation sharing service requires devices to use trust models that, in turn, require users to authenticate. Thus, Samantha's device has to authenticate in order to ask for recommendations; as the subject (kinky boots) is sensitive, the device authenticates itself without revealing Samantha's identity (anonymously).

Existing research in distributed reputation-based trust models does not offer any general solution for *unique* and *anonymous* authentication without relying on a central authority. Some distributed trust models [1] allow the use of anonymous pseudonyms

that, however, suffer from "Sybil attacks" [7]. Others tackle such attacks, but mostly with either centralized solutions [15] or approaches that only apply to limited scenarios [11] [12] [13] [17].

Our contribution lies in: firstly, systematically analyzing the general attack space that relates to anonymous authentication as it applies to distributed trust models; secondly, proposing a scheme that is decentralized, yet general enough to be applied to most of the existing trust models. More specifically, the scheme meets appropriate security requirements and supports desirable features. Security requirements include: (i) *anonymity* to prevent privacy breaches; (ii) *non-repudiation* to prevent false accusation; (iii) *unique identification* to avoid attacks caused by disposable pseudonyms; (iv) *pseudonym revocation* to cope with stolen pseudonyms. Desirable features include: (i) *general applicability*, in that our scheme is general-purpose so that any reputation-based system benefits from it; (ii) *off-line authentication* between two users without relying on anyone else; (iii) *distributed pseudonym issuing*, in that valid pseudonyms are issued without relying on a central authority.

The remainder of the paper is structured as follows. Section 2 discusses related work. Section 3 introduces a scenario that we will use to exemplify our model. Section 4 describes the attacks that relate to anonymous authentication. Starting from both those attacks and the general problem space, section 5 draws security requirements and desirable features for a protection scheme. Section 6 details our proposition and section 7 critically analyzes how it meets the security requirements and supports the desirable features. Section 8 concludes.

2 Related work

Over the course of nearly five years, cooperation and authentication have begun to diverge: authentication has relied on central authorities, while cooperation has migrated to decentralized solutions. Only recently, authentication for cooperative mechanisms started to be decentralized.

Disposable pseudonyms facilitate anonymity, yet hinder cooperation in the absence of a central authority. To see why, consider a collection of actors cooperating. If each actor authenticates himself with an anonymous pseudonym, then he does not have to disclose his real identity and, thus, he can remain anonymous. However, an actor may profit from ease of creating pseudonyms. For example, an actor may authenticate himself with a pseudonym, misbehave, create a new pseudonym, authenticate himself with the new pseudonym (pretending to be new actor), and misbehave again. As a result, the actor misbehaves without being traceable. Resnick and Friedman [15] formally laid down such a problem, presenting a game theoretical model for analyzing the social cost of allowing actors to freely change identities. They concluded that, if actors generate pseudonyms by themselves, all unknown actors should be regarded as malicious. To avoid mistreating all unknown actors, they proposed the use of free but unreplaceable (once in a lifetime) pseudonyms, which a *central* authority certifies through blind signature. A couple of years later, Doucer put similar ideas to test in P2P networks. He discussed the attacks resulting from P2P users who could use multiple identities and named them "Sybil attacks" [7]. He concluded with a critical take on decentralized au-

thentication support: in the absence of a trusted central identification authority, Sybil attacks undermine trust relationships and, thus, cooperation.

At the same time as centralized solutions were used to support authentication, distributed trust models were aiming at promoting cooperation in the absence of a central authority, whilst supporting users' anonymity. For example, EigenTrust [9] is a distributed trust model that suffers from "Sybil attacks" in the absence of a central entity. It uses reputation to decrease the number of downloads of inauthentic files in a P2P file sharing network. In such a network, pseudonyms are used to authenticate peers, thus enabling both peer anonymity and, at the same time, Sybil attacks. EigenTrust partly tackles such attacks, but assumes the presence of a central entity: to get a new identity, a user must perform an entry test (e.g., must read a text off a JPEG) and must send the result to a central authority. As such, it will be costly for a simple adversary to create thousands of users.

