Delphine Christin, Christian Roßkopf, Matthias Hollick, Leonardo A. Martucci, Salil S. Kanhere: *IncogniSense: An Anonymity-preserving Reputation Framework for Participatory Sensing Applications*. In: Pervasive and Mobile Computing, 9(3), pp. 353-371, 2013.

IncogniSense: An Anonymity-preserving Reputation Framework for Participatory Sensing Applications

Delphine Christin[1 a], Christian Roßkopf[a], Matthias Hollick[a], Leonardo A. Martucci[b], Salil S. Kanhere[c]

[a] *Secure Mobile Networking Lab, Technische Universität Darmstadt, Germany*
[b] *Department of Computer and Information Science, Linköping University, Sweden*
[c] *School of Computer Science and Engineering, University of New South Wales, Australia*

Abstract

Reputation systems are fundamental for assessing the quality of user contributions in participatory sensing. However, naively associating reputation scores to contributions allows adversaries to establish links between multiple contributions and thus de-anonymize users. We present the IncogniSense framework as a panacea to these privacy threats. IncogniSense utilizes periodic pseudonyms generated using blind signature and relies on reputation transfer between these pseudonyms. Simulations are used to analyze various reputation cloaking schemes that address the inherent trade-off between anonymity protection and loss in reputation. Our threat analysis confirms the robustness of IncogniSense and a prototype demonstrates that associated overheads are minimal.

Keywords:
Anonymity, reputation, cloaking, participatory sensing

1. Introduction

Recent mobile phones are equipped with a plethora of embedded sensors, and integrate widespread wireless technologies and complex processing capabilities. These technological features have contributed to the emergence of a new paradigm known as *participatory sensing* [1]. Participatory sensing applications involve volunteers collecting sensor readings from the surrounding environment using their mobile phones. The collected sensor readings are reported to application servers, where summaries are computed and published in the form of maps and statistics. Several practical systems based on this novel paradigm have been developed in recent years, which include creating noise pollution maps [2] and obtaining road traffic information [3].

Participatory sensing applications are however exposed to incorrect contributions due to their inherent open nature [4]. For example, participants may inadvertently position the phone in an undesirable position while collecting sensor readings (e.g., phone kept in bag while sampling street-level noise). Moreover, malicious participants may deliberately contribute bad data. Both behaviors result in erroneous contributions, which need to be identified and eliminated to ensure

[1] Mornewegstr. 32, 64293 Darmstadt, Germany, Phone: +49 6151 16-70922, Fax: +49 6151 16-70921, E-Mail: delphine.christin@seemoo.tu-darmstadt.de

the reliability of the computed summaries. For this purpose, reputation systems tailored to participatory sensing have been proposed such as [4]. They assign reputation scores to the participants based on the quality of their contributions and then use these scores to weed out bad contributions. Such systems however need to observe the contributions made by each device for an extended period of time to compute the reputation and hence, require linkability across multiple contributions from the same device. An adversary can exploit these links to de-anonymize the volunteers and compromise their privacy, since the sensor readings usually include spatiotemporal meta-data [5]. We herein propose a solution that addresses this inherent conflict between privacy and reputation requirements. Our specific contributions can be summarized as follows:

1. We present IncogniSense, an anonymity-preserving reputation framework based on blind signatures [6], which is agnostic to both the reputation assignment algorithm and the application. In IncogniSense, each user picks a new pseudonym for each time period, which is used to report sensor readings. Before the next period starts, the user transfers the reputation score associated with his current pseudonym to his next pseudonym. This allows the user to conserve his reputation across multiple periods, while limiting associations between his contributions to a unique period.
2. IncogniSense cloaks the reputation to be transferred to prevent an attacker from linking multiple pseudonyms. As reputation cloaking has an inherent trade-off between anonymity protection and loss in reputation, we explore this design space by undertaking a detailed simulation-based analysis of several cloaking mechanisms and we especially study their resilience against linking attacks. We significantly extend our work presented in [7] by analyzing the impact on anonymity protection of (1) transient vs. permanent reputation scores and (2) constant vs. variable client populations.
3. We conduct a thorough threat analysis showing the robustness of IncogniSense against reputation corruption.
4. We evaluate the feasibility of our approach by implementing a proof-of-concept on Android Nexus S mobile phones. Assuming that a new pseudonym is chosen every 5 minutes, IncogniSense only incurs an additional 2.3% of energy expenditure, which is a small price to pay for the enhanced privacy protection offered.

The paper is organized as follows. In Section 2, we discuss related work. We introduce our threat model in Section 3 and present the IncogniSense framework in Section 4. We analyze the robustness of IncogniSense against threats to reputation in Section 5 and conduct a multi-dimensional evaluation of IncogniSense in Section 6. We show the feasibility of our approach in Section 7, before making concluding remarks in Section 8.

2. Related Work

We compare IncogniSense with existing anonymity-preserving reputation systems designed for application domains orthogonal to participatory sensing, as this specific problem has not been addressed in the context of participatory sensing, other than a mention of its importance as future research [8]. To the best of our knowledge, we are the first to propose a concrete framework for this paradigm including a proof-of-concept implementation and a multi-dimensional evaluation. While this paper discusses our solution in the context of participatory sensing, IncogniSense has generic applicability and is not necessarily restricted to resource-constrained devices such as mobile phones.

Manifold reputation architectures based on pseudonyms have been proposed for peer-to-peer networks. For example, in [9], pseudonyms are created by Trusted Platform Modules and attested by Certificate Authorities. In the solution presented in [10], the users utilize a set of pseudonyms with the same reputation to collect electronic coins centralized at a third party. In contrast to our solution, both approaches are vulnerable to identity-based attacks since users can control an arbitrary number of pseudonyms. Another scheme presented in [11] proposes context-based, self-certified, and Sybil-free pseudonyms built using e-tokens [12]. The users, however, cannot change their pseudonyms once they are associated to a context. Finally, STARS [13] extends existing peer-to-peer reputation systems by traceability and anonymity. Complete anonymity of honest clients is however not guaranteed since the scheme reveals the identity of clients suspected to corrupt their reputation and is subject to false positives.

IncogniSense shares the most similarities with [14] and [15], which rely on periodic pseudonyms and transfer of reputation between pseudonyms. As our work builds upon the RuP algorithm [14], we present a detailed comparison in Section 4.3. In short, RuP requires the clients to create n temporary pseudonyms for each valid pseudonym and relies on probabilistic proofs. As such, a malicious client can create multiple pseudonyms and tamper with its reputation with a success probability of $1/n$. Increasing n reduces the threats to reputation manipulation, but increases associated overheads. In comparison to RuP, IncogniSense is robust against reputation corruption and the clients generate only one pseudonym per period. The protocol presented in [15], which extends the ideas from [14], is based on the concept of k-anonymity [16]. They assume the existence of a third party server which forms groups of clients sharing the same reputation and distributes a signature key per group to sign the pseudonyms. The clients must however trust each other not to collude with the third party and/or other clients to reduce the anonymity set. Moreover, the proposed protocol introduces additional overhead for the clients to create pseudonyms as compared to [14] and our solution.

3. Threat Model and Assumptions

In this section, we present our threat model and detail our assumptions. We consider that our adversary set includes malicious clients, application servers, and the *reputation and pseudonym manager* (RPM) (described in Section 4). The adversaries follow the Dolev-Yao threat model [17], i.e., they are able to listen to all communication, fabricate, replay, and destroy messages. They are, however, not able to break cryptographic mechanisms.

3.1. Threats to Reputation

The goal of the adversaries, primarily malicious clients, is to corrupt the reputation system in order to artificially increase their own reputation. Self-promotion can be achieved by means of Sybil attacks, in which the malicious client creates an arbitrary number of identities that vouch for each other [18]. Alternatively, malicious clients may replay old messages to gain reputation without contributing new sensor readings.

We assume that the application server and the RPM are protected against fraudulent access by well-established security mechanisms. Hence, adversaries are not able to access stored data, or change the behavior of applications. Denial of service attacks are considered out of the scope of this paper.

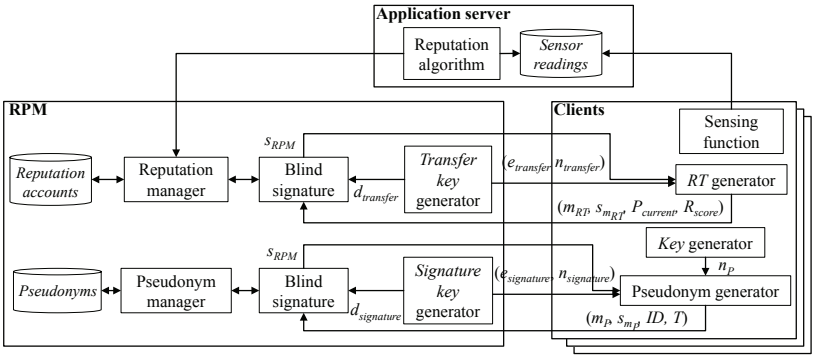


Figure 1: IncogniSense framework

3.2. Threats to Anonymity

Another goal of the adversaries is to breach the anonymity of the clients. They track the interactions of the clients with the reputation system and attempt to establish relationships between successive pseudonyms and link them to a unique real identity. We assume that the reported sensor readings do not contain any direct indication of the identity of the clients and that the interactions of the clients with the application server are anonymized using, e.g., disposable IP and MAC addresses and anonymous communication networks [19].

4. The IncogniSense Framework

We first provide an overview of the IncogniSense framework and detail the underlying mechanisms. Note that further details on the utilized blind signatures are available in Appendix A to Appendix C. We then present our reputation cloaking mechanisms and highlight the differences between IncogniSense and the framework proposed in [14], which forms the underlying basis for parts of our work.

