

# Re-identifying Anonymous Nodes

Stefan Schlott, Frank Kargl, and Michael Weber

University of Ulm, Dept. of Media Informatics  
{stefan.schlott, frank.kargl, michael.weber}@uni-ulm.de

**Abstract.** In mobile scenarios, privacy is an aspect of growing importance. In order to avoid the creation of movement profiles, participating nodes change their identifying properties on a regular basis in order to hide their identities and stay anonymous. The drawback of this action is that nodes which previously had a connection have no means to recognise this fact. A complete re-authentication would be necessary – if possible at all.

This paper discusses this new problem and proposes two possible solutions for re-identification of anonymous nodes, one based on symmetric encryption and one based on secure hashes.

## 1 Introduction

Information on a device's context – especially its location – offers a huge variety of interesting new applications. As a drawback, the same information in the wrong hands poses a threat to the users' privacy: It enables a malicious observer to track a user's movements; more detailed context information allows an attacker to infer additional personal data like personal habits.

As a consequence, privacy aspects have been addressed in several publications; naturally, location privacy has been the main issue. In order to defeat an attacker trying to track someone, most of these proposals suggest anonymisation of electronic device identity, i.e. make a device indistinguishable from others. A drawback of this measure is that the devices lose (probably very interesting) context information: The devices can no longer distinguish between nodes in their neighbourhood that they have met before and foreign ones. Our goal is to mitigate this disadvantage in order to allow scenarios analogous to the one described in the next section.

### 1.1 In the Real World...

In an environment without electronic surveillance, everybody is able to act with a reasonable amount of privacy. Imagine yourself walking through a crowded mall. You can stop at shops, browse through the goods for sale, and buy whatever you like. If you pay in cash, you will leave no trace of your transaction, and if the mall is visited by enough people, the shop owners (as well as the other customers in the hallways) won't remember you.

On the other hand, when you walk through the hallway, you might suddenly recognise a known face - just like "isn't that the guy I met on the party last

week?"). To everyone else, he is just another visitor of the mall; but you can single him out of the mass.

A digital representation of scenarios like the one described above can prove very useful in ubiquitous computing scenarios (e.g. allowing your colleague's PDA to re-identify your PDA for data exchange). But due to the possibility of automatic mass-surveillance, care has to be taken not to violate the individuals' privacy.

## 1.2 ...and Its Electronic Companion

The users' privacy is an important aspect in almost all distributed systems. In a ubiquitous scenario, privacy becomes even more important: As the miniature computers blend with everyday items, the users are no longer aware of potential threats to their privacy [1]. Some of these items will be carried around all day.

This rises the question of location privacy: How can users interact with (probably location-based) services while denying them the chance to create movement profiles? A first step toward the circumvention of movement profiles is to constantly change one's identifying attributes [2, 3].

In some cases however, this behaviour bears several disadvantages: Connecting to a known node requires knowledge of its current pseudonym. Solutions which have been discussed include a trusted relay [4], similar to the home location register of mobile IP [5].

There may be several reasons why other solutions would be preferable. First, in scenarios without any infrastructure or reliable Internet connection, this approach is unusable for obvious reasons. Further, it requires a service provider which the user is obliged to trust. Depending on the scenario, a local solution (i.e. without the involvement of a third party) may be more efficient. Finally, a protocol which does not rely on a third party may be employed in a wide variety of different contexts, e.g. in anonymous peer-to-peer overlay networks.

## 2 Scenario

In our scenario, we allow for nodes with different processing capabilities. Some nodes may be battery-powered and be equipped with very little CPU power. A very popular example are the MICA Motes [6], which are powered by 4 – 8 MHz 8-bit microcontrollers.

Regarding the limited capacity of memory and processing power, the use of public key cryptography becomes next to impractical: Even in an optimised implementation [7], a single RSA decryption operation takes up to 22 seconds. Therefore, costly algorithms like RSA should be substituted by less demanding public key schemes (like elliptic curves) and should be used sparsely.