Recently, doubts about central authentication solutions for decentralized trust models have surfaced. The SECURE [5] project marks the introduction of a decentralized trust management model with a fully decentralized anonymity support. Within that project, Nielsen *et al.* [6] presented a formal decentralized trust model, and Seigneur and Jensen [16] proposed the use of context-dependent pseudonyms: a user can have more than one pseudonym depending on his/her context of interaction. However, their approach suffers from "Sybil attacks". They thus recently enhanced the original trust model so that trust updates internalize the costs of Sybil attacks [17]. However, such a solution applies only to a specific representation of trust values (as counts of outcomes). More recently, Bussard *et al.* [4] proposed a general distributed privacy-enhancing scheme for trust establishment. However, their work focuses on making users' *recommendations* anonymous and untraceable.

We propose a *general-purpose* scheme based on threshold blind signatures. A user is free to choose his own pseudonym depending on the context he interacts in, although he has only one pseudonym per context. Pseudonyms are certified in a *distributed* fashion through threshold signatures, thus tackling "Sybil attacks". Furthermore, during the certification process, the threshold signature is blinded to ensure *anonymity*.

3 Scenario of mobile recommenders

Here, we introduce an application scenario that we will use throughout the paper to illustrate our scheme.

The scenario features electronic communities of mobile recommenders. A group of people found a community around the shops in Oxford Street: their breakthrough idea consists of customers sharing their shopping experiences through their mobile devices. The initial community starts to grow and lead to the creation of several *communities* around different shop types such as "bookshops", "beauty shops", and "music shops".

Recommendation sharing is automatic and *distributed*, that is, customers store experiences on their mobile devices that they will then automatically share in the form of recommendations, without relying on a central server.

As an incentive for contributions, recommenders remain anonymous so that both their shopping behavior is not associated with their identities and they do not fear retal-

iation from poorly-rated shops. As such, a *unique* and *anonymous* pseudonym authenticates each recommender.

To distinguish good recommendations from bad ones, each mobile device uses a *trust model*: it weights contributions from highly trusted recommenders more than those from untrustworthy recommenders. Most of the distributed trust models (e.g., [1], [10], and [14]) may be used for that. They generally evaluate a recommendation depending on the quality of its originator's past recommendations. Such quality varies: for example, it may be low because the recommender misbehaves (i.e., sends false recommendations).

All of this recommendation sharing service aims to improve and speed up shopping experience: based on the recommendations that their devices have collected, customers may better short-list the most useful shops for their current needs.

4 Attacker model

In order to devise a robust protocol scheme, we must identify possible attacks first. In this section we focus on the possible attacks from which we wish to protect ourselves.

Privacy breaching: A user (attacker) knows the identity of a victim and keeps track of all her interactions. As such, the attacker infers the victim's habits (privacy breaching). For example, the server at a corner shop may log recommendation exchanges among people happening to be in a certain area. Based on these logs, the server may profile people's habits (and, eventually, spending behavior).

False accusation: A user unjustly accuses another user of misbehaving. In our scenario, Cathy's device requests a recommendation Samantha's. The latter device sends the recommendation. The protocol for exchanging recommendations now requires that Cathy's device pays a fee. However, Cathy's device unjustly denies having either requested or received the recommendation.

