BalkanCryptSec 2014, Istanbul, Turkey, October 16-17, 2014, pp 175-193, LNCS 9024

Key-policy Attribute-based Encryption for Boolean Circuits from Bilinear Maps

Ferucio Laurenţiu Ţiplea[1] and Constantin Cătălin Drăgan[2]

[1] Department of Computer Science, "Al.I.Cuza" University of Iaşi
700506 Iaşi, Romania, e-mail: fltiplea@info.uaic.ro
[2] CNRS, LORIA, 54506 Vandoeuvre-lès-Nancy Cedex France
e-mail: catalin.dragan@loria.fr

Abstract. We propose a Key-policy Attribute-based Encryption (KP-ABE) scheme for (monotone) Boolean circuits based on bilinear maps. The construction is based on secret sharing and just one bilinear map, and it is a proper extension of the KP-ABE scheme in [7] in the sense that it is practically efficient for a class of Boolean circuits which strictly includes all Boolean formulas. Selective security of the proposed scheme in the standard model is proved, and comparisons with the scheme in [5] based on leveled multilinear maps, are provided. Thus, for Boolean circuits representing multilevel access structures, our KP-ABE scheme is more efficient than the one in [5].

1 Introduction

Attribute-based encryption (ABE) is a new paradigm in cryptography, where messages are encrypted and decryption keys are computed in accordance with a given set of attributes and an access structure on the set of attributes. There are two forms of ABE: *key-policy ABE* (KP-ABE) [7] and *ciphertext-policy ABE* (CP-ABE) [2]. In a KP-ABE, each message is encrypted together with a set of attributes and the decryption key is computed for the entire access structure; in a CP-ABE, each message is encrypted together with an access structure while the decryption keys are given for specific sets of attributes.

In this paper we focus only on KP-ABE. The first KP-ABE scheme was proposed in [7], where the access structures were specified by monotone Boolean formulas (monotone Boolean circuits of fan-out one, with one output wire). An extension to the non-monotonic case has later appeared in [9]. Both approaches [7] and [9] take into consideration only access structures defined by Boolean formulas. However, there are access structures of practical importance that cannot be represented by Boolean formulas, such as multilevel access structures [12, 13]. In such a case, defining KP-ABE schemes for access structures defined by Boolean circuits becomes a necessity. The first solution to this problem was proposed in [5] by using leveled multilinear maps (sets of bilinear maps with some special property). A little later, a lattice-based construction was also proposed [6].

Contribution The KP-ABE schemes for Boolean circuits proposed so far are either based on leveled multilinear maps or on lattices. Direct extensions of the scheme in [7] to Boolean circuits face the *backtracking attack* [5]. Moreover, it was conjectured in [5] that such extensions cannot be realized using bilinear maps.

In this paper we show that an extension of the KP-ABE scheme in [7] to accommodate the case of (monotone) Boolean circuits is possible. The scheme we propose is practically efficient for a subclass of Boolean circuits which strictly extends the class of Boolean formulas (and, therefore, it can be considered as a proper extension of the scheme in [7]). In order to reach this objective, the Boolean circuits are endowed with explicit *fanout-gates* (FO-gates). This is not really necessary, but it is quite helpful in describing the secret sharing procedure used by the scheme. The secret sharing procedure works top-down as in [7]. The outputs of FO-gates are encrypted and the encryption keys are "transmitted" to their input wires in order to be further processed by the sharing procedure. This prevents the backtracking attack because it is not possible to compute the value at an output wire (of a FO-gate) by knowing the value at the other output wire, without computing bottom-up the value at the input wire.

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

We then discuss the complexity of our scheme and compare it with the scheme in [5]. Thus, if the FO-gates are not path-connected in the Boolean circuit, our scheme may perform better than the one in [5]. We prove this by considering Boolean circuits representing conjunctive and disjunctive multilevel access structures, and we show that our scheme distributes shorter decryption keys than the one in [5]. Whatever the considered Boolean circuit, our KP-ABE scheme has the advantage of using just one bilinear map while the scheme in [5] uses leveled multilinear maps whose size quadratically depends on the Boolean circuit depth.

