Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
Corso di Laurea Magistrale in Ingegneria delle Telecomunicazioni

Tesi di Laurea Magistrale

A key Management Scheme for Access Control to GNSS Services

Candidato:
Silvia Ceccato

Relatore:
Prof. Nicola Laurenti

Correlatori:
Dr. Christian Wullems
Dr. Gianluca Caparra

Anno Accademico 2015–2016

Abstract

When applied to GNSS scenarios, conditional access becomes a rather challenging problem. Traditional key management schemes were developed for significantly different environments, where requirements on bandwidth are looser and data transmission is faster. The need for new GNSS services has spread to a big variety of applications, from road tolling to precision farming, and soon satellite navigation data will be exploited for high accuracy and authentication services. This interest encourages the research on key management with the aim of devising a new scheme that is flexible enough to support all service concepts that might come up in the future. Since GNSS services are possibly directed to a wide range of customers, security is one of the several application related issues: the statelessness of receivers, poor scalability, the absence of an aiding channel and the influence of channel errors on decoding performance make GNSS a very peculiar scenario. In this scope, this thesis investigates the feasibility of broadcast encryption algorithms to cope with the dynamics of the system.

After discussing the possibilities of adapting existing techniques, a novel key management scheme called *RevHash* has been devised with particular emphasis on guaranteeing revocation capabilities to the system, in order for it to be robust against anomalies and attacks. The performance of the scheme has been evaluated, showing that it is particularly suitable for low bandwidth subscription based scenarios.

Contents

Chapter 1

Introduction and state of the art

GNSS provides position, navigation and time information that are used by hundreds of millions of people worldwide. More and more applications are beginning to ask for new GNSS services: *precise point positioning* (PPP) allows to gain few centimeter accuracy for restricted budget receivers, and it can be interesting for precision farming, hydrological survey and the gathering of geographic and spatial data ([1]). On the other hand, *authentication* (a priori or a posteriori) might be important when a higher level of security needs to be guaranteed, e.g., for road tolling, professional navigation and law enforcement. Galileo, the European GNSS system, can offer these innovative services, possibly through a fee-based mechanism, which could be implemented in the scope of the commercial service. This concept would allow to ensure a revenue to the system, providing additional capabilities to interested customers. Thus securing GNSS data has become a primary concern, and when cryptography is employed, the secrecy of keys becomes critical. Since GNSS provides data via a broadcast channel, an attacker who discovers a key might compromise a huge number of users or break the conditional access system, stealing accounts and putting revenue at risk.

Key management is the security service which allows to regulate the use of these keys and provides countermeasures in case of key compromise. The study of key management techniques allows to devise a suitable scheme for GNSS and gives the opportunity to tune the system, trading off security with communication overhead, storage requirements and scalability. Nevertheless, building a key management scheme in such a peculiar scenario is a complicated task, since the schemes proposed in literature often refer to a completely different environment. This thesis focuses on group key management for access control, which is an interesting topic especially in light of the new services which are to be offered by GNSS.

One service that requires key management for ensuring conditional access is broadcast encryption (BE), that is the secret, authenticated and integrity protected transmission of broadcast content to a restricted set of receivers, often called "privileged users". In investigating possible solutions, the following parameters are to be taken into account.

- **Communication overhead**, which is the (additional) transmission rate required by key management messages;
- **Storage Requirement**, which is the amount of memory at the receiver side required to store all the key material.
- **Computational complexity** at the receiver side, in order for the scheme to work and maintain its functionality.

Since bandwidth is a critical parameter in GNSS applications, the encryption scheme should rely on a *group key* K_{g}, which should be available to the intended receivers, only. Messages shall be encrypted by a session key K_{s}, which is in turn encrypted through K_{g} and broadcast. The encryption of a key through another key (KEK) is a common paradigm in key management schemes, and it is known as *key layering*. Key layering is common practice in broadcast television [2], where three or four layers of keys are used to protect the transmission; external layer keys are *long lived*, while lower layer keys manage different groups of users and subscription types, and the encoding of data streams, and have shorter *cryptoperiods*.

In addition, many are the functions a broadcast encryption scheme can provide. Some of them are fundamental, while other can be traded off depending on the specific application:

- **Group confidentiality**: only intended users can decipher the broadcast messages; This is a fundamental requirement for a broadcast encryption system.
- **Revocation capability**: The scheme shall allow the revocation of users. Revocation can be due to the ending of the subscription period or to a dangerous situation in which the system is exposed. The former case can be called *subscription expiration*, and it can be performed at preset times, while the latter will be referred to as *random revocation*, since it may be needed at any time.
- **Tracing capabilities**: observing the access operation performed by an illegitimate device, the system may have the capability of determining the identities of those authorized users which are responsible for the leakage of the group keys.
- **Collusion resistance**: Two or more users *outside of the privileged set* who collude should not be able to find out the new group keys. A scheme is defined k-resilient if no coalition of size k outside the privileged set can find out the group key.
- **Forward secrecy**: An attacker who compromises any subset of old group keys should not be able to obtain any information on new group keys;
- **Backward secrecy**: An attacker who compromises any subset of group keys should not be able to obtain any information on previous group keys;

- **Scalability**: The scheme shall allow for a dynamic environment in which users can join and leave the system. If the system doesn't scale well, it can be renewed every T_{renewal}.

A broadcast encryption system shall allow users to join and leave the privileged set. It should be noticed that an encryption scheme doesn't provide authentication capabilities. For this reason each encrypted message shall be authenticated, e.g., with an asymmetric key infrastructure.

The existing broadcast encryption schemes adopt different approaches:

- The receiver can be *stateful* or *stateless*. Stateless receivers can be offline and thus they won't always receive management messages. In order for a key management scheme to support stateless receivers, either the key management messages must be independent from previous ones or the old key management messages must be broadcast cyclically. On the other hand, stateful receivers are always assumed to receive the broadcast messages. For what concerns GNSS, receivers are in general assumed stateless.

- The scheme can exploit *symmetric* or *asymmetric* keys. In the first case the encryption key is the same as the decryption key. In the latter case a public key is used for encryption, and everybody will in principle be able to encrypt a message. This underlines the importance of authentication to protect from spoofing in GNSS applications.

1.1 Literature review for broadcast encryption

The first broadcast key management schemes have been proposed by Fiat and Naor [3]. They build three 1-resilient protocols with constant communication overhead and $\mathcal{O}(n)$, $\mathcal{O}(\log n)$ and $\mathcal{O}(1)$ storage requirement, each with different cryptographic assumptions. k-resilience is obtained by extending 1-resilient protocols through the use of hashing functions. This extension exploits a *combinatorial distribution* of keys which allows to exclude subsets of up to k users. The proposed k-resilient protocol has $\mathcal{O}\left(k^2 \log^2 k \log n\right)$ communication overhead and $\mathcal{O}(k \log k \log n)$ secret storage requirement at the receiver side with k the maximum number of revocations. The key management messages are independent from one another, meaning that the scheme can support stateless receivers. [3] specifies that the communication overhead is composed of two contributions: the *identification set*, in which the privileged set is specified ($\mathcal{O}(n)$ bits in the worst case), and the *broadcast encryption transmission* or *ciphertext*. In this document and in literature the minimization of the communication overhead is equivalent to the minimization of the ciphertext.

The problem common to many k-resilient schemes is that they allow for revocation of up to k users, and the performances often depend on k. When more than k users need to be revoked, the system will need rekeying.

In [4] unconditionally secure zero-message k-resilient schemes are analyzed, and a lower bound is found for the number of secret keys stored by the users, which is $\sum_{i=0}^{k} \binom{n-1}{i}$. Being the memory

requirement a critical parameter, many subsequent works have tried to allow for smaller storage at the cost of an increased number of management messages. This trade-off between memory and communication overhead is analyzed in [5], where the combinatorial approach is examined in the case when the privileged set has a fixed size. The authors of [5] argue that system based on the combinatorial approach have either prohibitive requirements for what concerns storage or a long communication overhead. [6] studies the possibility of relaxing confidentiality requirements, admitting a number of "free riders" inside the privileged set in order to improve the bounds. In [7] the authors propose three schemes which exhibit different trade-offs and allow for one-time revocation of up to k users with: $\mathcal{O}(k)$ overhead and $\mathcal{O}(k \log n)$ storage; $\mathcal{O}(k \log n)$ overhead and $\mathcal{O}(k \log n)$ storage; $\mathcal{O}\left(k^2\right)$ overhead and $\mathcal{O}(k \log n)$ storage.

A different kind of approach is used by Naor and Pinkas in [8], where asymmetric encryption is used, combined with threshold secret sharing. Their asymmetric key scheme exploits Shamir's polynomial based secret sharing in order to revoke up to k users with a communication overhead of $\mathcal{O}(k)$. This systems also has tracing capabilities: given a pirate device and a subset of suspected users, a confirmation test can be run in order to establish whether any traitor belongs to that subset.

Another (symmetric key) approach exploits *logic key hierarchy* (LKH): [9] and [10] independently designed schemes based on this concept. User are organized as the leaves of a tree, and they are given a key for each node that lies on the path from the root to the user's leaf. When a user leaves the system, those keys must be changed, and this will affect some of the other users. In order to recover confidentiality, approximately $\mathcal{O}(d \log n)$ encrypted keys must be encrypted and broadcast, for a full d-ary tree. However, this approach supports stateful receivers only. In order to support stateless receivers, each management message should be rebroadcast cyclically. Various application of LKH schemes on pay-TV systems have been studied ([11], [12]).

The most interesting stateless version of this approach, named "subset difference", is described in [13]. The scheme organizes the revoked users into a *Steiner tree*, and groups the leaves into subsets $S_{i,j}$. Subset $S_{i,j}$ contains the leaves of the subtree of root v_i minus the leaves of the subtree of root v_j, where v_i is an ancestor of v_j. The leaves corresponding to the privileged set can be efficiently covered by these subsets. Revocation requires the broadcast of $2r$ messages ($\mathcal{O}(r)$ communication overhead), where r is the number of revoked users, and a storage requirement of $\mathcal{O}\left(\log^2 n\right)$. The keys along the tree are generated starting from a seed (the root) and iteratively applying a hash function to derive the keys of the children. This scheme can revoke up to n users ($k = n$), and it can use symmetric as well as asymmetric keys, with an increase in overhead for the latter case.

The authors of [14] provide an interesting bound for broadcast encryption schemes: they focus on symmetric encryption protocols and show that in order to guarantee collusion resistance, the number of messages needed for rekeying are $\Omega(n)$.

Since in GNSS applications the most critical resource is often bandwidth, there are two ways of minimizing communication overhead: relax the requirement on collusion resistance or work with asymmetric cryptography.

The former approach is followed in [15] and [16]. The authors of both papers propose a 1-resilient scheme which allows to achieve better overhead performances. [16] exploits multiplicative hashing functions and a linear organization of receivers in order to revoke one user at the cost of $\mathcal{O}(1)$ messages. With a slight modification of the protocol, however, batch revocation can also be allowed with the same overhead. With a careful adaptation of this scheme to subscription based scenarios, subscription expiration can be implemented with constant communication overhead and storage requirements, at the cost of $\mathcal{O}(n)$ computational complexity. With this protocol random revocation requires the broadcast of $2r$ messages.

[15] and [17] start from 1-resilient protocols and combine them with collusion resistant schemes in order to trade off overhead for security.

Asymmetric key schemes use a public key for encryption. In principle everyone could encrypt using that key and for this reason message authentication plays a fundamental role in these schemes. Boneh, Gentry and Waters present in [18] a public key broadcast encryption system based on the bilinear Diffie Hellman exponent assumption (BDHE) and bilinear maps. Their scheme achieves $\mathcal{O}(1)$ communication overhead and private key size, and has a public key of size $\mathcal{O}(n)$. The interesting point of this scheme is that it provides support for stateless receivers thanks to independent revocation messages: the last management message is enough for every receiver to recover the group key. Even though the public key is linear in n, which could lead to prohibitive requirements for the memory of a smart card, it is *public*, and users could easily store this information in non-secure memory of less than 1Gb.

This encryption system provides collusion resistance and chosen ciphertext security against *static* adversaries. Chosen ciphertext security against *adaptive* attackers has been achieved by [19] at the cost of $\mathcal{O}(n)$ ciphertext, public and private key size.

Another public key broadcast encryption system based on the same complexity assumptions as [18] is presented in [20], which has worse performances for the private key size ($\mathcal{O}(n)$).

For GNSS related applications a different kind of approach has been recently proposed by Curran in [21], where a number of hash chains at different hierarchies allow to have different user categories. The author suggests that trusted users shall be provided with intermediate values in the hash chains that will allow them to compute a certain number of future group keys. When adapted to a subscription based scheme in a temporal fashion, this approach is very efficient for what concerns bandwidth, since no message is needed in order to periodically change the group key: indeed each user would be given the seed corresponding to the expiration time of his subscription. The storage requirements are $\mathcal{O}(1)$. With this implementation the scheme handles subscription expiration, but random revocation is not possible, and the system is unprotected in case of key compromise.

Many schemes have been studied in literature, and some could be adapted to the challenging GNSS scenario. As this review has pointed out, communication overhead, storage requirements, security and statelessness of receivers are linked in a trade-off. For what concerns random revocation, the set identification problem is critical with such a low bandwidth: the overhead necessary to identify the

revoked receivers could be bigger than the management messages, and for high values of r it is almost equal to n bits (fewer if compression is performed).

Another issue is represented by user *join* operations: changing the group key anytime a new user joins the system is impractical and costly. A solution might be building a system with high capacity, accounting for future users who might join. Some of the schemes can allow to recover the "slots" left empty by revoked users ([16]), while others are progressively consumed, and need periodic rekeying operations to compensate their poor scalability.

In light of the above considerations, the mentioned revocation systems should be tested in order to find a feasible trade off between the various parameters, accounting for the problems of stateless receiver and low scalability. An interesting solution could be built through the combination of revocation systems with constant communication overhead ([18], [16]) with Curran's proposal in order to obtain some revocation capabilities and the ability to support autonomous users.

1.2 Environment related considerations

1.2.1 Scalability issues

Scalability is a major issue for revocation schemes, since almost all the schemes proposed in literature exploit some kind of structure (e.g., trees, linear structures, ...), or require to set the number of users as a parameter. In order for new users to join the system, there must be some "free slots" for them in the structure. A straightforward solution is to over-dimension the system setting n to be high enough to cover for new joining users, but a higher n impacts the performances of the system either for storage requirements or communication overhead. Since most subscription systems are dynamic, many users will eventually leave the system. However, the key material of those users can't be given to newcomers as it is, otherwise forward secrecy will not be ensured. For this reason there are two ways of dealing with new join events:

- Over-dimension the system in order to allow for a certain number of new join operation. The system space will be progressively consumed as new users enter the system, since the key material of former customers can't be reused;

- Once a revocation is performed, update the key material of all users, for example by encrypting update parameters with the new group key. Slots left empty by former customers can be recovered, since the corresponding keys are updated before they are given to new users. The system still needs to be over-dimensioned, fixing the maximum capacity, but the possibility of reusing empty slots allows for a much better scalability.

The latter solution is questionable: broadcasting an update value (also referred as seed) encrypted with the new group key can expose the system. If a user who belongs to the privileged set leaks the

rekeying seed, then former clients will be able to update their keys and decode all future communication. Note that the seed is part of the secret material that should not be accessible to the users. It is assumed that this material is kept and dealt with inside a tamper resistant device. If a user manages to break such a device, it will gain access not only to the seed, but also the group key, potentially distributing it to anyone. For this reason the second solution will be considered *secure*: no encryption protocol can guarantee confidentiality if users have access to their secret information. In the light of these consideration, the latter approach provides a better scalability and should be preferred.

1.2.2 Advantages of batch revocation

A broadcast encryption schemes deals with three types of events:

- *Join event*: when a new user enters the system. Key material shall be distributed to this new member through a secure unicast channel maintained by a central station. For GNSS applications backward secrecy is not necessary since old messages contain out of date information, and therefore join events won't require a rekeying operation;
- *Leave event due to subscription expiration*: when a user leaves the system because his subscription period is over. In order to ensure forward secrecy the group key shall be updated;
- *Revocation event*: when a user is evicted from the system before the expiration of his subscription for any reason (e.g., improper behavior).

Random revocation events might happen at any time and can't be foreseen; on the contrary, there are two ways to deal with leave events: either the system allows users to end their subscription at any time or it divides the time axis into slots of fixed duration and sell the service for multiples of this time slot. The latter case allows to deal with multiple leave events at a time, requiring less rekeying operations. When encryption is used to regulate the access to a service through a subscription system, it is reasonable to assume that clients will buy the service for a predefined amount of time. Therefore it is reasonable to assume the system knows in advance how long each user will stay. This can be a great advantage, since it allows to further differentiate between leave events and revocation events. **Leave events** are programmed and due to the expiration of the subscription of users, while **revocation events** are random and happen when the system faces unexpected circumstances. The idea of dealing with these two events separately has been proposed in articles [22] and [23] in the scope of multicast encryption (for stateless receivers). The first proposes a scheme which only takes care of leave events, without the need for any management message, as in [21]. The latter integrates [22] with an LKH based revocation system, which allows random revocation with $\mathcal{O}(d \log n)$. The structures that allow to implement such an efficient scheme for leave events are hash chains and binary hash trees. Both are based on the iteration of one way functions in order to obtain multiple keys starting from a single value.

1.3 Consideration on security evaluation of key management schemes

1.3.1 Models for security proofs

As discussed in [24], there are mainly two models to represent a key management scheme and analyze its security. The most widely used in literature is the **symbolic model**, which allows to simplify the problem and to obtain straightforward proofs of security. The symbolic model considers every cryptographic primitive as an ideal function. As an example, when a symmetric key encryption operation is considered, i.e., message M is encrypted with key K, in the symbolic model an attacker who retrieves $E_K(M)$ and does not know the key has no way to obtain any information about the original message M. In reality the situation is not so ideal, and many encryption primitives which were assumed to be secure have been weakened by cryptanalytic attacks. One-way hash chains are a paradigm of how symbolic models can be too optimistic in certain cases. The **random oracle model** is a symbolic model used for security evaluation of many encryption and key management schemes. This model considers hash functions as a black box which, given an input, returns an output chosen uniformly at random inside the output domain, and every unique input is always mapped to the same output. Having a hash function that mimics this behaviour in a way that is not distinguishable by attackers is not trivial, and many hash function have been proven not to be random oracles (e.g., all the hash function based on the Merkle-Damgård construction, like SHA-1, SHA-256 and MD5, are susceptible to the length extension attack, due to the block-padding mechanism). For this reason the symbolic model might not be accurate when used to evaluate the security of a scheme. Even in the ideal random oracle model, as explained in [25], hash chains still suffer from the effect of collision accumulation along the chain: the longer the chain, the higher the loss of entropy of the progressively generated hash values. Collision probability increases linearly with the length of the chain, but it decreases exponentially with the length of the output of the hash function. By increasing this length it is possible to compensate for the loss of entropy that derives from long hash chains.

The **computational model** is more realistic, as it assumes that the adversary is only limited by his computational resources, and it does not pose any limitations on the attacks to the system as long as they are time-efficient. Cryptographic primitives are not considered ideal, but their security is evaluated with respect to the computational feasibility of attacks against them. Nevertheless when it comes to security proofs, this model is often very complicated, especially when applied to dynamic contexts in which multiple users are present and the users set changes. Most of the protocols in literature provide proofs with respect to the symbolic model.

1.3.2 Collusion resistance

Collusion resistance is one of the most important features a broadcast encryption scheme can offer, and this paragraph will provide a definition of this property in the symbolic model. Let's call K_G^t the group key at time t, S^t the set of active users in the system at time t, and $\overrightarrow{S}^t$ the ordered sequence

$[S^0, S^1, \ldots, S^t]$. $B^t_{\overrightarrow{S}^t}$ represents all the broadcast key management messages transmitted by the system to the intended user sets up to time t, while $K(i)$ refers to all the keying material owned by user i. Let's take from [24] the notation $\text{Rec}\{\mathcal{M}\}$ which indicates the amount of protocol information (messages, keys, seeds...) that can be recovered starting from the information contained in set $\mathcal{M}$. A group-key distribution protocol is defined **resistant against single-user attacks** (i.e., 1-resilient) if:

$$\forall t > 0, \forall \overrightarrow{S}^t, \forall i \in S^t : \qquad K^t_G \notin \text{Rec}\{K(i) \cup B^t_{\overrightarrow{S}^t}\} \tag{1.1}$$

For most applications resistance against single user attacks does not guarantee a sufficient security level. Ideally a group key management scheme should resist attacks from collusions of any size. A group-key distribution protocol is defined collusion resistant if:

$$\forall t > 0, \forall \overrightarrow{S}^t : \qquad K^t_G \notin \text{Rec}\{\{\bigcup_{i \in \bar{S}^t} K(i)\} \cup B^t_{\overrightarrow{S}^t}\} \tag{1.2}$$

where $\bar{S}^t$ contains all former users who were once part of the system. This definition accounts for the possibility that an attacker manages to corrupt some devices belonging to former receivers and obtains their secret keys. [24] argues that for most group-key management schemes a "stronger" definition of security must be provided, accounting for the case in which an attacker manages to break *past group keys*. In GNSS application this concept, which goes under the name of *backward secrecy* is not always important, since, unlike in pay-TV systems, information loses its value with time: users are unlikely interested in authenticating past messages or receiving out to date positioning data.

1.3.3 Security and assumptions of a BE scheme

As for any encryption scheme, in broadcast encryption the existence of a tamper resistant device is fundamental. Without such a device the keys would be easily leaked, and conditional access would be impossible to provide. This device is supposed to contain all the key material, which will never be revealed to the users in the clear. What is accessible to the user is the input (i.e., the broadcast transmission) and the output (i.e., the decoded information) of the tamper resistant device.

As previously formalised, collusion resistance is a property of an encryption scheme which states that users who don't belong to the privileged set can't get any information about the group key even if they collude together. However, in literature the word "colluding" implies accessing the secret key material contained in tamper resistant devices, which is in contrast with the above assumption. Moreover, random revocation of users is sometimes justified in literature with the possibility of revoking clients who leaked their keys. This idea goes even further than collusion resistance, since it allows the possibility that authorized users break their tamper resistant device. Two are the main approaches:

1. Assume the existence of a tamper resistant device. Such a device can never be compromised, and the concept of collusion resistance becomes unnecessary for a key management scheme implemented in such a device: no user will be able to access his secret information.