Sybil-like attacks: A user can manage a set of pseudonyms and, thus, can carry out masquerading attacks (i.e., he masquerades as different entities through its pseudonyms). We categorize such attacks based on whether the attacked target is a single entity or a group of entities:

1. Attacks against a *single entity*. We can identify the following cases:

   Self collusion for ballot stuffing: Here we have a collection of colluding pseudonyms that the same attacker owns. These pseudonyms can be grouped into three categories: pseudonyms used to offer services; pseudonyms used to increase the remaining pseudonyms' reputations as recommenders; pseudonyms used to send positive recommendations about those in the subset of service providers. The attack unfolds as follows. A victim selects service providers based on faked positive opinions. The service providers then offer a low quality of service. The victim lowers its trust level in the abusing service providers. The attacker, which has orchestrated all of this, will never again use the service providers' pseudonyms because it can create other pseudonyms at will. As a consequence, the victim has been deceived and the attacker has profited without repercussions. Transposing

this situation into our scenario, it may happen that Cathy's device (the victim) gets and pays for fake opinions coming from Samantha's. Consider that Samantha's device manages three pseudonyms: S_1, S_2, and S_3. S_1 says to Cathy's device: "S_2 is good at suggesting good recommenders". Cathy's device queries S_2's, which says: "S_3 is good at sharing opinions about shops". Cathy's device then pays S_3's for its opinions. S_3's device shares fake opinions while gaining money. At that point, Samanantha will never again use her pseudonym S_3; she will instead replace it with a newly created pseudonym, S_4.

Self collusion for bad mouthing: A collection of pseudonyms corresponding to the same attacker colludes in that each of them spreads negative recommendations about the same benevolent user. After evaluating those unanimous recommendations, recipients build negative trust in the benevolent user. Hence, the attacker lowers the benevolent user's reputation without harming his own. For example, Samantha does not like Cathy and, thus, her device bad mouths Cathy under the pseudonyms S_1, S_2, and S_3. Upon receiving such opinions, other devices wrongly deduce that four other different devices (persons) dislike Cathy.

2. Attacks against a *group of entities*. We can identify the following cases:

Insider attack: The attacker chooses one pseudonym under which he joins the target group. It then externally misbehaves towards users of *other* groups. They consequently lower the trust in the target group. As such, the attacker lowers the target group's reputation at the price of lowering the reputation of his pseudonym, which he will never use again. For example, some members of the "bookshops" community and others from the "music shops" community look for recommendations about good beauty shops. Being a member of the "beauty shops" community, Samantha's device (under the pseudonym S_1) provides them recommendations about those shops, but fake ones. After experiencing the suggested beauty shops, the recommendation recipients lower the reputation both of S_1 and of the "beauty shops" community (to a certain extent). Samantha's device drops S_1 and will thus not suffer any repercussion in the future.

Outsider attack: Under one pseudonym, the attacker joins some groups *other* than the target group. Within each joint group, it builds up a good reputation in being a reliable recommender and, once reaching the planned reputation level, starts spreading negative recommendations about the target group. As a consequence, the attacker lowers the target group's reputation without harming its own. For instance, Samantha's device joins both "bookshops" and "music shops" communities under one pseudonym. It builds up a good reputation as recommender and then starts to bad mouth about the "beauty shops" community.

Stolen pseudonyms: A user (attacker) steals a victim's pseudonym so as to be able to use it in future interactions. For example, Samantha's device steals Cathy's pseudonym. It then misbehaves under the new pseudonym. Cathy will unjustly suffer from such a theft.

5 Security requirements and desirable features

In the previous section, we described some attacks. As we aim at designing a scheme robust to those attacks, we now infer from them the *security requirements* that our scheme should meet: *anonymity* (a pseudonym does not reveal any information about the real identity of its owner thus preventing privacy breaching); *non-repudiation* (in a reputation-based interaction, each user is prevented from denying previous commitments or actions, thus avoiding false accusation); *unique identification* (a user possesses a unique, valid identifier thus hindering attacks caused by disposable pseudonyms); and *revocation* (once a pseudonym gets stolen, its owner should get a new one at the price of revoking the old one).

Our scheme should also support the following desirable features: *general applicability* (any type of distributed reputation systems benefits from the proposed scheme); *off-line authentication* (users authenticate each other without needing to involve anyone else); *distributed issuing of pseudonyms* (issuing of pseudonyms does not rely on a central authority).

