An Efficient Scheme for Centralized Group Key Management in Collaborative Environments

Constantinos Patsakis[a], Agusti Solanas[b]

[a] *Distributed Systems Group, School of Computer Science and Statistics, Trinity College, College Green, Dublin 2, Ireland.*
[b] *UNESCO Chair in Data Privacy, Dept. Computer Engineering and Mathematics, Universitat Rovira i Virgili, Catalonia. (Spain).*

Abstract

The increasing demand for on-line collaborative applications has sparked the interest for multicast services, which in many cases have to guarantee properties such as authentication or confidentiality within groups of users. To do so, cryptographic protocols are generally used and the cryptographic keys, in which they rely, have to be managed (e.g. created, updated, distributed). The procedures to perform these operations are determined by the so-called *Group Key Management Schemes*. Many schemes have been proposed and some of them have been proven to be vulnerable. This is the case of the Piao et al. scheme, whose scalability/efficiency is very good but it is vulnerable to many attacks because its security is based on a "weak" mathematical problem, so it can be broken in polynomial time.

Inspired by the concepts proposed in the Piao et al. scheme we have redesigned the protocol and we have founded it on a hard mathematical problem and tweaked some of the procedures. This way, we propose a new scheme that is efficient, collusion free, and provides backward and forward secrecy.

Keywords: Group key management, Rekeying, collaborative platforms, Secure group communication, Access control

1. Introduction

Communication networks have become basic infrastructures in advanced societies, and their use is growing steadily throughout the world. Those networks are mainly used to send and receive a variety of information in most cases from producers (*e.g.* newspapers, web search engines, virtual shops, etc.) to consumers (*e.g* end users). However, this classical model is witnessing the appearance of new actors that play the role of both producers and consumers, those are the so-called *prosumers*.

Email addresses: `patsakik@scss.tcd.ie` (Constantinos Patsakis), `agusti.solanas@urv.cat` (Agusti Solanas)

Currently, the users of communication networks like Internet, are not only passive consumers of information but active creators of contents. A clear example of this model could be found in social networks like facebook or twitter in which users produce/create content and also consume the content created by their friends (within the virtual social network).

In most cases the communications that take place in these networks are 1-to-1 communications, this is, a message (m) is sent from user U_i to user U_j through a given channel consisting of routing nodes that propagate (m) from the sender to the receiver. This kind of communication is generally referred to as unicast communication. Although this is the most frequent communication mode, the fact that users tend to gather in groups to access common services [1] has popularized 1-to-n communications also known as multicast communications, in which a message (m) is sent from a user U_i to a set/group of users $G = \{U_j, U_k, \cdots U_n\}$.

Cryptography plays a fundamental role in making all these communications secure. Thanks to the use of the proper protocols, users are provided with properties such as authentication, integrity, non-repudiation, etc. However, cryptographic protocols are based on the use of cryptographic keys that could be understood as a common secret that users could use to communicate privately and securely. In the simple case of unicast communication, we could assume that user U_i shares a common secret, the key K, with a user u_j. This way, when U_i wants to send a message m to U_j, he/she encrypts the message using K. By doing so, only U_j will be able to decrypt m since he/she is the only one who also knows K. Thus, to send a message securely from U_i to U_j one encryption and one decryption operation are required and a single key K is used.

Notwithstanding, the simple case described above does not scale well when we consider a multicast group. In this case, since we have a number of users n that want to securely communicate with each other, each user u_i shares a key with the other users in the group (*i.e.* $n-1$ different keys), and the same happens with the other members. Therefore, the total number of keys is $n^2 - n$ or $(n-1) + (n-2) + \cdots + 1 = \frac{(n-1)(1+(n-1))}{2} = \frac{n^2-n}{2}$ if we consider that the key used to communicate from u_i to U_j is equal to the one used to communicate from u_j to U_i. Clearly, if the above approach is used, the number of keys grows quadratically with a linear growth in the number of users. Thus, the scalability of the system is poor.

To avert the aforementioned scalability problem, group keys are used. In this case, when a group of users share information, they can share it securely by encrypting it with a common group key K_G. This way, using a single key the communications are secured. However, any modification of the group of users implies a recalculation and redistribution of the key. Otherwise, new users could

[1] There are many services that are accessed in groups and have sense for multicast communications, namely video and audio streaming, vehicle-2-vehicle communications in platoons, content sharing in p2p communications, etc.

read previous messages and "old" users could keep reading new messages (*i.e.* backward and forward secrecy cannot be guaranteed).

