

Non-associative key establishment protocols and their implementation

A. G. Kalka¹ M. Teicher²

18 December 2013

Abstract

We provide implementation details for non-associative key establishment protocols. In particular, we describe the implementation of non-associative key establishment protocols for all left self-distributive and all mutually left distributive systems.

Contents

1	Introduction	1
2	LD-systems and their generalizations	2
3	Non-associative KEPs for mutually left distributive systems	4
4	Implementation	7
	References	9

1 Introduction

Currently public-key cryptography still relies mainly on a few number-theoretic problems which remain still unbroken. Nevertheless, after the advent of quantum computers, systems like RSA, Diffie- Hellman and ECC will be broken easily [11]. Under the label Post Quantum Cryptography, there have been several efforts to develop new cryptographic primitives which may also serve for the post quantum computer era. One approach became later known as

non-commutative cryptography where the commutative groups and rings involved in number-theoretic problems are replaced by non-commutative structures, and we consider computational problems therein [1]. The scope of non-commutative cryptography was broadened in [10, 7] as we go beyond non-commutative, associative binary operations. We utilize non-associative binary operations, i.e. magmas, thus hoping to establish *non-associative public-key cryptography*. Here we focus on key establishment protocols (KEPs) as cryptographic primitives, because they are the most important and the hardest to construct. In particular, the seminal Anshel-Anshel-Goldfeld (AAG) KEP for monoids and groups [5] was generalized to a general AAG-KEP for magmas in [10, 7] which emphasize the important and integrating role of the AAG protocol in non-commutative and commutative cryptography. Left self-distributive (LD) systems (and their generalizations) naturally emerge as possible non-associative platform structures for this AAG-KEP for magmas. Non-associative key establishment protocols for all LD-, multi-LD-, and other left distributive systems were introduced in [9, 8]. Braid groups (and their finite quotients), matrix groups and Laver tables as natural platform LD-structures were discussed in [10, 7, 9, 8]. The purpose of this paper is to provide details how our non-associative KEPs can be implemented for all the systems given in [9, 8]. We hope this will encourage cryptanalytic examination of these new and innovative non-associative KEPs.

Outline. In section 2 we provide examples of LD-systems and mutually left distributive systems. Section 3 describes the most improved non-associative KEP (for all mutually left distributive systems). It contains all other KEPs from [9, 8] as special cases. Finally, section 4 provides implementation details and pseudo-code.

2 LD-systems and their generalizations

Definition 2.1 (1) A left self-distributive (LD) system $(S, *)$ is a set S equipped with a binary operation $*$ on S which satisfies the left self-distributivity law

$$x * (y * z) = (x * y) * (x * z) \quad \text{for all } x, y, z \in S.$$

(2) Let I be an index set. A multi-LD-system $(S, (*_i)_{i \in I})$ is a set S equipped with a family of binary operations $(*_i)_{i \in I}$ on S such that

$$x *_i (y *_j z) = (x *_i y) *_j (x *_i z) \quad \text{for all } x, y, z \in S$$

is satisfied for every i, j in I . Especially, it holds for $i = j$, i.e., $(S, *_i)$ is an LD-system. If $|I| = 2$ then we call S a bi-LD-system.

(3) A mutually left distributive system $(S, *_a, *_b)$ is a set S equipped with two binary operations $*_a, *_b$ on S such that

$$x *_a (y *_b z) = (x *_a y) *_b (x *_a z) \quad x *_b (y *_a z) = (x *_b y) *_a (x *_b z) \quad \text{for all } x, y, z \in S.$$

More vaguely, we will also use the terms *partial multi-LD-system* and simply *left distributive system* if the laws of a multi-LD-system are only fulfilled for special subsets of S or if only some of these (left) distributive laws are satisfied. A mutually left distributive system $(L, *_a, *_b)$ is only a partial bi-LD-system. The left selfdistributivity laws need not hold, i.e., $(L, *_a)$ and $(L, *_b)$ are in general not LD-systems. We list examples of LD-systems, multi-LD-systems and mutually left distributive systems. More details can be found in [3, 4, 7, 9, 8].

