Group key management based on semigroup actions *

J.A. López-Ramos[†], J. Rosenthal[‡], D. Schipani[‡], R. Schnyder[‡]

July 25, 2016

Abstract

In this work we provide a suite of protocols for group key management based on general semigroup actions. Construction of the key is made in a distributed and collaborative way. Examples are provided that may in some cases enhance the security level and communication overheads of previous existing protocols. Security against passive attacks is considered and depends on the hardness of the semigroup action problem in any particular scenario.

1 Introduction

Traditional cryptographic tools for key exchange may not be useful when the communication process is carried out in a group of nodes or users. There exist several approaches for group key management, which may be divided into three main classes [12]:

- *centralized* protocols, where a single entity is in charge of controlling the whole group, minimizing storage requirements, computational power on both the client and server side and communication overheads,
- *decentralized*, where a large group is divided into subgroups in order to avoid concentrating the workload in a single point,
- *distributed*, where key generation is carried out in a distributed and collaborative way.

[*] The Research was supported in part by the Swiss National Science Foundation under grant No. 149716. First author is partially supported by Ministerio de Educacion, Cultura y Deporte grant "Salvador de Madariaga" PRX14/00121, Ministerio de Economia y Competitividad grant MTM2014-54439 and Junta de Andalucia (FQM0211). The last author is supported by Armasuisse.

[†] University of Almeria

[‡] University of Zurich

This last class of approaches has become particularly important since the emergence of ad hoc networks, where a set of nodes, possibly consisting of light and mobile devices, create, operate and manage a network, which is therefore solely dependent on the cooperative and trusting nature of the nodes. Moreover the limited capacity of the involved devices imposes both key storage and computational requirements. Such a network is commonly created to meet an immediate demand and specific goal, and nodes are continuously joining or leaving it. Thus, group key management based on distributed and collaborative schemes has proved to be of great interest (cf. for instance [17] and its references).

One of the most cited approaches in the distributed setting is due to Steiner et al. in [14] and [15]. In these works the authors provide two different group key management schemes that extend the traditional Diffie-Hellman key exchange [4] and feature very efficient rekeying procedures.

In [8], the authors generalize the aforementioned classical Diffie-Hellman key exchange to arbitrary group actions:

Protocol 1 (Semigroup Diffie-Hellman Key Exchange)**.** Let S be a finite set, G an abelian semigroup, and $\Phi : G \times S \to S$ a G-action on S. The semigroup Diffie-Hellman key exchange in (G, S, Φ) is the following protocol:

1. Alice and Bob publicly agree on an element $s \in S$.

2. Alice chooses $a \in G$ and computes $\Phi(a, s)$. Alice's private key is a, her public key is $\Phi(a, s)$.

3. Bob chooses $b \in G$ and computes $\Phi(b, s)$. Bob's private key is b, his public key is $\Phi(b, s)$.

4. Their common secret key is then

$$\Phi(a, \Phi(b, s)) = \Phi(ab, s) = \Phi(ba, s) = \Phi(b, \Phi(a, s)).$$

In the original Diffie-Hellman proposal, if an adversary is able to solve the so-called Discrete Logarithm Problem (DLP), then she is able to break the Diffie-Hellman key exchange. In this setting we can analogously consider the following more general problem:

Problem 1 (Semigroup Action Problem, SAP)**.** Given a semigroup G acting on a set S and elements $x, y \in S$, find $g \in G$ such that $\Phi(g, x) = y$.

It is clear that if an adversary, Eve, finds a $g \in G$ such that $\Phi(g, s) = \Phi(a, s)$, then she can find the shared secret by computing $\Phi(g, \Phi(b, s)) = \Phi(gb, s) = \Phi(bg, s) = \Phi(b, \Phi(a, s))$.

We can say that the security of the preceding protocol is equivalent to the following problem.

Problem 2 (Diffie-Hellman Semigroup Action Problem, DHSAP)**.** Given a finite abelian semigroup G acting on a finite set S and elements $x, y, z \in S$ with $y = \Phi(g, x)$ and $z = \Phi(h, x)$ for some $g, h \in G$, find $\Phi(gh, x)$.

Although, as noted above, solving the SAP implies solving the DHSAP, we do not know if both problems are (in general) equivalent, just like in the traditional setting of Diffie-Hellman, where however some equivalence results for particular scenarios are known [6].

Motivated by the above, our idea is now to define extensions of the semigroup Diffie-Hellman key exchange protocol to n users, by first generalizing those introduced in [14] and [15], and then considering other settings, which can feature more favorable characteristics compared to the original protocol. Since the capability of devices is often limited and authentication processes may be difficult to implement in a distributed network, we focus our attention on confidentiality under passive attacks. As in [8], some non-standard settings are introduced as more general examples, although the hardness of the SAP there may not be proven yet, so the security of the protocols in those cases is conditional on that.

The structure of the paper is as follows. In Section 2 we consider a suite of three protocols for group key management based on one-sided actions. While these naturally extend the results of [14] and [15], we consider different settings for a general semigroup action. Section 3 considers the security of the preceding protocols against passive attacks, including forward and backward secrecy. Finally, in Section 4, we introduce two protocols based on linear actions, i.e. semigroup actions on other groups satisfying a certain distributivity property. We give two different group key protocols in this setting, one of which runs very efficiently in only two rounds, independently of the number of members in the communicating group.

Throughout this paper we will consider a group of n users, $\mathcal{U}_1, \ldots, \mathcal{U}_n$, who would like to share a secret element of a finite set S, and G will denote a finite abelian semigroup acting on S.

2 Group key communication based on one-sided actions

In this section we consider three different extensions of the semigroup Diffie-Hellman key exchange with different computing requirements and communication overheads, but with possible applications in different cases. They are natural extensions of [14] and [15]. For completeness we report proofs in appendix to show soundness of the schemes.

2.1 A sequential key agreement

The first approach to extend the key exchange protocol consists of a sequence of messages, built using pieces of private information, along a chain of users and an analogous second sequence of messages in the opposite way. Therefore every user will send and receive two messages except for the user that initiates the communication and the last user receiving the sequence of messages.

The protocol is defined by the following steps.

Protocol 2 (GSAP-1)**.** Users agree on an element s in a finite set S, a finite abelian semigroup G, and a G-action on S given by Φ. For every $i = 1, \dots, n$, the user $\mathcal{U}_i$ holds a private element $g_i \in G$.