There are many different schemes that tackled the problem of group key management. Their goals are diverse and adapt to several types of multicast communications requirements. According to [1] we can distinguish the following main requirements for group key management schemes:

Security requirements: Forward secrecy, backward secrecy, collusion freedom, key independence and minimal trust;

Quality of Service requirements: Low bandwidth, no 1-affects-n, minimal delays and service availability;

Key server requirements: Low storage and low computation;

Group users requirements: Low storage and low computation.

According to the methods used to distribute the key, group key management schemes can be classified in three categories as follows:

Centralized: In these schemes there is a trusted third party, known as Key Distribution Center (KDC), which is responsible for generating, distributing, renewing and revoking group keys. This kind of schemes can be classified according to the way they share the key in pairwise, broadcast or hierarchical. Some examples can be found in: [2, 3, 4].

Decentralized: In these schemes, the responsibilities of the KDC are shared among several group members generally organized in a hierarchical structure that reduces the appearance of bottlenecks and eliminates the problem of a single point of failure. These schemes can be further classified into membership-driven or time-driven depending on when the key is updated. Examples of decentralized schemes can be found in: [5] and [6].

Distributed: In these schemes, each member of the group cooperates in the generation of the group key and one member distributes it. These schemes could be further classified according to the type of cooperation in Ring-based cooperation, hierarchical cooperation and broadcast cooperation. Typical examples of this kind of schemes can be found in: [7, 8].

For further and more complete information on group key management schemes, the reader may refer to [1, 9].

1.1. Contribution and Outline of the Article

The importance of properly managing keys within groups is apparent and many efforts are being devoted to proposing new methods that guarantee security (i.e. backward secrecy, forward secrecy and collusion freedom) while at the same time efficiency and QoS are achieved.

Recently Piao's et al. [10] proposed a scheme for group key management that has proven to be very efficient in terms of scalability. However, it has been shown that the scheme is not secure because the mathematical problem in which it is based can be easily solved, in polynomial time. Although the scheme has been broken, it has inspired us and we leverage some of its ideas to propose a novel scheme that shares the efficiency but solves the security issues related to the underlying mathematical problem, by setting it to a more solid ground. The security of the scheme now is based on the finding roots of polynomials modulo big composite numbers, for which the factorization is not known to the users.

The rest of the article is organized as follows: In §2 we recall the scheme by Piao et al. [10] and show why it is not secure. Next, in §3 we propose our scheme and we discuss its properties in §4. Finally, the article concludes in §5 with some final remarks.

2. Piao's et al. scheme

2.1. A brief overview

Piao et al. in [10] proposed the following scheme: They assume a group $\mathcal{G}$ with the initial users $U_1, U_2, ..., U_m$. These users send a message to the Key Distribution Center (KDC) asking for the generation of a group key. The KDC receives the request and generates m keys for the users *i.e.* the keys $k_1, k_2, ..., k_m \in GF_p$, as well as the group key $K_G \in GF_p$, where p is a large prime. From these values, the KDC computes the following univariate polynomial:

$$P(x) = \prod_{i=1}^{m}(x - k_i) + K_G$$

Afterwards, the KDC sends the message $[(k_i, P(x))]$ to each user U_i accordingly. On receiving the message, each user may compute K_G, by simply calculating $P(k_i)$:

$$P(k_i) = (k_i - k_1)(k_i - k_2)...(k_i - k_i)...(k_i - k_m) + K_G \equiv 0 + K_G \equiv K_G$$

When the group of users $\mathcal{G}$ changes (*i.e.* either one or more users leave or one or more users join) the procedure is essentially the same. In the case of a new user U_{m+1} joining the group, the KDC picks a new group key K'_G, creates a key k_{m+1} for U_{m+1} and calculates a new polynomial $P'(x)$ as follows:

$$P'(x) = \prod_{i=1}^{m+1}(x - k_i) + K'_G$$

This new polynomial is sent to all the users in the group (so that they can compute the new group key), and the new user receives the message $[(k_{m+1}, P'(x))]$.

In the case in which a user U_k leaves the group the procedure is almost the same. A new group key K'_G is generated by the KDC, and the resulting polynomial $P'(x)$ is sent to all the remaining users. Note that since the keys of the remaining users have not changed, they can obtain the group key by calculating $P'(k_i)$.

For further details on the generation of the polynomials in each case we refer the interested reader to the original article of Piao et al. [10].

2.2. Security of the scheme

Although the above scheme is very efficient, it is not secure. Independently, in [11] and [12] the scheme has been analyzed and proven to be insecure. Initially, Kamal in [11] showed that the scheme does not provide forward and backward secrecy. In addition, Liu et al. in [12] describe the linchpin of the security problem present in the scheme, *i.e.* the underlying mathematical problem in which the security of the scheme is based is not hard. Actually, the problem does not belong to the class of NP-hard problems and there are several deterministic and probabilistic algorithms that can efficiently solve it in polynomial time.