### 2.1 Prerequisites and Goals

The goal is to design a protocol which represents a digital equivalent of the scenario depicted in the motivation section. The protocol should enable nodes

to re-identify each other when meeting again. We assume that they met before and had the chance to communicate over a secure channel. Note that this does not imply that the nodes needed to authenticate each other: There is no need to exchange names or other identifiers during this first step. It is possible to re-identify another subject and only know that it is the same entity that you met at an earlier time.

As mentioned above, privacy is of great concern. Thus the protocol has to cope with the fact that an entity has no clue who its counterpart is (and vice versa). No part of the protocol must reveal any clue on the identity of the nodes. In case of a successful negotiation, both nodes should re-identify each other; otherwise, neither should gain information on the other node.

## 2.2 Anonymity

The definition of the term "anonymity" tends to differ depending on the literature. Since the year 2000, Pfitzmann et. al. are trying to fix a precise definition [8]. In their ongoing work [9], they define anonymity as "the state of being not identifiable within a set of subjects (the anonymity set)". This is sometimes referred as  $k$ -anonymity (with  $k$  being the size of the set of subjects).

We assume that nodes change their identifying attributes on a regular basis, either after a given time or after a completed communication with another node [10]. In the following, we call the union of these identifying attributes the node's pseudonym.

The pseudonyms chosen by the nodes are supposed to be random. This results in unlinkable IDs, i.e. a pseudonym gives an observer no clue on the future pseudonym.

## 2.3 The Attacker

In the following considerations, we assume an attacker according to the classic Dolev-Yao model [11]. This adversary has full control over the network, i.e. he can generate, modify, or delete messages. Side channels (like timing, etc.) are more difficult to deal with. Side channel attacks are a threat with a wide variety of possibilities, making a formal evaluation impossible. Some possible side channels will be discussed informally later in the paper.

## 3 Related Work

The fundamental privacy problem from a non-technological point of view is discussed in [1]. In [8], the authors define a terminology for privacy-related facts. Several works on security in ubiquitous computing also point out specific privacy issues [12, 13].

Classic identification schemes cope with the problem of certifying the identity of a communication partner; neither the observed communication nor the data obtained when being part of the protocol may be usable for impersonation [14, p. 283].

Well-known algorithms like Schnorr's scheme [15] fulfil these requirements. Since anonymity is a new requirement, these protocols do not regard this fact,

requiring the participants to disclose their identity (which later will be verified) in the course of the protocol.

In the context of location-based services, privacy issues are being discussed. Most approaches rely on some kind of mix net; the concept of mix nets was first introduced by David Chaum in [16]. The Mist routing infrastructure [4] employs such a network in order to allow untraceable use of location-based services. In [17], Kölsch et. al. suggest a single trusted intermediary for anonymising requests to services. To further increase the amount of privacy, several authors (e.g. [18]) suggest controlling the granularity and precision of the data given to a service. In the context described in the introduction, these approaches suffer from several drawbacks: First, they require a given infrastructure, which provides anonymity. Second, the resolution of the pseudonyms of nodes in the neighbourhood could be integrated in these designs, but this would require a single trusted entity knowing all node positions.

The usage of zero-knowledge protocols [19] is an obvious thought. However, most zero-knowledge protocols are interactive protocols using several rounds; many involve complex computing operations making them an inappropriate means on resource-limited devices. In [20], Goldreich notes the existence of non-interactive zero-knowledge protocols. They too are rather demanding for devices with limited computing resources. Further, these protocols also assume that both parties have a common understanding of precisely what secret should be verified. In case of a re-encounter of two anonymous nodes, neither of these two nodes knows the identity of the other node and hence the nodes have no idea what secret the other node wants to prove.

Balfanz et. al. presented in [21] a protocol addressing the problem of authentication while not revealing any information to observers or unknown communication partners. The scenario described here was the effort to prove one's group membership by performing a "secret handshake". Only members of a group should be able to perform the appropriate authentication steps; non-members and observers should gain no information on the outcome of the protocol. For our scenario, this protocol has several drawbacks: It requires a trusted entity issuing certificates, and it employs large number operations. Both makes it inappropriate for many ubiquitous scenarios. Finally, our scenario requires no group authentication, but merely a 1:1 authentication.