1. For $i = 1, \dots, n-1$, user $\mathcal{U}_i$ sends to user $\mathcal{U}_{i+1}$ the message

$$\{C_1, \dots, C_i\} = \left\{ \ \Phi(g_1, s), \ \Phi(g_2 g_1, s), \dots, \Phi\Big(\prod_{j=1}^{i} g_j, s\Big) \ \right\}.$$

2. User $\mathcal{U}_n$ computes $\quad \Phi(g_n, C_{n-1})$.
3. For $k = n, \dots, 2$, user $\mathcal{U}_k$ sends to user $\mathcal{U}_{k-1}$ the message $\{f_1^k, \dots, f_{k-1}^k\}$, where $f_j^k = \Phi(g_k, f_j^{k+1})$ for $2 \leq k \leq n-1$ and $f_j^n = \Phi(g_n, C_{j-1})$, $j = 1, \dots, n-1$, with $C_0 = s$.
4. User $\mathcal{U}_k$ computes $\Phi(g_k, f_k^{k+1})$.

2.2 A key agreement in broadcast

The following protocol presents a lower communication overhead than GSAP-1. The idea is again to get a first sequence of messages from user $\mathcal{U}_1$ to user $\mathcal{U}_n$, but now $\mathcal{U}_n$ will broadcast a message that allows the rest of the users to recover the common key.

Protocol 3 (GSAP-2)**.** Users agree on an element s in a finite set S, a finite abelian semigroup G, and a G-action Φ on S. For every $i = 1, \dots, n$, the user $\mathcal{U}_i$ holds a private element $g_i \in G$.

1. For $i = 1, \dots, n-1$, user $\mathcal{U}_i$ sends to user $\mathcal{U}_{i+1}$ the message

$$\{C_{i-1}^{i-1}, C_1^i, \dots, C_i^i\},$$

where $C_0^0 = s$, $C_1^1 = \Phi(g_1, s)$, and for $i \geq 2$, $C_1^i = \Phi(g_i, C_{i-2}^{i-2})$, $C_j^i = \Phi(g_i, C_{j-1}^{i-1})$ (with $j = 2, \dots, i$).

2. User $\mathcal{U}_n$ computes $\quad \Phi(g_n, C_{n-1}^{n-1})$.
3. User $\mathcal{U}_n$ broadcasts $\{f_1^n, \dots, f_{n-1}^n, f_n^n\}$, where $f_i^n = \Phi(g_n, C_{n-1-i}^{n-1})$ for $i = 1, \dots, n-2$, $f_{n-1}^n = \Phi(g_n, C_{n-2}^{n-2})$ and $f_n^n = C_{n-1}^{n-1}$.
4. User $\mathcal{U}_i$ computes $\quad \Phi(g_i, f_i^n)$.

Remark 2.1. It can be observed that the element f_n^n contained in the broadcast message in step 3 of Protocol GSAP-2, is not needed by any of the users $\mathcal{U}_i$, $i = 1, \dots, n-1$ to recover the shared key. However, the distribution of this value is required for future rekeying operations, as we will later show.

2.3 Examples

a) The two previous protocols are extensions of those introduced in [14] and [15] for the action of the multiplicative semigroup $\mathbb{N}^*$ on a cyclic group S of order q generated by g, given by $\Phi(y, g^x) = (g^x)^y$. It was pointed out that the first protocol presents excessive communication overheads mainly due to both the number of rounds and messages to be sent. Because of this, only the second one, referred to as IKA.1 in [15], was recommended. However, the first protocol could be interesting on its own when applied to a sensor network whose communications need to be secure and where it should be assessed whether every node is working properly. After user $\mathcal{U}_n$ receives the message in step 1, the absence of any of the messages (excepting the last one) in the descending chain of rounds would alert that the corresponding sender node is not working or the communication was interrupted.

b) In particular, consider a finite field $GF(q)$ and an element g of prime order. The semigroup $\mathbb{N}^*$ acts on the subgroup $\langle g \rangle \subset GF(q)^*$ by $\Phi(y, g^x) = (g^x)^y$ for $x, y \in \mathbb{N}^*$.

c) Let ε be the set of points in an elliptic curve. Then the action $\Phi : \mathbb{N}^* \times \varepsilon \to \varepsilon$ given by $\Phi(n, P) = nP$ for every $n \in \mathbb{N}^*$ and every $P \in \varepsilon$ provides the corresponding versions of the preceding protocols for elliptic curves. In [11] an implementation of the second protocol can be found.

d) In [8, Example 5.13] the authors illustrate the use of a semiring of order 6 to construct an example of a practical SAP. This was later cryptanalyzed in [16] due not to a general attack, but rather to the structure of this ring. However, we can use the semiring of order 20 given in [8, Example 5.8] to analogously define another SAP and its cryptanalysis is still an open question. This shows an example where SAP does not coincide with a traditional DLP on a semigroup and it is applicable to both preceding protocols.

e) In [10, Protocol 80] the author defines a key exchange protocol whose security is based on the SAP derived from the following semigroup action: let S be a semiring, T a finitely generated additive subsemigroup of S and let $\mathrm{End}_+(\mathrm{T})$ be its (additive) endomorphisms semigroup. Then the semigroup action that defines the security of this protocol is given by $\Phi : (S, T^{op}) \times \mathrm{End}_+(\mathrm{T}) \to \mathrm{End}_+(\mathrm{T})$, $((s,t), f) \mapsto (x \mapsto s * f(x) * t)$.

Remark 2.2. Many examples of semigroup actions suitable to defining a Diffie-Hellman type key exchange protocol can be found in [7]. The corresponding SAP is shown to be computationally equivalent to a DLP for some of them.

2.4 A key agreement given by a group action

The existence of inverses in the semigroup G acting on the set S can provide a way to agree on a common key with reduced communication overheads. Moreover, computations can be made more equally distributed among the users. We remark that in the protocols given in the two previous sections, these requirements are higher the further away the user is from the one that initialized the protocol.

Thus we assume that G is a group. The protocol is given by the following steps.

Protocol 4 (GSAP-3)**.** Users agree on an element $C_0 = s$ in a finite set S, a finite abelian group G, and a G-action Φ on S. For every $i = 1, \dots, n$, the user $\mathcal{U}_i$ holds a private element $g_i \in G$.