2. Assume the existence of a tamper resistant device. Since perfect tamper resistance is a strong assumption, assume that this device can be broken with non-zero probability. If this is the case, two are the potential events that can endanger the system:

   - Some of the *revoked users* manage to break the tamper resistant device and get access to their secret information, possibly using it to collude together. Here is where the concept of "collusion resistance" becomes useful for a scheme. A k-resistant scheme is capable of resisting collusion of up to k former users.
   - Some of the *privileged users* (current users of the system) manage to break the tamper resistant device and get access to their secret information. If this is the case, there is nothing any broadcast encryption scheme could do to prevent those users from leaking their information and break the system. The only countermeasure that could save the system is the revocation of these users, followed by a change of the group key, provided that the system has a mechanism to discover the identities of misbehaving users.

   Ensuring that a device will never be tampered with is very hard, since reality proves that everything could in principle be broken with enough time and resources. Sometimes it is useful to account for the small yet not negligible probability that somebody manages to break the device. This event is hard to detect and finding out the traitor's identity is even harder. Still it might be useful to ensure that once the traitor is revoked, the system can be fully restored, and it will not be permanently damaged.

Since it is not in the scope of this work to decide which assumption is best suited for GNSS applications, in the following sections different approaches will be studied, leaving the choice to protocol designers.

1.4 The problem of identifying revoked users

Random revocation faces issues in GNSS scenarios that haven't been taken into account in literature. This happens because GNSS bandwidth is dramatically lower than in other analogous settings (e.g., broadcast television).

When the system needs to revoke some users that have failed to pay their subscription fee or have violated the terms of the service, the group key will need to be updated. This requires the other users to know how to update the key: the identities of the revoked users need to be revealed to all the privileged users. This is reasonable from a logical point of view: if the group key could just be independent of the revoked users, then how would the system hide it from them? The group key *must*

rely on information that only revoked user don't possess, and for this reason part of the information communicated via broadcast shall aim at identifying the set of revoked users. This problem will be referred as "set identification problem" from now on.

This is an important issue when bandwidth is as low as a few tens of bits per second, and receivers are stateless: rekeying information has to be broadcast cyclically in order to support autonomous users. Each client in the system needs to be given an identification number, and a linear structure will be forced into the scheme. The trivial way to deal with this problem is to send all the k identities of the revoked users via broadcast; this approach takes $k \log_2 n$ bits.

Another option is to send n bits, where a 1 at position i indicates that user i has been revoked. This is impractical for big systems with many users, but the identification string can be compressed through source coding.

1.4.1 Source coding paradigms for set identification

A first option is to code runs of zeros, since for random revocation purposes the number of zeros will be much higher than the number of ones. Various code can be used to code run lengths, for example Elias gamma coding. This algorithm codes each length l concatenating as many zeros as $\lfloor \log_2 l \rfloor = N$, followed by the $N+1$ binary representation of the number. The representation of each run length l requires $2\lfloor \log_2 l \rfloor + 1 \sim 2 \log_2 l$. With this paradigm the number of bits needed to code the string strongly depends on the disposition of the 1 bits inside the string.

Let's consider the worst case: every chunk of 0 bits in the string can be coded with $2 \log_2 d_i$ bits, with d_i the length of the run. The sum of all d_i is equal to $n-k$.
$\sum_{i=1}^{k} 2 \log_2 d_i$ under the constraint $\sum_{i=1}^{k} d_i = n-k$ is maximum when all d_is are equal, as shown in Fig. 1.1. This happens when the identities of the revoked users are equally spaced inside the n-bit vector. In this case the amount of bits needed for coding is approximately $(k+1)2 \log_2 \frac{(n-k)}{k+1}$. This result is better than the one achieved by sending the identities of revoked users.

Another option exploits the opportunity of performing revocation of a predefined number of users. This allows users to know in advance how many ones will be present inside the n-bit string. The entropy of a source producing vectors $\mathbf{v}$ of n bits, k of which are ones, is:

$$H(\mathbf{v}) = \log_2 \binom{n}{k} = \log_2 \frac{n!}{k!(n-k)!}$$

The entropy can be reached by transmitting an index of $H(\mathbf{v})$ bits referring to a specific vector $\mathbf{v}$. Arithmetic coding is another option (setting $p(0) = \frac{k}{n}$), which allows to achieve a number of bits very close to the entropy, allowing to revoke a number of users close to the predetermined one without a big loss of performances. Fig. 1.2 shows the performances of the various compression schemes for different values of k and n.

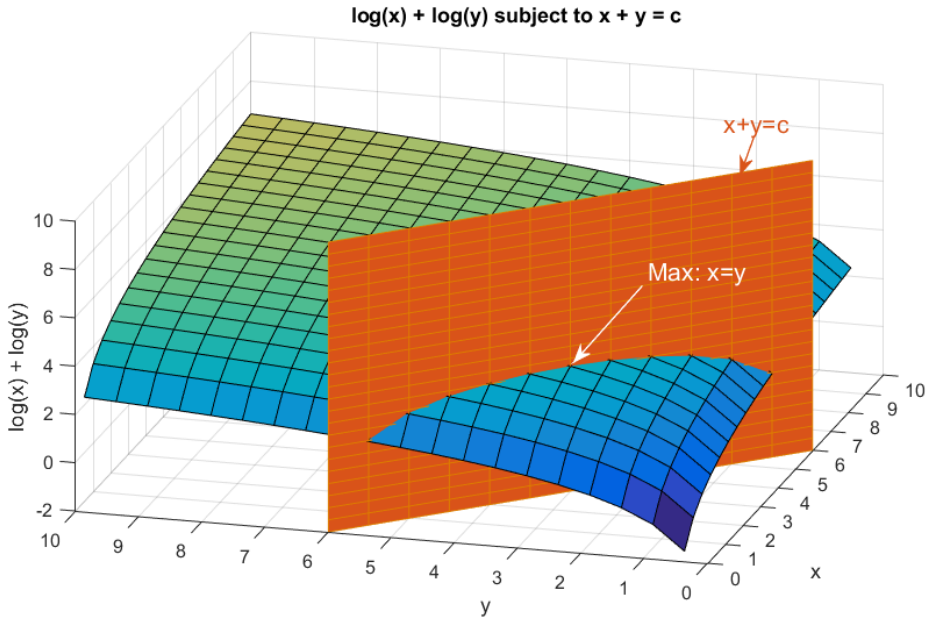


Figure 1.1: The sum of $\log x + \log y$ subject to $x + y = c$ is maximized for $x = y$.

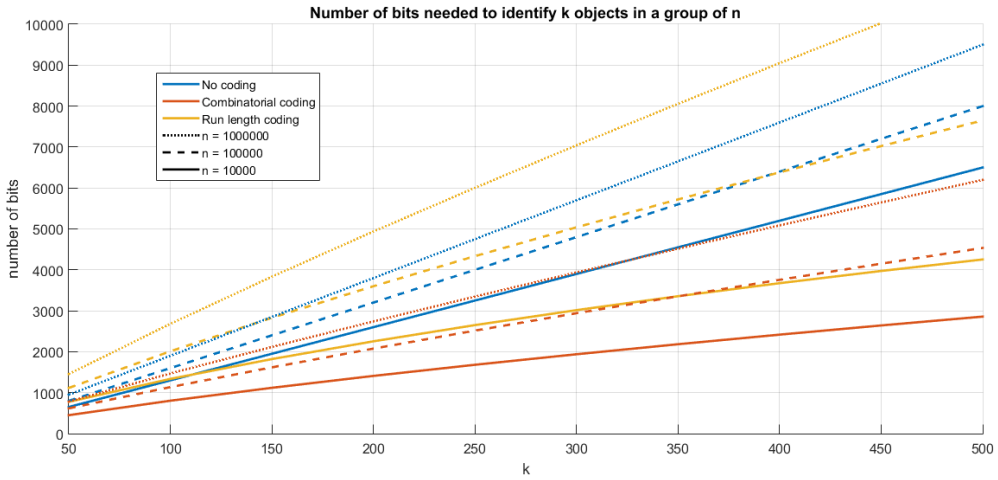


Figure 1.2: Number of bits for identification set with three different coding methods.

1.5 Comparison between various schemes

Since the most critical resource in GNSS transmissions is bandwidth, the following analysis focuses on the reduction of communication overhead. All the schemes will be discussed in their stateless version, since this is the classic setting in GNSS scenarios.

1.6 Naor and Pinkas' scheme

Naor and Pinkas propose three variants of a broadcast trace and revoke system in [8]. Their schemes are based on threshold secret sharing: given a polynomial P of degree k over a field $\mathcal{F}$, the secret to be shared is $\mathcal{S} = P(0)$. Given $k+1$ shares $P(i)$ of the secret it is easy to interpolate the polynomial and obtain $\mathcal{S}$, while k shares don't reveal any information about it. Exploiting this idea, the basic scheme requires to give to each user a secret key composed of an identifier $I_u \in \mathcal{F}$, with $|\mathcal{F}| \geq 2^{80}$ (for 80 bits of security) and a share of the secret, $P(I_u)$. In order to revoke k users the system needs to broadcast their identities and shares. This system, though, allows for a one time revocation which is secure against a coalition of all revoked users, while the ideal revocation scheme for the purposes of this thesis should be able to perform an unlimited number of revocations without the need of frequent rekeying.

For this purpose an extension of this basic scheme is proposed, which exploits the *decisional Diffie Hellman* assumption (DDH): given a cyclic group $\mathcal{G}$, a generator g and a, b and c chosen randomly in $[1, |\mathcal{G}|]$, there is no efficient algorithm that allows to distinguish between $(g^a, g^b, g^a b)$ and (g^a, g^b, g^c).

Setup

The scheme operates in a cyclic group of prime order Z_q, which is a subgroup of Z_p^*, where p is prime and q divides $p-1$. In the setup the group controller selects a polynomial P of degree k over Z_q, with generator g. A secret key, $K_u = (I_u, P(I_u))$ is distributed to each user u through a private and secure channel.

Revocation

When the system needs to revoke r users, the group controller randomly chooses $s \in Z_q$, and $g^{P(0)}$ becomes the new group key. The central station will then broadcast the message:

$$(g^s, I_{u_1}, g^{sP(I_{u_1})}), ..., I_{u_k}, g^{sP(I_{u_k})}) \tag{1.3}$$

Every legitimate user u_i can compute $(g^s)^{P(I_{u_i})}$ and combine it with the broadcast message in order to obtain the new secret key, $g^{sP(0)}$, thanks to the Lagrangian interpolation formula:

$$P(0) = \sum_{i=0}^{k} \lambda_i P(x_i) \tag{1.4}$$

where $(x_0, \ldots, x_k)$ are the identities over which the shares are computed and λ_i are the Lagrangian coefficients, which can be expressed as:

$$\lambda_i = \prod_{i \neq j} \frac{x_i}{x_j - x_i} \tag{1.5}$$

$$\lambda_i = \prod_{i \neq j} \frac{x_i}{x_j - x_i} \tag{1.6}$$

Then the shared key can be computed as:

$$g^{sP(0)} = g^{s \sum_{i=0}^{k} \lambda_i P(x_i)} = \prod_{i=0}^{t} g^{s\lambda_i P(x_i)} \tag{1.7}$$

With this revocation mechanism the system can resist a coalition of at most k revoked users. A variant is proposed where each revocation message is encrypted with the current group key in order to make them accessible only to users which are currently inside the system. While the first revocation mechanism allows to revoke up to k users before rekeying, the second mechanism allows to revoke a bigger amount of users as long as at most k users are revoked per round. The second mechanism can resist coalitions of up to k users, so there is a trade off between the number of revoked users and the security of the scheme.

The proof of security in the symbolic model can be found in [8] and it is based on the decisional Diffie Hellman assumption (the probability of breaking DDH assumption is the same as the probability of breaking the revocation scheme)

1.6.1 Performances

This scheme performs revocation by broadcasting secrets belonging to the users it wishes to revoke, so the **communication overhead** is $\mathcal{O}(t)$ with t the number of revoked users; the requirements for the **secret storage** are just $\mathcal{O}(1)$. The scheme doesn't provide a smart mechanism to reduce the overhead of subscription expiration with respect to random revocation: for this reason it is not suitable for low bandwidth scenarios. The scheme is **not easily scalable**, especially for high-dynamics applications: even though potentially the scheme could allow for large numbers of users, once they leave the system their shares can't be recovered, nor their identities reused without requiring additional communication via an aiding channel with all the privileged users.

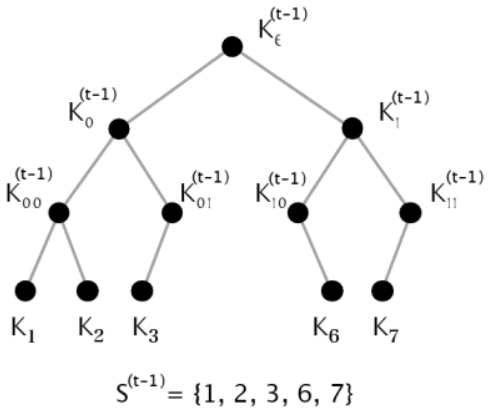
1.7 Logic key hierarchy schemes

As mentioned in the previous chapter, LKH is a data structure which is widely used for subscription systems like digital television. The users of the service are organized as the leaves of a binary tree, where the internal nodes correspond to symmetric keys. Each user is provided with all the keys in the path from its leaf, corresponding to the user's personal key, to the root, which corresponds to the current group key. LKH revocation scheme was born as a protocol for stateful receivers, where the group key is changed each time a user leaves the system. If this is the case, revocation is performed as shown in Fig. 1.3: when user 1 leaves the system, all the keys in the path from leaf 1 to the root must be changed. In order to do so, every user whose path to the root crosses user 1's path must receive some key management updates. Each of the new keys can be encrypted with other symmetric keys in the tree structure in order to minimize the broadcast transmission. This mechanism requires at most $2\lceil \log_2(n) \rceil$ encrypted messages for the revocation of one user. In the improved version of the protocol, as explained in [24], the new keys can be selected in a one-way chain fashion with the help of two distinct one way functions (or pseudo-random number generators), as shown in Fig. 1.3(c). This version allows to further reduce the communication overhead to $\lceil \log_2(n) \rceil$.

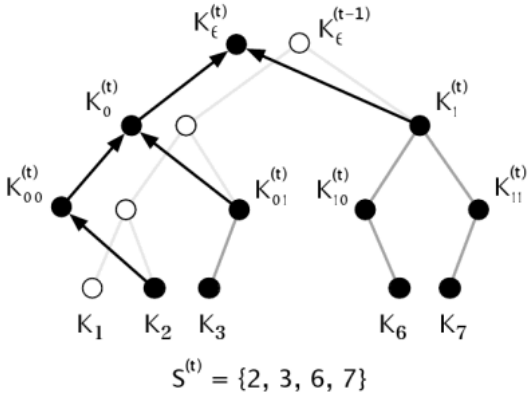
The protocol can be modified to use a d-ary tree instead of a binary one, with overhead of respectively $d\lceil \log_2(n) \rceil$ and $(d-1)\lceil \log_2(n) \rceil$ for the basic and improved version a single user revocation.

For what concerns security, both protocols (basic and improved version) are collusion resistant, and the proof, both in the symbolic and computational model (against non-adaptive adversary), can be found in [24].

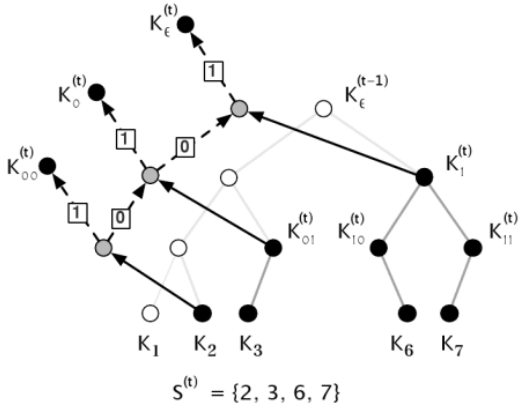
Scalability is not a problem in this scheme, since every time a user leaves the keys are changed and can be given to new users, but this comes at the price of a high **bandwidth consumption** in the stateless scenario. **Storage requirements** are logarithmic in the number of users: each device will have to store only $\lceil \log_2(n) \rceil$ keys: one for each level of the binary tree ($\lceil \log_d(n) \rceil$ for d-ary trees), which is reasonable for most applications: nowadays the average smart card has EEPROM size of 8 to 128 Kbit (in smart card terminology 1Kbit means one thousand bits); if a key has size of approximately 256 bits, for 1 million users even less than 6 Kbit would be enough [26]. In the most common applications of this revocation scheme, the bandwidth constraints are not as tight as in the environment considered in this thesis. In satellite communication it is indeed impractical to perform a group-key change every time a user enters or leaves the system, since this would require a huge amount of bandwidth. All the key management messages inducing a group-key update must indeed be available to autonomous users (users who can not count on a permanent aiding channel) via broadcast: if the protocol is used as it is, the bandwidth consumption due to the regular dynamics of the system will rapidly increase and become prohibitive.



(a) Before revocation of user 1



(b) After revocation of user 1 (original LKH version)



(c) After revocation of user 1 (improved LKH)

Figure 1.3: LKH revocation system

1.7.1 Subset cover protocols

Two stateless versions of the LKH revocation scheme have been proposed in [27] by Naor *et al.*, which are more suitable not only for GNSS scenarios, but also for all those applications in which users must be autonomous, and decode messages thanks to their initial state and the current broadcast transmission, only (e.g., copyright protection).

The general idea of all subset cover protocols consists in finding a collection of subsets $\mathbf{C} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_w\}$, such that $\mathcal{S}_j \subseteq \mathcal{N}$, with $\mathcal{N}$ the set of all users. Each subset is assigned a long-lived key K_j and every user inside $\mathcal{S}_j$ is provided with the corresponding key. This framework allows to revoke more than one user at a time, which is convenient in low bandwidth scenarios. Once the revocation set $\mathcal{R}$ (a subset of users that must be removed for whatever reason) has been selected, the system has to find a partition of the remaining users ($\mathcal{N} \setminus \mathcal{R}$), $\{\mathcal{S}_{i_1}, \mathcal{S}_{i_2}, \ldots, \mathcal{S}_{i_m}\}$, in such a way that $\mathcal{N} \setminus \mathcal{R} = \bigcup_{j=1}^{m} \mathcal{S}_{i_j}$. A session key (or group key) K is then encrypted m times with each of the m long-lived keys associated to the subsets.

A specific implementation of this revocation scheme must be specified by

- An algorithm which allows to compute the collection of cover sets;
- A key assignment for the subsets;
- A mechanism to cover all the non-revoked users with subsets in the collection;
- A method to allow each user to find his own cover.

Both the protocols proposed in [27] are implemented by assigning each receiver to a leaf of a binary tree structure, which will have $2n - 1$ nodes in total. The first protocol is called **complete subtree**, and it works as follows:

- The collection of covering subsets consists of all complete subtrees rooted at any node v_i of the binary tree ($2n - 1$ complete subtrees).
- A key K_{v_i} will be associated independently to each subtree, and it will be given to all the users corresponding to the leaves of the subtree. Each user will be provided with $\log n + 1$ keys, which are all the keys on the path from the user leaf to the root (similarly to the key distribution of LKH protocols).
- Let's call $\mathcal{ST}(\mathcal{R})$ the *Steiner tree* induced by the set of revoked users, which is the minimal subtree that connects all the revoked leaves. In order to cover the set $\mathcal{R}$, it is sufficient to take the collection of all complete subtrees $\mathcal{S}_{v_i}$ whose root is adjacent to nodes of outdegree 1 in $\mathcal{ST}(\mathcal{R})$. It can be proved that the average size of the cover is at most $m = r \log(\frac{n}{r})$.
- The final message consists in all the identifiers of the root nodes $\{i_{v_1}, \ldots, i_{v_m}\}$ plus the encryption of the session key under m different keys, $\{E_{v_1}(K), \ldots, E_{v_m}(K)\}$.

A variant of this protocol that is even more communication-efficient is called **subset difference**. This alternative implementation of the subset cover revocation scheme allows to further reduce the number of covering sets to at most $2r - 1$ at the cost of some additional memory. The users are atill represented with a binary tree, and the protocol proceeds as follows:

- Covering subsets have a different expression: they are defined as the difference between two sets. A user u belongs to subset $\mathcal{S}_{i,j}$ if it is a leaf of the subtree rooted at v_i but not of the one rooted at v_j: this means that v_i is an ancestor of leaf u, while v_j is not (see Fig. 1.4)

- A key $K_{i,j}$ will be associated independently to each $\mathcal{S}_{i,j}$, and it will be given to all the users belonging to that subset. With this variant, each user belongs to a higher number of subsets with respect to the complete subtree scheme.

- The cover can be found starting from $\mathcal{ST}(\mathcal{R})$: one subset will be added to the cover for each maximal chain of nodes of out-degree 1 in the Steiner tree. Such a chain has the form $< v_1, \ldots, v_l >$, where $< v_1 >$ is either the root or a node of out-degree 2, and $< v_l >$ is either a leaf or a node of out-degree 2. For each chain of this form a subset $\mathcal{S}_{1,l}$ is added to the cover. The authors of [27] state that the maximum number of subsets needed to cover $\mathcal{N} \setminus \mathcal{R}$ is $m = 2r - 1$.

- The suggested key assignment to users exploits an idea similar to the one used by Fiat and Naor in [3]: first each non-leaf node is assigned a random seed S_i. Let's consider a subtree T_i rooted at node v_i: for each internal node v_j a label $\text{LAB}^{(i,j)}$ can be derived from the seed of the root, S_i, thanks to a pseudo-random number generator G. The output of G is three times the size of the input: $G(\text{Label}) = G_L(\text{Label})|G_M(\text{Label})|G_R(\text{Label})$. Starting from the root of the subtree (labelled $\text{LAB}^{i,i} = \text{S}^i$), for every node v_j the label of a left child is $G_L(\text{LAB}^{i,j})$, and the label of a right child is $G_R(\text{LAB}^{i,j})$; the key $L^{(i,j)}$ associated to $\text{LAB}^{(i,j)}$ is $G_M(\text{LAB}^{(i,j)})$ (see Fig. 1.5).

  The keys are distributed as follows: user u which lies in subtree T_i muist be able to compute key $L^{(i,j)}$ if and only if v_j is *not* one of its ancestors (this approach is opposite to LKH key distribution method). How many seeds does a user need to store? Let's consider the path from the user to the root of T_i. The nodes which hang off this path (and therefore are not ancestors of the user leaf) are ancestors of all the other leaves of the subtree, and they correspond to the seeds the user should own in order to recover all the needed keys. There are $\log(n) + 1$ subtrees to which user u belongs, each of them contributing with a number of seeds equal to its length minus one. taking into account a spare key (used when no user is revoked), the total number of stored key material is:

$$\sum_{i=1}^{\log n+1} (k-1) + 1 = \frac{(\log n + 1)\log n}{2} + 1 = \frac{1}{2}\log^2 n + \frac{1}{2}\log n + 1 \tag{1.8}$$

- The final message consists in all the identifiers of the covering subsets, plus the encryption of the session key under at most $2r - 1$ different keys.