4.1. Overview and Underlying Mechanisms

The IncogniSense framework illustrated in Fig. 1 includes clients, an application server, and a RPM (introduced in Section 3). We assume a participatory sensing application in which several active clients collect sensor readings and report them to the application server. In this scenario, our framework relies on two main mechanisms: (1) the utilization of periodic pseudonyms by the clients to report the collected sensor readings, and (2) the transfer of reputation scores between consecutive pseudonyms in order to conserve the reputation gained by the clients across multiple time periods. In particular, the proposed solution is comprised of the following four steps, which are repeated sequentially for each time period T.

4.1.1. Pseudonym Generation

We assume that each client has a permanent identifier ID, a private key PR, a public key PU, and is registered with the RPM. Clients generate pseudonyms in association with the RPM and based on blind signatures [6]. By employing blind signatures, the objective is two-fold: (1) to ensure that each client has a unique pseudonym for each T in order to prevent Sybil attacks

(cf. Section 3.1), and (2) to hide the generated pseudonym from the RPM in order to prevent it from linking the pseudonym to the client's real identity. In fact, blind signatures ensure the authenticity of signed messages without revealing their content to the signing entity and prevent the signing entity from linking the message content with the identity of its creator. For each pseudonym, each client generates a new key pair (PR_P, PU_P), with PR_P the private key and PU_P the public key. Note that the public key is available to all, while the private key is only known by the client who generated it. Next, the client hands over the public key to the RPM for blind signature as detailed in Appendix B. For each T, the RPM uses a different private key $PR_{signature}$ in the blind signature that determines the period of validity of the pseudonym. Moreover, the RPM only signs one pseudonym per client per time period in order to prevent Sybil attacks. As a result, the client uses the blindly signed pseudonym and the newly generated private key to report sensor readings to the application server and to transfer reputation to its next pseudonyms. If the RPM signs more than one new pseudonym for each T, the application server and the RPM cannot link the reported sensor readings to the real identity of the client.

4.1.2. Reporting of Sensor Readings

Within T, the client periodically reports sensor readings to the application server using its current pseudonym $P_{current}$ (i.e., the pseudonym valid in T). The application server verifies the validity of pseudonym with the RPM, evaluates the sensor reading using a reputation model, and attributes a reputation score R_{score} to $P_{current}$, with $R_{score} \in \mathbb{Z}$. The application server eventually makes use of the reputation scores to identify inaccurate contributions. Depending on the selected reputation model, the application server can discard the contributions with low reputation scores or reduce the weighting for such contributions in the computation of summaries. Note that the design of both reputation algorithm and model is considered out of the scope of this paper. Existing solutions such as [4] can however be easily integrated into our generic framework. Next, the application server transmits R_{score} to the RPM, which maintains reputation accounts for each pseudonym and hence, adds R_{score} to $P_{current}$'s reputation account.

4.1.3. Generation of Reputation Tokens

Before the expiration of $P_{current}$ (i.e., at the end of T), the client generates its next pseudonym P_{next} for the next time period as described in Section 4.1.1. It then transfers the gained reputation with $P_{current}$ to P_{next} in order to conserve it after the expiration of $P_{current}$. The reputation transfer is realized using *reputation tokens* RTs generated by the clients in collaboration with the RPM to prevent reputation corruption. The transfer process also makes use of blind signatures with P_{next} being the blinded content in order to prevent the RPM from linking the client's consecutive pseudonyms. In fact, each client withdraws reputation scores from $P_{current}$'s account and then, deposits them using RTs in P_{next}'s account, both accounts being maintained by the RPM. For each generated RT, the client using $P_{current}$ indicates the reputation score to be transferred in the RT to the RPM as detailed in Appendix C. The RPM verifies the balance of the reputation account of $P_{current}$ and decrements it accordingly. The RPM has different key pairs referred to as *transfer keys*, each of them being associated to a different reputation value. It thus uses the key pair corresponding to the reputation score to be transferred on the RT to blindly sign it. Consequently, the signing key (and not the content of the RT) determines the reputation value of each RT that prevent clients from manipulating RTs' value. However, the linking between $P_{current}$ and P_{next} is only prevented if more than one RT is generated per T. Otherwise, the linking is straightforward.

4.1.4. Reputation Transfer

The client registers with the RPM using P_{next} within the current T and hands over RT(s). The RPM verifies that each RT has not been used before and credits the reputation account of P_{next} from RT's value.

4.2. Cloaking Mechanisms

In theory, the utilization of blind signatures prevents the linking of consecutive pseudonyms. However, in practice, the reputation transfer between two consecutive pseudonyms may reveal insights about their succession. For example, consider the case where $P_{current}$ has the highest reputation among all pseudonyms. Now assume that after reputation transfer, this same reputation score is associated with the pseudonym, P_{next}. It is fairly straightforward for an adversary to establish a link between $P_{current}$ and P_{next}. An adversary able to track the reputation scores over several time intervals can link pseudonyms used in different periods. To prevent such attacks, we propose that the clients cloak their reputation scores before their transfer. Since the RPM prevents unjustified reputation promotion by controlling the generation of the RTs, the clients only transfer reputation scores lower or equal to their actual reputation. While cloaking the reputation scores prevents a reputation analysis attack, it may cause degradation in reputation score since it entails adding a perturbation to the same. In this section, we present three different reputation cloaking schemes, which address this tradeoff in different ways.

4.2.1. Floor Function Reputation Transfer (Floor)

This scheme divides the entire spectrum of reputation scores into fixed reputation intervals and classifies the clients into these reputation intervals based on their reputation scores. For the actual transfer, the clients use the floor value of the reputation interval as reputation score and transfer it using a single RT. Note that we use the floor value to prevent unjustified reputation promotion. Similar to the concepts of k-anonymity [16], the scheme forms groups of pseudonyms sharing the same reputation score. The larger the groups, the more indistinguishable the pseudonyms, and the harder it is for the adversary to link consecutive pseudonyms. The anonymity protection is thus determined by the size of the groups, which depends on the selection of the reputation intervals. Selecting large reputation intervals may increase the number of pseudonyms within a group, but may negatively affect the reputation scores due to the coarse granularity of the chosen reputation intervals. The size of the reputation intervals is therefore an important design parameter for addressing the tradeoff between anonymity and reputation.

4.2.2. Transfer of Random Sets from Reputation Partition Sets (RandSet)

RandSet divides the reputation score to be transferred into multiple RTs based on a set partitioning (e.g., (10, 50, 250)) used by all users. For each reputation transfer, the division of the reputation score into the set partitioning is determined by each client individually. For example, a client can partition a reputation score of 70 into one RT of value 50 and two RTs of value 10 or 7 RTs of value 10. The client then randomly selects one or more RTs handed over to the RPM. We refer to p as the probability that an individual RT is used to transfer reputation. In other words, each RT is discarded and not transferred with a probability $1 - p$. Both the division and discarding mechanisms increase the entropy of the actually transferred reputation and prevent the linkage between consecutive pseudonyms. However, discarding RTs can reduce the reputation scores. The diminution in reputation is linearly correlated with the size of the RTs, i.e. discarding a large RT leads to greater loss in reputation. In summary, both set partitioning

and discarding probability influence the tradeoff between anonymity and reputation, and thus need to be analyzed.

4.2.3. Transfer of Random Scores from Reputation Partition Sets (RandScore)

The initial reputation score is split into different *RT*s of predefined size as in *RandSet*. During the transfer, all the *RT*s are transferred. However, the transferred score of the individual *RT*s is lowered according to a random function. The set partitioning impacts on the entropy of the transferred scores and the loss in reputation. The range of the possible scores linearly increases with the size of the *RT*s. Simultaneously, the transferred reputation statistically diminishes, as the probability of transferring the initial *RT* values decreases, while the size of the *RT*s increases. Note that the spectrum of possible cloaking mechanisms is not limited to the aforementioned schemes. We have specially selected these schemes based on the diversity of their key parameters to address the tradeoff between anonymity protection and reputation degradation. Moreover, we have implemented the proposed reputation transfer schemes using blind RSA signatures as detailed in Section 7. However, this signature scheme prevents clients from lowering the value of the *RT*s without the knowledge of the RPM when applying the *RandScore* scheme and using permanent reputation scores. With permanent reputation scores, non-transferred reputation scores gained with $P_{current}$ are not lost at the end of T, but can be transferred to any future P_{next}. As a result, another blind signature scheme needs to be applied in this case. For example, the scheme proposed in [20] is based on the principles of divisible electronic cash and enables to split the value of a *RT* into different *coins*. Each coin can then be handed over to the RPM independently of the others. The coins cannot be linked to the identity of the user, but to the originating *RT*. Therefore, the RPM obtains additional insights about the *RT*s that may improve its capabilities in linking consecutive pseudonyms. We, however, consider both the evaluation as well as the implementation of this signature scheme as future work.

4.3. Comparison with the Existing Framework

We contrast IncogniSense with the framework proposed in [14], since our work improves its ideas. In particular, we highlight weaknesses of the RuP algorithm in terms of cryptographic overhead and vulnerability to reputation manipulation and argue how IncogniSense addresses these issues.

4.3.1. Algorithmic Differences

In the RuP algorithm, the client includes the time interval of validity of the pseudonym in the blind signature. Each client creates n pseudonyms to generate a sole valid pseudonym. The n pseudonyms are transmitted to the RPM, which randomly selects $n-1$ pseudonyms and requests the client to send the corresponding random values r^{-1}. The RPM then encrypts the $n-1$ pseudonyms and verifies that they are valid for the same time interval. If the verification is successful, it signs the nth pseudonym, which becomes the valid pseudonym. The generation of *RT*s also necessitates that the client prepares n messages and the RPM verifies $n-1$ messages before signing the nth message. In IncogniSense, we decouple the time interval of validity of the pseudonym and the value of the reputation to transfer from the blind signatures by introducing periodic *signature keys* and different *transfer keys* for each value of *RT*, reciprocally.