6 The 2-protocol scheme

The scheme consists of two protocols: an induction protocol and an authentication protocol. During a one-shot *induction* protocol, a user obtains his pseudonym (i.e., a public key (anonymous) and corresponding signature). Each time two users wish to authenticate each other, they run through an *authentication* protocol, i.e., they exchange their pseudonyms and then verify their validity. After that, they will use the public key in each other's pseudonyms to encrypt their communication (thus avoiding main-in-the-middle and false accusation attacks).

In this section, we describe these protocols. We first briefly introduce the blind threshold primitives we will use. We then show the bootstrapping procedure. Finally, we describe the protocols in details.

6.1 Blind threshold signature algorithms and protocols

The scheme borrows the protocols and algorithms below from the blind threshold signature literature (see [2] [8] [18]). Blind (t, n) threshold signatures allow n parties to share the ability to blindly sign messages (i.e., sign messages without seeing them), so that any t parties can generate signatures jointly, whereas it is infeasible for at most $(t-1)$ parties to do so.

SETUP: A protocol that generates a public key K_C and n secrets.
Blinding : An algorithm that on input of a message, a random blinding factor r, and the public key K_C, produces a blinded version of the message.
DISTRSIGN: A protocol used by any subset of t parties that on input of t secrets, t randomizing factors, a blinded message and the public key K_C, produces the blind signature of the message.
Unblinding: An algorithm that on input of a blinded message and the random blinding factor r, extracts the message (i.e., removes the blinding factor).

Verify: An algorithm that on input of a message and corresponding signature, determines whether the signature is valid for the message.
PARTIAL: A protocol used by any subset of t parties that on input of t secrets produces t partial secrets.
Secret: An algorithm that on input of t partial secrets determines one single secret.
REFRESHING: A protocol used by $n\prime$ parties that on input of $n\prime$ old secrets produces $n\prime$ new secrets.

6.2 Community bootstrapping

The authentication scheme is based on pseudonyms and on threshold signature. When it is issued, each pseudonym needs to be *certified*, and when it is used, it has to be *verified* (i.e., its certification needs to be checked). This is to ensure that each user has only one pseudonym. Certifying a pseudonym means signing it: in a (t, n)-threshold scheme, the private key used for signatures is built up from t secrets, each owned each by a different party. Verifying a pseudonym means checking its signature; in a threshold scheme, there is a shared and unique public key for such a purpose. Therefore, to certify and verify a pseudonym, a collection of parties (community) needs jointly to create a common public key and a secret for each party. This is done in what we call the community bootstrapping phase.

To clarify, consider n community members (potential signers) and denote them by the set $(signer_1, \ldots, signer_n)$. Each i^{th} member $signer_i$ chooses a random string rs_i. All members submit their strings to the *SETUP* protocol. This produces both the community public key KU_C as public output to all members, plus a $secret_i$ as private output to each member.

6.3 The induction protocol

To avoid attacks caused by disposable pseudonyms, each community member must have no more than one pseudonym. To achieve this, pseudonyms are issued only for *prospective* community members (and not for already certified ones).

A prospective community member has to run through a 5-step *induction* protocol, after which he obtains his pseudonym (to anonymously authenticate himself) and his own secret (to take part in future inductions). To tackle replay and interleaving attacks, each of the following messages includes a timestamp and a signature, which the recipient checks.

