A secure multicast protocol based on pairings on elliptic curves

J.A. Alvarez-Bermejo[1], J.A. Lopez-Ramos[2]

[1] Department of Informatics, University of Almeria.
04120, Almerıa, Spain. e-mail: jaberme@ual.es
[2] Department of Mathematics, University of Almeria.
04120, Almerıa, Spain. e-mail: jlopez@ual.es

Abstract—The aim of this paper is to introduce a protocol for multicast distribution of secrets providing perfect forward and backward secrecy. The protocol is based on pairings on elliptic curves and has low cost communications and key storage. It is shown how this protocol improves some alternatives existing for real time communications.

Keywords—Information security, multicast communications, elliptic curves cryptography, pairings on elliptic curves, hash function.

1. Introduction

Information security is an ever-present concern that is gaining more and more attention as digitalization of information and use of Internet increases. Numerous privacy protocols, standards and applications exist in order to keep information away from unauthorized persons, as well as to authenticate its author. Traditional cryptosystems are very effective when trying to provide confidentiality to communications between two parties and are usually known as unicast communications. However, unicast secure protocols cannot be used in some situations mainly due to the nature of the information to be transmitted since it can collapse the server or the network with the big amount of data being transmitted. Examples of these are pay-per-view IPTV or P2PTV, private multiconferences, or any private service that involves several participants or clients. Multicast communications allow a host to simultaneously send information to a set of other hosts, avoiding the establishment of point-to-point connections with all of them.

Key management is a main issue in secure multicast. Efficiency in the key management, including key storage and refreshment, and perfect forward and backward secrecy are required. The idea is that a new user should not be able to decrypt the contents before joining the multicast group and a former user should not access the encrypted information after leaving, trying to minimize key storage and communication overcomes. Depending on how key distribution and management are carried out, secure multicast schemes are divided into centralized and distributed. Centralized schemes depend directly on a single entity to distribute every cryptographic key and this is the setting we are considering when

Manuscript received November 7, 2013; revised December 5, 2013.

proposing our protocol.

Let us recall some very well-known centralized secure multicast protocols. A group of these follows the scheme proposed in [26]. This protocol, known as Hierarchical Tree Approach, namely HTA, uses a logical tree arrangement of users in order to facilitate key distribution. In this approach the number of messages and the storage requirements are both logarithmic in the number of members in the multicast group. Other key tree approaches and extensions are LKH [27], LKH++ [9], OFT [21] or ELK [19].

The so-called Secure Lock is an alternative approach to the tree arrangement. In [8] the authors make a computational treatment of the problem and propose a solution based on the Chinese Remainder Theorem. However, it presents a drawback concerning the inefficient computations required at the Key Server side at every rekeying operation: the computing time needed quickly becomes huge when the number of members grows [15]. A divide-and-conquer extension to Secure Lock is proposed in [20] to address this matter. It combines the Hierarchical Tree Approach and the Secure Lock: members are arranged in a HTA fashion, but Secure Lock is used to refresh keys at every tree level. Thus the number of computations required by Secure Lock is reduced.

Another computational approach is introduced in [16] with the particular application on Pay-TV but extensible to any other secure multicast setting. It is based on an interpolator polynomial over some secret values in a finite field held by the authorized users and makes use of a hash function. The main drawback is that the implementation of this protocol needs changing the hash function used upon every rekeying due to security matters in the definition of the aforementioned polynomial. Similar protocols for secure multicast given by a polynomial equation based on a multiplicative group can be found in [24] and [25].

More recently, in [18], the authors introduce a method for distributing keys based on the Extended Euclid's Algorithm. The behavior of this protocol shows to be better than the others concerning key storage and re-keying messages. However, the length of a message grows linearly with respect to the sum of the every user's secret key length.

The aim of this paper is to give a new computational solution similar to that of [18], [16], [24] and [25] with the same good behavior concerning re-keying and reducing the length of the messages by means of a protocol based on elliptic curve cryptography. We focus our work in the security analysis, providing mathematical proofs of the equivalence problems to be solved in case of attacking the protocol. We also give an estimation on the required time for re-keying. Section 2 is devoted to introduce the protocol and analyze its security. In Section 3 we make a comparison with other centralized schemes concerning key storage, number and length of re-keying messages and provide an estimation of the re-keying time. Finally, Section 4 concludes the paper. An extended abstract of this work appears in [1].