Abadi discusses a private authentication protocol for two participants in [22]. According to the publication, the protocol does not require the participants to have met before. Both protocol variants introduced in the paper employ public key cryptography, making its frequent use on small devices very costly. Further, in both protocols, the first message is encrypted with the public key of the communication counterpart. This poses two unsolved problems: First, the identity of the communication partner is unknown, which means that either the identity has to be exchanged in advance, or all known keys have to be tested. Second, in contrast to the initial claim, the public keys of the communication partners must be known (at least in the first step).

The emerging RFID technology has several properties which are very similar to mobile and ubiquitous computing, e.g. portability, wireless communication, limited resources, and invisibility for most users. As industry deploys more and more RFID tags in end user products, the question for privacy has risen, too [23]. In consequence, researchers addressed that problem. Molnar et. al. propose in [24] a means for RFID chips to retain their anonymity unless they are contacted by an authenticated RFID reader. The protocol deals with the special circumstances of RFID technology, namely very limited memory and processing power. However, the protocol assumes that the RFID reader(s) reveal their identities.

In contrast to that, the two protocols that we present in the next sections protect the privacy of both communication partners.

## 4 Identification Using Symmetric Ciphers

In the following sections, these symbols are used:

- $E_k(v)$  denotes a symmetric encryption with key  $k$  of the value  $v$ .
- $H(v)$  is the cryptographic hash of the value  $v$ .
- $n$  represents a nonce. These values should be used only once. Additionally, we require values of  $n$  to be randomly chosen.
- $T$  is a token – an initially randomly chosen, fixed value which serves as unique identifier.
- *decoy* stands for random data.
- $A$  is the node initiating the protocol
- $B$  denotes the second communication partner. The identity of this node is initially unknown to  $A$  (and vice versa).
- The table storing old session data has  $s$  entries.

We specify neither specific algorithms nor bit lengths. These should be chosen with reasonable sizes but with regard to hardware limitations. The size of the values should be chosen in a way that brute-force enumeration is not possible within an acceptable time span.

### 4.1 Naive Approach

The naive approach shown below is a description of the basic approach without any optimisations. It should serve to describe the rough idea and to point out the problems.

When two nodes meet for the first time, they exchange a session key  $k_{ab}$  and an identifier for this session,  $T_{ab}$ , over the secure channel.

The re-identification protocol is initiated by  $A$ . Since  $A$  does not know who  $B$  is,  $A$  simply tries all keys stored in its session storage:

$$A \longrightarrow B : \forall b : E_{k_{ab}}(T_{ab}, n_1) \quad (1.1)$$

For each message,  $B$  tries to decrypt the payload with all stored session keys  $k_{ba}$ .  $B$  can verify the successful decryption by comparing the decrypted value with the stored  $T_{ba}$ . If  $T_{ba}$  matches,  $B$  replies with message 1.2:

$$B \longrightarrow A : E_{k_{ab}}(n_1, n_2) \quad (1.2)$$

Otherwise,  $B$  sends the decoy message 1.3 with identical length:

$$B \longrightarrow A : \text{decoy} \quad (1.3)$$

$A$  in turn can check for a successful identification by decrypting the data received from  $B$ . If  $A$  yields the nonce  $n_1$  (which was never transmitted in plain and thus is known to nobody except the correct recipient), the re-identification is successful.  $A$  and  $B$  may then continue to communicate encrypted using the session key  $k_{ab}$ . The first message sent by  $A$  should contain  $n_2$  in order to guarantee freshness of the protocol run.

If every participant stores  $s$  entries in his session list, the complexity may be estimated as followed:  $O(s)$  messages are sent,  $A$  has to perform  $O(s)$ ,  $B$   $O(s^2)$  symmetric crypto operations.

The protocol allows an anonymous re-identification of nodes which already exchanged their credentials. The names  $A$  and  $B$  may sound a bit misleading, since they imply a known, fixed identity; in common protocol descriptions, this is (hopefully) true – the communication partners know whom they are talking to (or, at least, think that they know; there still might be a man in the middle). In our case,  $A$  and  $B$  represent just some node. Since in the beginning, both  $A$  and  $B$  are anonymous, neither knows who he is talking to.