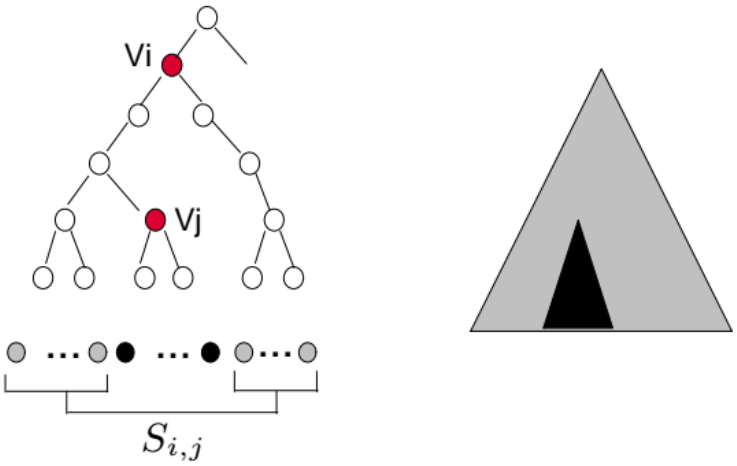


Figure 1.4: Subset difference scheme; covering subset structure.

Communication overhead is $\mathcal{O}\big(r \log(\frac{n}{r})\big)$ for the *complete subtree* implementation, since $r \log(\frac{n}{r})$ keys and $r \log(\frac{n}{r})$ node-identities must be broadcast. For *subset difference* the overhead is reduced to $\mathcal{O}(r)$, and in particular $2(2r - 1)$ messages, considering the subsets labels and the session key encryption. **Storage requirements** are $\mathcal{O}(\log n)$ and $\mathcal{O}\big(\log^2 n\big)$, which is reasonable for today's smart cards, as discussed above.

For both implementation the overhead depends at least linearly on the total number of revoked users since the start of the protocol, similarly to the stateful LKH version. This is not convenient for a highly dynamic system, and it might require to limit the statelessness of receiver, imposing a minimum period between each connection to the broadcast channel, in order to progressively eliminate old information. Storage overhead is $\mathcal{O}(\log n)$.

The main problem of these stateless protocols is **scalability**: the idea is to build a big enough system to be able to host users for a long enough period of time: when users are revoked, their leaves will never be occupied again and are considered "lost". This leads to a trade-off between scalability and efficiency: a very big system will allow to have a longer autonomy, but communication will be less efficient, and receivers will need to store a higher amount of keys. On the other hand, it is hard to know in advance how dynamic such a system will be: if users prefer short time subscriptions and keep entering and exiting the system, over-dimensioning might not even be enough. A solution has been suggested in [28], which presents a dynamic variant of subset difference which allows to reduce the broadcast cost by 50%, but it requires to shift the users inside the binary tree, redistributing keying material via unicast channel when the tree becomes sparse. Requesting a periodic connection to an aiding channel is even worse than limiting the statelessness of receivers, since it requires users to rely

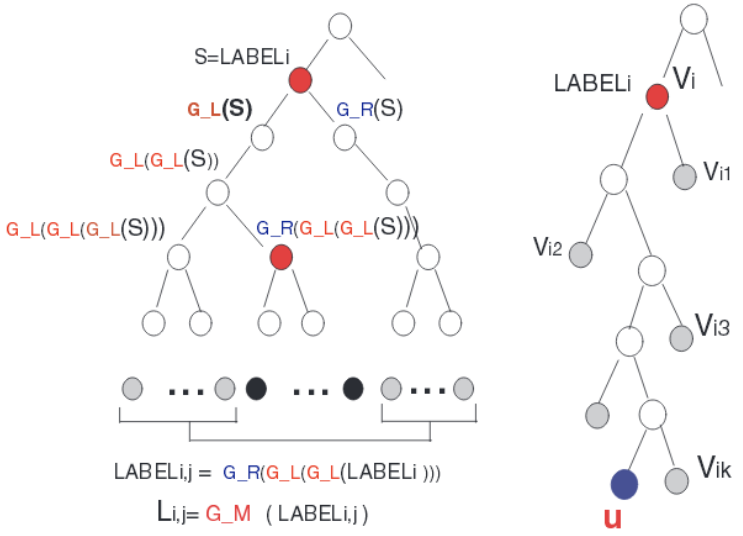


Figure 1.5: Subset difference scheme; label, keys and distribution mechanism.

on an aiding channel which might not always be available in GNSS applications. The authors of [28] claim that compacting users who arrive in the system at the same time could allow to achieve better performances, since there is higher probability that they will leave the system at close time instants, allowing to have a more compact tree. These considerations have been taken into account in the following chapters, where the underlying structure exploits the advantage of batch revocation thanks to the knowledge of the subscription time of users.

For what concerns **security**, both implementations of the subset cover mechanism have been proved to be secure (in the sense of key-indistinguishability) in the random oracle model. Furthermore, they are collusion resistant, since any colluding group of revoked users can not obtain any information about the new group key, even through combination of their secret keys [24].

1.8 LORE: a protocol based on linear ordering of receivers

It has been demonstrated that in order to achieve collusion resistance for symmetric key encryption systems, the communication overhead the system has to cope with is $\Omega(\log n)$. On the other hand, if collusion resistance is not strictly necessary, as it might happen for certain applications, a reduction on communication overhead can be achieved. The protocol LORE exchanges collusion resistance for an increase in efficiency: in order for the system to be safe assumption number 1 must be taken into account with reference to section 1.3.3: the secret information is stored in tamper resistant devices which are assumed to be safe.

LORE has been introduced by [15]: this protocol exploits a group structure which allows join and leave operations with a communication overhead of $O(1)$ for leave and $O(\log(n))$ for join operations.

Each of the users in the system has a secure unicast channel with a central key server (KS), established through a personal private key PK_i for $i = 1, \ldots, n$. KS will distribute privately to each user of the system two sets of secret keys: the *forward key set* (FSet) and the *backward key set* (BSet). The keys are distributed as illustrated in Fig. 1.6:

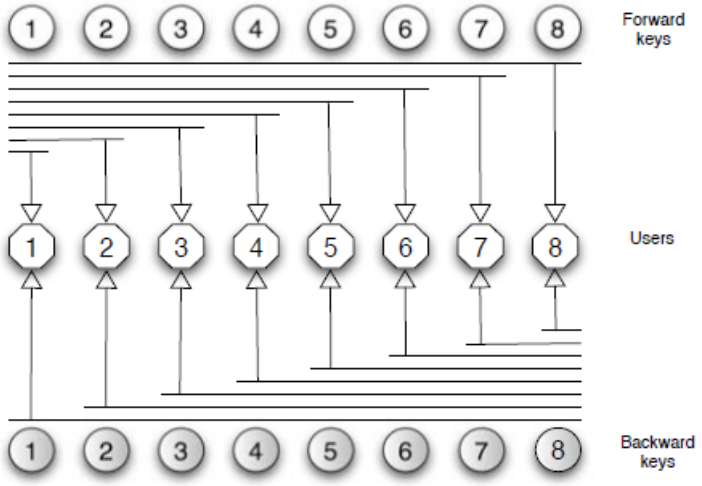


Figure 1.6: LORE forward and backward key assignment [16].

$$FSet(u_i) = \{f_j | 1 \leq j \leq i\}; \tag{1.9}$$

$$BSet(u_i) = \{b_j | i \leq j \leq n\}. \tag{1.10}$$

where f_j and b_j represent forward and backward keys for $j = 1, ..., n$.

The creators of the scheme argue that the keys can be generated with a binary hash tree, a structure which uses two one-way hash functions to create right and left branches from the value of each node. Such a construction allows an overhead for key distribution of $O(\log(n))$.

Joining new users

To guarantee backward secrecy, the group key must be updated right before a new user enters the group. When a user u_i enters the group G, the Key Server (KS) will send the following messages:

A **broadcast** message:

$$\text{KS} \rightarrow G : \left\{K'_G\right\}_{K_G} \tag{1.11}$$

A **unicast** message for u_i:

$$\text{KS} \rightarrow u_i : \left\{K'_G\right\}_{PK_i} \tag{1.12}$$

where K_G and K'_G represent the current and the new group keys respectively.

Revocation

To guarantee forward secrecy, the group key must be updated after a user leaves the group. When user u_i leaves, the following **broadcast** messages will be sent:

$$\text{KS} \rightarrow G : \left\{ \left\{ K_G^{'} \right\}_{f_{i+1}} \right\}_{K_G} \tag{1.13}$$

$$\text{KS} \rightarrow G : \left\{ \left\{ K_G^{'} \right\}_{b_{i-1}} \right\}_{K_G} \tag{1.14}$$

LORE requires $O(\log(n))$ broadcast messages for leave operations, since the key sets of each user should be updated.

Improvements on LORE

The authors of [16] propose an improvement on LORE in such a way to guarantee a communication overhead of $O(1)$ for join operations as well. This modification of the original protocol exploits cryptographic primitives called *multiplicative one-way functions*

Multiplicative one-way functions

A one way hash function has the multiplicative property if:

$$h(x_i)h(x_j) = h(x_i x_j) \qquad \text{where } x_i, x_j \in X \text{ for every } i, j. \tag{1.15}$$

Modular squaring

A modular squaring function, which is an instance of one-way functions, is be defined as follows:

$$f : Z_N \rightarrow Z_N; \quad f(x) = x^2. \tag{1.16}$$

where $N = pq$ and p, q are primes. Also $p = 3 \bmod 4$ and the same goes for q.

Inverting the square function is hard without the knowledge of p and q, and it's equivalent to the factorization of N. The square function has the multiplicative property.

The authors of [16] start from the key structure of LORE, but they compute $FSet$ and $BSet$ thanks to two one-way multiplicative hash functions:

$$h_F(x) = x^2 \, mod \, N_1 = x^2 \, mod \, p_1 q_1; \tag{1.17}$$

$$h_B(x) = x^2 \, mod \, N_2 = x^2 \, mod \, p_2 q_2; \tag{1.18}$$

The key distribution protocol requires the distribution of just two keys, f_i and b_i (one forward and one backward) to each user i, as shown in Fig. 1.7. The other keys of $FSet$ and $BSet$ of a generic user u_i can be computed applying the above hash functions:

$$f_{i-k} = h_F^k(f_i), \qquad 0 \leq k < i; \tag{1.19}$$

$$b_{i+k} = h_B^k(b_i), \qquad 0 < k \leq n - i; \tag{1.20}$$

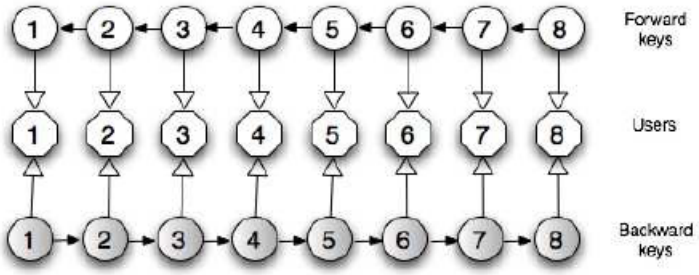


Figure 1.7: Modified forward and backward key assignment [16].

Join operations

The authors of [16] claim that KS will generate f_i and b_i for a new user u_i entering the system, but they don't explain how the system is supposed to do this. In fact, it is not possible to insert a new user randomly, breaking the linear structure to find a place between the other users. The hash key chain is static and it can't be expanded if not at its end point, because of the properties of one way functions. Also, since forward and backward chains go in opposite directions, it is not possible to insert any new user without having to recompute and redistribute the whole key sets. The only feasible solution is to build a bigger system with a lot of empty "slots" from the very beginning, fixing the maximum size n: a new user will be given an empty slot in the chain, together with the f_i and b_i linked to that slot. This fixed size causes **poor scalability** and it can represent a drawback for highly dynamic systems.

The messages the system has to send are very similar to those of LORE:

A **broadcast** message:

$$\text{KS} \rightarrow G : \left\{K_G^{t+1}\right\}_{K_G^t} \tag{1.21}$$

A **unicast** message for u_i:

$$\text{KS} \rightarrow u_i : \left\{K_G^{t+1} || f_i^{t+1} || b_i^{t+1}\right\}_{PK_i} \tag{1.22}$$

The authors claim that KS should also update the key sets in order to achieve **backward secrecy**. For this purpose they use the multiplicative property of the hash functions.

KS sends a new key seed s^t for rekeying:

$$\text{KS} \rightarrow G : \left\{s^t\right\}_{K_G^t},\ s^t \in Z_{N_1}. \tag{1.23}$$

The hash function $g : Z_{N_1} \rightarrow Z_{N_2}$ is exploited o allow each user u_m, $m \neq i$ to update their key seeds:

$$f_m^{t+1} = f_m^t h_F^{n-m+1}(s^t); \tag{1.24}$$
$$b_m^{t+1} = b_m^t h_B^m(g(s^t)); \tag{1.25}$$

The join operation of the proposed scheme requires only three broadcast messages per joined user, and it also offers the opportunity of updating the hash chains. Backward secrecy, as mentioned in previous sections, is not necessary for GNSS applications, but when it comes for free there is no reason to remove this property on purpose, as it will be explained below.

Let's now prove the correctness of these update operation, showing that user u_k can compute f_j^{t+1} with $j < k$ starting from the seed and f_k^t,:

$$\begin{aligned} h_F^{k-j}(f_k^{t+1}) &= h_F^{k-j}(f_k^t h_F^{n-k+1}(s^t)) && \text{(eq. 1.24);} \\ &= h_F^{k-j}(f_k^t) h_F^{k-j}(h_F^{n-k+1}(s^t)) && \text{(multiplicative property);} \\ &= f_j^t h_F^{n-j+1}(s^t) = f_j^{t+1} && \text{(properties of hash chains).} \end{aligned}$$

Similarly for the backward keys.

Leave operations

In order to ensure forward secrecy, the group key should be changed every time a user leaves the group. Similarly as for LORE:

$$KS \rightarrow G : \left\{\left\{K_G^{t+1}\right\}_{f_{i+1}^t}\right\}_{K_G^t} \tag{1.26}$$
$$KS \rightarrow G : \left\{\left\{K_G^{t+1}\right\}_{b_{i-1}^t}\right\}_{K_G^t} \tag{1.27}$$

Then another message containing a **new seed** is broadcast as after a join operation, encrypted with the new group key, as in eq. 1.23. Leave operations take three broadcast messages each, and the set of keys is updated every time.

Adaptation of the protocol to GNSS scenario

In a limited bandwidth scenario as the satellite broadcast channel it would be better to have a cheaper mechanism to add users, which does not require rekeying operations for every new user. There are two ways to simplify joining operations, improving communication overhead:

- Give up on backward secrecy, which is not necessary for GNSS, and let new members in by simply giving them privately the current group key and the two key seeds;
- Select a minimum value T_{min} for the subscription period and concentrate all the join requests in the beginning of the next subscription slot. This seems a reasonable approach in this kind of system. Since none of the users who are about to enter the system knows the current group key, they can join the system in a single step, requiring only two broadcast messages for the whole operation. e.g., if the minimum subscription period lasts a week, at the beginning of each week there will be an accumulated join operation which will require the broadcast of two messages, independently of the number of joining users. One message will be the broadcast of the new group key encrypted with the old one, and the other will contain the new seed for key update. These two messages will be broadcast in a cyclic fashion in order for the system to give broadcast support to **stateless receivers**. A period T_{max} could be chosen as the maximum amount of time in which the same messages will be cyclically broadcast (e.g., each couple of join messages could be broadcast for four weeks: this would create an overhead of eight broadcast messages at each time instant). Nevertheless, this solution **limits the statelessness of receivers** in the sense that new users will be required either to connect at least once every T_{max} or to rely on an aiding channel when this is not possible.

The authors of the article accounted for single leaving users, while in this subscription scenario it would be better to allow for batch leaving operation (referred before as *subscription expiration*).

Any two users of the scheme have, together, all the keys required to decipher all messages. This is why revoking more than one user at a time with constant overhead is not possible, since it would require three broadcast messages for each leaving user. If we take into account that these messages would need to be broadcast multiple times, the overhead becomes prohibitive.

A new idea comes from an observation: there is no cheap mechanism to revoke two random users in a single $O(1)$ operation, but if the user are next to each other in the linear structure, it becomes easy. Let's consider the scenario in which we want to revoke users u_3, u_4 and u_5 in a single operation, with reference to Fig. 1.6. The two broadcast messages the system will send are:

$$KS \rightarrow G : \left\{ \left\{ K_G^{t+1} \right\}_{f_6^t} \right\}_{K_G^t} \tag{1.28}$$

$$KS \rightarrow G : \left\{ \left\{ K_G^{t+1} \right\}_{b_2^t} \right\}_{K_G^t} \tag{1.29}$$

This way all and only the remaining users will be able to decode the message and subsequently update their keys. This requires a constant communication overhead, no matter how many users are revoked, **as long as they are placed next to each other**. This solution allows to repopulate old slots left empty by leaving users, since the key sets are updated every T_{min}.

Since the group key needs to be changed right after a leave operation and right before a join, this imposes a natural order for the two operations. Every T_{min} a programmed revocation will be performed, a new seed will be issued and broadcast, and a join operation will follow. This allows to spare a rekeying operation, and reduces the communication overhead to three broadcast messages per T_{min}, which can be cyclically broadcast for T_{max}, giving a constant overhead of $3 \cdot \frac{T_{\text{max}}}{T_{\text{min}}}$.

How is it possible to ensure that leaving users will be placed in a unique segment? As mentioned in sec. 1.2.2, it is reasonable to assume the system has knowledge of the user's subscription time. This information allows to smartly design the protocol in order to group users according to the date of their departure. This scheduling will divide the linear structure into different logical segments which represent different time slots: each time slot will represent a time period of T_{min}. The subscriptions will be offered to users in packets of one or more T_{min}. As soon as a user buys a packet, the system knows in which segment he should be inserted (and which keys he shall be provided with). As argued before, the system should be built with a maximum number of users in mind, since once the structure is built, it can't be expanded. The system should allow for a good margin of free space (i.e., keys) for every segment representing different subscription periods.

Performances

Since the number of keys a user has to store is only two, the large dimension of the system will not affect storage requirements; **computational complexity**, on the other hand, is affected by the dimension of the system in a linear fashion: each receiver will need to perform $\mathcal{O}(n)$ hash operations in order to retrieve the new group key. This is not a trivial requirement for big systems, but a trade-off can be considered between computational complexity and **storage overhead**: each user might store several intermediate values along the key hash chain, in order to allow for a faster group key retrieval. **Communication overhead** is constant, and this feature comes at the price of:

- A constraint in the statelessness of receivers, which will have to connect once every T_{max};
- A decrease in security, since the protocol is 1-resilient, and it can't resist collusion attacks of any size different from 1. If a service provider has reason to trust its tamper resistant devices, the weaker security of this protocol might not be an issue.

Nevertheless the consequences of the absence of collusion resistance should be carefully investigated, in order to understand which kind of attacks could affect the system. The absence of collusion resistance implies that any group of two or more former users (who might have been revoked in different time slots) can always break the scheme if they combine their private keys: let's say an attacker

buys two receivers with different expiration times, T_i and T_j. The two devices have different keys, and when device d_1 expires, device d_2 can still get the new group key and the seed to update the key chains. If the attacker manages to extract the private keys of device d_1 and observe the deciphered broadcast received by device d_2, he will be able to use the seed to update the private keys of device d_1, and thus obtain the ability of deciphering every broadcast transmission (even after the expiration of d_2), until the system performs a complete rekey operation. This is not the only threat: since the seed for private key update is equal for everybody, if a users leaks it, all former users will in principle get back their decryption capabilities. This explains why this protocol is suitable only for system which can rely on a secure tamper resistant device. Moreover, this revisited version of LORE doesn't make random revocation any easier: since the misbehaving users can not be known in advance, they will probably not be located in the same time slot, and random revocation will have an additional overhead of $3 \cdot t$, with t the number of revoked users.

1.9 Bilinear map based scheme

The security of this scheme, which was proposed by Boneh, Gentry and Waters [18] in 2005, is based on the bilinear Diffie-Hellman exponent assumption (BDHE): given a group $\mathbb{G}$ of prime order p, $\alpha \in \mathbb{Z}_p$ and the vector

$$(h, g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^i}, g^{\alpha^{i+2}}, \ldots, g^{\alpha^{2i}}) \in \mathbb{G}^{2\ell+1}$$

where g and h are generators of the group, it is hard to find $e(g,h)^{\alpha^{\ell+1}}$, where $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map.

The scheme can provide collusion resistance against **static** adversaries, and it is a **public key** encryption system. Being the encryption key public, everyone could spoof messages and pretend they come from the system. This is why a digital signature shall be provided in order to authenticate all the key management messages (note that this holds for symmetric encryption, too). This system allows for *random revocation* (opposed to *periodic revocation*, which manages the ordinary subscription system). The communication overhead is $\mathcal{O}(1)$ and it's independent of the number of revoked users, the storage requirements are $\mathcal{O}(1)$ and the public key size is $\mathcal{O}(n)$.

System setup

Let $\mathbb{G}$ be a bilinear group of prime order p. The central station picks a generator g, $\alpha \in \mathbb{Z}_p$ and a random $\gamma \in \mathbb{Z}_p$. The central station computes $g_i = g^{\alpha^i}$ for $i = 1, \ldots, n, n+2, \ldots, 2n$ and $v = g^\gamma \in \mathbb{G}$. The public key is:

$$K_{\text{pub}} = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) \in \mathbb{G}^{2n+1}. \tag{1.30}$$

The private key for user $i \in \{1, \ldots, n\}$ is:

$$d_i = g_i^\gamma \in \mathbb{G} \tag{1.31}$$
$$= v^{(\alpha^i)}. \tag{1.32}$$

Encryption

Encryption takes as input the set of privileged users $\mathcal{S}$ and the public key. The central station picks a random $t \in \mathbb{Z}_p$ and sets the group key $K_G = e(g_{n+1}, g)^t \in \mathbb{G}_1$ and broadcasts

$$H = \left(g^t, \left(v \prod_{j \in \mathcal{S}} g_{n+1-j} \right)^t \right) \in \mathbb{G}^2. \tag{1.33}$$

Decryption

Decryption at user i takes as input $\mathcal{S}$, d_i, H and the public key K_{pub}. Let's call $H = (C_0, C_1)$ for simplicity of notation. The decryption algorithm recovers K_G as follows

$$K_G = \frac{e(g_i, C_1)}{e\left(d_i \prod_{j \in \mathcal{S}, j \neq i} g_{n+1-j+i}, C_0 \right)} \tag{1.34}$$

Proof of correctness

The proof exploits the fact that $g_i^j = g_{i+j}$.

$$K_G = \frac{e(g_i, C_1)}{e\left(d_i \prod_{j\in\mathcal{S}, j\neq i} g_{n+1-j+i}, C_0\right)} = \frac{e\left(g^{(\alpha^i)}, \left(v \prod_{j\in\mathcal{S}} g_{n+1-j}\right)^t\right)}{e\left(v^{(\alpha^i)} \prod_{j\in\mathcal{S}, j\neq i} g_{n+1-j+i}, g^t\right)} \tag{1.35}$$

$$= \frac{e\left(g^{(\alpha^i)}, (g_{n+1-i})^t\right) e\left(g^{(\alpha^i)}, \left(v \prod_{j\in\mathcal{S}, j\neq i} g_{n+1-j}\right)^t\right)}{e\left(v^{(\alpha^i)} \prod_{j\in\mathcal{S}, j\neq i} g_{n+1-j+i}, g^t\right)} \tag{1.36}$$

$$= \frac{e\left(g_{n+1}, g\right)^t e\left(g^{(\alpha^i)}, \left(v \prod_{j\in\mathcal{S}, j\neq i} g_{n+1-j}\right)^t\right)}{e\left(v^{(\alpha^i)} \prod_{j\in\mathcal{S}, j\neq i} g_{n+1-j+i}, g\right)^t} \tag{1.37}$$

$$= \frac{e\left(g_{n+1}, g\right)^t e\left(g, \left(v^{(\alpha^i)} \prod_{j\in\mathcal{S}, j\neq i} g_{n+1-j+i}\right)^t\right)}{e\left(v^{(\alpha^i)} \prod_{j\in\mathcal{S}, j\neq i} g_{n+1-j+i}, g\right)^t} \tag{1.38}$$

$$= e\left(g_{n+1}, g\right)^t \tag{1.39}$$

And this proves the correctness of decryption.