Conjugacy. A classical example of an LD-system is $(G, *)$ where G is a group equipped with the conjugacy operation $x * y = x^{-1}yx$ (or $x *^{\text{rev}} y = xyx^{-1}$).

Laver tables. Finite groups equipped with the conjugacy operation are not the only finite LD-systems. Indeed, the so-called *Laver tables* provide the classical example for finite LD-systems. There exists for each $n \in \mathbb{N}$ an unique LD-system $L_n = (\{1, 2, \dots, 2^n\}, *)$ with $k * 1 = k + 1$. The values for $k * l$ with $l \neq 1$ can be computed by induction using the left self-distributive law. Laver tables are also described in [3].

LD-conjugacy. Let G be a group, and $f \in \text{End}(G)$. Set $x *_f y = f(x^{-1}y)x$, then $(G, *_f)$ is an LD-system.

Shifted conjugacy. Consider the braid group on infinitely many strands

$$B_\infty = \langle \{\sigma_i\}_{i \geq 1} \mid \sigma_i \sigma_j = \sigma_j \sigma_i \text{ for } |i - j| \geq 2, \sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j \text{ for } |i - j| = 1 \rangle$$

where inside σ_i the $(i + 1)$ -th strand crosses over the i -th strand. The *shift map* $\partial : B_\infty \rightarrow B_\infty$ defined by $\sigma_i \mapsto \sigma_{i+1}$ for all $i \geq 1$ is an injective endomorphism. Then B_∞ equipped with the shifted conjugacy operations $*$, $\bar{*}$ defined by $x * y = \partial x^{-1} \cdot \sigma_1 \cdot \partial y \cdot x$ and $x \bar{*} y = \partial x^{-1} \cdot \sigma_1^{-1} \cdot \partial y \cdot x$ is a bi-LD-system. In particular, $(B_\infty, *)$ is an LD-system.

Generalized shifted conjugacy in braid groups. Let, for $n \geq 2$, $\delta_n = \sigma_{n-1} \cdots \sigma_2 \sigma_1$. For $p, q \geq 1$, we set $\tau_{p,q} = \delta_{p+1} \partial(\delta_{p+1}) \cdots \partial^{q-1}(\delta_{p+1})$.

Proposition 2.2 $(B_\infty, *_1, *_2)$ with binary operations $x *_i y = \partial^p(x^{-1}) a_i \partial^p(y) x$ ($i = 1, 2$) with $a_1 = a'_1 \tau_{p,p}^{\pm 1} a''_1$, $a_2 = a'_2 \tau_{p,p}^{\pm 1} a''_2$ for some $a'_1, a''_1, a'_2, a''_2 \in B_p$ is a mutually left distributive system if and only if $[a'_1, a'_2] = [a'_2, a'_1] = [a''_1, a''_2] = 1$. (Note that $[a'_1, a''_1]$, $[a'_2, a''_2]$ and $[a''_1, a''_2]$ may be nontrivial. If, in addition $[a'_1, a''_1] = [a'_2, a''_2] = 1$ holds, then $(B_\infty, *_1, *_2)$ is a bi-LD-system.)

Symmetric conjugacy. For a group G , there exists yet another LD-operation. (G, \circ) is an LD-system with $x \circ y = xy^{-1}x$.

f-symmetric conjugacy. Let G be a group, and $f \in \text{End}(G)$ an endomorphism that is also a projector ($f^2 = f$). Then (G, \circ_f) , defined by $x \circ_f y = f(xy^{-1})x$ is an LD-system.

3 Non-associative KEPs for mutually left distributive systems

Here we describe a KEP that works for all mutually left distributive systems, in particular all bi-LD-systems (and all LD-systems). Consider a set L equipped with a pool of binary operations $O_A \cup O_B$ (O_A and O_B non-empty) s.t. the operations in O_A are distributive over those in O_B and vice versa, i.e. the following holds for all $x, y, z \in L$, $*_\alpha \in O_A$ and $*_\beta \in O_B$.

$$x *_\alpha (y *_\beta z) = (x *_\alpha y) *_\beta (x *_\alpha z), \text{ and} \quad (1)$$

$$x *_\beta (y *_\alpha z) = (x *_\beta y) *_\alpha (x *_\beta z). \quad (2)$$

Then $(L, *_\alpha, *_\beta)$ is a mutually left distributive system for all $(*_\alpha, *_\beta) \in O_A \times O_B$. Note that, if $O_A \cap O_B \neq \emptyset$, then $(L, O_A \cap O_B)$ is a multi-LD-system.