2. The proposed scheme

Our aim in this section is to introduce a protocol based on elliptic curves following the idea of [8], [16] and [18]. We construct a re-keying message that depends on some private information, that we will call a *ticket*, held by every authorized user, predistributed at the moment of subscribing the service, and different for each one, in such a way that only those users owning this private information are able to decrypt the re-keying message and thus get the distributed new session key. Our protocol will make use of pairings on elliptic curves. So let us start by recalling its definition.

Let $G_1,\ G_2$ and G_3 be abelian groups. A pairing is a non-degenerate bilinear map $e : G_1 \times G_2 \to G_3$, i.e.,

$$e(g_1 + g_2, h) = e(g_1, h)e(g_2, h),\ \text{for } g_1,\ g_2 \in G_1$$

$$e(g, h_1 + h_2) = e(g, h_1)e(g, h_2), \text{for } h_1,\ h_2 \in G_2$$

for all $g \neq 1,$ there is $x \in G_2$ such that $e(g, x) \neq 1$

for all $h \neq 1,$ there is $x \in G_1$ such that $e(x, h) \neq 1$

Here, we denote by 1 the identity element of group G_3. In the case of an elliptic curve, G_1 and G_2 are both the abelian group constituted by the points of the elliptic curve and G_3 is usually an extension of the basis finite field. An example of these mappings is the well-known Weil pairing. Other examples can be constructed from the Tate pairing. Their use in public cryptography is quite extended not only for constructing new encryption protocols as Identity Based encryption or signature (for example cf. [4, Chapter X]), but also even for the cryptanalysis as the well-known “pairing attack” on the the Elliptic Curve Discrete Logarithm Problem (ECDLP), by reducing it to the Discrete Logarithm Problem (DLP), [17].

Our protocol will use some predistributed information for every authorized user that will consist of a pair (x_i, P_i), where P_i is a point in an elliptic curve E over a finite field $I\!\!F_p$ and x_i is a scalar in some extension of $I\!\!F_p$ depending on the pairing over the group of points of E that will be used to build the re-keying messages. We will encrypt the session key using every authorized user’s ticket at the moment of re-keying.

The target scenario is the following: private communications must be established within a restricted group. There is a central server, that manages the key management issues. From now on, we will refer to the server as *Key Server*, and to the clients as *members* or *users*. Depending on the nature of the service, communications can be either one-to-many or many-to-many.

So let $e : E(I\!\!F_p) \times E(I\!\!F_p) \to I\!\!F_{p^k}$ be a pairing on the group of points of E. In the initial setup, each authorized user i is assigned a different element $x_i \in I\!\!F_{p^k}$ and a point $P_i \in E$. The pair (x_i, P_i) constitutes the *ticket* and both the elliptic curve E and the pairing $e(-,-)$ are public.

When the Key Server wants to refresh the session key, it makes the following steps to build a re-keying message:

The Re-keying Protocol.

1 The Key Server selects a secret value: $K \in I\!\!F_{p^k}, H \in E$ and an integer n such that $1 < n < ord(H)$.
2 The Key Server selects a point $P_{m+1} \in E$ and a random number $x_{m+1} \in I\!\!F_{p^k}$.
3 The Key Server computes the sum of all the points held by the users $P' = \sum_{i=1}^{m} P_i \in E$. P is kept private in the Key Server.
4 The Key Server computes the sum $P = P' + P_{m+1}$.
5 The Key Server computes the following values: $e(H, P)$, $e(nH, P)$, $\alpha_i = e(H, P - P_i)$ for every $i = 1, \ldots, m + 1$.
6 The Key Server computes the scalar $r = K + e(nH, P)$.
7 The Key Server computes the interpolator polynomial

$$p(x) = n + \prod_{i=1}^{m+1} (x - (x_i \alpha_i) \in I\!\!F_{p^k}[x]$$

8 The re-keying message $M = (r, H, e(H, P), p(x))$ is then broadcasted.

Remark At this point we note that we are assuming p greater than n and so, the constant term n of the interpolator polynomial represents the image of the integer n in $I\!F_{p^k}$. In case p is less than n, for instance a binary curve, the protocol is still valid, although in that case we would increase significantly the amount of information to be sent since if we represent n as $n = \sum_{i=1}^{j} n_i b^i$, then we should send j polynomials to deliver every n_j.

