Efficient Key-policy Attribute-based Encryption for General Boolean Circuits from Multilinear Maps

Constantin Cătălin Drăgan and Ferucio Laurenţiu Ţiplea

Department of Computer Science
"Al.I.Cuza" University of Iaşi
Iaşi 700506, Romania
e-mail: {constantin.dragan,fltiplea}@info.uaic.ro

Abstract. We propose an efficient Key-policy Attribute-based Encryption (KP-ABE) scheme for general (monotone) Boolean circuits based on secret sharing and on a very particular and simple form of leveled multilinear maps, called *chained multilinear maps*. The number of decryption key components is substantially reduced in comparison with the scheme in [6], and the size of the multilinear map (in terms of bilinear map components) is less than the Boolean circuit depth, while it is quadratic in the Boolean circuit depth for the scheme in [6]. Moreover, it is much easier to find chained multilinear maps than leveled multilinear maps. Selective security of the proposed schemes in the standard model is proved, under the decisional multilinear Diffie-Hellman assumption.

1 Introduction

Attribute-based encryption (ABE) was introduced in [10] as a generalization of *identity-based encryption* [11]. There are two forms of ABE: *key-policy ABE* (KP-ABE) and *ciphertext-policy ABE* (CP-ABE) [8, 2]. A KP-ABE scheme encrypts messages taking into consideration specific sets of attributes; decryption keys are distributed for an entire access structure build over the set of attributes so that correct decryption is allowed only to authorized sets of attributes (defined by the access structure). A CP-ABE scheme proceeds somehow vice-versa than a KP-ABE scheme: messages are encrypted together with access structures while decryption keys are given for specific sets of attributes. A general approach to specify access structures is the one based on Boolean circuits with exactly one output wire [12]. Roughly speaking, each attribute is associated with an input wire of the Boolean circuit. A set of attributes evaluates the circuit to one of the truth values. If this is the truth value true, then the set of attributes is considered authorized. The set of all authorized sets of attributes defines an access structure.

This paper focuses on KP-ABE. The first KP-ABE scheme was proposed in [8], where the access structures were specified by monotone Boolean formulas (Boolean circuits of fan-out one with no negation gates). An extension to non-monotonic Boolean formulas has later been proposed in [9]. The transition to the general case (access structures defined by general Boolean circuits) faces the backtracking attack. The first solution to this problem has been proposed in [6]. While the schemes in [8, 9] use secret sharing techniques in conjunction with bilinear maps, the scheme in [6] is based on leveled multilinear maps ("chains" of bilinear maps with some special property) which are much more complex mathematical structures than bilinear maps. A construction from the Learning With Errors (LWE) problem has also been proposed [7]. The two solutions [6, 7] are built for access structures definable by general Boolean circuits with fan-in two. Inspired by [7], Boneh et.al. [3] have proposed a KP-ABE scheme for functions that can be represented as (polynomial-size) arithmetic circuits. The scheme is based on the LWE problem and it can naturally handle access structures definable as general Boolean circuits. Its decryption key size is quadratic in the circuit depth, while for the schemes proposed in [6, 7] it is linear in the number of Boolean gates or wires in the circuit. On the other side, the size of its public parameters is quadratic in the number of input wires, while for the schemes in [6, 7] it is linear (in the number of input wires).

Contribution The KP-ABE schemes for general Boolean circuits proposed so far are either based on leveled multilinear maps [6] or on the LWE problem [7, 3]. In this paper we propose a new KP-ABE scheme based on a very particular and simple form of leveled multilinear maps, called *chained multilinear maps.* Our scheme is more efficient than the scheme in [6] both in terms of the decryption key size and of the multilinear map size and construction. The size of the chained multilinear maps we use is less than the circuit depth, while the leveled multilinear maps used in [6] have a quadratic size in the circuit depth. Moreover, it is much easier to define chained multilinear maps than leveled multilinear maps: once defined k bilinear maps from $G_i \times G_1$ into G_{i+1}, $1 \leq i \leq k$, any generator of the group G_1, together with the bilinear maps, defines a chained multilinear map.

Our construction uses general Boolean circuits where the logic gates of fan-out two or more are split into logics gates of fan-out one and FANOUT-gates whose role is to multiply the output of logic gates. Then, a secret sharing procedure works top-down to share some secret, and a bottom-up procedure reconstructs a "hidden" form of the secret by using chained multilinear maps. The generator of the chained multilinear map is changed each time a FANOUT-level (level that contains FANOUT-gates) is reached. Decryption key components are assigned to input wires, FANOUT-gates, and to circuit FANOUT-levels. The size of the decryption key is thus a third of the size of the decryption key in the construction in [6].

The selective security of our KP-ABE schemes is proved in the standard model under the decisional multilinear Diffie-Hellman assumption.

Paper organization The paper is organized into eight sections. The next section fixes the basic terminology and notation used throughout the paper. The third section discusses the scheme in [6] and how it thwarts the backtracking attack, and gives an informal overview of our solution. Our construction is presented in the fourth section, its security is discussed in the fifth one, while the sixth section presents some comparisons between our scheme and the one in [6]. Section seven proposes some extensions of our scheme, and the last one concludes the paper.

2 Preliminaries

This section fixes the terminology and notation used in our paper.

Access structures Recall first that [12], given a non-empty finite set $\mathcal{U}$ whose elements are called *attributes* in our paper, an *access structure* over $\mathcal{U}$ is any set $\mathcal{S}$ of non-empty subsets of $\mathcal{U}$. $\mathcal{S}$ is called *monotone* if it satisfies the following monotonicity property:

$$(\forall B \subseteq \mathcal{U})((\exists A \in \mathcal{S})(A \subseteq B) \;\Rightarrow\; B \in \mathcal{S})$$

The subsets (of $\mathcal{U}$) that are in $\mathcal{S}$ are called *authorized sets*, while those not in $\mathcal{S}$, *unauthorized sets.* An authorized set A is *minimal* if there is no $B \in \mathcal{S}$ such that $B \subset A$.

It is customary to represent access structures by Boolean circuits (for more details about Boolean circuits the reader is refereed to [1]). A Boolean circuit has a number of input wires (which are not gate output wires), a number of output wires (which are not gate input wires), and a number of OR-, AND-, and NOT-gates. The OR- and AND-gates have two input wires, while NOT-gate has one input wire. All of them may have more than one output wire. That is, the fan-in of the circuit is at most two, while the fan-out may be arbitrarily large but at least one. A Boolean circuit is *monotone* if it does not have NOT-gates, and it is of *fan-out one* if all gates have fan-out one. In this paper all Boolean circuits have exactly one output wire (for the sake of simplicity they will also be called "Boolean circuits"). Boolean circuits of fan-out one (with one output wire) correspond to *Boolean formulas*.

If the input wires of a Boolean circuit $\mathcal{C}$ are in a one-to-one correspondence with the elements of $\mathcal{U}$, we will say that $\mathcal{C}$ is a Boolean circuit over $\mathcal{U}$. Each $A \subseteq \mathcal{U}$ evaluates the circuit $\mathcal{C}$ to one of the Boolean values 0 or 1 by simply assigning 1 to all input wires associated to elements in A, and 0 otherwise. We will write $\mathcal{C}(A)$ for the value obtained by evaluating $\mathcal{C}$ for A. The access structure defined by a Boolean circuit $\mathcal{C}$ is the set of all A that evaluates $\mathcal{C}$ to 1.

Attribute-based encryption A KP-ABE scheme consists of four probabilistic polynomial-time (PPT) algorithms [8]:

Setup(λ)**:** this is a PPT algorithm that takes as input the security parameter λ and outputs a set of public parameters PP and a master key MSK;

Enc(m, A, PP)**:** this is a PPT algorithm that takes as input a message m, a non-empty set of attributes $A \subseteq \mathcal{U}$, and the public parameters, and outputs a ciphertext E;

KeyGen$(\mathcal{C}, MSK)$**:** this is a PPT algorithm that takes as input an access structure $\mathcal{C}$ (given as a Boolean circuit) and the master key MSK, and outputs a decryption key D (for the entire Boolean circuit $\mathcal{C}$);

Dec(E, D)**:** this is a deterministic polynomial-time algorithm that takes as input a ciphertext E and decryption key D, and outputs a message m or the special symbol $\perp$.