4.3.2. Cryptographic Overhead

In [14], the client generates n key pairs, selects n values r and r^{-1}, and executes n encryptions to generate one valid pseudonym. The generated key pairs of non-selected pseudonyms should not be reused to prevent the RPM from linking the pseudonyms to the client's identity. In IncogniSense, the client prepares one blind signature per pseudonym, i.e., it generates only one key pair and encrypts one message, and the RPM verifies only one signature per time interval and per client (instead of $n - 1$), which significantly reduces the computational overhead if we assume a large base of clients. Similar conclusions can be drawn for the *RT* generation, except that no key pair is generated by the client.

4.3.3. Reputation Manipulation

In the RuP algorithm, the RPM cannot verify the time interval of validity of the pseudonym included in the blinded message. It randomly verifies $n - 1$ of the n generated pseudonyms and signs the nth, hoping that it contains the same time interval of validity as the verified pseudonyms. This implies that malicious clients can generate multiple pseudonyms for a given time interval with a probability of $1/n$. If such an attack is successful, then the adversaries can seriously compromise the reputation system (see Section 3.1). Likewise, malicious clients can generate fraudulent *RT*s and increase their reputation. IncogniSense completely eliminates the possibility of Sybil attacks and also prevents adversaries from compromising the reputation transfer process. The utilization of periodic *signature keys* and the verification of already existing pseudonyms by the RPM guarantees that each client has a unique pseudonym per time interval. Similarly, the utilization of different *transfer keys* allows the RPM to easily verify and guarantee the value of the transferred reputation, preventing malicious clients from generating falsified *RT*s.

In summary, IncogniSense achieves better protection against reputation manipulation, while significantly reducing the cryptographic overhead for the client.

5. Analysis of Robustness against Threats to Reputation

We consider the assumptions and threat model presented in Section 3 and argue that IncogniSense is resilient against the following attacks.

5.1. Sybil Attacks

Malicious clients can attempt to generate multiple pseudonyms for a given time interval to increase their reputation through cross-recommendations. However, IncogniSense is protected against these attacks since the RPM maintains a list of the clients that have presented pseudonyms for blind signature along with their validity interval. Once a pseudonym has been signed for a given time interval, the RPM discards all other pseudonyms submitted by the same client.

5.2. Replay Attacks

Malicious clients may attempt to replay old messages for the following two reasons: (1) artificially increase their reputation by replaying *RT*s, (2) debit the reputation account of honest clients without the victims receiving the associated credit by replaying the message to be signed. The RPM, however, maintains a list of IDs associated with each *RT* (see Section 4.1.4). The RPM can thus detect malicious clients trying to corrupt the reputation system and hence, prevents both attacks. Malicious clients are prevented from forging *RT*s and manipulating the reputation of other clients, as the creation of the *RT*s requires the signature of the corresponding pseudonym, which is only possible using the pseudonym's private key.

Table 1: Probability distribution of transient reputation scores

Attribution probability	0.25	0.35	0.25	0.10	0.05
Reputation scores	10	5	0	-5	-10

5.3. Manipulation of Reputation Accounts

Malicious clients can attempt to alter the reputation stored in the RPM. The clients, however, do not have direct access to their reputation accounts and the RPM is protected against unauthorized access using standard cryptographic primitives. As such, this attack cannot be launched.

5.4. Reporting of Falsified Sensor Readings

Malicious clients can try to report falsified sensor readings on behalf of others to degrade their reputation. IncogniSense protects clients against this attack by requesting clients to authenticate with the application server and the RPM. These entities verify the validity of the pseudonyms before considering their contributions and/or delivering them information. Such an attack would only be successful if malicious clients can access the private keys of the targeted clients and those of their respective pseudonyms, which is beyond the scope of our attacker model.

In summary, we have shown that IncogniSense is robust by design against a variety of threats directed against the reputation system by malicious clients.

6. Analysis of Anonymity Protection

We analyze the resilience of IncogniSense against the threats to anonymity identified in Section 3.2. In particular, we measure the level of anonymity, i.e. unlinkability, that can be achieved by the different reputation cloaking schemes introduced in Section 4.2. In a first step, we assume that the reputation scores are lost if they are not transferred to the next pseudonym at the end of the period. We hence quantify the reduction in reputation score caused by the different cloaking approaches and investigate the trade-off between anonymity protection and loss in reputation. Next, we assume that the clients can store reputation for future use and analyze the impact of stored reputation scores on anonymity protection for both a constant and variable population of clients.

6.1. Transient Reputation Scores

In the following, we assume that the reputation scores are transient, i.e., their validity duration is limited to the duration of the period in which they have been gained, and we adopt the following simulation setup and method.

6.1.1. Simulation Setup and Method

We implemented a simulator in Java to model the behavior of clients, RPM and application server. Each simulation run is for 100 time intervals. We repeat each simulation 100 times and present the averaged results. All clients remain active for the duration of the simulation. During each time interval, the clients generate 5 random sensor readings and report them to the application server. The application server runs a simulated reputation algorithm, which randomly attributes reputation scores to the pseudonyms according to the distribution presented in Table 1. Note that the actual values of both sensor readings and reputation scores do not impact the performances of the cloaking schemes in terms of linkability and can thus be selected

Algorithm 1 Identification of potential successors for transient reputation scores

Require: $R_{P_{current}}$: $P_{current}$'s reputation, $R_{P_{next}}$: P_{next}'s reputation, $S_{RT_{created}}$: set of RTs created by $P_{current}$, $S_{RT_{used}}$: set of RTs used by P_{next}

1: **for all** $P_{current} \in S_{P_{current}}$ **do**
2: **for all** $P_{next} \in S_{P_{next}}$ **do**
3: **if** $R_{P_{next}} \leq R_{P_{current}}$ **and** $S_{RT_{used}} \subseteq S_{RT_{created}}$ **then**
4: create a link between $P_{current}$ and P_{next}
5: **end if**
6: **end for**
7: **end for**

randomly. Moreover, we consider the study of additional reputation models and distributions as future work.

We adopt the assumptions of our threat model (see Section 3) and assume that the RPM and the application server are malicious internal observers. We further assume that adversaries collude and aim at linking consecutive pseudonyms. For each simulated time interval, the RPM and the application server have access to the following information. The RPM observes the generated RTs, the utilized RTs, and the updates of the reputation accounts, while the application server observes the current pseudonyms, the initial and final reputation scores, and the updates of reputation scores.

Based on the observed information, the adversaries identify the set of pseudonyms active in each time interval. We refer to $S_{P_{current}}$ as the set of pseudonyms active in a given time interval and $S_{P_{next}}$ as the set of pseudonyms active in the subsequent time interval. As all clients remain active for the duration of the simulation, $|S_{P_{current}}| = |S_{P_{next}}|$ and there exists a bijection between the sets $S_{P_{current}}$ and $S_{P_{next}}$. For each $P_{current}$, the adversaries first identify all potential successors and create links between $P_{current}$ and the identified P_{next}s based on their reputation scores and the created and used RTs as detailed in Alg. 1. Note that the second condition in step 3 ($S_{RT_{used}} \subseteq S_{RT_{created}}$) is verified if the RT_{used}s' values are lower than those of the $RT_{created}$s if the *RandScore* scheme is applied. For the other schemes, the same condition is verified if the values are equal. Next, they iteratively eliminate the created links based on the bijection between $S_{P_{current}}$ and $S_{P_{next}}$ according to Alg. 2. Alg. 2 first searches for single links between $S_{P_{current}}$ and $S_{P_{next}}$. Due to the bijection between both sets, all other existing links of $P_{current}$ and P_{next} are removed (step 6). $P_{current}$ and P_{next} are then flagged (step 7) and the link between both pseudonyms is confirmed (step 8). This indicates that P_{next} has been identified as the successor of $P_{current}$. Flagged pseudonyms and confirmed links are then excluded from further searches. When all existing single links have been successively removed, Alg. 2 analyses the remaining pseudonyms and searches for $P_{current}$s having the same potential successors (step 12). This search returns one or several subsets of $S_{P_{current}}$ sharing the same successors. The algorithm then selects the smallest subset whose size is equal to the number of common successors (step 13). It removes all other links of these pseudonyms (step 14), flags the pseudonyms (step 15), and confirms the links (step 16). The analysis of sets of pseudonyms having the same successors is interrupted and the algorithm restarts at step 2 with the search for single links, which potentially appear while removing the previous links.

Finally, as a measure of the level of anonymity provided by the reputation transfer, we calculate how many potential successors x_k (i.e., links) a pseudonym k has on average. We express the amount of potential successors as the fraction of the total number of clients $N = |S_{P_{current}}|$. For each time interval, we calculate this metric as follows.

