BIONETS

WP 4 – SECURITY

UNIVERSITY OF TRENTO

D4.5 Reputation, Authentication, Anonymity, and Privacy in BIONETS

Reference:	BIONETS/CN-UNITN/wp4/1.0
Category:	Public deliverable
Editors:	Mauro Brunato (CN-UNITN)
Authors:	Jovan Golić (TI); Boldizsár Bencsáth, Levente Buttyán, László Dóra, Tamás Holczer (BUTE); Roberto G. Cascella, Mauro Brunato, Franco Mascia, Elisa Cilia (CN-UNITN)
Verification:	Daniel Schreckling (UNIPASSAU), Marinella Petrocchi (CNR)
Date:	September 28, 2009
Status:	Final
Availability:	Public

SUMMARY

This report collects the solutions that have been proposed for various aspects of BIONETS security related to node and service authentication, reputation, privacy and anonymity.

The BIONETS system allows for high node mobility and disconnected operation; this introduces new challenges for the authentication of entities, as it is not possible to rely on any centralized authority to mediate among nodes. Thus, authentication is achieved in a distributed fashion, sharing the cost and effort among the nodes. To this end, this deliverable proposes to identify U-nodes by names that include physical or logical identifiers along with the hash value of the public keys of the nodes.

The use of logical names inherently provides anonymity, because nodes are authenticated by public keys instead of physical identifiers; if, however, physical identifiers associated with the public keys are required, then a new authentication method is proposed in the absence of certification authorities, based on previously issued certificates along with the reputation values associated with the nodes.

Authentication between T-nodes and U-nodes also requires an original solution, because T-nodes cannot communicate to each other and have limited computational and storage capabilities and no initial setup schemes are allowed to involve U-nodes. Therefore, challenge-response protocols must be based on symmetric key cryptosystems, without pre-shared secret keys between T-nodes and U-nodes. The proposed solution makes use of dedicated T-nodes, with higher computational and storage capabilities, which can implement authentication protocols based on public key cryptosystems.

Authentication of services is another important issue, because the services are expected to change and migrate frequently. The solution is robust and is based on using the public keys for signing service elements, where the keys are arranged in a hierarchical structure with the root public key contained in the name by which the service is identified.

Another problem for which a new solution is proposed is the slow adaptation of common reputation protocols upon arrival of a new U-node. The node maintains a verifiable record of its reputation values to be used to speed up the bootstrap of the system upon formation of a new connectivity cloud, or upon the entrance of a previously unknown node in an existing cloud.

The high dynamicity and mobility of nodes, code and services in the BIONETS framework requires that services are carefully considered before they are migrated to a U-node. A reputation mechanism that allows U-nodes to select which services can be executed based on previous experiences by other U-nodes, and in general to evaluate the *risk* of executing services, is proposed as an extension of the U-node reputation scheme introduced earlier in the project.

While authentication and identification can partly disclose a node's identity to a reliable party, it is important that node anonymity and privacy is maintained with respect to malicious entities. In BIONETS, message dissemination is based on the store-carry-and-forward principle, which raises new aspects of the privacy problem. In particular, an attacker could build a user profile and trace the U-nodes based on this profile even if the message exchange protocol provides anonymity. In this report, a possible solution to this problem is presented, therefore enhancing privacy also on the messaging and service layer.

Contents

DOCUMENT HISTORY

Version History

Version	Status	Date	Author(s)
0.1	Draft	04 July 2008	Jovan Golić, TI
0.2	Draft	02 October 2008	Roberto Cascella, CN-UNITN
0.3	Privacy Section	31 October 2008	László Dóra Levente Buttyán, BUTE
0.4	Privacy via obfuscation	20 November 2008	Roberto Cascella, CN-UNITN
0.5	Reputation for authentication	20 December 2008	Roberto Cascella, CN-UNITN
0.6	Privacy section update	23 April 2009	László Dóra Levente Buttyán, BUTE
0.7	Authentication update	30 April 2009	Jovan Golić, TI
0.8	Service reputation	12 May 2009	Mauro Brunato, CN-UNITN
0.9	Moved parts to the appendix	15 May 2009	Mauro Brunato, CN-UNITN
1.0RC1 1.0RC2 1.0RC3	Drafts for internal review	21 May 2009 25 July 2009 11 September 2009	Mauro Brunato, CN-UNITN
1.0	Final version	28 September 2009	Mauro Brunato, CN-UNITN

Summary of Changes

Version	Section(s)	Synopsis of Change
0.1	Basically all - first Draft	Added
0.2	All sections	Latex formatting
0.3	Section 9 (now 5)	Added
0.4	Section 10 (now removed)	Added
0.5	Section 6 (now 4.1)	Added
0.8	Section 11 (now 4.2)	Added
0.8	Appendix A (now C.2)	Added
1.0RC2	All sections	modified according to first internal review

1 Introduction

This document provides further results of the studies on authentication, reputation, and privacy applied to the BIONETS system, with emphasis on the proposed new solutions and their applicability in the BIONETS context. It is is a continuation of the previous deliverables [1, 2, 3, 4], which, in particular, contain a detailed analysis of the prior art in the respective fields.

The main purpose of this deliverable is to finalize open issues related to authentication and reputation of nodes and services, to introduce a mechanism to ensure node privacy against attacks at the application level aimed at tracking U-nodes based on the contents they deliver, and finally to explore to which extent anonymity can be ensured by the trust model and by application-level privacy enforcement.

In order to achieve authentication in a distributed fashion, sharing the cost and effort among the nodes, Chapter 2 introduces identity definitions for the various entities in the BIONETS system; the system is made more flexible and suitable for anonymous operations by the introduction of two classes of identities, related to physical and logical identifiers. Protocol details are provided in Appendix A.

Chapter 3 defines, more specifically, the authentication issues between the various entities of the system, introduces a new specific mechanism for authenticating services. A detailed description of the protocols is provided in Appendix B.

Chapter 4 completes the description of the reputation system by addressing some issues that were not explicitly covered in previous deliverables [1, 2]. The first problem is that of maintaining a node's reputation when it moves to a new disconnected island by taking advantage of the trust model introduced in previous chapters; the system is based on the maintenance and exchange of tokens, and its robustness is experimentally examined in Appendix C.1. The second problem considered in Chapter 4 is the extension of the reputation scheme introduced in [1] to BIONETS services in order to explicitly cover issues that arise within the BIONETS service model such as hierarchical composability and genetic-like mutation. Appendix C.2 contains an experimental section on the effect of the existing asymmetry between recommending entities (U-nodes) and recommended entities (services), by whose means many selfish U-nodes may decide to support a single misbehaving service in order to have higher chances of success.

Chapter 5, finally, deals with node privacy protection. Some specific new problems are introduced by the store-carry-and-forward manner of the BIONETS networks. The privacy of a system and the anonymity of the users can be a problem on different levels of the communication stack. In the physical layer privacy can be harmed by the physical characteristics of the transceiver (by using remote device fingerprinting techniques), while in the network and transport layers the identifiers must be hidden or anonymized. These problems are considered in [5, 6, 7, 8], however, these methods needs very long interaction time or special expensive hardware equipments. The problem of hidden or anonymized identifiers in delay tolerant and BIONETS networks are addressed in [9, 10], as well as in previous deliverables [1, 2], but the application level problem has not been handled, yet. A BIONETS network specific application level privacy problem is that the nodes can be identified by their stored messages. If an attacker is able to build a user profile using the exchanged application data, the user becomes traceable even if the communication is completely anonymous. Therefore, a new mechanism is proposed in BIONETS networks to ensure untraceability of the U-nodes, namely, to avoid building traceable user profiles.

2 Trust Models

In this Chapter we introduce identity definitions for the various entities in the BIONETS system in order to achieve authentication in a distributed fashion, sharing the cost and effort among the nodes; identities can be based both on physical and logical properties of the entities. Appendix A provides protocol details.

2.1 Trust Model for Public Key Certificates

Initially, each U-node in a network generates and stores locally a pair (or a number of pairs) of public and secret (private) keys, according to an adopted public key cryptosystem. For example, public key cryptosystems using elliptic curves appear to be suitable for applications in mobile devices, because of relatively short public and secret keys and the corresponding digital signatures. In particular, one may apply the ECDSA digital signature protocol [11] for generating randomized signatures, for authentication purposes. Also, by using the same public and secret keys, one can apply the ECDHA key agreement protocol for establishing a common shared secret key between any two nodes in the network, to be used for confidentiality purposes.

2.1.1 Identities

On the other hand, each U-node also decides on a number of identifiers comprising its identity in a network which may depend on a particular service. An identifier that describes a verifiable physical property of a person or a device associated with the U-node is here called a physical identifier. Such an identifier may uniquely determine a person or a device or some other entity, in a given set of entities. For example, identifiers of this type include personal or company names, phone numbers, email or home address, web URIs, device MACs or other information stored in a tamper-resistant way in electronic devices. A physical identifier may also correspond to any chosen physical property, which may be related to the service provided (e.g., a person's age, sex, etc.). It is implicit that these physical properties possess a degree or persistency over a period of time of interest. In particular, it is desirable that they may be practically very difficult or infeasible to change. For a chosen physical identifier, the process of node identification consists of verifying the corresponding physical properties together with the knowledge of the secret key corresponding to the public key associated with the node. For enabling a successful identification, it is necessary that the involved physical properties associated with the same public key are not contradictory.

However, an identifier, as a name, need not correspond to any physical property and in such a case, it may be called a pseudonym. The only property of a pseudonym of a given node is that it is repeatedly used during a period of time (like the physical identifiers). Pseudonyms may be unique or not. In general, one may expect that confidence in physical identifiers is higher than in pseudonyms, especially if the underlying physical properties are difficult to change, but confidence in pseudonyms may also be high depending on the behavior of a node with a given (repeatedly used) pseudonym in the past.

Let u formally denote a generic U-node in a network, where a U-node is considered as a unique physical entity (e.g., device). Let ID_u, P_u, and S_u denote the corresponding identity, the public key, and the secret key, respectively, all stored in the device and used in a network in communication with other nodes. In this way, a U-node u is represented to other U-nodes in terms of the pair (ID_u, P_u), which can hence be considered as the effective name of u. In practice, as the bitsize of P_u may be long, one may instead use

$h(P_u)$, where h is a cryptographic (collision-free) hash function. Usage of public keys for the names of Internet hosts and data items in data-oriented networks can be found in [12] and [13], respectively. ID_u is called a physically verifiable identity (or, simply, a physical identity) if it includes at least one physical identifier. In this case, it makes sense to also introduce the notation $\underline{ID}_u$ for the physical equivalent of the physical identity ID_u, consisting of the physical properties corresponding to physical identifiers, where the identifiers that are not physically verifiable are kept as names only. If ID_u is not physically verifiable, then $\underline{ID}_u = ID_u$. The node u is thus a physical entity in the network with identity ID_u and physical properties $\underline{ID}_u$ that has knowledge of the secret key S_u corresponding to the public key P_u.

2.1.2 Identification

If ID_u is not a physical identity, then identification, i.e., authentication of a node u with the name (ID_u, P_u) at time t is the process of verifying knowledge of S_u by a node with (claimed) identity ID_u at time t (in accordance with the common terminology [14]). It can equivalently be regarded as the process of verifying the (fresh) claim made by u that (ID_u, P_u) and t are authentic. The claim needs to be fresh in order to avoid replay attacks. To this end, u issues a self-certificate by signing (ID_u, P_u) and t by using S_u, and this certificate is then verified by using P_u. A successful identification of u at time t means that (ID_u, P_u) is authentic at time t and thus establishes the binding $P_u \leftrightarrow ID_u$ at time t. This type of identification can be regarded as anonymous identification, because a node is authenticated as an abstract entity knowing a secret key, without any connection to physically verifiable properties of the node. It is here called *type-II identification*.

If ID_u is a physical identity, then identification of a node with the name (ID_u, P_u) at time t is the process of verifying knowledge of S_u by a node with (authentic) physical identity $\underline{ID}_u$ at time t. It can equivalently be regarded as the process consisting of verifying that $\underline{ID}_u$ is authentic for u and verifying the (fresh) claim made by u that (ID_u, P_u) and t are authentic. The second part is achieved in the same way as described above, thus establishing the binding $PK_u \leftrightarrow ID_u$ at time t. The (important) first part can be achieved directly, by verifying the physical properties in $\underline{ID}_u$, thus establishing the binding $\underline{ID}_u \leftrightarrow ID_u$ at time t. Direct verification of physical properties is the critical part and requires establishing some sort of physical contact or a reliable physical channel (e.g., visual contact or voice channel or tamper-resistant devices) with the node to be identified. To this end, a node may also use the Internet connection in the presence of access points (APs), since the Internet services provide a relatively high confidence in the identifiers such as email addresses and web URIs. Altogether, a successful identification of u at time t means that $(\underline{ID}_u, P_u)$ is authentic at time t and thus establishes the binding $P_u \leftrightarrow \underline{ID}_u$ at time t. This type of identification can be regarded as real identification, because a node is authenticated as a concrete entity with physically verifiable properties knowing a secret key. It is here called *direct type-I identification*.

Alternatively, verification of physical properties at time t can be performed indirectly, by verifying the claim made in the past by the verifying node or by any other node presently or in the past that a successful type-I identification of u was performed at that time. This type of claim is made in the form of a type-I public key certificate to be described in Section 2.1.3. In this case, a high level of trust of the verifying node in the node making the claim with respect to performing direct type-I identification is necessary. Since the times are typically different, persistency of the underlying physical properties is hence required. The corresponding type of identification is then called *indirect type-I identification*.

2.1.3 Type-I Certificates for Public Keys

If a node v with name (ID_v, P_v) performs a successful type-I identification at time t of a node u with name (ID_u, P_u), where ID_u is a physical identity, then it can issue a certificate that $(\underline{ID}_u, P_u)$ is authentic at time t along with the associated confidence or trust value from the range [0,1], which is here called the validity of a certificate. Such a certificate is called type-I certificate. It establishes the binding $P_u \leftrightarrow \underline{ID}_u$ at time t, and its validity is the trust of v in this binding. If the physical properties in $\underline{ID}_u$ are verified directly, then the certificate is called direct type-I certificate. If these properties are verified indirectly, by using other certificates, then the certificate is called indirect type-I certificate. For direct type-I certificates, the associated validity is normally equal to 1, but maybe somewhat smaller if the used physical channel is not fully reliable (e.g., a voice channel over the mobile phone or checking knowledge of some previously established common secret). More precisely, we thus have

$$valid_v(P_u, \underline{ID}_u) = trust_v(P_u \leftrightarrow \underline{ID}_u) \tag{1}$$

$$cert_v^I(P_u, \underline{ID}_u) = sign_{S_v}(ID_v, P_v, ID_u, P_u, valid, t, t'), \tag{2}$$

where $valid$ stands for $valid_v(P_u, \underline{ID}_u)$, t is the issuance time, t' is an optional expiry time, and it is assumed both here and throughout that a certificate as a digital signature of some data implicitly contains this data. It is assumed that each node in a network stores issued and received type-I certificates. Alternatively, instead of storing the issued certificates as a whole, a node may only store the information certified, whereas a digital signature can always be reproduced or verified, by using the underlying secret and public keys, respectively.

2.1.4 Type-II Certificates for Type-I Trust

For indirect type-I identification, one may combine different direct type-I certificates, e.g., by using certificate chains or their parallel combinations. To this end, however, apart from high validities of individual certificates, high levels of trust of the verifying node in the intermediate nodes, issuing individual type-I certificates, with respect to performing direct type-I identification is necessary. More precisely, we deal with the trust for assigning genuine validity values to issued direct type-I certificates and, in particular, for assigning the maximal validity 1 only upon successfully verifying the physical properties corresponding to the physical identifiers. This trust for issuing valid direct type-I certificates is here called type-I trust.

Consider a basic model where a node v wishes to perform indirect type-I identification of a node u on the basis of a direct type-I certificate previously issued to u by an intermediate node x. The objective of the node v is then to determine the trust $trust_v(P_u \leftrightarrow \underline{ID}_u)$ in this model. With respect to the probabilistic interpretation of trust, we then have

$$\begin{aligned} trust_v(P_u \leftrightarrow \underline{ID}_u) &= trust_x(P_u \leftrightarrow \underline{ID}_u) \cdot trust_v^I(P_x, \underline{ID}_x) \\ &= valid_x(P_u, \underline{ID}_u) \cdot trust_v^I(P_x, \underline{ID}_x), \end{aligned} \tag{3}$$

where $trust_v^I(P_x, \underline{ID}_x)$ denotes the type-I trust of v in x for issuing valid type-I certificates. In the presence of certification authorities, this trust can be maximal possible, that is, equal to 1. The same is the case if $x = v$, i.e., if v already issued a direct type-I certificate to u in the past. However, in the absence of certification authorities, this trust is typically lower. The main point of our approach is to decompose type-I trust into two components, one related to trust in the public key P_x and the other related to trust in the physical identity $\underline{ID}_x$. The first component, $trust_v^I(P_x)$, derives from a repeated usage of P_x (more precisely, the corresponding secret key S_x for signing), while the other component, $trust_v^I(\underline{ID}_x)$, derives from physical properties in $\underline{ID}_x$, which need to be verified. The two components are here called PK-based and ID-based type-I trusts, respectively. Since at least one of them, but not necessarily both of them, should be high in order for the overall type-I trust in x to be high, they are proposed to be combined by

$$trust_v^I(P_x, \underline{ID}_x) = \max\left(trust_v^I(P_x),\ trust_v^I(\underline{ID}_x) \cdot trust_v(P_x \leftrightarrow \underline{ID}_x)\right), \tag{4}$$

because the ID-based type-I trust in x needs to be coupled with the trust in the binding between the public key and the physical identity, whereas this is not required for the PK-based type-I trust. It should be noted that the PK-based and ID-based type-I trusts are not independent when conditioned on $\mathbf{valid}_v(P_x, \underline{ID}_x)$. In particular, $trust_v^I(\underline{ID}_x) = 1$ along with $\mathbf{valid}_v(P_x, \underline{ID}_x) = 1$ imply that $trust_v^I(P_x) = 1$.

Eqns. (3) and (4) combined establish a recursion for trusts of v in bindings between public keys and physical identities, which reduces this trust for u to trust for x. It is thus required that the PK-based or ID-based type-I trust of v in x is nonzero. If one of them is not defined, then it is formally treated as zero. The recursion is effective if the ID-based type-I trust is higher than the PK-based type-I trust, which should be normally expected. In this case, at least one other type-I certificate issued to x (e.g., by v) is needed for the recursion to be computed. Otherwise, if the ID-based type-I trust is not higher than the PK-based type-I trust, then (3) and (4) directly give

$$trust_v(P_u \leftrightarrow \underline{ID}_u) = valid_x(P_u, \underline{ID}_u) \cdot trust_v^I(P_x). \tag{5}$$

It is important to note that (5) does not require any other type-I certificates and the trust will be high if the validity of the type-I certificate issued by x to u and the PK-based type-I trust of v in x are both high.

While it is expected that the ID-based type-I trust is predominantly subjective, the PK-based type-I trust can be measured objectively by considering the previous practice of issuing valid direct type-I certificates by a node with a given public key. It is important to note that for this purpose, the nodes do not have to be authenticated by type-I identification, because type-II identification suffices for authenticating the public keys. As such, the PK-based type-I trusts can be exchanged between different nodes and combined together in terms of serial, parallel, and more complex combinations. To this end, the PK-based type-I trust of v in x can be signed in a form of type-II certificate

$$cert_v^{II}(P_x, ID_x) = sign_{S_v}(ID_v, P_v, ID_x, P_x, trust_v^I(PK_x), t, t'). \tag{6}$$

As, in principle, it is also possible to exchange the ID-based type-I trusts between different nodes, we may call the certificate in (6) a PK-based type-II certificate and then also define an ID-based type-II certificate by

$$cert_v^{II}(P_x, \underline{ID}_x) = sign_{S_v}(ID_v, P_v, ID_x, P_x, trust_v^I(\underline{ID}_x), t, t'). \tag{7}$$

Otherwise, if the exchange is not envisaged, then the ID-based type-I trusts do not have to be signed.