It is worth underlining that since this is an asymmetric cryptographic scheme everybody could spoof messages if no authentication technique is implemented. For example, a digital signature could be employed to sign the group key in such a way to ensure the message comes directly from the central station. The public key is $\mathcal{O}(n)$, but since it is public, it doesn't need to be stored in the smart card (or tamper resistant module), and it can be kept in the clear inside the decryption device.

Consideration on Bilinear Map system

This key management scheme has outstanding performances for what concerns ciphertext size and private key size. On the other hand it doesn't scale well nor it adapts easily to a dynamic broadcast system where users come and go. The system needs to be over-dimensioned since once a user is revoked, his "spot" in the system can't be occupied by another user. Moreover, the public key of size $\mathcal{O}(n)$ requires additional memory inside the decoding device. For what concerns computational complexity, it is dominated by the $\mathcal{O}(n)$ multiplications during the decoding phase. This result can be improved by providing each user i with the value $\prod_{j=1, j\neq i}^{n} g_{n+1-j+i}$. With this solution the computational complexity becomes $\mathcal{O}(r)$ divisions, with r the number of revoked users.

Security of the scheme

In [18] the authors prove that the scheme is collusion secure against chosen ciphertext attack for non-adaptive adversaries. The attack is modeled a game between a challenger and an attacker. The attacker is non-adaptive, or *static*, meaning that in the model it chooses the set of users it wants to attack before seeing the public key and messages exchanged by the system. This is a technical assumption which is useful to prove security, even though in reality attackers might strike after retrieving some information about the scheme. However, there is no reason to state that the scheme is weak against adaptive adversaries. The authors argue that achieving provable collusion resistance against adaptive adversaries comes at the cost of a higher overhead.

In the **setup phase** of the attack the adversary chooses the set of users it wants to target, and it receives from the system the private keys of all users in the complementary set. Afterwards, during the **query phase**, the attacker can issue as many decryption queries as it wants: it selects a subset of the target set for which the message is intended and forges the desired ciphertext, obtaining the decryption from the challenger.

In the **challenge phase** the challenger selects two random messages from the message space, and encrypts one of them, choosing at random, with the target set as the intended receiver set. The attacker will receive the resulting encryption header and both plaintext messages: if it manages to guess which of the messages corresponds to the ciphertext it wins the game.

The authors prove in [18] that if there is an algorithm that is capable of winning the game with non negligible advantage (with respect to probability $\frac{1}{2}$), then the bilinear Diffie-Hellman problem can be solved with non negligible advantage.

1.10 Hash chain based key management scheme for GNSS

The authors of [21] proposed a key management scheme for GNSS authentication services. The organization of the group keys exploit hash chains in a structure that is shown in Fig. 1.8. The group key is renewed after a period of time T_G, and the idea is to allow different categories of users according to their "level of trust". This level can depend on the security of the device they are relying on, or on their type of subscription. The structure is based on different levels of hash chains in hierarchy: the values from the top chain allow to retrieve a big number of keys, which allow to decode for a longer period of time, and they might be given to users with a high level of trust. Values from the lower chains allow for a lower autonomy, and can be given to lower level users. Each level in the hierarchical hash chain will expose the system for a different period of time in case of key compromise. The idea in Curran's proposal was that of using this scheme for spreading code generation, having the lower level values corresponding to chips of the spreading code itself. This way some of these chips can be given directly to low level users without compromising all the others.

An adaptation of this system to the current problem is the following: each user receives a certain type of key depending on his subscription type: if it is a short time subscription the user can be given a seed

of an intermediate hash chain, while for wider periods the user can be given an element of the top hash chain. This way periodic revocation is very easy and clean, as it is scheduled by the key distribution mechanism and it does not require any key management message. However, random revocation is impossible and a compromise of future keys would completely break the system, requiring a complete rekey operation.

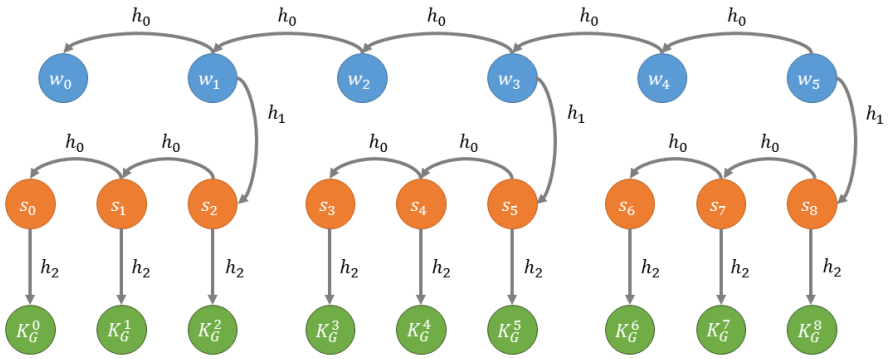


Figure 1.8: *HashOnly* key management system.

Chapter 2

Comparison between the key management schemes

The aim of this section is to provide a comparison of the broadcast encryption schemes presented so far.

From now on the size of the group key will be called γ, while the size of a user's private key in the key management system will be called λ. These are both important parameters for performance comparison. A symmetric key encryption operation is assumed to output a ciphertext of the same size of the corresponding plaintext. As it is usually done in literature, for the moment the overhead due to the communication of the identities of the revoked users (for random revocation events) is not considered as part of the ciphertext. The overhead due to message authentication will also be considered further in the next sections, since, as set identification messages, it is common to all protocols, and it is not useful in the scope of a comparison. The additional overhead due to the *set identification problem*, mentioned in section 1.4, will be further studied in the following chapters.

2.1 *NewLore*

This key management scheme refers to the adaptation of LORE [15] to GNSS scenario. For what concerns **Ciphertext size** for periodic revocation, *NewLore* broadcasts three messages: two of length γ (new group keys encoded twice) and one of length λ (new seed for private key update). The group key and two private keys of length λ must be kept inside the **secure memory**, and at least twice this amount of memory shall be available in order to perform the sequential computations needed to recover the other keys inside the chain. Secret storage requirements are $\mathcal{O}(1)$, while from the **security** point of view the scheme is not collusion resistant. This scheme allows for subscription expiration operations with overhead $\mathcal{O}(k)$, where k represents the number of minimum subscription periods T_{min} (subscription expiration sessions) during which the same rekeying message is broadcast (the duration of support for statelessness of receivers is thus limited to $kTmin$). Random revocation is

linear in the number of revoked users. Scalability is acceptable since older keys can be updated and redistributed. The computational complexity at receiver side is $\mathcal{O}(n)$ hash operations.

2.2 *BiMaps*

This is Boneh's key management scheme, which uses asymmetric cryptography, and it is the only one in literature who manages to achieve constant **storage requirements** and **communication overhead** while maintaining the property of **collusion resistance**. This system allows for *random revocation*, as well as periodic, which are performed in similar ways and with the same communication overhead (except for the set identification message, which is not considered here). Secret storage is $\mathcal{O}(1)$, while public storage is $\mathcal{O}(n)$, since it is required to store a public key of $2n + 1$ elements of a group, plus negligible scheduling information for the subscription expiration mechanism. The computational complexity at receiver side is $\mathcal{O}(t)$ divisions, with t the number of revoked users at each revocation operation. **Scalability** is very poor, since the structure has a fixed size which can not be expanded without requiring a rekey of all users, but a clever implementation of the protocol, as shown in the following sections, will improve this aspect.

2.3 *HashOnly*

This scheme is an adaptation of Curran's key management proposal for the commercial service. The idea is to exploit the temporal meaning of hash chains for the management of subscription expiration. The keys are renewed according to a hash chain, in such a way that an older key can be easily derived from a more recent one, but the opposite is not possible. Each user shall be provided with the key which is further in the future, according to the type of subscription. From the last key, all previous one can be easily derived inside the tamper resistant device. The nice feature of this scheme is that it allows to completely avoid key management messages, and still achieve **collusion resistance** (in the sense that revoked users do not have any information about next group keys). The **storage overhead** depends on the length of the subscription, but it is constant in n. The drawback of the scheme is **security**: everyone shares the same keys, and future group keys might be stored inside tamper resistant devices for a long amount of time. Since those keys are the same for every user, the probability of users breaking the tamper resistant device and leaking future keys grows. If this is the case the system will need a complete rekeying of all users. **Random revocation** is not an option with this kind of scheme, since it is all based on previous knowledge on the departure times of users.

2.4 Complete subtree scheme (*CS*)

In the subset cover framework the complete subtree algorithm gives a **communication overhead** of $\mathcal{O}\left(r \log \frac{n}{r}\right)$ with r the number of revoked users since the start of the protocol. Secret storage is

$\mathcal{O}(\log n)$, while **computational complexity** is logarithmic in the number of users. The protocols, as all those who fall in the subset cover framework, is collusion resistant, but **scalability** is quite poor, since old keys can't be reused, and for this reason frequent rekeying operation via aiding channel might be required. In order for the protocol to be reasonably applicable to GNSS scenarios, the **statelessness** of receivers needs to be limited.

2.5 Subset difference scheme (*SD*)

Subset difference implementation of the subset cover framework has similar features with respect to CS, but the **communication overhead** is improved up to $\mathcal{O}(2r-1)$ with r the number of revoked users since the start of the protocol. This comes at the price of a bigger **secret storage**, which is $\mathcal{O}\big(log^2 n\big)$. The same considerations made for CS are valid when it comes to scalability, security and statelessness of receivers.

2.6 *SecretSharing*

The scheme from Naor and Pinkas allows to achieve a communication overhead of $\mathcal{O}(r)$, with r the number of revoked users since the start of the protocol, while maintaining $\mathcal{O}(1)$ secret storage. This scheme does not have really good scalability, and its security is weaker than subset cover protocols: the notion of collusion resistance is replaced with t-resilience, with t the maximum number of users which can be revoked at each step (and the maximum collusion size). Computational complexity is $\mathcal{O}(t)$ multiplications. This protocol will probably require to significantly limit the statelessness of receivers in order to cope with the normal dynamic of users joining and leaving the system, since a smart subscription expiration mechanism can not be devised.

For what concerns **Ciphertext size** for periodic revocation, *NewLore* broadcasts three messages: two of length γ (new group keys encoded twice) and one of length λ (new seed for private key update). The group key and two private keys of length λ must be kept inside the secure memory, and there must be at least twice this memory in order to perform the sequential computations needed to recover the other keys inside the chain. *Bilinear maps* broadcasts a message which contains two elements of a group of order p: an element of this group can be represented with $\log_2(p)$ bits, which is the size of the private key, λ. Since the suggested value for $\log_2(p)$ is 160 bits, in this scheme the ciphertext size is const $= 2 \cdot \lambda = 2 \cdot 160 = 320$ bits. A private key of size λ and the group key of size γ must be kept inside a secure hardware. The public key of size $(2n+1)\lambda$ can be kept in a non-secure device. *HashOnly* on the other hand doesn't require any overhead for simple subscription expiration, and only two elements need to be stored in the secure hardware: the seed and the current group key, which have both size γ.

Scheme	*NewLore*	*BiMaps*	*HashOnly*	*CS*	*SD*	*Secret Sharing*
Ciphertext size for subscription expiration	$\mathcal{O}(1)$	$\mathcal{O}(1)$	0	$\mathcal{O}\left(r \log \frac{n}{r}\right)$	$\mathcal{O}(2r-1)$	$\mathcal{O}(r)$
Secret Storage	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(\log n)$	$\mathcal{O}\left(log^2 n\right)$	$\mathcal{O}(1)$
Additional memory	-	$\mathcal{O}(n)$	-	-	-	-
Computational complexity	$\mathcal{O}(n)$ hash	$\mathcal{O}(t)$ divisions	$\mathcal{O}(1)$ hash	$\mathcal{O}(1)$ decryption	$\mathcal{O}(\log n)$ hash	$\mathcal{O}(t)$ products
Collusion resistance	1-resilient	n-resilient	n-resilient	n-resilient	n-resilient	t-resilient
Scalability	decent	poor	good	poor	poor	decent
Needs to limit statelessness	yes	no	no	yes	yes	yes
Random rev. overhead	$\mathcal{O}(r)$	$\mathcal{O}(1)$	not allowed	$\mathcal{O}\left(r \log \frac{n}{r}\right)$	$\mathcal{O}(2r-1)$	$\mathcal{O}(r)$

Table 2.1: Features and performances of the schemes.

Tables 2.6 and 2.6 show the performances in detail and a numeric example, assuming the system is working in the normal subscription expiration dynamic, with $n = 10000$, $r = 100$, $t = 50$. Let's consider for *NewLore* hash values of 256 bits, in order to take into account reductions of entropy due to long hash chains. The group key can be 128 bits long. For *BiMaps* let's take the value $\lambda = 256$ (for 128 bits of security), and $\gamma = 3072$ bits, which will be reduced via hashing to a 256 bit key (see sec. 3.1.2). For *HashOnly* all keys will be of 256 bits because of the hash chains, while for *CS* and *SD* the group key will be 128 bit long, while the private keys will be 256 bit long (because of the hash key tree structure). For what concerns *SecretSharing*, which uses elliptic curve cryptography like *BiMaps*, the polynomial shares (secret keys) are of size 256 bit, while the group key size is 3072 bit.

Scheme	Ciphertext size for sub. expiration	Secret Storage	Additional memory
NewLore	$2\gamma + \lambda$	$\approx 2\lambda + \gamma$	-
BiMaps	2λ	$\approx \lambda + \gamma$	$\lambda(2n + 1)$
HashOnly	0	$\lambda + \gamma$	-
CS	$\lambda r \log \frac{n}{r} + r \log n + \gamma$	$\lambda \log n + \gamma$	-
SD	$\lambda(2r - 1) + r \log n + \gamma$	$\lambda \log^2 n + \gamma$	-
SecretSharing	$r\lambda + r\gamma$	$\lambda + \gamma$	-

Table 2.2: Detailed performances of the schemes.

Scheme	Ciphertext size for sub. expiration	Secret Storage	Additional memory
NewLore	640 bit	1152 bit	-
BiMaps	512 bit	3328 bit	5 Mbit
HashOnly	0	512 bit	-
CS	168 Kbit	25 Kbit	-
SD	50 Kbit	44 Kbit	-
SecretSharing	325 Kbit	3328 bit	-

Table 2.3: Example for a security parameter of 128 bit, with $n = 10000$, $r = 100$, $t = 50$.

2.7 Pros and Cons of the analyzed schemes

The choice of a suitable key management system should take into account all the critical aspects of the specific application. Here the aim is minimizing the **communication overhead** and providing an efficient method for **revocation**, both periodic (subscription expiration) and random. The **security** of the overall system is the primary aspect that should drive the choice of a key management scheme.

Sometimes the specific application allows to sacrifice some aspects of security (e.g., backward secrecy or even collusion resistance) in order to achieve feasible performance. However, any optimization on storage and bandwidth usage should come after a careful analysis on security requirements.

Table 2.6 underlines how the schemes which are usually preferred for broadcast encryption (*CS*, *SD*, *SecretSharing*) are not optimal for limited bandwidth applications supporting stateless receivers. In this scenario, indeed, it is desirable to achieve constant communication overhead at least for the normal dynamics of the system, where no random revocation is performed and users are joining and leaving the system according to their needs.

For what concerns **communication overhead**, *HashOnly* is the best possible solution, as it has a very simple structure which allows for subscription expiration based on the knowledge pf users' subscription types, without the need of broadcast messages for rekeying after *leave* operations. **Collusion resistance** is intrinsic in the structure of the system and in its temporal organization: the secret information given to each user has a specific expiration date, and one-way functions ensure that expired users lose the ability to recover new keys. Also, this system has no **scalability** issues, since all users which subscribes for the same period of time are given the same key. This impacts on security: since the keys are common to all users, not only the system has no random revocation capabilities, but it becomes easier to perform cryptanalysis, since attackers can target the same keys on multiple smart cards. A future group key compromise would have effect on the whole system, since the keys are decided in advance and distributed to all users. Changing future keys requires changing the hash chain and redistributing the seeds to all users, which corresponds to a complete rekey of the system. For this reason the convenient structure of *HashOnly* might be exploited for shorter time periods, to ensure a frequent change of the group key, and a more complex system should be on top of it, in order to allow random revocation.

In subscription expiration mode, *NewLore* allows to achieve a communication overhead which is independent of the number of revoked users (it only depends on the number of elapsed subscription units, k). When it comes to random revocation though, overhead is again linear in the number of revoked users. Moreover, this scheme is weaker from the security point of view, since it lacks collusion resistance.

BiMaps allows to have constant communication overhead in subscription expiration mode, and it surprisingly allows the same performances for random revocation: so far, this is the only scheme which is able to achieve this results in combination with constant storage requirements. The protocol proposed by Boneh has been adapted to subscription scenarios by adopting a batch revocation mechanism and making the reasonable assumption of knowing the subscription type of users. This modification brings the protocol closer to the examined scenario, but another issue need to be solved: scalability is still very poor, forcing users to frequently connect to the aiding channel for rekeying and space recovery. The following sections presents a solution to this issue through the combination of hash chains (with reference to *HashOnly*) and *BiMaps*.

Chapter 3

RevHash: a novel key management scheme based on hash chains and *BiMaps*

From the considerations seen in the previous section it is possible to summarize the ideal properties of a key management scheme for GNSS subscription services:

- *Subscription expiration* should require constant communication overhead and exploit knowledge of users' subscription types. This can be obtained by incapsulating the keys in a linear structure, known a priori by the receivers.

- *Random revocation* should be allowed, possibly with constant communication overhead. The overhead needed to identify revoked users can not be avoided and should be included in the communication overhead. Since this overhead will increase with the number of revoked users, a trade-off has to be considered between revocation capabilities and efficiency.

- *Scalability* is a major issue: the ideal scheme should be as scalable as possible, allowing a longer autonomy for users who can not rely on a permanent aiding channel for rekeying. Scalability is often expressed in terms of key reusability: since keys are incapsulated in a fixed structure (re-dimensioning requires communication overhead), an update mechanism should allow the system to modify old revoked keys and redistribute them to new users.

- Group keys should have a short lifetime in order to make cryptanalysis harder. Nevertheless, if every time the group key changes the receivers have to decode a new key management message, the performances might be degraded. Ideally every key management message allows for a certain period of autonomy, providing the users with all the group keys they might need for a defined amount of time. Many key management schemes use hash chains for this purpose: from the seed of a chain many other values can be obtained. The last values of the chain will be the first group keys: the property of *one-way* functions makes it hard to retrieve the previous keys

along the chain, even with knowledge of these values. The period of group-keys can be made small enough to limit the exposure of the system in case of key compromise.

The idea is to exploit the scheme called *BiMaps*, discussed in the previous section in order to deliver the seed for a one-way chain of group keys which should last for a predefine amount of time T_{CHAIN}, multiple of T_{min}, the minimum subscription time. Subscription expiration and random revocation will both be performed every T_{min}. The concept of the scheme, which will be called *RevHash* is represented in Fig. 3.1.

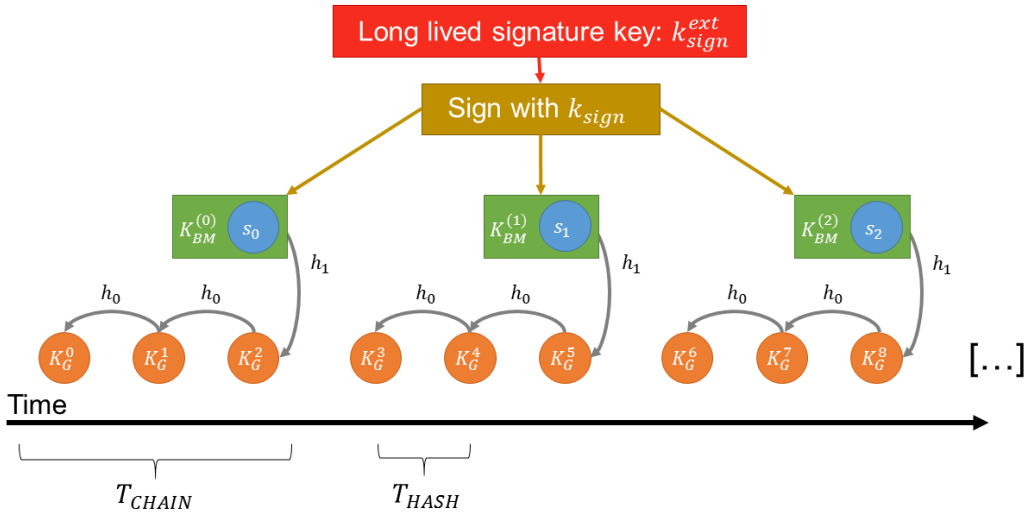


Figure 3.1: *RevHash* key management scheme. $K_{\text{BM}}^{(t)}$ represents the encrypting key material used in *BiMaps*; K_{sign} is a private key for authentication and K_{sign}^{ext} is a signature key higher in the *chain of trust* and therefore more secure.