The following correctness property is required to be satisfied by any KP-ABE scheme: for any $(PP, MSK) \leftarrow Setup(\lambda)$, any Boolean circuit $\mathcal{C}$ over a set $\mathcal{U}$ of attributes, any message m, any $A \subseteq \mathcal{U}$, and any $E \leftarrow Enc(m, A, PP)$, if $\mathcal{C}(A) = 1$ then $m = Dec(E, D)$, for any $D \leftarrow KeyGen(\mathcal{C}, MSK)$.

Security models We consider the standard notion of selective security for KP-ABE [8]. Specifically, in the *Init* phase the *adversary* (PPT algorithm) announces the set A of attributes that he wishes to be challenged upon, then in the *Setup* phase he receives the public parameters PP of the scheme, and in *Phase 1* oracle access to the decryption key generation oracle is granted for the adversary. In this phase, the adversary issues queries for decryption keys for access structures defined by Boolean circuits $\mathcal{C}$, provided that $\mathcal{C}(A) = 0$. In the *Challenge* phase the adversary submits two equally length messages m_0 and m_1 and receives the ciphertext associated to A and one of the two messages, say m_b, where $b \leftarrow \{0, 1\}$. The adversary may receive again oracle access to the decryption key generation oracle (with the same constraint as above); this is *Phase 2*. Eventually, the adversary outputs a guess $b' \leftarrow \{0, 1\}$ in the *Guess* phase.

The *advantage* of the adversary in this game is defined as $P(b' = b) - 1/2$. The KP-ABE scheme is *secure* (in the selective model) if any adversary has only a negligible advantage in the selective game described above.

Leveled multilinear maps and the decisional MDH assumption [6, 5, 4] Given G_1, G_2, and G_T three multiplicative cyclic groups of prime order p, a map $e : G_1 \times G_2 \to G_T$ is called *bilinear* if it satisfies:

- $e(x^a, y^b) = e(x, y)^{ab}$, for any $x \in G_1$, $y \in G_2$, and $a, b \in \mathbb{Z}_p$;
- $e(g_1, g_2)$ is a generator of G_T, for any generators g_1 of G_1 and g_2 of G_2.

Given k multiplicative groups $G_1, \ldots, G_k$ of the same prime order p with generators $g_1, \ldots, g_k$, respectively, a set $\mathbf{e} = \{e_{i,j} : G_i \times G_j \to G_{i+j} | i, j \geq 1,\ i+j \leq k\}$ of bilinear maps is caled a *leveled multilinear map* if $e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab}$, for all $i, j \geq 1$ with $i+j \leq k$ and all $a, b \in \mathbb{Z}_p$.

The *decisional Multilinear Diffie-Hellman* (MDH) problem for $\mathbf{e}$ is the problem to distinguish between $g_k^{sc_1 \cdots c_k}$ and a random element in G_k given $g_1, g_1^s, g_1^{c_1}, \ldots, g_1^{c_k}$, where $s, c_1, \ldots, c_k$ are randomly chosen from $\mathbb{Z}_p$.

The *decisional MDH assumption* for $\mathbf{e}$ states that no PPT algorithm $\mathcal{A}$ can solve the decisional MDH problem for $\mathbf{e}$ with more than a negligible advantage.

The (leveled) multilinear maps defined as above should be viewed at a generic level. Practical construction have also been obtained: [5] proposes a construction based on ideal lattices, while [4] proposes a construction based on integers. Both of them are developed inside the formalism called *graded encoding systems.*

3 An Informal View of Our Construction

Our approach to construct a KP-ABE scheme uses both secret sharing as in [8] and multilinear maps as in [6]. To clearly understand how these two techniques are combined, let us briefly recall them.

The approach in [8] works only for monotone Boolean formulas. The main idea is quite elegant and simple, and can be summarized as follows:

- choose a bilinear map $e : G_1 \times G_1 \to G_2$ and a generator g of G_1;
- to encrypt a message m by a set A of attributes, just multiply m by $e(g, g)^{ys}$, where y is a random integer chosen in the setup phase and s is a random integer chosen in the encryption phase. Moreover, an attribute dependent quantity is also computed for each attribute $i \in A$;
- the integer y is then shared to all attributes so that it can be recovered only by the authorized sets of attributes (the authorized sets are defined by monotone Boolean formulas). The shares associated to attributes are then used to compute the decryption key (which consists of a key component for each attribute);
- in order to decrypt $me(g, g)^{ys}$, one has to compute $e(g, g)^{ys}$. This can be done only if A is an authorized set of attributes. The computation of $e(g, g)^{ys}$ is bottom-up, starting from the key components associated to the attributes in A.

It was pointed out in [6] that the construction in [8] cannot directly be used to design KP-ABE schemes for general Boolean circuits. The reason is the *backtracking attack* [6]. In case of OR-gates, any value computed at an input wire should be the same with the value computed at the other input wire (this is because of the way secrets are shared at OR-gates). Therefore, knowing the value at one of the input wires of an OR-gate implicitly leads to the knowledge of the value at the other input wire (although these values are computed by different workflows), and this value can further "migrate" to other gates if the gate fan-out is two or more. This aspect leads to the possibility of computing the value at the output wire of the circuit starting from values associated to some unauthorized set of attributes. The backtracking attack cannot occur when access structures are defined by Boolean formulas as in [8] because, in such cases, the input wires of OR-gates are not used by any other gates.

In order to thwart the backtracking attack, [6] uses a "one-way" construction in evaluating general monotone Boolean circuits (the encryption technique is almost the same as the one in [8]). The idea is the next one:

- consider a leveled multilinear map (as the one in the previous section);
- the key components are associated to the input wires of the circuit and to each gate output wire (in [6], each gate has one output wire which may further be used by more than one gate);
- the circuit is evaluated bottom-up and the values associated to output wires of gates on level j are powers of g_{j+1};
- as the mappings $e_{i,j}$ work only in the "forward" direction, it is not feasible to invert values on the level $j+1$ in order to obtain values on the level j, defeating thus the backtracking attack.

Now, our approach can be described as follows. First, the logic gates of fan-out two or more are split into logic gates of fan-out one and FANOUT-gates. A FANOUT-gate multiplies the output of a logic gate. Then, a secret sharing procedure is used top-down to share a secret to all attributes (input wires of the circuit). There are two main tricks here:

1. the shares associated to the output wires of a FANOUT-gate are processed via a random value associated to the input wire of the gate, and this random value is passed down to logic gates for sharing;
2. the share associated to the output wire of a logic gate is shared among its input wires by taking into consideration the input wire levels of the gate.

When all input wires of the circuit get their shares, a "secret reconstruction" procedure evaluates bottom-up the circuit by computing values to each wire. Each value is the power of some group generator, and the generator is changed only when a FANOUT-level (level that contains FANOUT-gates) is reached. Due to the way secrets are shared, the multilinear map we use consists of just $r+1$ bilinear maps $e_i : G_i \times G_1 \to G_{i+1}$ with no other constraints, $1 \leq i \leq r+1$ (r denotes the number of FANOUT-levels). As the bilinear maps e_i work only in the forward direction, our scheme defeats the backtracking attack.

