Use of Pseudonyms in Smart Retail Applications*

PIC2 - Master Degree in Telecommunications and Informatics Engineering
Instituto Superior Técnico, Universidade de Lisboa

Guilherme Santos — 96740†
guilherme.silva.santos@tecnico.ulisboa.pt

Advisor: Professor Luís Rodrigues

Abstract Smart retail technologies have the potential to improve the customer experience but also pose a threat to the customer privacy. A possible strategy to preserve customer privacy in smart retail applications is the use of pseudonyms. In this report, we survey how pseudonyms have been used to preserve privacy in different application areas and discuss the advantages and limitations of different approaches to manage pseudonyms. Based on the limitations of previous work we propose to design and implement techniques to increase the privacy guarantees in the use of pseudonyms, which can be of value in the smart retail domain.

*This work was supported by national funds through IAPMEI via the SmartRetail project (ref. C6632206063-00466847).

†I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa (https://nape.tecnico.ulisboa.pt/en/apoio-ao-estudante/documentos-importantes/regulamentos-da-universidade-de-lisboa/).

Contents

1 Introduction

Smart retail is a general term that captures the use of technologies in retail, in particular digital technologies, to improve customer experience and the efficiency of operations for retailers. Most smart retail applications require customers to use digital identities while shopping. These entities are used to simplify payment but also to profile the user behaviour. Customer profiling may bring some advantages to the customer, such as having access to personalized recommendations or promotions but also poses a serious threat to the customer privacy. Information regarding the customer behaviour may disclose personal information, such as health conditions, drinking habits, food regime, etc, that can be used later, against the client interests, for instance, when defining the price of insurance policies, which raises deep ethical concerns.

A possible strategy to preserve customer privacy in smart retail applications is the use of pseudonyms. The use of pseudonyms has been proposed for several other applications areas, such as smart vehicles, where similar concerns are raised. In this report we survey how pseudonyms have been used to preserve privacy in different application areas and discuss the advantages and limitations of different approaches to manage pseudonyms.

We identify a number of properties of pseudonyms that are relevant in the context of smart retail applications. These properties include, among others: *pseudonym unlinkability*, that guarantee that two different pseudonyms from the same client should be impossible to relate between each other; *revocability*, such that misbehaving customers can be prevented from using previously acquired pseudonyms; *backward unlikability*, ensuring that the revocation of pseudonyms does not disclose information about the past use of non-revoked pseudonyms; and *revocation auditability*, that allows clients to verify if a pseudonym has been revoked or not, before using it. Besides assuring all these properties, a pseudonym scheme should also be eficient considering the time needed for each authentication, the memory needed for storage in customers, verifiers and in any central authority, the size of the revocation lists, and the amount of time and computation needed for a central authority to revoke a given client.

In this report we also identify some limitations of existing systems when providing these properties. In particular, most existing systems don't provide the essential properties of perfect *backward unlinkability* nor *revocation auditability*. The pseudonym schemes that provide these essential properties achieve them by, using Zero-knowledge proofs that are very resource intensive or purely based on asymmetric encryption at the expense of high latencies in customers' authentications and huge amounts of time and computation needed for revoking a client. Based on the limitations of previous work we propose to design and implement techniques to increase the efficiency of *revocation auditability* and perfect *backward unlinkability*, which can be of value in the smart retail domain.

The rest of the report is organized as follows. This report starts by summarizing our goals and expected results in Section 2. Then, Section 3 presents the key concepts in the use of pseudonyms. In Section 4, analyse existing works in this field, discussing their advantages and limitations. We propose a novel pseudonym scheme that aims at improving *revocation auditability* and perfect *backward unlinkability* in Section 5. We describe how we plan to evaluate our scheme in Section 6 and detail our work schedule in Section 7. Finally, Section 8 concludes the report.

2 Goals

Our work studies the use of pseudonyms on smart retail applications and, in particular, finding efficient ways of offering some key properties such as *revocation auditability*. More precisely:

> *Goals:* We aim at designing a novel scheme to support the properties of *revocation auditability* and perfect *backward unlinkability* efficiently, keeping reasonable, the authentication latencies, the size of revocation lists and the amount of time needed for a central authority to revoke a given client.

For this purpose we will consider the use of asymmetric encryption, bloom filters, and redactable signatures. The project is expected to produce the following results.

> *Expected Results:* The work will produce i) a specification of a novel *revocation auditability* and perfect *backward unlinkability* algorithm; ii) an implementation of the proposed algorithm, iii) an experimental evaluation of the performance of the resulting implementation.

3 Background

In this section, we introduce the most relevant concepts for our work.

3.1 Authentication

Authentication is the process of an entity proving to be what it claims to be. This process is usually done when an entity wants to access a protected system or when decides to do a sensitive operation. There are two entities in a process of authentication: the entity that tries to authenticate and verifier that controls this process. The system verifier ensures that the claimed identity is valid by checking information provided by the entity that is proving its identity, such as credentials or biometric data. This raises the problem of a system administrator being able to trace and get information about all the previous authentications and operations of all the entities that authenticated in the system. This is particularly concerning for smart retail applications, where customers need to authenticate to acquire goods and retailers can trace the consumer history. The use of pseudonyms during authentication can circumvent this threat.

3.2 Pseudonyms

Pseudonyms are alias that clients can use to hide their unique identity. They can be applied to a wide range of privacy sensitive applications. In information systems, pseudonyms can be used for authentication, replacing the clients' unique identity. Pseudonyms need to be used properly to ensure privacy preservation. In particular, the use of the same pseudonym for multiple authentications raises privacy risks. A system administrator with access to authentication records is able to link the authentications that use the same pseudonym and this may be enough to link the pseudonym to a concrete user. For this reason, users should avoid using the same pseudonym in different authentications. Ideally, users should use a different pseudonym for each authentication.

Changing pseudonyms can prevent the activity of the user from being tracked only if the pseudonyms are unlinkable. The *unlinkability* property states that two pseudonyms cannot be associated to each other, even if they belong to the same user, i.e., that there is no way to distinguish pseudonyms from the same user from pseudonyms from different users.

3.3 Pseudonym Schemes

In this section, we discuss some problems that pseudonym schemes must overcome and some properties that these schemes must provide.

The first challenge a pseudonym scheme must overcome is the need for systems to have enough trust to let customers authenticate with pseudonyms without knowing their real identity. Most of the existing pseudonym schemes have the pseudonyms used by the clients being issued by a central authority, a pseudonym provider, trusted by both the clients and the verifiers, with the verifiers trusting that the central authority will only provide valid pseudonyms to legitimate clients and with the clients, trusting that the central authority will not disclose the information of mapping between the clients and their pseudonyms.

In many systems, if a client misbehaves, it should not be allowed to further authenticate in the system. In order to achieve this requirement, the pseudonym scheme must allow *revocability*, which is the ability of revoking a pseudonym. Revoking pseudonyms in a way that preserves *unlinkability* may be challenging. For instance, assume that a pseudonym is revoked by including it in a revocation list that is sent to verifiers. If, when a client misbehaves, all pseudonyms of that client are included in the same revocation list, verifiers may infer that the pseudonyms in the list belong to the same client. If some of these pseudonyms have been used in the past, the verifiers can link the corresponding authentications. To avoid this problem, a pseudonym scheme should guarantee *backward unlinkability* and *revocation auditability*.

Revocation auditability states that a client should be able to see his revocation status, before each authentication, avoiding the situation where a client tries to authenticate with a pseudonym that has been revoked without being aware of that fact.