3.1 Protocol description

The *Bilinear Maps* revocation system stays on top of each hash chain (orange on Fig. 3.1). There might be two types of subscription: short term and long term subscriptions. Short term subscription lasts for the duration of an single hash chain (represented in orange in Fig. 3.1): users who pay for short term services are given the seed of a single hash chain and therefore they can get the group keys for the duration of that period. This kind of subscription does not come with a private key for the revocation system: should the system suspect a key compromise, it could issue an authenticated message informing users that a shift forward in time will be made and a new key-chain will be issued. The duration of this hash chain shall be determined after some security analysis, since hash chains are known to reduce the entropy of the key in the long run [25].

Users who wish to access the services for longer periods can subscribe for multiples of T_{min}. In this case they will be given a new hash chain seed before the beginning of each chain through the

BiMaps scheme. Users who buy long term subscriptions will be provided with a private key in a tamper resistant device (typically a smart card), and will become part of the revocation system.

$RevHash$ makes use of several keys:

- Each user has a key K_i^{sym}, a private, symmetric key to establish a secure channel with the central station;
- Each user with a long-term subscription has a private key d_i for the revocation system;
- All users with a long-term subscription share the same public key D_{pub}, which is $\mathcal{O}(n)$;
- The key K_G^{rev} can be obtained after the broadcast of a revocation header (see section 1.9) only by the authorized users;
- The first of the group keys, $K_G^{t_1}$ is the key obtained from K_G^{rev} by the application of a hash function, to be defined later. For this reason, K_G^{rev} will also be called *seed*.
- The group keys $(K_G^{t_2}, \ldots, K_G^{t_m})$ can be obtained as a hash chain from $K_G^{t_1}$ through the use of a hash function, to be defined later. Group keys are used to encode the actual data.
- All key management messages shall be signed by the system through a private key from an asymmetric key pair, $(K_{\text{priv}}^{\text{sign}}, K_{\text{pub}}^{\text{sign}})$.
- Another asymmetric key pair, $(K_{\text{priv}}^{\text{ext}}, K_{\text{pub}}^{\text{ext}})$ will be used as an external layer of authentication for exceptional key management messages i.e., alert messages, revocation and renewal of the signature key pair, revocation of a group key or a part of the chain.

Let's first concentrate on *subscription expiration*, which refers to user revocation due to the end of the subscription time. As commented above, the most convenient implementation for this function exploits a linear ordering of receivers. As can be seen in section 1.9, the private keys d_i are ordered by indexes, which force an order on receivers as well. If many bits are required to identify k among n users, this is not true when these users are contiguous in the linear structure. Identifying an interval of k bits inside n bits requires fewer bits, and if the revocation order is known in advance, receivers may even be programmed accordingly, without the need of any further message to specify who is leaving the system. For this reason, once the system is built, private keys should be assigned to users according to the duration of their subscription.

It is not possible to know in advance how many users will choose a certain subscription, but the system can be over-dimensioned in order to have some margin. The system could then be divided into slices, each containing a certain number of private keys. Users with the same subscription expiration time will be provided with a private key from the same slice. With this organization, at the end of each time slot only contiguous users will be revoked, allowing for a much shorter identification message. This idea is shown in Fig. 3.2.

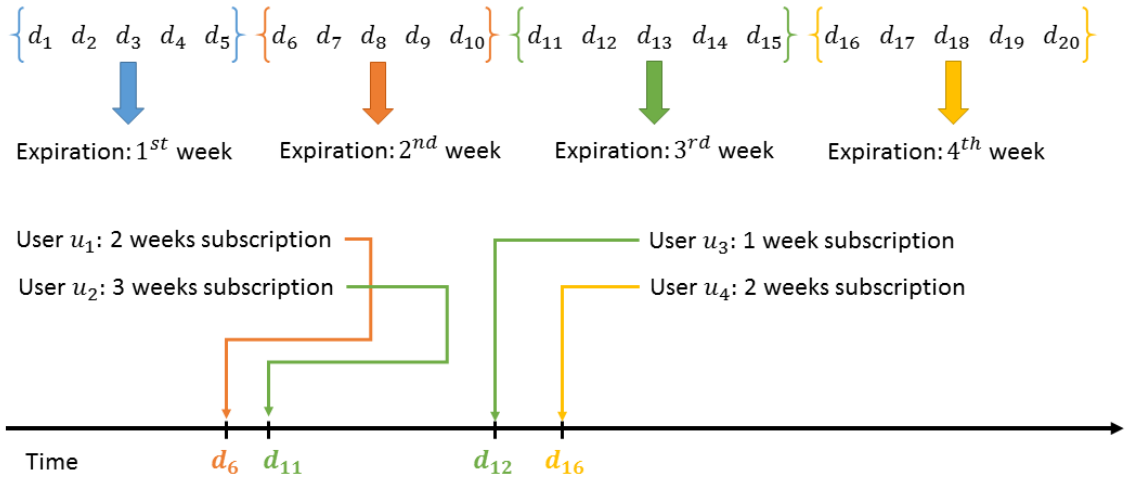


Figure 3.2: Linear ordering of receivers for facilitating subscription expiration.

3.1.1 Rekeying

As mentioned in the introduction, it is best practice for a key management scheme to limit the exposure of keys by limiting their cryptoperiod. In Boneh's scheme, *Bilinear Maps*, there are two ways of rekeying: one provides complete rekeying, and it requires changing both the private and the public keys, forcing every user to download a conspicuous amount of data (for $n = 10^6$, D_{pub} is around 60 megabytes). The second rekeying mechanism requires that each user receives a new private key through a private and secure channel. The public key can stay the same, while a new $v = g^{\gamma}$ will be issued and distributed to everyone. The first mechanism is more drastic and demanding and should be applied every T_{pub}, while the latter only requires the download of a new key, and can be carried out with shorter intervals, T_{priv}. The three rekeying periods are shown in Fig. 3.3.

Rekeying can be needed for two reasons:

- The cryptoperiod of the keys must be limited in order to maintain the system secure against attacks;
- The capacity of the scheme (intended as the number of spare private keys for decryption) has been consumed and new users can't enter the system without causing confidentiality issues.

The *Bilinear Maps* system as presented in [18] doesn't allow to recover old keys once the former users leave the system (i.e., their subscription ends). This means that even if the system is over-dimensioned at the start of service, eventually it will run out of space and have to rekey all the users, mining the continuity of service provision. For this reason it seems necessary to provide a mechanism that allows to progressively recover old keys without major impacts in performances. Let's call T_{upd} the update period of users' private keys finalized to old key recovery. Fig. 3.3 represents rekeying periods.

Two mechanisms were devised in this thesis:

- The first mechanism takes advantage of the structure of the users' private keys allows to introduce a broadcast key-update mechanism. Since the private keys of each user i is $d_i = g_i^{\gamma}$, with γ chosen at random by the system, all the private keys could be updated via broadcast of a new random value γ_1, through the operation $d_i' = d_i^{\gamma_1} = (g_i^{\gamma})^{\gamma_1} = g_i^{\gamma \cdot \gamma_1}$, which would take place inside the smart card. This random value could be encrypted by the current group key, and broadcast for a period of T_{upd}, with $T_{\text{upd}} < T_{\text{priv}} < T_{\text{pub}}$, before imposing the private key update. This mechanism would allow to recover old keys reintroducing them in the system after at least $2T_{\text{upd}}$, since the old users don't possess the new random values to correctly update their keys. This comes at the price of a reduction in both the statelessness of receivers and security, as will be explained and further discussed in the following sections.

- The second mechanism, explored as a solution to security related issues which arise with the first, is based on the provision of a number of updated private keys inside each smart card. Instead of relying on the broadcast of a value γ_1, each user will be provided with values $d_i^t = g_i^{\gamma_t}$ for $t = 1, \ldots, m$ and mT_{min} the duration of the subscription period. This way the current random value γ_t used by the protocol will be protected by the discrete logarithm problem, and the update can be scheduled without need of any broadcast message. With this solution T_{priv}

These two solutions will be discussed further in this section; for now let's just assume that the scalability issues of the schemes can be solved at least in part thanks to a periodic update of the private keys of receivers.

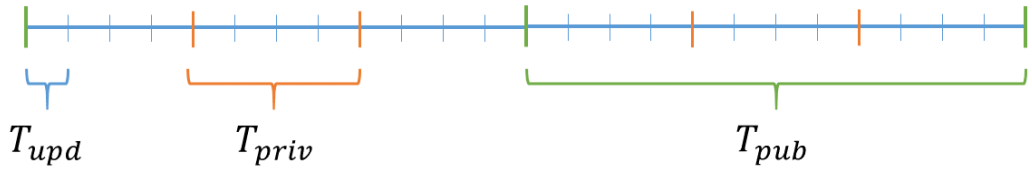


Figure 3.3: Periods for the three different rekeying operations.

3.1.2 Authentication

Even though the main focus of this thesis is on encryption, authentication has an important role, too. Every key management message should indeed be authenticated to prevent attackers from impersonating the system and sending forged messages to clients. In general the attacks against a signature scheme are:

1. **key only attack**: where the adversary only knows the public key;
2. **known message attack**: where the adversary also has a list with a certain number of signature-message pairs;
3. **chosen message attack**: where the adversary can choose a number of messages and get the system to sign them.

Where the adversary grows more powerful from number 1 to number 3. The goals of the attacker can be ordered by decreasing damage to the system:

- **Total break**: the attacker aims at obtaining the private key for signatures forgery;
- **Selective forgery**: prior to the attack the attacker will choose a forged message m; then he will access the public key used for verification and be able to provide a correct signature s for m with non-negligible probability;
- **existential forgery**: the attacker will be able to produce a valid signature for some message of his choice, possibly depending on the public key and the observed message-signature pairs.

In general most signature schemes in literature aim at providing protection against the least difficult attack carried out by the most powerful adversary, being *existentially unforgeable against chosen message attack*. The first definition of this concept was provided by Goldwasser in [29].

On the other hand, since the focus is on the broadcast of key management messages, and users can hardly influence the future content of such messages, it would be enough to consider *existential forgery against known message attack*, where attackers can only collect a certain amount of message-signature pairs. A key management system which provides this kind of protection with shorter signatures could in principle be enough.

RSA digital signature scheme is based on the problem of integer factorization. The keys are generated as follows:

- Randomly select two large primes p and q, and compute $N = p \cdot q$;
- Compute $\varphi = (p-1)(q-1)$;
- Choose an integer e such that $1 < e < \varphi$ and $\gcd\{e, \varphi\} = 1$;

- Compute d such that $1 < d < \varphi$ and $e \cdot d = 1 \pmod{\varphi}$;

The public key is e, while the private key is d.

When the signer needs to generate a signature for a message m, this message is hashed to an integer modulo N, $H(m)$. The signature is computed as $s = H(m)^d \pmod{N}$ and sent together with the message.

Verification is performed with the public key: receivers compute $v = s^e \pmod{N}$, then they obtain the hash of the message m and compare v with $H(m)$; if they are equal the verification is successful.

Since the RSA cryptosystem was proposed, in 1977 by Rivest, Shamir and Len Adleman, many cryptographic attacks have been discovered that mine the security of schemes based on the RSA problem. Some of these attacks are due to a misuse of the protocol, while some others can be successful when the private or public keys are too small. The fastest known brute-force algorithm which aims at factoring N is called General Number Field Sieve, and it takes subexponential time in n, the number of bits of the modulus. For these reasons it is best practice to select $n = 1024$ bits at least, which will also be the smallest secure signature which can be obtained with RSADSA, and it corresponds to 80 bits of security (see [30] for a complete discussion).

Since the aim is reducing the amount of bits as much as possible, more efficient signature schemes should be investigated.

DSA was proposed by NIST in 1991, and it is based on the intractability of the discrete log problem in prime order subgroups of $\mathbb{Z}_p^*$. The parameters of the scheme are: a 160-bit prime q, a 1024-bit prime p such that q divides $p - 1$, and g, a generator of the unique cyclic group of multiplicative order q contained in $\mathbb{Z}_p^*$.

Key generation proceeds as follows: a random integer x is selected ($1 < x < q - 1$), and $y = g^x \pmod{p}$ will be the public key, while x is the private key.

The **signature** of a message m can be carried out as follows, the value $e = H(m)$ is computed, and a random k ($1 < k < q - 1$) is selected. The algorithm proceeds with the computation of $X = g^k \pmod{p}$ and $r = X \pmod{q}$ (if $r = 0$ the algorithm should be restarted). Then the sender computes $s = k^{-1}(e + xr)$ (if $s = 0$ the algorithm should be restarted). The signature corresponds to (r, s).

Verification is carried out as follows: the receiver computes $e = H(m)$, $w = s^{-1} \bmod q$, $u_1 = ew \bmod q$ and $u_2 = rw \bmod q$. Then the $X = g^{u_1} y^{u_2}$ can be computed, eventually obtaining $v = X \bmod q$. Verification is successful if $v = r$.

DSA allows to have a much shorter signature: it only requires 320 bits.

This scheme relies on two discrete logarithm problems [31]; the first is the discrete logarithm problem on $\mathbb{Z}_p^*$, where the Number Field Sieve algorithm applies: if p is a number of 1024 bits then the expected amount of time required by this attack is too high to represent a threat (for now). The second discrete log problem is that of determining k given $g^k \bmod p$, where g has multiplicative order q. The most efficient known algorithm to solve this problem is the Pollard rho algorithm, which, for large p, takes about $\frac{\sqrt{\pi q}}{2}$ steps. In order to make DSA resistant to this attack, it is best practice to choose q as a 160 bits number (to provide at least 80 bits of security).

ECDSA is a digital signature scheme which is very similar to DSA, but relies on the hardness on the discrete logarithm problem over elliptic curves, which are plane algebraic curves defined by the equation $y^2 = x^3 + ax + b$. The signature schemes uses elliptic curves defined over a prime field $\mathbb{F}_p$ or $\mathbb{F}_{2^m}$. The scheme signature, as in DSA, is composed of two elements of n bits each, with n the order of the generator point in the elliptic curve. Similar considerations on the security of the scheme guide the choice of n towards a prime number of at least 160 bits in order to take into account the Pollard rho algorithm and the Pohlig-Hellman algorithm against the discrete logarithm problem in elliptic curves [31].

In RSA, DSA and ECDSA the length of the signature is mostly due to security considerations on the intractability of the discrete logarithm problem. For this reason defending the system against known message attacks instead of chosen message attacks would not allow to reduce the size of signatures nor to gain anything in terms of bandwidth. This is true not only for these specific schemes, but for all signature schemes which are based on the same problems. If it's true that the size of the key groups can't be reduced without exposing the system to attacks, it is still possible to find schemes which use a more efficient signature, containing a number of bits corresponding to less than two elements of that group. This is the case of ETA (efficient and tiny authentication), a scheme which is based on the discrete logarithm problem on elliptic curves [32]. ETA allows to have a signature of 240 bits while maintaining the same level of security (80 bits); however, this is a k-times signature schemes, meaning that it requires the verifiers to store a public key which is linear in the number of messages the system wishes to sign. This might be a heavy requirement on the receivers, especially since the public key could ideally last for a long time. As Boneh argues in [33], some variants to DSA have been proposed in order to reduce the length of digital signatures to approximately 240 bits without decreasing security [34] [35]. In [33] Boneh proposes a new signature schemes based on bilinear pairings which allows to obtain signatures of approximately 170 bits. This seems currently one of the most promising results, since for the reasons presented in the previous discussion it is very hard to build a scheme based on groups of size lower than 2^{160}.

BLS digital signature scheme

This digital signature scheme uses groups where the computational Diffie-Hellman problem is hard, while the decisional Diffie-Hellman problem is easy. This scheme has been proven to be existentially unforgeable under chosen message attack in the random oracle model.

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of order p, with generators g_1 and g_2 respectively. Let e be a bilinear map $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The existance of an isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ is necessary for the security proof (if $\mathbb{G}_1 = \mathbb{G}_2$ the isomorphism can be the identity).

Decisional Diffie-Hellman: Given g_2, $g_2^a \in \mathbb{G}_2$ and h, $h^b \in \mathbb{G}_1$ output "yes" if $a = b$ and "no" otherwise. If the answer is "yes", then (g, g^a, h, h^b) is called a co-Diffie-Hellman-tuple.

Computational co-Diffie-Hellman: Compute $h^a \in \mathbb{G}_1$ given g_2, $g_2^a \in \mathbb{G}_2$ and $h \in \mathbb{G}_1$.

A pair of groups $(\mathbb{G}_1, \mathbb{G}_2)$ where co-DDH is easy, but co-CDH is hard is called Gap co-Diffie-Hellman.

As Boneh argues, the fact that GDH groups might be the starting point for the implementation of digital signature schemes had already been observed by Okamoto and Pointcheval, even though most of these groups will probably lead to long signatures. The proposed scheme is the following:

Key generation: Choose a random $x \in \mathbb{Z}_p$ as the private key. The public key will be $v = g_2^x$:

Signature: To sign a message m first compute the hash $h = H(m)$, then calculate the signature of the hash as $\sigma = h^x$;

Verification: To verify the signature σ of a message m, compute the hash $h = H(m)$, then check whether (g_2, v, h, σ) is a valid co-Diffie-Hellman tuple;

The paper further discusses how to choose a hash function, and how to obtain an elliptic curve, listing a number of curves which are suitable for the scheme.

It is important to notice that the BLS signature scheme provides 80 bits of security, and it is still an open problem to find appropriate elliptic curves to allow for a higher security level.

An interesting signature scheme which according to [36] would provide post-quantum security is the Rainbow signature scheme. The length of the signatures this scheme can provide is of 264 bis for a security level of 128 bits, which is a great improvement with respect to ECDSA. Reducing the size of the signature comes at the price of much bigger public and private keys, which have size of 103 and 67 kilobytes, respectively, with respect to the 768 and 512 bits for ECDSA at the same security level [36].

According to [37] the reference security parameter for the near future is 128 bit long, which corresponds to ECDSA signatures of 512 bits. In [38], which contains key management recommendations from NIST, it is stated that 80 bits of security are not enough for protection of federal government applications, while 112 bits of security will be enough until 2030. Since the scope of this thesis regards Galileo's commercial service, which will probably cover a longer period of time, it seems better to provide 128 bits of security, which according to [38] will be enough after 2031. A few digital signature schemes have been proposed literature which can reduce the communication overhead, improving performances with respect to ECDSA. However, since the scheme has been developed with a specific application in mind, it seems natural to evaluate its performances in a more practical manner, taking into account approved signature algorithms, as ECDSA with 128-bits security.

3.1.3 Random revocation

Before describing the protocol in detail, some assumptions must be made on the cause of random revocation. There are two main reasons why a receiver might need to be revoked:

- The system has been exposed and it is in danger because some keys have been corrupted. In this case the system might want to revoke a group of users who are suspected to be guilty;
- Some users are misbehaving (e.g., they are not paying the subscription fee).

It would be ideal for the system to have the ability of revoking single users, but if the number of users is too high the set identification problem will become an important issue: Fig. 1.2 shows that the number of bits require to identify k users in a set of n rapidly increases with both n and k, limiting the revocation capabilities of a system with very low bandwidth. A solution might be that of limiting revocation capabilities to group random revocation instead of single user revocation. This paradigm would allow the system to limit the damage to the other user groups when one or more groups have to be revoked. The idea of "group of users" is very general and can vary according to applications and needs. As an example, a user group could contain:

- All the receivers of a certain client (e.g., the client might be a company which buys the service for a number of its receivers);
- All the users with a certain level of "trust" on the basis of reputation or the security of the receiver device;
- Any other type of users a service provider might decide to consider as a single entity for revocation purposes;
- Users legally registered in the same geographical area.

The concept of grouping receivers together in order to devise feasible random revocation algorithms aims at obtaining a system with a limited number of independent clients, in order to reduce communication overhead. All the revocation schemes which can be found in literature suffer from what is called here *set identification* problem. None of the analyzed schemes allows to revoke random entities without specifying their "position" inside the structure of the scheme. The reason can be easily inferred: let's assume the existance of a scheme which allows to arbitrarily choose any set $\mathcal{S}_r$ of users and revoke them without broadcasting any information about the set. Revocation implies that all non-revoked users must be able to find out the new group key, while no information must be made available to revoked users. This consideration implies that the group key must depend on $\mathcal{S}_r$, which must be known to the users. Since in random revocation $\mathcal{S}_r$ can not be known a priori, the only way to communicate it to the users is via broadcast.

Under these consideration, from now on the performances of the revocation schemes will be derived with respect to N_g, the number of groups inside the system, instead of n, since random revocation will be carried out at user level.

As shown in Fig. 1.2, the length of the set identification message varies according to:

- The number of groups in the system;
- The number of revoked groups from the start of the protocol;
- Possibly the position of the revoked groups inside the linear structure. Two are the most suitable source coding approaches to code the message: coding the number of revoked users k, and all

possible combinations of k revoked groups in a total of N_g, which takes a fixed amount of $\log_2 \binom{N_g}{k} + \log_2 k$ bits; or coding runs of 0 in a message of N_g bits, where "0" means that a group is revoked. This second approach might be preferred in case that the position of revoked groups form big runs inside the set identification message.

In general it is not possible to know in advance how many groups the system will need to revoke and how big the set identification message will be. Certainly even with the group oriented approach, revoking up to half of the groups might still lead to a very big overhead: with 1000 groups, revoking 500 will take $\log_2 \binom{1000}{500} + \log_2 500 = 1004$ bits.

Fig. 3.4 represents the worst case communication overhead for revocation of k of the N_g groups.

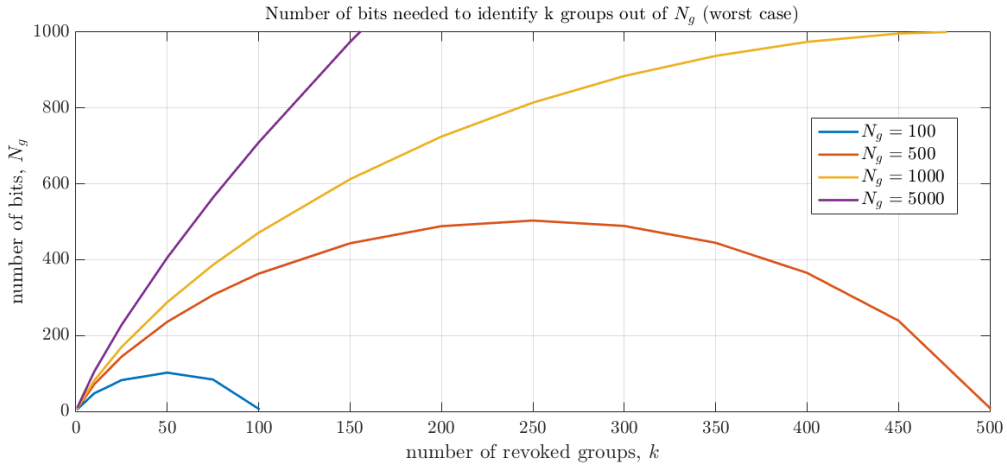


Figure 3.4: Dimension of set identification message for group based random revocation.