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 [7], illustrates the backtracking attack, discussed the solution in [5] which thwarts the backtracking attack, and gives an informal overview of our solution. It also fixes the terminology on the Boolean circuits we use. 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 [5]. The seventh section shows that our scheme performs better than the one in [5] for Boolean circuits representing multilevel access structures. We conclude in the last section.

2 Preliminaries

Access structures Recall first that [11], 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 contains all subsets $B \subseteq \mathcal{U}$ with $A \subseteq B$ for some $A \in \mathcal{S}$. 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-gates have 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. Boolean circuits of fan-out one 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; then the Boolean values are propagated bottom-up to all gate output wires in a standard way. $\mathcal{C}(A)$ stands for the Boolean value obtained by evaluating $\mathcal{C}$ for A. The access structure defined by $\mathcal{C}$ is the set of all A with $\mathcal{C}(A) = 1$.

Figure 1a pictorially represents a Boolean circuit $\mathcal{C}$ over $\mathcal{U} = \{1, 2, 3, 4\}$. For $A = \{1, 2\}$ we have $\mathcal{C}(A) = 1$ and for $B = \{2, 4\}$ we have $\mathcal{C}(B) = 0$.

Attribute-based encryption A KP-ABE scheme consists of four probabilistic polynomial-time (PPT) algorithms [7]:

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 a 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 [7]. 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 $P(b' = b) - 1/2$. The KP-ABE scheme is *secure* (in the selective model) if any adversary has only a negligible advantage in the selective game described above.

Bilinear maps and the decisional BDH assumption Given G_1 and G_2 two multiplicative cyclic groups of prime order p, a map $e : G_1 \times G_1 \to G_2$ is called *bilinear* if it satisfies:

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

G_1 is called a *bilinear group* if the operation in G_1 and e are both efficiently computable.

The *Decisional Bilinear Diffie-Hellman* (DBDH) problem in the bilinear group G_2 is the problem to distinguish between $e(g, g)^{abc}$ and $e(g, g)^z$ given g, g^a, g^b, and g^c, where g is a generator of G_1 and a, b, c, and z are randomly chosen from $\mathbb{Z}_p$. The *DBDH assumption* for G_2 states that no PPT algorithm $\mathcal{A}$ can solve the DBDH problem in G_2 with more than a negligible advantage.

3 The Backtracking Attack

The closest approaches to, and the starting point of, our paper are [7, 5]. [7] introduces the first KP-ABE scheme. The main idea here is quite elegant and simple, and can be summarized as follows:

- let $e : G_1 \times G_1 \to G_2$ be a bilinear map and g a generator of G_1, where G_1 and G_2 are of prime order p;
- to encrypt a message $m \in G_2$ 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 decryption key is generated as follows. The integer y is shared to all attributes so that it can be recovered only by authorized sets of attributes (the authorized sets are defined by monotone Boolean formulas). The sharing procedure is based on linear secret sharing schemes because linear combinations can be efficiently obtained as exponents of the bilinear map e. 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 [5] that the construction in [7] cannot be used to design a KP-ABE scheme for Boolean circuits. The reason is that, 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. 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). 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. In order to illustrate this attack, called the *backtracking attack* in [5], we consider the monotone Boolean circuit in Figure 1a (remark that it has a fan-out of two). As we can easily see, the minimal authorized sets are $\{1, 2\}$, $\{1, 3\}$, and