The security of the above scheme is based on the wrong assumption that users cannot find the keys of other users. Given that we have a univariate polynomial in a finite field, the problem falls down to finding it's roots. If we assume that there is a malicious user U_M in $\mathcal{G}$, he knows $P(x)$, K_G and his own key k_M, then U_M may calculate the following polynomial:

$$f(x) = \frac{P(x) - K_G}{x - k_M}$$

Thus, to obtain the keys of the other users, U_M simply has to factorize $f(x)$ in a finite field by using any of the algorithms that solve this problem efficiently [13, 14, 15]. For more on the topic of factoring polynomials over finite fields, the reader may refer to [16].

U_M can recover all the keys of the other users with a cost in the order of $O((m-1)^3 log(m-1)\ logp)$, where m is the degree of the polynomial (in this case, equal to the number of users) and p the order of the finite field $GF(p)$ used. This means that for as long as U_M belongs to the group, he might compute the keys of the other users. Even after leaving the group, U_M might still find the new group key if he intercepts the new polynomial $P'(x)$, as he known the keys of the other users.

Liu et al. assume that in the Piao's et al. scheme, the authors mistook the problem of super-sparse univariate polynomials, which are known to be NP-hard [17, 18], for the problem of normal univariate polynomials. The reason why this assumption seems valid is that Piao's et al. cite Gao et al. in [19], which correctly states which classes of univariate polynomial problems belong in each complexity class.

3. Our proposed scheme

3.1. Main actors and desiderata

In our scheme, we consider two main actors, the Key Distribution Center (KDC) and a set of users that want to form a group by sharing a common key *i.e.* the group key. Each user is denoted as U_i and the group key is K_G. The proposed scheme does not handle the messages between users to form their group or between users and the KDC. We assume that there is a secure channel that can be used once, on the first contact of a user with the KDC, so that their first key can be exchanged securely. Afterwards, all the messages that the KDC creates can be multicast. Also, the initial messages or the agreement about which users join the group is beyond the scope of this article.

The scheme that we propose takes for granted that all the operations are authenticated. This means that during the initialization phase the group of users (that requested the creation of the group) have already been authenticated by another party. Also, we assume that the admission or exclusion of a user is already approved. Hence, our scheme provides strictly key management.

The proposed scheme is designed to:

- Be secure.
- Scale easily, providing keys for thousands of users.
- Be collusion free
- Provide forward & backward secrecy

3.2. Initialization phase

As in the original scheme by Piao at al., we assume that we have a set of initial users $U_1, U_2, ..., U_m$ that want to build a group $\mathcal{G}$ and they send the proper message to the KDC. The KDC receives these messages from the requesting users, generates two prime numbers (p, q) and calculates their product $n = pq$. After that, the KDC picks $m+1$ random values $k_1, k_2, ..., k_{m+1} \in \mathbb{Z}_n$, so that the following two conditions are satisfied:

$$k_i > \sqrt{n}, \forall i \in \{1, ..., m+1\}$$
$$|k_i - k_j| > n^{1/4}, \forall i \neq j,\ i, j \in \{1, ..., m+1\}$$

The KDC then generates a random value $K_G \in \mathbb{Z}_n$ and calculates the following polynomial:

$$P(x) = \prod_{i=1}^{m+1} (x - k_i) + K_G \ mod\ n$$

Finally, the KDC sends a message with the triple: $[(k_i, n, P(x))]$ to each user $U_i, i \in \{1, ..., m\}$. Note that like in the Piao et al. scheme, each user U_i can extract K_G by computing $P(k_i)$.

Our scheme always calculates an additional key (k_{m+1}) that we denote as k_S and that does not belong to any user. This additional key is used for salting the polynomial so that the scheme can be collusion free. Its role is discussed in detail in the next section.

3.3. User addition

If a new user U_N is accepted to join the group $\mathcal{G}$, the KDC calculates a new group key by using a secure hash function h as follows:

$$K'_G = h(K_G)$$

Then, it sends the message $[k_N, K'_G]$ to the new user U_N through a secure channel and an update message to all the current users. Upon receiving the update message, each user $U_i, i \in \{1, ..., m\}$ can determine the new group key, by calculating $K'_G = h(K_G)$. Note that this is possible because the hash function h is public and all the former users in the group already know the "old" group key K_G. Also, note that since h is a secure hash function it is not possible to compute K_G from K'_G, thus, the new user cannot obtain any information from the messages sent using the "old" group key.