The PK-based type-I trust can be computed automatically on the basis of direct type-I certificates issued by x as well as those issued by v, where only the certificates with a high validity count. However, the direct type-I certificates issued by x should preferably be sent to v, not by x, but as received certificates by other nodes in a network, over a period of time, preferably signed by their public keys in a form of type-II identification. Otherwise, x might generate a large number of type-I certificates issued to fictive nodes whose validity cannot be verified. The fictive nodes may include fictive identities and public keys or fake identities for targeted public keys. These certificates are then contrasted with those issued by v over the same period of time in order to find possible conflicts. A conflict of type-I certificates is a pair $cert_x^I(P_u, \underline{ID}_u)$, $cert_v^I(P_u, \underline{ID}'_u)$ such that the physical properties in $\underline{ID}'_u$ and $\underline{ID}_u$ are contradictory (e.g., two different personal names ID'_u and ID_u). If there is at least one conflict found, then v assigns $trust_v^I(P_x) = 0$ and, if v previously determined that $valid_v(P_x, \underline{ID}_x) = 1$, it then also assigns $trust_v^I(\underline{ID}_x) = 0$. If not, then, given a security parameter m and a period of time T, v computes

$$trust_v^I(P_x) = 1 - \left(1 - \frac{1}{m}\right)^{\sum_u valid_x(P_u, \underline{ID}_u)}, \tag{8}$$

where the summation is only over high validities of type-I certificates issued by x and received from x by other nodes in the past over T. The PK-based type-I trust is computed only if there is at least one such certificate. The zero PK-based type-I trust thus implies that the corresponding distrust is equal to 1, whereas, if the PK-based type-I trust is not computed, then the corresponding uncertainty is equal to 1 (in the model [15] where the sum of trust, distrust, and uncertainty is equal to 1). The same interpretation is also given to the ID-based type-I trust. The security parameter m is the transition threshold from low to high values of trust, and for the sum in the exponent equal to m, the trust value is roughly $1 - e^{-1} \approx 0.63$.

More generally, in the case of a conflict found, v puts the found fake node name (ID'_u, P_u) on a black list of nodes with zero public key validities. Also, if the found contradictory type-I certificate is signed by a receiving node u, then v assigns $trust_v^I(P_u) = 0$ and $trust_v^I(\underline{ID}_u) = 0$, because the conflict found then means that a node knowing the corresponding secret key S_u and having the real identity $\underline{ID}_u$ attempted to use a fake type-I certificate and, hence, cannot be trusted itself for issuing valid type-I certificates to other nodes.

It is assumed that each node in a network stores issued and received PK-based type-II certificates, apart from type-I certificates. It also stores the ID-based type-I trusts in other nodes, possibly in the form of ID-based type-II certificates, but these certificates do not have to be exchanged with other nodes necessarily. The issued PK-based type-II certificates can be updated, possibly periodically, by storing the relevant information related to type-I certificates. Instead of storing the whole type-I certificate issued to u by x, for comparison purposes it is sufficient to store only $ID_x, h(P_x), ID_u, h(P_u), valid, t, t'$ in appropriately sorted tables. The black list of the found fake node names is stored similarly.

2.2 Trust Model for Public Key Certificate Graphs

In a general scenario for indirect type-I identification, consider any set of network nodes that issued type-I and PK-based type-II certificates to each other in the past. The certificates can be represented as branches in the associated certificate graphs $cert^I G$ and $cert^{II} G$, respectively. Nodes in these directed graphs correspond to the node names (ID_x, P_x) and branches to the issued type-I and PK-based type-II certificates, respectively. In practice, only the type-I certificates with medium or high validities are taken into account. The fake node names from the local black list of v are all removed from these graphs. More precisely,

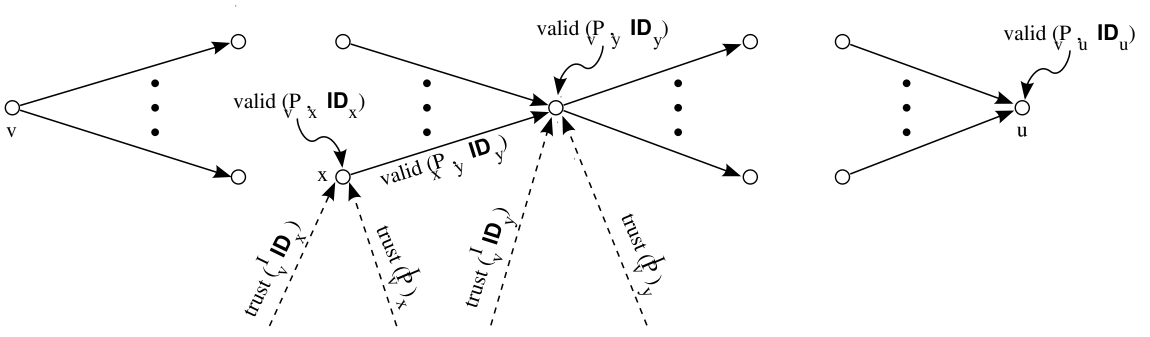


Figure 1: Validity computation on multiple type-I certificate reception

$cert^I G$ or $cert^{II} G$ is a directed graph in which a directed branch (x, y) between nodes x and y represents a type-I or PK-based type-II certificate issued to y by x, respectively. For each node, the inbound branches thus represent the certificates issued to this node by other nodes, and the outbound branches represent the certificates issued by this node to the others. With respect to a given node, the inbound nodes are those from which the inbound branches originate, and the outbound nodes are those to which the outbound branches lead. Two nodes are adjacent if they are connected by a branch and are considered to be close to each other if they are connected by a relatively short directed path.

As each node in a network stores only locally issued and received type-I and type-II certificates, the certificate graphs $cert^I G$ and $cert^{II} G$ needed for computing the public key validities are relative to a number of nodes in a dynamic network that can exchange the locally stored information regarding the certificates, including the verifying node v and the node to be identified u. In the basic scenario, the certificate graphs are derived from v and u only, and, a more general scenario, a number of intermediate nodes, currently present in the network, can be included too. For example, the nodes involved may include v, u, and the one-hop neighbors of v. The certificate graphs are then formed by aggregating all the certificates issued and received by these nodes.

To perform indirect type-I identification of the node u on the basis of the certificate graphs $cert^I G$ and $cert^{II} G$, the first objective of the node v is to compute its trust in the binding $P_u \leftrightarrow \underline{ID}_u$, that is, the public key validity $\mathbf{valid}_v(P_u, \underline{ID}_u) = trust_v(P_u \leftrightarrow \underline{ID}_u)$, where the boldface notation is used to distinguish the validity conditioned on certificate graphs. A method for doing this is proposed below. Secondly, if the computed validity turns out to be sufficiently high, then all the type-I certificates involved in the computation of $\mathbf{valid}_v(P_u, \underline{ID}_u)$ need to be verified by v. In the integrated trust model to be described in Section 2.3, the PK-based type-II certificates need to be verified too, either before or after the public key validity computation. Finally, if this verification is successful, then v needs to verify the fresh claim made by u that (ID_u, P_u) and t are authentic at time t. Altogether, a successful indirect type-I identification of u establishes the binding $P_u \leftrightarrow \underline{ID}_u$ at time t with the validity $\mathbf{valid}_v(P_u, \underline{ID}_u)$. The node v can then also issue and send to u the corresponding indirect type-I certificate $^*cert^I_v(P_u, \underline{ID}_u)$, where the star superscript denotes indirect certificates, which are also usually distinguished by lower validities. Previously issued indirect type-I certificates may optionally be included in $cert^I G$.

Given certificate graphs $cert^I G$ and $cert^{II} G$, $\mathbf{valid}_v(P_u, \underline{ID}_u)$ can be computed by using the following generalizations of (3) and (4), respectively, which relate to the general case of a generic node y with multiple received type-I certificates (see Fig. 1):

$$1 - \mathbf{valid}_v(P_y, \underline{ID}_y) = \prod_x \left(1 - valid_x(P_y, \underline{ID}_y) \cdot trust^I_v(P_x, \underline{ID}_x)\right), \tag{9}$$

$$trust^I_v(P_x, \underline{ID}_x) = \max\left(trust^I_v(P_x),\ trust^I_v(\underline{ID}_x) \cdot \mathbf{valid}_v(P_x, \underline{ID}_x)\right). \tag{10}$$

In the probabilistic model like the one in [16], (9) corresponds to disjunction (OR) of independent events. These recursive equations reduce the computation of $\mathbf{valid}_v(P_y, \underline{ID}_y)$ for any node $y \neq v$ to the computation of the respective validities for the inbound nodes that issued type-I certificates to y, where the validities are relative to v. For $x = v$, $trust^I_v(P_v, \underline{ID}_v) = 1$ and $\mathbf{valid}_v(P_v, \underline{ID}_v) = 1$. Note that the type-I trusts in an inbound node x relate to v and not to other inbound nodes that issued type-I certificates to x (as usual in the literature).

For any intermediate node $x \neq u$ to be included in the computation, it is thus necessary that at least one of the corresponding type-I trusts $trust^I_v(P_x)$ or $trust^I_v(\underline{ID}_x)$ is nonzero and it is preferable that at least one of them has a high value. Inbound nodes that issued type-I certificates to x are not included if $trust^I_v(P_x) \geq trust^I_v(\underline{ID}_x)$ (in particular, if $trust^I_v(P_x) = 1$), because they are not needed for the recursion (9). For the same reason, the validity for each such node x is not computed. Such nodes, including v, are called the source nodes of $cert^I G$ conditioned on type-I trusts of v. For $\mathbf{valid}_v(P_y, \underline{ID}_y) > 0$ (in particular, for $y = u$), it is thus necessary that there is at least one type-I certificate issued to y with a nonzero validity by another node in $cert^I G$.

If the underlying directed graph $cert^I G$ is acyclic, then the computation of $\mathbf{valid}_v(P_u, \underline{ID}_u)$ can be performed recursively, starting from u, by repeatedly applying (9) and (10) as long as there are other inbound nodes to be included. More generally, if the underlying graph $cert^I G$ is allowed to contain cycles, the computation can be performed iteratively, starting from the initial conditions $\mathbf{valid}_v(P_x, \underline{ID}_x) = 0$, for $x \neq v$, and $\mathbf{valid}_v(P_v, \underline{ID}_v) = 1$, until the convergence of the validities for all the involved nodes is reached. More precisely, (9) and (10) are applied repeatedly for a number of rounds and, in each round, they are applied to all the nodes except v, starting from the outbound nodes of all the source nodes in $cert^I G$ (including v) and then moving on to the outbound nodes of these nodes and so on until all the nodes are reached. For an acyclic graph, only one round will thus suffice, but for a cyclic graph, more than one round is needed for the convergence. The convergence to fixed points of the underlying continuous operator acting as a self-composition on the compact space of the validity values is to be expected. In this computation, only the validities are being updated, while the type-I trusts of v in other nodes involved are fixed and are thus used as parameters. It is preferable that $cert^I G$ is acyclic. For illustration, a number of special cases of practical interest are considered in the sequel.

Case 1: Consider first the case where $cert^I G$ consists only of a single direct or indirect type-I certificate issued to u by v at some previous time. Then we have $\mathbf{valid}_v(P_u, \underline{ID}_u) = valid_v(P_u, \underline{ID}_u)$.

Case 2: More generally, consider a chain of $k \geq 2$ type-I certificates ending up in u defined as a directed path $(u_1, \ldots, u_k, u_{k+1})$ of length k, where $u_{k+1} = u$, such that $trust^I_v(\underline{ID}_{u_1}) \leq trust^I_v(P_{u_1}) > 0$ and

$$trust^I_v(P_{u_1}) \cdot valid_{u_1}(P_{u_2}, \underline{ID}_{u_2}) \cdot \prod_{j=3}^{i} trust^I_v(\underline{ID}_{u_{j-1}}) \cdot valid_{u_{j-1}}(P_{u_j}, \underline{ID}_{u_j}) > \frac{trust^I_v(P_{u_i})}{trust^I_v(\underline{ID}_{u_i})}, \tag{11}$$

for each $2 \leq i \leq k$. Let $cert^I G$ consist of v and this chain. Then, we have

$$\mathbf{valid}_v(P_u, \underline{ID}_u) = trust_v^I(P_{u_1}) \cdot valid_{u_1}(P_{u_2}, \underline{ID}_{u_2}) \cdot \prod_{j=3}^{k+1} trust_v^I(\underline{ID}_{u_{j-1}}) \cdot valid_{u_{j-1}}(P_{u_j}, \underline{ID}_{u_j}). \tag{12}$$

It is readily seen that (11) translates into $trust_v^I(\underline{ID}_{u_i}) \cdot \mathbf{valid}_v(P_{u_i}, \underline{ID}_{u_i}) > trust_v^I(P_{u_i})$, $2 \leq i \leq k$. For the validity to be high, it is thus needed that the validities of all the involved type-I certificates as well as all the involved type-I trusts are high. The type-I trusts include the PK-based type-I trust of v in u_1 and the ID-based type-I trusts of v in $u_2, \ldots, u_k$. It thus follows that for type-I identification of u, a chain of type-I certificates does not have to start from v, but can also start from any other node in which v has a sufficiently high PK-based type-I trust.

Case 3: The trust model introduced can also be applied to classical systems including type-I certificate chains issued by certification authorities. Namely, for the validity corresponding to a considered chain to be equal to 1, it is necessary and sufficient that all type-I certificates have validities equal to 1 as well as that the PK-based type-I trust of v in u_1 and the ID-based type-I trusts of v in $u_2, \ldots, u_k$ are all equal to 1. This practically means that v needs to trust the public key of only the first certification authority in the chain, whereas for the other certification authorities, only the identities need to be trusted (as names of physical entities). Of course, U-nodes in BIONETS are allowed to store and use for their type-I identification type-I certificate chains issued by certification authorities.

Case 4: Consider now a set of distinct chains of type-I certificates all ending up in u, i.e., a parallel connection of chains. Then the desired validity $\mathbf{valid}_v(P_u, \underline{ID}_u)$ can be computed by first computing the validities for the inbound nodes of u by using individual chains as described above and, then, by combining these validities by applying (9) and (10) for $y = u$.

2.3 Integrated Trust Model for Type-I and Type-II Certificate Graphs

In the trust model described in Section 2.2, the type-I trusts are all relative to the verifying node v and are treated as parameters. The nodes for which the PK-based and ID-based type-I trusts are both equal to 0, as the corresponding distrusts or uncertainties are equal to 1, are all excluded from $cert^I G$, because they do not have influence on the overall validity. However, in practice, in addition to single type-II certificates issued by v, it is reasonable to also take into account relatively short chains of PK-based type-II certificates in $cert^{II} G$ leading from v to other nodes in $cert^I G$. In this model, the PK-based type-I trusts of v in other nodes in $cert^I G$ need not be computed directly, but indirectly, on the basis of $cert^{II} G$. More precisely, the indirect PK-based type-I trust of v in x can be computed as the probability that there exist a directed path from v to x in a random $cert^{II} G$ where the directed branches in $cert^{II} G$ are randomly chosen with probabilities equal to the corresponding direct PK-based type-I trusts, respectively. In particular, for a chain (i.e., serial connection) of type-II certificates, the indirect PK-based type-I trust is computed as the product of individual direct PK-based type-I trusts along the chain, and for a parallel connection of chains, the product rule is applied to the complementary type-I trusts associated with the chains to obtain the complementary resulting type-I trust, where the complement of a trust value is defined as 1 minus this value. In order to distinguish the zero PK-based type-I trust due to the maximal distrust (where a branch is effectively present) from that due to the maximal uncertainty (where a branch is absent), one can first remove from $cert^{II}G$ all the nodes having at least one inbound branch with the assigned zero trust value.

In the trust-I model described as above, the underlying implicit assumption is that the PK-based type-I trusts are transitive. Namely, if the individual trusts along a chain are equal to 1, so is the resulting trust. Of course, this is only an approximation, and the exact model would require higher order trusts, i.e., recommendation trusts, to be multiplied together as the conditional probabilities. For example, for a chain of two branches, the second branch corresponds to type-I trust, but the first branch should correspond to the trust (recommendation) for assigning type-I trusts. However, the resulting trust model would be too complicated for practical applications. Since the PK-based type-I trusts are more objective than the ID-based ones, it is expected that the transitivity is a more realistic assumption for PK-based than ID-based type-I trusts. This is why it is assumed that $cert^{II}G$ contains only PK-based type-II certificates, whereas the ID-based ones need to be assigned directly, by v.

In a more realistic (i.e., less optimistic) type-I trust model, one can also assign global trust values to individual nodes in $cert^{II}G$ according to the reputation system, for particular services or averaged over the services. In a soft model, one can assign the probabilities to individual nodes as their reputation values and then compute the indirect type-I trusts in a similar way as above, by assuming that the nodes in $cert^{II}G$ (except a verifying node v and a node u to be identified) are also randomly chosen, according to these probabilities. In order to avoid low type-I trust values due to the product rule applied, the reputation values can previously be nonlinearly transformed by an expanding or quantizing function (e.g., the values bigger than a threshold are mapped to 1 and the values smaller than another threshold are mapped to 0). The reputation values can be computed by using designated agents and a distributed hash table (DHT) as in [1]. Other, similar ways of integrating the node reputation values may also be envisaged.

Consequently, in the proposed integrated trust model based on $cert^{I}G$ and $cert^{II}G$, the public key validities are computed as in Section 2.2 and the required PK-based type-I trusts are computed on the basis of $cert^{II}G$ and the reputation system as described in this section. More precisely, one can only consider an acyclic subgraph of $cert^{II}G$ containing only relatively short chains from v to other nodes in $cert^{I}G$ with the associated high values of PK-based type-I trusts. The used ID-based type-I trusts are directly assigned by v.

The same model can also be applied if $cert^{I}G$ is allowed to contain previously issued indirect type-I certificates. However, it should be pointed out that the validity corresponding to each such indirect type-I certificate is computed on the basis of the type-I trusts of the node that issued this certificate in the involved nodes, whereas in the exact trust model, the type-I trusts of the verifying node are required. Accordingly, using indirect type-I certificates is only an approximation.

Approximate computation of public key validities on the basis of significant public key certificate chains and verification of public key certificates and certificate chains are addressed in Appendix A.

3 Authentication

This chapter discusses in more detail the issue of authenticating the entities that compose a BIONETS network using the physical and logical identities described in Chapter 2, in particular how U-nodes can authenticate each other (Section 3.1), services (Section 3.2) and a proposal on mutual authentication between

U-nodes and the much less powerful T-nodes (Section 3.3, with further details in Appendix B), for which a complete public-key cryptosystem cannot be implemented.

3.1 Authentication of U-Nodes

Normal authentication of U-nodes in BIONETS is performed by the process of type-II identification, described in Section 2.1.2. Type-II identification binds a public key to a claimed identity thus ensuring that a node identity ID_u authentically corresponds to the used public key P_u at the time of identification. Consequently, it proves that the claimed ID_u is authentic for the used P_u. A node u is thus authenticated under the effective node name (ID_u, P_u), where instead of P_u, which may be long, a hash value of P_u suffices. In other words, u is authenticated as an entity that repeatedly uses (ID_u, P_u) as its effective name and not only ID_u, because any node can claim the identity ID_u, but only a node knowing the corresponding secret key S_u can claim the name (ID_u, P_u). The node names including the (hashed) public keys are used in the reputation system and, as such, do not require type-I identification of nodes. Type-II identification can be performed whenever there is a communication link between two nodes.

In addition, U-nodes in BIONETS can also be authenticated by the process of type-I identification, described in Section 2.1.2. Type-I identification binds a public key to a physical identity thus ensuring that a node using the public key P_u at the time of identification possesses the physical properties $\underline{ID}_u$ corresponding to the identity ID_u. In particular, if ID_u is a unique name of a physical entity, then type-I identification ensures that this physical entity uses the corresponding secret key S_u at the time of identification. Consequently, it proves that the physical properties in $\underline{ID}_u$ are authentic for the used P_u. Type-I identification requires the bindings $\underline{ID}_u \leftrightarrow ID_u$ and $ID_u \leftrightarrow P_u$ to be verified either directly or indirectly, by using previously issued type-I (i.e., public key) certificates and type-I trusts of the verifying node in nodes that issued these certificates, as described in Sections 2.1-2.3. At least one of these type-I certificates needs to have been issued to u and the type-I trusts in all the involved intermediate nodes need to be relatively high. However, type-I trusts do not have to be ID-based, but can be PK-based instead, and the PK-based type-I trust in a node with a given public key can be estimated objectively by observing the behavior of this node in the past with respect to issuing valid type-I certificates. For this purpose, it is sufficient to authenticate the nodes by type-II identification only, because what is important are the public keys and not necessarily the physical identities. In the basic model, the PK-based type-I trusts used relate directly to the verifying node, whereas, in the integrated model, they can be obtained indirectly, by using type-I trusts of other nodes possibly together with the node reputation values. If the required type-I trusts are not sufficiently high for the resulting public key validity to be high, then the type-I identification cannot be performed successfully.

3.2 Authentication of Services

Authentication of services in BIONETS is a delicate issue since the services are expected to be very dynamic as well as to frequently migrate from one U-node to another. Consequently, U-nodes need to have confidence that the services they get are authentic, i.e., in the same form as generated. Of course, for some services one may argue that the authentication is essentially provided by the quality of the services. However, in general, services need to be authenticated in order to detect if unauthorized users inserted malicious or fake code or data. Authentication can be achieved by using digital signatures based on public key cryptosystems. To this end, the name structure of a service should include the underlying public key that is used for authentication. In this way, the confidence in dynamic services is based on the trust in the persistent public keys used for their authentication, where authorized users are defined as entities in possession of the secret keys corresponding to the service name. Authorized users are allowed to perform service changes, and authentication then consists in verifying that a service with a given name is in its original form as generated by authorized users. Intrinsically, cheating by authorized users cannot be detected.