Backward unlinkability states that past authentications must remain private when a client is revoked. Many existing pseudonym systems achieve this property by having the pseudonym providers issuing pseudonyms

divided by periods of time, denoted time slots, they should be used in. When a client chooses a pseudonym to use, they use a pseudonym associated with the current time slot. When a client is revoked, only the pseudonyms associated with future time slots are distributed to the verifiers. This solution has one problem, which is, it cannot guarantee perfect *backward unlinkability*. If a client is revoked in a given time slot and the pseudonyms associated to that time slot are distributed to verifiers, if the client has already authenticated in the system in that given time slot, then those authentications' privacy is compromised. If the pseudonyms of the current time slot are not distributed, then the client can authenticate using those pseudonyms until the end of the time slot, when they should be revoked and not authenticating anymore. In order to alleviate this problem, one could consider to increase the granularity of the time slots, making each time slot really small. This solution has the drawback of increasing substantially the number of pseudonyms that need to be issued to each client. In systems that use this approach, the number of pseudonyms issued increases linearly with the time slots' granularity, providing the clients with many more pseudonyms than they actually need, increasing the memory needed to store them.

Pseudonym schemes should also include the property of *accountability*, which states that it must be possible to trace the actions of a user using a pseudonym back to their real identity. This property is usually assured by maintaining a map between user-pseudonyms in a central trusted authority.

Additionally, a pseudonym scheme must also be efficient, namely, it should be possible to authenticate with low latency, the amount of information distributed in revocation lists should not be large, the memory space required to execute the protocol should be small, etc.

3.4 Abstractions Used in Pseudonym Implementations

In this section, we provide some key concepts that are used in existing pseudonym schemes or that we use in our proposed architecture.

3.4.1 Asymmetric Cryptography

Asymmetric cryptography is a cryptographic system based on a pair of keys: a private key and a public key. This type of system states that each entity involved must have a pair of keys, publish the public key and keep the private key, private. Each key works as the decryption key of its pair. Asymmetric cryptography can be used to achieve important properties in information security such as: authentication, integrity, and non-repudiation. For this reason, asymmetric cryptography is used in most existing pseudonym schemes, using public keys as pseudonyms.

Encryption with Asymmetric Cryptography The encryption is done by encrypting the message with the public key and decrypting it with the private key. Since anyone can use the public key, but only the owner has access to the private key, only the owner can decrypt the message.

Digital Signatures with Asymmetric Cryptography Asymmetric encryption allows the creation of digital signatures that are a cryptographic tecnique that allows to verify the authenticity and integrity of a digital message or document. The verification of digital signatures ensures that the content has not been altered and prove the identity of the sender. Digital signatures are usually created, simply, by passing the message through an hash function and encrypting the resulting digest with the private key. They are sent attached to the messages. The receivers can check a signature by decrypting it with the public key of the sender, passing the message through the same hash function and comparing the results. The integrity property comes from the properties of hash functions, that it is infeasible to find two messages with the same digest. The authenticity property comes from the fact that only the owner of the pair of keys has access to the private key, so that no one else has the ability to create such digital signature.

Authentication with Asymmetric Cryptography As explained before, authentication is the process of an entity proving to be what it claims to be. In the context of using public keys as pseudonyms, this corresponds to an entity proving that it is the owner of a pseudonym (public key) that it is presenting to the system verifier. This is achieved by creating a digital signature and attaching it to the messages sent using the private key associated

to the public key in the pseudonym. The receiver, the system verifier, can then verify the digital signature and check if the authenticating entity is who it claims to be.

Ensuring Integrity with Asymmetric Cryptography Integrity ensures that messages have not been altered during their transmission, by man-in-the-middle attacks for instance. This property is vital for sensitive applications, where users' actions must not be changed by an attacker. Asymmetric encryption can also be useful to provide this property. Having public keys as pseudonyms allow the users to use the corresponding private keys in order to create digital signatures that enable the system to check if the messages sent by the users were altered during their transmission.

Ensuring Non-Repudiation with Asymmetric Cryptography Non-repudiation is achieved when a sender cannot deny the authenticity of the messages sent and operations made. This property is also accomplished using digital signatures. Since digital signatures are only created using the private key and only the owner of the pair of keys has access to it, it is impossible for the senders to deny the authenticity of the messages they sent. This property is closely related to the property of *accountability* of pseudonym schemes, and these two properties together, make it possible to hold an authenticated user accountable for their actions.

3.4.2 Blind Signatures

Blind Signatures are a cryptographic technique that is a form of digital signature in which the content of the digital message or document being signed, is hidden during the process of signing. Different parties are still able to verify the signature when they receive the digital message and its signature, if they have access to the signer's public key. The signer is not privy to what he signed. Blind signatures can be useful in pseudonym schemes where a signer must not know what they signed. For example, in V-token [1], during the process of pseudonym issuance, the scheme uses blind signatures to prevent the central authority of storing pseudonym-identity maps.

3.4.3 Zero-Knowledge Proofs

Zero-knowledge proofs are cryptographic protocols that allow a given entity to prove knowledge about a certain information or statement to another entity wihout revealing the content of that information. Zero-knowledge proofs can be used in pseudonym schemes, during authentications where the clients prove to the system verifiers that none of their previous used pseudonyms are revoked, without revealing which pseudonyms they used before, preventing a system administrator from relating different past authentications. Zero-knowledge proofs have the obvious advantage of privacy and preventing the system verifiers of knowing which pseudonyms, clients used before. On the other side, Zero-knowledge proofs are computationally expensive.

3.4.4 Accumulators

Dynamic accumulators [2] are a constant-size cryptographic construct that represents a set membership. It is possible to add or remove elements from the accumulator. Besides that, accumulators allow anyone to prove in zero knowledge that some element is in that accumulator if and only if, the element is in the accumulator. It is possible to use these accumulators as a "whitelist" of valid pseudonyms. Universal accumulators [3], allow users to prove instead that an element is not in the accumulator. Universal accumulators can be used as a "blacklist" of revoked pseudonyms.

3.4.5 Bloom Filters

Bloom filters [4] are a probabilistic structure designed to store members of a set and to determine if a given element is member. In pseudonym systems, verifiers often use bloom filters for the storage of the revoked pseudonyms. Bloom filters are very eficient and have constant computation costs $O(1)$ for storing and searching elements. The filter is made up of N addressable bits, with addresses from 0 to N-1, being N its size. Besides, its size, a filter also has another paramenter which is the number of hash functions, k. When an element is to be inserted in the filter, it is hash coded using the hash functions, obtaining k hash codes, that will be used as addresses inside the filter. After that, the bits of the filter with that addresses, are set to 1. To test whether an

element belongs to the filter, the element is hash coded using the hash functions obtaining k addresses, if and only if the bits in the filter with that addresses are all set to 1, the element is considered to be in the filter. When searching for an element, bloom filters may indicate that the element is in the filter when it is not, allowing false positives to happen. On the contrary, false negatives are not possible. The rate of false positives depends on the size of the filter, N, the number of elements inserted, m, and the number of hash functions used, k. The rate of false positives can be calculated using the following formula:

$$P = (1 - (1 - \frac{1}{m})^{kn})^k \tag{1}$$

Bloom filters also provide an efficient operation to merge different filters of the same size into one. If the bloom filters have the same characteristics, as hash functions and size, it is possible to merge different filters by simply performing a bitwise OR operation.

3.4.6 Redactable Signatures

Redactable Signatures [5] make possible to send a signed message with certain parts deleted but allowing the receiver to verify its integrity even when can't read the deleted parts. To create such a signature, it's created a binary tree, the message is split into chunks of arbitrary size and then, each chunk is associated to a leaf of the binary tree. In the next step, it is assigned a key, k, for each node in the tree. To achieve this, it is generated a random key k for the root and, recursively, starting at the root, are calculated the keys k of the children nodes, with a pseudorandom generator, $< k_{left}, k_{right} >= G(k)$. After the keys are assigned, it is created a hash value v for each leaf as $v = H(0, k, x)$, being x the chunk of the message associated with the leaf. Then, recursively, are calculated the values v for each one of the remaining nodes, calculating the hash of the concatenation of the childrean nodes' values: $v = H(1, v_{left}, v_{right})$. Finally, root's value v is signed. When sending a partial message, the sender sends the chunks he wants and the keys k associated with those leaf nodes, the signature of the root's value v and all the hash values v of leaf nodes associated to chunks that were hidden. It is possible to compact the hash values v and the keys being sent to the client, using the structure of the binary tree. In order to check the validity of the message, the receiver would, recursively, calculate all the hash values v until the root, finishing by verifying the signature of the root's value v.