1. For $i = 1, \dots, n-2$, user $\mathcal{U}_i$ sends to user $\mathcal{U}_{i+1}$ the message $C_i = \Phi(g_i, C_{i-1})$.
2. User $\mathcal{U}_{n-1}$ computes $C_{n-1} = \Phi(g_{n-1}, C_{n-2})$ and broadcasts it to the other users $\{\mathcal{U}_1, \dots, \mathcal{U}_{n-2}, \mathcal{U}_n\}$.
3. User $\mathcal{U}_n$ computes the element $\Phi(g_n, C_{n-1})$.
4. For $i = 1, \dots, n-1$, user $\mathcal{U}_i$ computes $D_i = \Phi(g_i^{-1}, C_{n-1})$ and sends it to user $\mathcal{U}_n$.
5. For $i = 1, \dots, n-1$, user $\mathcal{U}_n$ computes $\Phi(g_n, D_i)$ and sends to users $\{\mathcal{U}_1, \dots, \mathcal{U}_{n-2}, \mathcal{U}_{n-1}\}$ the set of values $\{\Phi(g_n, D_1), \dots, \Phi(g_n, D_{n-1}), C_{n-1}\}$.
6. For $i = 1, \dots, n-1$, user $\mathcal{U}_i$ computes $\Phi(g_i, \Phi(g_n, D_i))$.

After protocol GSAP-3, the users $\mathcal{U}_1, \dots, \mathcal{U}_n$ share a common key given by $\Phi\Big(\prod_{i=1}^{n} g_i, s\Big)$. This follows easily from the commutativity of G and the fact that for every $g_i, g_j \in G$, $i, j = 1, \dots, n$ and $s \in S$, we get that $\Phi(g_i g_j, s) = \Phi(g_i, \Phi(g_j, s))$.

Remark 2.3. As in Protocol GSAP-2, we also point out that the element C_{n-1}, which is broadcast by $\mathcal{U}_n$ in step 5 of Protocol GSAP-3, is needed only for future rekeying purposes.

Remark 2.4. Using the action $\Phi(y, g^x) = (g^x)^y$ for $x, y \in \mathbb{Z}_q^*$, with g a generator of a cyclic group S of order q, we get the third protocol introduced in [14] and [15] and referred to as IKA.2 in CLIQUES [15]. In this case, user $\mathcal{U}_i$ sends to user $\mathcal{U}_n$ the message $g^{\prod_{j=1, j\neq i}^{n-1} x_j}$, which is computed with the element $x_i^{-1} \bmod q$, given that the x_i's are selected either to be coprime with q or, as the authors suggest, q is chosen to be a prime.

An elliptic curve version is clearly also feasible. An implementation in this sense can be found in [11].

3 Security of the key agreements and rekeying operations

In [8] it was pointed out that if an adversary is able to solve the SAP, then she will be able to break the two party Diffie-Hellman key exchange, i.e. solve the DHSAP. It is easy to observe that being able to solve the DHSAP allows getting the shared key in all the protocols proposed above.

Proposition 3.1. *If an adversary is able to solve the DHSAP, then she can get the shared key in GSAP-1, GSAP-2 and GSAP-3.*

Proof. This follows from the fact that the adversary can access the pair of values

- $(C_1, f_1^2) = \big(\Phi(g_1, s), \Phi\big(\prod_{i=2}^n g_i, s\big)\big)$ in GSAP-1;
- $(C_1^1, f_1^n) = \big(\Phi(g_1, s), \Phi\big(\prod_{i=2}^n g_i, s\big)\big)$ in GSAP-2;
- $\big(C_1, \Phi(g_n, D_1)\big) = \big(\Phi(g_1, s), \Phi\big(\prod_{i=2}^n g_j, s\big)\big)$ in GSAP-3.

□

The preceding result shows, as could be expected, that the multiparty key exchange protocols do not enhance the security that the corresponding two-party protocol offers. However, as in [14] and [15], it is possible to show that increasing the number of messages does not produce any information leakage whenever the corresponding key exchange based on the SAP for two communicating parties is secure. Here we are referring to security against passive attacks; a totally different picture would arise if we assume that the attacker can control communications from and to one or more particular users, see e.g. [13].

Let $X = \{g_1, \dots, g_n\}$ be a set of elements of the semigroup G, s an element of a set S and Φ a G-action on S. Let us define the (ordered) set of elements of S

$$V_\Phi^G(s, n, X) = \Big\{\Phi\Big(\prod_{j=i_1}^{i_m} g_j, s\Big) : \{i_1, \dots, i_m\} \subsetneq \{1, \dots, n\}\Big\}$$

and the value $K_\Phi^G(s, n, X) = \Phi\Big(\prod_{j=1}^n g_j, s\Big) \in S$.

We point out that the messages that any adversary observes in any of the protocols is a subset of $V_\Phi^G(s, n, X)$, and the key that the users agree on is precisely $K_\Phi^G(s, n, X)$. Let us assume now that Φ is a transitive action, i.e., for every pair of elements $s, s' \in S$ there always exists a $g \in G$ such that $\Phi(g, s) = s'$. Thus if $s \in S$ is a public element, given any two elements in S, s_1, s_2, there always exist $g_1, g_2 \in G$ such that $\phi(g_i, s) = s_i$, $i = 1, 2$. Let $s_3 = \Phi(g_1, \Phi(g_2, s)) = \Phi(g_1 g_2, s)$. If, given s, s_1 and s_2, it is not feasible to distinguish s_3 from a random value in polynomial time, then an induction argument like that given in [15, Theorem 1] allows us to show the following result.

Theorem 3.2. *Let Φ be a transitive G-action on S. Then the group key that users derive as a result of any of the protocols GSAP-1, GSAP-2 and GSAP-3 is indistinguishable in polynomial time from a random value, given only the values exchanged between users during the protocol, whenever the corresponding Diffie-Hellman protocol induced by Φ for two users satisfies this property.*

Another important issue in any group key management is rekeying after the initial key agreement. There exist three different situations that require a rekeying operation. The first is simply due to key caducity and the group of users remains the same. In the other two cases, we may find a new user that wishes to join the group or a user who leaves the group. In both situations it is required that the new (resp. former) user cannot access the former (resp. new) distributed key. In the following lines we describe the procedures as well as their security.