As a natural consequence, when using a broadcast medium for message transmission, there might be a node  $C$  within range, which also exchanged a session key with  $A$  earlier.  $C$  will be able to identify  $A$ , too. This is analogous to the scenario of the introduction: All passers-by in the mall can see you, and if someone knows you, he will recognise you.

If this side effect is unwanted,  $A$  and  $B$  have to make sure that no one else is listening - either by using a direct link for transmission, or by establishing an encrypted communication. The setup of such an encrypted channel is out of the focus of this paper, see [25, 26, 27] for possible approaches.

## 4.2 Enhancing Performance with Indexes

Although this simple approach fulfils the goal of anonymous re-identification, it causes high traffic and high processing load, especially for  $B$ . This section shows a variation of the protocol that enhances the performance at the cost of a slightly increased chance that an adversary can successfully identify the communication partners.

In order to reduce the amount of decryption operations for  $B$ ,  $A$  and  $B$  store an additional index identifier  $i_{ab}$  during their first meeting.  $i$  is chosen randomly. The purpose of the index is to narrow down the amount of possible session keys. This index should not be unique, since this would provide a simple way for an

attacker to identify a node. The domain of  $i$  should be small enough to have enough collisions to maintain privacy. In many applications, a small number of bits will be sufficient; for example, a node with 1000 stored session keys can reduce the number of possible keys to an average of approximately 60 keys with a four bit index.

A sends packets as depicted in message 2.1:

$$A \longrightarrow B : \forall b : i_{ab}, E_{k_{ab}}(T_{ab}, n_1) \quad (2.1)$$

Similar to the naive approach,  $B$  tries to decrypt the received packets – but only with keys which have a matching  $i_{ab}$ . The theoretical complexity remains the same; in practise, with proper values for  $s$  and the domain of  $i$ , the amount of computations can be reduced to a certain degree.

The rest of the protocol is identical to the naive approach: On success,  $B$  replies with message 2.2, otherwise with a decoy message of identical length.

$$B \longrightarrow A : E_{k_{ab}}(n_1, newi_{ab}) \quad (2.2)$$

In message 2.2,  $B$  sends a new (random)  $i_{ab}$  to  $A$ .  $A$  replaces the old value with the new one, which will be used during the next authentication.

### 4.3 Problems with the Index

The introduction of the index is a trade-off between performance and privacy preservation: Even though the index is changed during every successful handshake, most index entries remain unchanged. That means that the sequence of indexes forms a characteristic string, which may be used by an adversary for identification. For the sake of simplicity, we assume in this section that all nodes store the same number of sessions.

Two unsuccessful protocol runs would result in identical index sequences. The degree of anonymity can be amounted to the probability that two different nodes bear the same characteristic string. In this case, the probability is  $1 : 2^{s \cdot l}$ , with  $s$  being the number of indexes and  $l$  the length of the index (in bits). Obviously, this does not provide adequate privacy.

Randomly changing the sequence of keys during each run of the protocol reduces the information which can be gathered by an observer. A drawback of this procedure is the requirement of more randomness, which may be difficult to gain.

The information exploitable by an adversary is the frequency of the index values. Assuming that the index values are equally distributed, the probability of two nodes having the same frequency of index values<sup>1</sup> is  $1 : \binom{2^l + s - 1}{s}$ , which equals  $1 : \frac{(2^l + s - 1)!}{s! \cdot (2^l - 1)!}$ . This results in an adequate amount of privacy only for small numbers of  $s$ .

---

<sup>1</sup> Combination with repetition (order does not matter, objects can be chosen more than once), under the assumption that the hints are equally distributed.

Alternatively,  $A$  could suggest a new index to  $B$  (instead of receiving a newly chosen index generated by  $B$  in message 2.2). This enables  $A$  to pick the new index value in a way that balances the differences in the occurrences of the index values in  $A$ 's session list. The key sequences sent by  $A$  should be sorted by the index values. That results in very homogeneous (and indistinguishable) sequences.

The only information left for the adversary is the difference from the average count ("one less than the average" or "average"). This results in a probability of  $1 : 2^l$ , which can be considered an adequate result.