4 Related Work

In this section, we discuss some relevant work in the field. We explain some existing pseudonym schemes detailing which properties they achieve and how, while discussing their flaws.

We start by explaining a general structure that almost every pseudonym scheme follows. Generally, this kind of schemes are usually composed by 3 entities: the clients, the system where the clients want to authenticate themselves while preserving their privacy, and a central authority trusted by both parties, a pseudonym provider, which provides clients with enough pseudonyms for their authentications and handles clients' revocations, distributing the revoked pseudonyms to the system verifiers using revocation lists.

The pseudonyms issued have normally an asymmetric key pair associated and are made up of the public key and the pseudonym provider's signature. When a client is authenticating in the system, he uses a pseudonym as his digital identity and sends a different one to the system verifier in each authentication. The purpose of the pseudonym's signature is to prove to the verifier the authenticity and integrity of the pseudonym showing that the pseudonym was indeed created by the pseudonym provider. This scheme takes advantage of the asymmetric encryption properties to prove that the client is the real owner of a given pseudonym, by using the associated private key. So, the messages exchanged between the system verifier and a client are usually appended with the pseudonym and a signature made by the client using the private key associated to that pseudonym.

These schemes also follow similar steps, in a similar order. Whenever a client joins the system must start by asking the pseudonym provider for pseudonyms for his authentications. After acquiring his pseudonyms, a client is ready to authenticate in the system with a verifier. The authentication is divided in three phases: the verifier sends to the client, information about his revocation status, the client checks this information and if he is not revoked, sends a pseudonym to the verifier, the verifier checks the validity and authenticity of this

pseudonym and if everything is correct the user accesses the system. If an authenticated client misbehaves, the system generates a complaint, inserting the pseudonym the client used, and sends it to the pseudonym provider. The pseudonym provider links the pseudonym inserted in the complaint to the real identity of a client and then revokes all the pseudonyms issued before to that client and distributes them in a revocation list to the system verifiers.

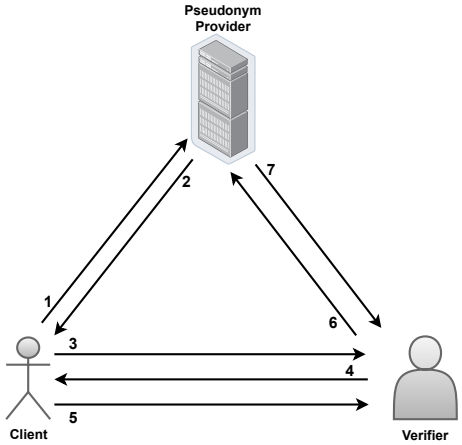

Figure 1: (1) The client sends a request for pseudonyms. (2) The pseudonym provider sends a batch of pseudonyms to the client. (3) The client sends a request for authentication. (4) The system verifier sends the revocation list to the client. (5) The client checks his revocation status and sends a pseudonym. (6) If a client misbehaves, the system verifier decides to generate a complaint and sends it to the pseudonym provider. (7) The pseudonym provider revokes the client and sends the pseudonyms previously issued to the verifiers.

This is just a general approach followed by most schemes, but some other schemes may present some radical approaches or simply do not implement some steps of this approach. In the next sections we take a deeper look at existing pseudonyms and discuss them.

4.1 Approaches based on Epochs and Time Slots

As mentioned in the Background section, some pseudonym schemes have a similar approach to achieve *backward unlinkability*. They divide the time into large periods of time denoted as epochs and these are divided into smaller intervals denoted as time slots. The pseudonyms issued to a given client are divided into groups and each group is associated to one of the slots and the pseudonyms cannot be used outside of their slot. When revoking a client, only pseudonyms associated to future slots are inserted in the revocation list and distributed to the verifiers.

As explained in section 3.3, this type of approach does not achieve perfect *backward unlinkability*, due to the fact that revoking a client in a given time slot has the problem of exposing the user's authentications in that given time slot or the problem of letting a misbehaving user continue to authenticate in the system until the end of that time slot. This can be alleviated augmenting the granularity of the time slot which would increase linearly the number of pseudonyms issued. This type of schemes have the challenge of finding a good trade-off between the size of the slots, that should be as small as possible, and the number of pseudonyms issued.

Another disadvantage is the fact that this approach forces to provide the client with pseudonyms for all the time slots that the client can possibly authenticate in the system, even if the client only authenticates in the system in one time slot, providing the client many more pseudonyms than he actually needs. An ideal solution would be to provide the client as many pseudonyms as the number of authentications he performs.

Next, we present some pseudonym schemes that use this notion of epochs and time slots.

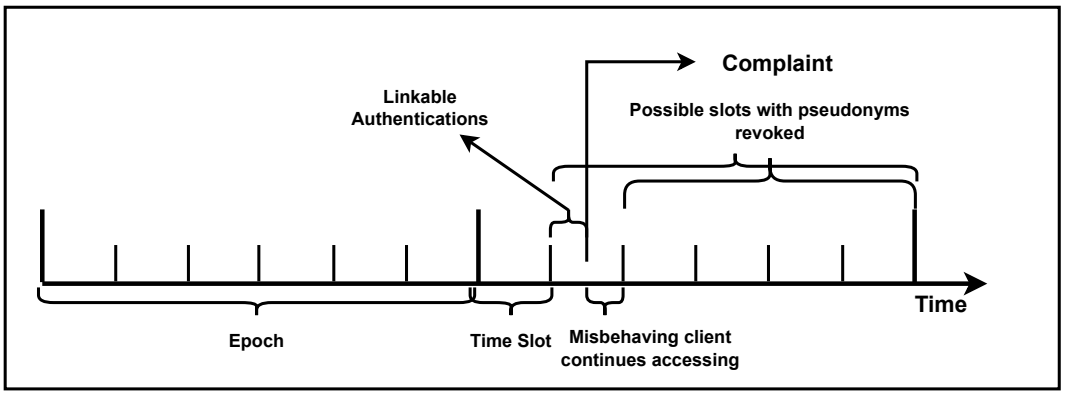

Figure 2: Epochs and time slots approach.

4.1.1 Haas et al.

Haas *et al.* [6] proposes a lightweight mechanism to revoke pseudonyms in an anonymous scheme developed for the authentication of messages exchanged between vehicles in VANETs.

The pseudonyms are issued by a central certificate authority, being also responsible for revoking the pseudonyms of misbehaving vehicles. This work uses the concept of epochs, a large period of time for which the pseudonyms are issued, and each epoch is divided in multiple time slots. The pseudonyms that are issued for a given vehicle and epochs are divided in groups, and each group is assigned a time slot where the pseudonyms of the group must be used. The paper achieves partly the property of *backward unlinkability* keeping the past authentications of the client private, by revoking only pseudonyms of future time slots.

This work developed an algorithm for the creation of the pseudonyms to achieve an efficient way of generating and revoking the pseudonyms. Whenever the certificate authority receives a registration request from a client, it creates a nonce which will be used to create a hash chain as long as the number of time slots in the epoch. Each element of the chain is associated to a time slot and is created hashing the previous element, being the first one obtained hashing the nonce created for that client. After generating the chain, the certificate authority generates the groups of pseudonyms to be sent to the client. For each time slot, it generates a sequence of values from 1 to R, being R the size of the pseudonym group, and encrypts each of these values using the associated element of the hash chain as the key of a cipher. Each of these ciphertexts is the identifier of a pseudonym and is inserted on it. The certificate authority generates pairs of asymmetric keys and include one of these pairs in each pseudonym, finishing their creation by adding its digital signature generated from the identifier and the public key of the pseudonym. After all these steps, the certificate authority replies to the client sending the pseudonyms it just created.