More precisely, at the time when a service s is first generated, it is assigned a pair of public and secret keys, (P_s, S_s). It is proposed that the service name has a structure $(ID_s, h(P_s), v_s, n_s)$, where ID_s is a semantic description of s, v_s is an optional service version number, n_s is an optional, possibly hierarchically structured, description of a service part or element for distributed implementations, and P_s is the underlying public key, which in practice is kept in the form of a hash value $h(P_s)$. It is envisaged that v_s should be used only to mark significant changes in a service, which itself is expected to change frequently in BIONETS. The part of the name $(ID_s, h(P_s))$ should identify the service in a globally unique way and should not change during the life cycle of the service, despite the service changes during its life cycle.

A service element to be authenticated contains the name, the code or data to be authenticated, and the signature metadata consisting of a digital signature of the code or data (including the service name) along with the public key to be used for verifying the signature. In particular, only service elements considered to be sensitive are authenticated, whereas the others may not be digitally signed. It is important to emphasize that if $h(P_s)$ were not part of the service name, then an attacker could change the code or data, use its own secret key for signing, and then modify the metadata, without changing the service name. If $h(P_s)$ is part of the service name and if this name is known to be authentic, then the service is authenticated by verifying the signature from the metadata. The trust in the service (see Section 4.2) then derives from the trust in the public key from the name of the service. For this reason, it is also possible to use the same public key in the name of different services, in which case the trusts associated with these services can be aggregated.

The digital signature is generated by a physical entity or device (e.g., a U-node in BIONETS) where the service element is produced or changed, by using the secret key for signing, which is securely stored in this device. In particular, the signature secret key may be S_s and the signature verification key may be P_s. It should be noted that when a service element migrates from one node to another, then the secret key S_s should also be allowed to migrate, in a protected way, to the new node. To this end, the new node needs to be authenticated and sufficiently trusted, with respect to the trust and reputation system. However, if the number of nodes storing S_s becomes large, then the security of the system may be compromised. It is then preferable that, after the migration of a service element, S_s is removed from the memory of the old node, and this can be ensured by software mechanisms. Storing the secret keys for different services in a distributed hash table may also be possible.

In practice, especially if the same public key P_s is used in the name of different services, it is desirable not to use S_s for signing, but a different secret key instead. The corresponding public key for verification is then put in signature metadata along with the signature, but should preserve the underlying trust in P_s. In fact, it would be preferable to use many such secret/public key pairs for authentication, in order to limit the damage if some of them are compromised. Negative effect of a compromised secret key on the signatures previously generated by this secret key can be reduced by using forward-secure digital signatures. A solution here proposed consists in using a hierarchical structure of public keys with P_s as the root public key, where each public key is certified by one or more preceding public keys in the structure. This means that the secret/public key pairs can be associated with individual nodes. They may be the same as the secret/public key pairs used for node authentication, except possibly for the root public/secret key pair, which may depend on the service itself. When a service element passes from one node to another, the old node issues a certificate for the public key of the new node signed by the secret key corresponding to its public key, provided that there exists a sufficient level of trust in the authenticated new node. Then the signature metadata for the new node includes a certificate chain from the root public key P_s to the current public key associated with this node. The signature verification includes verifying this certificate chain. It should be noted that these public key certificates are not related to node identities, but only to public keys and issuance times. More precisely, a public key certificate for P_y signed by S_x is a signature

$$cert_{P_x}(P_y) = sign_{S_x}(P_x, P_y, ID_s, h(P_s), t, t'), \tag{13}$$

where t is the issuance time and t' is an optional expiry time. The service version number v_s and the service element description n_s may also be included in the certificate, especially if they are a part of the service name. This certificate represents authorization by P_x for P_y to be used for authentication, given a service with the name $(ID_s, h(P_s))$. (Recall that a certificate also contains the data that are signed.) To minimize the negative effect of potential cheating by already authorized users, the difference between the expiry and issuance times could be relatively short.

The proposed solution is robust in the following sense. If a secret/public key pair is compromised in this hierarchical structure, then only the certificate chains including the compromised secret/public key pair are compromised. In other words, any secret/public key pair that can be certified by at least one chain not including the compromised secret/public key pair remains to be secure. Any part of any chain can be replaced by a new certificate directly certified by the initial public key of that part. In this way, the length of certificate chains can be reduced, if desired. The compromised public keys can be put on locally stored black lists.

Consequently, the trust in a given service also implicitly depends on the trust in the nodes authorized to perform service changes by valid certificates or certificate chains, and all these trusts dynamically change in the underlying trust and reputation systems for nodes and services.

3.3 Authentication between T-Nodes and U-Nodes

In BIONETS, it is adopted that T-nodes cannot communicate to each other, so that in authentication protocols they need to directly communicate to U-nodes. Also, the majority of T-nodes have limited computational and storage capabilities, so that they cannot implement complex public key cryptosystem operations for authentication. This implies that such T-nodes need to be authenticated by using challenge-response protocols based on symmetric key cryptosystems. However, such protocols require pre-shared secret keys between T-nodes and U-nodes, while no initial setup schemes are allowed to involve U-nodes, which should behave in a completely self-organized manner.

A solution that we propose is to make use of dedicated T-nodes, with higher computational and storage capabilities, which can implement authentication protocols based on public key cryptosystems. We may use one or a number of such nodes for a group of related T-nodes. Each T-node t, except dedicated T-nodes, is assigned a pair (ID_t, S_t), where ID_t is a locally unique identity and S_t is a secret key corresponding to a symmetric key cryptosystem (e.g., a block cipher), and this pair is locally stored in t during the setup phase. To reduce the storage requirements for dedicated T-nodes, S_t can be derived from ID_t and the master key K via a key derivation function F (e.g., based on a cryptographic hash function), i.e., $S_t = F(K, ID_t)$. During the setup, the master key K is stored in a dedicated T-node along with all the identities of the associated T-nodes. Dedicated T-nodes thus play the role of trusted third parties with respect to T-nodes, in addition to their usual role as sensors.

For authentication purposes, dedicated T-nodes are treated as U-nodes. Accordingly, each dedicated T-node locally generates a pair of public and secret keys according to an adopted public key cryptosystem. These keys may also depend on the service to be provided. Since a dedicated T-node is a stand-alone device, its identity is desirable to be physically verifiable in a sense that it is stored and transmitted in a tamper-resistant manner (e.g., as a globally unique MAC stored in ROM). The tamper resistance makes it possible to verify this ID by U-nodes in a remote way, without any direct physical contact with a dedicated T-node. In this way, U-nodes can easily issue type-I certificates to dedicated T-nodes, with a high validity. On the other hand, dedicated T-nodes can issue type-I certificates to U-nodes if the physical identifiers of U-nodes can be verified in an automatized way. Dedicated T-nodes can issue type-II certificates for PK-based type-I trusts in U-nodes, in the way described in Section 2.1.4, but they do not have ID-based type-I trusts in U-nodes which are subjective in nature. This simplifies the computation of the public key validities, described in Section 2.2. Dedicated T-nodes can also compute PK-based type-I trusts indirectly by using PK-based type-II certificates issued by U-nodes, as explained in Section 2.3. Similarly, U-nodes compute PK-based type-I trusts in dedicated T-nodes, whereas it can be assumed that the ID-based type-I trust of U-nodes in dedicated T-nodes is equal to 1. Consequently, U-nodes and dedicated T-nodes can be authenticated to each other by type-I or type-II identification.

The authentication protocol is original with respect to previous work as it requires neither any communication between dedicated T-nodes and other T-nodes nor any initial setup scheme involving U-nodes. If S_t is used directly in a challenge-response protocol, then it needs to be distributed to U-nodes and, hence, successful authentication would require trust in U-nodes not to reveal this secret key to other U-nodes or to set up fake T-nodes. Accordingly, a more robust solution would be to use one-time dynamic secret key derived from S_t and a fresh sequence number i, generated by a counter, $S_t' = G(S_t, ID_u, P_u, i)$, by using a key derivation function G, which may be the same as F. The dynamic secret key is bound to the name (ID_u, P_u) of a U-node u, in order to prevent replay attacks described below. It is also bound to the name ID_t of a T-node t through S_t. This dynamic key can be generated by a T-node when needed and does not have to be stored. It plays a role of a one-time secret key for authentication. The secret key S_t is thus never distributed to U-nodes. The dynamic secret keys are distributed to U-nodes by dedicated T-nodes in an encrypted form, by using the public keys of U-nodes and the corresponding public key encryption functions. Dedicated T-nodes are trusted to perform what is required by the protocol and to keep their master keys secret. Details of the authentication protocol are presented in the Appendix B.

The security of the proposed protocol is now examined with respect to a number of attack scenarios. Since the challenge-response protocol step requires knowledge of S_t' at the time of authentication, a replay attack, where the sequence number i is repeatedly used for the same ID_t, requires S_t' to be compromised. This may happen if a U-node sets up a fake T-node supplied with a previously generated and received

S'_t, which then claims the identity ID_t with the same sequence number i. However, since S'_t is bound to (ID_u, P_u) and U-node is each time freshly authenticated to t_d by type-II or type-I identification, it follows that the fake T-node cannot be authenticated to the U-nodes other than the one that originally received S'_t. On the other hand, using a false dedicated T-node supplied with the same S'_t is prevented, because t_d is each time freshly authenticated to the U-node. For the same reasons, if a T-node repeats the sequence number (e.g., due to implementation error or induced fault attack), then the same S'_t can only be used by the same U-node already authenticated to this T-node and not by the other U-nodes in possession of the compromised S'_t.

In any case, to prevent possibly unforeseen replay attacks, it is also feasible that t_d stores the last value of the sequence number for each T-node under its control. Alternatively, instead of the sequence numbers generated by T-nodes, the protocol may also be based on random nonces independently generated by U-nodes and T-nodes when authenticating each other.

The security of the proposed protocol requires that the master keys stored in dedicated T-nodes should not be compromised. To this end, dedicated T-nodes should be implemented as tamper-resistant devices with the property that the master secret keys stored are destroyed if a dedicated T-node is compromised. A more robust solution for a given group of T-nodes would be to simultaneously employ a number of dedicated T-nodes with different master keys and then compute S_t as the bitwise XOR of the individual keys, i.e., as $S_t = \bigoplus_j F(K_j, ID_t)$, where $F(K_j, ID_t)$ are computed by individual dedicated T-nodes. However, the problem is then how to compute the XOR sum S_t and the resulting dynamic keys $S'_t = G(S_t, ID_u, P_u, i)$, since dedicated T-nodes are not allowed to communicate to each other. Of course, a solution would be to relax the constraints and allow the communication between (a small number of) dedicated T-nodes and then use multiparty computation protocols to securely compute S_t and S'_t in a distributed way. In this case, a U-node may communicate to any of the dedicated T-nodes in order to get the desired dynamic key.

Another approach would be to avoid the usage of dynamic keys and thus use S_t directly for authentication. The computation of S_t can then be done by U-nodes, which need to communicate to all dedicated T-nodes, but successful authentication then requires trust in U-nodes as explained above.

4 Reputation

In a potentially non-cooperative environment, system stability must be ensured by a reputation management scheme that work throughout the lifecycle of its components. As explained in Chapter 2, type-I trust values can be determined also on the basis of reputation values, which are in turn based on type-II names. A sound reputation management scheme is therefore crucial not only for system stability, but also for security purposes.

In this Chapter we complete the description of the reputation system by addressing some issues that were not explicitly covered in previous deliverables [1, 2]. In Section 4.1 we address the problem of maintaining a node's reputation when it moves to a new disconnected island by leveraging the previously described embedding of reputation values into trust information. In Section 4.2 we extend the reputation scheme to the BIONETS service lifecycle by introducing the management of consistent reputation values for service cells composing a service individual, and by discussing the maintenance of previous reputation values for mutating services.

4.1 Reputation to Complement Authentication: An Approach to Correlate Reputation

In Section 2.1, we discuss the use of the global reputation value to compute the type-I trust on public keys rather than on the physical identities of nodes. The novelty of this approach is that it grants privileges to nodes who have behaved consistently in the past by exposing the same identifier for multiple services. This model is the basis to enable the introduction in the trust chain of nodes that have not any initial personal acquaintances to rely on or to contribute for services.

In a nutshell, the trust mechanism can be modified to bind the reputation of a node with the quality of the services it provides, rather than with its capability to authenticate nodes. This new correlation introduces the notion of *risk* for a transaction when authentication cannot be accomplished and it is the basis of our approach to enable nodes to carry their reputation in different contexts and mobile communities.

In this section, we define a token-based mechanism to enable nodes to *maintain* the history of past transactions for different contexts [17]. This solution moves the burden of storing personal information to the nodes as it is their interest to trace and maintain their reputation. The token is used to correlate the reputation values earned in different contexts and it gives a first view of the node's willingness to cooperate. In Appendix C.1 an experimental setup is discussed and the model is validated.

4.1.1 Node Identity and Structure of the Scheme

Each node is associated with an identity ID_u as defined in Section 2.1. A node can use multiple identities, i.e., pseudonyms, each one for a specific scope and service, which might be traced back to the same identifier. The reputation is formed by considering past transactions of the nodes with other peers and each node maintains a local opinion of the trustworthiness of the peers it interacts with.

In our model we do not make any specific assumption for the architecture to maintain the reputation values for every single node as the reader can refer to [1, 2] for the details on how to implement such a system.

When a node joins the system with a new pseudonym for providing a specific service, its initial reputation value is null (0), which corresponds to the value of an uncooperative node. Thus, a node has no incentives to create multiple pseudonyms for the same service. Nevertheless, to mitigate collusion attacks and bad mouthing attacks, we assume that a node can generate only one valid self-signed certificate for each type of service. This can be verified by checking the credentials of the nodes and the original identity that generates the pseudonyms. In case the node is found to be untrustworthy we set $trust_v^I(P_x) = 0$ (see Section 2.1.4 and (8)).

Nodes periodically receive information on how they behaved in the system. This report is added to the *trust token*, which is personal and stores the reputation values associated to the activity of a node for a specific service, as shown in Fig. 2. The token consists of *n* entries, as many as the types of services. This is important to judge over more samples of the node's behaviours and to detect nodes that have inconsistent behaviour for different services. Each entry contains the Identity of the node (Id), the type of service (s-Id), a timestamp (T) that indicates when the report has been updated, the reputation value (R) of the node and the quality value (Q) associated to the reported score. The report is digitally signed (Sig) by using the private key S_u of the node to bind the pseudonyms and the real identity of the node.

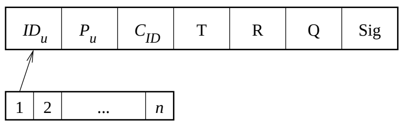


Figure 2: The trust token is formed by n entries and each entry contains: 1) Name (ID_u, P_u) of the node, 2) Cluster Identity (C_{ID}), 3) Timestamp (T) that indicates when the reputation value has been issued, 4) Reputation (R) value of the node, 5) Quality (Q) value associated to the entry, 6) Node digital signature (Sig) on the report.

The node name (ID_u, P_u) or $(ID_u, h(P_u))$, where $h(P_u)$ is a hashed value of P_u, is used to bind a report to the node identity ID_u together with its public key P_u used within a specific context. This binding ensures that the node does not transfer the token to other entities. For instance, let us suppose that a node has built a high reputation value and it creates several identities for the same service, it could decide to use these multiple identities to behave maliciously so that its attack has higher impact on the system performances.

The cluster identity (C_{ID}) specifies which cluster supports the node's reputation. This value allows other nodes to verify the token authenticity in case the system is operated by means of shared cryptography (see Section 4.1.3 for alternative approaches).

The timestamp (T) specifies when the report has been updated. The timestamp is required to verify the identity of a specific node at a given time as defined in 2.1.

The reputation value (R) gives an estimate of the behaviour of the node for a specific service. It is updated after each transaction the node participates in. The reported score is the global trust value the node has at the time when the token-entry is updated.

A quality value (Q) is associated to the reputation. It represents the confidence that the nodes have in their reports. Giving incorrect reports can decrease the credibility of a node. So, nodes can lower the quality value for reported reputations if they are not sure, therefore risking less loss of credibility in case their judgment is incorrect: this happens if there few samples to estimate the reputation value of a node or the node behaves inconsistently in the system.

The quality value is defined as the confidence level that the actual mean calculated for the reputation of a peer lies within the confidence interval. The equations for the estimation of the quality value can be found in [18].

The digital signature is done on the hash of the report. The node signs the report to provide integrity and a proof of participation of the node to the system. The signature is verified by any other node to check the correctness of the information stored in the token before a report is considered to be valid. This procedure is required to avoid fake reports from the nodes and possible modification of the reputation values.

4.1.2 Reputation Correlation

The transferability of the reputation value among different contexts is not discussed here; an effort in the direction of computing the similarity of services based on their semantic description and composition has been proposed in [19].

In order to support mobility in virtual communities, we introduce the concept of *cluster credibility*, i.e., the confidence that a node has in the reporting capability of an "island" of connected U-Nodes. The use of cluster credibility can be modified to handle the correlation of the reputation in different contexts. We extend the computation of the initial reputation value by using this credibility factor to weigh the reputation values of an entrant node in a new cluster.

The behaviour of nodes entering a new community will impact on the credibility of the cluster that issues reports for the nodes. If a cluster gives wrong reports about peers, its credibility rating is decreased and its subsequent reports have a reduced impact on the reputation of another entrant peer coming from the same community. Similarly, if a cluster's report is consistently good, i.e., in agreement with the behaviour of the nodes inside the new cluster, its credibility rating goes up.

The cluster credibility has an initial value of 0.5 and is computed upon the agreement of the old reported reputation values and the behaviour of any new entrant node coming from the same cluster. The computation is similar to the confidence of a node for ROCQ [20]:

$$C_{mc}^{k+1} = \begin{cases} C_{mc}^{k} + \frac{(1-C_{mc}^{k}Q_{cj})}{2}\left(1 - \frac{|R_{cj}-O_j^{avg}|}{s_{mj}}\right) & \text{if } |R_{cj} - O_j^{avg}| < s_{mj} \\ C_{mc}^{k} - \frac{C_{mc}^{k}Q_{cj}}{2}\left(1 - \frac{s_{mj}}{|R_{cj}-O_j^{avg}|}\right) & \text{otherwise,} \end{cases} \tag{14}$$

where C_{mc}^{k} is the credibility of cluster c after k reports to peer m, O_j^{avg} is the opinion being currently reported on the new entrant j, Q_{cj} is the quality value of the cluster on the previous reported reputation value R_{cj} for j and s_{mj} is the standard deviation of all the reported opinions about peer j.

The cluster credibility ratings are based on first-hand experience only and they are not shared with other peers to avoid the recursive problem of trusting nodes.

When a new node joins the community, the designated agents [1] verify the integrity and the signatures of the token and compute the new reputation value which is stored locally for future use inside the cluster. These agents form the *first* reputation value by aging and weighting the information the token contains with the credibility of the clusters that have issued the entries. The credibility of the cluster is not shared inside a community because it would introduce extra signaling. It is important noticing that a node can erase the entries contained inside the token, but these deleted entries count as negative transactions in the computation of the reputation value, as discussed earlier for the usage of the timestamp.

4.1.3 Security Analysis

Appendix C.1 presents simulation results to validate the token-based approach. The mechanism improves the proportion of correct decisions taken by nodes in all scenarios we have analyzed. Specifically, it enables to account for transactions in other communities to detect misbehaving nodes timely. The fact that nodes are rewarded for their good behaviour fosters cooperation which is a basic property that self-organizing system must have to function properly.

In this Section we assume that an identification scheme is already in place (as per Chapter 3), so that the physical or logical binding of the reputation token's identifier with the node is already taken care of.

A potential criticism to the token-based approach consists of the possibility for the nodes to fake their old reports to hide their malicious behaviour or to simply sell/lend their tokens to other nodes. We tackle these problems by imposing that the score for a node is bound to the identity of the nomadic node and it is digitally signed by the community. Each message is released at periodic intervals and, in new clusters, the aggregated reputation value must reflect the reporting time to age reputation in accordance. If a record is not present in the token or the node does not disclose part of the token, designated agents consider the missed information as the node misbehaved during that period.

From a privacy and anonymity point of view, the token-based extension requires nodes to reveal which communities the node has joined in the past, thus, the location of the node is fully traceable. However, a node has completely control over its token and it can decide which information wants to reveal and to which community.

We now analyze the security solutions to mitigate the impact of impersonation and bad mouthing attacks. The token-based scheme uses digital signatures to ensure integrity and correctness of the reports. The signature must be recognized to belong to the cluster otherwise the report is not valid. We assume the existence of an off-line certification authority that issues certificates to the nodes, associated with a pair of keys. This is required by any reputation management scheme to avoid non-repudiation of an opinion.