We also point out that the fact that the Key Server generates a new point of the curve in every re-keying messages makes impossible for an external observer to determine if the set of users changes after two consecutive rekeyings even if P is made public.

Proposition 2.1. Every user i is able to decrypt K from M using her ticket (x_i, P_i).

Let us illustrate this with the following example.

Example Let us consider the elliptic curve $E : y^2 = x^3 + 11$ over $I\!F_{31}$. Then the group of points $E(I\!F_{31})$ has structure $I\!F_5 \times I\!F_5$ and it is generated by $S(2, 9)$ and $T(3, 10)$. Let us consider now the Tate pairing $e(P, Q) =< P, Q >_l= f_{l,P}(S + R)/f_l(R)$, where $l = 5$. If we assume $R = T$, then we get that $e(S, T) = 20 \cdot 10^{-1} = 2$. From this, we get easily the following:

Let us assume users u_1, u_2 and u_3 holding respectively $(x_1, P_1) = (3, 2T)$, $(x_2, P_2) = (5, 3T)$ and $(x_3, P_3) = (11, 4T)$ as predistributed tickets.

Then the Key Server chooses $K = 7$ as session key and $H = S$. The sum of all the points is $P = 9T$ and so, $e(H, P) = e(S, T)^9 = 16$. The Key Server now selects $n = 8$ and thus $e(8H, P) = e(H, P)^8 = 4$. Since $e(H, P - P_1) = 4,\ e(H, P - P_2) = 2$ and $e(H, P - P_3) = 1$, we get that $p(x) = 8 + \prod_{i=1}^{m}(x - x_i\alpha_i) = 21 + 21x + 29x^2 + x^3$. Then the re-keying message is

$$M = (11,\ (2, 9),\ 16,\ 21 + 21x + 29x^2 + x^3)$$

Once user u_2 receives M then she computes $e(H, P_2) = 8$ and thus $\alpha_2 = 16 \cdot 8^{-1} = 2$. Now substituting, we get that $p(5 \cdot 2) = 8$ and so $K = 11 - 16^8 = 7$.

Assume that $M = (r, H, e(H, P), p(x))$ is a re-keying message for a given key K. If c is any positive integer and we consider $e(H, P)^c$, then the message $M' = (e(H, P)^c \cdot r, H, e(H, P), p(x) + c)$ is an encryption message for $K \cdot e(H, P)^c$, i.e., given a re-keying message M, an attacker is able to produce an encryption M' such that the two corresponding encrypted secrets are related, thus our protocol does not provide *non-malleability* (cf. [10], [11]).

However, we can show the *indistinguishably* of the encrypted information (cf. [12]), i.e., an adversary is not able to learn anything on the plain-text from the encrypted one.

Theorem 2.2. Let us assume an attacker $A = (A_1, A_2)$, where A_1 is an algorithm that provides the discrete logarithm in $I\!F_{p^k}$. The re-keying protocol provides indistinguishability for a brute force attack carried out by A.

Traditional public key cryptosystems use a public key to encrypt a message and its corresponding private key to decrypt it. In the above protocol we are using some public information and some other private parts that are predistributed between legal users, the so-called tickets, to encrypt it and, as we showed in Proposition 2.1, these use their tickets to get the encrypted information. Thus the main interest when attacking this protocol will be to get some private info that allow the attacker to access the distributed information.

Suppose that somehow the adversary gets a valid K corresponding to a re-keying message $M =$

$(r, H, e(H,P), p(x))$ at some time. Then she could factorize the polynomial $p(x) - n = \prod_{i=1}^{m}(x - x_i\alpha_i) \in \mathbb{F}_{p^k}[x]$. Thus, she would get the products $x_i \cdot \alpha_i,\ i = 1, \ldots, m$, that do not reveal any information on the tickets (x_i, P_i) since there is not a unique factorization in $\mathbb{F}_{p^k}$ of such elements, even in case the Key Server also makes P public.