For security reasons that are going to be discussed in the next section, if KDC wants to publish the new polynomial $P(x)$, then the salting polynomial has to be rekeyed.

3.4. User revocation

Without loss of generality, we assume that user U_m leaves the group or he/she is expelled from the group. In this situation, the KDC selects a new group key K'_G, a new random $k_m \in \mathbb{Z}_n$, that is kept secret, and a new salt $k'_S \in \mathbb{Z}_n$ and calculates the new polynomial as follows:

$$P'(x) = (x - k'_S) \prod_{i=1}^{m} (x - k_i) + K'_G \ mod\ n$$

Then, the KDC distributes the polynomial $P'(x)$ to all the users that remain in the group $U_i, i \in \{1, 2, ..., m-1\}$.

From the above, it becomes apparent that in our scheme the degree of the polynomial $P(x)$ will never decrease (in fact, it increases when a new user joins and remains equal when a user leaves the group). This procedure might work fine if few users decide to leave the group. Notwithstanding, the degree of the polynomial will increase rapidly if the group is very dynamic, as a result the ratio between the degree of the polynomial and number of actual users in the group will grow (which is not good for efficiency). With the aim to mitigate this problem, we propose the following workaround.

Since the KDC always knows how many keys belong to legitimate users and how many are used for coverings, it can easily control the growing of the latter. To do so, we propose to fix a threshold T. When the number of covering keys reaches the threshold, the KDC picks a random number r between 1 and $T-1$ and removes r covering keys from the polynomial. This will allow the polynomial's degree to be only slightly above the number of users, on average $T/2$.

4. Discussion and Experimental Results

In this section we discuss the security properties of our proposed scheme and provide the necessary proofs regarding backward and forward secrecy, as defined in [20]. Finally, we discuss the experimental results obtained with a real implementation of the proposed scheme.

4.1. Security analysis

In opposition to the original work by Piao et al. which is based on a weak mathematical problem, our approach is founded on a strong one, *i.e.* finding the roots of univariate polynomials modulo large composite numbers, for which the factorization is not known. For the security of our scheme, we pick the composite number n with the same criteria as those for RSA moduli numbers. This constraint keeps the factorization of n secret, thus, preventing any attacker from reducing the problem to finding the roots of univariate polynomials modulo prime numbers, which can be solved efficiently.

Additionally, in order to prevent attacks based on Coppersmith's theorems, the keys of each user are selected accordingly. The first theorem of Coppersmith [21] states:

Theorem 1. *Let N be an integer and $f \in \mathbb{Z}[x]$ a monic polynomial of degree d. Set $X = N^{\frac{1}{d}-\epsilon}$ for some $\epsilon \geq 0$. Given N and f, one can efficiently find all integers x_0, with $|x_0| < X$, satisfying*

$$f(x_0) \equiv 0 \; mod \; N$$

The running time is dominated by the time it takes the Lentra-Lenstra-Lovász (LLL) algorithm on a lattice of dimension $O(w)$ with $w = min(\frac{1}{\epsilon}, logN)$.

Therefore, an attack based on this theorem will rely on the fact that one can find efficiently the roots of univariate polynomials modulo n using lattice based methods, even when the factorization of n is not known if the roots are quite small. Hence, in our scheme the KDC selects all the keys k_is so that $k_i > \sqrt{n}, \forall i \in \{1, ..., m+1\}$.

Extending the theorem above, Coppersmith [22] proved the following theorem for bivariate polynomials.

Theorem 2. *Let $p(x, y)$ be an irreducible polynomial in two variables over $\mathbb{Z}$, of maximum degree δ in each variable separately. Let X and Y be upper bounds on the desired integer solution (x_0, y_0), and let*

$$W = max_{i,j} |p_{ij}| X^i Y^j$$

If $XY < W^{2/(3\delta)}$, then in time polynomial in $(logW, 2^\delta)$, one can find all integer pairs (x_0, y_0) such that $p(x_0, y_0) = 0$, $|x_0| \leq X$, and $|y_0| \leq Y$.

Therefore, in our scheme the KDC selects all the keys k_is so that: $|k_i - k_j| > n^{1/4}, \forall i \neq j, \; i, j \in \{1, ..., m+1\}$.

Thanks to this two constrains that we force ib the selection of the keys k_is allow them to be adequately large and at the same time separated enough from each other, so that our solution is immune against lattice-based attacks founded on the above Coppersmith theorems.