Clusterheads can digitally sign the reports simply, but this solution has two main drawbacks: 1) the leaders in a cluster are many and they do not share the same key pair; 2) the verifiers must know the public key of the signer, i.e., each mobile node should store the public keys of all possible signers present in the system. The storage is not a big issues for mobile devices; in fact a typical public key has a size that ranges from 512 bits to 2048 bits if no elliptic cryptography is used, thus, in the worst case (2048 bits) 1 MB is sufficient to store 4096 keys.

The main issue of this solution consists of the form of the signature: if clusterheads use their own private key to sign the report, this report is associated to the signer and not to the cluster, i.e., what we want to achieve. In this setting, malicious nodes can collude and generate valid false reports unless the verifier has the complete list of authorized clusterheads, which is not feasible.

To implement a signature on behalf of the group and to guarantee anonymity of the signer, two possible schemes are *group signature* [21] and *ring signature* [22]. In the former, an authority generates the private signing keys and distributes them to the members of the group which uses its private signing key to generate the signature; a verification key, common for the group, is used to validate the signature. In *ring signature* a clusterhead, responsible for the report, creates an ad-hoc ring signature composed by other cluster entities *without* their approval or their aid. This mechanism preserves anonymity of the signer, but there is no control on the formation of the ad hoc group and the verifier can hardly know the members of a group authorized to sign a message; moreover, these approaches assume the existence of a trusted certification authority; however, this is not required to be online, and only needs to be invoked at appropriate times.

An alternative solution can be the implementation of *threshold cryptography* [23], requiring a minimum number of clusterheads to sign the same item and ensuring that a smaller number of colluding members has a negligible probability of generating a valid signature for a different token. Two schemes that implement threshold cryptography in ad hoc mobile networks are proposed in [24, 25]; in particular, the latter proposes a dynamic scheme to increase the threshold to produce a valid signature in order to react to network condition changes and to the number of possibly malicious nodes. The main drawback of this solution is the message complexity which makes it not recommendable in congested networks.

To create a strong relationship between the signer and the cluster, we propose to use *Id-based cryptography* [26]. In Id-based cryptography the public key is an identifier and the private key can only be generated from the public key by a trusted authority. In our setting, the identifier is a tuple that contains the node identifier and the cluster id (ID_u, P_u, C_{ID}: see Fig. 2), to ensure that a signature has been issued by a clusterhead. In a dynamic environment, we might want to give the opportunity to clusterheads to outsource the responsibility to sign reports of other nodes. To serve our goal, we can exploit Id-based signatures schemes organized in a hierarchical structure as proposed in [27].

In the simplest case scenario, reports may also be signed by the involved designated reputation agents, thereby eliminating the need for cluster identities and group signatures. The main drawback for this choice would be the need of a mechanism to prevent agents from cheating, for instance by a reputation-based reward mechanism for good behavior.

4.2 Service Reputation Mechanisms

The high dynamicity and mobility of nodes, code and services in the BIONETS framework requires that services are carefully considered before they are executed in a U-node. In this section we extend the contribution of previous deliverables [1, 2] to the service reputation case. The proposed mechanism is robust with respect to the entrance of malicious U-nodes with biased opinions. The main issue is the need to consider service composability: a mechanism for propagation of reputation values from service individuals to their component hierarchy and vice-versa must be defined. Another important issue, already considered in [2] for the similar case of T-node reputation, is the asymmetry between recommending and recommended entities: a single malicious or selfish node can try to inject (and later support) many bad services in the system, or many malicious nodes can try to support a single bad service for a better probability of success. Additional results on this problem have been reported in Appendix C.2.

We focus on the case of migrating services. In fact, a fixed service abstracts the capabilities of a specific U-node, and in that case the provider responds with its own reputation, therefore the already described mechanisms for node reputation maintenance shall account for this case. We will therefore assume that a U-node migrates a service in order to use it. In this case, a protocol for sharing other nodes' experiences with the same service is required. In this context, the reputation of a *service* helps a U-node select which migrating service to host and execute when provided with multiple (semantically equivalent) choices. Section 4.2.1 deals with the case of service cells, while in section 4.2.2 we discuss the modifications needed when a service individual is built by composition of other services. A proposal regarding the reputation inheritance of mutated services (or new releases of existing services) is introduced in section 4.2.3.

4.2.1 Reputation Maintenance of Service Cells

For the purposes of this Section, we assume that a service cell s can be authenticated as described in Sec. 3.2; in particular, its name contains a semantic description ID_s of its functionality. Binding reputation

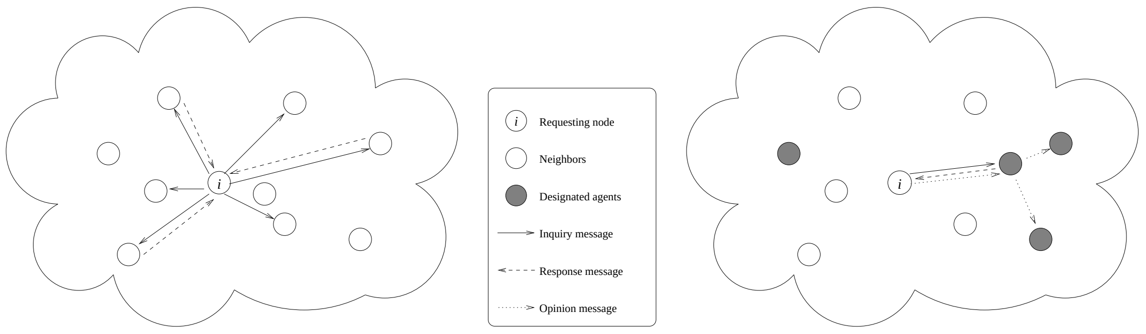


Figure 3: The two basic approaches in service reputation management. Left: a U-node requests other U-node opinions and aggregates them; only some neighbors might receive the query (solid arrows), and even fewer of them may respond (dashed arrows). Right: a U-node queries a designated agent (solid arrow) and receives an aggregated answer (dashed arrow); upon service execution, the requesting node completes the protocol by proactively sending its own opinion, which can be spread via a distributed protocol to other designated agents.

values to the complete service name is important to avoid cases in which a service has different behaviors with respect to different semantics.

We assume that the reputation mechanisms for feedback and aggregation described in [1, 2] are in place. The number of designated agents used to aggregate opinion values depends on system connectivity (how likely is it that a disconnected cloud will contain a designated agent for reputation aggregation?), on the dynamicity of the service migration and execution mechanism, on the acceptable level of message overhead before service execution. In general, the same agents will be used for service and node reputation.

As described in [2], the exact mechanism for opinion gathering and data aggregation depends on the availability of reputation agents and on the required freshness of information. The two extreme cases are represented in Fig. 3.

Assume that U-node i needs to execute a service with a specified semantic, and that it has been provided with a set of n alternative services $s_1, \dots, s_n$. If $n > 1$, the reputation mechanism can be used in order to migrate and execute the most reliable service; if $n = 1$, a threshold can be set in order to decide whether to execute the service at all.

Distributed case. Assuming that no reputation designated agents are available in its connected cloud (Fig. 3, left), U-node i broadcasts a request for opinions with the service's name. Upon receipt of the request opinion, every U-node i' locates its own record of past experiences with the same service name and, if any, sends an aggregate answer in the form $(O^{\text{avg}}_{i'j}, Q_{i'j})$, where $O^{\text{avg}}_{i'j}$ is the *opinion* value that U-node i' stores about service cell s_j, while $Q_{i'j}$ is a relative confidence measure(a *quality* value). Once a number of such opinions have been collected by U-node i, they are used to compute the service cell's *reputation* value R_{ij} that U-node i will use to take its decision. Based on the comparison among reputation values, U-node i decides to migrate service cell s_j; if no reputation value lies above a predefined threshold, however, the node can decide to supersede. After receiving service s_j and checking its authenticity, U-node i executes it. At the end of the execution, a new opinion $O^k_{ij} \in [0, 1]$ (where k is the number of opinions that U-node i generated about service s_j) is formed and stored about the service, thereby modifying the average opinion and quality values that U-node i stores about that service, and the new *credibility* value $C_{ii'}$ that U-node i will associate to each responding U-node i' in future computations.

Designated-agent approach. If aggregation agents are available, the request for opinions is not broadcast to neighbors, but the service of a designated agent is invoked (Fig. 3, right). The designated agent contains all necessary opinion values, which have been proactively provided by other nodes. After the execution of the chosen service, the requesting U-node will send its own (signed) opinion to the designated agent for later use.

4.2.2 Reputation Maintenance of Service Individuals

The structure of a service individual is tree-like with the individual itself at the root and its components as subtrees; the service structure can be described by the hierarchical field n_s in the service name. In Fig. 4 (left) an example is provided, where service individual s is the combination of three components (f_1, f_2 and f_3), some of which are in turn composed, with service cells at the leaves. On the basis of a reputation system, every component can be chosen among a set of semantically compatible services with the algorithm described in Sec. 4.2.1. Moreover, associating reputation to individuals, rather than to cells alone, enables a node to choose between service individuals with different but semantically equivalent composition, where the overall reputation value of the service individual is either known or recursively computed on the basis of its components.

As previously described, after the execution of s the U-node is required to formulate an opinion O_{is}^k, but we also need to evaluate the behavior of the lower tree levels. Forming an opinion O_{if_j} about each component f_j of service s, however, can be difficult. In general, we can assume that the behavior of some components can be evaluated (and in particular we can always evaluate the outcome of the root individual s), while others provide results whose quality is not measurable by itself. This means, in particular, that if the opinion on the root service individual is low we might not be able to assume that the U-node can isolate responsibilities and identify the faulty components.

Based on the opinions that can be formed, however, we can propagate them down the tree according to the following scheme. First, we try to assign an opinion O_{if}^k to each node f in the tree. If some nodes remain unlabeled because it is not possible to form an opinion about their contribution, we navigate the tree breadth-first, and for each unlabeled node f:

- If at least one sibling of f is labeled with an opinion lower than its parent, then we leave f unlabeled.
- Otherwise (all siblings of f are either unlabeled or have an opinion higher than their parent), we label f with the parent's label (none if the parent was unlabeled).

Fig. 4 (right) provides an example of the opinion propagation mechanism. After some opinions are formed (numbers in grey squares), these opinion are propagated to unlabeled siblings. Node f_{31} remains unlabeled because its sibling f_{32} is already labeled with an opinion lower than its parent's: the proposed algorithm holds f_{32} responsible for f_3's behavior, and concludes that nothing can be deduced about f_{31}. On the other hand, the fact that node f_3 has a higher label than its father s makes us conclude that at least

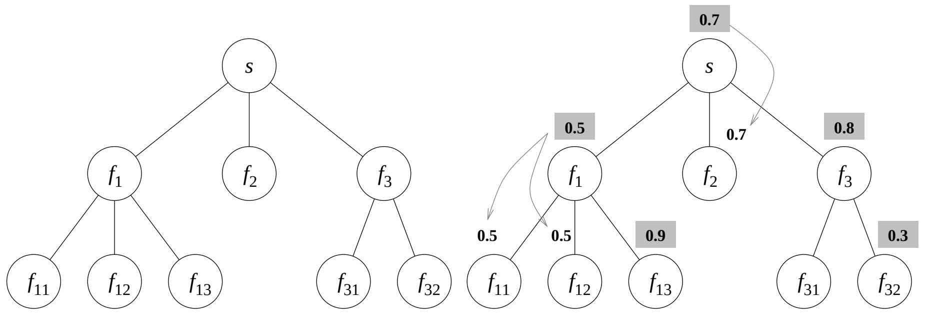


Figure 4: Left: Service composition hierarchy. A service s can be composed by local components f_i, some being in turn composed. Right: Opinion propagation after service execution.

one of the unlabeled siblings is responsible for the overall outcome, and both are precautionarily labeled with their father's label.

The main drawback of this approach is the low reputation that can be achieved by an otherwise good component just by being often associated to a bad one within a composed service. This problem can be overcome by frequently recombining components with the help of a randomization scheme in the component selection phase. Before executing a service, once all component reputations have been computed, U-node i will employ the following heuristic: if all components corresponding to a specific functionality within the service are below a given threshold, then one is chosen at random, with probability inversely proportional to the corresponding average information quality. Let I_j be the set of all U-node indices that have provided information about component f_j. Then let

$$\bar{Q}_j = \frac{\sum_{i' \in I_j} Q_{i'j}}{|I_j|}$$

and let component f_j be chosen with probability proportional to $\bar{Q}_j^{-1}$, disregarding the opinion value. Therefore, unless a good candidate appears for a given function, all components will be used and will have a chance of being evaluated.

As described at the end of Sec. 4.2.1, if a service component is performed by another U-node, the latter will respond with its own node reputation.

4.2.3 Service Mutation

Finally, let us consider the case of evolutionary services, where a service can generate a mutated offspring; the simplest form of mutation is, of course, a new revision of the code. Let s_j mutate into $s_{j'}$; assuming that the service maintains the same semantics, mutation appears in the new service name as a new version label $v_{s_{j'}}$.

When a reputation-aggregation agent receives an inquiry about service $s_{j'}$, it will build the aggregated reputation value using pairs (O, Q), where

$$(O, Q) = \begin{cases} (O^{\text{avg}}_{i'j'}, Q_{i'j'}) & \text{if some node } i' \text{ already has provided an opinion about service } s_{j'} \\ \left(O^{\text{avg}}_{i'j}, \eta(v_{s_j}, v_{s_{j'}}) Q_{i'j}\right) & \text{if we can only find a node } i' \text{ with opinions about the parent service } s_j. \end{cases}$$

The second condition allows the reputation aggregator to use existing opinions about the parent service, but with a lower quality value; the actual lowering is controlled by function $0 \le \eta(v, v') \le 1$ whose value depends on the version labels. The reduced quality value has two important effects. First, if a group of nodes has opinions about the mutated service $s_{j'}$ and another group only knows its parent s_j, the weight of the latter is much reduced when U-node i computes the reputation value $R_{ij'}$, and opinions about the offspring will prevail as expected. Second, because of the low quality value, U-node i' ensures that its credibility $R_{ii'}$ does not change much in the likely case of a wrong opinion.

While each mutated service could be treated as a new service without associated opinions, the proposed phylogenetics-based approach can help reduce the bootstrapping problem which affects most reputation systems. On the other hand, opinion inheritance implies that a bad service evolving to a better one is still penalized because it won't be used. However, if the previously introduced random scheme based on inverse quality is used (see Sec. 4.2.2), then if all service reputations are below threshold, a newly mutated service has a higher chance of being executed because of the low quality (confidence about opinion) value Q.

5 Enhancing Application Level Privacy

In the BIONETS networks, the messages are disseminated according to the *store-carry-and-forward* principle. In particular, the messages are stored and carried by the mobile devices and forwarded to intermediate devices when they have connection (i.e., when they are in vicinity of each other). In BIONETS networks, local information needs to be distributed to a set of nearby destinations based on their interest in the information.

As a motivating example, let us consider a small advertisement application. Using this application, a diligent student participating in each lecture can earn some money. He types an advertisement about helping other students. There is a group of students who participate in some lectures meeting the diligent student. Their handheld devices can download the advertisement, because they are interested in any messages related to the university. They go back to the dormitory and meet students who do not visit the lectures at all. Lazy student's handheld device downloads the advertisement and before the exam they can contact the diligent student.

Without privacy protection, no new technology should spread widely. The privacy of the users must be ensured in BIONETS networks as well. Some of the problems can be mitigated by traditional technologies, but some new problems are introduced by the store-carry-and-forward manner of the BIONETS networks.

The privacy problem can be clarified through the above described example. Let us consider a lecturer who wants to make a list of the students attending the lectures without the consent of the students. This list can be easily constructed if the ownership between the students and their handheld devices are disclosed once, and their handheld devices are traceable.

The privacy of a system and the anonymity of the users can be a problem on different levels of the communication stack. Using a rough partitioning, the problem can be related to the physical layer of the network, the network and transport layers, and the application layer. In the physical layer, the physical characteristics of the transceiver, in the application layer, the application data, and in any layers, especially in the network and transport layers, the identifiers must be hidden or anonymized.

It is essential that the communication is anonymous. Anonymity (or at least pseudonymity) can be easily achieved by the usage of pseudonyms (i.e., temporal identifiers). A more serious and BIONETS network specific application level privacy problem is that the nodes can be identified by their stored messages. If an attacker is able to build a user profile using the exchanged application data, the user becomes traceable even if the communication is completely anonymous. Therefore, new mechanism or adaptation of proposed mechanism is required in BIONETS networks to ensure untraceability of the nodes, namely, to avoid building traceable user profiles.

The main contributions of this work are the following: We characterize and simulate efficient software based attackers that can link different appearances of the same U-node with high probability using only regular handheld devices. We suggest and evaluate an efficient defense strategy, too, which is useful against this attacker without jeopardizing the U-node's main goal, the message collection.

The rest of this section is organized as follows: In Section 5.1, the most important parts of the system and attacker model are described. Then, in Section 5.2, the proposed defense mechanism is presented. The main conclusions of the simulation results are described in Section 5.3. Finally, we summarize our work in Section 5.4.

5.1 System and Attacker Model

It is assumed that there are some categories, and each message belongs to a single category. An interest profile (IP) of a U-node, which is a part of the user profile, is a binary vector representing a list of categories the U-node is interested in. A message belonging to category k is called primary for a U-node if $IP[k] = 1$, the message is secondary otherwise.

The attacker wants to track the target U-node to breach its privacy. To do so, it tries to link the profiles acquired in different times together. If the profiles can be linked correctly, the attack against the privacy is successful.

The communication between the U-nodes can leak some information about the interests of the participants. In this report, attacks based on these leaked information are considered. We assume that the attacker can estimate the following user profile (UP) from a U-node u at time t (this assumption is supported in Appendix D):

$$UP_u(t) = (EIP_u(t), CHM_u(t), IDL_u(t)) \tag{15}$$

The UP consists of the following triple:

- Estimated Interest Profile (EIP) is a binary vector. The value of the vector at the k^{th} position equals to 1 if category k seems to be interesting for U-node u.
- Category Histogram of offered Messages (CHM) shows, for each category, how many messages in the ID list belong to that category.
- IDL is the ID list of offered messages.

Based on this triple, the attacker can define the similarity of the target and a suspected U-node based on the $UP_u(t)$. Using the user profiles of the U-nodes, the attacker can calculate the similarity using an attacker function. We proposed four kind of attacker functions. The detailed attacker model, the description of the attacker functions, their properties and their efficiency are investigated in Appendix D.

5.2 Defense Mechanism

In order to preserve a U-node's privacy, the User Profile (UP) should be obfuscated to distract the attacker. Against an eavesdropping attacker, another solution is to design the message exchange protocol in a way that it ensures that no sensitive information can leak during the communication. We are focusing on developing an obfuscation based mechanism because that can be used if the attacker actively takes part in the communication.

Two simple methods can be used to modify the UP through modifying the Interest Profile (IP) of the U-node. The first one is to *hide* some interesting categories, and claim them as uninteresting. The second one is to *lie* about some uninteresting categories, and claim them as interesting. These techniques can be used at the same time, this is what we call *Hide-and-Lie Strategy* (HLS). The temporarily obfuscated IP is the Temporal Interest Profile or the EIP from the attacker point of view. The EIP can be transient, which means that a new EIP can be generated by every U-node, periodically.

Obviously, the required and offered messages during the message exchange must be synchronized with the EIP: 1) messages relating to hidden categories must be hidden as well, and 2) when a U-node lies about being interested in a given category, it collects and offers messages belonging to that uninteresting category.

Many different HLSs can be envisioned. Different strategies can hide or lie about different categories in different situations. In the following, a simple but rather general solution is given: every U-node generates its EIP from its IP by inverting every category in the IP with a given probability λ. Inverting means indicating an uninteresting category as interesting or vice versa. This parameter λ is the Hide-and-Lie strategy value.

As λ is a probability, it can vary between 0 and 1. The U-nodes which do not use any obfuscation techniques can be modeled as U-nodes using HLS with $\lambda = 0$ as the U-node never modifies its EIP. When λ is 0.5, the mechanism totally randomizes the IP, making the EIP a uniformly distributed random binary vector. It is the best strategy for the privacy sensitive users. While, values of λ greater than 0.5 are useless for the U-nodes, as they make the EIP as traceable as the inverse EIP with $\lambda' = 1 - \lambda$, but the U-nodes collect more uninteresting messages than interesting ones. Consequently, only $0 \leq \lambda \leq 0.5$ are considered.

5.3 Results

The efficiency of different attacker functions and the efficiency of the defense mechanism are investigated by means of simulations. We exhaustively analyzed two representative scenarios presented in Appendix D. Beyond this analysis, we show the differences compared to the other simulated scenarios through analytical investigation. In the two considered scenarios, we investigate the effect of the Hide-and-Lie Strategy on the reached gain and the number of downloaded primary and secondary messages and the maximum memory required to follow the proposed Hide-and-Lie strategy. Because of the space limits, in this section, we only show the main conclusions, however, the statements are supported in Appendix D where the simulation parameters are presented, too.