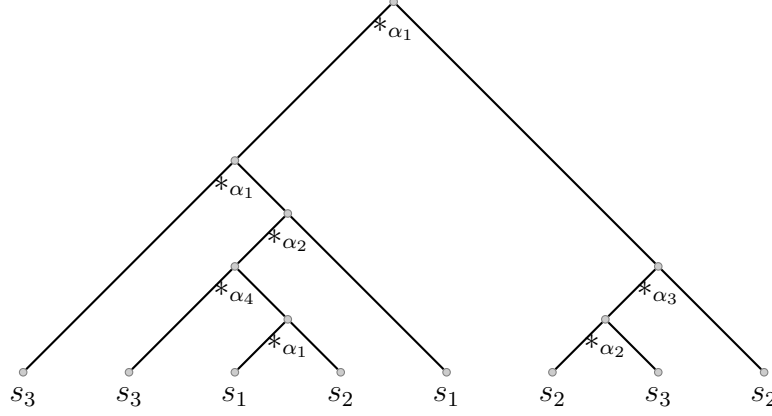
Let $s_1, \dots, s_m, t_1, \dots, t_n \in L$ be some public elements. We denote $S_A = \langle s_1, \dots, s_m \rangle_{O_A}$ and $S_B = \langle t_1, \dots, t_n \rangle_{O_B}$, two submagmas of $(L, O_A \cup O_B)$. For example, an element y of S_A can be described by a planar rooted binary tree T whose k leaves are labelled by these other elements r_1, \dots, r_k with $r_i \in \{s_i\}_{i \leq m}$. Here the tree contains further information, namely to each internal vertex we assign a binary operation $*_i \in O_A$. We use the notation $y = T_{O_A}(r_1, \dots, r_k)$. The subscript O_A tells us that the grafting of subtrees of T corresponds to the operation $*_i \in O_A$. Consider, for example, the element $y = (s_3 *_\alpha (s_3 *_\alpha (s_1 *_\alpha s_2)) *_\alpha s_1) *_\alpha ((s_2 *_\alpha s_3) *_\alpha s_2)$. The corresponding labelled planar rooted binary tree T is displayed in the Figure 3.

Let $*_\alpha \in O_A$ and $*_\beta \in O_B$. By induction over the tree depth, it is easy to show that, for all elements $e, e_1, \dots, e_l \in (L, O_A \cup O_B)$ and all planar rooted binary trees T with l leaves, the following equations hold.

$$e *_\alpha T_{O_B}(e_1, \dots, e_l) = T_{O_B}(e *_\alpha e_1, \dots, e *_\alpha e_l), \quad (3)$$

$$e *_\beta T_{O_A}(e_1, \dots, e_l) = T_{O_A}(e *_\beta e_1, \dots, e *_\beta e_l). \quad (4)$$

Proposition 3.1 (See Proposition 4.1 in [8].) Consider $(L, O_A \cup O_B)$ such that $(L, *_A, *_B)$ is a mutually left distributive system for all $(*_A, *_B) \in O_A \times O_B$, and let $k \in \mathbb{N}$. Then, for all $x = (x_1, \dots, x_k) \in L^k$, $o_A =$

Figure 1: $(s_3 *_{\alpha_1} ((s_3 *_{\alpha_4} (s_1 *_{\alpha_1} s_2)) *_{\alpha_2} s_1)) *_{\alpha_1} ((s_2 *_{\alpha_2} s_3) *_{\alpha_3} s_2) \in S_A$ 

$(*_{A_1}, \dots, *_{A_k}) \in O_A^k$, and $o_B = (*_{B_1}, \dots, *_{B_k}) \in O_B^k$, the iterated left multiplication maps

$$\begin{aligned} \phi_{(x, o_A)} : \quad y &\mapsto x_k *_{A_k} (x_{k-1} *_{A_{k-1}} \cdots *_{A_3} (x_2 *_{A_2} (x_1 *_{A_1} y)) \cdots) \text{ and} \\ \phi_{(x, o_B)} : \quad y &\mapsto x_k *_{B_k} (x_{k-1} *_{B_{k-1}} \cdots *_{B_3} (x_2 *_{B_2} (x_1 *_{B_1} y)) \cdots) \end{aligned}$$

define a magma endomorphisms of (L, O_B) and (L, O_A) , respectively.