## 5 Identification Using Hashes

A further reduction of messages and computation steps can be achieved by the following protocol while retaining all the positive properties of the naive approach. Now all nodes possess an identity token  $T$ . On their first meeting,  $A$  and  $B$  exchange their identity tokens  $T_a$  and  $T_b$  together with a session key  $k_{ab}$ .

$A$  initiates the protocol by sending a nonce in message 3.1:

$$A \longrightarrow B : n_1 \tag{3.1}$$

$B$  calculates the hash of the nonce and its identity token  $T$ . An additional nonce  $n_2$  is inserted to defeat identification via message replay.

$$B \longrightarrow A : n_2, H(T_b, n_1, n_2) \tag{3.2}$$

After receiving message 3.2,  $A$  calculates the hash of  $n_1, n_2$  and all stored  $T_b$ ; if the result matches the data received,  $A$  has successfully identified its communication partner. To allow  $B$  to identify  $A$ ,  $A$  sends the respective message 3.3.

$A$  includes an additional challenge in form of an encrypted nonce. This is necessary because all associated nodes know the identities  $T$  of other nodes. This challenge proves that neither party is lying about the identity  $T$  employed during the protocol, because session keys are only used pairwise.

$$A \longrightarrow B : H(T_a, n_1, n_2), E_{k_{ab}}(n_3) \tag{3.3}$$

$B$  can now determine the identity of  $A$ , and proves the possession of the correct session key by replying with message 3.4:

$$B \longrightarrow A : E_{k_{ab}}(n_3 + 1, n_4) \tag{3.4}$$

Message 3.5 concludes the handshake with  $A$ 's session key possession proof.

$$A \longrightarrow B : E_{k_{ab}}(n_4 + 1) \tag{3.5}$$

This protocol reduces the complexity to a fixed number of messages and  $O(s)$  cryptographic operations on both sides.

## 6 Verification

In this section we will prove the correctness of our two approaches, namely the naive and the hash-based alternative. We assume a Dolev-Yao style attacker that can eavesdrop and even modify all packets.

### 6.1 Assumptions

Let  $A$  be the originator of the communication,  $B$  the recipient. From a previous session,  $A$  and  $B$  share a common cryptographic secret  $k_{ab} = k_{ba}$  of reasonable length that is unknown to any other party. They have also exchanged a pair-wise token  $T_{ab} = T_{ba}$  for the symmetric cipher case, and per-node tokens  $T_a$  and  $T_b$  for the identification with hashes.  $E_k$  is a symmetric cipher that is assumed to be secure, i.e. it cannot be broken faster than by brute-forcing the key space and produces output that cannot be distinguished from random noise.  $H$  is a cryptographically perfect hash-function, which means it generates uniformly and completely random output, and is strong collision resistant. Finally, we assume that  $A$  and  $B$  are able to generate true random numbers for generating nonces  $n_i$ .

### 6.2 Identification Using Symmetric Ciphers

In the naive case,  $A$  knows  $n$  keys  $k_{ax_i} \forall i = 1 \dots n$  and has no knowledge about the identity of its communication partner  $B$ .  $\forall i = 1 \dots n$ ,  $A$  sends a message  $E_{k_{ax_i}}(T_{ab}, n_1)$  resulting in a total of  $n$  messages (message 1.1). As assumed above, any attacker  $E$  eavesdropping these messages and not knowing the correct key will see only random noise and cannot derive any information from the message, besides that one party with a random address sends a message to another party with a random address. The nonce  $n_1$  ensures that  $A$  will never send the same message twice, so  $E$  cannot correlate the current message to any messages sent earlier in order to gain some information.

Next,  $B$  will try to decrypt the received messages with all known keys  $k_{by_j} \forall j = 1 \dots m$ . If  $y_j \neq a$  (i.e. the selected key is not  $k_{ba}$ ), the result of this process will again be randomly distributed data, so  $B$  learns nothing about  $A$ 's relationship to other nodes. Only if  $y_j = a$ ,  $B$  will recognise the common token  $T_{ab}$  and learn the nonce  $n_1$ .