Algorithm 2 Elimination of potential successors for transient reputation scores

Require: S_{Links}: set of links created in Alg. 1
1: $linkConfirmed \leftarrow true$
2: **while** ∃ non-flagged pseudonym ∈ ($S_{P_{current}} \cup S_{P_{next}}$) **and** $linkConfirmed$ = true **do**
3: $linkConfirmed \leftarrow false$
4: **for all** non-flagged pseudonym ∈ ($S_{P_{current}} \cup S_{P_{next}}$) **do**
5: **if** ∃ a single link between $P_{current}$ and P_{next} **then**
6: remove all other links of $P_{current}$/P_{next}
7: flag $P_{current}$ and P_{next}
8: $linkConfirmed \leftarrow true$
9: **end if**
10: **end for**
11: **if** $linkConfirmed = false$ **then**
12: search for other non-flagged $P_{current}$s having the same potential successors
13: **if** number of found $P_{current}$s = number of common P_{next}s **then**
14: remove all other links of the found $P_{current}$s and P_{next}s
15: flag the found $P_{current}$s and P_{next}s
16: $linkConfirmed \leftarrow true$
17: **break** search
18: **end if**
19: **end if**
20: **end while**

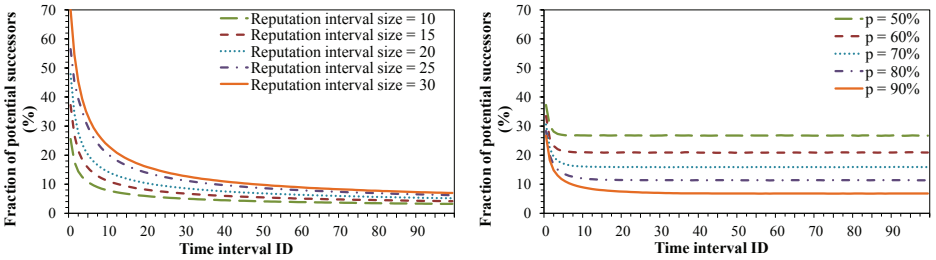
$$\frac{1}{N} \cdot \sum_{k=1}^{N} x_k \tag{1}$$

A value of 1 implies perfect anonymity achieved during the transfer, since all pseudonyms within $S_{P_{next}}$ are potential successors to each individual pseudonym of $S_{P_{current}}$. The lower the value, the smaller the set of pseudonyms that are the potential successors, which in turn implies that it may be easier to establish a link between consecutive pseudonyms. We can therefore directly assess the quality of the anonymization achieved by the cloaking schemes using this metric.

6.1.2. Evaluation Results

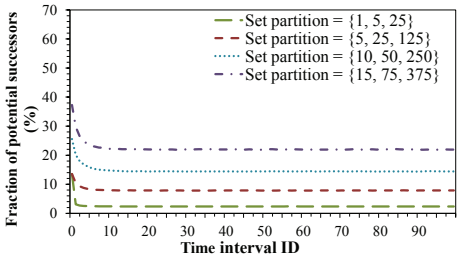
By applying the above setup and method, we first individually evaluate the reputation cloaking schemes (see Section 4.2). In particular, we analyze the effect of the most important configuration parameter relevant to each scheme and the resulting impact on the anonymity of the clients. Secondly, we compare how these schemes fare against each other. In this comparison, we use the transfer scheme used in [14] as a baseline. This scheme, also referred to as the *Full Reputation Transfer* (*Full*) scheme, always transfers the entire reputation in a single *RT*. It hence represents the worst case in terms of anonymity protection, since the reputation score remains the same for consecutive pseudonyms, which allows for easy linkability by the adversaries. Note that we compare the cloaking schemes based on selected variants that show a good tradeoff between anonymity protection and loss in reputation.

In the *Floor* scheme, the clients use the floor value of given reputation intervals as reputation score and transfer it using a single *RT*. Fig. 2(a) illustrates the influence of different reputation interval sizes on the quality of the anonymity protection. The results show that the choice of large



(a) Floor: Fraction of potential successors over time for selected sizes of reputation interval

(b) RandSet: Fraction of potential successors over time for different discarding probabilities $1 - p$ and a partition set of (10, 50, 250)

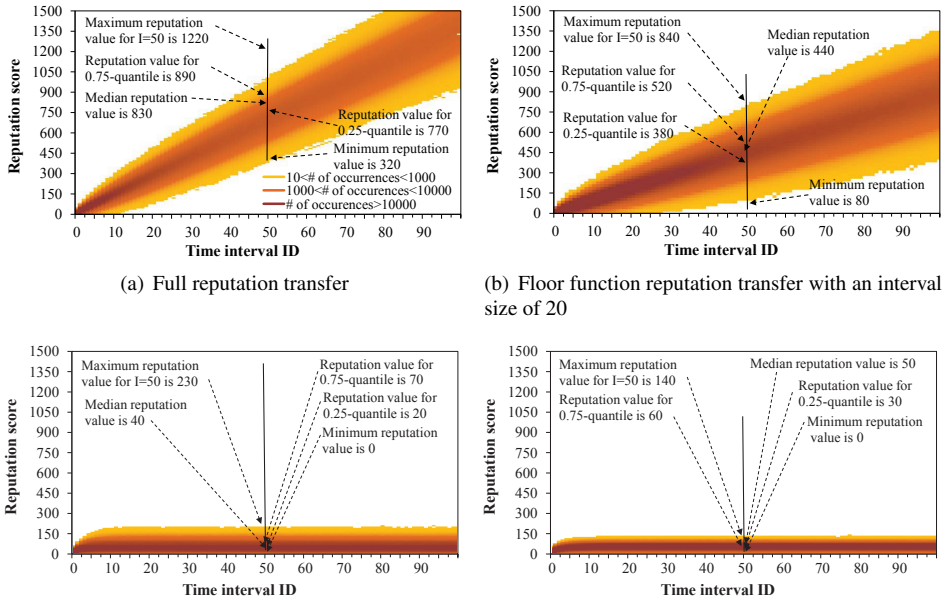


(c) RandScore: Fraction of potential successors over time for different set sizes and a reduction in reputation between 0% and 50%

Figure 2: Level of unlinkability for reputation transfer schemes Floor, RandSet, and RandScore for a constant population of 100 clients

reputation intervals results in a higher level of protection, as more clients are grouped within the same reputation interval and therefore become indistinguishable during the *RT* transfer. This protection comes at the expense of greater perturbation in reputation for some of the clients, since the difference between the actual reputation value and the floor of the reputation interval also increases on average. Over time, the protection level decreases, since the reputation values of the individual clients spread over wider reputation intervals as shown in Fig. 3(b) for a reputation interval size of 20. Fig. 3 shows the development of reputations scores over time and is annotated with the 0.25- and 0.75-quantile of reputation scores of the clients for time interval 50. As a result, individual clients are more easily distinguishable in our attacker model, since potential successors can be identified more easily. The *Full* scheme shows a similar behavior (cf. Fig. 3(a)), but is able to preserve higher reputation scores. This is illustrated in Fig. 5 that shows the observed diminution in reputation scores per time interval caused by the cloaking for selected variants of the proposed cloaking schemes that show a good tradeoff between anonymity protection and loss in reputation. With both schemes, clients are therefore easy to identify after a number of rounds since only a small number of peers have similar reputation scores.

In the *RandSet* scheme, the reputation score is partitioned into different *RT*s of predefined size and a reputation transfer is performed with probability p for each *RT*. The partition in *RT*s is implemented as follows. The clients first generate as many *RT*s with the highest value of the set of partition as possible. Next, they use the second highest and so on, until the remaining reputation becomes lower than the lowest value of the set partition. Fig. 2(b) illustrates the in-



(a) Full reputation transfer

(b) Floor function reputation transfer with an interval size of 20

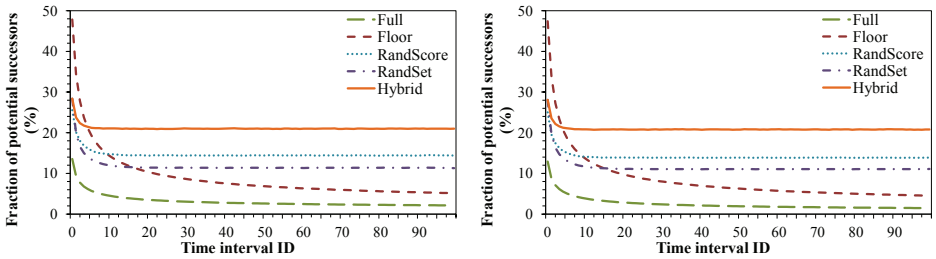
(c) Random set reputation transfer with p=80%

(d) Random score from set reputation transfer with a partition set of (10, 50, 250) and p=75%

Figure 3: Distribution of reputation score amongst clients over time for a constant population of 100 clients

fluence of the choice of p using the set partitioning (10, 50, 250) on the anonymity protection level. The fraction of potential successors stabilizes early, which means that attackers do not gain further evidence even if monitoring for extensive periods of time. This is due to the reputation scores, which are not constantly increasing over time, but fluctuate in a certain reputation interval as shown in Fig. 3(c) for a probability p of 80%. Indeed, at a certain time instant the average increase in reputation due to the contribution to the application equals the average loss due to the discarding of reputation during the transfer. In contrast, the *Full* and *Floor* schemes constantly increase the reputation values as shown in Fig. 3(a) and 3(b). The percentage of potential successors using *RandSet* lies between 7% and 26% for p equal to 90% and 50%, respectively (see Fig. 2(b)). The degree of anonymity protection increases with decreasing p, i.e., increasing discarding probability $1 - p$. Fig. 5 shows the preserved reputation score for p equal to 80%. Obviously, for increasing p, the preserved reputation per round increases likewise. Note that the preserved reputation stays slightly below p, since the partitioning of the sets might not exactly fit the exact reputation value.

In the *RandScore* scheme, the reputation score is also partitioned into several *RT*s, which are all used in the reputation transfer. The transferred score is, however, lowered according to a random function. We analyze the influence of different set partitions while lowering between 0% and 50% (following an uniform distribution) of the reputation score of a *RT*. Fig. 2(c) shows that the level of anonymity protection increases with the set partitioning moving towards larger *RT* values. The smaller the lowest *RT* size in a set (1, 5, 10, 15), the worse the protection of anonymity, since individual clients can now be tracked by the number of *RT*s exchanged (but not the random reputation score exchanged) as detailed in Alg. 2. Similar to *RandSet*, the per-



(a) 100 clients (b) 300 clients

Figure 4: Scheme comparison and impact of the number of clients on the percentage of successors. *Floor* uses an interval size of 20. *RandScore* uses a partition set of (10, 50, 250) and p=75%. *RandSet* uses p=80%. *Hybrid* uses a partition set of (10, 20, 250) and p=80%.