4 Our Construction

In this section we propose a KP-ABE scheme for monotone Boolean circuits based on a particular and simpler form of leveled multilinear maps. We begin first by fixing the terminology and notation regarding the way Boolean circuits are used in our construction:

1. each Boolean circuit has a number of *circuit input gates*, but at least one. Each input gate has no input wire and exactly one output wire (which is called a *circuit input wire*);
2. each Boolean circuit has exactly one *circuit output gate*, which has one input wire (which is called the *circuit output wire*) and no output wire;
3. each Boolean circuit has a number of *logic gates* of two types: OR-gates and AND-gates. Each of them has exactly two input wires and exactly one output wire;
4. each Boolean circuit may have a number of *FANOUT-gates*. Each FANOUT-gate has exactly one input wire and at least two output wires. Their role is to propagate (multiply) the logic gate outputs;
5. no two FANOUT-gates are directly connected (no output wire of a FANOUT-gate is the input wire of another FANOUT-gate).

The restriction to Boolean circuits that are monotone does not constitute a loss of generality, as it has been shown in [6] (see page 7 in [6]).

Each non-input gate of a Boolean circuit has one or two input wires and, therefore, one or two *input gates*. Figure 1 pictorially represents a Boolean circuit under our conventions ("FO" stands for "FANOUT"). Assuming that the wires are labeled, we may write the gates

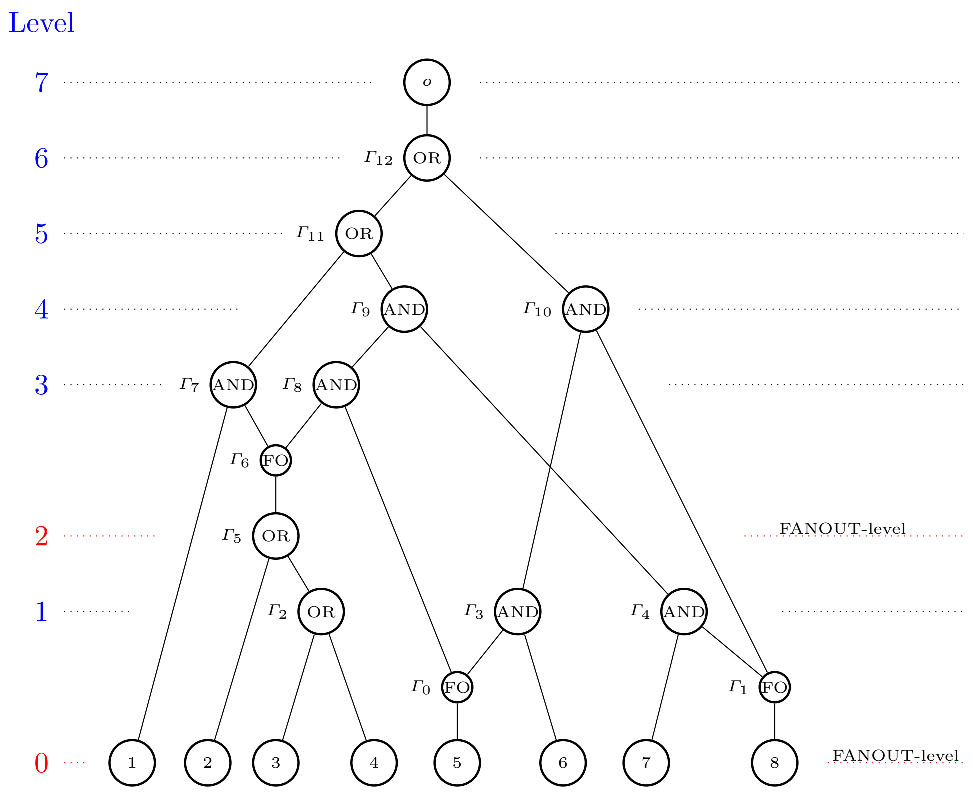

Fig. 1. Boolean circuit with FANOUT-gates

as tuples (w_1, w_2, OR, w), (w_1, w_2, AND, w), and $(w, FANOUT, w_1, \ldots, w_j)$. The elements before (after) the gate name are the input (output) wires of the gate. The output wire of a Boolean circuit will always be denoted by o, and the input wires by $1, \ldots, n$ (assuming that the circuit has n input wires).

All gates of a Boolean circuit are distributed on *levels* which are defined as follows:

1. the 0*th level*, also called the *input level*, consists of all circuit input gates together with all FANOUT-gates directly connected to them;
2. if the $(i-1)$st level has been defined and there are logic gates whose input gates are on the first $(i-1)$ levels but at least one input gate on the $(i-1)$st level, then the ith level consists of all such logic gates together with all FANOUT-gates directly connected to them;
3. if the $(i-1)$st level has been defined and there is no logic gates as above, then the ith level consists only of the output gate (this is also called the *output level* of the circuit).

Figure 1 illustrates the way levels are counted in our Boolean circuits. For instance, the 2nd level consists of Γ_5 and Γ_6.

By $level(\Gamma)$ we denote the level of the gate Γ. The *depth* of a Boolean circuit $\mathcal{C}$, denoted $depth(\mathcal{C})$, is the number of $\mathcal{C}$'s output level. A level is called a *FANOUT-level* if it contains FANOUT-gates. Remark that the input level may be a FANOUT-level, but the output level as well as its direct predecessor cannot be FANOUT-levels (in fact, assuming that no logic gate has a FANOUT-gate as input gate for both its inputs, each FANOUT-level i satisfies $i < depth(\mathcal{C}) - 2$).

Let Γ be a logic gate and Γ' be a gate such that Γ and Γ' are directly connected and $i = Level(\Gamma) > Level(\Gamma') = j$ (that is, Γ' is an input gate of Γ). The *FANOUT-level sequence* from Γ to Γ' is a sequence, possible empty, of FANOUT-level indexes defined as follows:

1. if Γ' is an input or logic gate, then the FANOUT-level sequence from Γ to Γ' is the sequence of all FANOUT-level numbers taken in decreasing order from $i-1$ to j;
2. if Γ' is a FANOUT-gate, then the FANOUT-level sequence from Γ to Γ' is the sequence of all FANOUT-level numbers taken in decreasing order from $i-1$ to $j+1$.

As an example, in the Boolean circuit in Figure 1, (2,0) is the FANOUT-level sequence from Γ_7 to the input gate 1, and (2) is the FANOUT-level sequence from Γ_8 to Γ_0.

To each logic gate Γ, two FANOUT-level sequences are associated: the *left* one, from Γ to its left input gate, and the *right* one, from Γ to its right input gate. It is clear that both of them can be empty and one of them is a prefix of the other one. These two sequences will play an important role in the sharing procedure described below.

Now, we need to fix the terminology on the multilinear maps we use in our construction.

Definition 1. *A* chained multilinear map *is a sequence $(e_i | 1 \leq i \leq k)$ of bilinear maps $e_i : G_i \times G_1 \to G_{i+1}$, $1 \leq i \leq k$, where $G_1, \ldots, G_{k+1}$ are multiplicative groups of the same prime order p.*

Remark 1. Let $(e_i | 1 \leq i \leq k)$ be a chained multilinear map as above. If $g_1 \in G_1$ is a generator of G_1, then g_{i+1} defined recursively by $g_{i+1} = e_i(g_i, g_1)$ is a generator of G_{i+1}, for all $1 \leq i \leq k$ (because e_i is a bilinear map). Therefore, $(e_i | 1 \leq i \leq k)$ can be regarded as a special form of leveled multilinear map. Moreover, finding a chained multilinear map is considerable more easier than finding a leveled multilinear map because no constraints on generators is required.

Chained multilinear maps will be used in our construction as follows. Assume that r is the number of FANOUT-levels in the Boolean circuits we consider, and $(e_i|1 \leq i \leq r+1)$ is a chained multilinear map as above. A message $m \in G_{r+2}$ will be encrypted by mg_{r+2}^{ys}, where y is a random integer chosen in the setup phase and s is a random integer chosen in the encryption phase. To decrypt this message, one needs to compute g_{r+2}^{ys}, and this will be done by using a secret sharing procedure and a secret reconstruction procedure.