Let us start by considering the protocol GSAP-1 described in Section 2.1. In this case, we could simply require that a new initial key agreement is needed. However, we may shorten the rekeying process, keeping somehow the spirit of the protocol. If rekeying is due to key caducity, then user $\mathcal{U}_n$ chooses a new private element $g_n' \in G$ and defines a new sequence $f_j^n = \Phi(g_n' g_n, C_{j-1})$, $j = 1, \ldots, n-1$, with $C_0 = s$, as is done in step 3 of GSAP-1. The rest of the users also proceed as in step 3 and recover (using their private keys as described in GSAP-1) the new key $\Phi\Big(g_n' \prod_{j=1}^n g_j, s\Big)$.

In case some user, say $\mathcal{U}_i$, leaves the group, then the corresponding value f_i^n is omitted in the new message made by $\mathcal{U}_n$.

Finally, in case a user $\mathcal{U}_{n+1}$ joins the group, then user $\mathcal{U}_n$ chooses a new element g_n' and sends the message

$$\Big\{\ \Phi(g_n' g_1, s),\ \Phi(g_n' g_2 g_1, s), \ldots, \Phi\Big(g_n' \prod_{j=1}^{n} g_j, s\Big)\ \Big\}$$

to user $\mathcal{U}_{n+1}$. Then this user starts step 3 of GSAP-1.

Security of all new subsequent key distributions follows from Theorem 3.2.

In the case of protocols GSAP-2 and GSAP-3, described in Sections 2.2 and 2.4 respectively, we may use the information that every user holds after the initial key agreement to rekey very efficiently as is suggested in [15]. In this case, given that every user remembers the same information, say

$$\Big\{\Phi\Big(\prod_{r=2}^{n} g_r, s\Big), \Phi\Big(\prod_{r=1;r\neq 2}^{n} g_r, s\Big), \ldots, \Phi\Big(\prod_{r=1;r\neq c}^{n} g_r, s\Big), \ldots, \Phi\Big(\prod_{r=1}^{n-1} g_r, s\Big)\Big\},$$

the rekeying process may be carried out by any one of them. Let us call this user $\mathcal{U}_c$. If rekeying is due to key caducity, then he chooses a new $g_c' \in G$, changes his private key to $g_c' g_c$ and sends the following rekeying message:

$$\Big\{\Phi\Big(g'_c\prod_{r=2}^{n}g_r,s\Big),\Phi\Big(g'_c\prod_{r=1;r\neq 2}^{n}g_r,s\Big),\ldots,\Phi\Big(\prod_{r=1;r\neq c}^{n}g_r,s\Big),\ldots,\Phi\Big(g'_c\prod_{r=1}^{n-1}g_r,s\Big)\Big\}.$$

Then every user, using his private information, recovers the new common key given by $\Phi\Big(g'_c\prod_{r=1}^{n}g_r,s\Big)$.

In case some user leaves the group, the corresponding position in the rekeying message is omitted. If a new user joins the group, then $\mathcal{U}_c$ adds the element $\Phi\Big(g'_c\prod_{r=1}^{n}g_r,s\Big)$ and sends the following to the new user $\mathcal{U}_{n+1}$:

$$\Big\{\Phi\Big(g'_c\prod_{r=2}^{n}g_r,s\Big),\ldots,\Phi\Big(\prod_{r=1;r\neq c}^{n}g_r,s\Big),\ldots,\Phi\Big(g'_c\prod_{r=1}^{n-1}g_r,s\Big),\Phi\Big(g'_c\prod_{r=1}^{n}g_r,s\Big)\Big\}.$$

This user proceeds (in both protocols GSAP-2 and GSAP-3) to step 5 of protocol GSAP-3.

Again, security in every case is a consequence of Theorem 3.2.

4 Secure group communication based on linear actions

As can be observed in the protocols given in the previous section, user $\mathcal{U}_n$ plays a central role, and in two of them, every user is required to do a different number of computations and store a different number of values, depending on his proximity to $\mathcal{U}_n$. The aim of this section is twofold. On one hand, we give a similar approach to that of GSAP-3 in order to get a protocol with the same advantages that is applicable in situations where the semigroup G acting on S does not contain inverses. On the other hand, we give a new approach based on linear actions that in some cases not only significantly decreases communication overheads, but also reduces the number of rounds to just 2, which will significantly enhance the efficiency.

We say that, given G and S semigroups, an action $\Phi : G \times S \to S$ defined by $\Phi(g,s) = g \cdot s$ is linear in case $\Phi(g,ss') = \Phi(g,s)\Phi(g,s')$.

The following protocol is a modification of GSAP-3 for a linear G-action Φ on S, but instead of requiring G to be a group, we require this of S. We get a similar protocol that is also an extension of Diffie-Hellman to the multiparty case.

Protocol 5 (GSAP-3’)**.** Users agree on an element s in a finite group S, a finite abelian semigroup G, and a linear G-action Φ on S. For every $i = 1, \ldots, n$, the user $\mathcal{U}_i$ holds a private element $g_i \in G$.

1. For $i = 1, \dots, n-2$, user $\mathcal{U}_i$ sends to user $\mathcal{U}_{i+1}$ the message $C_i = \Phi(g_i, C_{i-1})$.

2. User $\mathcal{U}_{n-1}$ computes $C_{n-1} = \Phi(g_{n-1}, C_{n-2})$ and broadcasts it to the other users $\{\mathcal{U}_1, \dots, \mathcal{U}_{n-2}, \mathcal{U}_n\}$.

3. User $\mathcal{U}_n$ computes the element $\Phi(g_n, C_{n-1})$.

4. For $i = 1, \dots, n-1$, user $\mathcal{U}_i$ computes $D_i = \Phi(g_i, s)^{-1} C_{n-1}$ and sends it to user $\mathcal{U}_n$.

5. For $i = 1, \dots, n-1$, user $\mathcal{U}_n$ computes $\Phi(g_n, D_i)$ and sends to users $\{\mathcal{U}_1, \dots, \mathcal{U}_{n-2}, \mathcal{U}_{n-1}\}$ the set of values $\{\Phi(g_n, D_1), \dots, \Phi(g_n, D_{n-1}), \Phi(g_n, D_n)\}$ and his public key $\Phi(g_n, s)$, where $D_n = \Phi(g_n, s)^{-1} C_{n-1}$.