In particular, the following equations hold for all $k, l \in \mathbb{N}$, $a, b \in L^k$, $o_A \in O_A^k$, $o_B \in O_B^k$, $e, e_1, \dots, e_l \in L$ and all planar rooted binary trees T with l leaves.

$$\phi_{(a, o_A)}(T_{O_B}(e_1, \dots, e_l)) = T_{O_B}(\phi_{(a, o_A)}(e_1), \dots, \phi_{(a, o_A)}(e_l)), \quad (5)$$

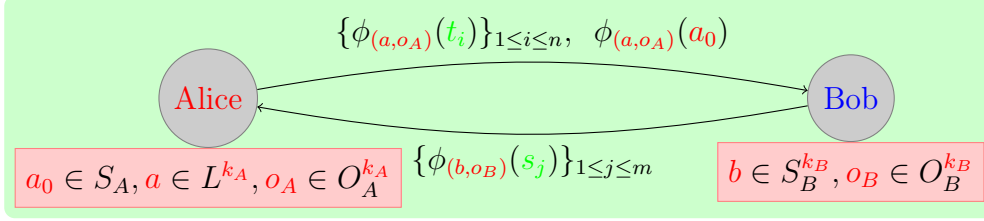
$$\phi_{(b, o_B)}(T_{O_A}(e_1, \dots, e_l)) = T_{O_A}(\phi_{(b, o_B)}(e_1), \dots, \phi_{(b, o_B)}(e_l)) \quad (6)$$

Now, we are going to describe a KEP that applies to any system $(L, O_A \cup O_B)$ as described above. We have two subsets of public elements $\{s_1, \dots, s_m\}$ and $\{t_1, \dots, t_n\}$ of L . Also, recall that $S_A = \langle s_1, \dots, s_m \rangle_{O_A}$ and $S_B = \langle t_1, \dots, t_n \rangle_{O_B}$. Alice and Bob perform the following protocol steps.

Protocol KEY ESTABLISHMENT FOR THE PARTIAL MULTI-LD-SYSTEM

$$(L, O_A \cup O_B).$$

- 1 Alice generates her secret key $(a_0, a, o_A) \in S_A \times L^{k_A} \times O_A^{k_A}$, and Bob chooses his secret key $(b, o_B) \in S_B^{k_B} \times O_B^{k_B}$. Denote $o_A = (*_{A_1}, \dots, *_{A_{k_A}})$ and

Figure 2: KEP FOR THE PARTIAL MULTI-LD-SYSTEM $(L, O_A \cup O_B)$.

$o_B = (*_{B_1}, \dots, *_{B_{k_B}})$, then Alice's and Bob's secret magma morphisms α and β are given by

$$\begin{aligned}\alpha(y) &= a_{k_A} *_{A_{k_A}} (a_{k_A-1} *_{A_{k_A-1}} \cdots *_{A_3} (a_2 *_{A_2} (a_1 *_{A_1} y)) \cdots) \quad \text{and} \\ \beta(y) &= b_{k_B} *_{B_{k_B}} (b_{k_B-1} *_{B_{k_B-1}} \cdots *_{B_3} (b_2 *_{B_2} (b_1 *_{B_1} y)) \cdots),\end{aligned}$$

respectively.