Since Fig. 3.4 represents the worst case, a solution could be that of allowing a maximum amount of bits, $B_{\max}$ devoted to set identification messages for every service provider. When random revocation is not necessary, there will be no set identification message, and the broadcast of regular key management messages will be faster. On the other hand, should a provider need to revoke some of its groups, it will be able to decide the new group configuration, provided that the set identification message is below the maximum size $B_{\max}$. If a service provider needs a bigger set identification messages the solutions are two: either it will exceed the bounds at its own expenses, causing its receivers to experience higher latency in decoding, or it will need to devise a different pattern in the set identification, obtaining long runs of zeros and ones and thus a shorter set identification message thanks to run length encoding. This result can be achieved by revoking groups that are close in the linear structure, which might require to revoke some of the authorized groups along with the misbehaving ones, or the opposite: allowing some groups to stay in the system if the loss of revenue that derives from this choice does not affect the system too deeply.

Another possibility that would guarantee a longer lifetime to the scheme is partially limiting the

statelessness of receivers, which means that the system will assume that once every T_{on} every receiver will switch on and decode the current key management messages from the broadcast channel. Such an assumption would allow to stop broadcasting information about the revocation of those groups which were revoked more than T_{on} ago. This period T_{on} can be chosen according to the types of subscription the system allows: for example, if subscriptions are based on monthly time units with a maximum of one year, it is reasonable to assume every user will switch on at least once a month, and that might be T_{on}. Of course users which for some reason do not respect this minimum switch on period will still be able to decode, provided that they retrieve revocation information via another aiding channel (which can be broadcast and does not need to be secure). Another way to recover space in the set identification message comes from the natural flow of time: the keys whose life time was prematurely ended by random revocation will eventually expire anyway, and will not need to be forced out of the scheme after their expiration.

Chapter 4

Protocol implementation of the proposed scheme

RevHash will be further discussed in this section, where two *modes* will be defined. Each mode corresponds to a different state of the system, and has a different broadcast transmission. For the reasons previously described the focus will be on group revocation instead of single user revocation; in the case where the number of users is small and the overhead due to set identification messages can be tolerated, user revocation can allow a better control over the system.

4.1 Regular mode

This is the mode the scheme will hopefully adopt for most of the time, which corresponds to the normal dynamics of the subscription based system, where users enter and leave according to their subscription time, known a priori. Let's define some parameters:

- N_{TS} represents the total number of *time slots* inside the scheme. A time slot is a "segment" containing a certain number of private keys: all the receivers linked to those keys will be removed from the system at the same time, specified by the position of the time slot in the time line. Time slots force a linear structure and a temporal behavior on the scheme: after the start of the protocol time slots will be progressively consumed, meaning that the users linked to their keys will no longer be privileged receivers. In the meantime, users keep entering the system. This means that if the maximum subscription period lasts T_{max}, which contains N_{TS} time slots of duration T_{min}, a scheme with just N_{TS} time slots will not be enough. The system needs to be able to provide access to users even at the end of the first T_{max}, when most of the time slots are consumed. The idea is to have two segments with N_{TS} time slots each, so that the second segment can be filled right after the first. After a period of T_{max}, the private keys will all be updated thanks to a private key update mechanism (which will be discussed further). The updated keys belonging to the old segment can be given to new users, since the former values of these keys

can not be used for decryption anymore. This mechanism allows to support autonomous users, at least for a maximum period of $T_{\max}$. Fig. 4.1 and 4.2 represents this concept.

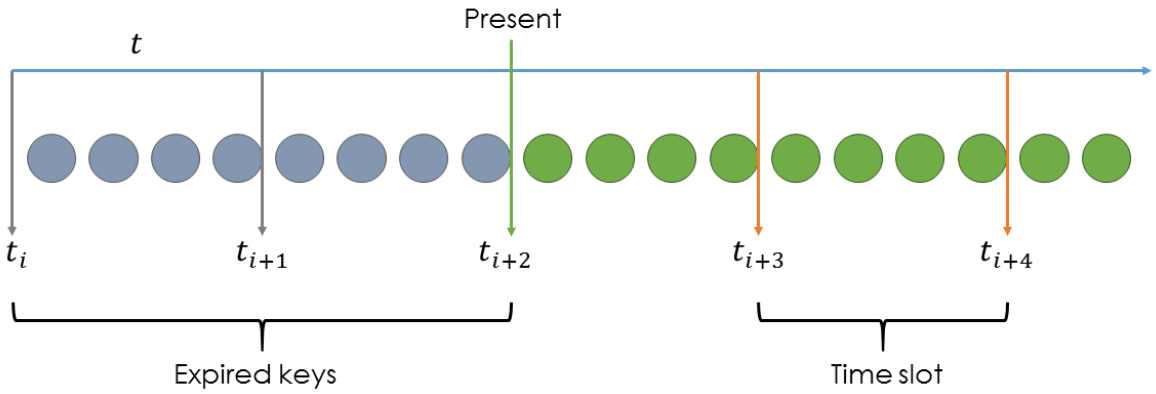


Figure 4.1: Simple idea for the organization of the keys.

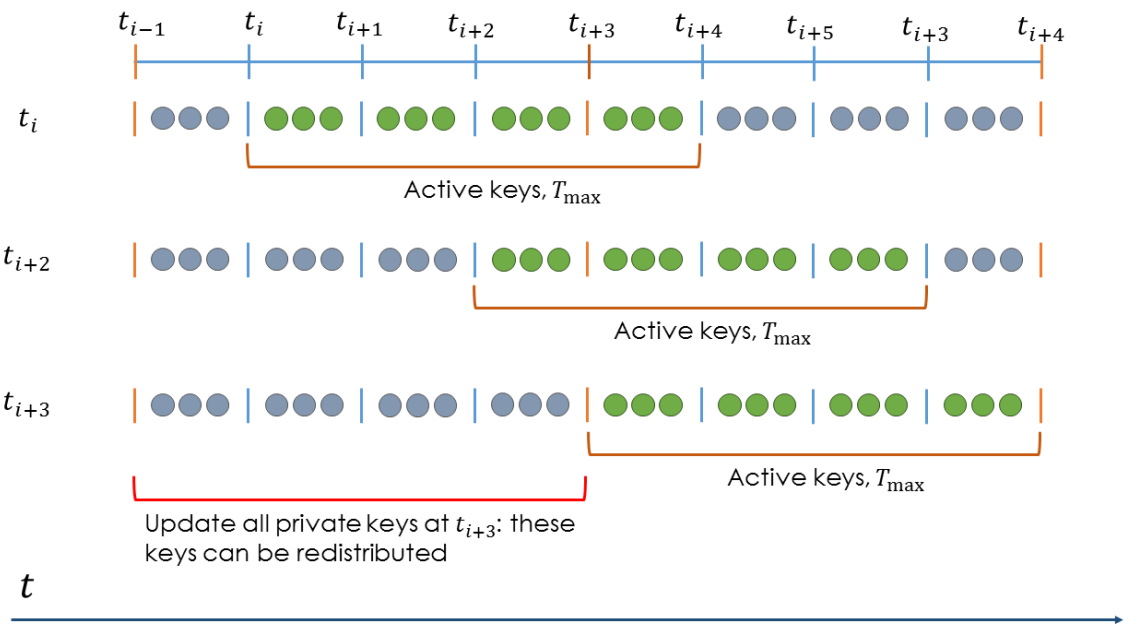


Figure 4.2: Simple idea for the scheduling of the scheme in the regular mode.

- An interesting idea could be that of using different "scheme segments" for different subscription lengths. Some users might only need to subscribe for a few weeks, while others might prefer a longer subscription (a few months). One segment of the linear structure could be used to distribute keys lasting less than a month (daily time slots), while the time slots in the rest of the structure will have monthly duration. In general, let's define as $T_{\min}$ (e.g., a day) the shortest

time slot and let's call the two segments described above as *short term* and *long term* segments; T_{med}, multiple of T_{min}, represents the duration of each time slot in the long term segment (e.g., a month). The short term segment will be consumed much faster than the long term one, but the time slots in each segment will still be ordered in a timely manner. As discussed before, the short term segment will be composed by two identical portions, each containing $n_{\text{min}} = \frac{T_{\text{med}}}{T_{\text{min}}}$ slots. The long term segment will contain $n_{\text{med}} = \frac{T_{\text{max}}}{T_{\text{med}}}$ slots, since every n_{med} the private keys value will be updated, allowing the recovery of one of the two portions of the short term segment, and one of the time slots in the long term segment.

- N_{g} represents the number of logic groups inside each time slot. The choice on the number of groups is left to each service provider, according to a trade-off between revocation capabilities and communication overhead. Each time slot shall contain N_{g} keys, one for each logic groups, plus a number of spare keys βN_{g} private keys, which will be used in case of revocation to reintroduce the authorized users in the system, forming a new group with a new private key. The value of βN_{g} depends on the expected dynamics of the system, but in general $\beta = 1$ should be more than enough, allowing to revoke and reintroduce N_{g} groups before rekeying. Fig. 4.3 shows the structure of the scheme.

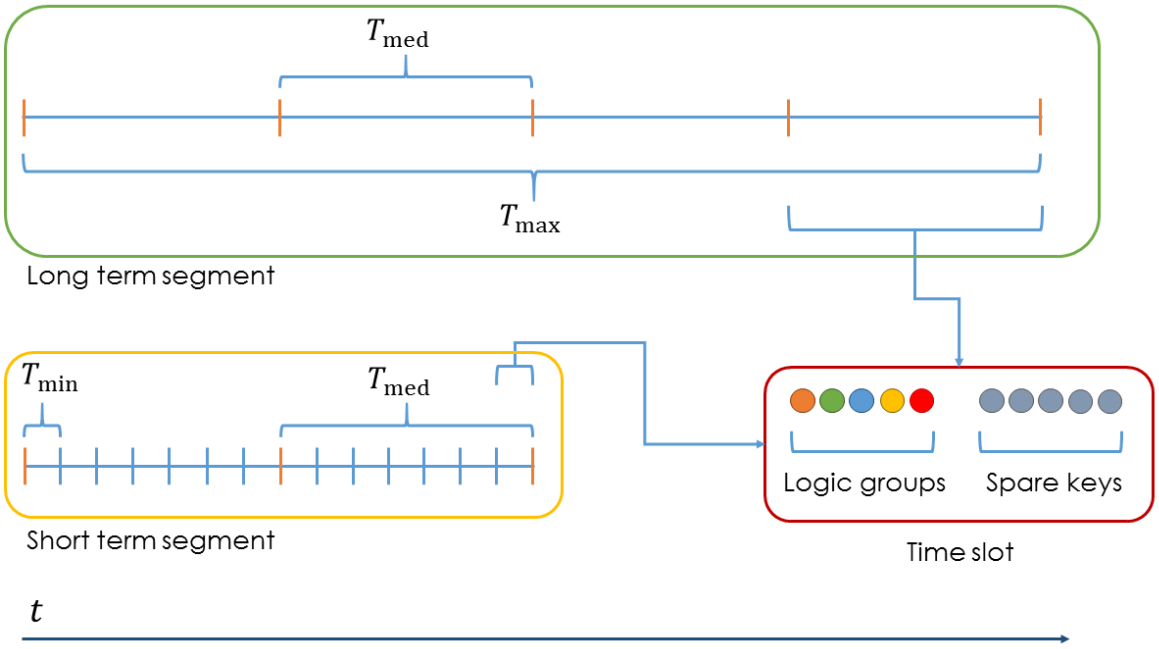


Figure 4.3: Structure of the scheme.

In the regular mode the system does not need to identify the expired users, since they are already known: receivers are programmed to automatically exclude expired keys, according to the scheduling of the system. As soon as T_{min} elapses, the new group key seed will be delivered in the form of the

1024-bit header of the revocation system (512 bit for group key recovery and 512 bit for ECDSA signature). The scheme automatically excludes from the privileged set all keys from the expired time slot, and there is no need for additional overhead. The group key used to encrypt data is updated through a one-way function, allowing all users who received the group key seed to decode for a duration of T_{min}.

The scheme will have $(N_{\text{g}} + \beta N_{\text{g}})2\left(\frac{T_{\text{max}}}{T_{\text{med}}} + \frac{T_{\text{med}}}{T_{\text{min}}}\right)$ user keys, which, for $T_{\text{min}} = 1$ day, $T_{\text{med}} = 1$ month, $T_{\text{max}} = 1$ year, $\beta = 1$ and 500 logic groups gives 74000 user keys, which results in a public key of roughly 5 megabytes that can be stored in non-secure memory.

4.2 Scalability issues

The main concerns when it comes to scalability are the following:

- The linear structure of the system has also a temporal meaning: older slots are progressively consumed and the keys associated with those slots are excluded from the privileged set. The main problem is that those keys can never be reintroduced in the privileged set and distributed to new clients, because this might allow revoked clients who manage to break their old device to decrypt new messages.
- If the system is highly dynamic (i.e., users stay inside the system for relatively short time), keys will be consumed at a high rate, and over-dimensioning the system might not be enough.
- Once the parameters of the system have been established, the number of private keys is fixed and can't be modified without requiring to distribute new information via an aiding channel.

4.2.1 Naive solution: broadcast of a private key seed

Since the scalability problem is a major issue for a subscription-based system, different solutions have been investigated: a first idea consists in the broadcast of a random value γ_1, encrypted with the current group key K_G, which allows to update the private keys inside the tamper resistant device of every user (as explained in section 3.1.1). Every $T_{\text{upd}} = T_{\text{med}}$ a new random value is cyclically broadcast, and the update of the private key with the new value γ_1 will take place starting from the following period T_{med}. This means that since every new seed is encrypted with the current group key and its transmission is repeated for a duration of T_{med}, the minimum switch on period for receivers is T_{med}, after which the private key seed must be retrieved from a different channel. The long term segment will have to contain another slot, which is always idle. There is a trade-off between the frequency of the private key seed broadcast (the maximum period for a receiver to switch on) and the dimension of the system. The smallest the former, the biggest the system needs to be, leading to a bigger number of time slots and consequently of private keys. The increase in the number of private

keys has the result of increasing the dimension of the public key, and consequently of the non-secure storage requirements for each user.

The consequences of this key update mechanism are the following:

- users who will be revoked next T_{med} will know the next random value for private key update. This, however is not a major problem, since the revocation system will exclude automatically their expired private keys;

- privileged users with long subscriptions will have to connect to the broadcast channel at least every T_{med} in order to retrieve the new value for the private key update. This limits the statelessness of receivers, and it could be a hard requirement for certain categories of applications.

- security issues arise from this mechanism: The private key seed must be encrypted to keep it safe from former users, and its aim is to update ciphertext and private keys used for the transmission of the group key at a specific time slot S_i. Then it must be transmitted in advance, at least during slot S_{t-1} (encrypted with the current group key), so that it can have immediate effect at S_t. If a user with long term subscription gets revoked and leaks the seed to the public before the end of his subscription period, it might be possible for former users to update their old private keys and be able do decrypt messages as privileged users. This possibility not only mines the collusion resistance of the scheme, but it also makes it impossible for the system to fix the problem without a complete rekey of all groups.

  In general, the original scheme is in danger only when privileged users break their devices and keep leaking the group key seeds at every time slot: this is the only situation in which the scheme might be considered as "completely broken" and fixable only by a complete rekeying. The private key update mechanism introduces an element which is known and shared by a big number of users (all privileged users). If the private key seed is leaked by revoked or even privileged users, the system will be completely broken as well, since it's impossible to heal the scheme just through the revocation of a small amount of users.

For this reason the private key update mechanism via broadcast has been discarded, but the problem of scalability requires a new solution, possibly avoiding the distribution of a value common to all users which allows everyone to update their keys in the same way.

4.2.2 An alternative solution: multiple private keys

Let's sum up the ideas on how to solve the problem of scalability of this key management scheme:

- The private keys need to be updated before reusing them (distributing them to new user groups), and the old ones must become useless for decryption purposes;

- The private keys have form $d_i = g_i^{\gamma}$, and they are linked to the key management ciphertext through the value γ, randomly selected by the system, which appears in the factor $v = g^{\gamma}$ in the header, as explained in section 1.9. This means that if the value γ can somehow be updated to $\gamma^{'}$ (as explained in 3.1.1) during the lifetime of the scheme, every privileged user must in the end have the same value of $\gamma^{'}$ in order for the system to work;

- The only way for the system to recover space is to schedule some private key updates (e.g., every T_{med}) in which the value of γ will be updated in a safe and communication-efficient way;

- transmitting or storing values of γ inside the smart cards weaken the protocol, increasing the probability of system compromise.

Instead of broadcasting an update value γ^{upd}, which is dangerous for the reasons discussed above, every receiver shall be provided in advance with $m = \lceil T_{\text{sub}}/T_{\text{med}} \rceil$ private keys of this form:

$$\{d_i^t, \ldots, d_i^{t+m-1}\} = \{g_i^{\gamma_{S_t}}, \ldots, g_i^{\gamma_{S_{t+m-1}}}\} \tag{4.1}$$

At the beginning of the t-th period of length T_{med} the system will set $d_i^t = g_i^{\gamma_{S_t}}$ for every i, and it will use the value $v = g^{\gamma_{S_t}}$ as a factor in the key management header (see section 1.9).

This solution offers the same security as the original key management scheme, allowing for a major improvement in scalability at the price of a limited increase in secure storage requirements. The additional amount of secret material is m times the length of a private key, which is 256 bits for 128 bits of security. The structure proposed for the scheme allows to smartly organize subscription and have small minimum subscription time and high maximum subscription time without requiring big amounts of memory. As an example the scheme could provide $T_{\text{min}} = 1\text{day}$ and $T_{\text{max}} = 1\text{year}$, allowing subscriptions of one day granularity at the cost of $12 \cdot 256 = 3072$ bits of secure memory storage, which is a very small amount.

The aspects which make this solution secure and efficient are the following:

- The difficulty of the discrete logarithm problem over finite fields makes it hard to find out the value of γ from the knowledge of g_i^{γ} and g_i. The previous mechanism required the value of γ to be stored in the clear inside the tamper resistant device at some point, making it vulnerable to users who might break it. This method, on the other hand, directly stores the updated values of the user's private key: if a user manages to break the device and leak these values, once the system authorities find out, the system can still be restored by revoking the misbehaving client group and its leaked keys, without any consequence on the integrity of the scheme.

- Knowing the updated keys in advance will not prevent the system from performing random revocation on groups: in random revocation mode the group private key gets revoked and a new one is distributed only to the rightful users.

- No broadcast communication is required, since all the key material is installed inside the smart card.
- There is no need to limit the statelessness of receivers, requiring that they switch on after a certain period of time: all the information needed to access the service is either stored inside the smart card or transmitted via broadcast at any time.

For these reasons this key update mechanism has been selected as the most suitable and convenient for the improvement of the scalability of the scheme.

This private key update scheduling is represented in Fig. 4.4.

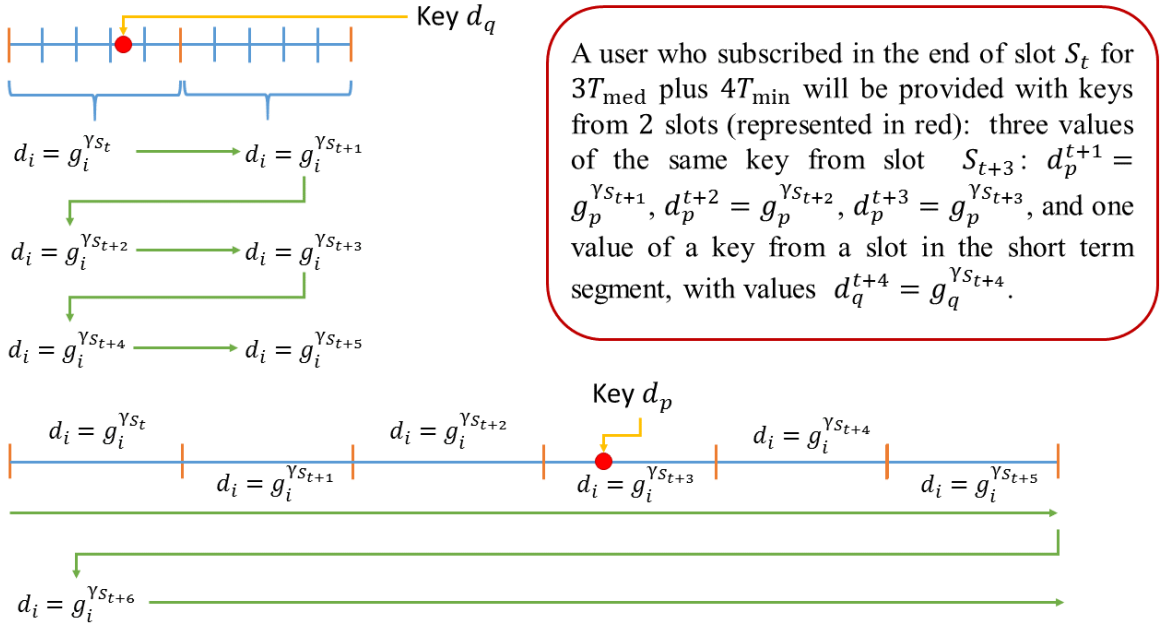


Figure 4.4: Private key update scheduling.

Note that in the short term segment subscription expiration is performed with a different mechanism with respect to the long term segment: in the former the expired keys are simply excluded by the system, which controls the access through the broadcast message:

$$H = \left(g^q, \left(v \prod_{j \in \mathcal{S}} g_{n+1-j} \right)^q \right) \in \mathbb{G}^2.$$

with q chosen at random, where only users who belong to the authorized set $\mathcal{S}$ are included in the product in the second term. In the long term segment all keys are included in the scheme, and expiration comes from the fact that only authorized users possess the private key with the correct

value of γ. With the key distribution mechanism explained above, indeed, every user is provided with as many "versions" of their private key as they need for the duration of their subscription period. Let's say user i has keys $\{d_i^t, \ldots, d_i^{t+m-1}\} = \{g_i^{\gamma_{S_t}}, \ldots, g_i^{\gamma_{S_{t+m-1}}}\}$. At the time slot S_{t+m} the scheme will distribute header

$$H = (C_0, C_1) = \left(g^{q_{t+m}}, \left(v_{t+m} \prod_{j \in \mathcal{N}} g_{n+1-j} \right)^{q_{t+m}} \right) \in \mathbb{G}^2.$$

where $v_{t+m} = g^{\gamma_{S_{t+m}}}$. In order to obtain the value which, by hashing operations, will allow to retrieve the first group key of the chain, the user (whose index is now included in the product of the second term, C_1), has to compute:

$$K_G = \frac{e(g_i, C_1)}{e\left(d_i^{t+m} \prod_{j \in \mathcal{N}, j \neq i} g_{n+1-j+i}, C_0\right)}$$

Note that, as shown in the proof of correctness, the only way for the user to retrieve the group key is using the private key with the value of γ the system is using. Indeed the term $d_i^{t+m} = g_i^{\gamma_{S_{t+m}}}$ will allow the retrieval of the group key only if the γ at the exponent is the same as the one of the factor v_{t+m} in the term C_1. If the user does have this value inside his device, it will be impossible for him to retrieve the new group key. The validity of this statement is proved in [18], where this approach was suggested for mailing list applications, where different lists use different encryption schemes with the same public key and different values of γ. Applied to this revocation scheme in a key renewal mechanism, this idea allows to obtain good scalability, improving the performances of the original scheme, *BiMaps*, whose scalability is very poor.