We found that the efficiency of the attacker functions changes according to the parameters of the model (e.g., one is very efficient if the U-nodes does not protect their privacy at all). However, a common tendency is that if the U-nodes does not protect their privacy, an attacker can trace the U-nodes with high probability using any of the proposed attacker functions. But, if the U-nodes apply the Hide-and-Lie Strategy with high value of λ, none of the attackers is able to distinguish them better, than a naïve attacker which picks up one of the U-nodes by random.

We have measured the achieved gain of the U-nodes. The gain is defined as the ratio of the obtained primary messages and all of the primary messages. The Hide-and-Lie Strategy has two contradictory effects: On the one hand, when the U-nodes happen to hide what they are interested in, they may miss some primary messages to download. On the other hand, when the U-nodes happen to lie being interested in some category, they store-carry-and-forward secondary messages, which increases the data delivery ratio in general as we have shown in [28], and in Deliverable 4.2 [2], and 4.3 [29]. The cumulative effect depends on the system parameters. E.g. in a case when U-nodes are interested only in a small subset of categories and they do not carry secondary messages, they can exchange messages only with small probability. Therefore, the Hide-and-Lie Strategy in some cases can be viewed as a motivation to store-carry-and-forward secondary messages. On the other hand, when the U-nodes have many possibilities to get primary messages, the latter effect has no considerable benefit while the former effect degrades the gain. Surprisingly, the two effects can be balanced as we show in Appendix D.

Even though we did not take into consideration the energy consumption and the memory costs of the communication when we calculated the gain, we collected related information during the simulation. As one can expect, the number of the downloaded secondary messages increases with increasing λ value. However, we found that even if the U-nodes download more and more secondary messages as λ increases, the maximal memory usage does not increase at the same order. Thus, the U-nodes do not need to maintain much larger memories when they want to protect their privacy.

5.4 Final remarks

In this section of the report and in Appendix D, the application level privacy in BIONETS networks has been investigated. In particular, an attacker can build a user profile of a U-node based on what messages the U-node stores and what messages it wants to download. After profiling, the attacker can trace the U-node based on the user profile even if the U-node communicates with the other U-nodes through anonymous links. A system and an attacker model was built and some attacker functions were proposed. A defense mechanism called Hide-and-Lie Strategy against such attacks was proposed, too. This mechanism has a free parameter with which the U-nodes can select between high privacy level and low data-forwarding overload. In our model, we analyzed the efficiency both of the attacks at different parameter values and the proposed defense mechanism. We showed that without any defense mechanism, the U-nodes are traceable, but with the proposed Hide-and-Lie Strategy, the success probability of an attacker can be decreased substantially. The message delivery ratio and the costs at different Hide-and-Lie parameter values are also investigated. We found that in some scenarios, the Hide-and-Lie Strategy can be viewed as a motivation for other U-nodes to carry messages that they are not interested in. Therefore, as a positive side effect, the message delivery ratio is also increased.

6 Conclusion

The BIONETS network model poses a number of challenges in terms of privacy, authentication, anonymity and reputation. Its heterogeneous composition, with nodes having much different CPU capabilities, requires that different protocols for trust and reputation are put in place, and that operations are eventually assisted by dedicated nodes. The introduction of services as first-class entities requires them to be subjected to the trust and reputation protocols as well. In this deliverable, therefore, we have described novel proposals for trust and reputation of all involved parties.

The introduction of type-I (physical) and type-II (logical) identifiers for nodes makes it possible to maintain a suitable level of security and anonymity in the network, and a U-node can maintain a level of reputation when moving to a new disconnected cloud by carrying a reputation token with itself. However, given the data-centric architecture of the system, it is still possible for an attacker to try to track a U-node based on the information it delivers and accepts, even when it maintains anonymity. A defense “Hide-and-Lie” strategy has been proposed allowing U-nodes to select the appropriate tradeoff between privacy level and communication overhead.

However, this document did not finalize all security issues associated to the BIONETS model, and in some cases it purposely did not propose unique solutions. As an example, the choice of a method to certify reputation tokens in Section 4.1 strongly depends on the size, capability and security level of the system, and therefore must be regarded as an implementation choice. The same can be said of reputation protocol details. Other issues, such as anonymity, are probably too complex to give rise to a single framework for their solution, and need to be treated at separate levels, with solutions that depend on the network level and on the type of threat. We believe, however, that the techniques described in this deliverable compose a coherent picture of what can be done in the proposed network model.

A Finding and Verification of Public Key Certificate Chains

Chapter 2 introduces identities and trust models to be used in the BIONETS system. This Appendix provides further details on the collection and verification of certificate chains.

A.1 Finding Public Key Certificate Chains

In the basic approach, v first collects the information regarding the certificate graphs $cert^I G$ and $cert^{II} G$ from the other nodes it can communicate with and then computes the corresponding public key validity $\mathbf{valid}_v(P_u, \underline{ID}_u)$ by applying (9) and (10), according to the trust models described in Sections 2.2 and 2.3. If indirect type-I certificates are chosen to be included in $cert^I G$, then the lengths of the involved chains of direct type-I certificates are effectively longer. In principle, the certificate graphs can be based on all the nodes able to communicate to each other in a current network, via one-hop (or multi-hop) links. The set of these nodes is denoted as G. Note that in this case, apart from G, $cert^I G$ and $cert^{II} G$ also include the nodes that, in the past, issued certificates to the nodes in G and received certificates from the nodes in G. The constraint satisfied by the certificate graphs is that no two nodes in these graphs that do not belong to G are connected by a branch, because the certificates included need to be issued or received by the nodes in G.

It follows that this public key validity is lower-bounded by the product of the validities of the type-I certificates associated with the branches in $cert^I G$ along any certificate chain as a directed path leading from any source node of $cert^I G$ to u, multiplied by the product of the PK-based type-I trust associated with the source node and the ID-based type-I trusts associated with intermediate nodes along the path. Recall that the source nodes are conditioned on the type-I trusts of v, which themselves are given directly (as in Section 2.2) or are computed on the basis of $cert^{II} G$ (as in Section 2.3). The paths with the associated product of branch validities being maximal or close to maximal are called optimal or close to optimal, respectively. Consequently, instead of computing the public key validity, the initial objective of v may be to find a sufficiently large lower bound corresponding to a path in $cert^I G$ leading from any source node to u. If a desirable single path is not found, then v may look for a number of paths and compute the corresponding public key validity accordingly. This essentially means that, instead of considering $cert^I G$ as a whole, v may consider only significant simplified subgraphs of $cert^I G$ which reduce to (preferably distinct) individual paths or their parallel connections. It is important to note that by adding nodes and branches to $cert^I G$, the corresponding public validity may only increase.

Alternatively, finding optimal or close to optimal paths between every pair of nodes in G can also be performed in a distributed way by using distributed graph algorithms, which need to be adapted to deal with the situation where the branches in $cert^I G$ also involve nodes out of G. To this end, it is interesting to study distributed versions of the well-known Floyd-Warshall (all pairs shortest path) algorithm as well as to adapt the distributed algorithm [30] from undirected to directed graphs. Note that for a directed graph with n nodes, the centralized Floyd-Warshall algorithm requires $O(n^3)$ time and $O(n^2)$ space, whereas for an undirected graph with n nodes and e branches, the algorithm [30] is optimal and requires $O(n)$ time and $O(e + n \log n)$ messages. After one run of the distributed algorithm chosen, each node locally stores a set of optimal paths to (or from) all other nodes in G. Accordingly, if a node u wishes to be authenticated to a node v, then v checks if there is a locally stored path to u and then verifies if its validity is acceptably high by also using the respective type-I trusts for the nodes involved. Since G changes in time, the distributed graph algorithm needs to be repeated periodically. Alternatively, the algorithm may be updated whenever G and the corresponding $cert^I G$ change, and this update is less complex if the changes in G are small.

A.2 Verification of Public Key Certificates

If the computed public key validity $\mathbf{valid}_v(P_u, \underline{ID}_u)$ is acceptably high, then v needs to verify all the type-I certificates used for this computation, which itself does not require the signatures in these certificates to be known and verified. The type-II certificates for the PK-based type-I trusts used in the integrated trust model also need to be verified. To this end, the nodes in G that contributed to the computation of the public key validity need to send the type-I and type-II certificates to v for the verification.

Another possibility that is interesting for revocation purposes, which is described in more detail in ID4.4 [4], can be called on-line verification of public key certificate chains. Namely, it relates to certificate chains that involve only the nodes in G, which are currently present in the network, or, equivalently, to the paths belonging to the restriction of $cert^I G$ to G, which is denoted as $\underline{cert^I G}$. More generally, a chain may also involve nodes out of G provided that they are connected to the adjacent nodes in G by inbound and outbound branches with the associated type-I certificate validities equal to 1 and the ID-based type-I trusts of v in the issuing nodes also equal to 1. Namely, such type-I certificates need not be verified on line and are then skipped in the on-line certificate verification process. To this end, each node in G should effectively mark and use only the certificates that it issued to or received from nodes in G or, equivalently, filter out the other certificates (except for the ones with the associated maximal validities). This process should be dynamic since G changes in time. Certificate chains may be found either by using distributed graph algorithms or, in a simplified version, by a flooding algorithm essentially proposed in ID4.4 [4]. The difference is that the flooding algorithm proposed in ID4.4 [4] includes on-line verification and generation of certificates, whereas the flooding algorithm proposed here serves for finding certificate chains only.

Given a pair of nodes v and u, a unidirectional flooding algorithm starts by u sending a message to each inbound node in $\underline{CertG}$, where the message contains the ordered list of previously visited nodes (initially, only u) along with the added branch validity. The message also contains the maximal depth d allowed to be reached, after which the flooding stops. Upon receiving a message, each node extends the message accordingly and forwards it to each inbound node, upon decreasing the maximal depth parameter by 1. The process stops when the depth parameter is reduced to 0. In case of success, the node v will be among the reached nodes and will hence contain a set of paths from v to u, from which it may choose optimal or close to optimal paths, by using the locally stored type-I trusts, as explained in Section A.1. The length of the found paths is at most d. The process may be pruned by keeping and forwarding only the partial paths with relatively high validity.

In a bidirectional flooding algorithm, apart from the node u, the node v also starts sending a message, however, to each outbound instead of inbound node and the flooding process thus goes on in both directions, i.e., in the inbound direction from u and the outbound direction from v, until a specified depth is reached in each of them. In case of success, there will be at least one node reached by both processes and it can then supply the found paths to v. The depth can now be roughly halved to $d/2$ while keeping a high probability that a path of length at most d be found, provided that it exists. Reducing the depth two times roughly reduces the number of the involved nodes to the square root of the number corresponding to the unidirectional algorithm.

It is also possible to investigate versions of the flooding algorithm, unidirectional or bidirectional, for finding certificate chains between every pair of nodes in G simultaneously.

Each found and chosen certificate chain involving the nodes in G only can be verified on-line by using the method described in ID4.4 [4]. In this method, each node along the chain, starting from the last node u, issues a fresh certificate to u and then sends this certificate to the preceding node along the chain (i.e., the next node in the upstream direction), upon verifying a certificate issued to the succeeding node along the chain and the fresh certificate previously issued by the succeeding node to u. In this way, a chain is verified in a distributed way and the intermediate nodes along the chain besides v may all issue indirect type-I certificates to u, by using the locally stored type-I trusts, which may then be used in future authentications.

B Authentication Protocol between T-Nodes and U-Nodes

Authentication of a U-node to a T-node consists of the following steps:

- U-node u sends its name (ID_u, P_u) to t, upon receiving request for authentication;
- T-node t sends its identity ID_t and sequence number i to u, and u then forwards ID_t and i to dedicated T-node t_d along with a request to be authenticated;
- u is freshly authenticated to t_d by type-II or type-I identification;
- t_d computes $S_t = F(K, ID_t)$ and $S'_t = G(S_t, ID_u, P_u, i)$, sends S'_t to u encrypted by P_u, and u decrypts the message and obtains S'_t, provided that the access rights for u to t are granted by t_d;
- u is then authenticated to t by using a challenge-response protocol based on the shared key symmetric key S'_t: essentially, t sends a (random) nonce to u, u encrypts this nonce and the names (ID_u, P_u) and ID_t by using S'_t, and t verifies u's knowledge of S'_t by decryption;
- If required, u can then send messages to t encrypted by S'_t and vice versa.

Similarly, authentication of a T-node to a U-node consists of the following steps:

- U-node u sends its name (ID_u, P_u) and a request for authentication to t;
- T-node t sends its identity ID_t and sequence number i to u, and u then forwards ID_t and i to dedicated T-node t_d along with a request for authentication of t;
- t_d is freshly authenticated to u by type-II or type-I identification and u is freshly authenticated to t_d by type-II (or type-I) identification;
- t_d computes $S_t = F(K, ID_t)$ and $S'_t = G(S_t, ID_u, P_u, i)$, sends S'_t to u encrypted by P_u, and u decrypts the message and obtains S'_t, provided that the access rights for u to t are granted by t_d;
- t is then authenticated to u by using a challenge-response protocol based on the shared key symmetric key S'_t: essentially, u sends a (random) nonce to t, t encrypts this nonce and the identities ID_t and (ID_u, P_u) by using S'_t, and u verifies t's knowledge of S'_t by decryption;

- If required, t can then send messages to u encrypted by S'_t and vice versa.

C Validation of the Proposed Reputation Techniques

In this Appendix we propose some experimental evidence that supports the techniques proposed in Chapter 4. In particular, Section C.1 refers to the token-based reputation correlation scheme proposed in Section 4.1, while Section C.2 provides results that refer to the service reputation described in Section 4.2.

C.1 Reputation Correlation Approach

In Section 4.1 we define a token-based mechanism to enable nodes to maintain the history of past transactions for different contexts [17]. This solution moves the burden of storing personal information to the nodes as it is their interest to trace and maintain their reputation. The token is used to correlate the reputation values earned in different contexts and it gives a first view of the node's willingness to cooperate. This section is meant to expand and integrate with experimental results the

Here we validate the performance of the token-based approach by comparing the results of this mechanism with the ROCQ reputation management scheme, also named hereafter *basic* reputation scheme. In both cases, we simulate that nodes clean the stored information on the reputation and credibility values of other peers and the quality values of the nodes with whom they have interacted before moving. This simulates that nodes always join different clusters.

Table 1 shows the parameters we use to derive the traces of the nodes' positions. The movement of nodes can be slow or fast and it depends on the speed and on the time a node remains in a position before moving again. For all settings, nomadic users are divided in two classes to simulate a more heterogeneous population: the difference consists in the time nodes spend in a place and are identified as class I and class II.

In our experiments we use the parameters listed in Table 2. There are 5 initial clusterheads in each cluster and 6 designated agents to aggregate and store the reputation value of a node in a cluster. To compute the trust value, either the reputation value, if available, or the average opinion, if there exist a past direct transaction between the same two nodes, is used. Then, nodes use a deterministic threshold 0.5 to decide if peers are trustworthy. At each iteration, two nodes of same cluster are selected randomly to

Table 1: Parameter settings for mobility

Slow Mobility		
	Speed	Stay Time
Class I	[0.18; 0.46] m/s	[30; 60] s
Class II	[0.18; 0.46] m/s	[36; 120] s
Clusterheads	[0.018; 0.046] m/s	[3,000; 6,000] s
Fast Mobility		
	Speed	Stay Time
Class I	[0.74; 1.85] m/s	[7.5; 15] s
Class II	[0.74; 1.85] m/s	[9; 30] s
Clusterheads	[0.074; 0.185] m/s	[750; 1,500] s

Table 2: Parameter settings for the simulation of the reputation management scheme

Type of Nodes			
Total # Nodes	$1,000$	Initial # CHs per community	5
# Nodes Class I	315 − # CHs	# Nodes Class II	685
Transactions			
# Transactions	470,000	Before nodes' movement	500; 2,500
To bootstrap the reputation	1,000	To bootstrap the token approach	29,000
Simulation settings			
Topology	Random	Mobility model	Random waypoint
Experiments run	6	# Designated agents	6
Type of node maliciousness	Report and service	Type of decision	Deterministic
Trust threshold	0.5	Size of the Token	100

interact. The result of the interactions is used to evaluate the performance of the reputation management system, such as the success rate of transactions defined as follows:

$$Success\ rate = \frac{\#Tr_{good} + \#Av_{malicious}}{Total\ \#\ of\ transactions} \tag{16}$$

where $\#Tr_{good}$ is the number of interactions with good peers that go ahead and $\#Av_{malicious}$ the number of avoided interactions with malicious peers.

We run an initial number of transactions to bootstrap the reputation management system and the token-based mechanism, as shown in Table 2. This is required to have an initial reputation value for the nodes, an initial estimation of the clusters' credibility and initial reports inside the token. The cluster and node credibility are initially set to the uncertain value of 0.5: 0 means no confidence and 1 the node is fully confident in the reporting cluster/agent.

The size of the token is limited to 100 records, thus, it stores the history of the node for last 150 minutes; a smaller token will not account for the relevant history of the node and few samples are not sufficient to estimate the behaviour of the node. Reports are collected at regular intervals, with a rate that is $\frac{1}{5}$ of the frequency of mobility, which is given by the number of transactions in the system between two subsequent movements. Clusters' membership is updated after the movement in accordance with the trace file.

We simulate two metrics for a node to take decisions: reputation if there is no direct transaction with the correspondent node or opinion if there is no estimation of the reputation value, indicated by *or* and *ro* in the plots respectively. We also simulate a different number of transactions in the system between two subsequent movements, indicated by 500 and $2,500$ iterations in the plots.

In the following sections we analyze the impact of the node speed, i.e., how often the cluster membership changes, the frequency of report collection and the size of the cluster on the capability of the reputation management schemes in identifying malicious nodes.

Impact of the speed of the nodes

In Fig. 5 we compare the effectiveness of the token-based mechanism, in terms of fraction of correct decisions, when the nodes move slowly in the area, as defined in Table 1. Fig. 5 shows that the token-based

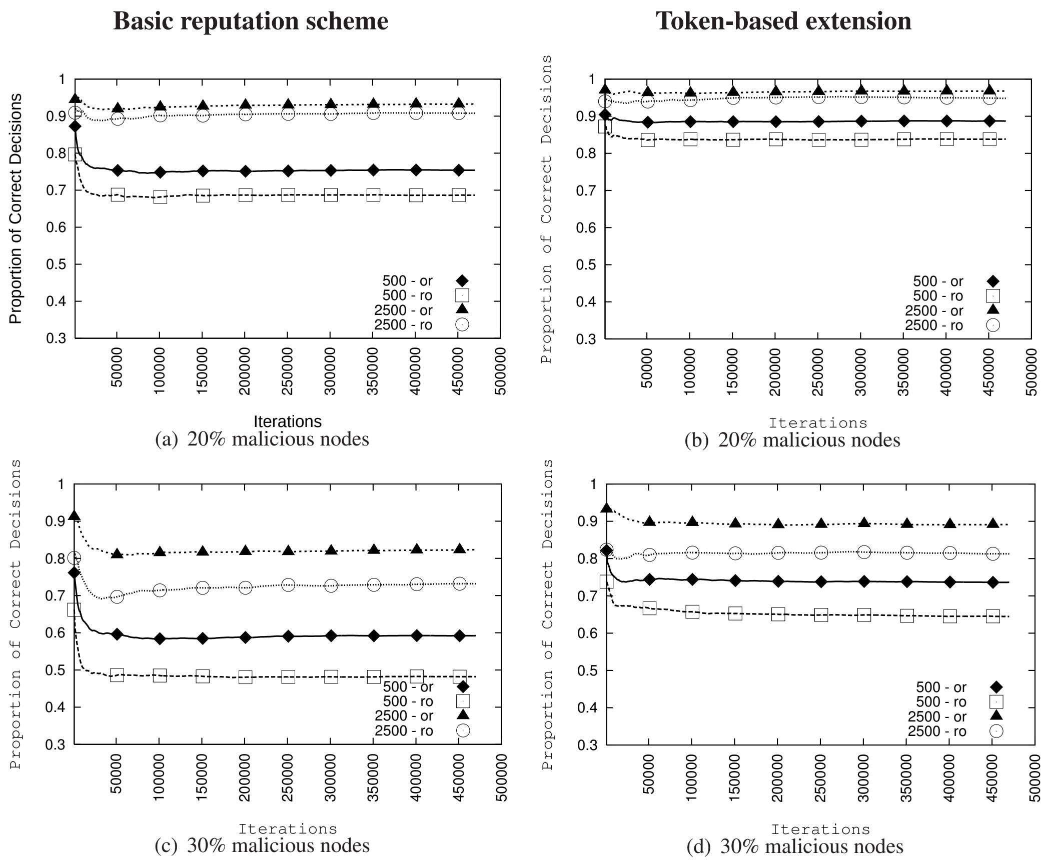
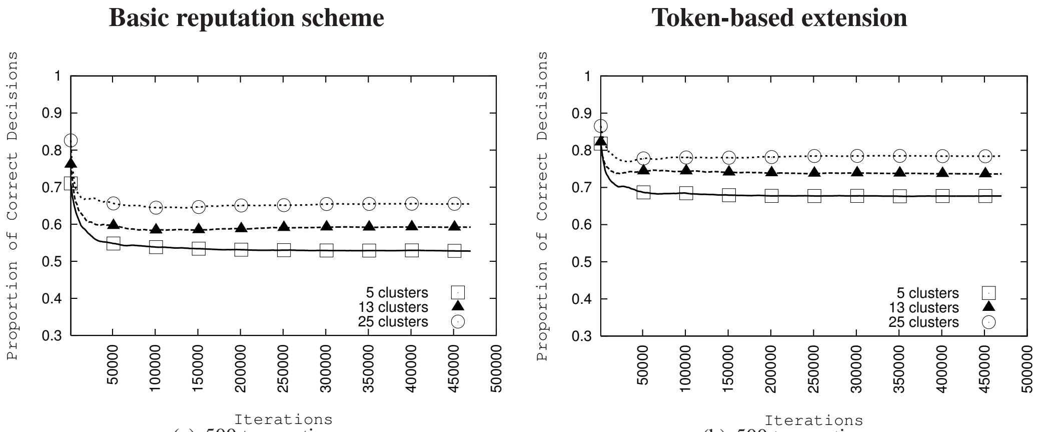


Figure 5: Proportion of correct decisions for 13 clusters and slow mobility of the nodes.

mechanism improves the performance of the reputation management system, specifically, when the fraction of malicious nodes increases in the system (plots (c) and (d)). The improvement is greater when there are few interactions available to form an opinion or to estimate the reputation of the nodes. This is shown by the curves plotted for 500 transactions before each movement with an increase of 15% of correct decisions compared to 10% for $2,500$ transactions.