4.1.1. Attack models

The attack models for group key management schemes, depending on the nature of the attacker, can be generally divided into insider attacks and outsider attacks. In the first class we consider attackers who are legitimate users that want to curiously find the key of another user in the group. The attackers may form subgroups, within the whole group, aiming at one or more users. In this scenario, we find the so-called collusion attacks, which are going to be analysed latter in this section. Another insider attack scenario takes place when a new member tries to determine the previously used group keys. For this scenario we demand from our scheme to fulfil the so-called backward secrecy property.

In the second class of attacks, we face outsiders who are not legitimate users. Since the KDC makes a broadcast of the information that allows legitimate users to compute the group key, outsiders can access it and try to obtain the keys of legitimate users. Generally, the strength of such attacks depends on the hardness of the underlying mathematical problem (which in our case has been proved to be NP-Hard). However, in this class of attacks, given the nature of the scheme, we have the case of users that were legitimate in the past, which have their old keys and try to access the new content. For these attacks we demand from our scheme to fulfil the so-called forward secrecy property.

Both backward and forward secrecy, that are going to be discussed in the following section, guarantee that the group key is available only to the legitimate users. In other words, the new users cannot derive the old keys and the past users cannot derive the new ones.

4.1.2. Backward and forward secrecy

Theorem 3. *The proposed scheme provides backward secrecy.*

Proof. Let's assume that user U_N is accepted to join group $\mathcal{G}$, then we must prove that he cannot find the previous key of the group. U_N has access to the new group key K'_G, for which we have $K'_G = h(K_G)$. Therefore, U_N has to calculate the preimage of K'_G for a given secure hash function h, which is computationally infeasible. □

Theorem 4. *The proposed scheme provides forward secrecy.*

Proof. Let's assume that user U_a wants to leave group $\mathcal{G}$ with users $U_1, ..., U_m$. All users know either:

- The initial triplet $(k_i, n, P(x))$, where k_i is their key, if they were on the initial formation of group $\mathcal{G}$ and the current group key K_G.
- The current group key K_G and their own key k_r, if they joined afterwards.

If U_a does not belong to the initial formation of group $\mathcal{G}$, he cannot find the new group key K'_G as he doesn't have any other information. Alternatively, if U_a belongs to the initial formation of group $\mathcal{G}$ and intercepts $P'(x)$, he cannot calculate K'_G. Clearly, $P'(k_m) \neq K'_G$. Moreover:

$$d(x) = P(x) - P'(x)$$

$$= \left((x-k_S)\prod_{i=1}^{m+1}(x-k_i)+K_G\right)-\left((x-k'_S)\prod_{i=1}^{m}(x-k_i)+K'_G\right)$$

$$= [(x-k_S)(x-k_a)-(x-k'_S)]\prod_{i=1}^{m}(x-k_i)+(K_G-K'_G)$$

From the equation above we see that the presence of the two salting polynomials $(x-k_S)$ and $(x-k'_S)$ make common multiplier of $\prod_{i=1}^{m}(x-k_i)$ non-linear. Hence, if U_a evaluates $d(k_a)$, he will not gain any information.

In the case where the KDC decides to make multiple deletions (to avoid the unnecessary growth of the degree of the polynomial), the calculations are almost the same, with the only difference that the common multiplier is of smaller degree.

Therefore, U_m cannot find the new value of K'_G, so the scheme provides forward secrecy. □

Remark 1. *In the above proof, the utility of the salting factor $x-k_S$ becomes apparent. Also, it is important to emphasize the need for rekeying k_S after each user removal as well (if it wasn't rekeyed the polynomial would be linear).*

4.1.3. Security against collusion attacks

In the worst case scenario, we would have $m-1$ from the m members of $\mathcal{G}$ cooperating to reveal the key of the m^{th} member. Without loss of generality, we assume that the $m-1$ members are the first ones and they try to determine the key of user U_m. Clearly, the colluded members can calculate the polynomial:

$$G(x)=\prod_{i=1}^{m-1}(x-k_i)\ mod\ n$$

Therefore, in order to recover U_m's key, they have to factor the polynomial:

$$g(x)=\frac{P(x)-K_G}{G(x)}=\frac{\prod_{i=1}^{m+1}(x-k_i)}{\prod_{i=1}^{m-1}(x-k_i)}=(x-k_S)(x-k_m)$$

It can be observed that $g(x)$ is a univariate polynomial of degree two. Since the values k_m and $x-k_S$ have not been disclosed to the colluded members of the group, they must find the roots of a univariate polynomial modulo a composite integer n, for which the factorization is not known (As we already stated above, this is an NP-hard problem). With proper transformations, the problem can be reduced to finding the square root of a number modulo n. However, this problem, as proved by Rabin, is equivalent to factoring n [23], hence the key of U_a is secure against collusion attacks from the members of G.