In [16] the authors use an interpolator polynomial on the users' tickets to distribute the session key. In order to avoid the factorization attack above mentioned, they need to use a hash of these tickets and so the the polynomial is of the form $f(x) = K + \prod_{i=1}^{m}(x - h(x_i))$, where x_i denotes the users' tickets. However, they need to change somehow the hash of the ticket in every refreshment, which leads to consider some padding or similar that the authors do not cite in their work. In our case, the polynomial changes for every re-keying message even if the audience remains equal since values interpolating the polynomial depend on two random points H and P_{m+1} of the curve E. This argument also applies to the protocols introduced in [24] and [25] since an attacker does not need to know the private information held by a user to get the new key. She just need to know one of the factors that form the considered product added to the new distributed key, that can be got, as above was pointed out, by factoring the value $M - K$, where M is the information that the Key Generator Center distributes and K is the corresponding encrypted key to this information M. Actual computing capabilities and results given in [5] and [22] imply that size of the integers should be increased considerably with respect to the considered cases in [25] (at least 1024 bits length). In this case the attack could be be avoided by changing the public value p (considered prime to compute the private secrets) with every re-keying.

Now let $A = (A_1, A_2, A_3)$ be an attacker where A_1 is an algorithm that provides the discrete logarithm in $\mathbb{F}_{p^k}$ and A_2 is an algorithm that outputs the factorization of any polynomial in $\mathbb{F}_{p^k}[x]$.

Theorem 2.3. Let $A = (A_1, A_2, A_3)$ be an attacker knowing two couples of plain-encrypted text (M_1, K_1) and (M_2, K_2). Then she will be able to compromise a valid ticket whenever she is able to determine the owning of tickets.

From the proof of the above Theorem (cf. Appendix A) we get that security of our cryptosystem is based in two main facts:

- Strong assumptions on the computation of the Discrete Logarithm Problem (DLP).
- Difficulty for an attacker to determine the owning of tickets, or at least, if two products of the form $x_i \cdot \alpha_i$ correspond to a same user.

In the first case, it is clear that increasing of computing capabilities of actual computers and works as [5] and [22] have provoked that cryptosystems based on DLP are no longer recommended. Therefore, let us introduce a slight modification in order to enhance security of the proposed protocol.

So let $h(-) : \mathbb{F}_{p^k} \to \mathbb{F}_{p^k}$ be a hash one-way function. Then we modify the definition of the polynomial appearing in step 7 of our re-keying protocol. Now

$$p(x) = n + \prod_{i=1}^{m+1}(x - h(x_i\alpha_i)) \in \mathbb{F}_{p^k}[x]$$

To recover the session key, the user just operates in the same way as in the previous protocol, but she has to compute $h(x_i\alpha_i))$ previously to evaluation of $p(x)$.

Then we observe that the attacker has not just to solve DLP, but also to determine owning of tickets and break a one-way function.

3. Comparisons with other schemes

In this section, we will compare the proposed secure multicast protocol with other well-known and currently used alternatives cited briefly through this paper. More precisely we will compare our proposal with HTA ([26]), LKH++ ([9]), OFT ([21]), Secure Lock ([8]), Secure Lock + HTA ([20]), CAS ([16]) Euclides ([18]) and protocols introduced in [24] and [25].

To do so we will refer to key storage, broadcasted re-keying messages per join and leave operations and computational requirements. So let n be the number of users in any of the considered schemes and, for those with a tree arrangement (HTA, LKH++, OFT and Secure Lock + HTA) let d and h be the tree degree and depth respectively.

In Table 1, we can observe a comparison of our protocol with some other algorithms cited above. Protocols introduced in [24] and [25] present a similar behaviour to CAS and thus they are referred as similar in this table.

From [7] it can be checked that the number of keys stored in the key server for all these tree based solutions are either $d^h - 1$ or $2n - 1$. In the case of Secure Lock and Euclides this number is n, (cf. [8], and [18] respectively). From the definition of our protocol, denoted by *Pairings* in the table, it can be derived immediately that this number equals the number of all computational approaches, i.e., n and thus, as the number of user grows, this shows a better behavior than those based on a tree arrangement. Concerning keys stored in every member the results are similar. Those with a tree arrangement behaves again worse: $h+1$ is this number for all those tree-based and only one key is stored at the member side for all those with a computational approach. The main reason is that all those based on a tree arrangement are state-full, i.e., re-keying messages contain modifications on the previous keying states and thus, users must be aware of all these re-keying operations since their arrival. In the case of stateless protocols, the session key is distributed through a message that depends just on every authorized user's private and predistributed information which means that no other extra information is required. Thus, our protocol has the same requirements as those with a computational approach, just one key, namely (x_i, P_i), is stored. The number of messages per join or leave operations coincides in every considered case. For the tree-based schemes these are $(d-1)h$ for HTA, $h+1$ for LKH++ and OFT and h for the Secure Lock + HTA. In the computational approaches this number is simply 1, including our case. This is due again to the fact that they are stateless. Up to this moment we can see that computational approaches are considerably ahead of those with a tree arrangement.