(a) 500 transactions (b) 500 transactions

Figure 6: Success rate for slow mobility of the nodes and 30% of malicious nodes.

As expected, a decision based on direct experience (indicated by *or* in the plots) increases the success rate in all cases and reduces the improvement of the token based solution over the basic scheme. In fact, it improves the evaluation of the reputation but the decisions are local and not biased by the maliciousness of the reporting agents. It is worth noticing that the token-based mechanism does not increase the success rate when opinion is used first for the decisions, as it modifies the reputation management system for what concerns the evaluation of the reputation and reputation is only used initially.

In Fig. 6, we simulate the presence of a different number of communities in the area. We plot only the case when nodes decide to transact based on direct experience, since it is closer to a real scenario. When there are few interactions between two subsequent movements, the presence of more clusters gives a higher success rate, plots (a) and (b) in Fig. 6. This is true for both the traditional reputation scheme and the extended version as nodes interacts more frequently with the same nodes, and, as such, they can form a more accurate estimation of the nodes' trustworthiness. We expect that when the number of subsequent transactions increases the impact of the number of clusters is smoothed by a higher number of samples to evaluate reputation values.

In Fig. 7 we plot the success rate when the nodes move with fast mobility. The higher mobility does not allow nodes to stay in a cluster for the time sufficient to have an accurate estimation of the reputation value. In fact, the success rate decreases by 10% compared to slow mobility. We also expect that higher mobility slightly reduces the impact of the number of clusters on the system's performance as the same pair of nodes interacts less frequently. In particular this is true for big clusters because nodes change cluster less often.

Thus, we can conclude that small clusters enables the node to rely more on their direct experience, as they can exploit the amount of satisfaction in previous transactions to predict the behaviour of the nodes. Moreover, the token helps nodes in taking decisions when they must evaluate the trustworthiness of new-comers.

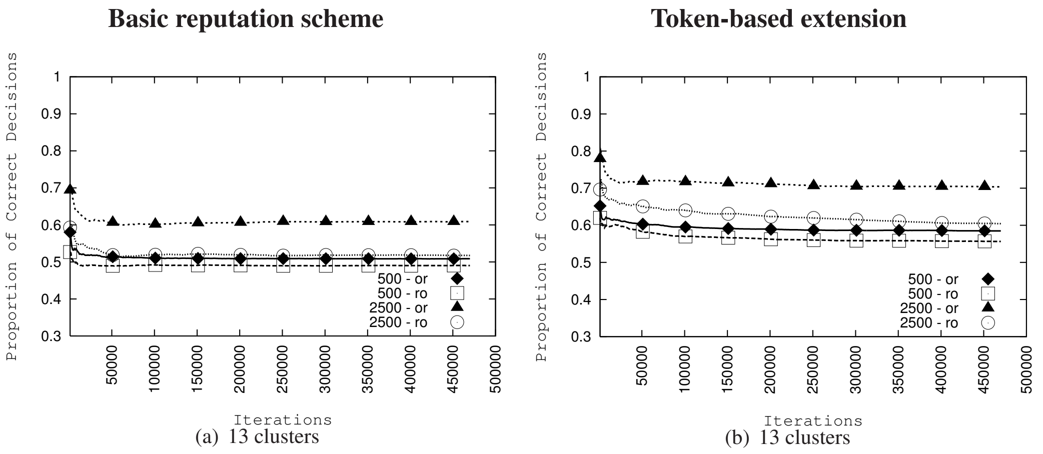


Figure 7: Success rate with fast mobility of the nodes and 30% of malicious nodes.

C.2 Service reputation

Simulations have been presented for the node reputation scenario [1, 2]. The main issue that we want to investigate here is the asymmetry between nodes and services: while in the previous scenarios malicious nodes supported each other (supporters and supportees were the same actors), now it is possible that a single malicious node decides to support many bad services, or that many malicious nodes support only one service.

In order to test whether the asymmetry between nodes and services affects the system's performance, the evolution of reputation values with time has been simulated assuming a network with high mobility where nodes interact with a random subset of peers at each request. The simulation considers a fixed set of 10 semantically equivalent services, some of which (1 or 5) being faulty. The network is composed of a fixed population of 20 U-nodes, of which a variable number (1, 10 or 17) is biased in favor of the faulty services. Upon execution, good services yield a high opinion having normal distribution with $\mu = .7, \sigma = .1$, while the faulty service generates a much lower opinion by nodes that execute it ($\mu = .3, \sigma = .1$). While all U-nodes provide their true average opinion value about all services, biased U-nodes provide a constant opinion equal to 1 about the faulty services. Biased U-nodes provide their true opinion value about good services in order to mask their bias and try to preserve their credibility.

At every step of the simulation, a random U-node requires the execution of a service. We assume that, by running the service discovery protocol, each service has probability $p_d = 1/3$ of being discovered. The U-node computes the reputation of each service in the list. In order to compute it, we assume that each U-node has probability $p_r = 1/3$ of responding. After all responses have been collected, the service with the highest reputation is executed and the resulting opinion is stored in the executing U-node.

Results are shown in Fig. 8 for 50 runs of 500 steps (service execution events) each on 6 different settings. Average results of the 50 runs are shown; error bars are omitted for clarity. Every graph corresponds to a different number of faulty services or bad nodes, and contains three curves.

The first curve reports the ratio between the number of executions of good services and the total number of service executions. The optimal ratio corresponds to the probability that at least one good service is reported by the service discovery protocol. With our assumptions, its value is $1 - (2/3)^g$, where g is the

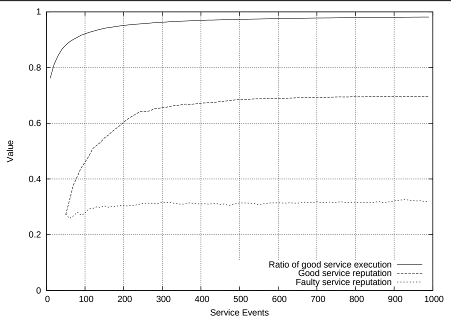
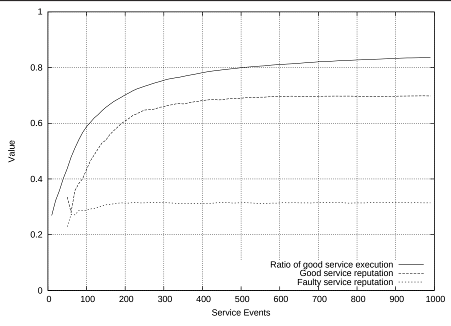
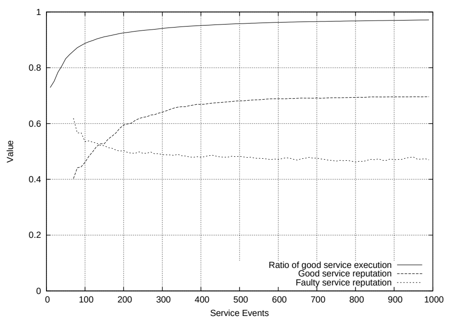
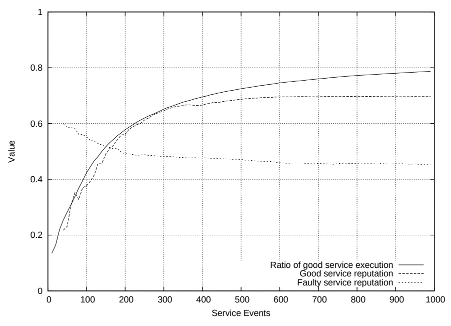
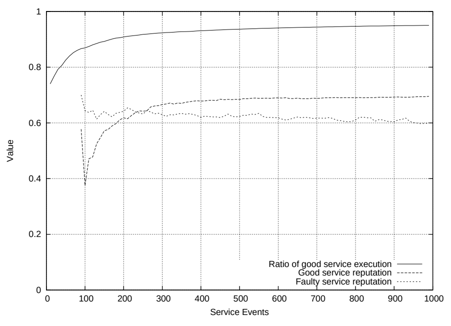
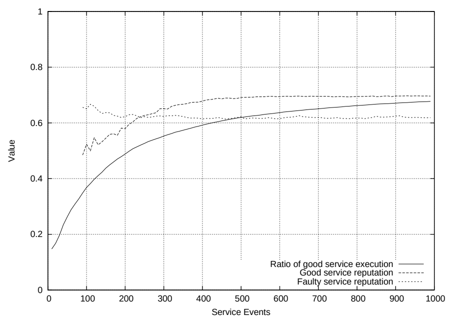
	1 faulty service out of 10	5 faulty services out of 10
1 biased U-node out of 20		
10 biased U-nodes out of 20		
17 biased U-nodes out of 20		

Figure 8: Ratio of good service executions and average reputation of good and bad services vs. number of service execution events.

number of good services. Therefore, the ideal ratio of good service execution would be 0.974 for the graphs in the first column and 0.868 for the second. The second and third curve of each graph show the evolution of the average service reputation, respectively for good and for faulty services.

While the setting is very simplified, some facts can be observed. First, if many nodes are biased (see in particular the central row), the initial interactions lead to a higher reputation of the faulty services because of the high probability that a large number of biased nodes is involved in the reputation phase. After an initial transient, however, reputation values stabilize to an acceptable stationary value because of the lowering credibility of the biased nodes (not reported in the graphs for clarity). An even larger fraction of biased nodes (see bottom graphs) is required before the reputation system is broken. The main difference between the left and right column is clearly a lower good service execution ratio, due to the higher probability that the service discovery protocol provides a large number of faulty services.

C.3 Conclusion

The application of the proposed reputation scheme to an asymmetric setting where nodes exchange opinions about services in order to build reputations has been proposed and simulated in a setting that exemplifies the asymmetry between the two sets of entities (nodes and services). Simulations suggest that the robustness of the model is not compromised in the new setting, and motivate the implementation of the scheme in the BIONETS simulation environments for further investigation.

D Enhancing Application Level Privacy (Extended Version)

In the BIONETS networks, the messages are disseminated according to the *store-carry-and-forward* principle. In particular, the messages are stored and carried by the mobile devices and forwarded to intermediate devices when they have connection (i.e., when they are in vicinity of each other). In BIONETS networks, local information needs to be distributed to a set of nearby destinations based on their interest in the information.

As a motivating example, let us consider a small advertisement application. Using this application, a diligent student participating in each lecture can earn some money. He types an advertisement about helping other students. There is a group of students who participate in some lectures meeting the diligent student. Their handheld devices can download the advertisement, because they are interested in any messages related to the university. They go back to the dormitory and meet students who do not visit the lectures at all. Lazy student's handheld device downloads the advertisement and before the exam they can contact the diligent student.

Without privacy protection, no new technology should spread widely. The privacy of the users must be ensured in BIONETS networks as well. Some of the problems can be mitigated by traditional technologies, but some new problems are introduced by the store-carry-and-forward manner of the BIONETS networks.

The privacy problem can be clarified through the above described example. Let us consider a lecturer who wants to make a list of the students attending the lectures without the consent of the students. This list can be easily constructed if the ownership between the students and their handheld devices are disclosed once, and their handheld devices are traceable.

The privacy of a system and the anonymity of the users can be a problem on different levels of the communication stack. Using a rough partitioning, the problem can be related to the physical layer of the network, the network and transport layers, and the application layer. In the physical layer, the physical characteristics of the transceiver, in the application layer, the application data, and in any layers, especially in the network and transport layers, the identifiers must be hidden or anonymized.

The physical layer privacy problems are also referred to remote device fingerprinting or remote device identification. In [5], the authors can fingerprint a device remotely without any modification on the target machine, from any distance, measuring only the clock skew of the target machine. This technique can be very accurate, but needs very long interaction time. In [6], a wireless device driver identification technique is given, which can reliably identify a device driver remotely, but cannot differentiate between devices using the same driver. In [7, 8], very accurate hardware fingerprint based device identification methods are given which can identify the devices very precisely, but utilizes special expensive hardware equipments.

In the network and transport layer, the anonym emailing and anonym web browsing are well studied problems (e.g., in [31, 32, 33]). A more similar problem to the BIONETS network's arises in Vehicular Ad-hoc Networks (VANET). In various aspects, the VANETs are similar to BIONETS networks. They are mobile, the transaction times are short, and the devices belong to a well defined person. However, in VANETs, some infrastructure elements can be assumed in contrast to BIONETS networks. Nevertheless, the solutions proposed for privacy issues can be a good inspiration for solving the privacy problems in BIONETS networks. Many authors addressed the privacy problem in VANETs, (e.g., in [34, 35, 36]). An overview of the problem of providing location privacy for VANETs is given in [37]. That paper shows that messages contain state dependent information (e.g. speed, location, time) and contain no personal information. Thus, the most beneficial attack is the tracing of identities.

The problem of privacy in opportunistic and delay tolerant networks is considered in some papers. In [38], the authors raise the problem of data privacy when a node sends sensitive information to another node and it does not want it to be available for intermediate nodes. The author of [9] proposes a pseudonym generating technique using public keys for supporting the anonymous communication. In [10], an anonymous communication solution is presented and a new anonymous authentication protocol is introduced for delay tolerant networks. The solution is based on identity-based cryptography and solves the problem of anonymous communication on the network level, but the application level problems are not handled.

A relevant, but somewhat different application level privacy problem can be found in [39], where a re-identification algorithm is given targeting anonymized social network graphs. The de-anonymization algorithm is based purely on the network topology. This mechanism can be hardly adapt into BIONETS networks, because this algorithm only relies on the static social connections of the nodes.

It is essential that the communication is anonymous. Anonymity (or at least pseudonymity) can be easily achieved by the usage of pseudonyms (i.e., temporal identifiers). A more serious and BIONETS network specific application level privacy problem is that the nodes can be identified by their stored messages. If an attacker is able to build a user profile using the exchanged application data, the user becomes traceable even if the communication is completely anonymous. Therefore, new mechanism or adaptation of proposed mechanism is required in BIONETS networks to ensure untraceability of the nodes, namely, to avoid building traceable user profiles.

The main contributions of this section are the following: We characterize and simulate efficient software based attackers that can link different appearances of the same U-node with high probability using only regular handheld devices. We suggest and evaluate an efficient defense strategy, too, which is useful against this attacker without jeopardizing the U-node's main goal, the message collection.

After reviewing the state-of-the-art and focusing on the problem of the application level privacy in BIONETS networks, the remainder of this section is organized as follows. In Section D.1, we describe the system model. The attacker model is presented and four different attackers are defined in Section D.2. In Section D.3, we describe our proposed privacy enhancing technique, called Hide-and-Lie Strategy. The simulation environment is defined in Section D.4. The efficiency of different attackers and the privacy enhancing technique is exhaustively analyzed by means of simulations in Section D.5. Finally, we conclude this section of the report in Section D.6.

D.1 System Model

In our model, the users are placed on a field of arbitrary shape. They own devices which can communicate with each other within their radio range. The used wireless technology can be Bluetooth, Wi-fi, or any suitable wireless technique. The messages are generated and disseminated among the devices/users but each user is only interested in a subset of messages. The dissemination process is based on the store-carry-and-forward principle. A U-node includes a user (the owner of the device) and a device. We assume that the data dissemination has no impact on the user's movement.

The communication between the U-nodes is assumed to be anonymous. Hence, an attacker is not able to trace a U-node by e.g. simply tracing a network identifier.

The messages are generated by T-nodes. In our system model, the time is slotted, and each T-node generates a new message with a fixed average rate: ϱ messages per time step. The T-nodes are static and each one stores only the most recently generated message. This message can be downloaded by any U-node that passes by the T-node.

We assume that a mechanism can filter out the fake messages from the network. This assumption is necessary, otherwise, an attacker is able to create a special decoy message and trace a U-node by following it. The analysis of the effectiveness of this kind of attacks and countermeasures is out of the scope of this document. However, to justify that this assumption is realistic, we introduce some mechanisms:

1. A U-node downloads only those messages that are generated by a trusted entity and the authenticity is proven.

2. A U-node downloads any message but offers only those whose reputation is higher than a threshold. The reputation can be computed automatically or manually based on the content of the message.

3. A U-node downloads any messages but offers to other U-nodes only the widespread disseminated messages. This technique can be used only by the minority of the U-nodes.

Note that the first two mechanisms can be based on security services provided by BIONETS networks presented in [2].

For the sake of simplicity, it is assumed that there are C categories, and each message belongs to a single category. When a T-node generates a message, it specifies which category the new message belongs to. Each message is classified into a category uniformly at random. Therefore, a new message belongs to a specific category with probability $\frac{1}{C}$.

An interest profile (IP) of a U-node, which is a part of the user profile, is a binary vector representing a list of categories the U-node is interested in. A message belonging to category k is called primary for a U-node if $IP[k] = 1$, the message is secondary otherwise. A U-node is interested in any given category (i.e. all the messages belonging to that category are primary for the U-node) with probability ε. As the participating U-nodes are interested in at least one category, the case when $\varepsilon = 0$ can be excluded, therefore, the cases when $0 < \varepsilon \leq 1$ are considered. For the sake of simplicity, ε is equal for each U-node in each considered scenarios.

Each message is assumed to have a unique identifier and a U-node can decide based on this identifier if a message M is stored in its memory before downloading from another U-node.

According to our assumptions, a message M has the following formula:

$$M = [ID|CAT|data] \tag{17}$$

where the ID is the unique identifier of the message, CAT is the identifier of the category which the message belongs to, and $data$ is the content of the message. The length of the $data$ may be some magnitude higher than the length of the ID and the CAT.

When two U-nodes get in the vicinity of each other, they start to exchange messages. Each U-node u wants to download those messages from the other participant that u does not store and fit to its interest profile.

The whole message exchange may be not completed because the U-nodes are mobile and they may leave the radio range of the other participant before exchanging all the required messages. Therefore, according to the system model, the U-nodes are not able to obtain as many messages as they want but at maximum one for each participant per time step. It is assumed that a message is downloaded without interruption in a time step.

In our imagined scenario, the batteries of the handheld devices can be easily recharged day by day, hence, the cost related to the battery consumption due to communications is negligible. However, the average number of downloaded messages per U-node is investigated in the simulations to get an insight of the message exchange rate.

The storage cost has two aspects: 1) The messages need storage space and storage constraints may limit the number of stored messages. No explicit limitation for the storage space is defined, however, the maximum quantity of the stored messages in the devices is investigated. 2) The time needed to determine which messages the U-nodes want to download increases polynomially with the number of messages stored by the other participant. Therefore, the increasing number of messages is controlled by deleting the messages of the system. Each message is deleted D time steps after its generation.

In order to investigate the effect of the defense mechanism on the message delivery ratio, a formula is defined for the gain of the U-nodes. Until time step t, a U-node u obtains $O_u(t)$ number of primary messages while in the system, there have been $A_u(t)$ number of messages generated which is primary for u. The gain ($G_u(t)$) of U-node u is the ratio of these values.

$$G_u(t) = \frac{O_u(t)}{A_u(t)} \tag{18}$$

The fact that the gain reaches its steady state value is proven in [28]. Therefore, in what follows, we approximate the steady state value of the gain by considering the gain after sufficiently large simulation time and we denote it by G_u.