For each message received,  $B$  will send back either a message containing random noise (message 1.3) or  $E_{k_{ab}}(n_1, n_2)$  (message 1.2). In both cases, the eavesdropper  $E$  will see only a random string from which he cannot derive any additional information. In order to prevent a side-channel attack,  $B$  will send all answers at the same, fixed time intervals, e.g. strictly 100 ms after receiving the incoming message. Again the nonce  $n_2$  prevents  $E$  from correlating the response message to any response sent earlier, and defeats a possible identification by replaying a captured packet 1.1.

When  $A$  receives the response  $Res_i$  from  $B$ , it will try to decrypt it with the corresponding key used in the request  $Req_i$ . If it detects  $i$  it now knows the identity of  $B$  and can reuse  $k_{ab}$  for further communication. If the keys do not

match, the result of the decryption operation will be random noise from which  $A$  gains no additional knowledge.

Finally, we will look at the effects of an active attacker  $M$ . When  $M$  inserts a request  $Req$  into the communication from  $A$  to  $B$ , eventually replacing original requests from  $A$ ,  $B$  will only decrypt random noise or the identity of  $M$ , if they share a common key  $k_{bm}$  and token  $T_{bm}$  from a previous session and if  $M$  constructs conforming requests. In the first case  $B$  will answer with random noise and  $A$  and  $B$  will not recognise each other which is an Denial of Service attack on the protocol. In the second case  $M$  will recognise the communication with  $B$  which is equivalent to  $M$  running the protocol instead of  $A$ , which is perfectly legal in our scenario. Again this leads to a Denial of Service for  $A$ , as  $A$  might not recognise  $B$ .

When  $M$  replaces response messages,  $A$  will not be able to decrypt messages containing the nonce  $n_1$  so it will not recognise  $B$ . This is again a Denial of Service attack.

If  $M$  shares both a key with  $A$  and with  $B$  it might capture all the request from  $A$  to  $B$ , replacing them with requests from  $M$  to  $B$ . It can do the same with all responses from  $B$  to  $A$ . In this case, both  $A$  and  $B$  will recognise the communication with  $M$ , so this is no Man-in-the-Middle attack, as the Man-in-the-Middle cannot impersonate  $A$  or  $B$ .

So for the naive approach the best attack possible against identification using symmetric ciphers is a Denial of Service.

### 6.3 Identification Using Hashes

When using hashes,  $A$  first sends a nonce  $n_1$  for initiating the communication (message 3.1). Therefore an eavesdropper  $E$  will learn the nonce whereas a malicious intruder  $M$  may also modify the nonce. As a result,  $A$  will refuse message 3.2 which results in a Denial of Service attack.

In the next step (message 3.2),  $B$  replies with a message containing a nonce  $n_2$  and a hash value calculated for the values  $T_b$ ,  $n_1$ , and  $n_2$ .  $A$  will now calculate hash values for all  $T_x$  that it has stored from earlier authenticated communications with hosts  $x$ . When it knows  $T_b$ , it will find a match and discovers the identity of its communication partner. Likewise, an eavesdropper  $E$  that had earlier communicated with  $B$  and knows  $T_b$ , is able to discover the identity of  $B$ . As this is equivalent to running the protocol between  $E$  and  $B$ , this is not considered an attack.

If a malicious attacker  $M$  intercepts and changes message 3.2, it can change the nonce  $n_2$  which leads to a corrupted message that will be rejected by  $A$ . This is again a Denial of Service attack. Alternatively,  $M$  can try to forge a hash value  $H(T_x, n_1, n_2)$  using any token  $T_x$  that it possesses. This is why in later steps (messages 3.3 to 3.5) a symmetric encryption using a shared secret key  $k_{ab}$  is used to verify the authenticity of the communication partner.

Message 3.3 sends another hash  $H(T_a, n_1, n_2)$  from which  $B$  can learn the identity of  $A$ . Again, an eavesdropper  $E$  that already knows  $T_a$  can learn this identity too and again this is not considered an attack, as it is equivalent to a

correct protocol run. As stated earlier, the encryption of nonces  $n_3$  and  $n_3 + 1$  (and  $n_4$  and  $n_4 + 1$  accordingly) in steps 3.3 to 3.5 using a shared secret key negotiated in earlier communication prevents an attacker  $M$  from impersonating other nodes  $X$  for which the token  $T_x$  is known.