However, problems of computational approaches arise with computational requirements. As noted in the introduction, Secure Lock presents strong problems with efficiency as the number of user grows ([15]) due to the big number of calculations to be developed when preparing a re-keying message. CAS presents a similar problem. As noted by the authors in [16] the hash function used to construct every re-keying message needs to be changed with every message, which may suppose strong computational and hardware requirements. The solution given in [18] and [25] avoid this computational or hardware requirements, but in order to avoid an active attack as explained in the previous section, the length of the re-keying messages may grow in an undesirable manner. In that case, such an attacker, for instance a legal user, has access to a multiple of the product of all the tickets in use. Therefore,

	HTA	LKH++	OFT	SecLock	SecLock+HTA	CAS & similars	Euclides	Pairings
Keys stored in server	d^h-1	$2n-1$	$2n-1$	n	d^h-1	d^h-1	n	n
Keys stored in member	$h+1$	$h+1$	$h+1$	1	$h+1$	$h+1$	1	1
Broadcast messages per join	$(d-1)h$	$h+1$	$h+1$	1	$h+1$	$h+1$	1	1
Broadcast messages per leave	$(d-1)h$	$h+1$	$h+1$	1	h	$h+1$	1	1
State-full/Stateless	State-full	State-full	State-full	Stateless	State-full	State-full	Stateless	Stateless

TABLE 1
Comparison with other protocols

we would have to ensure that a factorization attack is not possible by enlarging the length of these tickets, which derives that the length of the re-keying messages grows linearly with this length and so, we would be forced to reduce the number of users by grouping them in small groups in order to keep both efficiency and security. However, we again refer to attacks appearing given in [5] and [22] that make less recommendable those cryptosystems based on DLP or integer factorization.

From the analysis of the protocol that we are introducing we can observe that this new protocol improves all the situations above described. Firstly, our re-keying messages could be viewed as another version of the polynomial used in [16], [24] and [25]. The polynomial used in our case is of the form $p(x) = K + \prod_{i=1}^{m}(x - h(x_i\alpha_i))$ and as can be observed, the values where this polynomial interpolates change with every re-keying message since α_i depend on a different arbitrary point of the elliptic curve and therefore we do not need to change the hash function. We are changing the hash using every user's same key at a determined moment.

Secondly, the length of a re-keying message is mostly due to the polynomial, but in this case, the length of this polynomial over a finite field of standard size in these situations would produce, even when the audience is large, a message whose length is perfectly affordable.

Finally, let us consider computational requirements that use to be the main disadvantage of computational approaches respect to those with a tree arrangement of users. Although arithmetic on elliptic curve provides a much bigger security level considering a shorter key length and shows to be very appropriate even for light devices due to its small computational requirements, construction of linear pairings on elliptic curves are not so easy to manage and compute. However, there exists many works on linear pairing computation with cryptographic applications as [6] and [13]. Many other authors are nowadays developing more and more efficient software and hardware implementations of most popular and used pairings such as Weyl or Tate pairings as can be checked in [3], [14] or [23]. Computation timings given in these works show to be quite efficient.

Concerning computation complexity for computing the polynomial used in the protocol, it can be checked that computing such an interpolator polynomial is equivalent to solve a system of congruences over a finite field, which is done through the well-known Chinese Remainder Algorithm, CRA. Concerning this issue we showed in [2] that it is possible to develop a parallel algorithm that outputs a solution within a reasonable time.