Step 1 The prospective member P broadcasts an induction request, that includes the prospective member's certificate $Cert_P$ (i.e., the certified pair of his identity ID_P and public key KU_P). The recipients must know ID_P to verify whether it corresponds to an already certified member or to someone new. Furthermore, to ensure that the request has been generated by a member with that identity, P signs the request ($Signature_{KR_P}$) with his public key KU_P. The use of public key certificates is limited to the induction protocol and does not require to contact any central authority as the certification authority's public key is available in the community.

$$P \rightarrow: \{Cert_P, Timestamp_P, Signature_{KR_P}\}$$

Step 2 Each member who wishes to participate in the induction (denoted by $signer_j$) sends a positive response. The response contains the member's public key KU_j and the threshold t because, to reply back, the prospective member has to know both the public key of the responder and the current t (number of members needed to proceed with an induction; note that t may change as the community size changes). The response also contains a hash value of a randomizing factor $h(rf_j)$ and the community public key KU_C because, after selecting the responses, the prospective member has to generate a public key and blind it, and to do so, it needs KU_C and a set of hash values of randomizing factors. The response is then encrypted with the prospective member's public key.

$$signer_j \rightarrow P : KU_P\{KU_j, t, h(rf_j), KU_C, Timestamp_j, Signature_{KR_j}\}$$

Step 3 Once enough responses (at least t) have been collected, a quorum of t members must have decided to admit the prospective member. At this point, the prospective member chooses t responders and sends them a blinded pseudonym (a blinded public key) and a list containing the chosen responders' public keys and identities. From the list, all the selected responders will know each other's identities and will thus be able jointly to sign the blinded public key. Let us now focus on the composition of the message for this step. Without loss of generality, we indicate the t members with the set $(signer_1, \ldots, signer_t)$. The prospective member randomly creates a key pair (public key AKU_P and private key AKR_P) and submits that public key together with a random number r (blinding factor), the community public key KU_C, and the set $(h(rf_1), \ldots, h(rf_k))$ to the *Blinding* algorithm. From that, it obtains the blinded anonymous public key AKU'_P, encrypts it along with the list with each responder's public key, and sends the encrypted bits to the respective responders. For $j \in [1, t]$:

$$P \rightarrow signer_j : KU_j \ \{AKU'_P, (KU_1, ID_1, \ldots, KU_t, ID_t), Timestamp_P, Signature_{KR_P}\}$$

Step 4 At the end of the induction, the prospective member has to obtain the signature of its anonymous public key and a secret (to participate in future inductions). Thus, in this second-last step, the t-group of members jointly computes and sends the signature of the anonymous public key and a set of partial secrets (from which a secret can be computed). The group does so only if the requesting member has never received a pseudonym before (i.e., if he is actually a new community member and not an old one). More specifically, from the list of responders that the prospective member sent, a group forms. From the initial prospective member's request (step 1), the just-formed group knows the prospective member's identity. It thus checks with up to $(n - 2t + 1)$ community members (external to the group) whether they already released a pseudonym for that identity. If not, the prospective member is entitled to receive its pseudonym and additional information to generate a secret. The group submits AKU'_P, their secrets $(secret_1, \ldots, secret_t)$ and their randomizing factors $(rf_1, \ldots, rf_t)$ to the distributed signing protocol *DISTRSIGN*. This produces the blinded signature s'_P of AKU'_P. As the anonymous public key is *blinded*, the group signed without seeing it. Using the *PARTIAL* protocol, the group then computes a

set of partial secrets $(secret_1^{\mathrm{P}}, \ldots, secret_t^{\mathrm{P}})$ from which the prospective member will be able to compute its own secret. Each group member ($signer_j$) encrypts and sends both s'_P and $secret_j^{\mathrm{P}}$ to the prospective member. For $j \in [1, t]$:

$$signer_j \rightarrow P : KU_P\{s'_P, secret_j^{\mathrm{P}}, Timestamp_j, Signature_{KR_j}\}$$

Step 5 The prospective member first removes from s'_P the random blinding string r through the *Unblinding* algorithm (s_P = *Unblinding*(s'_P, r)). It then submits the received partial secrets to the *Secret* algorithm that computes the single secret $secret^{\mathrm{P}}$.

Now the member has its own valid pseudonym, which consists of its anonymous public key (no one knows it) along with corresponding signature, and its secret so that it can participate in future inductions.