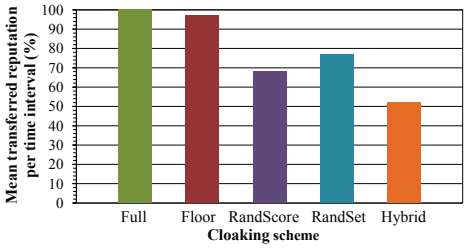


Figure 5: Comparison of transferred reputation for a constant population of 100 clients. *Floor* uses an interval size of 20. *RandScore* uses a partition set of (10, 50, 250) and p=75%. *RandSet* uses p=80%. *Hybrid* uses a partition set of (10, 20, 250) and p=80%.

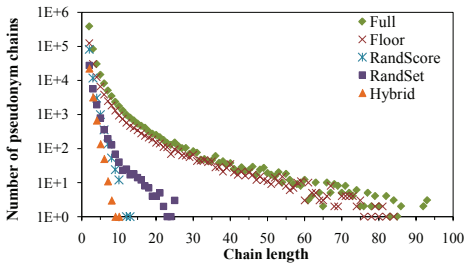


Figure 6: Number of identified pseudonym chains for a constant population of 100 clients. *Floor* uses an interval size of 20. *RandScore* uses a partition set of (10, 50, 250) and p=75%. *RandSet* uses p=80%. *Hybrid* uses a partition set of (10, 20, 250) and p=80%.

formance quickly stabilizes since the reputation values inherently stay within certain bounds as shown in Fig. 3(d), making identification very hard even for sophisticated attackers. Note that *RandSet* and *RandScore* perform similarly, if the same partitioning is applied and p equals the mean of the score discarding probability used in *RandScore*. Additionally, *RandSet* behaves similarly to the *Full* scheme for very low discarding probabilities (< 1%). In other words, reputation continuously increases for very low discarding probabilities. In contrast, reputation stabilizes within reputation bounds for greater probabilities. The stabilization time, however, depends on the discarding probability: the greater the probability, the faster the stabilization. For example, the reputation reaches its upper bound after 80 time intervals for a discarding probability of 5%.

For comparison purposes, we select one variant of each scheme that demonstrated a good tradeoff between anonymity protection and reputation loss based on the previous analysis. The following results thus only hold for these variants. In the following, *Floor* utilizes a reputation interval size of 20; *RandScore* partitions the *RT* sets into: 10, 50, 250 and has a mean value of the discarding probability of 25%; *RandSet* retains *RT*s with a probability p of 80% and utilizes a similar partitioning (10, 50, 250). To model a participatory sensing scenario, we investigate populations of 100 and 300 clients. We present the results in Fig. 4 for all proposed reputation cloaking schemes for both scenarios under consideration. We include an additional scheme referred to as *Hybrid*, which combines the *RandSet* and *RandScore* schemes. The *Hybrid* scheme

utilizes the partitioning (10, 50, 250) and retains *RT*s with a probability p of 80%. The reputation score of the retained *RT*s is randomly lowered as in the *RandScore* scheme. The *Hybrid* scheme is robust against observers that can track the number of *RT*s or the reputation scores obtained/transferred, since it cloaks both values during transfer. As seen from both graphs in Fig. 4, all schemes are rather robust when increasing the client base from 100 to 300. In summary, the *Hybrid* scheme provides the best anonymity protection followed by *RandScore*, *RandSet*, *Floor*, and the *Full* scheme. *Hybrid*, *RandScore*, and *RandSet* provide a constant protection faster than *Full* and *Floor* due to the constantly increasing reputation observed for the latter schemes.

Fig. 6 shows the number and the length of the pseudonym chains, i.e., the number of identified consecutive pseudonyms, for all schemes and 100 clients. Note that chains of length two represent identified links between two consecutive pseudonyms. Schemes that allow for long pseudonym chains are weaker than schemes that only allow the attacker to identify short chains. The *Hybrid* scheme is the most resilient scheme against attacks since it only allows adversaries to build very short chains (always shorter than 10). The *RandScore* (no chains longer than 13) and the *RandSet* (no chains longer than 25) have a vast majority of unlinkable clients, which is in line with our previous observations of a good anonymity protection. The *Floor* and the *Full* schemes perform worst in terms of identified chain length and number of identified chains. In a few cases, they allow adversaries to identify chains of up to 84 and 93 successive pseudonyms, respectively. Since the reputation scores with *Full* and *Floor* constantly increase, it is much easier for an attacker to follow individual pseudonyms. As a countermeasure, one could limit the maximum reputation score a client can obtain. The introduction of an upper bound would help to keep well behaving clients together in terms of reputation scores, thus making it much harder for an attacker to build pseudonym chains. However, a study of the effects of upper bounds is considered out of the scope of this work.

The anonymity of the clients is, however, protected at the price of a reduction in reputation as shown in Fig. 5. Clients utilizing the *Hybrid* scheme suffer the highest loss in reputation, followed by the *RandScore* and the *RandSet* schemes. The *Full* and the *Floor* schemes manage to transfer nearly the complete reputation scores.

In summary, we have demonstrated that some of the presented schemes can provide an effective anonymity protection, even if adversaries are able to track the reputation transfer between pseudonyms for long periods of time. We have shown that the cost of a high degree of anonymity protection is the loss of reputation. If we assume that all clients adopt the same cloaking scheme, all clients will be similarly affected by the incurred loss in reputation. This is consistent with the typical use of reputation in participatory sensing applications, where the application server is mainly interested in comparing the reputation scores of different clients and correspondingly associates a weight to their sensor readings. In the rare case where the absolute reputation score is of interest, the users may consider using a scheme, such as *Full* or *Floor*, which preserves the reputation value but is more susceptible to attacks. However, if minor perturbations in the scores are acceptable, then schemes such as *RandScore*, *RandSet*, and *Hybrid* may be considered.

6.2. Permanent Reputation Scores

Next, we assume permanent reputation scores, i.e. the reputation scores have an infinite period of validity. As such, in contrast with the previous scenario, the reputation cloaking schemes do not result in a loss of reputation at the end of each period, since the cloaked reputation can be stored for future use in subsequent time periods. In consequence, the adversaries need to consider the evolution of the reputation over several periods of time (instead of one period in the previous scenario). As the clients determine both the amount of transferred reputation and the

Table 2: Probability distribution of permanent reputation scores

Previous reputation score \ New reputation score	-10	- 5	0	5	10
-10	0.35	0.30	0.25	0.08	0.02
-5	0.25	0.30	0.30	0.10	0.05
0	0.05	0.10	0.25	0.35	0.25
5	0.05	0.10	0.30	0.30	0.25
10	0.02	0.08	0.25	0.30	0.35

time of the transfer, the number of possibilities for the reputation transfer increases. This in turn increases complexity of the reputation analysis, thus making it harder for potential adversaries to de-anonymize users. In order to investigate the impact of permanent reputation scores on the anonymity protection, we adopt the following simulation setup and method. In particular, we highlight the differences with the previous evaluation detailed in Section 6.1.1.

6.2.1. Simulation Setup and Method

We investigate a population of 100 clients. Each simulation run covers 100 time intervals (if not noted otherwise) and we present the average results of 1000 simulations. The application server attributes reputation scores to the pseudonyms according to the distribution presented in Table 2. Instead of arbitrarily attributing a random reputation score as in Section 6.1.1, the application server models possible behaviors of the clients by considering the last attributed reputation score in the computation of the new reputation score. The chosen distribution reflects our assumption that the participant behavior is unlikely to change at such short time scales. We thus assume that the probability that a client contributes a correct sensor reading is greater when it has already contributed correct samples and vice versa. For example, assuming that a client obtains a reputation score of 10 for its last sensor reading, the probability that it obtains the same reputation score for its new sensor reading is equal to 0.35. In contrast, the probability that it obtains a reputation score of -10 is equal to 0.02. Note that we could apply the same distribution in the previous scenario, but the evolution of the reputation scores over time has a lower impact on anonymity protection due to the transient nature of the reputation scores.

Again, we assume that the RPM and the application server are malicious internal observers with the aim of identifying consecutive pseudonyms based on the information they have access to. The adversaries first identify the set of pseudonyms active in each time interval. $S_{P_{current}}$ is the set of pseudonyms active in a given time interval and $S_{P_{next}}$ is the set of pseudonyms active in the subsequent time interval. For each pseudonym $P_{current}$, the adversaries attempt to identify its successor P_{next} as follows. The adversaries first identify the set of possible successors of $P_{current}$ based on their reputation value and establish a temporary link between $P_{current}$ and each candidate of $S_{P_{next}}$ according to Alg. 3. Next, the adversaries identify all pseudonyms in $S_{P_{current}}$ that have a single link to a pseudonym in $S_{P_{next}}$ and eliminate all other links, since a pseudonym has at most one successor. Similarly, they identify all pseudonyms in $S_{P_{next}}$ having a single link to a pseudonym in $S_{P_{current}}$ and eliminate all other links, as a pseudonym has at most one predecessor (see steps 4 to 8 in Alg. 4). Note that we compute the number of potential successors x_k (i.e., links) a pseudonym k has on average after this step. For the remaining multiple links, the adversaries compute the probability of each link based on the probability that $P_{current}$ handed RTs to the RPM and the probability that P_{next} shows a behavior similar to $P_{current}$ (steps 16 and 23 in Alg. 4). The probability that P_{next} shows a behavior similar to $P_{current}$ is determined using the probability distribution presented in Table 2 (known by the application server which uses it

Algorithm 3 Identification of potential successors for permanent reputation scores

Require: $R_{P_{current}}$: $P_{current}$'s reputation, $R_{P_{next}}$: P_{next}'s reputation, $S_{RT_{created}}$: set of RTs created by $P_{current}$, $S_{RT_{hidden}}$: set of RTs not used by $P_{current}$, $S_{RT_{used}}$: set of RTs used by P_{next}

1: **for all** $P_{current} \in S_{P_{current}}$ **do**
2: **for all** $P_{next} \in S_{P_{next}}$ **do**
3: **if** $R_{P_{current}} + Value(R_{P_{hidden}}) \geq R_{P_{next}}$ **and** $S_{RT_{used}} \subseteq (S_{RT_{created}} \cup S_{RT_{hidden}})$ **then**
4: create a link between $P_{current}$ and P_{next}
5: **end if**
6: **end for**
7: **end for**

for the computation of the reputation scores), while the probability that $P_{current}$ handed RTs to the RPM is computed based on the probability p of using a RT, the number of transferred RTs k, and the total number of RTs n, as follows.

$$p(k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} \tag{2}$$

For each $P_{current}$ having multiple links, the probability of each link is normalized by the sum of the probability of all links. The link with the greatest probability is then chosen as the potential successor of $P_{current}$. The other links are removed. The adversaries continue the reputation analysis by analyzing the remaining links according to the above steps (i.e., identification of single links and weighting of multiple links) until each pseudonym of $S_{P_{current}}$ has at most one identified successor in $S_{P_{next}}$. At this stage, we calculate the length of the established pseudonym chains and the fraction of successors correctly identified by the adversaries in each time interval.