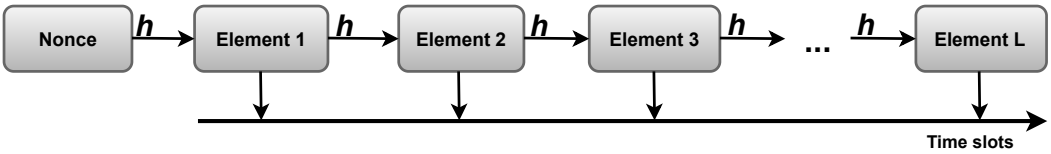

Figure 3: Hash chain.

In order to authenticate himself, a client sends his message along with a pseudonym of a group associated with the current time slot. Whenever a client receives a message, it checks the pseudonym that comes with it, verifying the digital signature and checking that its identifier is not in its "blacklist" of revoked pseudonyms. This work also includes a way to improve the efficiency of the memory necessary to store the revoked clients' information and the look-ups in the revocation list, which is the usage of bloom filters to store the revoked pseudonyms.

To revoke a client, the certificate authority publishes and distributes the element of the hash chain associated

to the current time slot. With this information, all the vehicles perform the same steps used in the creation of the pseudonyms and obtain the identifiers of the revoked client's pseudonyms for the current time slot and the future ones. The privacy of the revoked client's past authentications is assured by the irreversibility property of hash functions which guarantees that given an element of the chainm, it is not possible to calculate the previous elements.

This approach has several advantages such as the minimal information necessary for the certificate authority to store about each customer which is just the nonce associated to the client's registration, the amount of information necessary for the certificate authority to revoke a client which is just the element of the hash chain associated with the time slot when the client was revoked.

On the other side, this scheme is based in epochs and time slots, which does not allow to achieve revocation with perfect *backward unlinkability*. It is possible to argue that it would be possible to shorten the time slots in order to make it infeasible to revoke a client in a time slot that the client has already authenticated in, but this would increase the number of pseudonyms necessary to be issued to each client linearly with the granularity of the time slots leading to a huge number of pseudonyms issued and not used by the clients. It must be found a good trade-off between the size of time slots and number of pseudonyms issued.

4.1.2 EDGAR

EDGAR [7] introduces a new class of pseudonyms known as "Range-Revocable Pseudonyms" that can be revoked for any time-range within its original period. This scheme, also constructed with the concept of epochs and time slots, tries to overcome some flaws of Haas *et al.* [6]. This scheme has a pseudonym manager issuing clients' pseudonyms and each pseudonym has an associated key pair, used by the clients to authenticate themselves. The proposed scheme solves the problem of issuing many more pseudonyms than the client actually needs, permitting to issue only as many pseudonyms as the number of authentications done by the client. This is achieved because the information distributed when revoking a given pseudonym cannot be linked with its usage outside of the revoked time-range. This paper also improves the perfect *backward unlinkability* problem, but does not achieve it totally.

This work uses the concepts of pseudonyms and capabilities. When joining the system, the client asks the pseudonym provider for pseudonyms and receives a bunch of these, valid for the current epoch. The pseudonym manager signs the pseudonym to assure its integrity and authenticity. Capabilities are used for authenticating with the system verifiers.

In the beginning of each epoch, it is constructed a tree with as many leaf nodes as the number of slots in an epoch and each slot is associated with a leaf node. Each node is assigned a label. When authenticating, the clients must generate a capability using one of their pseudonyms, for the specific time slot they are authenticating in. To generate a capability, the user signs the label of each node that form the path between the root node and the leaf node associated to that specific time slot, using the pseudonym's private key. These signatures are denoted latchkeys and a capability is defined as the set of these latchkeys, the public key and the signature of the pseudonym provider of the pseudonym which originated this capability. After creating the capability, the client sends the capability and the pseudonym to the system verifier, that checks if that capability is valid using the pseudonym's public key, and not revoked.

To revoke a pseudonym, the pseudonym manager must revoke the capabilities of that pseudonym for future time slots. A capability is valid only when none of its latchkeys is revoked. In order to revoke a capability associated to a given time slot, the pseudonym manager can revoke the latchkey of the leaf node. If the pseudonym manager wants to revoke capabilities of several time slots, it can revoke inner nodes' latchkeys of the tree, revoking all the capabilities associated with leaf nodes of the sub-tree that has those inner nodes as root. The pseudonym provider then calculates the latchkeys for the pseudonym being revoked, using its private key, and distributes them to the verifiers.

When a verifier receives a capability, verifies the pseudonym manager's signature, with the public key, verifies if the latchkeys were originated using the private key of the pseudonym and if they were originated from the labels of the tree's nodes associated with that time slot. Then it checks, if any of the latchkeys is in the revocation list. If none of them is, the client is authenticated.

The capabilities generated from the same pseudonym have intersecting information, so the user must never use the same pseudonym twice, as the two authentications could be linked. When revoking a client, the pseudonym

manager must not revoke latchkeys which may have already been used in capabilities of previous time slots.

This work allows to increase the granularity of the time slots, without increasing the number of pseudonyms used, but at the cost of incresing the revocation information to be distributed to the verifiers, because all the future possible capabilities have to be revoked and the number of capabilities increase with the granularity. Despite this increase, the tree system allows to alleviate this problem, revoking latchkeys of inner nodes.

This aproach has the advantage of the amount of information necessary for the pseudonym manager to store about each client, which is just the clients' identifiers. This work uses a pseudorandom function with a seed composed of the client identifier, epoch and time slot, in order to create the pseudonyms for a given client. With these client identifiers, when revoking a client, the pseudonym manager can reconstruct the seeds, generate the pseudonyms again and distribute them to the verifiers.

When compared with Haas *et al.* [6], this work allows to increase the granularity of time slots, in order to attenuate the perfect *backward unlinkability* problem, at the expense of having much bigger revocation lists.

4.1.3 NYMBLE

This work [8] designed a pseudonym scheme for anonymizing networks such as Tor. Clients use this kind of networks, when accessing websites, to hide their IP addresses through a set of nodes belonging to the network. Since website administrators can't block a misbehaving user by blocking his IP address, they usually block the IP address of the exit node, blocking not only the misbehaving user but also all the other legitimate users. The presented scheme tries to solve this problem.

The scheme is composed by four entities: user, pseudonym manager, nymble manager and server. The users start by asking the pseudonym manager for pseudonyms and then use those pseudonyms to get a credential from the nymble manager. With this credential the user is able to authenticate in a given server keeping their privacy. The detailed process is explained below.

Like in the previous systems, time is also divided in epochs and each epoch is divided in slots. The client starts by asking the pseudonym manager for a pseudonym when joining the system. After this first step, when a client decides to access a given server, the client uses his pseudonym to get a credential issued by the nymble manager. Each credential is associated with only one server, so if the client wants to access multiple servers, he needs to get as many credentials from the nymble manager as different servers he wants to access. Each credential is made up of as many nymble tickets as the number of slots of an epoch, and each nymble ticket is associated with a given slot and must be used during it. Posteriorly, the client presents one of these nymble tickets to authenticate in the server. In the first step of this system, the pseudonym manager receives a request from the client and returns a pseudonym valid for the current epoch. This pseudonym is made up of two MACs. The first one is created using the client's identifier, the epoch and a secret key only known by the pseudonym manager. The second one is created with the first MAC, the epoch and a secret key that is shared by the pseudonym manager and the nymble manager.

When a client presents this pseudonym to the nymble manager, the nymble manager starts by verifying the pseudonym is legitimate by checking the second MAC. If the MAC is valid, it proceeds to the creation of the credential. It generates a seed associated with the pseudonym presented by the client and then, creates a hash chain using that seed, as long as the number of slots per epoch plus one, using an irreversible hash function f. Each element of this chain, besides the initial element, is associated with a slot. The second element is associated with the first slot, the third element is associated with the second slot and so forth... With this chain, the nymble manager creates the credential, it passes the first element of the chain through another irreversible hash function g to create the credential identifier, denominated nymble* and after that, starts iterating through the chain passing the elements through the hash function g to create the nymble tickets' identifiers. In order to create each nymble ticket, are created two MACs and a ciphertext, the first MAC is created with the server's identifier, the slot associated to the nymble ticket, the epoch, the nymble ticket's identifier and a secret key known only by the nymble manager. The second MAC is created with all the information used in the creation of the first one plus the first MAC itself, but using a secret key shared between the server where the client will use this credential and the nymble manager. The ciphertext, is created encrypting the concatenation betweeen the identifier of the credential, nymble*, with the element of the hash chain associated with the slot that the nymble is being created for. Each nymble ticket will be a tuple made up of, the slot that the ticket must be used in, the nymble ticket's identifier, the two MACs and the ciphertext. After the previous step is completed and all the

tickets are created, the tickets are put together and inserted in the credential and the last is sent to the client.