From the privacy point of view, an extreme approach consists in denying the participation in the network. We assume that the U-nodes want to obtain messages while they want to preserve their privacy, too. A beneficial and in the same time selfish behavior would be to download any kind of messages but not forwarding them. To prevent this kind of selfish behavior, we assume that a mechanism encourages the message dissemination among the U-nodes [28]. Therefore, the U-nodes themselves want to increase the number of offered messages.

D.2 Attacker Model

The attacker wants to track the target U-node to breach its privacy. To do so, it tries to link the profiles acquired in different times together. If the profiles can be linked correctly, the attack against the privacy is successful.

In this section, we describe what information an attacker can get from the U-nodes, how he can obtain this information and how he can link the U-nodes.

D.2.1 Leaking Information

The communication between the U-nodes can leak some information about the interests of the participants. In this report, attacks based on these leaked information are considered. We assume that the attacker can estimate the following user profile (UP) from a U-node u at time t:

$$UP_u(t) = (EIP_u(t), CHM_u(t), IDL_u(t)) \tag{19}$$

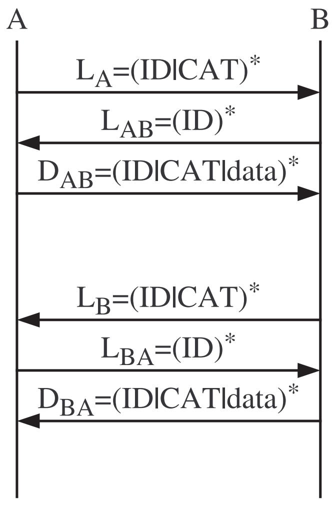
The UP consists of the following triple:

- Estimated Interest Profile (EIP) is a binary vector. The value of the vector at the k^{th} position equals to 1 if category k seems to be interesting for U-node u.
- Category Histogram of offered Messages (CHM) shows, for each category, how many messages in the ID list belong to that category.
- IDL is the ID list of offered messages.

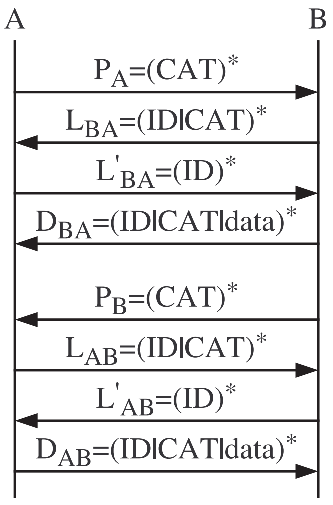
In rest of this section, we show how an attacker can obtain the $UP_u(t)$ in the case of two (a push and a pull) sample message exchange protocols.

Push-based message exchange protocol When two U-nodes get in the vicinity of each other, they interact as it is shown in Figure 9(a). First, U-node A, which starts the communication, sends a list of the stored messages (L_A) consisting the ID and the category CAT of each message. B sends back a list of required messages (L_{AB}) containing the IDs of those ones which are primary for B and B does not store. A sends the content (D_{AB}) of each message listed in L_{AB}. In the second part of the protocol, the roles change and A obtains through the same steps those messages which are primary and not stored in its memory.

An attacker can obtain the triple with the following mechanism. Considering B as an attacker, it can easily calculate the CHM and the IDL by obtaining L_A. In the second part of the message exchange, B creates a special L_B. B first collects those categories (C_A) which were not present in L_A. Then, B creates L_B such that each category from the list C_A is represented at least by one message. Getting the response L_{BA}, B reads what are the categories that A is interested in but it could not obtain or deleted the messages belonging to those categories before. The EIP can be calculated by getting the union of the categories of the stored messages (L_A) and the category of the required messages (L_{BA}).



(a) Push model



(b) Pull model

* indicates that the sent message contains a list of described elements

Figure 9: Message exchange protocols

Pull-based message exchange protocol When two U-nodes get in the vicinity of each other, they interact as it is shown in Figure 9(b). First, U-node A, which starts the communication, sends a list of categories (P_A) according to its IP. U-node B collects the ID of messages belonging to categories listed in P_A (L_{BA}) and sends to A. A removes from L_{BA} the IDs which is already stored and sends back the list (L'_{BA}). B sends the contents of the required messages (D_{BA}). In the second part of the protocol, the roles change and B obtains through the same steps those messages which are primary and not stored in its memory.

An attacker can get the same set of information as in the push-based message exchange protocol. Considering B as an attacker again, B can easily get the EIP from the P_A. In the second part, B can claim that it is interested in all the messages and list all categories in P_B. As a response, A will send the list of stored messages, and B can get CHM and IDL as we have shown in the push model.

In this section of the report, we abstract away what message exchange protocol is used, we only assume that an attacker can obtain the current value of the $UP_u(t)$ for each U-node u in time step t.

D.2.2 Attacker Behavior

The attacker, in our model, behaves according to the following attacker model:

1. The attacker identifies its target U-node (u_T) from N U-nodes.
2. The attacker reads the current user profile of the target: $UP_{u_T}(t_0)$. The time step when this happens is considered as a reference time, i.e. t_0.
3. τ time later ($t_1 = t_0 + \tau$), the attacker reads $UP_{u_i}(t_1), i \in [1..N]$ of each U-node and calculates a metric how similar is u_i to u_T. τ is also referred as the attacker delay. To mislead the attacker, the U-nodes can slightly modify their UPs. The UP perturbation is defined in Section D.3.
4. The attacker chooses the U-node most similar to the target U-node. If more than one have the maximal similarity value, it chooses randomly between them. If the chosen U-node is u_T, the attacker is successful.

We have chosen for the analysis the success probability of the attacker as the privacy metric, because it is widely used and tells the most about the expected outcome of the attack. In the cryptographic literature, a widely used metric is the indistinguishability of the target from a randomly chosen U-node [40]. This metric differs from ours slightly as the attacker wants to distinguish the target from every other U-node. Our extended metric can be imagined as the conventional metric used N times one after the other. More precisely, if the attacker can recognize its target from two U-nodes with probability p, then it can recognize it from N U-nodes with probability p^{N-1}, if the U-nodes are independent. The conventional model is more sensitive for p close to 0.5. In contrast, the extended model is more informative for p close to 1. As the results show, p can be close to 1 when no defense mechanism is used, so the extended model is used.

To fully define the attacker, a similarity metric must be defined. Some possible and useful similarity functions are defined in the next section.

D.2.3 Attacker Functions

The attacker can define the similarity of the target and a suspected U-node based on the $UP_u(t)$. Using the user profiles of the U-nodes, the attacker can calculate the similarity using an attacker function $\mathcal{A}$.

More formally the input of $\mathcal{A}$ are $N + 1$ user profiles, and the output is an ID of a U-node:

$$\mathcal{A} : (UP_{u_T}(t_0), UP_{u_i}(t_1), i \in [1..N]) \rightarrow j, j \in [1..N] \tag{20}$$

The attack is successful if and only if $j = T$.

It is clear that any attacker can reach a minimal value of the success probability $\frac{1}{N}$ by simple guessing. Higher values can also be achieved using more sophisticated attacker functions. In the following, four different simple attacker functions are defined.

Prefiltered ID Based attacker function assumes that U-nodes show their real interest profiles. The attacker can filter out every suspect who has different EIPs, considering only the U-nodes whose $EIP_u(t_1)$

equals to $EIP_{u_T}(t_0)$. From the remaining set, it selects the one whose $IDL_u(t_1)$ is the most similar to $IDL_{u_T}(t_0)$. Under similarity, the cardinality of the intersection of the target's ID list and the suspect's ID list is meant. If the remaining set is empty, the attacker selects the target by pure guessing. The intuition behind this attacker is that after some time the target can get some new messages and delete some old ones, but mainly its ID list is unchanged. This attacker can be very efficient if the U-nodes show their real IPs which means that EIPs are not changed over time, but can be very inefficient if the EIPs are changed.

Unfiltered ID Based attacker function is a simplified version of the previous function, as it uses only the cardinality of the intersection of $IDL_{u_T}(t_0)$ and $IDL_u(t_1)$, but it does not prefilter the U-nodes by their EIP. This attacker is not so efficient in case of time invariant EIPs, but less sensitive for changing EIPs.

Category Histogram Based attacker function selects the U-node u whose $CHM_u(t_1)$ is the most similar to the $CHM_{u_T}(t_0)$. The similarity of two histograms is calculated using the χ^2–test. The intuition behind this attacker function is that a U-node can show a modified EIP but the histogram represents its real interest profile if the U-node collects messages according to its real interests.

Significant Category Based attacker function is the most complex function analyzed in this report. It assumes that the interested categories are overrepresented in the ID list and the uninterested categories are underrepresented. This categorization only depends on the real IP of the target, and is hard to influence without totally changing the IP. To find the interested categories, the C categories must be classified into two clusters: the significant categories, and the remaining categories. This task can be easily done using the k-means clustering algorithm [41] on the CHMs. The result of the clustering is a binary vector of length C with ones at the significant categories. The similarity of two binary vectors is defined as the Hamming distance of the vectors.

The properties and efficiency of the different attacker functions are analyzed in Section D.5.

D.3 Defense Mechanism

In order to preserve a U-node's privacy, the User Profile (UP) should be obfuscated to distract the attacker. Against an eavesdropping attacker, another solution is to design the message exchange protocol in a way that it ensures that no sensitive information can leak during the communication. We are focusing on developing an obfuscation based mechanism because that can be used if the attacker actively takes part in the communication.

The more $UP_u(t_1)$ is different from $UP_u(t_0)$, the less likely the attacker can link the two profiles. The continuously changing profile hardens the task of the attacker, however, it may thwart the U-node from collecting primary messages. This is thoroughly analyzed in Section D.5.

Two simple methods can be used to modify the UP through modifying the Interest Profile (IP) of the U-node. The first one is to *hide* some interesting categories, and claim them as uninteresting. The second one is to *lie* about some uninteresting categories, and claim them as interesting. These techniques can be used at the same time, this is what we call *Hide-and-Lie Strategy* (HLS). The temporarily obfuscated IP is the Temporal Interest Profile or the EIP from the attacker point of view. The EIP can be transient, which means that a new EIP can be generated by every U-node in every time step.

Obviously, the required and offered messages during the message exchange must be synchronized with the EIP: 1) messages relating to hidden categories must be hidden as well, and 2) when a U-node lies about being interested in a given category, it collects and offers messages belonging to that uninteresting category.

Many different HLSs can be envisioned. Different strategies can hide or lie about different categories in different situations. In the following, a simple but rather general solution is given: every U-node generates its EIP from its IP by inverting every category in the IP with a given probability λ. Inverting means indicating an uninteresting category as interesting or vice versa. This parameter λ is the Hide-and-Lie strategy value.

As λ is a probability, it is between 0 and 1. The U-nodes which do not use any obfuscation techniques can be modeled as U-nodes using HLS with $\lambda = 0$ as the U-node never modifies its EIP.

The other interesting value of λ is 0.5. It totally randomizes the IP, making the EIP a uniformly distributed random binary vector. It is the best strategy for the privacy sensitive users. For demonstrating this, let us assume two U-nodes, u_1 and u_2 at time τ of the attack. They show temporal interests in every category with probability 0.5, thus, their interest profiles are independent from their real interest profiles. If τ is greater than the D message expiration time, then none of them has any messages from the reference time of the attack (t_0). On average, every user shows interest in a given category in every second round (as $\lambda = 0.5$), so every message is collected by every U-node with the same probability. The CHMs of the U-nodes are close to the uniform distribution considering those categories where the EIP shows that the U-node is interested in (other categories are represented by 0 messages). The reason is that every message is generated and collected with the same probability. Therefore, u_1 and u_2 show user profiles, which are independent from the user and statistically the same.

The used HLS transformation of the users' profiles generates every possible UP with the same probability, thus, no statistical test can distinguish between u_1's and u_2's UP.

Values of λ greater than 0.5 are useless for the U-nodes, as they make the EIP as traceable as the inverse EIP with $\lambda' = 1 - \lambda$, but the U-nodes collect more uninteresting messages than interesting ones. Consequently, in the following, only $0 \leq \lambda \leq 0.5$ are considered.

D.4 Simulation

In the simulations implemented in C++, the fixed-number of mobile U-nodes move in discrete time steps according to one of the two mobility models: the random walk (RW) and restricted random waypoint (RRW) model.

In the RW model, 300 U-nodes move on a grid of size 15×15. In each time step, a U-node can move to one of the four neighboring grid points (in what follows, these are called meeting points), or stay at the current place. The probability of each of these actions is 0.2. In each time step, the U-nodes that happen to be at the same meeting point are paired randomly and each pair executes the message exchange protocol. These pairs are able to download one message from each other as described in Section D.1.

In the RRW model, 300 U-nodes (initially placed uniformly at random) move on a field of size 20×20 unit. On the field, there are some special points chosen at random; these are called meeting points. Each U-node selects a meeting point randomly, and moves towards this meeting point along a straight line with a fixed speed. When the meeting point is reached, the U-node stops and stays there for randomly chosen time

(10 time steps on average). Then, it chooses another meeting point and begins to move again. The U-nodes that happen to be at the same meeting point in the same time step are paired randomly and these pairs are able to download one message from each other as described in Section D.1.

In the case of RW, 30 T-nodes are placed on the subset of all meeting points uniformly at random. In the case of RRW, one T-node is placed on each meeting point, and the number of meeting points is 30. All the 30 T-nodes together generate one new message per time step on average both in case of RW and RRW mobility model.

The parameters of the mobility models were determined such that the number of message exchanges are equal on average.

The length (number of time steps) of the simulation was determined in an empirical way by taking into account that the gain has to reach its steady-state value. In the beginning of the simulation, the U-nodes do not store any messages. Therefore, their gains are volatile in the first time steps. When the simulations were run for 3000 time steps, the average gain has not changed considerably for upcoming 1000 time steps in the analyzed simulations. Therefore, the attacker started its attack after 3000 time step long bootstrap ($t_0 = 3000$) and the simulator was run for additional 1000 upcoming time steps to investigate the effectiveness of the attacker for different τ values.

Some of the parameters that describe our envisioned system were fixed in order to reduce the number of simulation scenarios and other ones which have the highest effect on the success probability of the attacker were varied. The fixed simulation parameters are summarized including the mobility model specific ones in Table 3.

Table 3: **Fixed simulation parameters**

Parameter	**RRW**	**RW**
Simulation length in time steps	4000	
Number of U-nodes (N)	300	
Number of T-nodes	30	
Message generation rate (ϱ)	0.0333	
Simulation area (unit)	20×20	15×15
Number of meeting points	30	225
Probability of leaving a meeting point	0.1	0.8
Velocity	1 unit/time step	
Lifetime of messages (D)	500	

The simulation parameters that are related to the interest profile were varied: number of message categories (C) and probability of being interested in a category (ε) as these parameters affect most the success probability of the attack.

Parameter C should be higher than 1, otherwise all the U-nodes have the same IP. Therefore, the chosen lowest value is 2. We think that 50 categories is high enough, because a higher value would not affect the simulation results considerably (the results for 30 and 50 are similar). The considered values are 2, 5, 10, 30, 50.

To reduce the complexity of the simulations, we selected some $0 < \varepsilon \leq 1$ values in a way that instead of $\varepsilon = 0$ we included $\varepsilon = 0.05$, and we also investigated a special case, $\varepsilon = 0.5$. The considered values are 0.05, 0.2, 0.4, 0.5, 0.6, 0.8, 1. Note that with probability $(1 - \varepsilon)^C$, the simulator generates such an interest profile that the U-node is not interested in any category. In that case, a new IP is generated.

Recall that a U-node can choose a Hide-and-Lie strategy value from the interval $0 \leq \lambda \leq 0.5$. For the sake of simplicity, only those cases are considered when each U-node chooses the same λ value from the following set: $\{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$.

We also investigate how the time elapsed between t_0 and t_1 (i.e., τ) affects the success probability of the attack. We considered the following values of τ: $\{1, 50, 250, 500, 1000\}$. In the special case when $\tau = 1$, the ID list of stored messages does not change considerably, however, the Hide-and-Lie Strategy affects the UP. Recall that the messages are deleted from the system after 500 time steps. Therefore, when $\tau > 500$, no message will match to the target U-node's ID list in t_0.

Table 4: **Varied simulation parameters**

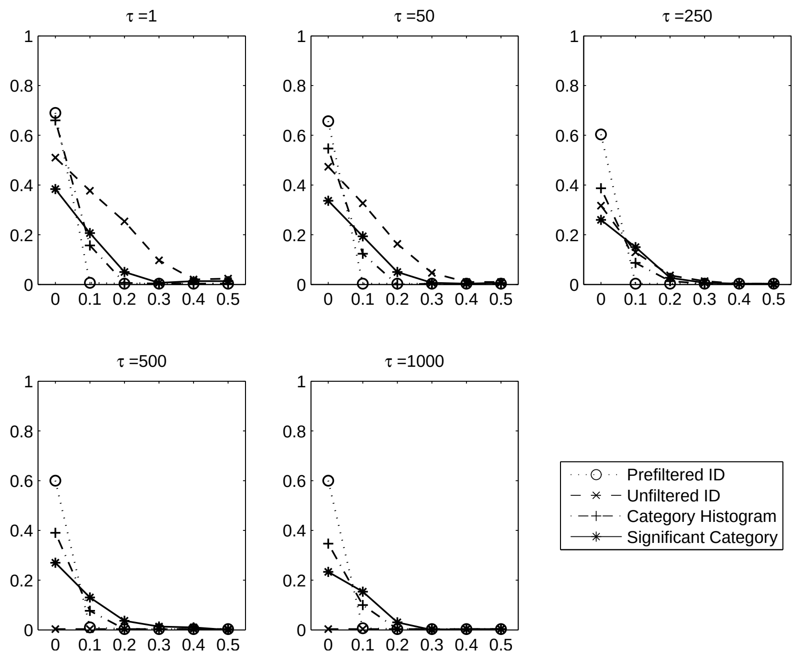
Parameter	**Possible values**
Number of categories (C)	2, 5, 10, 30, 50
Probability that a U-node is interested in a category (ε)	0.05, 0.2, 0.4, 0.5, 0.6, 0.8, 1
Hide-and-Lie strategy value (λ)	0, 0.1, 0.2, 0.3, 0.4, 0.5
Attacker delay (τ)	1, 50, 250, 500, 1000

The main objectives are to investigate the success probability of the attacks and the efficiency of the HLS. The analysis was performed in each combination of the parameter values summed up in Table 4. In one parameter set, the success probability of every attacker function was calculated using the following method: Only one simulation run for each parameter set was executed. In each execution, the attacker selects each U-node as a target U-node one-by-one at time t_0 and performs the attacker function with the target U-node and all the U-nodes as the input of the function at each $t_1 = t_0 + \tau$ time. The success probability is the ratio of the successful attacks.

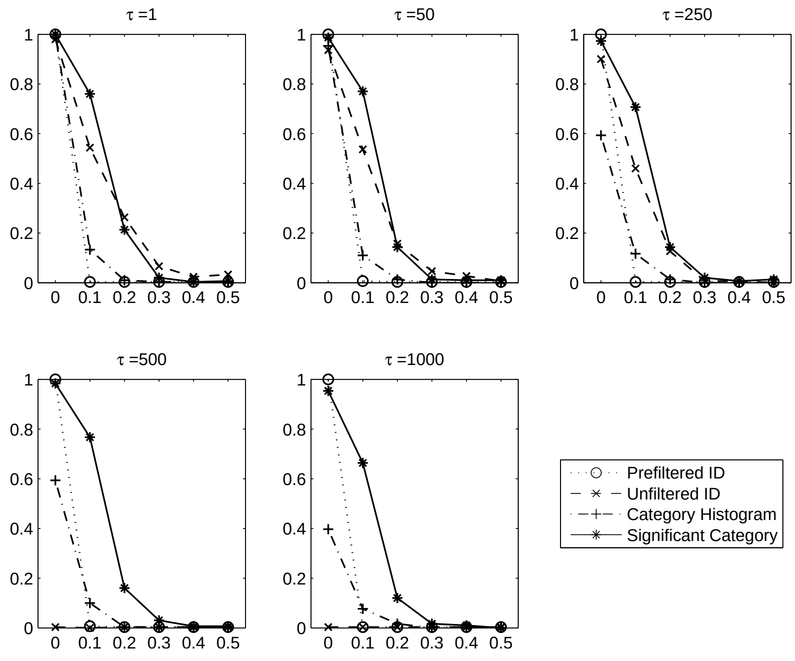
D.5 Results

In this section, two representative scenarios (see Table 5) are exhaustively analyzed. In particular, the efficiency of different attacker functions presented in Section D.2.3 and the efficiency of the defense mechanism presented in Section D.3 are investigated. Beyond the analysis of two emphasized scenarios, we show the differences compared to the other simulated scenarios. In the two considered scenarios, we investigate the effect of the Hide-and-Lie Strategy on the reached gain and the number of downloaded primary and secondary messages and the maximum memory required to follow the proposed Hide-and-Lie strategy.