The secret sharing procedure, denoted $Share(y, \mathcal{C})$, inputs a Boolean circuit $\mathcal{C}$ and a value $y \in \mathbb{Z}_p$, and outputs three functions S, P, and L with the following meaning:

1. S assigns to each wire of $\mathcal{C}$ an element in $\mathbb{Z}_p$;
2. P assigns to each output wire of a FANOUT-gate an element (also called *FANOUT-key*) in G_1;
3. L assigns to each FANOUT-level an element (also called *FANOUT-level-key*) in G_1.

The sharing procedure is the following one.

<u>$Share(y, \mathcal{C})$</u>

1. Initially, all gates of $\mathcal{C}$ are unmarked;
2. For each FANOUT-level i, $0 \leq i < depth(\mathcal{C}) - 2$, choose uniformly at random $a_i \in \mathbb{Z}_p$ and assign $L(i) := g_1^{a_i}$;
3. $S(o) := y$;
4. If $\varGamma = (w_1, w_2, OR, w)$ is an unmarked OR-gate and $S(w) = x$, then mark $\varGamma$ and assign $S(w_1) := xa_{i_1}^{-1} \cdots a_{i_u}^{-1} \bmod p$ and $S(w_2) := xa_{j_1}^{-1} \cdots a_{j_v}^{-1} \bmod p$, where $i_1 \cdots i_u$ and $j_1 \cdots j_v$ are the left and right FANOUT-level sequences of $\varGamma$, respectively (if the left FANOUT-level sequence is empty, then $S(w_1) := x$, and similarly for the other case);
5. If $\varGamma = (w_1, w_2, AND, w)$ is an unmarked AND-gate and $S(w) = x$, then mark $\varGamma$ and do the followings:
   (a) choose x_1 uniformly at random from $\mathbb{Z}_p$ and compute x_2 such that
   $$x = (x_1 a_{i_1} \cdots a_{i_u} + x_2 a_{j_1} \cdots a_{j_v}) \bmod p,$$
   where $i_1 \cdots i_u$ and $j_1 \cdots j_v$ are the left and right FANOUT-level sequences of $\varGamma$, respectively (if $i_1 \cdots i_u$ is the empty sequence then $a_{i_1} \cdots a_{i_u} = 1$, and similarly for the other case);
   (b) assign $S(w_1) := x_1$ and $S(w_2) := x_2$;
6. If $\varGamma = (w, FANOUT, w_1, \ldots, w_j)$ is an unmarked FANOUT-gate and $S(w_i) = x_i$ for all $1 \leq i \leq j$, then mark $\varGamma$ and do the followings:
   (a) choose uniformly at random $x \in \mathbb{Z}_p$ and compute b_i such that $x_i = xb_i \bmod p$, for all $1 \leq i \leq j$;
   (b) assign $S(w) := x$ and $P(w_i) := g_1^{b_i}$, for all $1 \leq i \leq j$;
7. repeat the last three steps above until all gates get marked.

We will write $(S, P, L) \leftarrow Share(y, \mathcal{C})$ to denote that (S, P, L) is an output of the probabilistic algorithm *Share* on input $(y, \mathcal{C})$. $S(i)$ will be called the *share* of the input wire i associated to the secret y, for all $1 \leq i \leq n$, where n is the number of input wires of $\mathcal{C}$.

The secret reconstruction procedure $Recon(\mathcal{C}, P, L, A, V_A)$ reconstructs a "hidden form" of the secret y starting from "hidden forms" of shares associated to some set A of attributes. This procedure is deterministic and outputs an evaluation function R which assigns to each wire either a value in some group $G_1, \ldots, G_{r+2}$ or the undefined value $\perp$, where r is the number of FANOUT-levels of $\mathcal{C}$. The notation and conventions here are as follows:

- $\mathcal{C}$ is a monotone Boolean circuit;
- $A \subseteq \{1, \ldots, n\}$ is a subset of attributes (input gates/wires of $\mathcal{C}$), where n is the number of input wires of $\mathcal{C}$;
- (S, P, L) is an output of $Share(y, \mathcal{C})$, for some secret y;
- $V_A = (V_A(i) | 1 \leq i \leq n)$, where $V_A(i) = g_2^{\alpha_i}$ for all $i \in A$ and some $\alpha_i \in \mathbb{Z}_p$, and $V_A(i) = \perp$ for all $i \notin A$;
- $\perp$ is an *undefined value* for which the following conventions are adopted: $\perp \notin \cup_{i=1}^{r+2} G_i$, $\perp < x$, $\perp \cdot z = \perp$, $z/\perp = \perp$, and $\perp^z = \perp$, for all $x \in \cup_{i=1}^{r+2} G_i$ and $z \in (\cup_{i=1}^{r+2} G_i) \cup \{\perp\}$, where r is the number of FANOUT-levels of $\mathcal{C}$.

Before describing the secret reconstruction procedure we need one more notation. Given $g_i^\alpha \in G_i$ for some i and α, a FANOUT-level sequence $i_1 \cdots i_u$, and an output L of the *Share* procedure, denote by $Shift(g_i^\alpha, i_1 \cdots i_u, L)$ the element $g_{i+u}^{\alpha a_{i_1} \cdots a_{i_u}} \in G_{i+u}$ obtained as follows:

$$g_{i+u}^{\alpha a_{i_1} \cdots a_{i_u}} := \begin{cases} g_i^\alpha, & \text{if } i_1 \cdots i_u \text{ is empty} \\ e_{i+u-1}(\cdots e_i(g_i^\alpha, L(i_u)) \cdots, L(i_1)), & \text{otherwise} \end{cases}$$

(recall that $i_u < \cdots < i_1$).

Now, the *Recon* procedure is the next one.

<u>$Recon(\mathcal{C}, P, L, A, V_A)$</u>

1. Initially, all gates of $\mathcal{C}$ are unmarked;
2. $R(i) := V_A(i)$, for each input wire i of $\mathcal{C}$;
3. If $\Gamma = (w_1, w_2, OR, w)$ is an unmarked OR-gate and both $R(w_1)$ and $R(w_2)$ were defined, then mark Γ and assign $R(w)$ by
$$R(w) := sup\{Shift(R(w_1), i_1 \cdots i_u, L), Shift(R(w_2), j_1 \cdots j_v, L)\},$$
where $i_1 \cdots i_u$ and $j_1 \cdots j_v$ are the left and right FANOUT-level sequences of Γ, respectively. Remark that either $Shift(R(w_1), i_1 \cdots i_u, L) = \perp$ or $Shift(R(w_2), j_1 \cdots j_v, L) = \perp$ if $Shift(R(w_1), i_1 \cdots i_u, L) \neq Shift(R(w_2), j_1 \cdots j_v, L)$;
4. If $\Gamma = (w_1, w_2, AND, w)$ is an unmarked AND-gate and both $R(w_1)$ and $R(w_2)$ were defined, then mark Γ and assign $R(w)$ by
$$R(w) := Shift(R(w_1), i_1 \cdots i_u, L) \cdot Shift(R(w_2), j_1 \cdots j_v, L),$$
where $i_1 \cdots i_u$ and $j_1 \cdots j_v$ are the left and right FANOUT-level sequences of Γ, respectively. Remark that there exists i such that both $Shift(R(w_1), i_1 \cdots i_u, L)$ and $Shift(R(w_2), j_1 \cdots j_v, L)$ are powers of g_i;
5. If $\Gamma = (w, FANOUT, w_1, \ldots, w_j)$ is an unmarked FANOUT-gate and $R(w)$ was defined, then mark Γ and assign $R(w_i) = e_u(R(w), P(w_i))$ for all $1 \leq i \leq j$, where $R(w)$ is of the form g_u^α for some u and α. Remark that $P(w_i)$ is of the form $g_1^{b_i}$ for all i and some b_i. Therefore, $R(w_i)$ is of the form $g_{u+1}^{\alpha b_i}$, for all i;

6. repeat the last three steps above until all gates get marked.