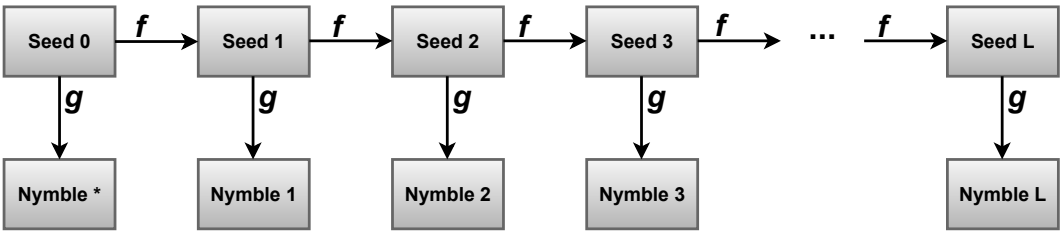

Figure 4: Creation of Nymble tickets.

With the credential, a client can authenticate in a server, sending the ticket associated with the current slot. When the server receives a ticket, it verifies the ticket, checking if the ticket was indeed created by the nymble manager using the second MAC of the ticket and checking if the nymble ticket is not revoked, by verifying if it is present in the server's "blacklist". The server allows the authentication if this verification is successful.

When a client misbehaves at a given server, the server has the capability of revoking that client by generating a complaint and sending it to the nymble manager, including the nymble ticket that was used by the client when connecting. Receiving a complaint, the nymble manager checks the authenticity of that nymble ticket using the first MAC inserted in it, decrypts the ciphertext and gets the nymble* which is the identifier of the credential where that nymble ticket* is inserted. The nymble manager returns the element of the chain created when creating that credential, associated to the next time slot. With this element, the server is capable of generating the nymble ticket's identifiers for the future time slots and revoke them, inserting them in its "blacklist".

As mentioned before, this scheme provides the clients with *revocation auditability*, by allowing them to verify their revocation status before authenticating in the system. In the beginning of each authentication, the server sends its "blacklist" to the user and the user only proceeds with the authentication if his revocation status is negative.

This method could raise some problems, such as, the server sending an outdated "blacklist" to the client. The authors present a solution, that allows the clients to check the integrity and freshness of the "blacklist". This is accomplished by having the nymble manager signing the "blacklist" for every server in the beginning of each time slot. When a client receives the "blacklist" from a given server, he checks the nymble manager's signature and verifies that the "blacklist" corresponds to that time slot and it is complete.

Comparing this system with the previous ones, Nymble, obviously, provides *revocation auditability* which is the biggest improvement. Besides that, the pseudonym issuing is divided by 2 central authorities and none of them, has the ability to resolve, nymble tickets or pseudonyms to a real identity, which increases the privacy of user that is not exposed to a single central authority. Despite of these improvements, the nymble manager keeps some of the flaws of the previous works. It is a scheme based in epochs and time slots, which does not allow to achieve revocation with perfect *backward unlinkability*. This problem is aggravated by the mechanism used to achieve *revocation auditability*. The need of having the nymble manager signing the "blacklists" in the beginning of each time slot, forcing the nymble manager to sign all the "blacklists" in the beginning of each slot, forces the nymble manager to store these "blacklists", or to generate them in the beginning of each slot, increasing the memory space used or the computation necessary. This problem, makes impossible to reduce the time slots to alleviate the perfect revocation problem, since, once reducing them, this signing process would be repeated more often.

4.1.4 IFAL

IFAL [9] proposes a pseudonym scheme for the authentication of messages exchanged between vehicles in VANETs. This work presents an original approach. While other existing pseudonym schemes issue pseudonyms to clients, that are able to authenticate in the system right after receiving the pseudonyms, IFAL has the approach of pre-issuing the pseudonyms to clients and activate them later by sending activation codes. These activation codes allow the vehicle, to generate the private keys associated to the pseudonyms.

This work has the advantage of not needing to revoke the pseudonyms and distribute them to the verifiers in revocation lists, unlike other systems. Instead of revocating the pseudonyms of a misbehaving client, IFAL stops sending activation codes to a revoked client and this client runs out of valid pseudonyms and is prevented from authenticating again.

This system has the following entities: Enrollment Authority, EA, responsible for the vehicles' registration in the system, Authorisation Authority, AA, responsible for issuing the vehicles' pseudonyms. In this model, the vehicle starts by generating a registration request and sending it to the Enrollment Authority. The EA assigns a unique uid to the vehicle, signs the request together with the uid and sends it back to the vehicle. Next, the vehicle sends the information, it received from the EA, to the AA in a request for pseudonyms. The AA checks the information received and generates a certificate file with multiple pseudonyms, and the activation codes for the pseudonyms. The certificate file is returned to the vehicle and the activation codes are stored together with the vehicle's uid.

Time is divided in epochs and the pseudonyms inserted in the certificate file are divided into groups and each group is associated to a given epoch. Pseudonyms cannot be used outside of the epoch they are associated to. In the activation phase, the AA distributes activation codes to vehicles and an activation code permits to generate the private keys for all pseudonyms associated to a given epoch. The AA maintains in its storage, a mapping between uids and activation codes generated for each uid. The AA iterates through this list and sends the activation code for the next epoch to the client, through the EA. The EA maintains in its storage, a mapping between uids and the canonical identities of the vehicles. The EA receives the activation codes from the AA and sends them to the vehicle corresponding the uid indicated by the AA.

When a vehicle receives an activation code, it calculates the private keys for the pseudonyms of that epoch, and once it gets the keys, the vehicle is ready to authenticate its messages.

In order to revoke a vehicle, there are two possible ways. The first way is that the EA receives a request to revoke a vehicle with its canonical identity. In this case, the EA gets the vehicle's uid and sends it to the AA informing the AA to stop issuing activation codes to the vehicle with this uid. In the second mechanism, the AA receives a complaint about a misbehaving vehicle with the pseudonym used in the authentication. The AA uses this pseudonym to recover the uid of the vehicle and stops sending activation codes for this vehicle.

This system does not achieve immediate revocation, as it is necessary to wait for the beginning of the new epoch for the revoked client to run out of pseudonyms. This is as problematic as the size of the epochs and reducing them would not be a solution, since it would increase the number of pseudonyms used and the number of activation codes sent. This work has a good improvement which is the replacement of distribution of revocation lists by the the distribution of activation codes. The distribution of activation codes would be a lot much efficient since the activation codes are much smaller than the revocation lists. This would be good in a scenario where the number of verifiers are similar to the number of clients as a VANET, where all vehicles work as verifier and client, but not in a scenario where there are very few verifiers comparing to the number of clients as in smart retail applications. The last scenario, would be a scenario where the overhead caused by revocation lists distribution would not be so high and there would be much more activation codes circulating if we used this approach, than revocation lists if we used the previous approaches. Even if the activation codes are more efficient, this would be a downgrade overall.

4.2 Other Approaches

Some schemes present different strategies, other than the notion of epochs and time slots. This is the case of V-tokens [1], PEREA [10] and Privacy Keeper [11].

4.2.1 V-token.

V-token [1] is a pseudonym scheme developed for VANETs where vehicles need to authenticate their messages. While most of other similar works rely the resolvability of pseudonyms to issuance authorities that assure this by storing pseudonym-identity mappings, this work argues that these mappings are too privacy sensitive and present a pseudonym issuance protocol that satisfies the resolvability of pseudonyms while preventing the issuance authorities from storing pseudonym-identity mappings.