Given a system of equations in congruences, the problem is recursively divided into a problem of a certain and predefined size, forming the base case. These are then parallelized and solved. Each solved

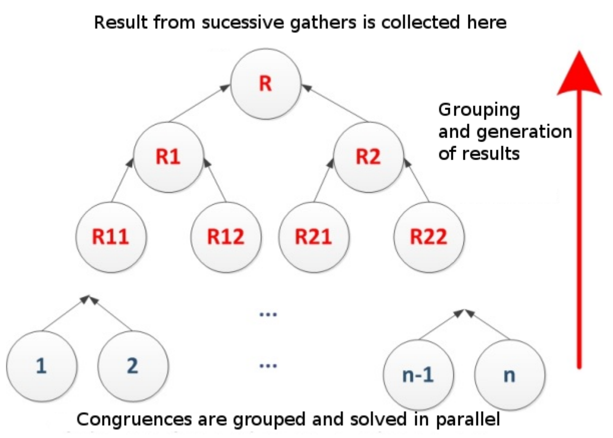
Result from sucessive gathers is collected here
R
R1
R2
R11
R12
R21
R22
Grouping and generation of results
1
2
...
n-1
n
Congruences are grouped and solved in parallel

Fig. 1. Algorithm that combines both methods

case means, as CRA establishes, a new equation that is propagated towards its father leave in the tree and composed with the equations of its brother leaves that constitutes a new system of equations in congruences. Figure 1 depicts our method.

Next, we show some of the results obtained in a mid-range common laptop computer, that has a GPU graphic card. Details of the architecture used are: DDR3 1333 Mhz 2GB x 2 (PC3-10700) ram modules. The processor was an intel core i7 model 920 operating at 2.67GHZ with four cores and eight virtual processors, on am Asus P6T mother board. The GPU was an NVIDIA GForce GTX 260 Asus ENGTX260 GL+. The base case was set up to six equations and in table 2 we show the obtained results from a number of 10 to 1000 equations in congruences on 256 bit-length primes.

In table 2 tests were run to select the most appropriate prime number implementation, native prime number library refers to the implementation offered by the platform (.NET) whereas GNU MP refers to GMP which is a free library for arbitrary precision arithmetic, operating on signed integers, rational numbers, and floating-point numbers. There is no practical limit to the precision except the ones implied by the available memory in the machine GMP runs on. It has been shown that the latter library is the one that offers the best fit for our algorithm.

Size	Native prime number library (ms)	GNU MP library (ms)
10	0.35	1.33
50	5.80	3.15
100	18.27	3.71
250	98.05	7.25
500	365.37	18.20
750	796.49	22.20
1000	1395.62	37.35

TABLE 2
256 bits efficiency

Size	Pure parallel version (ms)	Efficiency version (ms)
10	0.27	1.79
50	0.34	2.48
100	0.43	3.33
250	0.48	3.25
500	0.38	4.36
750	0.51	4.09
1000	0.42	4.38
2000	0.49	5.78
4000	0.54	8.75
6000	0.69	11.09
8000	0.83	13.82
10000	0.91	15.72
15000	out of memory	21.51
20000	out of memory	27.77

TABLE 3
Results using 32 bits and up to 20000 equations

Table 3 and using the GMP arithmetic library (as exposed in table 2, shows the results of the two versions of our CRA implementation, the first one (pure parallel version) consists in running in parallel all the equations of the congruence system. It is a fast method but it requires all the equations to be in memory, so this solution does not scale properly as it runs out of memory from 15000 equations in advance. The efficiency version is that depicted in Figure 1 where the equations are operated by turns and their results are aggregated and passed to their parent nodes in the tree. The efficiency version is the version that scales properly for huge congruence systems.

4. Conclusions

We have introduced a protocol for secure multicast distribution of secrets based on parirings on elliptic curves. Re-keying messages depend, at any time, on the users' private information. This private information allows to construct a unique message for the whole multicast group and every member of this group can decrypt it by using this private information, which is different for each one. We have shown its security against several passive and active attacks and show how it behaves better than any other similar secure multicast protocol concerning key storage, number and length of re-keying messages and computational requirements. Perhaps the only disadvantage of our protocol with respect to any other concerns to huge audiences. When trying to reach an audience formed by millions, the length of the message could be excessive to consider a computational approach. Then a tree arrangement of users as that given in [20] for Secure Lock could be appropriate. In this case, a modification of our protocol in same way behaves again better since its advantages extend also to this situation.

Appendix A
Proof of results of Section 2.