- 2 $(\alpha(t_i))_{1 \leq i \leq n} \in L^n$, $p_0 = \alpha(a_0) \in L$, and sends them to Bob. Bob computes the vector $(\beta(s_j))_{1 \leq j \leq m} \in L^m$, and sends it to Alice.
- 3 Alice, knowing $a_0 = T_{O_A}(r_1, \dots, r_l)$ with $r_i \in \{s_1, \dots, s_m\}$, computes from the received message

$$T_{O_A}(\beta(r_1), \dots, \beta(r_l)) = \beta(T_{O_A}(r_1, \dots, r_l)) = \beta(a_0).$$

And Bob, knowing for all $1 \leq j \leq k_B$, $b_j = T_{O_B}^{(j)}(u_{j,1}, \dots, u_{j,l_j})$ with $u_{j,i} \in \{t_1, \dots, t_n\} \forall i \leq l_j$ for some $l_j \in \mathbb{N}$, computes from his received message for all $1 \leq j \leq k_B$

$$T_{O_B}^{(j)}(\alpha(u_{j,1}), \dots, \alpha(u_{j,l_j})) = \alpha(T_{O_B}^{(j)}(u_{j,1}, \dots, u_{j,l_j})) = \alpha(b_j).$$

- 4 Alice computes $K_A = \alpha(\beta(a_0))$. Bob gets the shared key by

$$K_B := \alpha(b_{k_B}) * (\alpha(b_{k_B-1}) * (\cdots (\alpha(b_2) * (\alpha(b_1) * p_0)) \cdots)) \stackrel{\alpha \text{ homo}}{=} K_A.$$

Here the operation vectors $o_A \in O_A^{k_A}$ and $o_B \in O_B^{k_B}$ are part of Alice's and Bob's private keys. Also explicit expressions of $a_0 \in S_A$ and all $b_i \in S_B$ as treewords $T, T^{(i)}$ (for all $1 \leq i \leq k_B$) are also parts of the private keys - though we did not mention it explicitly in step 1 of the protocols. But here T_{O_A} and T_{O_B}' also contain all the information about the grafting operations (in O_A or O_B , respectively) at the internal vertices of $T, T^{(1)}, \dots, T^{(k_B)}$.

4 Implementation

Planar rooted binary trees We need some efficient way to encode the planar rooted binary tree which determines the bracket structure of an element given as product of other elements. Let PBT_n denote the set of planar rooted binary trees (also known as *full binary trees*) with n internal nodes (and $n + 1$ leaves), then $|PBT_n| = Cat(n)$ where $Cat(n) = \frac{1}{n+1} \binom{2n}{n}$ denotes the n -th Catalan number. There exists a rich variety of other Catalan sets with well understood bijections between them, e.g., diagonal avoiding paths (aka mountain ranges), polygon triangulations, Dyck words, planar rooted trees (not only binary) and non-crossing partitions. We use the following succinct representation for Catalan sets taken from [2]. Denote $[n] = \{1, \dots, n\}$. To each $T \in PBT_n$ we associate a vector (array) $T \in [n]^n$ such that $T[i] \leq T[j]$ for $i < j$ and $T[i] \leq i$ for all $i \in [n]$. By abuse of notation we call the set of such vectors in $[n]^n$ also PBT_n .

```

function EvaluateTree;
Input:  $(T, o, (e_1, \dots, e_{n+1})) \in PBT_n \times O^n \times L^{n+1}$ .
Output:  $e = T_o(e_1, \dots, e_{n+1})$ 
for  $j := n$  to  $1$  by  $-1$  do
     $pos := T[j]$ ;
     $Seq[pos] := Seq[pos] *_{o[pos]} Seq[pos + 1]$ ;
     $Remove(\sim Seq, pos + 1)$ ;    $Remove(\sim T, pos)$ ;    $Remove(\sim o, pos)$ ;
end
return  $Seq[1]$ ;

```

Let L be a magma and O be a set of binary operations on L . Given a vector of operations $o = (*_{o[1]}, \dots, *_{o[n]}) \in O^n$ and a sequence of leave elements $(e_1, \dots, e_{n+1}) \in L^{n+1}$, then the function **EvaluateTree** evaluates the product of e_1, \dots, e_{n+1} where the bracket structure is given by the tree T and the operations on the internal vertices of T are given by o . For example, the tree in Figure 1 is given by $T = [1, 1, 2, 2, 3, 6, 6]$ and $o = (*_{\alpha_2}, *_{\alpha_3}, *_{\alpha_1}, *_{\alpha_4}, *_{\alpha_2}, *_{\alpha_1}, *_{\alpha_1})$.