6. For $i = 1, \dots, n-1$, user $\mathcal{U}_i$ computes $\Phi(g_i, \Phi(g_n, s))\Phi(g_n, D_i)$.

Theorem 4.1. *After protocol GSAP-3', the users $\mathcal{U}_1, \dots, \mathcal{U}_n$ share a common key given by $\Phi\Big(\prod_{i=1}^{n} g_i, s\Big)$.*

Proof. This follows from the linearity of the action Φ. $\Phi(g_i, \Phi(g_n, s))\Phi(g_n, D_i) = \Phi(g_i g_n, s)\Phi\big(g_n, \Phi(g_i, s)^{-1}\Phi\big(\prod_{r=1}^{n-1} g_r, s\big)\big) = \Phi\big(\prod_{r=1}^{n} g_r, s\big)$, since $\Phi(g_i, e) = e$, e being the neutral element in S, and $\Phi(g_i, s)^{-1} = \Phi(g_i, s^{-1})$, again by the linearity of the action. □

Example 1. a) Given again a cyclic group S of order q generated by g, the action $\Phi : \mathbb{N}^* \times S \to S$ defined by $\Phi(y, g^x) = (g^x)^y$ is clearly linear, so the above argument applies. D_i assumes the form $g^{\prod_{j=1}^{n-1} x_j} g^{-x_i}$.

b) If ε is the group of points of an elliptic curve, then ε is a $\mathbb{Z}$-module via the linear action $\Phi(k, P) = kP$ for every $k \in \mathbb{Z}$ and $P \in \varepsilon$. D_i assumes the form $(\prod_{i=1}^{n-1} k_j)P - k_i P$.

c) Let us introduce an example where the preceding protocols can be run over a module structure. Let us recall from [2] the following ring:

$$E_p^{(m)} = \left\{[a_{ij}] \in \mathrm{Mat}_{m\times m}(\mathbb{Z}) \mid a_{ij} \in \mathbb{Z}_{p^i} \text{ if } i \leq j, \text{ and } a_{ij} \in p^{i-j}\mathbb{Z}_{p^i} \text{ if } i > j\right\},$$

with addition and multiplication defined, respectively, as follows

$$\begin{aligned}\big[a_{ij}\big] + \big[b_{ij}\big] &= \big[(a_{ij} + b_{ij}) \bmod p^i\big],\\ \big[a_{ij}\big] \cdot \big[b_{ij}\big] &= \left[\left(\sum_{k=1}^{m} a_{ik} b_{kj}\right) \bmod p^i\right].\end{aligned}$$

Here $\mathrm{Mat}_{m\times m}(\mathbb{Z})$ denotes the set of $m \times m$ matrices with entries in $\mathbb{Z}$, and $p^r\mathbb{Z}_{p^s}$ denotes the set $\{p^r u \mid u \in \{0, \dots, p^s - 1\}\} \subset \mathbb{Z}$ for positive integers r and s. This ring is clearly non-commutative and its product defines an action of the multiplicative semigroup $E_p^{(m)}$ on the set $\mathbb{Z}_p \times \mathbb{Z}_{p^2} \times \cdots \times \mathbb{Z}_{p^m}$. However, to ensure that the key exchange works, we need that the elements in the semigroup commute. In this non-commutative setting, this may be achieved by considering that the selected elements in the semigroup $E_p^{(m)}$ are of the form $\sum_{i=0}^{r} C_i M^i$, such that for every $i = 0, \ldots, r$, C_i is in the center Z of $E_p^{(m)}$ and $M \in E_p^{(m)}$ is a public element such that its set of powers is large enough. In other words, if we denote the set of elements of this form by $Z[M]$, then we are using for G the multiplicative subsemigroup $Z[M]$ of $E_p^{(m)}$.

From [3, Theorem 2] we can deduce conditions on the public information that will be sent in order to prevent an attacker from solving the SAP in the subsemigroup of $Z[M]$ given by the center Z of the ring, with cardinality p^m (cf. [2]). Thus if M has high order, i.e. M is such that the least integer n satisfying $M^{k+n} = M^k$ for every sufficiently large k is high, we will obtain that $Z[M]$ is big enough.

Note that our aim in this paper is not to prove the hardness of the SAP for this particular example, but rather to present protocols which rely on the hardness of the SAP in a particular scenario once it has been established there. The non-commutative scenario in particular may present hidden vulnerabilities, as was shown in recent cryptanalyses, e.g. [5, 9], although these seem not to directly apply in this setting. For example [5] introduces a cryptanalysis for the case of two users when the ring $E_p^{(m)}$ acts on itself, which can be countered by choosing p and m appropriately in order to avoid the existence of inverses [2]. In the case of [9], the cryptanalysis requires building a system of equations, which does not seem to be straightforward in this new setting of $Z[M]$. In [7, Proposition 3.9] it is asserted that if the commutative semigroup has a big number of invertible elements, then it is possible to develop a square root attack to the SAP. Again we point out that $E_p^{(m)}$ could be chosen in order to avoid this attack.

Given that both $\Phi\Big(\prod_{i=1}^{n-1} g_i, s\Big)$ and $\Phi(g_n, s)$ are public we immediately get the following.

Proposition 4.2. *If an adversary is able to solve the DHSAP, then she can get the shared key in GSAP-3'.*

Let us recall from [8] that given any G-action Φ on S, we can easily define an ElGamal type of public key cryptosystem. We define the following ElGamal type of protocol.

1. Alice and Bob publicly agree on an element $s \in S$.

2. Bob chooses $b \in G$ and computes $\Phi(b, s)$. Bob's private key is b, his public key is $\Phi(b, s)$.

3. If Alice wants to send the message $m \in S$ to Bob, then she gets Bob's public key $\Phi(b, s)$.

4. Alice chooses randomly $a \in G$ and computes $\Phi(a, s)$ and $\Phi(a, \Phi(b, s))$.
5. Alice sends to Bob the pair $(c, d) = \big(\Phi(a, s), m\Phi(a, \Phi(b, s)\big)$.
6. Bob recovers $m = d\Phi(b, c)^{-1} = m\Phi(a, \Phi(b, s))\Phi(b, \Phi(a, s))^{-1}$, given that S has a group structure.

It can be easily observed that solving the DHSAP is equivalent to breaking the preceding algorithm: if given the public information

$$(s, \Phi(a, s), \Phi(b, s), m\Phi(ab, s))$$

one is able to get m, then the input $(s, \Phi(a, s), \Phi(b, s), e)$, for $e \in S$ the neutral element, produces $\Phi(ab, s)^{-1}$, which solves the DHSAP. Conversely, given Bob's public key $\Phi(b, s)$ and the pair $\big(\Phi(a, s), m\Phi(a, \Phi(b, s))\big)$, one can use $\Phi(ab, s)$ from the DHSAP to recover m.