Remark 2. *The importance of the salting factor $x-k_S$ and the rekeying of values becomes apparent once more. If the salting factor was not used, then the recovery of the key of user U_m would be trivial.*

In a different collusion attack scenario, all users decide to recover the key of user U_v that leaves the group. To do this they decide to disclose their keys to each other. If there was no rekeying of k_S, then the key of U_v could be easily recovered by the rest of the users. To understand this, the attackers could calculate:

$$g'(x) = \frac{P(x) - K_G}{P'(x) - K'_G} = \frac{\prod_{i=1}^{m+1}(x - k_i)}{\prod_{i=1}^{m}(x - k_i)} = \frac{(x - k_S)(x - k_v)}{(x - k_S)} = (x - k_v)$$

Thus recovering the k_v. By rekeying the salting polynomial, the scheme not only provides backward and forward secrecy, but provides secrecy to the leaving users, by protecting their keys.

4.2. Experimental results

An implementation of our proposed scheme has been made in Sage 5.9. The experiments were carried out on a machine with Intel® Core™ i7-2600 CPU at 3.40GHz processor with 16GB of RAM, running on 64 bit Ubuntu GNU/Linux kernel 3.2.0-29. The results that are presented represent the average timings of 100 experiments for groups of up to 100,000 users.

From the conducted experiments, it can be observed that the major time constraint resides in the initialization of the protocol. Notwithstanding, the performance is remarkable, for example it takes around 9 seconds to generate the polynomial for 100,000 users. It has to be noted that this process can be parallelized, which in practical applications leads to much faster implementations. Figure 1 illustrates the performance of the polynomial generation procedure for different key sizes. The generation of the RSA-like modulo n is in all cases 1024 bits, so the cost is independent of the users and the average cost is around seconds.

Adding or removing a user in our scheme is very fast, the cost is in the scale of milliseconds, even for thousands of users. To generate a new key in our experiments we have used the SHA-2 function, so the time in Figure 2 is the sum of hashing and caching the new polynomial. The latter has been added here as in many cases the KDC may choose to calculate the polynomial, in order to decrease the cost of calculating the new polynomial when someone decides to leave the group. The performance of this process can be seen in Figure 3.

Figure 4 illustrates the amount of information that has to be multicast/distributed from the KDC, depending on the number of legitimate users. Since the KDC has to broadcast the polynomial and the modulo n integer, most of which are numeric data, we compressed the information using the bz2 library, which manages to shrink the file with a factor around 2.

Finally, in Figure 5, the time needed for a client to recover the key is illustrated. Quite remarkably, the time even for keys of thousands of users can be considered of the scale of decrypting an AES message.

All the figures indicate that the amount of time required for deleting or creating a user, the amount of information to multicast, as well as the size of the key to multicast, is almost linear with the number of users, making it efficient for practical applications.

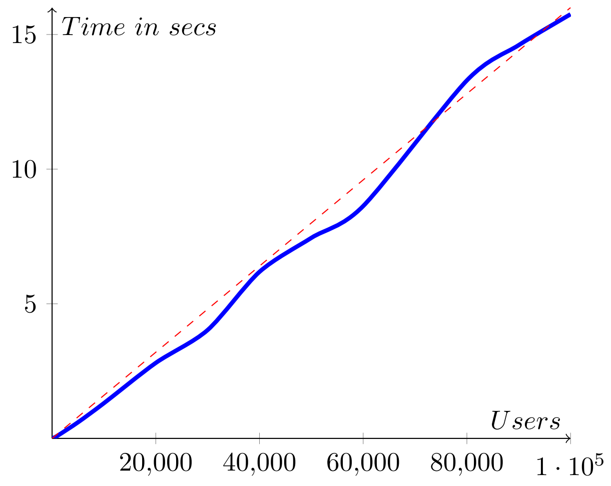

Figure 1: Time to create polynomial

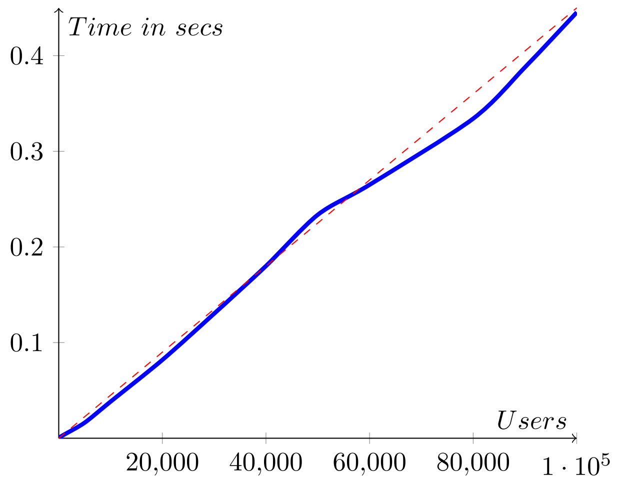

Figure 2: Time to add one user. This is hashing the old key and cache the new polynomial.