We are now in a position to define our new KP-ABE scheme, called *KP-ABE_Scheme*.

KP-ABE_Scheme

Setup(λ, n, r)**:** the setup algorithm uses the security parameter λ and the parameter r to choose a prime p, $r+2$ multiplicative groups $G_1, \ldots, G_{r+2}$ of prime order p, a generator $g_1 \in G_1$, and a sequence of bilinear maps $(e_i | 1 \leq i \leq r+1)$, where e_i is from $G_i \times G_1$ into G_{i+1} for all $1 \leq i \leq r+1$. Denote also $g_{i+1} = e_i(g_i, g_1)$, for all $1 \leq i \leq r+1$. Then, it defines the set of attributes $\mathcal{U} = \{1, \ldots, n\}$, chooses $y \in \mathbb{Z}_p$ and, for each attribute $i \in \mathcal{U}$, chooses $t_i \in \mathbb{Z}_p$. Finally, the algorithm outputs the public parameters

$$PP = (n, r, p, G_1, \ldots, G_{r+2}, g_1, e_1, \ldots, e_{r+1}, Y = g_{r+2}^{y}, (T_i = g_1^{t_i} | i \in \mathcal{U}))$$

and the master key $MSK = (y, t_1, \ldots, t_n)$;

Encrypt(m, A, PP)**:** the encryption algorithm encrypts a message $m \in G_{r+2}$ by a non-empty set $A \subseteq \mathcal{U}$ of attributes as follows:

– $s \leftarrow \mathbb{Z}_p$;

– output $E = (A, E' = mY^s, (E_i = T_i^s = g_1^{t_i s} | i \in A))$;

KeyGen$(\mathcal{C}, MSK)$**:** the decryption key generation algorithm generates a decryption key D for the access structure defined by a monotone Boolean circuit $\mathcal{C}$ with n input wires and r FANOUT-levels, as follows:

– $(S, P, L) \leftarrow \mathit{Share}(y, \mathcal{C})$;

– output $D = ((D(i) | i \in \mathcal{U}), P, L)$, where $D(i) = g_1^{S(i)/t_i}$, for all $i \in \mathcal{U}$;

Decrypt(E, D)**:** given E and D as above, the decryption works as follows:

– compute $V_A = (V_A(i) | i \in \mathcal{U})$, where

$$V_A(i) = e_1(E_i, D(i)) = e_1(g_1^{t_i s}, g_1^{S(i)/t_i}) = g_2^{S(i)s}$$

for all $i \in A$, and $V_A(i) = \perp$ for all $i \in \mathcal{U} - A$;

– $R := \mathit{Recon}(\mathcal{C}, P, L, A, V_A)$;

– compute $m := E'/R(o)$.

It is straightforward to prove the correctness of our KP-ABE_Scheme.

Theorem 1. *The KP-ABE_Scheme above satisfies the correctness property. That is, using the notation above, for any encryption $E = (A, mY^s, (E_i | i \in A))$, any circuit $\mathcal{C}$ with n inputs wires and r FANOUT-levels and $\mathcal{C}(A) = 1$, and any $(S, P, L) \leftarrow \mathit{Share}(y, \mathcal{C})$, the valuation R returned by $\mathit{Recon}(\mathcal{C}, P, L, A, V_A)$ satisfies $R(o) = Y^s$.*

Proof. By a simple inspection of the *Share* and *Recon* procedures. □

Our KP-ABE_Scheme can be translated into the graded encoding system formalism exactly as in [6] and, therefore, the details are omitted.

5 Security Issues

To show that our scheme defeats the backtracking attack we have to remark first that the "migration" of a value g_i^α associated to an input wire w_1 of a logic gate Γ_1 to an input wire w_2 of another logic gate Γ_2 is possible only via FANOUT-gates; more precisely, only if w_1 and w_2 are output wires of some FANOUT-gate Γ. If w is the input wire of Γ, the value associated to w cannot be computed from g_i^α (because of the one-wayness property of the chained multilinear map), whereas the value of w_1 can be computed only by using the value associated to w. Therefore, to compute the value for w_2, one has to evaluate bottom-up the circuit and to obtain first the value of w.

The security of KP-ABE_Scheme is proven under the decisional MDH assumption. Remark first that the decisional MDH problem can be formulated for chained multilinear maps as well (with generators defined as in Remark 1).

Theorem 2. *The KP-ABE_Scheme is secure in the selective model under the decisional MDH assumption.*

Proof. Due to the space limitation, the proof can be found in the appendix. □

6 Complexity of the Construction

We will discuss in this section the complexity of our construction KP-ABE_Scheme in terms of size of the decryption key and chained multilinear map, and we will compare it with the complexity of the construction provided in [6].

The approach in [6] associates keys to the input wires of the circuit and to its output gates. Each input wire gets two keys, each OR-gate output wire gets four keys, and each AND-gate output wire gets three keys. The approach does not use explicit FANOUT-gates (but an output wire of some gate may be used as an input wire for more than one gate). Therefore, the total number of keys is bounded from below by $2n + 3q$ and from above by $2n+4q$, where n is the number of inputs and q is the number of gates of the Boolean circuit. If the Boolean circuit $\mathcal{C}$ has the depth ℓ, then the leveled multilinear map used in [6] has $\ell(\ell + 1)/2$ components.

Assuming that the Boolean circuit in our approach has n inputs, r FANOUT-levels, and the total number of outputs of the FANOUT-gates is f, the complexity of our KP-ABE_Scheme is given by

1. $n + r + f$ decryption key components;
2. $r + 1$ bilinear maps.

To compare the two approaches (the one in [6] and ours), we need to examine the complexity of the conversion of Boolean circuits as used in [6] to Boolean circuits as used in our paper. Assume that $\mathcal{C}$ is a Boolean circuit as considered in [6], with n inputs and q logic gates. Let $n = n_1 + n_2$ and $q = q_1 + q_2$, where n_1 (q_1) is the number of input (logic) gates with fan-out one (called *type 1 input* (*logic*) *gates*) and n_2 (q_2) is the number of input (logic) gates with fan-out greater than one (called *type 2 input* (*logic*) *gates*).

Each type 1 input gate "consumes" one input wire and "produces" one output wire, each type 2 input gate "consumes" one input wire and "produces" at least two output wires, each logic gate "consumes" two input wires and "produces" one output wire, and each type 2 logic gate "consumes" two input wires and "produces" at least two output wires. As the Boolean circuit has n input wires and one output wire, it follows that

$$n - n_2 - q_1 - 2q_2 + f = 1,$$

where f is the total number of output wires of type 2 input and logic gates. We can easily transform $\mathcal{C}$ into a Boolean circuit $\mathcal{C}'$ according to our notation by simply adding a FANOUT-gate to each type 2 input and logic gate. This leads to $n_2 + q_2$ FANOUT-gates with a total of f output wires. This FANOUT-gates may be distributed on at least two levels and on at most $1 + q_2$ levels (remark that the FANOUT-gates associated to input gates are all on the 0th level). Therefore, the number of decryption key components (that is, $n + r + f$) is

$$n_2 + q + q_2 + 3 \leq \text{number of key components } \leq n_2 + q + 2q_2 + 2$$

Now, let us estimate the depth ℓ of a Boolean circuit as in [6]. The number of logic gates needed to "consume" n input wires and to generate just one input wire is at least $\log n$ and at most $n - 1$. If the Boolean circuit has n_2 type 2 input gates and q_2 type 2 logic gates (see the notation above), then the wires produced by them is

$$n - n_2 - 2q_2 + f = n_1 - 2q_2 + f = q_1 + 1$$

To consume these wires by type 1 logic gates, at least $\log(q_1 + 1)$ and at most q_1 levels are needed. The q_2 type 2 logic gates can be distributed on at least one level and at most q_2 levels. Therefore, the number ℓ of levels satisfies

$$1 + \log(q_1 + 1) \leq \ell \leq q_1 + q_2 = q$$

Remark that $q_1 \geq n + n_2$.