6.4 The authentication protocol

To decide whether to interact, two community members have to authenticate each other (running the authentication protocol) first, then retrieve reputation information associated with each other's pseudonyms and finally evaluate whether each other's reputations are promising enough for embarking on an interaction.

During the *authentication* protocol, two members send their anonymous public keys and associated signatures, verify whether the counterpart's pseudonym is valid and, if so, use the counterpart's anonymous public key (part of the pseudonym) to retrieve reputation information. To clarify, consider that Cathy's device wishes to interact with Samantha's. It thus sends Samantha's device its pseudonym (its anonymous public key and corresponding signature), and so does Samantha's device. They then check the validity of each other's pseudonyms through the *Verify* algorithm. More specifically, Samantha's device submits Cathy's anonymous public key AKU_{Ca}, the corresponding signature s_{Ca}, and the community public key KU_C to the *Verify* algorithm. This returns $true$ if s_{Ca} is a *valid* signature for the anonymous public key (i.e., if a group of at least t members generated it). Cathy's device does the same.

If both verifications run positively, the members involved have each other's valid anonymous public keys and use them to encrypt their subsequent communication, thus avoiding man-in-the-middle attacks and false accusation.

Periodically, a community needs to refresh its members' secrets as some of them may get compromised or the community size changes. To do so, n' members team up ($n' < n$) and submit their secrets to the *REFRESHING* protocol, which generates n' new secrets. Note that the community public key does not change.

7 Analysis of the security requirements and desirable features

Having presented our scheme, we now discuss how it meets the security requirements and the desirable features previously pointed out.

Anonymity: A user pseudonym includes an anonymous public key, i.e., a public key that a group of users certified while knowing the corresponding user identity, but without seeing the key itself. Thus, users are authenticated through their pseudonyms, which do not link to real users' identities.

Non-repudiation: After authenticating, two users encrypt their communication with each other public keys, which are part of their corresponding pseudonyms. As they encrypt the communication, the users cannot repudiate any of the message exchanged.

Unique identification: As we use a (t, n) threshold scheme, we can only have one pseudonym for each context, unless we collude with more than t devices. By properly setting t, we have increased the probability that this will not happen.

Revocation: A user can revoke its pseudonym, e.g., if it is stolen. For that, the user should broadcast its anonymous public and private keys so that he can run the induction protocol once again.

Off-line authentication: A user locally verifies the pseudonyms of his interacting parties through a community public key. As he stores such key, he does not need to contact anyone else for authentication purpose.

Distributed pseudonym issuing: We conceived pseudonym certification to be highly available in that it does not rely on the availability of a unique identification authority, but rather just needs that any t users team up.

General applicability: Most of the existing distributed trust frameworks perform the same steps, i.e., they: authenticate the interacting party, retrieve reputation information about that party, compute trust assessments from the reputation information, make a decision whether to interact and eventually interact. Our scheme is general as it applies to these steps. More precisely, it enhances the first and the last steps: it ensures off-line, anonymous, unique authentication; and it then provides non-repudiation support when interacting.

8 Conclusion

We have proposed a general and distributed authentication scheme (as opposed to existing solutions that rely either on a central entity or on a specific trust framework). The scheme supports user anonymity, whilst being resistant to a wide range of attacks, including "Sybil-like" ones. Most of the existing distributed trust frameworks could make use of it to offer flexible (off-line) authentication without relying on a central service.

The scheme shows one relevant limitation though: that of *weak identification*. If a new person joins (a new anonymous identity appears), and that is the only recent joining, one can link the anonymous identity with the real one. As part of future work, we will investigate whether introducing delays into the scheme will address the problem, whilst not affecting usability.

9 Acknowledgements

The authors gratefully acknowledge the support of the European Commission through the SEINIT and RUNES projects.