6.2.2. Evaluation Results

We first investigate the impact of permanent reputation scores on the anonymity protection for a constant population of 100 clients based on the above setup and method. For this analysis, we consider that a reputation transfer is performed with probability p for each RT as in the *RandSet* scheme, except that the cloaked reputation is not lost at the end of the period but stored for future use. Fig. 7 illustrates the influence of the choice of p on the fraction of potential successors over time. Note that p=100% corresponds to a full transfer of reputation scores, i.e., no reputation cloaking is applied. The results show that applying reputation cloaking significantly improves anonymity protection. In particular, the degree of anonymity protection increases with decreasing p. For p=100%, the percentage of potential successors is under 2% from t=17, whereas it lies between 49% and 90% for p equal to 90% and 50%, respectively. In comparison, using transient reputation scores, the percentage of potential successors is between 7% and 26% for the same values of p (see Fig. 2(b)). The degree of anonymity protection thus considerably increases when using permanent reputation scores. This is due to the reputation scores stored by the clients, which individual value is unknown at both the RPM and the application server. Fig. 9 highlights the impact of the stored reputation on the distribution of the reputation scores. Fig. 9(a) presents the distribution of the reputation scores handed to the RPM to credit the reputation accounts and the reputation scores attributed by the application server. In comparison, Fig. 9(b) shows the distribution of all reputation scores present in the system, i.e., the reputation scores stored by the clients for future use and those illustrated in Fig. 9(a). While adversaries know the total value of stored reputation scores, they however need to infer the individual value stored by each client

Algorithm 4 Elimination of improbable successors for permanent reputation scores

Require: S_{Links}: set of links created in Alg. 3

while $\exists$ non-confirmed links $\in S_{Links}$ **do**
  $linkConfirmed \leftarrow false$
  for all non-flagged pseudonym $\in (S_{P_{current}} \cup S_{P_{next}})$ **do**
    if $\exists$ a single link between $P_{current}$ and P_{next} **then**
      remove all other links of $P_{current}$ and P_{next}
      flag $P_{current}$ and P_{next}
      update P_{next}'s hidden reputation
      $linkConfirmed \leftarrow true$
    end if
  end for
  if $linkConfirmed = false$ **then**
    search for other non-flagged $P_{current}$s having the same potential successors
    if number of found $P_{current}$s = number of common P_{next}s **then**
      remove all other links of the found $P_{current}$s and P_{next}s
      flag the found $P_{current}$s and P_{next}s
      select for each $P_{current}$ one of the found P_{next}s based on probability weighting
      update P_{next}s' hidden reputation
      $linkConfirmed \leftarrow true$
      break search
    end if
  end if
  if $linkConfirmed = false$ **then**
    select the most likely link based on probability weighting
    remove all other links of the corresponding $P_{current}$ and P_{next}
    flag $P_{current}$ and P_{next}
    update the hidden reputation of P_{next}
    $linkConfirmed \leftarrow true$
  end if
end while

each time a successor is selected according to the above steps. Fig. 8 illustrates the success rate of the adversaries in inferring the value of the stored reputation and hence, the correctly identified successors. In absence of reputation cloaking, adversaries are able to correctly identify up to 60% of potential successors and to establish very long chains of unique predecessor-successor relations (up to a length of 92 in few cases) (see Fig.10). In contrast, applying reputation cloaking reduces the fraction of correctly identified successors to 7% and 14% for p equal to 90% and 50%, respectively. Additionally, it leads to the construction of fewer and shorter pseudonym chains (up to a length of 75 for p=90%). We, however, observe that 99.9% of the pseudonyms are involved in chains of length of at most 4 pseudonyms. Reputation cloaking therefore effectively protects the anonymity of the clients, as the adversaries are able to link most of the pseudonyms for only short periods of time. Note that the chain lengths obtained for p=80% cannot be directly compared to those obtained for the *RandSet* scheme (see Fig. 6), as they are computed once the adversaries have weighted and selected one of the multiple links—a step not included in the first evaluation. In summary, the degree of anonymity protection is increased by both the utilization of permanent reputation scores and the application of reputation cloaking. Additionally, reputation cloaking does not incur loss in reputation as in our transient scheme, since the clients can store

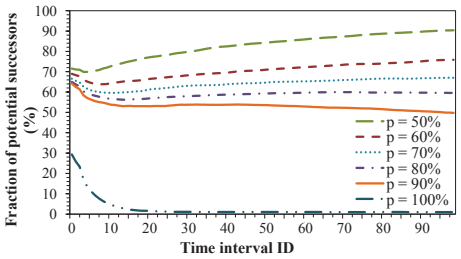


Figure 7: Fraction of potential successors over time for different transfer probabilities p and a constant population of 100 clients

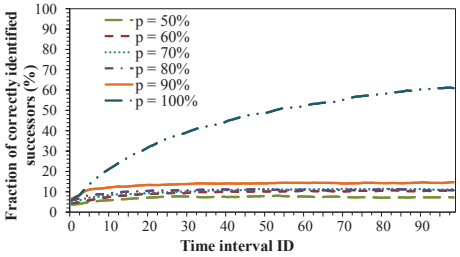
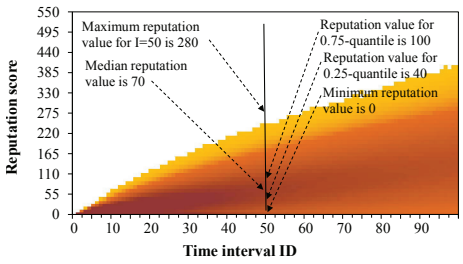
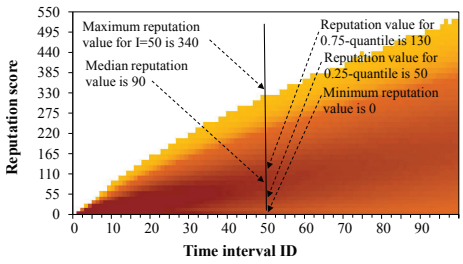


Figure 8: Fraction of correctly identified successors over time for different transfer probabilities p and a constant population of 100 clients



(a) Transferred and attributed reputation scores



(b) Total reputation scores

Figure 9: Distribution of reputation scores over time amongst clients for p=80% and a constant population of 100 clients

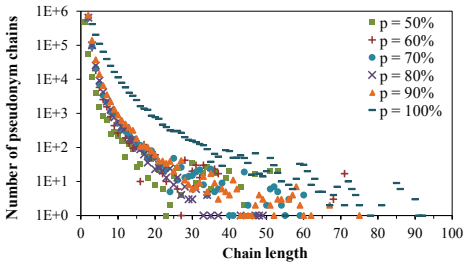


Figure 10: Number of identified pseudonym chains for a constant population of 100 clients

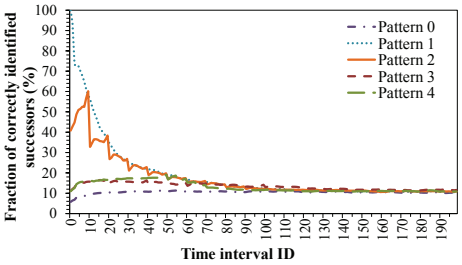


Figure 11: Fraction of correctly identified successors over time for different activity patterns and a variable population of up to 100 clients

the cloaked reputation for future use.