Our constructions, using the notation above, needs a chained multilinear map with $r+1$ components, where r is the number of FANOUT-levels. According to the estimate above, $1 \leq r \leq q_2$ if $n_2 = 0$, and $2 \leq r \leq q_2 + 1$ if $n_2 \neq 0$. Moreover, $r < \ell - 1$.

Another main difference between our KP-ABE_Scheme and the construction in [6] is that it is much easier to find chained multilinear maps than leveled multilinear maps. Indeed, given k multiplicative groups of the same prime order p, any $k - 1$ bilinear maps $e_i : G_1 \times G_1 \rightarrow G_{i+1}$, $1 \leq i \leq k - 1$, define a chained multilinear map. This is simply seen by taking any arbitrary generator g_1 of G_1 and recursively defining $g_{i+1} = e_i(g_i, g_1)$, for any $1 \leq i \leq k - 2$. On the contrary, not any $k(k-1)/2$ bilinear maps $e_{i,j} : G_i \times G_j \rightarrow G_{i+j}$ define a leveled multilinear map. This is because of the constraint $e_{i,j}(g_i, g_j) = g_{i+j}$, for all $i, j \geq 1$ with $i + j \leq k - 1$.

7 Extensions and Improvements

It is straightforward to see that our scheme can be extended to Boolean circuits with logic gates of fan-in more than two, without increasing the size of the decryption key or of the chained multilinear map. Such an extension could be useful in order to reduce the number of gates and the depth of a given Boolean circuit, resulting in a possible smaller decryption key.

Another simple extension of KP-ABE_Scheme is to consider e_i as bilinear maps from $G_i \times G_i$ into G_{i+1}, for all i. Of course, in such a case, g_{i+1} is defined as being $e_i(g_i, g_i)$, for all i. For this extension to work what we have to do is to replace "$L(i) := g_1^{a_i}$" by "$L(i) := g_i^{a_i}$" in step 2 of *Share*, and "$P(w_i) := g_1^{b_i}$" by "$P(w_i) := g_{level(\Gamma)}^{b_i}$" in step 6 of *Share*. Moreover, "...$P(w_i)$ is of the form $g_1^{b_i}$..." in step 5 of the procedure *Recon* should be replaced by "...$P(w_i)$ is of the form $g_{level(\Gamma)}^{b_i}$...".

Our KP-ABE_Scheme is defined for a fixed number r of FANOUT-levels. However, we can easily extend it to correspond to an arbitrary but upper bounded number of such levels. The main idea is to add FANOUT-level-keys for the "missing FANOUT-levels". More precisely, let r be an upper bound of the number of FANOUT-levels. Define two procedure *Share′* and *Recon′* by modifying *Share* and *Recon* as follows:

1. $Share'(y, \mathcal{C})$ outputs (S, P, L, H) and it is obtained from *Share* by changing the second and third steps into
   "2′. For each FANOUT-level i, $0 \leq i < depth(\mathcal{C}) - 2$, choose uniformly at random $a_i \in \mathbb{Z}_p$ and assign $L(i) := g_1^{a_i}$. For each $1 \leq i \leq h$, where $h = r - r'$ and r' is the number of FANOUT-levels in $\mathcal{C}$, choose uniformy at random $c_i \in \mathbb{Z}_p$ and assign $H(i) := g_1^{c_i}$, $1 \leq i \leq r - r'$;"
   "3′. $S(o) := yc_1^{-1} \cdots c_h^{-1} \bmod p$ if $h > 0$, and $S(o) = y$, otherwise;"
2. $Recon'(\mathcal{C}, P, L, H, A, V_A)$ is obtained from *Recon* by simply adding one more step
   "7. $R(o) := Shift(R(o), h \cdots 1, H)$"

The new scheme KP-ABE_Scheme_1 is the next one:

KP-ABE_Scheme_1

$Setup(\lambda, n, r)$**:** the same as in KP-ABE_Scheme;
$Encrypt(m, A, PP)$**:** the same as in KP-ABE_Scheme;
$KeyGen(\mathcal{C}, MSK)$**:** the decryption key generation algorithm generates a decryption key D for the access structure defined by a monotone Boolean circuit $\mathcal{C}$ with n input wires and $r' \leq r$ FANOUT-levels, as follows:
- $(S, P, L, H) \leftarrow Share'(y, \mathcal{C})$;
- output $D = ((D(i)|i \in \mathcal{U}), P, L, H)$, where $D(i) = g_1^{S(i)/t_i}$, for all $i \in \mathcal{U}$;

$Decrypt(E, D)$**:** given E and D as above, the decryption works as follows:
- compute $V_A = (V_A(i)|i \in \mathcal{U})$, where

$$V_A(i) = e_1(E_i, D(i)) = e_1\left(g_1^{t_i s}, g_1^{S(i)/t_i}\right) = g_2^{S(i)s}$$

  for all $i \in A$, and $V_A(i) = \perp$ for all $i \in \mathcal{U} - A$;
- $R := Recon'(\mathcal{C}, P, L, H, A, V_A)$;
- compute $m := E'/R(o)$.

An important improvement of our scheme consists of using the FANOUT-level-key of a FANOUT-level as a FANOUT-key for the first output wire of each FANOUT-gate on that level. More precisely, define the procedure *Share″* by modifying the sixth step of *Share* into:

$6'$. If $\Gamma = (w, FANOUT, w_1, \ldots, w_j)$ is an unmarked FANOUT-gate and $S(w_i) = x_i$ for all $1 \le i \le j$, then mark Γ and do the followings:
   (a) compute x such that $x_1 = xa_{level(\Gamma)} \bmod p$;
   (b) compute b_i such that $x_i = xb_i \bmod p$, for all $2 \le i \le j$;
   (c) assign $S(w) := x$, $P(w_1) = g_1^{a_{level(\Gamma)}}$, and $P(w_i) := g_1^{b_i}$, for all $2 \le i \le j$;

Now, define the scheme KP-ABE_Scheme_2 as the scheme obtained by replacing *Share* by *Share''* in KP-ABE_Scheme.

The main benefit of this new KP-ABE scheme consists of the fact that the number of decryption key components is decreased by the number of FANOUT-gates. Thus, according to our notation in Section 6, the size of the key provided by KP-ABE_Scheme_2 is

$$n_2 + q + 3 \le \text{ number of key components } \le n_2 + q + q_2 + 2$$

Of course, the extensions and the improvement above can be combined. Their security can be proved as for the KP-ABE_Scheme.

8 Conclusions

We have proposed in this paper a KP-ABE scheme for general monotone Boolean circuits. The scheme is based on secret sharing and a particular and special form of leveled multilinear maps, called *chained multilinear maps.* The efficiency of our scheme (the improved version in Section 7), in comparison with the scheme in [6] which falls in the same class of schemes as ours, is presented in the following table.

Monotone Boolean circuits with – n_1 input gates of fan-out 1 – n_2 input gates of fan-out > 1 – q_1 logic gates of fan-out 1 – q_2 logic gates of fan-out > 1 – r FANOUT-levels – depth ℓ	No of keys	Type and size of multilinear map
KP-ABE scheme in [6]	$2(n_1+n_2)+3(q_1+q_2) \le no.\,keys \le$ $2(n_1+n_2)+4(q_1+q_2)$	• leveled • $\frac{\ell(\ell+1)}{2}$
Our KP-ABE_Scheme	$n_2+q_1+q_2+3 \le no.\,keys \le$ $n_2+q_1+2q_2+2$	• chained • $r+1<\ell$

One can see a great improvement over the decryption key size as well as over the multilinear map size. Apart of this, the use of chained maps instead of leveled maps may constitute a significant benefit.