4.3 Random revocation mode

As explained in section 3.1.3, the scheme offers random revocation capabilities to some extent, which depends on the number of logic groups the service provider decides to fit in the scheme. A reasonable solution is guaranteeing to the service provider a maximum amount of bits for random revocation, which shall be used according to its own needs. For example, let's consider 500 logic groups, with the same amount of spare keys for reintroducing revoked users in the system. A maximum set identification message of 1000 bit will allow to revoke up to 500 groups throughout the lifetime of the scheme, *in the worst case*. The expression *in the worst case* is used for two reasons:

- First of all, if the revoked logic groups form a pattern which allows for efficient run length encoding, then the system might have the possibility of revoking even more users;

- After a long enough time (precisely: after the natural expiration time of the revoked keys), the system might decide to reintroduce some of the revoked keys: the leaked keys are indeed relative to values of γ which are not used anymore, and can thus be updated by the system and assigned to new user groups.

- The system might decide to partially limit the statelessness of receivers, assuming that once every kT_{med} every user will have switched on at least once, retrieving the set identification message. This would allow to reset those bits of the set identification message corresponding to users revoked long before. If some of the users did not receive the message on time, they will not be able to retrieve the group keys anymore, and they will have to connect to an aiding channel in order to retrieve the old set identification messages.

Should the system find itself in a situation where the keys are almost consumed, or the set identification message is becoming too heavy, a rekeying should be performed.

The system is in random revocation mode whenever a set identification message for random revocation is currently broadcast. In addition to that, the usual key management message of 1024 bits will be broadcast as in the regular mode. In order to ensure the continuity of service for those users who are using the service at the moment when random revocation takes place, the system might use an alert flag to inform users that a change in the normal scheduling of membership has been made, and the group key will be changed after a certain amount of time, in order to allow the currently active users to retrieve the next group key without interrupting their access to the service.

4.4 Communication overhead

The overhead of the scheme will now be investigated in order to have insight on its efficiency.

In the regular mode the key management message, which allows to retrieve the group key seed, contains two elements of a bilinear group $\mathbb{G}$. According to [39], acceptable lower bounds for bilinear maps implemented with Weil pairings are 1024 bits for the elements of $\mathbb{G}_1$ (as K_{rev}) and 160 bits for elements of $\mathbb{G}$ (i.e., the group of the private keys and the elements of the header). However, since 128 bits of security are considered an adequate security level for the near future, as stated by NIST and ENISA in [38] and [37], it is more appropriate to consider 256 bits for the elements of $\mathbb{G}$ and 3072 bits for the elements of $\mathbb{G}_1$. The group key K_G^{rev} which can be obtained by authorized users is 3072-bit long, but it could be hashed by SHA-256 to obtain a 256-bit key, which is K_G^{first}, preserving the 128 bits of security. The other group keys of the chain will be obtained through the same hash function, SHA-256, applied multiple times: the last key obtained through the one-way chain will be the first to be used as data-encryption symmetric key. Each of these keys will provide 128-bits of security. As stated in [25], the length of the hash chain, which is used to provide symmetric keys with shorter cryptoperiod, must be investigated in order to keep under control the entropy reduction along the one-way chain. There is a trade-off between the cryptoperiod of the group key and the length of

the one-way chain: a shorter cryptoperiod would improve security, but if the one-way chain is too long, there could be major security losses.

Under these considerations, the length of the key management messages in the normal mode is 1024 bit, while in random revocation mode, the set identification message must be added to the count. Let's consider a maximum of 1024 bit for this message. These messages can and shall be sent in the clear, since encryption would only create problems: if the message which is supposed to deliver the group key is encrypted with the previous group key either receivers will have to switch on once every $T_{\min}$ or older group keys should be broadcast for a long enough time.

Key management messages will be sent in the clear, and they will need to be authenticated. As previously discussed, authentication will also be performed through elliptic curve cryptography, and in particular 256-ECDSA with signature of 512 bit length (two elements of a group of order 256).

How should messages be signed when more than one scheme (i.e., service provider) is present? Let's call N_p the number of service providers who share the broadcast channel and have their own independent scheme which needs authentication. There are two opposite approaches for authentication:

- Authenticate all messages from all service providers with a single signature. This allows to have a small overhead of 512 bits independently of the number of service providers. The main drawback is that verification will require the correct reception of all messages of every single service provider, plus the signature, giving a total of $N_p \cdot 512 + 512$ (up to $N_p \cdot 1512 + 512$ in revocation mode) which might be quite demanding. Moreover, service providers will share the same authentication key, and they would need to communicate with each other when the ECDSA key needs to be changed. Since receivers would need to decode all key management messages for the purpose of authenticating their own service provider, decoding performances would strongly depend on the length of the messages of the other providers (at least in revocation mode).

- Use a separate signature for each service provider. This allows verification to be independent of all other services, but the overhead increases linearly with the number of service providers. If the service providers are not many, this could be the best solution, allowing for complete independence also for what concerns the change of authentication keys. Moreover, receivers would only need to receive their own key management message (512 bits in regular mode), plus a 512 bit signature: a shorter message takes shorter time to be correctly received.

Note the difference with the two extreme approaches: the first would force the receivers to retrieve $512 + 512 \cdot N_p$ bits, plus additional revocation overhead, introducing only 512 bit overhead for authentication. The second approach would allow the receivers to retrieve the minimum amount of bits (only those of the key management message they need to authenticate), introducing $N_p \cdot 512$ bits for independent signatures.

Other approaches might be devised that lie in the middle between the two. As an example, each key management message might be independently hashed by SHA-256, and each of the hash value

could be broadcast. Then the concatenation of all hash values could be signed. This approach requires users to receive $N_p \cdot 256 + 512 + 512$ bits for verification in normal mode (N_p hash values, the key management message and the signature), with a total overhead of $N_p \cdot 256 + 512$ for authentication purposes.

Chapter 5

Application to Galileo

As reported in [40], when Galileo was conceived, in 1990s, the Commercial Service was thought as a means to build a partnership between the European Union and privates, which would invest in the European satellite navigation system, sharing both the risk and the revenue return this service would provide. However, the added value Galileo would bring with respect to the GPS system remained unclear for several years, and in 2000 it was stated that the project would be fully supported by the Member States of the EU. For this reason the priority went to other services, like the Open Service, for civil use, the PRS, the Safety Of Life and the Search And Rescue service. The first definition of requirements for the Commercial Service brought up some ideas of what could be offered that is not distributed with the OS: a High Accuracy service and an Authentication service. In 2010 the SOL service was re-profiled, and part of the resources allocated to it were freed and made available for the Commercial Service. In 2013 some studies began to investigate how to use those resources (e.g., high data bandwidth), to improve Galileo navigation performances, encourage the development of new services and create added value and a source of revenue for investment recovery.

5.1 Commercial Service signal

The Commercial Service signal has two components, the data component, E6B and the pilot component, E6C. The power of the signal is equally split between these two components, which are transmitted in the E6 band, ranging from 1260 to 1300 MHz. Both components are modulated on the in-phase component of the E6 signal, leaving E6A, the quadrature component, to the PRS service. A peculiar feature of this signal is that the *spreading codes* of both data and pilot components (i.e., the sequences which allow to raise the signal on top of the noise through autocorrelation), can be encrypted. Spreading code encryption allows to limit the access to the signal to the users who possess the encryption key. The features of the CS signal are summarized in Fig. 5.1 [40]. The CS also allows to have different messages transmitted by different satellites, which gives the possibility of improving the performances in robustness and time to decode.

	E6B	E6C
Component	Data	Pilot
Carrier Frequency	1278.75 MHz	1278.75 MHz
Spreading Modulation	BPSK(5)	BPSK(5)
Chip Rate	5.115 Mcps	5.115 Mcps
Primary Code Length	5115 chips	5115 chips
Primary Code Duration	1 ms	1 ms
Secondary Code Length	N/A	100 chips
Secondary Code Duration	N/A	100ms
Symbol Rate	1000 sps	N/A
Data Rate	492 bps	N/A
Data Encoding	As per SIS ICD	N/A
Data interleaving (col. x row)	123 x 8	N/A
Spreading code encryption capability	Yes	Yes
Power sharing	50%	50%
Received Minimum Power (E6B + E6C)	-155 dBW	

TABLE 1. Galileo E6-B/C signal characteristics

Figure 5.1: Features of the CS signal.

5.2 High accuracy and authentication

High accuracy (at a precision of a few centimeters) can be provided with two approaches: Real Time Kinematic and Precise Point Positioning. Galileo CS is well suited for the latter, which is preferred to RTK because it provides global positioning and timing without the need of reference stations. The main drawback it the time it takes to converge to a high enough accuracy, which is in the order of 15 to 30 minutes, while RTK is almost instantaneous.

PPP uses the accurate information on satellite orbits and timing data provided by GNSS services in order to precisely determine the user's position through measurements on the carrier phase. A high enough bandwidth is needed to guarantee precision at centimeter level: clock and orbit correction information must be updated at a higher rate with respect to the other services.

Authentication has becoming more and more important for the protection of GNSS signals from attacks such as spoofing, where fake messages can be built and fed to receivers, with serious consequences for the users. Since the CS allows for spreading code encryption, users who possess the key will be able to perform spreading code authentication. The key for spreading code generation (called *NAVSEC* key) of the pilot component, for example, would allow to directly generate the chips and improve tracking performances; for this reason the key management scheme used to protect the NAVSEC key must be extremely secure: the leakage of this key would allow to spoof the signal until the end of its cryptoperiod.

Authentication often does not need to be performed at a high rate, thus it is not necessary to provide the NAVSEC key directly. A few chips of the spreading code might be enough for some application to verify the authenticity of the signal, without giving away all the spreading code. These chips shall be provided encrypted to the privileged users of an "a priori" authentication service, which Galileo CS might provide. This can be useful to detect spoofing when two signals at different frequencies are involved. If the user knows the relative delay between the two, the authentication chips provided a priori can verify timing information on one of the two components and thus recognize if the other component is out of range. When chunks of encrypted pilot are distributed a priori, a potential leakage must be taken into account: spoofers might benefit from this information and authenticate fake messages. For this reason, a priori authentication material shall be processed inside a security module. Authentication of the commercial service signal can benefit the open service as well: once the chips expire, it is possible to release the encrypting keys in order to allow for "a posteriori" authentication in the OS. Users of the OS might be offered the possibility to authenticate the navigation message for free, provided that this happens a posteriori (and at a feasible rate). Users of the commercial service will be charged for the possibility of performing authentication without latency, or even tracking the pilot.

5.3 Possible uses for the devised scheme

Since the services that will be offered have not been explored already, it is interesting for a candidate key management scheme to be elastic enough to allow the integration of multiple service ideas. For what concerns high accuracy, key management will only take care of the delivery of the hey used to encrypt data. Authentication, on the other hand, can be performed with different paradigms, and directed to different users.

As explained in the previous section, the key management scheme delivers a value from which a group key can be obtained, and new group keys can be generated through a hash chain of desired length. Users who possess the starting value of the chain can generate all group keys for a period $T_{\min}$. In case of authentication, group keys can be replaced with keys that allow to generate the actual chips of the spreading code. The spreading code is a bit stream used for pilot encryption: giving the first key of the hash chain to privileged users will allow them to get all the other keys and generate all the spreading code for tracking purposes.

A different category of users might only need to authenticate the signal every once in a while. In this case a service exploiting the same underlying hash chain can be offered, where only the last key of the chain is distributed, allowing the generation of a small number of chips, high enough (e.g., 10 ms) to perform authentication. This paradigm is interesting because it allows to integrate in the same key management scheme three or more different services: pilot tracking, a priori authentication and even a posteriori authentication, since last key of each chain can be released to the public a posteriori. If the keys for spreading code generation are organized in a hierarchical hash chain structure, similar

to the one represented in Fig. 1.8, then it is even possible to provide users with keys which will give access to a predefined portion of spreading code chips to be used at a certain point in the future. Those values could be either distributed via an aiding channel or charged in the smart cards, in case the devices have a sufficient level of trust. With this solution it is necessary to have an emergency mode (possibly exploiting the whole bandwidth, or a considerable part of it) when security problem arise and scheduled keys must be changed. The proposed key management scheme allows to integrate all these service ideas, allowing a variety of different possibilities for service provision, which can be further defined and selected once the needs of clients become clear.

5.4 Frame structure for broadcast

Sync Symbols	Data Symbols				Total
16	984				1000 symbols
	Page type	**CS data**	CRC	Tail	492 bits
	16	**448**	24	6	

TABLE 2 Galileo CS E6B per-second data structure

Figure 5.2: Structure of the CS page [40].

As reported in Fig. 5.2, a page in the commercial service is a 500 bit packet of which 448 bit are reserved for data and key management. As stated in [41], it is foreseen that 75-90% of the CS bandwidth will be devoted to the high accuracy service, while the rest will be left to authentication and key management. Moreover, the commercial service allows to have different satellite broadcast different messages. This might allow to:

- Assign different sets of satellites to different services (e.g., two providers might offer high accuracy services, each of them using half of the satellites for its own frames). In the implementation of this solution the system should take into account the problem of availability, which will get worse in this situation.

- Have different satellites broadcast the same pages, but shifted in time, so that a receiver can exploit all the satellites in view to improve the decoding time.

The first approach impacts availability but it can improve the bandwidth use, achieving independence between some of the service providers. The second approach improves decoding performances by exploiting the presence of multiple satellites in view that can broadcast different information. When satellites are transmitting the same message at the same time, receiving from more than one can improve robustness, since errors from one transmission can be corrected by others.

Let's assume there are four service providers: two provide high accuracy data, and they can be mapped to two separate sets of satellites. The other two provide authentication services, one with tracking capabilities and the other just "a priori" and "a posteriori". This means that each satellite will have to broadcast data and key management information for three independent key management schemes (one HA provider and two authentication providers).

Let's assume that high accuracy data will occupy 380 bit of every page, while authentication will be assigned 28 bit. The remaining 40 bit field will be dedicated to key management for the three service providers.

Since each page has its own CRC and FEC, it is desirable to concentrate all the key management information of each service provider into the least possible amount of pages: each user will be required to correctly decode those pages before being able to decode data. Since the communication overhead varies depending on the broadcast mode the system is using (revocation mode requires more bits), it is not possible to know in advance how many pages should be broadcast for each service provider every T_{min}.

For what concerns authentication, let's take the approach with independent signatures, so that each service provider will have complete control on its own key management scheme.

Let's look at the best and worst case:

- In the best case each service provider is in regular mode, which implies an overhead of 512 bit for key management, plus a 512 bit ECDSA digital signature, giving in total $1024 \cdot 3 = 3072$ bits for three service providers.

- In the worst case a service provider is in revocation mode, broadcasting B_{max} bits for the set identification message. This requires overall $3072+3000 = 6072$ bit for three service providers.

If the key management allocation is a 40 bit field for every page, the number of required pages is roughly 26 per service provider for the best case and 51 for the worst case.

Every T_{min} each service provider needs to cyclically broadcast the key management messages necessary for its users to decrypt the service data. The system will need to schedule these messages, dividing them into pages. There are several possibilities:

- Since each subframe contains 15 pages, the system could allocate subframes alternating between service providers: e.g., every *key management cycle* of six subframes, each provider will transmit its data on one of them, while the fourth will contain the signature. In the best case a service provider will need only two subframe for the entire key management message (which will be retransmitted right away, ending in the next key management cycle); in the worst case each service provider will need almost three and a half subframes to transmit the whole key management message, requiring four key management cycles of 18 subframes.

- The key management scheme could transmit all the messages of each service provider without scheduling, one provider after another. This would cause dependence between providers, since when one is in revocation mode, the others will experience reduced performances.
- The system might allow to change field size inside each page, allowing providers to have a bigger field for key management, reducing the field for data, in revocation mode.

These are just some of the paradigms that might be used to organize the broadcast of key management messages. Let's take into account the first, represented in Fig. 5.3, and have a look to the decoding performances for the CS signal. What might be interesting, indeed, is the time it takes for a user to correctly decode the key management messages that are needed in order to access the service.

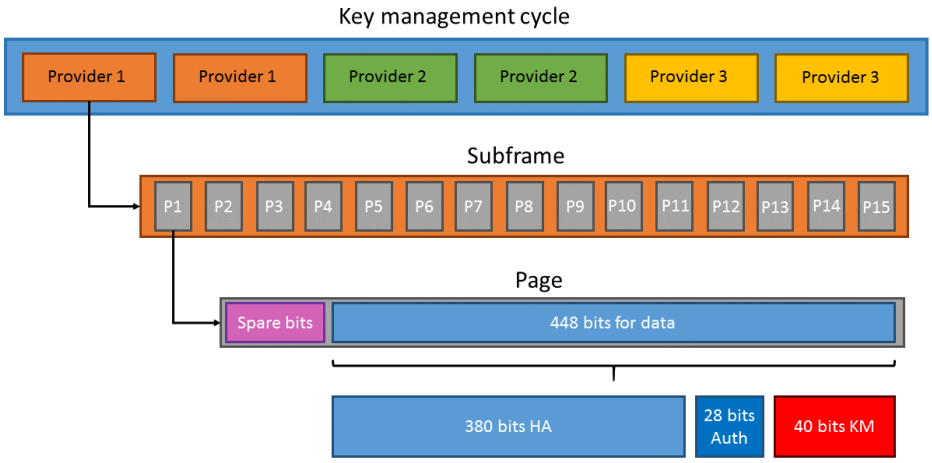


Figure 5.3: Organization of the key management overhead into pages, subframes, and key management cycle.

5.5 Decoding procedure

When a receiver switches on after an idle period of at least $T_{\min}$, it has to retrieve the new group key in order to decode the service data. This requires several steps:

- First of all the receiver needs to understand which mode the provider is using;
- The main key management message (called *header*):

$$H = (C_0, C_1) = \left(g^t, \left(v \prod_{j \in \mathcal{S}} g_{n+1-j} \right)^t \right) \in \mathbb{G}^2 \tag{5.1}$$

  which is 512 bit long and needs to be correctly decoded;

- It the provider is in revocation mode, a variable length message (with a maximum length of 1000 bits under the previous assumptions) shall be correctly received. This message (called *set identification message* in the previous sections) allows the receiver to identify which operations shall be performed on the header in order to retrieve the group key.

- The set identification message and the header need to be authenticated. The concatenation of the two will be verified through the public ECDSA key.

- Once everything has been received correctly (possibly even before verification), the receiver computes the value which allows to retrieve the group keys as:

$$K_G^{\text{rev}} = \frac{e(g_i, C_1)}{e\left(d_i \prod_{j \in \mathcal{S}, j \neq i} g_{n+1-j+i}, C_0\right)}$$

  Alternatively, if every receiver i is already provided with the value $\prod_{j=1, j\neq i}^{n} g_{n+1-j+i}$, the operation gets faster and only requires as many divisions as the number of revoked (or expired) groups since last time the receiver was on.

  By hashing this value with SHA-256, the first group key, $K_G^{t_1}$ can be obtained. By iterating this hashing function, all the keys of the chain can be obtained, which allow to decode the service data for a period of time T_{min}.

Before being able to use the service, each receiver needs to receive 26 to 51 pages. However the transmitted pages are subject to channel errors, and will not be all correctly decoded at the first transmission. The satellite navigation channel is subject to multipath, which causes delayed reflections of the signal, strongly impacting the decoding performances. Moreover, the impact of this effect depends on several factors, as the elevation of the satellites, the type of environment and the speed of receivers.

As the authors of [42] argue, the most accurate model for the land mobile satellite channel is the 2-state LMS model, which is a generative model, capable of generating time series according to a combination of distributions. This model simulates propagation according to the intensity of shadowing events, which lead to the system being either in the "good state" or the "bad state".

For the scope of this thesis, which aims at providing a rough idea of the decoding performances, the results of the field tests reported in [42] will be exploited. The test campaign aims at evaluating the tracking availability and data dissemination performances of the Galileo CS signal, which has been tested in realistic environments (rural, urban and suburban vehicle scenarios). Their results underline that:

- Tracking of the E6B component only (without the aiding of E6C) results in a degradation of both tracking availability and PER on broadcast pages.

- The real propagation conditions in urban and suburban environments strongly impact the PER, as well as the satellite elevation. Even in the rural scenario, the PER is around 10^{-1} for low elevations, improving to 10^{-2} or 10^{-3} for medium to high elevations.

Some of the results in [42] are represented in Fig. 5.4, where the rural vehicle scenario is taken into account with respect to C/N_0, and in Fig. 5.5, where the PER in different scenarios are compared with respect to elevation.

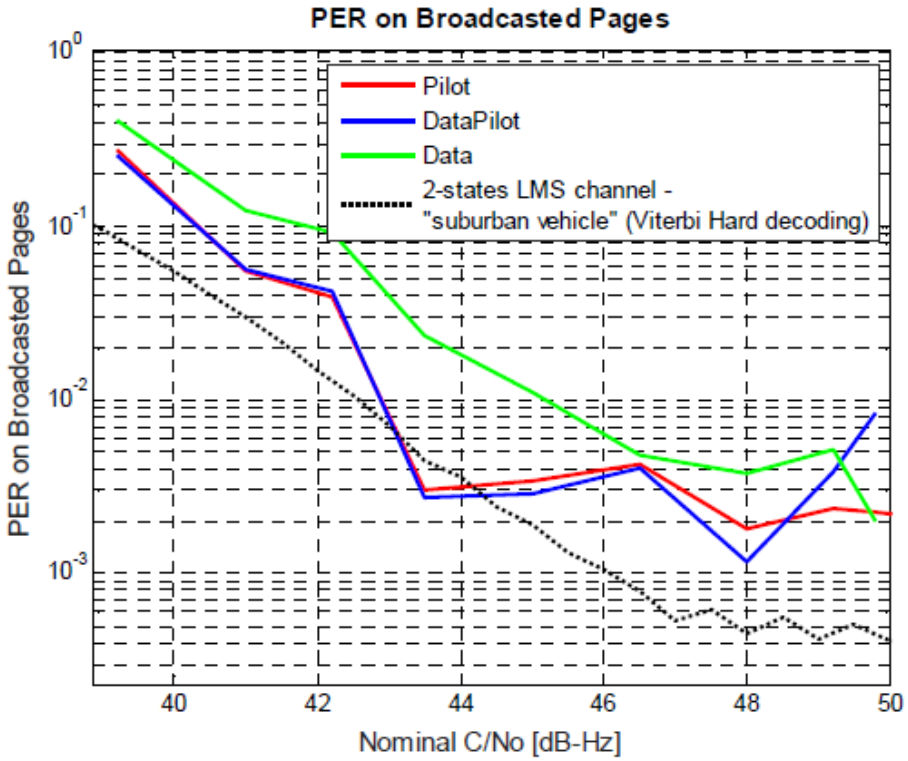


Figure 5.4: PER in rural vehicle scenarios compared with the 2-state LMS model in suburban conditions.