Proof of Proposition 2.1. Given $M = (r, H, e(H, P), p(x))$, user i uses H and her private P_i to compute the value $e(H, P_i)$. Then she is able to compute $e(H, P)e(H, P_i)^{-1} = e(H, P - P_i) = \alpha_i$. Now Using her private x_i and the computed α_i she evaluates $p(x_i\alpha_i) = n$. Finally, using r and n she is able to recover the secret $K = r - e(H, P)^n$.

Proof of Theorem 2.2. From Proposition 2.1 it is clear that knowing n provides K. On the other hand if K is revealed somehow to any adversary, then solving the discrete logarithm we get that $n = \log_{e(H,P)}(r - K)$. Thus we can observe that a brute force attack is infeasible without getting any information that allows to derive n from $p(x)$, the only part of the re-keying message containing information on it.

Now assume that K is encrypted using the point $P = P' + P_{m+1}$ as in the above protocol and that it gives the re-keying message $M_1 = (r, H, e(H, P), p_1(x))$. Let us consider now a second re-keying message $M_2 = (r, H, e(H, P), p_2(x))$, with $p_2(x)$ of the same degree than $p_1(x)$, but such that it does not interpolates on the users' tickets to give n. Assume that the triple (M_1, M_2, K) is given to an attacker $A = (A_1, A_2)$ that does not know any valid ticket (x_i, P_i) intervening in the encryption. Then A_1 solves the discrete logarithm $n = \log_{e(H,P)}(r - K)$. Now A_2's job is to determine whether M_1 or M_2 is the corresponding re-keying message for K. However, to do so, she would need to know every ticket (x_i, P_i), or more precisely, the products $x_i \cdot \alpha_i$, $i = 1, \ldots, m + 1$ to determine whether $p_1(x)$ or $p_2(x)$ was used to distribute n.

Proof of Theorem 2.3. Let $M = (r, H, e(H, P), p(x))$ and $M' = (r', H', e(H', P'), p'(x))$ be the two different re-keying messages. Then A_2 provides the corresponding factorizations for $p(x) - K$ and $p(x) - K'$ and the attacker determines that $x_j\alpha_j$ and $x'_j\alpha'_j$ belongs to some user j.

Now A_3 gets

$$\frac{x_j\alpha_j}{x_j\alpha'_j} = \frac{\alpha_j}{\alpha'_j} = \frac{e(H, P)e(H', P_j)}{e(H', P')e(H, P_j)}$$

We point out that P and P' will be different even if the audience does not change. Thus the attacker gets $e(H' - H, P_j)$ since $e(H, P)$ and $e(H', P')$ are public.

Now A_3 selects Q expecting that the point P_j belonging to user j is in the same cyclic subgroup as Q. Thus $P_j = aQ$ for some a and therefore $e(H' - H, P_j) = e(H' - H, Q)^a$ and A_1 outputs a. If no solution is given (probably since Q is not in the same cyclic subgroup), then a new Q is chosen.

Thus A_3 gets α_j from $e(H, P_j)$ and since $x_j \cdot \alpha_j$, the ticket (x_j, P_j) would be compromised.

Acknowledgments

First contributing author is supported by grants from the Spanish Ministry of Science and Innovation (TIN2008-01117), and Junta de Andalucía (P08-TIC-3518), in part financed by the European Regional Development Fund (ERDF).Second contributing author is supported by grant Junta de Andalucía (FQM 0211).

References

[1] N. Antequera and J. A. Lopez-Ramos, *Pairings and secure multicast*, Proceedings of the 11th International Conference on Computational and Mathematical Methods in Science and Engineering CMMSE 2011, Alicante, 2011, 114–119.

[2] J. M. Arrufat, J.A. Alvarez-Bermejo and J. A. Lopez-Ramos, *Una implementación paralela del CRA con aplicaciones criptográficas,* VIII Jornadas de Matemática Discreta y Algorítmica, Proceedings, Almería 2012, July 11th-13th, 2012, 267–274.

[3] J.-L. Beuchat, E. Lopez-Trejo, L. Martinez-Ramos, S. Mitsunari, and Francisco Rodriguez-Henriquez, *Multi-core Implementation of the Tate Pairing over Supersingular Elliptic Curves,* Cryptology and Network Security (CANS 2009), LNCS 5888, 2009, 413–432.