The improvement in Section 7 is based on the fact that the FANOUT-level-key of some FANOUT-level can be used to reduce a FANOUT-key for each FANOUT-gate on that level. Now, the following question arises: if the maximum fan-out of the FANOUT-gates on some level is α, is there any way to associate α FANOUT-level-keys to that level and remove all FANOUT-keys of all FANOUT-gates on the level? If this can be done, then the size of the decryption key would be cr, where r is the number of FANOUT-levels and c is the maximum fan-out of all FANOUT-gates in the circuit.

References

1. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 784–796, New York, NY, USA, 2012. ACM.
2. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, S&P 2007, pages 321–334. IEEE Computer Society, 2007.
3. Dan Boneh, Valeria Nikolaenko, and Gil Segev. Attribute-based encryption for arithmetic circuits. *IACR Cryptology ePrint Archive*, 2013, 2013.
4. Jean-Sbastien Coron, Tancrde Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493. Springer Berlin Heidelberg, 2013.
5. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013. Preprint on IACR ePrint 2012/610.
6. Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology CRYPTO 2013*, volume 8043 of *Lecture Notes in Computer Science*, pages 479–499. Springer Berlin Heidelberg, 2013. Preprint on IACR ePrint 2013/128.
7. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *STOC*, pages 545–554. ACM, 2013. Preprint on IACR ePrint 2013/337.
8. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006. Preprint on IACR ePrint 2006/309.
9. Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conference on Computer and Communications Security*, pages 195–203. ACM, 2007. Preprint on IACR ePrint 2007/323.
10. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.
11. Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.
12. Douglas R. Stinson. *Cryptography: Theory and Practice.* Chapman and Hall/CRC, 3 edition, 2005.

Appendix

In this appendix to prove the security of our KP-ABE_Scheme.

Theorem 2. *The KP-ABE_Scheme is secure in the selective model under the decisional MDH assumption.*

Proof. It is sufficient to prove that for any adversary $\mathcal{A}$ with an advantage η in the selective game for KP-ABE_Scheme, a PPT algorithm $\mathcal{B}$ can be defined, with the advantage $\eta/2$ over the decisional MDH problem. The algorithm $\mathcal{B}$ plays the role of challenger for $\mathcal{A}$ in the selective game for KP-ABE_Scheme. Taking into account that

1. any leveled multilinear map $\{e_{i,j} | i, j \geq 1,\ i + j \leq k\}$ includes a chained multilinear map $(e_{i,1} | 1 \leq i < k)$;
2. if some PPT algorithm can decide the decisional MDH problem with chained multilinear map instances then it can decide, with at least the same advantage, the decisional MDH problem with leveled multilinear map instances,

we conclude that it is sufficient to give the algorithm $\mathcal{B}$ a chained multilinear map instance of the decisional MDH problem consisting of $r+2$ multiplicative groups $G_1, \ldots, G_{r+2}$ of the same prime order p, $r+2$ generators $g_1, \ldots, g_{r+2}$ of these groups, respectively, $r+1$ bilinear maps $e_i : G_i \times G_1 \to G_{i+1}$ such that $e_i(g_i^a, g_1^b) = g_{i+1}^{ab}$ for all $1 \leq i \leq r+1$ and $a, b \in \mathbb{Z}_p$, and the values g_1^s, $g_1^{c_1}, \ldots, g_1^{c_{r+2}}$, $Z_0 = g_{r+2}^{sc_1 \cdots c_{r+2}}$, and $Z_1 = g_{r+2}^z$, where $s, c_1, \ldots, c_{r+2}, z$ are chosen uniformly at random from $\mathbb{Z}_p$.

Now, the algorithm $\mathcal{B}$ runs $\mathcal{A}$ acting as a challenger for it.

Init Let A be a non-empty set of attributes the adversary $\mathcal{A}$ wishes to be challenged upon.

Setup $\mathcal{B}$ chooses at random $r_i \in \mathbb{Z}_p$ for all $i \in \mathcal{U}$, and computes $Y = g_{r+2}^{c_1 \cdots c_{r+2}}$ and $T_i = g_1^{t_i}$ for all $i \in \mathcal{U}$, where

$$t_i = \begin{cases} r_i, & \text{if } i \in A \\ c_2 r_i, & \text{otherwise} \end{cases}$$

($\mathcal{B}$ can compute Y by using $g_1^{c_1}, \ldots, g_1^{c_{r+2}}$ and $e_1, \ldots, e_{r+1}$, as well as T_i by using r_i and $g_1^{c_2}$). Then, $\mathcal{B}$ publishes the public parameters

$$PP = (n, r, p, G_1, \ldots, G_{r+2}, g_1, e_1, \ldots, e_{r+1}, Y, (T_i | i \in \mathcal{U}))$$

The choice of T_i in this way will be transparent in the next step.

Phase 1 The adversary is granted oracle access to the decryption key generation oracle for all queries $\mathcal{C}$ with n input wires and r FANOUT-levels and $\mathcal{C}(A) = 0$. Given such a query, the decryption key is computed by the following general methodology. First, the algorithm $\mathcal{B}$ uses a procedure $FakeShare$ which shares $g_1^{c_1}$ by taking into account a set A of attributes and using FANOUT-level-keys based on $g_1^{c_3}, \ldots, g_1^{c_{r+2}}$. Then, $\mathcal{B}$ delivers decryption keys based on $g_1^{c_2}$. Two requirements are to be fulfilled:

1. from the adversary's point of view, the secret sharing and distribution of decryption keys should look as in the original scheme;
2. the reconstruction procedure *Recon*, starting from the decryption keys and an authorized set of attributes, should return $g_{r+2}^{c_1 \cdots c_{r+2} s}$.

In order to describe the procedure $FakeShare$ we adopt the following notation: given a wire w of $\mathcal{C}$, denote by $\mathcal{C}_w(A)$ the truth value at w when the circuit $\mathcal{C}$ is evaluated for A. The main idea in $FakeShare$ is the following:

1. if the output wire w of a logic gate $\mathit{\Gamma} = (w_1, w_2, X, w)$ satisfies $C_w(A) = 0$, where X stands for "OR" or "AND", then the value to be shared at this wire is of the form g_1^x, for some $x \in \mathbb{Z}_p$; otherwise, the value to be shared at this wire is an element $x \in \mathbb{Z}_p$;
2. the shares obtained by sharing the value associated to w, and distributed to the input wires of $\mathit{\Gamma}$, should satisfy the same constraints as above. For instance, if $C_{w_1}(A) = 0$ and $C_{w_2}(A) = 1$, then the share distributed to w_1 should be of the form $g_1^{x_1}$ while the share distributed to w_2 should be of the form x_2;
3. the same policy applies to FANOUT-gates as well.