As it can be seen from the images, the most meaningful values for the PER go roughly from 0.9 to 10^{-3}. These range of values will be considered for the scope of performance evaluation.

Evaluation of the decoding performance

In order to provide a rough estimate of the time required to decode key management messages, let's assume the transmission of each page is independent of the others, and subject to errors with a probability P, which can be derived from the experimental results in [42]. Let's called **key management cycle** a cycle of six subframes, where each service provider has rights to fill two contiguous subframes with its own key management messages. Summarizing, in the best case, when the service provider is

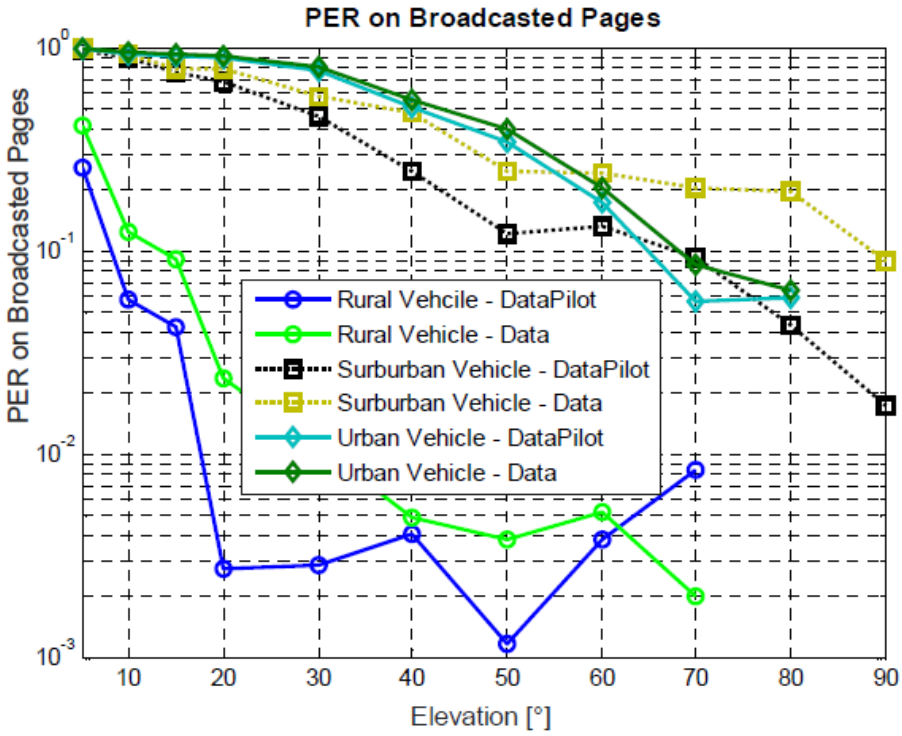


Figure 5.5: PER for different satellite elevation.

in regular mode, 26 pages are enough to contain (in the 40 bit field reserved to key management) all the messages. In the worst case, when the service provider is in revocation mode, and the set identification message has maximum length ($B_{\text{max}} = 1000$ bit), the messages are contained in 51 pages. 30 pages per service provider can fit inside a key management cycle, and if a key management message exceeds 30 pages, receivers will need to wait the following cycle before receiving the following part.

Let's for the moment ignore the key management cycle, and assume the whole key management pages are transmitted in a contiguous fashion, and get repeated right after each transmission. Let's assume that a receivers switches on at the beginning of the transmission (without any loss of generality), and aims at correctly retrieving all the pages. Each page is correctly decoded with probability $(1 - P)$ independently of the others. After completing the first transmission, all the pages will be repeated for the second time, and so on for many cycles. After the first transmission is completed, the receiver will probably have missed some pages, and at the next cycle it will ignore the correctly decoded pages (there is no need to process them again), and retrieve some of the missing pages, and so on, until all pages have been correctly received at least once. This can be seen as a sequence of independent trials: the measure of interest is the time it takes for the receiver to correctly decode all N_{page} pages.

Let $A_{i,n}$ denote the event that at transmission i, page n is correctly decoded. The events $A_{i,n}$, with $n = 1, \ldots, N_{\text{page}}$ and $i = 1, \ldots,$ are all independent with the same probability $P[A_{i,n}] = 1 - P$. Then, the probability that the receiver can collect all pages within the first k transmissions, that is the

CDF of the random variable T, is given by:

$$P[T \leq k] = P\left[\bigcap_{n=1}^{N_{\text{page}}} \left(\bigcup_{i=1}^{k} A_{i,n}\right)\right] = \prod_{n=1}^{N_{\text{page}}} \left(1 - \prod_{i=1}^{k} \left(1 - P\left[A_{i,n}\right]\right)\right) = \left(1 - P^{k}\right)^{N_{\text{page}}} \quad (5.2)$$

This expression allows to find the probability that the receiver has to wait k cycles of transmission from *its own service provider* before receiving all the needed key management messages. This allows to find the CDF of the random variable T, and the number of trials which allows to correctly decode with a high enough probability (let's consider 99% probability, corresponding to T_{99}). Once the number of transmissions is known, an approximation of the time to correctly decode can be computed as $\lceil \frac{N_{\text{page}} \cdot T_{99}}{N_{\text{cycle}}} \rceil T_{\text{KMcycle}}$, where N_{cycle} is the number of pages granted to each service provider every key management cycle (30 pages, corresponding to 2 subframes, in the example considered before), and T_{KMcycle} is the duration of the entire key management cycle (6 subframes in the example, corresponding to 90 seconds).

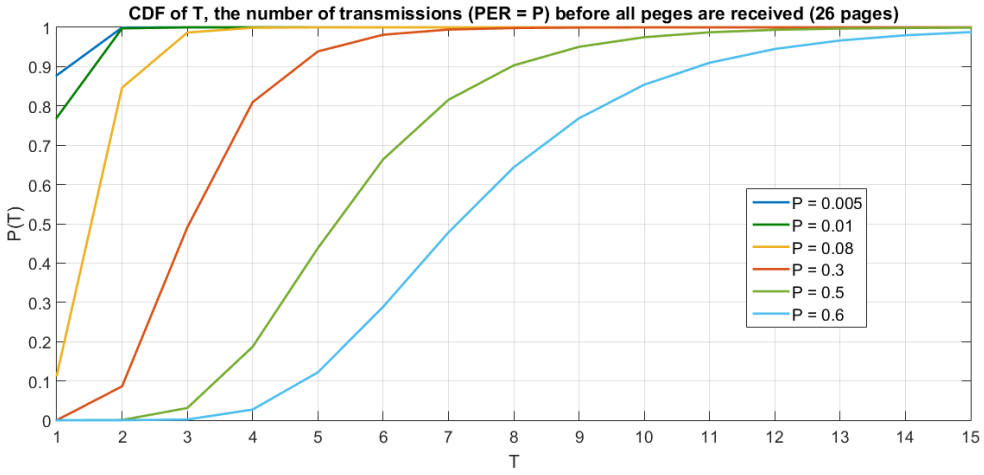


Figure 5.6: CDF for random variable T.

Fig. 5.6 reports the CDF of the number of transmission for different values of the packet error rate. Fig. 5.7 shows the number of transmissions that allow to correctly receive all key management messages with probability 0.99.

- If the packet error rate lies below 0.01, only two transmissions are required in both best and worst case. This leads to a time to decode of $\lceil \frac{N_{\text{page}} \cdot T_{99}}{N_{\text{cycle}}} \rceil T_{\text{KMcycle}} = \lceil \frac{26 \cdot 2}{30} \rceil 90s = 180s$, which is 3 minutes (best case). In the worst case this value rises up to 6 minutes, since the maximum length of key management messages is 51 pages.
- If the packet error rate lies between 10^{-2} and 10^{-1}, then the best case leads to a time to decode of 6 minutes, while the worst case takes roughly 10 and a half minutes.

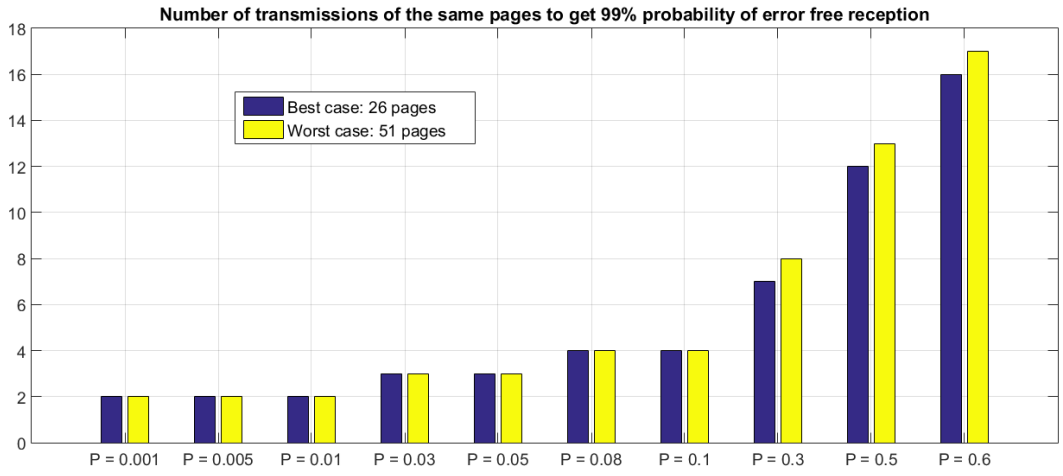


Figure 5.7: Required transmissions to correctly decode all messages with 99% probability.

- If the packet error rate goes up to 50%, the best case will take 16 and a half minutes, which become more than half an hour for the worst case.

What here is referred as "the worst case" can be tuned by service providers in order to cope with the dynamics of the channel, in order to achieve the desired trade off between time to decode and random revocation capabilities.

5.5.1 Considerations on continuity of service and public keys

The protocol devised in the previous sections allows the system to schedule the change of group keys, using a new one every T_{min}. Some problems may arise around the renewal time of group keys: receivers that are currently using the service will have to retrieve the new group key, which requires several minutes, as seen in the previous sections. During this retrieval time, the service will no longer be available, which could be a problem for users who need it right away. There are at least two straightforward solutions:

- Key management messages related to the new group key will be broadcast some time before the switching instant, in order to provide continuity to all connected receivers. Service providers should estimate the maximum decoding time for key management messages which is reasonable to assume for receivers in regular conditions, T_{dec}^{max}. Key management messages for the new group key will be broadcast T_{dec}^{max} before the scheduled group key update. This will guarantee continuity with the desired probability. The drawback is that receivers who switch on T_{dec}^{max} before the group key change, and who don't possess the current group key, will be able to use the service only after T_{dec}^{max}. This should not be a major issue, since once the receivers know the scheduling, they can easily adjust to it.

- Some time before the group key update two headers (one relative to the current group key and one to the new one) might be broadcast instead of one. This will heavily impact the communication overhead of the scheme, but might be a preferable solution when continuity is more critical than latency.

Another important consideration involves ECDSA signature keys. 128 bits of security can be tolerated for low bandwidth scenario as long as a key update is scheduled in order to preserve security. This key update should rely on much safer (and therefore longer) signature keys, and on aiding channels for the transmission of new key material to all receivers. If a 128 bit ECDSA key is a compromise between security and efficiency, the external layer key used to sign public key updates shall be longer and therefore more secure. This *chain of trust* is necessary for the system to remain operational and deal with possible security issues: if the external layer authentication key becomes compromised, there system can be considered completely broken.

5.6 Conclusions

A novel key management scheme, called *RevHash*, has been proposed in this thesis, which is particularly adequate for subscription based services in low bandwidth scenarios. The literature review and analysis on the existing broadcast encryption schemes has revealed that only a few of them can be adapted to access control for GNSS services, since in most cases the bandwidth requirements for the scheme are not feasible, leading either to prohibitive decoding times or very poor autonomy. *RevHash* can achieve constant communication overhead and constant private key size, as the scheme *BiMaps* on which it is based. Moreover, the proposed scheme solves the scalability issues of *BiMaps* while preserving security. The combination of efficiency and scalability differentiates *RevHash* from any other key management protocol in literature. This scheme allows to support stateless and autonomous users, and is tunable in a trade-off between statelessness, efficiency and autonomy, without requiring to sacrifice security. Not only the scheme is collusion resistant against non-adaptive adversaries, but the protocol implementation also provides mechanisms to recover security even after private keys are compromised, thanks to random revocation capabilities at group level. Random revocation can be performed by each service provider according to the system needs, as two or more source coding paradigms can be selected to optimize the overhead required to identify the set of revoked users. The proposed implementation exploits the system knowledge on users subscription periods to efficiently deal with subscription expiration, treating random revocation as a different problem.

The performance of the scheme with the proposed implementation has been assessed, by choosing parameters values on the ground of considerations and assumptions found in literature ([40], [41], [42]). The decoding performance indicates that, for reasonable PER values, the time to decode key management messages with 99% probability is around three to ten minutes. This results are quite appealing, especially in such low bandwidth conditions.

The results on performance have been derived under a memoryless page channel model without considering that the channel might experience long periods in a "bad state" due to shadowing phenomena: it would be interesting to assess the decoding performance with a more accurate model, both from the theoretic point of view and through simulation. Moreover, the improvement in robustness due to the reception from different satellites at the same time should be taken into account to improve the trade-offs. Satellite scheduling will be impacted by the reduced number of uplink stations, not to mention that a precise scheduling mechanism has not been devised, yet.

All these ideas for protocol optimization, and many others, would be a most challenging research topic, which might soon find its application in the first satellite commercial service.

Bibliography

[1] S. Lo, D. D. Lorenzo, P. Enge, D. Akos, and P. Bradley, "A Secure Civil GNSS for Today," *InsideGNSS2*, pp. 30–39, 2009.

[2] H.-S. Koo, O. H. Kwon, and S.-W. Ra, "An active entitlement key management for conditional access system on digital tv broadcasting network," in *2006 IEEE/IFIP Network Operations and Management Symposium NOMS 2006*, pp. 1–4, April 2006.

[3] A. Fiat and M. Naor, "Advances in Cryptology — CRYPTO' 93," *CRYPTO '93: Proceedings of the 13th Annual International Conference on Advances in Cryptology*, vol. 773, pp. 480–491, 1994.

[4] C. Blundo and A. Cresti, *Advances in Cryptology — EUROCRYPT'94: Workshop on the Theory and Application of Cryptographic Techniques Perugia, Italy, May 9–12, 1994 Proceedings*, ch. Space requirements for broadcast encryption, pp. 287–298. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995.

[5] M. Luby and J. Staddon, "Combinatorial Bounds for Broadcast Encryption," 1998.

[6] M. Abdalla, Y. Shavitt, and A. Wool, "Key management for restricted multicast using broadcast encryption," *{IEEE/ACM} Transactions on Networking*, vol. 8, no. 4, pp. 443–454, 2000.

[7] R. Kumar, S. Rajagopalan, and A. Sahai, *Advances in Cryptology — CRYPTO' 99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings*, ch. Coding Constructions for Blacklisting Problems without Computational Assumptions, pp. 609–623. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999.

[8] M. Naor and B. Pinkas, "Efficient trace and revoke schemes," *International Journal of Information Security*, vol. 9, no. 6, pp. 411–424, 2010.

[9] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 16–30, Feb 2000.

[10] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: Issues and architectures," tech. rep., United States, 1999.

[11] K. Y. Chou, Y. R. Chen, and W. G. Tzeng, "An efficient and secure group key management scheme supporting frequent key updates on Pay-TV systems," *APNOMS 2011 - 13th Asia-Pacific Network Operations and Management Symposium: Managing Clouds, Smart Networks and Services, Final Program*, 2011.

[12] Jia-Yin Tian, Cheng Yang, Yi-chun Zhang, and Jian-Bo Liu, "Distributed Key Management Scheme for Large Scale Pay-TV Application," pp. 30–33, 2010.

[13] D. Naor, M. Naor, and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers," *Advances in Cryptology CRYPTO 2001*, vol. 2139, no. June, pp. 1–36, 2001.

[14] D. Micciancio and S. Panjwani, "Optimal communication complexity of generic multicast key distribution," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 803–813, 2008.

[15] J. Fan and M. H. Ammar, "HySOR : Group Key Management with Collusion-Scalability Tradeoffs Using a Hybrid Structuring of Receivers," vol. 00, no. 1, pp. 196–201, 2002.

[16] H. M. Sun, S. Y. Chang, S. M. Chen, and C. C. Chiu, "An efficient rekeying scheme for multicast and broadcast (M&B) in mobile WiMAX," *Proceedings of the 3rd IEEE Asia-Pacific Services Computing Conference, APSCC 2008*, pp. 199–204, 2008.

[17] J. Liu and C. Wang, "Exclusive Key Based Group Rekeying Protocols," no. 61173189, pp. 1–29.

[18] D. Boneh, C. Gentry, and B. Waters, "Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys," *Advances in Cryptology – CRYPTO 2005*, vol. 3621, no. 1, pp. 258–275, 2005.

[19] D. Boneh and B. Waters, "A fully collusion resistant broadcast, trace, and revoke system," *Proceedings of the 13th ACM conference on Computer and communications security - CCS '06*, p. 211, 2006.

[20] C. Delerable, P. Paillier, and D. Pointcheval, *Pairing-Based Cryptography – Pairing 2007: First International Conference, Tokyo, Japan, July 2-4, 2007. Proceedings*, ch. Fully Collusion Secure Dynamic Broadcast Encryption with Constant-Size Ciphertexts or Decryption Keys, pp. 39–59. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[21] J. T. Curran and M. Paonni, "Securing GNSS: An End-to-End Feasibility Study for the Galileo Open Service," in *International Technical Meeting of the Satellite Division of The Institute of Navigation, ION GNSS*, pp. 1–15, 2014.

[22] B. Briscoe, "MARKS: Zero side effect multicast key management using arbitrarily revealed key sequences," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1736, pp. 301–320, 1999.

[23] L. Veltri, S. Cirani, S. Busanelli, and G. Ferrari, "A novel batch-based group key management protocol applied to the Internet of Things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2724–2737, 2013.

[24] S. Panjwani, S. D. C. S. University of California, and Engineering, *Private Group Communication: Two Perspectives and a Unifying Solution*. University of California, San Diego, 2007.

[25] G. Caparra, S. Sturaro, N. Laurenti, and C. Wullems, "Evaluating the security of one-way key chains in TESLA-based GNSS Navigation Message Authentication schemes," in *2016 International Conference on Localization and GNSS (ICL-GNSS)*, (Barcellona), pp. 1–6, IEEE, jun 2016.

[26] "Smart Card Technology, cardwerk - smarter card solutions." `http://www.cardwerk.com/smartcards/smartcard_technology.aspx`. Last modified: 2016-08-02.

[27] D. Naor, M. Naor, and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers," *Advances in Cryptology CRYPTO 2001*, vol. 2139, no. June, pp. 1–36, 2001.

[28] W. Chen, Z. Ge, C. Zhang, J. Kurose, and D. Towsley, *On Dynamic Subset Difference Revocation Scheme*, pp. 743–758. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[29] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM J. Comput.*, vol. 17, pp. 281–308, Apr. 1988.

[30] D. Boneh, "Twenty years of attacks on the rsa cryptosystem," *NOTICES OF THE AMS*, vol. 46, pp. 203–213, 1999.

[31] D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[32] A. A. Yavuz, "ETA: efficient and tiny and authentication for heterogeneous wireless systems," in *ACM conference on Wireless network security, WiSec*, pp. 67–72, 2013.

[33] D. Boneh, *BLS Short Digital Signatures*, pp. 158–159. Boston, MA: Springer US, 2011.

[34] D. Naccache and J. Stern, "Signing on a postcard," in *Proceedings of the 4th International Conference on Financial Cryptography*, FC '00, (London, UK, UK), pp. 121–135, Springer-Verlag, 2001.

[35] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '01, (London, UK, UK), pp. 514–532, Springer-Verlag, 2001.

[36] ETSI, *Quantum Safe Cryptography and Security; An introduction, benefits, enablers and challenges*. 2014.

[37] N. P. Smart, V. Rijmen, B. Gierlichs, K. G. Paterson, M. Stam, B. Warinschi, and G. Watson, "Algorithms, Key Size and Parameters Report," tech. rep., ENISA, 2014.

[38] R. Blank, A. Secretary, P. D. Gallagher, E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Nist special publication 800-57, recommendation for key management part 1: General (revision 3)," 2012.

[39] B. Lynn, "Ben Lynn June 2007," no. June, pp. 1–126, 2007.

[40] I. Fernández-Hernández, I. Rodríguez, G. Tobías, J. D. Calle, E. Carbonell, G. Seco-Granados, J. Simón, and R. Blasi, "Galileo Commercial Service. Testing GNSS High Accuracy and Authentication," *InsideGNSS*, vol. 10, no. 1, pp. 38–48, 2015.

[41] I. Fernandez-Hernandez, "The Galileo Commercial Service : Current Status and Prospects The Galileo Commercial Service : Current Status and Prospects," no. April, 2014.

[42] J. A. Garcia, M. Navarro, M. Cordero, and J. Miguez, "E6 CS Encryption Flexibility- User Receiver Impact Based on Field Testing," pp. 1–32, 2016.

Acknowledgements

I would like to thank Dr. Massimo Crisci and Dr. Rigas T. Ioannides for having me at ESTEC, where I had the chance of working with people I admire and look up to, especially Dr. Christian Wullems, whom I thank for his patience, spirit and for the time we spent together. My gratitude goes to prof. Nicola Laurenti, Dr. Gianluca Caparra, Dr. James Curran and Silvia, who encouraged me and inspired me with their passion.

A special thank you goes to my parents and my family, who give me their unconditional support and never doubt me; to Riccardo, who has always been there for me; and to my sister Martina, who has always been my role model.

My deepest gratitude goes to my closest friends, Davide, Alessandra, Giulio, Elisa, Andrea and Alvise, with whom I have shared more than I can express, and to all my colleagues from university, whose friendship has filled these five years with laughter and dearest memories.

I am grateful to all friends from ESTEC, who taught me to always keep a positive attitude, and made my adventure in the Netherlands memorable; in particular I'd like to thank Sebastian for his precious support, both from the professional and personal point of view.