Protocol implementation. Now, let (L, O_A, O_B) be as described in the KEP. We fix some distributions on L , O_A and O_B , so that we may generate random elements from these sets (according to these distributions). Given $m_a, m_B \in \mathbb{N}$, Alice and Bob first choose random vectors $\mathcal{G}_A = (s_1, \dots, s_{m_A}) \in L^{m_A}$ and $\mathcal{G}_B = (t_1, \dots, t_{m_B}) \in L^{m_B}$ which determine the public submagmas $S_A = \langle \mathcal{G}_A \rangle_{O_A}$ and $S_B = \langle \mathcal{G}_B \rangle_{O_B}$, respectively. Then Alice and Bob generate their secret, public and shared keys as described in the following functions.

The KEPs were implemented using MAGMA [12] which also contains an implementation of braid groups following [6].

```

function GeneratePrivateKeyAlice;
Input:  $\mathcal{G}_A \in L^{m_A}$ .
Output:  $(Ia_0, Ta_0, oa_0, a_0, a, oA) \in$ 
 $[m_A]^{n_{a_0}+1} \times PBT_{n_{a_0}} \times O_A^{n_{a_0}} \times L \times L^{k_A} \times O_A^{k_A}$ 
 $Ia_0 \leftarrow \text{Random}([m_A]^{n_{a_0}});$ 
for  $i := 1$  to  $n_{a_0} + 1$  do  $Seqa_0[i] := \mathcal{G}_A[Ia_0[i]];$ 
 $Ta_0 \leftarrow \text{Random}(PBT_{n_{a_0}}); \quad oa_0 \leftarrow \text{Random}(O_A^{n_{a_0}});$ 
 $a_0 := \text{EvaluateTree}(Ta_0, oa_0, Seqa_0);$ 
 $a \leftarrow \text{Random}(L^{k_A}); \quad oA \leftarrow \text{Random}(O_A^{k_A});$ 
return  $(Ia_0, Ta_0, oa_0, a_0, a, oA);$ 

function GeneratePrivateKeyBob;
Input:  $\mathcal{G}_B \in L^{m_B}$ .
Output:  $(Ib, Tb, ob, b, oB) \in$ 
 $([m_B]^{n_b+1})^{k_B} \times (PBT_{n_b})^{k_B} \times (O_B^{n_b})^{k_B} \times L^{k_B} \times O_B^{k_B}$ 
for  $k := 1$  to  $k_B$  do
 $Ib[k] \leftarrow \text{Random}([m_B]^{n_b});$ 
for  $i := 1$  to  $n_b + 1$  do  $Seqb[k][i] := \mathcal{G}_B[Ib[k][i]];$ 
 $Tb[k] \leftarrow \text{Random}(PBT_{n_b}); \quad ob \leftarrow \text{Random}(O_B^{n_b});$ 
 $b[k] := \text{EvaluateTree}(Tb[k], ob[k], Seqb[k]);$ 
end
 $oB \leftarrow \text{Random}(O_B^{k_B});$ 
return  $(Ib, Tb, ob, b, oB);$ 

function GeneratePublicKeyAlice;
Input:  $(a, oA, a_0, \mathcal{G}_B) \in L^{k_A} \times O_A^{k_A} \times L \times L^{m_B}$ .
Output:  $(p_A, p_0) \in L^{m_A} \times L$ 
for  $k := 1$  to  $m_A$  do
 $p_A[k] := \mathcal{G}_B[k];$ 
for  $i := 1$  to  $k_A$  do  $p_A[k] := a[i] *_{oA[i]} p_A[k];$ 
end
 $p_0 := a_0;$ 
for  $i := 1$  to  $k_A$  do  $p_0[k] := a[i] *_{oA[i]} p_0[k];$ 
return  $(p_A, p_0);$ 

function GeneratePublicKeyBob;
Input:  $(b, oB, \mathcal{G}_A) \in L^{k_B} \times O_B^{k_B} \times L^{m_A}$ .
Output:  $p_B \in L^{m_B}$ 
for  $k := 1$  to  $m_B$  do
 $p_B[k] := \mathcal{G}_A[k];$ 
for  $i := 1$  to  $k_B$  do  $p_B[k] := b[i] *_{oB[i]} p_B[k];$ 
end
return  $p_B;$ 