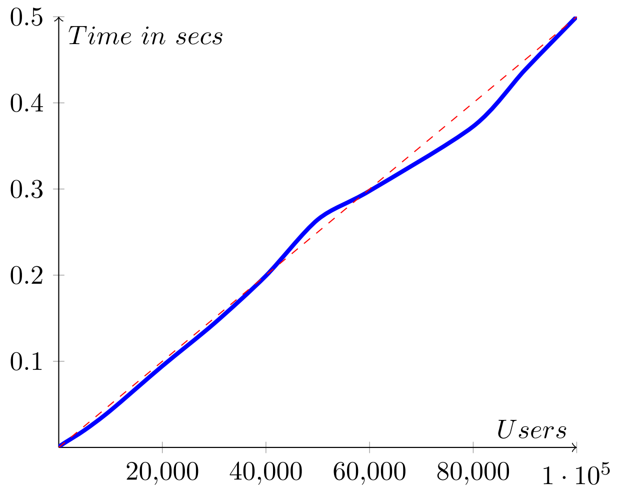

Figure 3: Time to remove one user

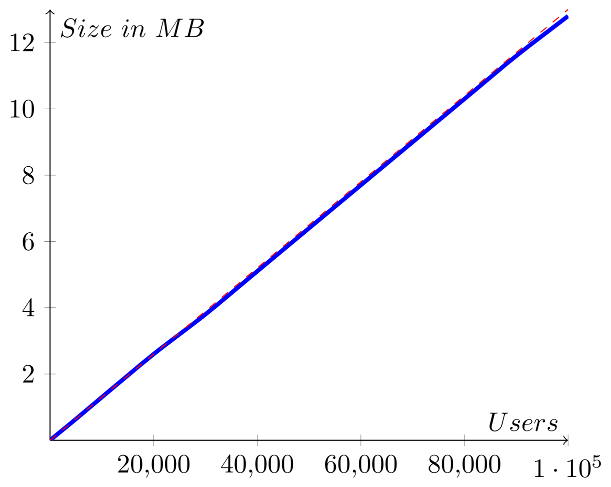

Figure 4: Size of key to multicast in MB.

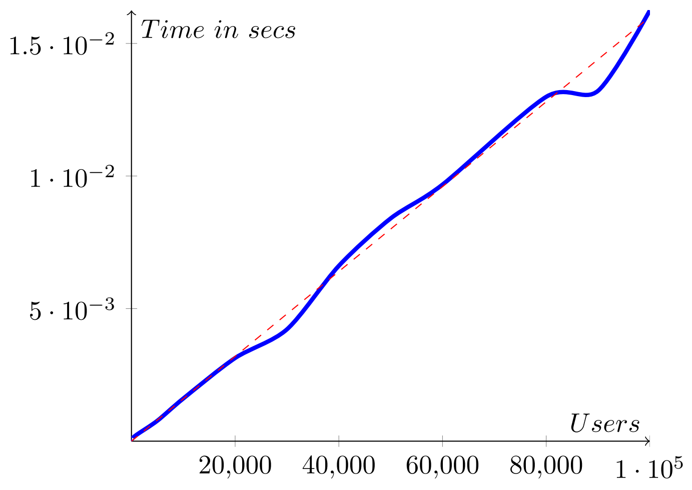

Figure 5: Time needed for client to recover the key

5. Conclusions

Group key management is a complex problem that is gaining increasing importance. Since users tend to gather in groups, it has been shown that to secure the communications within those groups it is much more efficient to share a common traffic encryption key than maintaining a key for each pair of users. However, managing a group key is not straightforward specially when the group is very dynamic (i.e. users join and leave the group frequently). In this case, the group key has to be changed frequently so as to guarantee security properties such as backward secrecy and forward secrecy.

In this article we have briefly described the state of group key management. We have recalled a scheme by Piao et al. that is efficient but has been broken because the underlying mathematical problem, in which the scheme is based, can be solved in polynomial time. Inspired by the lessons learnt from the Piao's scheme, we have proposed a novel scheme that has been proved to be efficient and secure since it is based on a mathematical problem that is know to be NP-Hard. Our proposed scheme has proven to guarantee forward and backward secrecy and collusion freedom.