As our experience showed that the chosen mobility model does not affect the results considerably, we have selected the random walk mobility model in the analyzed scenarios. In this report, we emphasize rather the effect of the probability of being interested in a category instead of the number of categories. Therefore, in the presented simulation results, the number of categories is fixed to 30, which can be a realistic value for a lot of applications. In the two investigated scenarios, the probability of being interested in a category takes the values 0.05 and 0.4. The former value refers to those scenarios where the U-nodes are interested in a small subset of messages, like in the example scenario presented at the beginning of this Appendix, while in the latter scenario, the U-nodes are interested in a large subset of messages. In Table 5, we summarize the parameter values of the scenarios beyond the already fixed parameters introduced in Table 3.



(a) Scenario 1



(b) Scenario 2

Figure 10: Success probability of $\mathcal{A}$ as a function of the Hide-and-Lie strategy values (λ)

The success probability of the attacker functions is plotted against different Hide-and-Lie strategy values (λ) and different attacker delay (τ) values of Scenario 1 and 2 in Figure 10(a) and 10(b), respectively. For the sake of better understanding, the plots are separated by different attacker delay values.

The Prefiltered ID Based attacker function assumes that the U-nodes do not apply any privacy enhancing

Table 5: **Parameter values of investigated scenarios**

	Mobility model	**C**	ε
Scenario 1	RW	30	0.05
Scenario 2			0.4

technique. According to this, it is the most efficient attacker function when $\lambda = 0$, but in any other cases, the attacker function can not distinguish the target U-node from the others, because even one entry changing in the EIP misleads the attacker.

A more robust solution can be obtained by omitting the prefiltering which results in the Unfiltered ID Based attacker function. The success probability of this function decreases when $\lambda = 0$ compared to the prefiltered function but considerably increases in other cases. The reason is that the number of all the combinations of the messages give enough variety to the attacker to identify the U-nodes with higher probability even if they hide a small subset of the messages when they meet other U-nodes. As the U-nodes increase the λ value, they collect messages from larger sets and they can hide more messages. Hence, the U-nodes are able to deceive the attacker with high probability. Therefore, the success probability of the attacker function decreases with the increasing λ value. If $\lambda = 0.5$, the attacker function is as inefficient as a naïve attacker.

The Unfiltered ID Based attacker function is very sensitive for the attacker delay. As τ increases the U-nodes delete more and more messages making the attack less and less efficient. Finally, the U-nodes delete all the messages that could match the $IDL_{u_T}(t_0)$ after D time steps and this attack becomes inefficient in cases where $\tau = 500$ or $\tau = 1000$. Recall that $D = 500$ in the considered scenarios.

The Category Histogram Based attacker function is less sensitive to the τ value, but it is less efficient when τ is lower than the ID Based attacker function. The inefficiency of this attacker function comes from the fact that the Hide-and-Lie Strategy causes intolerable differences for the χ^2–test when all the messages appear or disappear belonging to a category when EIP changes.

The attack that is least sensitive to τ is the Significant Category Based attacker function. The advantageous characteristic comes from the fact that this function tries to reveal the real interest profile. However, it still does not work when the U-nodes hide their identity with $\lambda = 0.5$ strategy, because there are no over- and underrepresented categories in that case.

The Significant Category Based attacker function is the most efficient attacker function in Scenario 2, but it is less efficient in Scenario 1.

Taking all the considered attacker functions into consideration, we can conclude that the efficiency of the attacker functions changes according to the parameters of the model. However, a common tendency is that if the U-nodes apply the Hide-and-Lie Strategy with high value of λ, none of the attackers is able to distinguish them better, independently of the value of τ, than a naïve attacker which picks up one of the U-nodes by random.

Even if an attacker can distinguish two U-nodes if their IPs are different (we call this attacker ideal IP based attacker $\mathcal{A}_{IP\ \text{ideal}}$), the probability that two U-nodes have the same IP is not negligible. The success probability of an ideal IP based attacker can be viewed as an upper bound for any other IP based attacker, such as, e.g. the Significant Category Based attacker function. This value can be determined analytically. Through this analysis, we show how different C and ε values affect the success probability of the attackers.

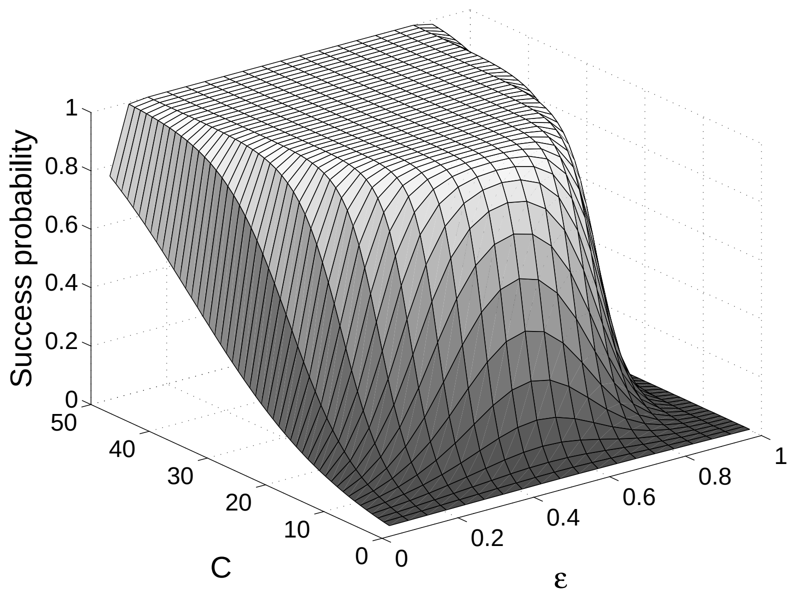


Figure 11: Analytically determined success probability of an ideal IP based attacker functions when 300 U-nodes are present in the network

The success probability of the ideal IP based attacker is determined by the number of equal IPs. To compute the success probability, first the probability p of two IPs being equal is computed as follows:

$$p = \frac{\sum_{w=1}^{C} \binom{C}{w} \left(\varepsilon^2\right)^w \left((1-\varepsilon)^2\right)^{C-w}}{\left(1-(1-\varepsilon)^C\right)^2} = \frac{\left(\varepsilon^2 + (1-\varepsilon)^2\right)^C - (1-\varepsilon)^{2C}}{\left(1-(1-\varepsilon)^C\right)^2} \tag{21}$$

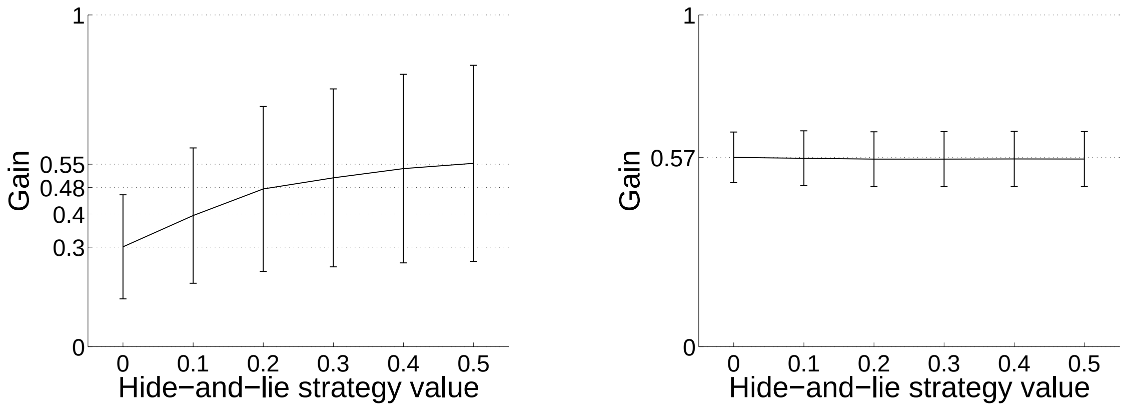
where w is the weight of the IP varying between 1 and C (at least every U-node is interested in one category).

The success probability of $\mathcal{A}_{IP\,\text{ideal}}$ is the reciprocal of the average number of U-nodes with the same IP:

$$\Pr(\mathcal{A}_{IP\,\text{ideal}}(UP_{u_T}(t_0), UP_{u_1}(t_1), \ldots, UP_{u_N}(t_1)) = u_T) \simeq \frac{1}{1 + p\,(N-1)} \tag{22}$$

The ideal values according to Eq. (22) are 0.341 and 1 for Scenario 1 and 2, respectively. These values are valid only for $\lambda = 0$, and confirmed by Figure 10. These values are shown in Figure 11, too, where Eq. (22) is plotted against different C and ε values.

The characteristic of the success probability of the attacker in the case of the two emphasized scenarios are similar to each other as Figures 10(a) and 10(b) show and these are similar to the other scenarios which are simulated but not presented here. However, as Figure 11 shows, the success probability of the ideal IP based attacker depends on the parameter value of the number of categories and the probability of a U-node being interested in a category. As one can read from the figure, when there are large number of categories in the system, the success probability of an ideal attacker is high. On the other hand, when the number of the categories is low, the success probability highly depends on the value of ε. As the value ε gets closer to 0.5, the success probability increases. The reason is that an attacker can distinguish U-nodes when the probability that the IPs of two U-nodes are equal is low. All these statements are confirmed by



(a) Scenario 1 (b) Scenario 2

Figure 12: Average gain with the empirical standard deviation

the simulation results that are not presented here, and these effects can be observed even in cases when $\lambda > 0$.

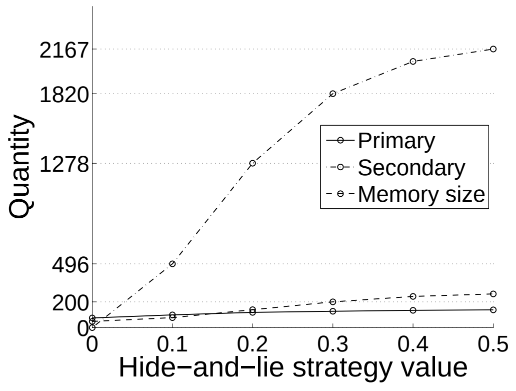
In Figure 12, we show the average gain of all the U-nodes as a function of the Hide-and-Lie strategy in the two scenarios and its empirical standard deviation. We have to stress that these two figures do not represent all the appeared characteristic of the figures, however, Figure 12(a) shows an interesting property of the Hide-and-Lie Strategy. Namely, increasing λ does not degrade but increases the data delivery ratio in some scenarios.

The Hide-and-Lie Strategy has two contradictory effects: On the one hand, when the U-nodes happen to hide what they are interested in, they may miss some primary messages to download. On the other hand, when the U-nodes happen to lie being interested in some category, they store-carry-and-forward secondary messages, which increases the data delivery ratio in general as we have shown in [28], and in Deliverable 4.2 [2], and 4.3 [29]. The cumulative effect depends on the system parameters. E.g. in a case when U-nodes are interested only in a small subset of categories and they do not carry secondary messages, they can exchange messages only with small probability. Therefore, the Hide-and-Lie Strategy in some cases can be viewed as a motivation to store-carry-and-forward secondary messages as it can be seen in Figure 12(a). On the other hand, when the U-nodes have many possibilities to get primary messages, the latter effect has no considerable benefit while the former effect degrades the gain. Surprisingly, the two effects are balanced in Scenario 2 as one can see in Figure 12(b).

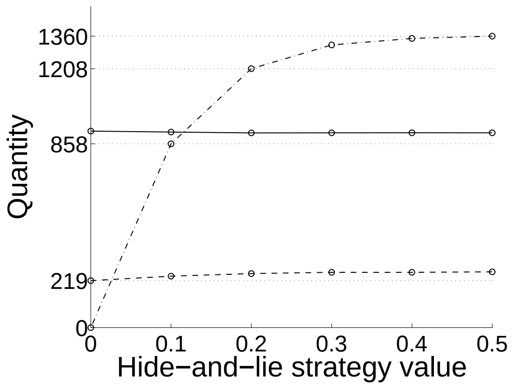
Even though we did not take into consideration the energy consumption and the memory costs of the communication when we calculated the gain, we collected related information during the simulation. We plotted the average number of primary and secondary messages downloaded by one U-node and maximum memory usage as a function of the Hide-and-Lie strategy in the two considered scenarios in Figure 13.

As one can expect, the number of the downloaded secondary messages increases with increasing λ value. The number of the downloaded primary messages changes as the gain changes because the gain is a normalized value of the number of obtained primary messages.

Even though the gains are comparable in the two scenarios as Figure 12 shows, there is almost one order of magnitude difference in the number of downloaded primary messages. The reason is that in Scenario 1,



(a) Scenario 1



(b) Scenario 2

Figure 13: Costs (Average number of primary and secondary messages downloaded by a U-node and the maximum memory usage)

the U-nodes are interested in 5% of the messages and in Scenario 2, the U-nodes are interested in 40% of the messages while the number of the generated messages does not change considerably in the two scenarios. Due to the same reason, the number of the secondary messages for a U-node is less in Scenario 2 than in Scenario 1. The ratio of the number of downloaded primary and the number of secondary messages is $\varepsilon(1-\lambda) : (1-\varepsilon)\lambda$.

Note that even if the U-nodes download more and more secondary messages as λ increases, the maximal memory usage does not increase at the same order. Thus, the U-nodes do not need to maintain much larger memories when they want to protect their privacy.

D.6 Final remarks

In this section of the report, the application level privacy in BIONETS networks has been investigated. In particular, an attacker can build a user profile of a U-node based on what messages the U-node stores and what messages it wants to download. After profiling, the attacker can trace the U-node based on the user profile even if the U-node communicates with the other U-nodes through anonymous links. A system and an attacker model was built and some attacker functions were proposed. A defense mechanism called Hide-and-Lie Strategy against such attacks was proposed, too. This mechanism has a free parameter with which the U-nodes can select between high privacy level and low data-forwarding overload. In our model, we analyzed the efficiency both of the attacks at different parameter values and the proposed defense mechanism. We showed that without any defense mechanism, the U-nodes are traceable, but with the proposed Hide-and-Lie Strategy, the success probability of an attacker can be decreased substantially. The message delivery ratio and the costs at different Hide-and-Lie parameter values are also investigated. We found that in some scenarios, the Hide-and-Lie Strategy can be viewed as a motivation for other U-nodes to carry messages that they are not interested in. Therefore, as a positive side effect, the message delivery ratio is also increased.

References

[1] "Trust and reputation management system definition," BIONETS (IST-2004-2.3.4 FP6-027748) Deliverable (D4.1), June 2007.

[2] "Towards Security in BIONETS," BIONETS (IST-2004-2.3.4 FP6-027748) Deliverable (D4.2), August 2007.

[3] "Identities, authentication, trust, and reputation in BIONETS," BIONETS (IST-2004-2.3.4 FP6-027748) Internal Deliverable (ID4.10), December 2008.

[4] "Authentication, privacy and anonymity," BIONETS (IST-2004-2.3.4 FP6-027748) Internal Deliverable (ID4.4), June 2007.

[5] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.

[6] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker, "Passive data link layer 802.11 wireless device driver fingerprinting," in *Proc. of the USENIX Security Symposium '06*, Berkeley, CA, USA, 2006, pp. 167–178.

[7] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proc. of the ACM MobiCom*, New York, NY, USA, 2008, pp. 116–127.

[8] S. Čapkun, R. Rengaswamy, I. Tsigkogiannis, and M. Srivastava, "Implications of Radio Fingerprinting on the Security of Sensor Networks," in *Proc. of the 3rd SecureComm*, 2007.

[9] A. Heinemann, "Collaboration in opportunistic network," Ph.D. dissertation, Fachbereich Informatik der Technischen Universität Darmstadt, 2007.

[10] A. Kate, G. Zaverucha, and U. Hengartner, "Anonymity and security in delay tolerant networks," in *Proc. of the 3rd SecureComm*, 2007, pp. 504–513.

[11] "ANSI X9.62: The elliptic curve digital signature algorithm (ECDSA)," 1998.

[12] R. Moskowitz and P. Nikander, "Host Identity Protocol (HIP) architecture," *IETF, RFC 4423*, May 2006.

[13] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *Proceedings of ACM SIGCOMM '07, Kyoto, Japan*, Aug. 2007.

[14] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, USA, 1997.

[15] A. Jøsang, "A logic for uncertain probabilities," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 9, no. 3, pp. 279–311, 2001.

[16] U. Maurer, "Modelling a public-key infrastructure," in *Proceedings of ESORICS '96*, vol. LNCS 1146, 1996, pp. 325–350.

[17] R. Cascella, "Enabling fast bootstrap of reputation in p2p mobile networks," in *Proc. 23rd IEEE AINA Conference, Bristol, UK*, May 2009.

[18] A. Garg, R. Battiti, and R. Cascella, "Reputation management: Experiments on the robustness of ROCQ," in *ISADS - First International Workshop on Autonomic Communication for Evolvable Next Generation Networks, Chengdu, China*, Apr. 2005.

[19] M. Tavakolifard, S. J. Knapskog, and P. Herrmann, "Trust transferability among similar contexts," in *4th ACM symposium on QoS and security for wireless and mobile networks (Q2SWinet)*, Oct. 2008.

[20] A. Garg and R. Battiti, "The reputation, opinion, credibility and quality (ROCQ) scheme," Università di Trento, Tech. Rep. DIT-04-104, Nov. 2004.

[21] M. Bellare, D. Micciancio, and B. Warinschi, "Foundations of group signatures: formal definition, simplified requirements and a construction based on trapdoor permutations," in *Advances in cryptology - EUROCRYPT*, vol. LNCS 2656, May 2003, pp. 614–629.

[22] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret: Theory and applications of ring signatures," in *ASIACRYPT*, Dec. 2001.

[23] Y. G. Desmedt, "Threshold cryptography," *European Transactions on Telecommunications*, vol. 5, no. 4, pp. 449–457, July-August 1994.

[24] G. D. Crescenzo, R. Ge, and G. R. Arce, "Improved topology assumptions for threshold cryptography in mobile and ad hoc networks," in *SASN'05: Proceedings of the 3rd ACM workshop on Security for ad hoc and sensor networks*. Alexandria, VA, USA: ACM Press, 2005, pp. 53–62.

[25] R. D. Pietro, L. V. Mancini, and G. Zanin, "Efficient and adaptive threshold signatures for ad hoc networks," *Electronic Notes in Theoretical Computer Science*, vol. 171, no. 1, pp. 93–105, 2007.

[26] D. Boneh and M. Franklin, "Identity based encryption from the weil pairing," *SIAM Journal of Computing*, vol. 32, no. 3, pp. 586–615, 2003.

[27] T. H. Yuen and V. K. Wei, "Constant-size hierarchical identity-based signature/signcryption without random oracles," Cryptology ePrint Archive, Tech. Rep. Report 2005/412, 2005.

[28] L. Buttyán, L. Dóra, M. Félegyházi, and I. Vajda, "Barter trade improves message delivery in opportunistic networks," *Elsevier Ad Hoc Networks*, 2009.

[29] "Denial-of-Service Resistant Data Dissemination in BIONETS," BIONETS (IST-2004-2.3.4 FP6-027748) Deliverable (D4.3), February 2008.

[30] S. Kanchi and D. Vineyard, "An optimal distributed algorithm for all-pairs shortest-path," *International Journal "Information Theories & Applications"*, vol. 11, no. 2, pp. 141–146, 2004.

[31] D. Chaum, “The dining cryptographers problem: unconditional sender and recipient untraceability,” *J. Cryptol.*, vol. 1, no. 1, pp. 65–75, 1988.

[32] D. Kesdogan and J. Egner and R. Buschkes, “Stop-and-Go-MIXes: Providing Probabilistic Anonymity in an Open System,” in *Proc. of the 2nd IH*, 1998, pp. 83–98.

[33] M. K. Reiter and A. D. Rubin, “Crowds: anonymity for web transactions,” *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, 1998.

[34] F. Dötzer, “Privacy Issues in Vehicular Ad Hoc Networks.” in *Proc. of the PET*, 2005, pp. 197–209.

[35] J.-P. Hubaux, S. Čapkun, and J. Luo, “The security and privacy of smart vehicles,” *IEEE Security and Privacy*, vol. 2, no. 3, pp. 49–55, 2004.

[36] M. Raya and J.-P. Hubaux, “Securing vehicular ad hoc networks,” *Journal of Computer Security*, vol. 15, no. 1, pp. 39–68, 2007.

[37] M. Gerlach, “Assessing and Improving Privacy in VANETs,” in *Proc of the 4th ESCAR*, November 2006.

[38] L. Lilien, Z. Kamal, V. Bhuse, and A. Gupta, “Opportunistic Networks: The Concept and Research Challenges in Privacy and Security,” in *Proc. of the WSPWN*, 2006, pp. 134–147.

[39] A. Narayanan and V. Shmatikov, “De-anonymizing Social Networks,” in *Proceedings of the 30th IEEE Symposium on Security and Privacy*, 2009.

[40] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, Inc., 1996.

[41] J. Hartigan, *Clustering algorithms*. John Wiley & Sons, Inc. New York, NY, USA, 1975.