Compared to the transient scheme addressed in Section 6.1, we additionally consider a variable population of clients, i.e., clients become active after the beginning of the simulation. We selected four different activity patterns presented in Table 3 and use a constant population of 100 active clients as baseline for our analysis (*Pattern 0*). Each activity pattern is defined by (a) the number of clients active at the simulation start, (b) start of the first period, (c) duration of the period between newly activated clients, and (d) number of activated clients after each period. For comparison purposes, we assume that the total number of clients is the same for all scenarios and that they become active latest at the 100th time interval. We further assume that all clients

Table 3: Selected activity patterns

	Initial number of clients	Period start (Time interval ID)	Period duration	Number of new clients
Pattern 0	100	-	-	-
Pattern 1	1	0	1	1
Pattern 2	10	0	10	10
Pattern 3	50	0	20	10
Pattern 4	50	50	5	10

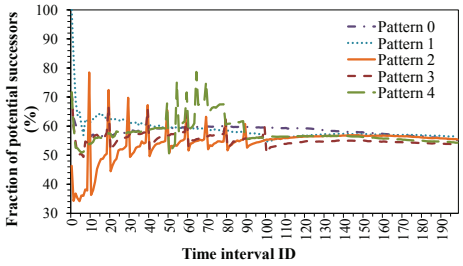


Figure 12: Fraction of potential successors over time for different activity patterns and a variable population of up to 100 clients

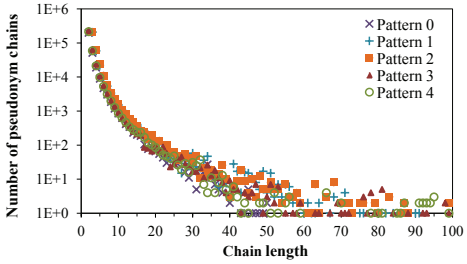
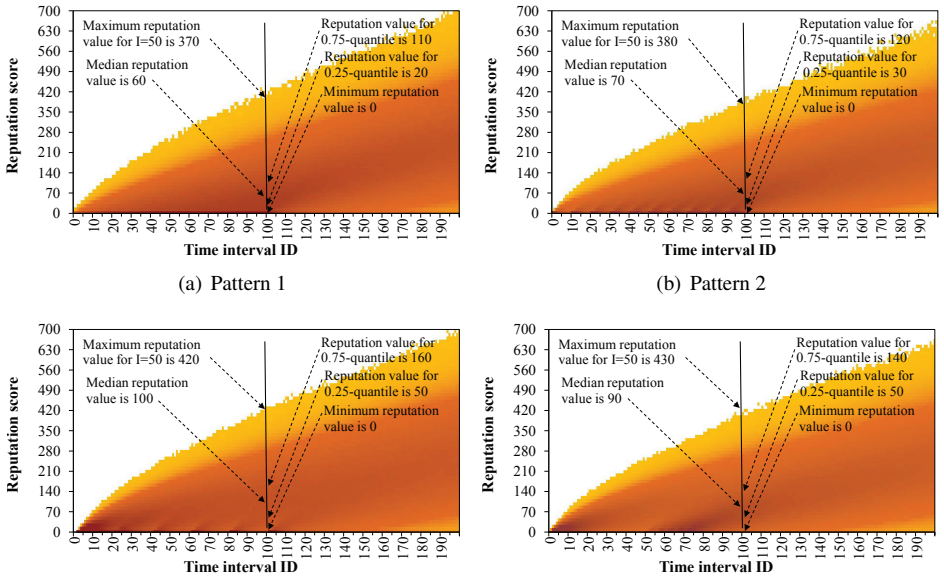


Figure 13: Number of identified pseudonym chains for different activity patterns and a variable population of up to 100 clients

transfer their reputation scores with a probability p equal to 80%. This implies that the results for Pattern 0 are the same as those previously discussed for a constant population of clients and p=80%. In order to observe the effects of the latest activated clients on the degree of anonymity protection, we run each simulation for 200 time intervals.

Fig. 11 illustrates the impact of the introduction of new clients on the fraction of correctly identified successors for each activity pattern. It shows that introducing clients according to Patterns 3 and 4 increases the number of correctly identified successors of up to 7% (t=20) and 8% (t=50) compared to a constant population of clients (i.e., Pattern 0), respectively. In comparison, Patterns 1 and 2 lead to the correct identification of a higher number of successors. This difference is due to the number of introduced clients. Since new clients present similar reputation scores, the larger the group of new clients, the higher the number of potential successors for these clients and hence, the better the reciprocal protection between clients. Moreover, the difference between Pattern 2 (i.e., 10 clients introduced every 10 time intervals) and 3 (i.e., 20 clients introduced every 10 time intervals) shows that the introduction of 10 more clients at each period significantly improves anonymity protection, since a lower number of successors can be identified by potential adversaries. Note that we consider the determination of the exact value of clients to introduce in order to guarantee anonymity protection as future work. Furthermore, Pattern 1 reaches 100% of correctly identified successors in the first time interval because there is only one client introduced at this time. Subsequently, the difficulty to correctly identify successors increases at each introduction of a new client. After the last introduction of new clients, the values of Patterns 1 to 4 progressively stabilize around the value of Pattern 0.

These results are mirrored in Fig. 12. In particular, it shows for Patterns 3 and 4 the augmentation of the number of potential successors caused by each introduction of clients, which variations are not apparent in Fig. 11. Note that Pattern 4 shows a similar behavior to Pattern 0 until the 50th time interval as the population of clients remains constant. However, the de-



(a) Pattern 1 (b) Pattern 2

(c) Pattern 3 (d) Pattern 4

Figure 14: Distribution of reputation score amongst clients over time for different activity patterns and a variable population of up to 100 clients

gree of anonymity protection of the clients in Pattern 4 is lower than those in Pattern 0 during this time interval, since the population in Pattern 4 is half the population of Pattern 0. For all patterns, the variations in number of potential successors caused by new clients are temporary, as new clients progressively gain reputation scores and become indistinguishable from clients active for a longer period. This is illustrated in Fig. 14, where the number of occurrences of low reputation scores temporarily augments at each introduction of new clients and later stabilizes. Since new clients have low reputation scores, they become potential successors of already active clients. While the weight associated to the links between new and already active clients is low at the beginning, it increases as the new clients gain reputation as shown in Fig. 14. The length of identified pseudonym chains increases with the introduction of new clients as illustrated in Fig. 13. However, no difference can be clearly identified between Patterns 1 to 4 and hence, no concrete conclusion can be drawn on the effect of the introduction patterns on the chain lengths. In summary, the results show that the degree of anonymity protection for a variable population of clients is lower than for a constant population. However, the anonymity protection increases with the number of simultaneously introduced clients, as they protect each other.

7. Empirical Evaluation of Overheads

We implemented a proof-of-concept of IncogniSense to demonstrate the feasibility of our approach for transient reputation scores. In particular, we quantified the overhead in terms of energy consumption for the clients. The overhead should be maintained as low as possible so as to not drain the battery of the mobile phones. In our implementation, the client program was developed on Android Nexus S phones, while the application server and RPM were implemented

Table 4: Measured overhead per *RT* and key generation for the clients

	RT	Key pair
Average battery lifetime (hour)	5:09	5:18
Average number of executions	5037500	16860
Average execution time (ms)	4	1129

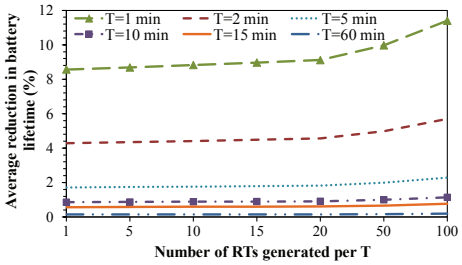


Figure 15: Estimated daily cost in battery lifetime

as two Apache Tomcat servlets. The activities of the clients are managed by a background thread. The blind RSA signatures are based on 1024-bit private/public key pairs. The pseudonyms and *RT*s are stored in a SQLite database on the clients, the access to which is strictly restricted to our application. The RPM maintains lists of the generated pseudonyms and utilized *RT*s in MySQL databases. It also controls the common time intervals and runs a synchronization function that determines the time drift between the server and the phones. The clients, the RPM, and the application server communicate via Wi-Fi and the communications are secured using HTTPS.

We conducted a first experiment to measure the impact of the generation of pseudonyms, *RT*s, and keys on the clients' battery lifetime. We configured a benchmarking client to repeatedly execute the aforementioned sequence of operations until the exhaustion of the battery. We disabled all other programs and repeated each experiment 20 times on different clients. The results presented in Table 4 indicate that the overhead to generate keys for the new pseudonyms is significantly higher than the overhead to create *RT*s. Note that the overhead to create a pseudonym is equal to the overheads caused by both *RT* and key pair generation. IncogniSense saves 90% of energy while reducing the probability of reputation corruption from 0.1 to 0 as compared to [14] assuming $n = 10$. In a second experiment, we configured a client to generate pseudonyms and *RT*s with a period T. We examined the impact of both the duration of T and the number of generated *RT*s on the battery usage. Note that the impact of the cloaking schemes on the battery usage is negligible compared to the cryptographic operations required to generate *RT*s and pseudonyms. Fig. 15 shows the overall reduction in battery lifetime per day imputed to our approach. By selecting T greater than 5 minutes, the daily cost in battery lifetime remains below 2.3%, rendering our approach feasible for resource-constrained mobile phones.

8. Conclusions

We have proposed an anonymity-preserving reputation framework called IncogniSense, which is agnostic to both the applications and the applied reputation algorithm. Our system utilizes periodic pseudonyms which are generated using blind signature and relies on a secure reputation

transfer mechanism between these pseudonyms. We have introduced the concept of reputation cloaking to prevent an adversary from de-anonymizing the users by linking the reputation scores associated with their contributions. As cloaking has an inherent trade-off between anonymity protection and loss in reputation, we have explored the design space and extensively examined the performances of several different schemes. Based on this analysis, we have provided guidelines for the choice of the appropriate cloaking scheme in accordance with the application requirements. We have demonstrated the resilience of our system against typical threats and a prototype implementation confirms the feasibility of our solution.

Acknowledgments

Our thanks go to the anonymous reviewers for their valuable suggestions. This work was partially supported by CASED (`www.cased.de`), EIT ICT Labs (`www.ictlabs.eu`), EC SPRIDE (`www.ec-spride.de`), and the Swedish National Graduate School in Computer Science (CUGS).

Appendix A. Blind Signatures, RSA Signatures and Blind RSA Signatures

A cryptographic system that can be used to create blind signatures was first proposed by Chaum [6]. The underlying idea is to use a public-key cryptographic system with signers and providers. It works as follows. The signer has a secret signing function s' and its public inverse s, such that $s(s'(m)) = m$ for a message m. The provider has two secret cryptographic functions c and c', where c' is the inverse of c. Hence, the provider can send a ciphertext $c(m)$ to the signer, which returns $s'(c(m))$ to the provider. The provider can then obtain the message $s'(m)$ signed by the supplier using the function c', such that $c'(s'(c(m))) = s'(m)$. Anyone can then verify the signature on message m by checking that $s(s'(m)) = m$. Hence, the supplier signed the message m without knowing m, i.e. a blind signature.