The procedure $FakeShare$ is as follows (for the sake of simplicity we adopt the convention $a_{i_1} \cdots a_{i_u} = 1 = a_{i_1}^{-1} \cdots a_{i_u}^{-1}$ whenever $i_1 \cdots i_u$ is the empty sequence):

$\underline{FakeShare(g_1^{c_1}, g_1^{c_3} \ldots, g_1^{c_{r+2}}, \mathcal{C}, A)}$

1. Initially, all gates of $\mathcal{C}$ are unmarked;
2. Assuming that the FANOUT-levels in $\mathcal{C}$ are $h_1 < \cdots < h_r$, we denote c_j by $c'_{h_{j-2}}$, for all $3 \leq j \leq r+2$. The aim of this notation is just technical, in order to have a correspondence between the c's and the FANOUT-levels (see below).
   Now, for each FANOUT-level i, $0 \leq i < depth(\mathcal{C}) - 2$, choose uniformly at random $a_i \in \mathbb{Z}_p$ and assign $L(i) := g_1^{a_i c'_i}$;
3. $S(o) := g_1^{c_1}$;
4. If $\mathit{\Gamma} = (w_1, w_2, OR, w)$ is an unmarked OR-gate and $S(w)$ was defined, then mark $\mathit{\Gamma}$ and do the followings:
   (a) compute $i_1 \cdots i_u$ and $j_1 \cdots j_v$ the left and right FANOUT-level sequences of $\mathit{\Gamma}$, respectively;
   (b) if $\mathcal{C}_w(A) = \mathcal{C}_{w_1}(A) = \mathcal{C}_{w_2}(A) = 0$, then $S(w_1) := S(w)^{a_{i_1}^{-1} \cdots a_{i_u}^{-1}}$ and $S(w_2) := S(w)^{a_{j_1}^{-1} \cdots a_{j_v}^{-1}}$;
   (c) if $\mathcal{C}_w(A) = \mathcal{C}_{w_1}(A) = \mathcal{C}_{w_2}(A) = 1$, then $S(w_1) := S(w) \cdot a_{i_1}^{-1} \cdots a_{i_u}^{-1}$ and $S(w_2) := S(w) \cdot a_{j_1}^{-1} \cdots a_{j_v}^{-1}$;
   (d) if $\mathcal{C}_w(A) = 1 = \mathcal{C}_{w_1}(A)$ and $\mathcal{C}_{w_2}(A) = 0$, then $S(w_1) := S(w) \cdot a_{i_1}^{-1} \cdots a_{i_u}^{-1}$ and $S(w_2) := g_1^{S(w) \cdot a_{j_1}^{-1} \cdots a_{j_v}^{-1}}$;
   (e) if $\mathcal{C}_w(A) = 1 = \mathcal{C}_{w_2}(A)$ and $\mathcal{C}_{w_1}(A) = 0$, then $S(w_1) := g_1^{S(w) \cdot a_{i_1}^{-1} \cdots a_{i_u}^{-1}}$ and $S(w_2) := S(w) \cdot a_{j_1}^{-1} \cdots a_{j_v}^{-1}$.
   Remark that $S(w) \in \mathbb{Z}_p$ in the cases (c), (d), and (e);
5. If $\mathit{\Gamma} = (w_1, w_2, AND, w)$ is an unmarked AND-gate and $S(w)$ was defined, then mark $\mathit{\Gamma}$ and do the followings:
   (a) compute $i_1 \cdots i_u$ the left FANOUT-level sequence of $\mathit{\Gamma}$ and $j_1 \cdots j_v$ the right FANOUT-level sequence of $\mathit{\Gamma}$;
   (b) choose x_1 uniformly at random from $\mathbb{Z}_p$;
   (c) if $\mathcal{C}_w(A) = 1$, then:
      i. compute x_2 such that

$$S(w) = (x_1 a_{i_1} \cdots a_{i_u} + x_2 a_{j_1} \cdots a_{j_v}) \bmod p;$$

ii. assign $S(w_1) := x_1$ and $S(w_2) := x_2$;

(d) if $\mathcal{C}_w(A) = 0 = \mathcal{C}_{w_2}(A)$ and $\mathcal{C}_{w_1}(A) = 1$ then assign $S(w_1) := x_1$ and

$$S(w_2) = \left(S(w)/g_1^{x_1 a_{i_1} \cdots a_{i_u}}\right)^{a_{j_1}^{-1} \cdots a_{j_v}^{-1}}$$

(e) if $\mathcal{C}_w(A) = 0 = \mathcal{C}_{w_1}(A)$ and $\mathcal{C}_{w_2}(A) = 1$ then do as above by switching w_1 and w_2;

(f) if $\mathcal{C}_w(A) = \mathcal{C}_{w_1}(A) = \mathcal{C}_{w_2}(A) = 0$ then $S(w_1) := g_1^{x_1}$ and $S(w_2)$ is computed as in the case (d);

6. If $\Gamma = (w, FANOUT, w_1, \ldots, w_j)$ is an unmarked FANOUT-gate and $S(w_i)$ was defined for all $1 \leq i \leq j$, then mark Γ and do the followings:
   (a) choose uniformly at random $x \in \mathbb{Z}_p$;
   (b) if $\mathcal{C}_w(A) = \mathcal{C}_{w_1}(A) = \cdots = \mathcal{C}_{w_j}(A) = 1$ then $S(w) := x$ and

   $$P(w_i) := g_1^{c'_{level(\Gamma)} S(w_i) x^{-1}}$$

   for all $1 \leq i \leq j$;
   (c) if $\mathcal{C}_w(A) = \mathcal{C}_{w_1}(A) = \cdots = \mathcal{C}_{w_j}(A) = 0$ then $S(w) := g_1^{c'_{level(\Gamma)} x}$ and $P(w_i) := S(w_i)^{x^{-1}}$, for all $1 \leq i \leq j$;
7. repeat the last three steps above until all gates get marked.

Let $(S, P, L) \leftarrow FakeShare(g_1^{c_1}, g_1^{c_3}, \ldots, g_1^{c_{r+2}}, \mathcal{C}, A)$. The algorithm $\mathcal{B}$ delivers to $\mathcal{A}$ the decryption key $D = ((D(i)|i \in \mathcal{U}), P, L)$, where

$$D(i) = \begin{cases} (g_1^{c_2})^{S(i)/r_i}, & \text{if } i \in A \\ S(i)^{1/r_i}, & \text{if } i \notin A \end{cases}$$

for any $i \in \mathcal{U}$. The key component $D(i)$ is of the form $g_1^{y_i/r_i} = g_1^{c_2 y_i / c_2 r_i}$ for all $i \notin A$ (for some $y_i \in \mathbb{Z}_p$) because the shares of $i \notin A$ are all powers of g_1 (remark that $C_i(A) = 0$).

The distribution of this decryption key is identical to that in the original scheme. Moreover, it is straightforward to see that the reconstruction procedure *Recon*, applied to $V_A(i) = g_2^{S(i) c_2 s}$ for all $i \in A$, where A is an authorized set, returns $g_{r+2}^{c_1 \cdots c_{r+2} s}$. Indeed, in the reconstruction process each FANOUT-level h_j changes the generator (by applying a bilinear map) and multiplies the exponent by c'_{h_j}. As $c_3 \cdots c_{r+2} = c'_{h_1} \cdots c'_{h_r}$, the claim follows.

Challenge The adversary $\mathcal{A}$ selects two messages m_0 and m_1 (of the same length) and sends them to $\mathcal{B}$. The algorithm $\mathcal{B}$ encrypts m_u with Z_v, where $u \leftarrow \{0, 1\}$, and sends it back to the adversary (recall that Z_v was randomly chosen from $\{Z_0, Z_1\}$). The ciphertext is

$$E = (A, E' = m_u Z_v, \{E_i = T_i^s = g_1^{s r_i}\}_{i \in A})$$

If $v = 0$, E is a valid encryption of m_u; if $v = 1$, E' is a random element from G_2.

Phase 2 The adversary may receive again oracle access to the decryption key generation oracle (with the same constraint as in *Phase 1*).

Guess Let u' be the guess of $\mathcal{A}$. If $u' = u$, then $\mathcal{B}$ outputs $v' = 0$; otherwise, it outputs $v' = 1$.

We compute now the advantage of $\mathcal{B}$. Clearly,

$$P(v' = v) - \frac{1}{2} = P(v' = v | v = 0) \cdot P(v = 0) + P(v' = v | v = 1) \cdot P(v = 1) - \frac{1}{2}$$

Both $P(v = 0)$ and $P(v = 1)$ are 1/2. Then, remark that

$$P(v' = v | v = 0) = P(u' = u | v = 0) = \frac{1}{2} + \eta$$

and $P(v' = v | v = 1) = P(u' \neq u | v = 1) = \frac{1}{2}$. Putting all together we obtain that the advantage of $\mathcal{B}$ is $P(v' = v) - \frac{1}{2} = \frac{1}{2}\eta$. $\square$