[4] I. F. Blake, G. Seroussi and N. P. Smart, *Advances in Elliptic Curve Cryptography*, London Mathematical Society LNS Series 317, Cambridge University Press, Cambridge, 2005.

[5] D. Boneh and R. J. Lipton, *Quantum cryptanalysis of hidden linear functions*, Advances in Cryptology-CRYPTO '95, Lecture Notes in Computer Sciences 963, Springer-Verlag, Berlin, 1995, 424–437.

[6] D. Boneh and A. Silverberg, *Applications of Multilinear Forms to Cryptography,* Contemp. Mathematics 324, 2003, 71–90.

[7] K. -C. Chan and S. -H. G. Chan, *Key management approaches to offer data confidentiality for secure multicast*, Network, IEEE 17(5), 2003, 30–39.

[8] G. -H. Chiou and W. -T. Chen, *Secure broadcasting using the secure lock*, IEEE Trans. Softw. Eng. 15(8), 1989, 929–934.

[9] R. Di Pietro and L. V. Mancini, *Efficient and Secure Keys Management for Wireless Mobile Communications*, Proceedings of the second ACM international workshop on Principles of mobile computing, 2002, 66–73.

[10] D. Dolev, C. Dwork, and M. Naor, *Non-malleable cryptography*, Proceedings of the 23rd Annual Symposium on Theory of Computing, ACM, 1991.

[11] D. Dolev, C. Dwork, and M. Naor, *Non-malleable cryptography*, SIAM J. Comput. 30(2), 2000, 391–437.

[12] S. Goldwasser and S. Micali, *Probabilistic encryption*, J. Comput. System Sci. 28, 1984, 270–299.

[13] A. Joux, *A One Round Protocol for Tripartite Diffe-Hellman,* J. Cryptology 17, 2004, 263–276.

[14] D. Kammler, D. Zhang, P. Schwabe, H. Scharwaechter, M. Langenberg, D. Auras, G. Ascheid, R. Leupers, R. Mathar, and H. Meyr, *Designing an ASIP for cryptographic pairings over Barreto-Naehrig curves,* Cryptology ePrint Archive, Report 2009/056, 2009.

[15] P. S. Kruus and J. P. Macker, *Techniques and issues in multicast security*, Proceedings of Military Communications Conference, MILCOM 1998, 1998, 1028–1032.

[16] B. Liu, W. Zhang and T. Jiang, *A Scalable Key Distribution Scheme for Conditional Access System in Digital Pay-TV System*, IEEE Trans. Consum. Electron. 50(2), 2004, 632–637.

[17] A. Menezes, T. Okamoto, and S. Vanstone, *Reducing elliptic curve logarithms to logarithms in a finite field*, IEEE Transactions on Information Theory, 39, 1993, 1639–1646.

[18] J. A. M. Naranjo, N. Antequera, L. G. Casado and J. A. Lopez-Ramos, *A suite of algorithms for key distribution and authentication in centralized secure multicast environments*, J. Comp. Appl. Math. 236 (12), 2012, 3042–3051

[19] A. Perrig, D. Song, and J. D. Tygar, *Elk, a new protocol for efficient large-group key distribution*, Proceedings of IEEE Symposium on Security and Privacy (S& P), 2001, 247–262.

[20] O. Scheikl, J. Lane, R. Boyer and M. Eltoweissy, *Multi-level secure multicast: the rethinking of secure locks*, Parallel Processing Workshops, 2002. Proceedings. International Conference on, 2002, 17–24.

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

[22] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput. 26(5), 1997, 1484–1509.

[23] C. Shu, S. Kwon, and K. Gaj, *Reconfigurable computing ap-*

proach for Tate pairing cryptosystems over binary fields, IEEE Transactions on Computers 58(9), 2009, 1221–1237.

[24] P. Vijayakumar, S. Bose, A. Kannan, *Key Distribution for Pay-TV System with Reduced Computation Cost Using XOR Operation*, ADCONS 2011, 478–485.

[25] P. Vijayakumar, S. Bose, A. Kannan, L. J. Deborah, *Computation and Communication Efficient Key Distribution Protocol for Secure Multicast Communication*, TIIS 7(4), 2013, 878–894.

[26] D. Wallner, E. Harder and R. Agee, *Key management for multicast: Issues and architectures*, RFC 2627, 1999.

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