The RSA [21] signature scheme can be used for implementing a blind signature scheme [22]. We first briefly recall how the RSA signature scheme works. In RSA, the provider signs its own messages, i.e., the provider and the signer are the same entity. A signature s of message m is $s = m^e \bmod n$, where (e, n) is the *signature key*, $n = p \cdot q$ and p and q are two arbitrarily chosen large prime numbers. The parameter e, $(e < n)$, is a relative prime to the totient of n, $\phi(n) = (p-1)(q-1)$. For checking the correctness of an RSA signature, the verifier needs a *verification key* (d, n), where d is the inverse of $e \bmod \phi(n)$, i.e., $e \cdot d \bmod \phi(n) \equiv 1$. The verification of signature is done by checking if s^d is equivalent to $m \bmod n$.

In a blind RSA signature scheme, the provider and the signer are different entities. The signer is a third party that does not know the content of the message to be signed. The blind RSA signature scheme incorporates a blinding factor r to the RSA signature scheme and works as follows. The provider of the message m, which is to be blind signed, selects a random blinding factor $r \bmod n$ and generates a blinded message $m' = m \cdot r^e \bmod n$. The blinding factor r is a secret only known to the provider and (e, n) is publicly known. The blinded message m' is sent to the signer, who returns the signature $s' = (m')^d \bmod n$ to the provider. The signature s' on message m' is obtained using the signature key (d, n), which is a secret only known to the signer. The provider can then calculate the signature $s = m^d \bmod n$ on message m as

$$s' = (m')^d \bmod n \;=\; (m \cdot r^e)^d \bmod n \;=\; m^d \cdot r^{e \cdot d} \bmod n \;=\; m^d \cdot r \bmod n \qquad \text{(A.1)}$$

The provider then divides s' by r to obtain the signature s, i.e., $s = s'/r$. Anyone can verify the signature s on message m by checking if s^e is equivalent to $m \bmod n$.

Appendix B. Generation of Pseudonyms using RSA Signatures

We assume that each client has a permanent identifier ID, a private key (d_{client}, n_{client}), a public key (e_{client}, n_{client}), and it is registered with the RPM. The blind RSA signature scheme, which is presented in Appendix A, is the blind signature scheme used in the rest of this appendix. For each T, the RPM generates a new private/public pair of *signature keys* common to all clients: $(d_{signature}, n_{signature})$ and $(e_{signature}, n_{signature})$. The client first generates a private/public key pair (d_P, n_P) and (e_P, n_P) for its new pseudonym. The client uses n_P as its new pseudonym referred to as P and generates the corresponding signature s_P as follows. The client first prepares the message m_P using n_P, the RPM's public *signature key* for the time period T, $(e_{signature}, n_{signature})$, and the blinding factor r modulo n_P, as follows:

$$m_P = n_P \cdot r^{e_{signature}} \; mod \; n_{signature} \tag{B.1}$$

The client creates a signature s_{m_P} using a function of the concatenation f of the triplet (m_P, ID, T) signed with its permanent private key to guarantee the authenticity of m_P:

$$s_{m_P} = f(m_P \parallel ID \parallel T)^{d_{client}} \; mod \; n_{client} \tag{B.2}$$

The client transmits m_P, s_{m_P}, its ID, and the time interval of validity T for P to the RPM for blind signature. The RPM verifies the authenticity of m_P and that the client has no existing pseudonym for this time interval. After verification, the RPM generates the blind signature s_{RPM} signing m_P:

$$s_{RPM} = {m_P}^{d_{signature}} \; mod \; n_{signature} \tag{B.3}$$

The client finally generates the pseudonym's signature s_P from the blind signature s_{RPM} that achieves the generation of P, which becomes $P_{current}$:

$$s_P = s_{RPM} \cdot r^{-1} \; mod \; n_{signature} \tag{B.4}$$

Appendix C. Generation of Reputation Tokens using RSA Signatures

We also assume that the RPM generates a set of *transfer keys* in the bootstrapping phase. Each *transfer key* pair is associated to a reputation value and determines the reputation associated to a given *RT*. For each *RT*, the client selects a random bit string ID_{RT} as identifier and prepares the message m_{RT} for blind signature using the public *transfer key* corresponding to the *RT*'s value:

$$m_{RT} = ID_{RT} \cdot r^{e_{transfer}} \; mod \; n_{transfer} \tag{C.1}$$

The client signs the message m_{RT} with the signature $s_{m_{RT}}$ using the private key $(d_{P_{current}}, n_{P_{current}})$ associated to $P_{current}$. The real identity of the client is hence not revealed while transferring reputation from one pseudonym to the next.

$$s_{m_{RT}} = f(m_{RT} \parallel P_{current} \parallel R_{score})^{d_{P_{current}}} \; mod \; n_{P_{current}} \tag{C.2}$$

The client transmits m_{RT}, $s_{m_{RT}}$, $P_{current}$, and R_{score} to the RPM for blind signature. The RPM verifies that m_{RT} is used for the first time and the balance of the reputation account of $P_{current}$ before decrementing it by R_{score}. After verification, the RPM blindly signs m_{RT} with the corresponding private *signature key*. The client uses the blind signature to generate the final signature of the *RT*.

References

[1] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, M. Srivastava, Participatory Sensing, in: Proceedings of the 1st Workshop on World-Sensor-Web (WSW), 2006, pp. 1–5.

[2] R. Rana, C. Chou, S. S. Kanhere, N. Bulusu, W. Hu, Ear-Phone: An End-to-end Participatory Urban Noise Mapping System, in: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2010, pp. 105–116.

[3] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, S. Madden, CarTel: A Distributed Mobile Sensor Computing System, in: Proceedings of the 4th ACM International Conference on Embedded Networked Sensor Systems (SenSys), 2006, pp. 125–138.

[4] K. L. Huang, S. S. Kanhere, W. Hu, Are You Contributing Trustworthy Data?: The Case for a Reputation System in Participatory Sensing, in: Proceedings of the 13th ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM), 2010, pp. 14–22.

[5] D. Christin, A. Reinhardt, S. S. Kanhere, M. Hollick, A Survey on Privacy in Mobile Participatory Sensing Applications, Journal of Systems and Software 84 (11) (2011) 1928–1946.

[6] D. Chaum, Blind Signatures for Untraceable Payments, in: Advances in Cryptology: Proceedings of Crypto 82, 1983, pp. 199–203.

[7] D. Christin, C. Roßkopf, M. Hollick, L. Martucci, S. Kanhere, IncogniSense: An Anonymity-preserving Reputation Framework for Participatory Sensing Applications, in: Proceedings of the 10th IEEE International Conference on Pervasive Computing and Communications (PerCom), 2012, pp. 135–143.

[8] I. Krontiris, N. Maisonneuve, Participatory Sensing: The Tension Between Social Translucence and Privacy, in: Trustworthy Internet, 2011, pp. 159–170.

[9] M. Kinateder, S. Pearson, A Privacy-Enhanced Peer-to-Peer Reputation System, in: E-Commerce and Web Technologies, Vol. 2738, 2003, pp. 206–215.

[10] E. Androulaki, S. G. Choi, S. M. Bellovin, T. Malkin, Reputation Systems for Anonymous Networks, in: Privacy Enhancing Technologies, 2008, pp. 202–218.

[11] L. A. Martucci, S. Ries, M. Mühlhäuser, Sybil-Free Pseudonyms, Privacy and Trust: Identity Management in the Internet of Services, Journal of Information Processing 19 (1) (2011) 1–15.

[12] C. Andersson, M. Kohlweiss, L. Martucci, A. Panchenko, A Self-certified and Sybil-Free Framework for Secure Digital Identity Domain Buildup, in: J. Onieva, D. Sauveron, S. Chaumette, D. Gollmann, K. Markantonakis (Eds.), Information Security Theory and Practices. Smart Devices, Convergence and Next Generation Networks, Vol. 5019 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2008, pp. 64–77.

[13] Z. Zhang, J. Liu, Y. Kadobayashi, STARS: A Simple and Efficient Scheme for Providing Transparent Traceability and Anonymity to Reputation Systems, in: Proceedings of the International Workshop on Autonomous and Spontaneous Security (SETOP), 2010, pp. 170–187.

[14] H. Miranda, L. Rodrigues, A Framework to Provide Anonymity in Reputation Systems, in: Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services (MobiQuitous), 2007, pp. 1–4.

[15] Y. Wei, Y. He, A Pseudonym Changing-based Anonymity Protocol for P2P Reputation Systems, in: Proceedings of the 1st International Workshop on Education Technology and Computer Science (ETCS), 2009, pp. 975 –980.

[16] L. Sweeney, K-anonymity: A Model for Protecting Privacy, International Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems 10 (5) (2002) 557–570.

[17] D. Dolev, A. C. Yao, On the Security of Public Key Protocols, IEEE Transactions on Information Theory 29 (2) (1983) 198–208.

[18] J. R. Douceur, The Sybil Attack, in: Revised Papers from the First International Workshop on Peer-to-Peer Systems, 2002, pp. 251–260.

[19] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, N. Triandopoulos, AnonySense: A System for Anonymous Opportunistic Sensing, Journal of Pervasive and Mobile Computing 7 (1) (2010) 16–30.

[20] T. Okamoto, An Efficient Divisible Electronic Cash Scheme, in: D. Coppersmith (Ed.), CRYPTO, Vol. 963 of Lecture Notes in Computer Science, Springer, 1995, pp. 438–451.

[21] R. L. Rivest, A. Shamir, L. Adleman, A Method for Obtaining Digital Signatures and Public-key Cryptosystems, Commununications of the ACM 21 (2) (1978) 120–126.

[22] R. Gennaro, Digital Cash, in: S. Goldwasser, M. Bellare (Eds.), Lecture Notes on Cryptography, 2008, Ch. 12.5, pp. 233–237.