Like in the previous section on identification using symmetric cipher,  $M$  may try to act as Man-in-the-Middle by inserting its  $T_m$  instead of  $T_a$  and  $T_b$ . It then also needs shared keys  $k_{am}$  and  $k_{mb}$ ;  $A$  and  $B$  will recognise that they are communicating with  $M$ . This is again no valid Man-in-the-Middle attack.

So the worst attack possible is again a Denial of Service.

## 7 Discussion

In practise, some optimisations can be made. To reduce the number of transmissions, the network datagrams may be filled with several guesses/replies.

Further, implementors can exploit the broadcast nature of the wireless media: Since the first step(s) of the described protocols are independent of the addressee, they can be sent via broadcast to all neighbouring nodes.

An intrinsic problem are observing nodes which have an association with the protocol initiator ( $A$  in the first,  $B$  in the second protocol), too. They will be able to reveal the initiator's identity. Due to the fact that the initiator does not know who is addressing when starting the communication, this is inevitable.

The revocation of an association to a node may be an issue, too. In case of the first protocol variant, it is sufficient to delete the respective entry from the session table. The second protocol variant allows to abort an unwanted communication only by  $A$ ; in message 3.2,  $B$  has no idea who he is talking to. The only means of revocation of  $B$  would be to change  $T_b$ , revoking all associations simultaneously. Alternatively, a node may chose to have several  $T$ 's for different roles.

The second protocol reduces the number of messages, but uses a single identifier  $T$  for each node. Depending on the attack model, this can be a problem: Malicious communication partners who exchange collected data can easily identify the communications made by the same node. In contrast, when using the first protocol, this is not possible since every node uses a different identifier in each communication.

Especially on resource-limited nodes, the session data cannot be stored indefinitely. After a given amount of memory has been filled, the node will have to purge some data. This could be done according to a timeout or a least recently used policy. Purging an association results in "forgetting" a node.

## 8 Summary and Outlook

We have introduced two protocols for re-identifying anonymous nodes, allowing all communication partners to maintain their anonymity. Both protocols do not

use public key cryptography. Due to the assumption that the nodes had prior contact over a secure channel, no third party is necessary.

Both protocol variants have their advantages and disadvantages. It depends on the scenario which variant should be used.

Future efforts will focus on the time-privacy trade-off when using indexes. Further we plan to do timing evaluations using a protocol implementation on small devices (e.g. MICAz motes).

A very interesting problem is the hinting problem: In this paper, we described a method using an index value. Using other techniques, probably from the field of steganography, could further improve efficiency of the protocol.

## References

1. Langheinrich, M.: Privacy by design - principles of privacy-aware ubiquitous systems. In Abowd, G.D., Brumitt, B., Shafer, S.A., eds.: Ubicomp. Volume 2201 of Lecture Notes in Computer Science., Springer (2001) 273–291
2. Beresford, A.R., Stajano, F.: Location privacy in pervasive computing. *IEEE Pervasive Computing* **2** (2003) 46–55
3. Görlach, A., Heinemann, A., Terpstra, W.W.: Survey on location privacy in pervasive computing. In Robinson, P., Vogt, H., Wagealla, W., eds.: Privacy, Security and Trust within the Context of Pervasive Computing. The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers (2005) 23–34
4. Al-Muhtadi, J., Campbell, R., Kapadia, A., Mickunas, M.D., Yi, S.: Routing through the mist: Privacy preserving communication in ubiquitous computing environments. In: ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02), IEEE Computer Society (2002) 74
5. Perkins, C.: IP Mobility Support for IPv4. Internet Engineering Task Force: RFC 3220 (2002)
6. Crossbow: Mica motes. <http://www.xbow.com/> (2005)
7. Gupta, V., Millard, M., Fung, S., Zhu, Y., Gura, N., Eberle, H., Shantz, S.C.: Sizzle: A standards-based end-to-end security architecture for the embedded internet (best paper). In: PerCom, IEEE Computer Society (2005) 247–256
8. Pfitzmann, A., Köhntopp, M.: Anonymity, unobservability, and pseudonymity - a proposal for terminology. In Federrath, H., ed.: Workshop on Design Issues in Anonymity and Unobservability. Volume 2009 of Lecture Notes in Computer Science., Springer (2000) 1–9
9. Pfitzmann, A., Köhntopp, M.: Anonymity, unlinkability, unobservability, pseudonymity, and identity management - a consolidated proposal for terminology. [http://dud.inf.tu-dresden.de/Anon\\_Terminology.shtml](http://dud.inf.tu-dresden.de/Anon_Terminology.shtml) (2005)
10. Schlott, S., Kargl, F., Weber, M.: Random IDs for preserving location privacy. In: SecureComm. (2005) 415 – 417
11. Dolev, D., Yao, A.C.C.: On the security of public key protocols. *IEEE Transactions on Information Theory* **29** (1983) 198–207
12. Stajano, F.: Security for ubiquitous computing. Wiley (2002)
13. Stajano, F.: Security for whom? the shifting security assumptions of pervasive computing. In Okada, M., Pierce, B.C., Scedrov, A., Tokuda, H., Yonezawa, A., eds.: Software Security – Theories and Systems, Mext-NSF-JSPS International Symposium, ISSS 2002, Tokyo, Japan, November 8-10, 2002, Revised Papers. Volume 2609 of Lecture Notes in Computer Science., Springer (2003) 16–27