There are 4 entities in this scheme: vehicles, certificate authorities, CA, pseudonym providers, PP and resolution authorities, RA. When a vehicle joins the system, it is given a long term identifier, id-V, which is a certificate and the corresponding pair of keys, by a certificate authority. A vehicle can obtain pseudonyms from pseudonym providers and before the pseudonyms are issued, it is verified if the vehicle has been revoked. The resolution authorities take part in the resolution process. This work achieves its purpose of preventing the issuance authorities from storing pseudonym-identity mappings by inserting the resolution information directly in the pseudonym. This piece of information is known as V-token.

The protocol of issuance is divided in two phases: Authentication Phase and Acquisition Phase. In the Authentication Phase the vehicle starts by sending the certificate authority a request in order to receive V-tokens, including his id-V and a signature using his long term certificate. The CA checks the vehicle is not revoked and sends back an id which is the concatenation between id-V and id-CA, being id-CA the identifier of this CA, the public key of the resolution authorities and requests N commitments. After this, the vehicle creates N V-tokens, by generating N unique random number for each and appending them to the id it just received, and encrypting this concatenation with the public key of the resolution authority. After generating the V-tokens, the vehicle chooses N random distinct blinding factors and blinds each commitment using one of these factors. The client stores the used factors and the unique random numbers used when creating the V-tokens and sends the set of blinded V-tokens to the CA. The vehicle must prove probabilistically that it has created the V-tokens correctly, using the id provided by the CA. This is achieved by having the CA requesting some blinded V-tokens for the vehicle to reveal. The client then sends the blinding factors and the unique random numbers used when created each of the V-tokens that he must reveal now. With this information the CA, unblinds the V-tokens and check if they were well constructed. At the end of this phase, the CA signs the remaining blinded V-tokens and sends these signatures to the vehicle. The certificate authorities are not able to understand which V-tokens they signed because of the blind signatures, so, they won't be able to trace the pseudonyms to a given vehicle.

During the Acquisition phase, the vehicle gets pseudonyms from a pseudonym provider. Once a vehicle has in its possession signed V-tokens, it contacts a pseudonym provider to get a pseudonym for each V-token. To generate a pseudonym, the vehicle starts by generating a key pair and generates a request to the pseudonym provider, that includes the public key generated and one V-token and signs the request with the private key, showing its ownership of the key pair. When the pseudonym provider receives such request, it starts by verifying the vehicle's signature and checks the validity of the V-token, by checking the CA's signature using the CA's public key. If everything is good, the PP creates a pseudonym, inserting the public key and the V-token received in the request, signs it and sends it to the vehicle.

The process of mapping a pseudonym to a real identity, corresponds to decrypt the V-token inserted in the pseudonym with the private key of the Resolution Authority, in order, to obtain the id-V and the id-CA. With this information would be possible to map the id-V of the vehicle to a real identity, by contacting the Certificate Authority that registered the vehicle and created its V-tokens.

This work does not provide any process for the revocation of already issued V-tokens.

The main purpose of not having a central authority storing pseudonym-identity mappings is achieved. Another good advantage of this scheme, is that the vehicles only need to generate as many pseudonyms as the number of authentications they perform, unlike other schemes. On the other side, the process of generating V-tokens and pseudonyms is complex and perhaps, it creates a little overhead on the vehicles' system. Also, it would be possible for a lucky vehicle to generate invalid V-tokens during the Authentication Phase. For a revoked vehicle, it would be necessary to wait until the client runs out of valid V-tokens before he loses the ability to authenticate in the system, which is also a disadvantage.

4.2.2 PEREA

PEREA [10] presents a pseudonym scheme designed for websites, that provides perfect *backward unlinkability* and *revocation auditability*, without the use of third trusted parties. It uses an accumulator as "blacklist" for revoked pseudonyms, kept by the website system. The property of *revocation auditability* is achieved by having the website sending the "blacklist" to the user in the beginning of each authentication. The property of *backward unlinkability* is achieved by never letting the website system know a set of pseudonyms used by a client.

The clients start by authenticating themselves with the website system and obtain a credential that allows them to generate future pseudonyms to authenticate in the system. Whenever a client authenticates, the client

generates a new "ticket" himself and authenticates in the system with that "ticket" that serves as a pseudonym. In order to authenticate successfully, the client also needs to prove that none of his k last "tickets" is in the "blacklist" of the website system. The key feature of this work, is the use of Zero-knowledge proofs to generate a non-membership proof for all the k last "tickets" the client used, proving that the k last "tickets" are not inserted in the accumulator. The use of Zero-knowledge proofs prevents the website system of relating the set of the last k "tickets" to the user and gain information about his last k authentications. Furthermore, the client also sends another Zero-knowledge proof in order to prove that he didn't replace any of the last legitimate k "tickets". The variable "k" is application dependent, it must be a reasonable value slightly higher than the number of expected authentications before the website system recognizes a misbehavior.

This pseudonym scheme has the advantages of providing perfect *backward unlinkability* and *revocation auditability*. The number of pseudonyms created for each user is equal to the number of authentications performed. On the other side, this scheme uses Zero-knowledge proofs to a very large extent, techniques that are very resource intesive. Moreover, the number of Zero-knowledge proofs used in each authentication depends on the value of k. This scheme may be impractical for applications with a large "k". This pseudonym scheme also has the disadvantage that if a user is able to authenticate more than k times, after he has misbehaved, and before the system notices this misbehavior, he will continue being able to authenticate in the system, even after the system has "blacklisted" his pseudonym used during the misbehavior. The authors argue that it would be possible to limit the clients' authentications. This measure would be a disadvantage by itself and would be impossible to implement in a physical system, like smart retail applications.

4.2.3 Privacy Keeper

This work [11] presents a totally different aproach to achieve perfect *backward unlinkability* when revoking a client, assuring at the same time *revocation auditability* to the clients. This work presents an approach where the client sends a pseudonym and a proof of non revocation in order to authenticate in the system. Instead of revoking the pseudonyms and distributing them directly to the verifiers, the central authority distributes the proofs used by the clients to authenticate. The fact that the verifiers never get to know which pseudonyms are revoked, makes it impossible for them to associate a bunch of pseudonyms to the same user as in other anonymous schemes. Also, this work is not based in epochs and time slots which permits the central authority to provide the client with only as many pseudonyms as authentications done by the client.

The pseudonyms are formed by a public key and the non revocation proofs are digital signatures of a seed created with the private key associated to a pseudonym. When revoking a misbehaving client, the central authority starts by creating a new revocation list, a new seed associated to that revocation list, generated randomly, and then generates the proofs of non revocation for the pseudonyms of the client revoked using the seed it just created and the proofs of non revocation for all the previous revoked pseudonyms. The central authority keeps track of pseudonyms issued to each user. Finally, gets all the proofs together in the revocation list, signs the concatenation between the list and its seed and distributes these elements to the verifiers.

When a client is authenticating, the verifier sends him the last revocation list it received from the central authority, the seed and the signature of the central authority. The client verifies the signature, chooses one of his pseudonyms and calculates the proof of non revocation with the seed received for that pseudonym. If this proof is not in the revocation list received, then it means that the client is not revoked and if he is not, he finishes his authentication sending the pseudonym and proof to the verifier, otherwise, he aborts immediately. The verifier, checks the authenticity of the proof using the pseudonym's public key and checks if the proof is in the revocation list, if it is not, the client is authenticated.

The property of *revocation auditability* is achieved because the client can see if is revoked before sending any information to the verifier. The perfect *backward unlinkability* is achieved because the verifiers never get to know the revoked pseudonyms, only their non revocation proofs. The verifier has the ability of associating a bunch of proofs to the same client, but the client will never send one of this proofs because he checks if his proofs are in the revocation list sent by the verifier before sending them. Even if the verifier sends an outdated revocation list to a revoked client, of a time where he was not revoked yet, the client will calculate the proofs based on the seed associated to this revocation list and they will be different of his proofs that are present in the updated revocation list. This is assured by always having a different seed for each revocation list. Of course, the verifier can't also forge revocation lists because their integrity and authenticity is protected by the digital

signature of the central authority.