In addition to the theoretical proofs, we have discussed the properties of our scheme and we have shown the experimental results of a real implementation to prove its efficiency and feasibility in practice.

Disclaimer and acknowledgments

Agusti Solanas is partly supported by the Government of Catalonia under grant 2009 SGR 1135, by the Spanish Government through projects TIN2011-27076-C03-01 "CO-PRI-VACY" and CONSOLIDER INGENIO 2010 CSD2007-

00004 “ARES”, by the European Comission under FP7 projects “DwB” and “Inter-Trust”, and by La Caixa under program RECERCAIXA 2012.

Agusti Solanas is with the UNESCO Chair in Data Privacy, but he is solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organisation.

Bibliography

[1] Y. Challal, H. Seba, Group key management protocols: A novel taxonomy, International Journal of Information Technology 2 (1) (2005) 105–118.

[2] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung, Perfectly-secure key distribution for dynamic conferences, in: Advances in cryptology-CRYPTO’92, Springer, 1993, pp. 471–486.

[3] A. T. Sherman, D. A. McGrew, Key establishment in large dynamic groups using one-way function trees, Software Engineering, IEEE Transactions on 29 (5) (2003) 444–458.

[4] C. K. Wong, M. Gouda, S. S. Lam, Secure group communications using key graphs, Networking, IEEE/ACM Transactions on 8 (1) (2000) 16–30.

[5] S. Mittra, Iolus: A framework for scalable secure multicasting, in: ACM SIGCOMM Computer Communication Review, Vol. 27, ACM, 1997, pp. 277–288.

[6] R. Molva, A. Pannetrat, Scalable multicast security in dynamic groups, in: Proceedings of the 6th ACM conference on Computer and communications security, ACM, 1999, pp. 101–112.

[7] M. Burmester, Y. Desmedt, A secure and efficient conference key distribution system, in: Advances in Cryptology-EUROCRYPT’94, Springer, 1995, pp. 275–286.

[8] Y. Kim, A. Perrig, G. Tsudik, Tree-based group key agreement, ACM Transactions on Information and System Security (TISSEC) 7 (1) (2004) 60–96.

[9] S. Rafaeli, D. Hutchison, A survey of key management for secure group communication, ACM Computing Surveys (CSUR) 35 (3) (2003) 309–329.

[10] Y. Piao, J. Kim, U. Tariq, M. Hong, Polynomial-based key management for secure intra-group and inter-group communication, Computers & Mathematics with Applications.

[11] A. A. Kamal, Cryptanalysis of a polynomial-based key management scheme for secure group communication, International Journal of Network Security 15 (1) (2013) 68–70.

[12] N. Liu, S. Tang, L. Xu, Attacks and comments on several recently proposed key management schemes, IACR Cryptology ePrint Archive 2013 (2013) 100.

[13] E. R. Berlekamp, Factoring polynomials over large finite fields, Mathematics of Computation 24 (111) (1970) 713–735.

[14] D. G. Cantor, H. Zassenhaus, A new algorithm for factoring polynomials over finite fields, Mathematics of Computation (1981) 587–592.

[15] V. Shoup, On the deterministic complexity of factoring polynomials over finite fields, Information Processing Letters 33 (5) (1990) 261–267.

[16] J. von zur Gathen, D. Panario, Factoring polynomials over finite fields: A survey, Journal of Symbolic Computation 31 (1-2) (2001) 3 – 17.

[17] D. A. Plaisted, New np-hard and np-complete polynomial and integer divisibility problems, Theoretical Computer Science 31 (1) (1984) 125–138.

[18] D. Plaisted, Some polynomial and integer divisibility problems are np-hard, SIAM Journal on Computing 7 (4) (1978) 458–464.

[19] S. Gao, M. van Hoeij, E. Kaltofen, V. Shoup, The computational complexity of polynomial factorization, American Institute of Mathematics (2006) 364.

[20] W. Diffie, P. Oorschot, M. Wiener, Authentication and authenticated key exchanges, Designs, Codes and Cryptography 2 (2) (1992) 107–125. doi:10.1007/BF00124891.

[21] D. Coppersmith, Small solutions to polynomial equations, and low exponent rsa vulnerabilities, Journal of Cryptology 10 (4) (1997) 233–260.

[22] D. Coppersmith, Finding a small root of a bivariate integer equation; factoring with high bits known, in: Advances in cryptology-EUROCRYPT'96, Springer, 1996, pp. 178–189.

[23] M. O. Rabin, Digitalized signatures and public-key functions as intractable as factorization, Tech. rep., Massachusetts Institute of Technology (1979).