References

[1] A. Abdul-Rahman and S. Hailes. Supporting Trust in Virtual Communities. In Proceedings of the 33^{rd} IEEE Hawaii International Conference on System Sciences, volume 6, page 6007, Washington DC, USA, 2000.

[2] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. Journal of the ACM, pages 702–722, Volume 48(4), 2001.

[3] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In Proceedings of Advances in Cryptology, volume 2139, pages 213-229, August 2001. LNCS.

[4] L. Bussard, Y. Roudier, R. Molva. Untraceable Secret Credentials: Trust Establishment with Privacy. In Proceedings of the 2^{nd} IEEE Annual Conference on Pervasive Computing and Communications Workshops, page 122, Orlando, USA, March 2004.

[5] V. Cahill, E. Gray, J.-M. Seigneur, C. Jensen, Y. Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nixon, G. Serugendo, C. Bryce, M. Carbone, K. Krukow, and M. Nielsen. Using Trust for Secure Collaboration in Uncertain Environments. IEEE Pervasive Computing Mobile and Ubiquitous Computing, 2(3):5261, August 2003.

[6] M. Carbone, M. Nielsen, and V. Sassone. A Formal Model for Trust in Dynamic Networks. In Proceedings of the 1^{st} International Conference on Software Engineering and Formal Methods, pages 5463, Brisbane, Australia, September 2003. IEEE.

[7] J. R. Douceur. The Sybil Attack. In Proceedings of the 1^{st} International Workshop on Peer-to-Peer Systems, pages 251-260, Cambridge, U.S., March 2002. Springer-Verlag.

[8] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. Robust and Efficient Sharing of RSA Functions. In Proceedings of the 16^{th} Annual International Cryptology Conference on Advances in Cryptology, pages 157-172, volume 1109, London, UK, 1996.

[9] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In Proceedings of 12^{th} Conference World Wide Web. Budapest, Hungary, pages 640651, 2003. ACM.

[10] J. Liu and V. Issarny. Enhanced Reputation Mechanism for Mobile Ad Hoc Networks. In Proceedings of the 2^{nd} International Conference on Trust Management, volume 2995, pages 4862, Oxford, UK, March 2004. LNCS.

[11] D. Quercia, S. Hailes. MATE: Mobility and Adaptation with Trust and Expected-utility. To appear in the International Journal of Internet Technology and Secured Transactions.

[12] D. Quercia and S. Hailes. Risk Aware Decision Framework for Trusted Mobile Interactions. In Proceedings of the 1^{st} IEEE/CreateNet International Workshop on The Value of Security through Collaboration, Athens, Greece, September 2005.

[13] D. Quercia, M.Lad, S. Hailes, L. Capra, S. Bhatti. STRUDEL: Supporting Trust in the Dynamic Establishment of peering coaLitions. In Proceedings of the 21^{st} ACM Symposium on Applied Computing, Dijon, France, April 2006.

[14] D. Quercia and S. Hailes and L. Capra. B-trust: Bayesian Trust Framework for Pervasive Computing. In Proceedings of the 4^{th} International Conference on Trust Management, Pisa, Italy, May 2006. LNCS.

[15] Resnick, P.: The Social Cost of Cheap Pseudonyms. Journal of Economics and Management Strategy 10(2): 173-199, June 2001.

[16] J.-M. Seigneur and C. D. Jensen. Trading Privacy for Trust. In Proceedings of the 2^{nd} International Conference on Trust Management, volume 2995, pages 93107, 2004. Springer-Verlag. J.-M.

[17] J.-M. Seigneur, A. Gray, and C. D. Jensen. Trust Transfer: Encouraging Self-Recommendations without Sybil Attack. In Proceedings of the 3^{rd} International Conference on Trust Management, volume 3477, pages pages 321-337, 2005. Springer-Verlag.

[18] V. Shoup. Practical Threshold Signatures. In Proceedings of Eurocrypt, LNCS, Volume 1807, pp. 207-220, 2000.