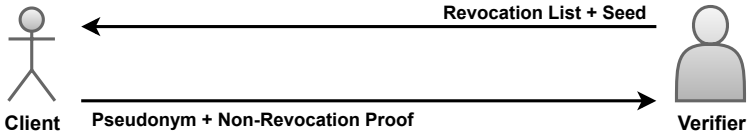

Figure 5: Authentication in Privacy Keeper.

This work has 2 disadvantages, which are the total amount of time and computation for the central authority to revoke a client, because it has to calculate the non revocation proofs for all the pseudonyms revoked until that moment using digital signatures, and the latency of clients' authentications, because of the need of sending the revocation list to the client in the beginning of each authentication. The authors present two possible optimizations: usage of bloom filters for revocation lists decreasing their size and pre-computed non revocation proofs for pseudonyms already revoked to be inserted in future revocation lists. Even if the bloom filters are used to implement revocation lists the problem of authentications' latency will still happen due to the size of bloom filters that are usually very big to reduce the amount of false positives.

4.3 Discussion

In this section, we summarize and discuss the main differences and accomplishments of existing pseudonym schemes presented in the previous sections. Table 1 summarizes the properties offered by each scheme and the costs of the respective operations.

Analysing Table 1 it is possible to notice that in all pseudonym schemes, the information required for storage in the pseudonym providers is dependent only on the number of clients registered. This is achieved by using pseudorandom functions [7,11], or by using a hash chain [6,8], to generate pseudonyms, using seeds constructed from a user identifier. The fact that the pseudonyms issued to a client can be generated from the seed associated to that client, removes the need to store all the pseudonyms issued, as the pseudonym provider can generate them again, when revoking a client, and distribute them to the verifiers.

All the schemes based on epochs and time slots [6–8], have the number of pseudonyms issued dependent on the number of time slots and on the pseudonyms used per time slot. This situation is not ideal, because it forces the user to store many more pseudonyms than he will ever need. Also, this is problematic because of the number of pseudonyms that need to be revoked when a given client is revoked, increasing the size of revocation lists and the size of the "blacklists" of revoked pseudonyms in the verifiers.

It is also possible to notice the impossibility of schemes that are based on the notion of epochs and time slots [6–8], to provide perfect *backward unlinkability*. EDGAR improves this, by making it possible to reduce the size of the time slots, at the expense of having much bigger revocation lists. The growth of the size of the revocation lists is mitigated by the fact that this scheme allows to provide the clients only with as many pseudonyms as the number of their authentications, and by the latchkeys tree, as explained in Section 4.1.2, but it still does not achieve *backward unlinkability* perfectly. Although IFAL is a scheme based in epochs and time slots, it achieves the property by presenting a radical approach of issuing certificates to clients and activate them later by sending activation codes to the clients, every time slot, for the pseudonyms of that time slot. This scheme revokes clients by stop sending them the activation codes and waiting until the next time slot, for the client to run out of activating pseudonyms, which leads to the disadvantage of letting misbehaving users keep authenticating in the system until the end of the revocation time slot. This work has the advantage of swapping the distribution of revocation to the verifiers by the distribution of smaller activation codes to the users. This advantage is only an advantage in systems where the number of clients is similar to the number of verifiers. This does not happen in smart retail applications, where the number of customers is much higher than the number of verifiers. Despite of also achieving perfect *backward unlinkability*, V-token has a similar problem, it can't revoke a client right away after a complaint, as it needs to let the revoked user run out of V-tokens.

Concerning the property of *revocation auditability*, it is observed that only Nymble, PEREA and Privacy Keeper provide it. Nymble and Privacy Keeper provide it by having the revocation list sent to the client in

the beginning of each authentication. This revocation list is signed by the trusted central authority in order to prove its freshness and authenticity. This type of approach has the disadvantage of having high latencies in each authentication, due to the need of sending the revocation list to the user. PEREA achieves this property also by sending the revocation list to the client in the beggining of each authentication, but as it uses accumulators, a constant-size construct, does not have the previous problem. It uses Zero-knowledge proofs, so the client does not need to know that the revocation list is authentic and updated.

Only PEREA and Privacy Keeper achieve perfect *backward unlinkability* and *revocation auditability* simultaneously. PEREA achieves this using Zero-knowledge proofs and the scheme is impractical for physical systems, where users' authentications cannot be limited, such as smart retail applications. In this type of applications, users usually authenticate using a smart card or their smartphones, that have limited computational power and are not suitable for generating resource intensive Zero-knowledge proofs. Privacy Keeper achieves the properties solely, by asymmetric encryption, but does not achieve them efficiently. It has the disadvantages of high latencies in authentications and the high quantity of computation needed for the central authority to revoke a client.

Pseudonym Scheme	Provider Storage	Pseudonyms Issued	Revocation List Size	*Backward Unlinkability*	*Revocation Auditability*
Haas *et al.* [6]	*O(N)*	*O(p*T)*	*O(R)*	Limited	No
EDGAR [7]	*O(N)*	*O(A)*	*O(R*A*log(T))*	Limited	No
Nymble [8]	*O(N)*	*O(p*T)*	*O(R)*	Limited	Yes
IFAL [9]	*O(N)*	*O(p*T)*	-	Full	No
V-token [1]	*O(N)*	*O(A)*	-	Full	No
PEREA [10]	-	*O(A)*	-	Full	Yes
Privacy Keeper [11]	*O(N)*	*O(A)*	*O(E*A)*	Full	Yes

Table 1. Comparison of the different schemes. (N) Number of clients registered. (p) Pseudonyms per time slot. (T) Number of time slots in an epoch. (R) Number of revoked clients. (A) Number of authentications performed by a client. (E) Clients revoked since the beginning of the epoch.

5 Architecture

We present a pseudonym scheme that can be used in smart retail applications in order to provide customers a higher level of privacy, preventing the store system from storing information about previous purchases. Our proposed scheme aims to provide perfect *backward unlinkability* and *revocation auditability* efficiently. Our scheme is composed of 3 main entities: Clients, System Verifiers and a Pseudonym Provider. We will consider a threat model based on previous works, contemplating malicious clients and malicious system verifiers. Malicious clients may try to generate false pseudonyms to access resources they are not authorized. It can also use revoked or expired credentials to authenticate in the system, or try to use another clients' pseudonyms. Malicious verifiers may try to link past users' authentications breaking users' privacy and anonymity, or try instead to send outdated revocation lists to clients trying to mislead a revoked client to authenticate in the system and gain valuable information about him.

We base our scheme on Privacy Keeper [11] and try to overcome its flaws by presenting three possible solutions. This work has the approach of authenticate the clients based of non revocation proofs for their pseudonyms, which are digital signatures created using the private key associated to a pseudonym and a seed. This seed is associated to a revocation list composed by non revocation proofs for revoked pseudonyms using the associated seed. These revocation lists are distributed by a central authority responsible for issuing and revoking pseudonyms and when a new client is revoked, all his pseudonyms are revoked and a new seed and revocation list is created. The central authority calculates the non revocation proofs for the pseudonyms of this client and for all the pseudonyms that had been revoked so far, using the new seed, and distributes the revocation list to the verifiers. As mentioned in the previous sections, this approach presents two disadvantages which are: the need to calculate the non revocation proofs for all the revoked pseudonyms when a new client is revoked, and the need to send the revocation list to the clients in each authentication imposing a high latency on each

authentication, with the severity of this issue proportional to the size of the revocation list, even if it uses bloom filters as revocation lists due to their normal big size.