Now using the above we are able to show the security of GSAP-3'.

Theorem 4.3. *The group key that users derive as a result of GSAP-3' is indistinguishable in polynomial time from a random value whenever the corresponding Diffie-Hellman protocol induced by Φ for two users also satisfies this property.*

Proof. Given that both $C_{n-1} = \Phi\Big(\prod_{i=1}^{n-1} g_i, s\Big)$ and $D_i = \Phi(g_i, s)^{-1}C_{n-1}$ are public, an adversary is able to get all the public values $\Phi(g_i, s)$, $i = 1, \ldots, n$. Now user $\mathcal{U}_n$ sends the message $\{\Phi(g_n, D_i)\}_{i=1}^{n-1}$ jointly with $\Phi(g_n, s)$, in other words, due to linearity of Φ, user $\mathcal{U}_n$ sends a "a family of pairs", $i = 1, \ldots, n$,

$$\Big(\Phi(g_n, s), \Phi(g_n, \Phi(g_i, s)^{-1})\Phi\Big(g_n, \Phi\Big(\prod_{j=1}^{n-1} g_j, s\Big)\Big)\Big),$$

which can be seen as a set of ElGamal encryptions of the message

$$\Phi\Big(\prod_{i=1}^{n} g_i, s\Big) = \Phi\Big(g_n, \Phi\Big(\prod_{i=1}^{n-1} g_i, s\Big)\Big)$$

using the public keys $\Phi(g_i, s)$, $i = 1, \ldots, n$. Alternatively, one can consider the pairs

$$\Big(\Phi(g_i, s), \Phi(g_n, \Phi(g_i, s)^{-1})\Phi\Big(g_n, \Phi\Big(\prod_{j=1}^{n-1} g_j, s\Big)\Big)\Big),$$

which can also be seen, given the commutativity in G, as a set of ElGamal encryptions of the message

$$\Phi\Big(\prod_{i=1}^{n} g_i, s\Big) = \Phi\Big(g_n, \Phi\Big(\prod_{i=1}^{n-1} g_i, s\Big)\Big)$$

using the public key $\Phi(g_n, s^{-1})$, and the g_i's as random numbers, for $i = 1, \dots, n$.

Thus, as we pointed out above, given the equivalence of the security of the ElGamal type of public key cryptosystem and the DHSAP, the result follows. $\square$

The rekeying process in this setting is analogous to that described in Section 3 for protocols GSAP-2 and GSAP-3.

We first note that every user remembers the following keying information.

$$\{\Phi(g_n, D_1), \dots, \Phi(g_n, D_{n-1}), \Phi(g_n, D_n)\}$$

In case of key caducity, user $\mathcal{U}_c$ for some $c = 1, \dots, n$ chooses a new element $g'_c \in G$, computes a new key given by $\Phi\Big(g'_c \prod_{i=1}^{n} g_i, s\Big)$ and his keying information $\Phi((g'_c)^2 g_c g_n, s)^{-1}\Phi\Big(g'_c \prod_{i=1}^{n} g_i, s\Big)$ and broadcasts the following message

$$\{\Phi(g'_c, \Phi(g_n, D_1)), \dots, \Phi((g'_c)^2 g_c g_n, s)^{-1}\Phi\Big(g'_c, \Phi\Big(\prod_{i=1}^{n} g_i, s\Big)\Big), \dots,$$
$$\Phi(g'_c, \Phi(g_n, D_{n-1})), \Phi(g'_c, \Phi(g_n, D_n))\},$$

jointly with the value $\Phi(g'_c, \Phi(g_n, s))$. User $\mathcal{U}_c$ changes his private information to $g_c g'_c$.

In case rekeying is due to some user leaving the group, then the corresponding value is omitted in the above message.

Finally, let us assume that $\mathcal{U}_{n+1}$ joins the group. The process corresponds in this case to something similar to a "double rekeying"as above. First, $\mathcal{U}_c$ sends to $\mathcal{U}_{n+1}$

$$\Big\{\Phi(g'_c, \Phi(g_n, D_1)), \dots, \Phi((g'_c)^2 g_c g_n, s)^{-1}\Phi\Big(g'_c, \Phi\Big(\prod_{i=1}^{n} g_i, s\Big)\Big), \dots,$$
$$\Phi(g'_c, \Phi(g_n, D_{n-1})), \Phi(g'_c, \Phi(g_n, D_n)), \Phi\Big(g'_c, \Phi\Big(\prod_{i=1}^{n} g_i, s\Big)\Big)\Big\}$$

jointly with the value $\Phi(g'_c, \Phi(g_n, s))$. Then, $\mathcal{U}_{n+1}$ broadcasts a rekeying message given by

$$\Big\{\Phi(g_{n+1}g'_c, \Phi(g_n, D_1)), \dots, \Phi(g_{n+1}(g'_c)^2 g_c g_n, s)^{-1}\Phi\Big(g'_c, \Phi\Big(\prod_{i=1}^{n+1} g_i, s\Big)\Big), \dots,$$
$$\Phi(g_{n+1}g'_c, \Phi(g_n, D_{n-1})), \Phi(g_{n+1}g'_c, \Phi(g_n, D_n)),$$
$$\Phi(g_{n+1}^2 g'_c g_n, s)^{-1}\Phi\Big(g'_c, \Phi\Big(\prod_{i=1}^{n+1} g_i, s\Big)\Big)\Big\}$$

jointly with the value $\Phi(g_{n+1}g'_c g_n, s)$.

Security of these processes is shown with a similar argument as in Theorem 4.3.

A more symmetrical use of linear actions is the following protocol, which decreases the number of rounds to just 2, but which is only applicable in some cases.

Protocol 6 (GSAP-4)**.** Users agree on an element s in a finite abelian semigroup S, a finite abelian semigroup G, and a linear G-action Φ on S. For every $i = 1, \ldots, n$, the user $\mathcal{U}_i$ holds a private element $g_i \in G$.