14. Stinson, D.R.: *Cryptography: Theory and Practice*. CRC Press, Inc., Boca Raton, FL, USA (1995)
15. Schnorr, C.P.: Efficient signature generation by smart cards. *J. Cryptology* **4** (1991) 161–174
16. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. **24** (1981) 84–88
17. Kölsch, T., Fritsch, L., Kohlweiss, M., Kesdogan, D.: Privacy for profitable location based services. [28] 164–178
18. Wishart, R., Henriksen, K., Indulska, J.: Context obfuscation for privacy via ontological descriptions. In Strang, T., Linnhoff-Popien, C., eds.: *LoCA*. Volume 3479 of *Lecture Notes in Computer Science.*, Springer (2005) 276–288
19. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. *SIAM J. Comput.* **18** (1989) 186–208
20. Goldreich, O.: Zero-knowledge twenty years after its invention. Technical report, Weizmann Institute of Science, Israel (2002) updated 2004.
21. Balfanz, D., Durfee, G., Shankar, N., Smetters, D., Staddon, J., Wong, H.C.: Secret handshakes from pairing-based key agreements. In: *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, Washington, DC, USA, IEEE Computer Society (2003) 180
22. Abadi, M., Fournet, C.: Private authentication. *Theor. Comput. Sci.* **322** (2004) 427–476
23. Sarma, S., Weis, S., Engels, D.: RFID systems and security and privacy implications. In Kaliski, B., Kaya o, c., Paar, C., eds.: *Cryptographic Hardware and Embedded Systems – CHES 2002*. Volume 2523 of *Lecture Notes in Computer Science.*, Redwood Shores, CA, USA, Springer-Verlag (2002) 454–469 Knapper berblick ber technische Funktionsweise von RFID. Eine der ersten Publikationen zum Thema RFID und Privacy.
24. Molnar, D., Wagner, D.: Privacy and security in library RFID: Issues, practices, and architectures. In Pfitzmann, B., Liu, P., eds.: *Conference on Computer and Communications Security – ACM CCS*, Washington, DC, USA, ACM, ACM Press (2004) 210–219
25. Diffie, W., Hellman, M.E.: New directions in cryptography. *IEEE Transactions on Information Theory* **IT-22** (1976) 644–654
26. Stajano, F., Anderson, R.: The resurrecting duckling: Security issues for ad-hoc wireless networks. In Christianson, B., Crispo, B., Roe, M., eds.: *Security Protocols*, 7th International Workshop Proceedings. (1999) 172–194
27. Hoepman, J.H.: Ephemeral pairing on anonymous networks. [28] 101–116
28. Hutter, D., Ullmann, M., eds.: *Security in Pervasive Computing*, Second International Conference, SPC 2005, Boppard, Germany, April 6-8, 2005, Proceedings. In Hutter, D., Ullmann, M., eds.: *SPC*. Volume 3450 of *Lecture Notes in Computer Science.*, Springer (2005)