We propose three possible solutions for these problems and suggest that revocation lists be implemented, based on bloom filters. The first one is the usage of redactable signatures for sending the revocation list to the client in the beginning of each authentication. This implies the usage of bloom filters to implement the revocation lists and takes advantage of the fact that the filter's hash functions can be known in advance by the clients and clients only need to check certain bits of the bloom filter. The client can first receive the seed without the revocation list, choose a pseudonym, generate the respective non revocation proof, pass the proof through the hash functions and then only request the filter's bits necessary to check his revocation status. Using redactable signatures the verifier has to send the bits requested and some values of the tree as explained in Section 2.2. This allows to send a lot less information to the client, reducing the latency, while keeping it possible for the client to check the revocation list's authenticity and integrity.

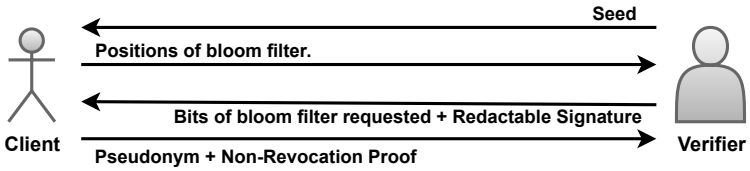

Figure 6: Authentication in our first proposed solution.

We present a second possible solution, that also tackles the problem of high latencies during authentications. This solution also implies the use of bloom filters as revocation lists. We propose the use of a set of bloom filters to store the non-revocation proofs of the revoked pseudonyms in the verifiers. This set would contain several bloom filters of different sizes. The non-revocation proofs would be inserted in all bloom filters and obviously, the filters with a smaller size would have a higher rate of false positives. In each authentication, the client would start by receiving the smallest filter bloom and would verify his revocation status. If he was not revoked, he would immediately authenticate in the system. On the other hand, if he was revoked it could be a false positive, so he asks for the next smaller bloom filter and repeats the process. This process is repeated until the client authenticates in the system or until the client receives the last bloom filter and is not able to authenticate. This solution takes advantage of the fact that false positives are rare and most clients would be able to authenticate in the first steps, so this solution would allow to send bloom filters to the clients smaller than the original one. This solution has the obvious problem that, in rare unlucky authentication cases, clients would be forced to receive multiple bloom filters achieving a worse performance than in the original case. Still, this solution would allow to increase performance in most of the cases. The efficiency of this solution would depend on the number of bloom filters used and in the size of them. We intend to study the influence of this parameters in our future work.

Our third solution mitigates the amount of computation necessary for the central authority to revoke a client. Whenever a client is revoked, a new revocation list is created and only the non revocation proofs for the new pseudonyms revoked are inserted on it. This list would be then distributed to the verifiers and in each authentication, the client would receive multiple revocation lists and would have to compute a non revocation proof for each list. Periodically, the revocation lists would be merged by the central authority. This would alleviate the computation necessary to revoke a client, at the expense of having the client receiving multiple lists and generating a proof for each. It is unfeasible to have a different list for each revoked client, but it is possible to have the time divided by slots and to have a different revocation list for each time slot. For example, we could use slots of a day, and in each revocation list would be non revocation proofs of pseudonyms of clients revoked during that day and whenever a client was revoked, a new list and a new seed would be created, and only the pseudonyms' proofs associated to clients revoked in that day would be inserted in that list. After some time slots, the lists could be merged by the central authority, creating a new revocation list and seed and generating all the non revocation proofs for all the revoked pseudonyms.

Combinations of these solutions are also be possible. For instance, it is possible to combine the first and third solutions or the second and third solutions. We intend to evaluate these combinations in our work.

6 Evaluation

For the evaluation, we aim to explore how our system performs when varying different internal parameters, similar to the evaluation presented in the EDGAR [7]. We plan to evaluate our scheme in a VANET scenario, despite our work being centered on a smart retail application, because a VANET scenario is one of the most demanding use cases for any authentication scheme. We will utilize a publicly available dataset from a taxi company operating in the city of Porto, which includes taxi trajectories. This dataset can be used for running simulations and obtaining concrete values, such as the time duration that a vehicle is connected to a tower to download the revocation list. In addition to these approaches, we plan to conduct experimental evaluation to compare different our proposed solution against the related work, such as EDGAR, Haas *et al.* and Privacy Keeper. We will use machines available at our research center to experimentally measure and compare the following values:

- **Revocation Duration.** Empirically measure the time it takes for a central authority to generate a new revocation list, from the moment a client is revoked until 1) the revocation list is published and 2) the revocation list reaches all verifiers. This is an important measure since a revocation will only take effect after this duration.
- **Authentication Latency.** Measure the time it takes for a client to authenticate towards a verifier. This includes the network communication to download the required revocation list and the generation/verification of digital signatures. The authentication latency is a critical value for any authentication scheme, therefore is vital that we achieve low latency for this procedure, this value is vital for the practicality for our scheme in the smart retail case.
- **Revocation List Size.** By varying different parameters, such as the number of valid users, revoked users, tolerable false positive rate, and epoch size, we will measure how the size of the revocation list based on Bloom Filters varies. We will also measure the time it takes to transfer these large lists over the network and discuss their practicality in our smart retail scenario.

7 Work Schedule

Future work is scheduled as follows:

- January 13 - March 29: Detailed design and implementation of the proposed architecture, including preliminary tests.
- March 30 - May 3: Perform the complete experimental evaluation of the results.
- May 4 - May 23: Write a paper describing the project.
- May 24 - June 15: Finish the writing of the dissertation.
- June 15: Deliver the MSc dissertation.

8 Conclusions

Smart retail has seen an enormous growth in recent years, with the increasingly popular use of technologies in retail, in particular digital technologies. While these technologies can be used to improve customer experience, they also enable smart retail providers to profile user behaviour, not preserving customer privacy. Pseudonyms can be a solution for this problem, allowing customers to authenticate in these smart retail systems using identities other than their real identities.

Several pseudonym schemes have been proposed for several applications, but most of them do not achieve all the properties that we consider relevant for smart retail applications, such as perfect *backward unlinkability* and *revocation auditability*. The systems that achieve them, do so at the expense of high latencies in the authentication procedure and high costs of computation when revoking a misbehaving client.

In this work, we discuss different avenues to implement a pseudonym scheme that provides perfect *backward unlinkability* and *revocation auditability*. In this context, we have identified 3 possible solutions to improve the efficiency of existing pseudonym schemes.

Bibliography

[1] F. Schaub, F. Kargl, Z. Ma, and M. Weber, "V-tokens for conditional pseudonymity in vanets," in *2010 IEEE Wireless Communication and Networking Conference*. IEEE, 2010, pp. 1–6.

[2] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Advances in Cryptology—CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18–22, 2002 Proceedings 22*. Springer, 2002, pp. 61–76.

[3] J. Li, N. Li, and R. Xue, "Universal accumulators with efficient nonmembership proofs," in *Applied Cryptography and Network Security: 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007. Proceedings 5*. Springer, 2007, pp. 253–269.

[4] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[5] R. Johnson, D. Molnar, D. Song, and D. Wagner, "Homomorphic signature schemes," in *Cryptographers' track at the RSA conference*. Springer, 2002, pp. 244–262.

[6] J. J. Haas, Y.-C. Hu, and K. P. Laberteaux, "Efficient certificate revocation list organization and distribution," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 3, pp. 595–604, 2011.

[7] C. Correia, M. Correia, and L. Rodrigues, "Using range-revocable pseudonyms to provide backward unlinkability in the edge," in *Proceedings of the ACM Conference on Computer and Communications Security. Copenhagen, Denmark*, 2023.

[8] P. P. Tsang, A. Kapadia, C. Cornelius, and S. W. Smith, "Nymble: Blocking misbehaving users in anonymizing networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 2, pp. 256–269, 2009.

[9] E. Verheul, C. Hicks, and F. D. Garcia, "Ifal: Issue first activate later certificates for v2x," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 279–293.

[10] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith, "Perea: Towards practical ttp-free revocation in anonymous authentication," in *Proceedings of the 15th ACM conference on Computer and communications security*, 2008, pp. 333–344.

[11] C. Correia, "Providing revocation auditability: Privacy keeper," private communication, 2023.