1. For every $i = 1, \ldots, n$, user $\mathcal{U}_i$ makes public $\Phi(g_i, s) = g_i \cdot s$.
2. For some $j = 1, \ldots, n$, user $\mathcal{U}_j$ computes and makes public
$$D_i = \Phi\Big(g_j, \prod_{r\neq j,i} \Phi(g_r, s)\Big), \; i \neq j, \; i = 1, \ldots, n.$$
3. For every $i = 1, \ldots, n$, $i \neq j$, user $\mathcal{U}_i$ computes $D_i\Phi(g_i, \Phi(g_j, s))$. User $\mathcal{U}_j$ computes $\Phi(g_j, \big(\prod_{r\neq j} \Phi(g_r, s)\big)$.

Theorem 4.4. *After protocol GSAP-4, the users* $\mathcal{U}_1, \ldots, \mathcal{U}_n$ *share a common key given by* $\Phi(g_j, \prod_{r\neq j} \Phi(g_r, s))$.

Proof. For every $i = 1, \ldots, n$, $i \neq j$,

$$\begin{aligned} D_i\Phi(g_i, \Phi(g_j, s)) &= \Phi\big(g_j, \textstyle\prod_{r\neq j,i} \Phi(g_r, s)\big)\Phi(g_i, \Phi(g_j, s)) \\ &= \Phi\big(g_j, \textstyle\prod_{r\neq j,i} \Phi(g_r, s)\big)\Phi(g_i g_j, s) \\ &= \Phi\big(g_j, \textstyle\prod_{r\neq j,i} \Phi(g_r, s)\big)\Phi(g_j g_i, s) \\ &= \Phi\big(g_j, \textstyle\prod_{r\neq j,i} \Phi(g_r, s)\big)\Phi(g_j, \Phi(g_i, s)) \\ &= \Phi\big(g_j, \textstyle\prod_{r\neq j} \Phi(g_r, s)\big). \end{aligned}$$

□

Example 2. a) Let us consider again a cyclic group S of order q generated by g, with the action $\Phi : \mathbb{N}^* \times S \to S$ given by $\Phi(y, g^x) = (g^x)^y$. Then GSAP-4 implies sharing a key of the form $K = g^{k_j \sum_{r=1,r\neq j}^{n} k_r}$. An adversary can access the messages

$$D_i = \Phi\Big(g_j, \prod_{r\neq j,i} \Phi(g_r, s)\Big), \; i \neq j, \; i = 1, \ldots, n,$$

from which she can compute $\prod_{r=1,r\neq j}^{n} D_r = K^{n-2}$. In the case where the order q of S is known, the adversary can now recover the key K from K^{n-2} by inverting $n-2$ modulo q. This is in particular the case where S is a subgroup of a finite field, or where it is the group of points of an elliptic curve. However, we can avoid this weakness by adding some authentication information as is done in [1].

b) Let $m = pq$ with p and q two large primes and let $G = \mathbb{Z}^*_{(p-1)(q-1)}$. Then the action $\Phi : G \times \mathbb{Z}_m \to \mathbb{Z}_m$ given by $\Phi(x, g) = g^x \bmod m$ shows an example where the above attack cannot be developed unless the adversary is able to factorize m. The shared key in this case is of the form $g^{x_j \sum_{i=1, i\neq j}^{n} x_i} \bmod m$.

c) We recall that a semiring R is a semigroup with respect to both addition and multiplication and the distributive laws hold. It is also understood that a semiring is commutative with respect to addition and the existence of neutral elements is not required, although some authors do require it. Then given a semiring R, a left R-semimodule M is an abelian semigroup with an action $\Phi : R \times M \to M$, $\Phi(r, m) = rm$, satisfying $r(sm) = (rs)m$, $(r+s)m = rm+sm$ and $r(m+n) = rm + rn$ for all $r, s \in R$ and $m, n \in M$. Thus, based on the previous two examples, we can assert in general that any semimodule S over a semiring R fits with GSAP-4 and the shared key is of the form $k_j(\sum_{r=1, r\neq j}^{n} k_r)s$ for $k_i \in R$, $i = 1, \ldots, n$ private and $s \in S$ public.

Remark 4.5. Due to the attack shown in example a), the hardness of the Diffie-Hellman problem is not enough to show security in this case. We leave it as an open question whether the hardness of factoring would be enough to do so.

Remark 4.6. We can also give protocols based on two-sided actions. To this end we recall that given a semiring S, right S-semimodules are defined dually to left ones. Then, given two semirings R and S, an (R, S)-bisemimodule M is both a left R-semimodule and a right S-semimodule such that $(rm)s = r(ms)$ for every $r \in R$, $m \in M$ and $s \in S$.

Now we are able to provide key exchange protocols similar to those given in the previous sections based on two-sided linear actions over a (R, S)-bisemimodule M. In the case of GSAP-3', since we need the existence of inverses with respect to addition in M, we may suppose that M has an (R, S)-bimodule structure for some rings R and S.

5 Appendix GSAP1

Theorem 5.1. *After protocol GSAP-1, users $\mathcal{U}_1, \ldots, \mathcal{U}_n$ agree on the common key $\Phi\Big(\prod_{j=1}^{n} g_j, s\Big)$.*

Proof. User $\mathcal{U}_n$ computes

$$\Phi(g_n, C_{n-1}) = \Phi\Big(g_n, \Phi\Big(\prod_{j=1}^{n-1} g_j, s\Big)\Big) = \Phi\Big(\prod_{j=1}^{n} g_j, s\Big).$$

Let us show now that the rest of the users recover exactly the same key. For $k = 1, \ldots, n-1$, user $\mathcal{U}_k$ computes $\Phi(g_k, f_k^{k+1})$.

It is straightforward to show that for every $i = 1, \dots, n-2$, $j = 1, \dots, n-i-1$, the following equality holds:

$$f_j^{n-i} = \Phi\Big(\Big(\prod_{r=n-i}^{n} g_r\Big)\Big(\prod_{r=1}^{j-1} g_r\Big), s\Big),$$

with the empty product being equal to 1.