```



```

function GenerateSharedKeyAlice;
Input:  $(Ia_0, Ta_0, oa_0, a, oA, p_B) \in$ 
 $[m_A]^{n_{a_0}+1} \times PBT_{n_{a_0}} \times O_A^{n_{a_0}} \times L^{k_A} \times O_A^{k_A} \times L^{m_B}.$ 
Output:  $K_A \in L$ 
 $K_A := \text{EvaluateTree}(Ta_0, oa_0, (p_B[Ia_0[i]])_{i \leq n_{a_0}+1});$ 
for  $k := 1$  to  $k_A$  do  $K_A := a[k] *_{oA[k]} K_A;$ 
return  $K_A;$ 

function GenerateSharedKeyBob;
Input:  $(Ib, Tb, ob, b, oB, p_A, p_0) \in$ 
 $([m_B]^{n_b+1})^{k_B} \times (PBT_{n_b})^{k_B} \times (O_A^{n_b})^{k_B} \times L^{k_B} \times O_B^{k_B} \times L^{m_A} \times L.$ 
Output:  $K_A \in L$ 
Initialize  $lfactors := [];$   $K_B := p_0;$ 
for  $k := 1$  to  $k_B$  do
 $lfactors[k] := \text{EvaluateTree}(Tb[k], ob[k], (p_A[Ib[k][i]])_{i \leq n_b+1});$ 
 $K_B := lfactors[k] *_{oB[k]} K_B;$ 
end
return  $K_A;$ 

```

References

- [1] Vladimir Shpilrain Alexei Myasnikov and Alexander Ushakov. *Non-commutative Cryptography and Complexity of Group-theoretic Problems*, volume 177 of *Mathematical Surveys and Monographs*. 2011.
- [2] Matej Crepinsek and Luka Mernik. An efficient representation for solving Catalan number related problems. *International Journal of Pure and Applied Mathematics*, 56(4):589–604, 2009.
- [3] Patrick Dehornoy. *Braids and Self-Distributivity*. Progress in Math. , No. 192. Birkhäuser, 2000.
- [4] Patrick Dehornoy. Using shifted conjugacy in braid-based cryptography. *Contemporary Mathematics*, 418:65–73, 2006.
- [5] Michael Anshel Iris Anshel and Dorian Goldfeld. An algebraic method for public-key cryptography. *Mathematical Research Letters*, 6(3):287–291, 1999.
- [6] Sang Jin Lee Jae Woo Han Jae Choon Cha, Ki Hyoung Ko and Jung Hee Cheon. An efficient implementation of braid groups.

- Advances in Cryptology - ASIACRYPT 2001, Lecture Notes in Computer Science*, 2248:144–156, 2001.
- [7] Arkadius Kalka. Non-associative public-key cryptography. *arXiv*, abs/1210.8270, 2012.
- [8] Arkadius Kalka and Mina Teicher. Iterated LD-Problem in non-associative key establishment. *arXiv*, 2013.
- [9] Arkadius Kalka and Mina Teicher. Non-associative key establishment for left distributive systems. *Groups Complexity Cryptology*, 5(2), 2013.
- [10] Arkadius G. Kalka. *Linear representations of braid groups and braid-based cryptography*. PhD thesis, Ruhr-Universität Bochum, 2007.
- [11] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.
- [12] C. Playoust W. Bosma, J. Cannon. The magma algebra system, i: The user language. *J. Symbolic Comput.*, 24:235–265, 1997.

Author addresses

1. **A. G. Kalka**, Department of Mathematics, Bar-Ilan University, Ramat Gan 52900, ISRAEL.
<http://homepage.ruhr-uni-bochum.de/arkadius.kalka/>
<mailto:arkadius.kalka@rub.de>
2. **M. Teicher**, Department of Mathematics, Bar-Ilan University, Ramat Gan 52900, ISRAEL.
<mailto:teicher@math.biu.ac.il>