We then have:

$$\begin{aligned} f_k^{k+1} &= f_k^{n-(n-k-1)} \\ &= \Phi\Big(\Big(\prod_{r=k+1}^{n} g_r\Big)\Big(\prod_{r=1}^{k-1} g_r\Big), s\Big) \\ &= \Phi\Big(\prod_{r=1;r\neq k}^{n} g_r, s\Big). \end{aligned}$$

Thus, user $\mathcal{U}_k$ computes

$$\Phi(g_k, f_k^{k+1}) = \Phi\Big(g_k, \Phi\Big(\prod_{r=1;r\neq k}^{n} g_r, s\Big)\Big) = \Phi\Big(\prod_{r=1}^{n} g_r s\Big),$$

as we wanted to show. □

6 Appendix GSAP2

Theorem 6.1. *After protocol GSAP-2, users $\mathcal{U}_1, \dots, \mathcal{U}_n$ agree on a common key given by $\Phi\Big(\prod_{r=1}^{n} g_r, s\Big)$.*

Proof. User $\mathcal{U}_n$ computes $\Phi(g_n, C_{n-1}^{n-1}) = \Phi\Big(g_n, \Phi\Big(\prod_{r=1}^{n-1} g_r, s\Big)\Big) = \Phi\Big(\prod_{r=1}^{n} g_r, s\Big)$.

Now, let us show that $f_i^n = \Phi\Big(\prod_{r=1;r\neq i}^{n} g_i, s\Big)$ for $i = 1, \dots, n$.

To do so, we will prove that $C_s^{i+s} = \Phi\Big(\prod_{r=1;r\neq i}^{i+s} g_r, s\Big)$ for $s = 1, \dots, n-2$ and $i = 1, \dots, n-s-1$.

Let us make induction on s. For $s = 1$, we get by definition that $C_1^{i+1} = \Phi(g_{i+1}, C_{i-1}^{i-1})$. Now it is clear that $C_j^j = \Phi\Big(\prod_{r=1}^{j} g_r, s\Big)$ for every $j = 1, \dots, n-1$. Therefore

$$C_1^{i+1} = \Phi(g_{i+1}, C_{i-1}^{i-1}) = \Phi\Big(g_{i+1}, \Phi\Big(\prod_{r=1}^{i-1} g_r, s\Big)\Big) = \Phi\Big(\prod_{r=1;r\neq i}^{i+1} g_r, s\Big).$$

Suppose now that $C_{s-1}^{i+s-1} = \Phi\Big(\prod_{r=1;r\neq i}^{i+s-1} g_r, s\Big)$. Then, by definition,

$$C_s^{i+s} = \Phi(g_{i+s}, C_{s-1}^{i+s-1}) = \Phi\Big(g_{i+s}, \Phi\Big(\prod_{r=1;r\neq i}^{i+s-1} g_r, s\Big)\Big) = \Phi\Big(\prod_{r=1;r\neq i}^{i+s} g_r, s\Big).$$

Thus $C_{n-1-i}^{n-1} = C_{n-1-i}^{i+n-1-i} = \Phi\Big(\prod_{r=1;r\neq i}^{i+(n-1-i)} g_r, s\Big) = \Phi\Big(\prod_{r=1;r\neq i}^{n-1} g_r, s\Big)$.

Therefore

$$f_i^n = \Phi\Big(g_n, \Phi\Big(\prod_{r=1;r\neq i}^{n-1} g_r, s\Big)\Big) = \Phi\Big(\prod_{r=1;r\neq i}^{n} g_r, s\Big),$$

and so user $\mathcal{U}_i$ computes $\Phi(g_i, f_i^n) = \Phi\Big(g_i, \Phi\Big(\prod_{r=1;r\neq i}^{n} g_r, s\Big)\Big) = \Phi\Big(\prod_{r=1}^{n} g_r, s\Big)$, as we wanted to show. □

References

[1] G. Ateniese, M. Steiner, G. Tsudik, New multiparty authentication services and key agreement protocols, IEEE Journal of Selected Areas in Communications, vol. 18(4), 1–13, 2000.

[2] J.-J. Climent, P. R. Navarro, L. Tortosa, An extension of the noncommutative Bergman's ring with a large number of noninvertible elements, Applicable Algebra in Engineering, Communication and Computing, 25(5), 347–361, 2014.

[3] J.-J. Climent, J.A. Lopez-Ramos, L. Tortosa, Public Key Protocols over the Ring $E_p^{(m)}$, ArXiv.

[4] W.D. Diffie, M.E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory, vol. 22(6), 644–654, 1976.

[5] A.A. Kamal, A.M. Youssef, Cryptanalysis of a key exchange protocol based on the endomorphisms ring $\mathrm{End}(\mathbb{Z}_p \times \mathbb{Z}_{p^2})$, Applicable Algebra in Engineering, Communication and Computing 23(3), 143–149, 2012.

[6] U. Maurer, S. Wolf, The Diffie-Hellman protocol, Designs, Codes and Cryptography 19, 147–171, 2000.

[7] G. Maze, Algebraic Methods for Constructing One-way Trapdoor Functions, Ph.D. Thesis, University of Notre Dame, April 2003.

[8] G. Maze, C. Monico, J. Rosenthal, Public key cryptography based on semigroup actions, Advances of Mathematics of Communications, vol. 1(4), 489–507, 2007.

[9] G. Micheli, Cryptanalysis of a non-commutative key exchange protocol, Advances in Mathematics of Communications 9(2), 247–253, 2015.

[10] O. Gnilke, The Semigroup Action Problem in Cryptography, Ph.D Thesis, University College Dublin, December 2014.

[11] N. Qiuna, ECDH-based Scalable Distributed Key Management Scheme for Secure Group Communication, Journal of Computers 9(1), 153–160, 2014.

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

[13] R. Schnyder, J.A. Lopez-Ramos, J. Rosenthal, D. Schipani, An active attack on a multiparty key exchange protocol, Journal of Algebra Combinatorics Discrete Structures and Applications 3(1), 31–36, 2016.

[14] M. Steiner, G. Tsudik, M. Waidner, Diffie-Hellman key distribution extended to group communication, Proceedings of the 3rd ACM Conference on Computer and Communications Security, ACM: New York, NY, 31–37, 1996.

[15] M. Steiner, G. Tsudik, M. Waidner, Key agreement in dynamic peer groups. IEEE Transactions of Parallel and Distributed Systems, 11(8), 769–780, 2000.

[16] R. Steinwandt, A. Surez Corona, Cryptanalysis of a 2-party key establishment based on a semigroup action problem, Advances in Mathematics of Communications 5(1), 8792, 2011.

[17] J. Van der Merwe, D. Dawoud, S. McDonald, A survey on peer-to-peer key management for mobile ad hoc networks, ACM Computing Surveys 39 (1) 2007.