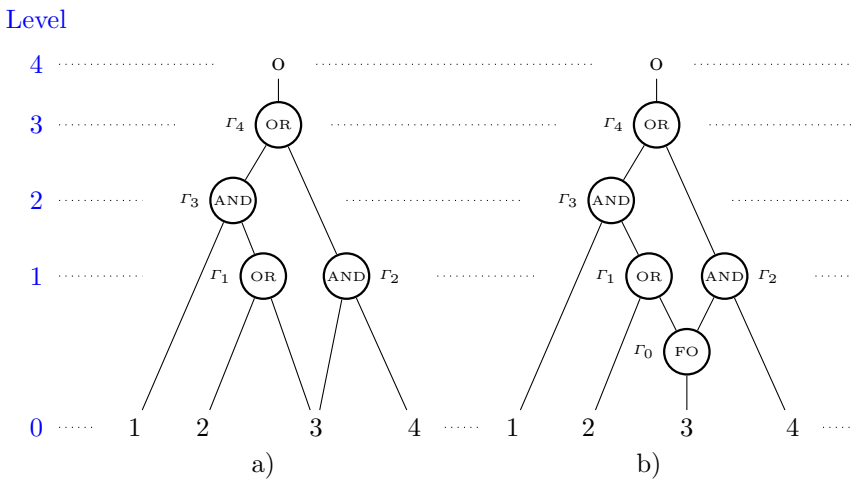


Fig. 1. a) The backtracking attack; b) Boolean circuit with FO-gates

$\{3, 4\}$. Consider now the following scenario. Assume that a given message is encrypted by the authorized set of attributes $\{2, 3, 4\}$ and a user with the set of attributes $\{1, 2, 4\}$ asks for a decryption key. His set of attributes is authorized and, therefore, he has the right to obtain a decryption key. According to the definition of a KP-ABE, the decryption key for the set $\{1, 2, 4\}$ of attributes must not be valid to decrypt a message encrypted by the attributes $\{2, 3, 4\}$. However, the user can do as follows. The value computed at the input wire 2 "migrates" to the input wire 3 due to the existence of the OR-gate Γ_1. Corroborating this with the values at the input wires 1 and 4, a valid value will be computed at the output wire. This value is the same as the value computed by the set

$\{2,3,4\}$ and, therefore, it allows the decryption of the message. Remark also that $\{2,4\} = \{1,2,4\} \cap \{2,3,4\}$ is unauthorized.

The backtracking attack illustrated above cannot occur in case of access structures defined by Boolean formulas as in [7] because, in such a case, the input wires of OR-gates are not used by any other gates (the circuit is of fan-out one).

To avoid the backtracking attack, [5] uses a "one-way" construction in evaluating monotone Boolean circuits. The idea is the next one:

- consider a *leveled multilinear map*, consisting of k groups $G_1, \ldots, G_k$ of prime order p, k generators $g_1, \ldots, g_k$ of these groups, respectively, and a set $\{e_{i,j} : G_i \times G_j \to G_{i+j} | i, j \geq 1,\ i+j \leq k\}$ of bilinear maps satisfying $e_{i,j}(g_i^a, g_j^b) = g_{i+j}^{ab}$, for all i and j and all $a, b \in \mathbb{Z}_p^*$, where k is the circuit depth plus one;
- the key components are associated to the circuit input wires and to each gate output wire (in [5], each gate has one output wire which may 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.

As with respect to the existence of leveled multilinear, [5] shows how this scheme can be translated into the GGH graded algebra framework [4].

Looking more carefully at the example in Figure 1a, we remark that the value obtained at the wire 3 via the input wire 1 and the OR-gate Γ_1 is then used at the AND-gate Γ_2. The backtracking attack illustrated above would be thwarted if the two outputs from 3 (one leading to Γ_1 and one leading to Γ_2) were different. This is in fact the starting point of our proposal. That is, we use explicit *fanout-gates* (FO-gates) with encrypted outputs to multiply input wires and gate output wires. Therefore, the Boolean circuits we use in the rest of the paper have FO-gates too. A FO-gate has one input wire and at least two output wires, and its role is to propagate its input to all outputs. In this way, the fan-out of all logic gates will be restricted to one. Moreover, as FO-gates may have arbitrary fanout, we assume that no two FO-gates are directly connected. Figure 1b pictorially represents the Boolean circuit in Figure 1a using FO-gates.

We close this section by informally describing our solution and why it thwarts the backtracking attack (details will be given in the next sections):

- the information at the output wires of OR-gates are simple passed to the input wires, while the information at the output wires of AND-gates are shared as in the Karnin-Greene-Hellman scheme [8];
- the FO-gates, which are not present in [7], are processed by associating random keys to their input wires in order to deal with the output wires. In this way, the value computed at one of the output wires cannot be used to derive values at the other output wires. For instance, the value computed at

the input wire 2 in Figure 1b can "migrate" to the left output wire of the FO-gate Γ_0, but cannot be used as an input value for the AND-gate Γ_2.

4 Our Construction

In this section we propose a KP-ABE scheme for monotone Boolean circuits based on bilinear maps. The restriction to Boolean circuits that are monotone does not constitute a loss of generality (see page 7 in [5]). However, recall from the previous section that our Boolean circuits have FO-gates and all the logic gates have fan-out one. Assuming that the wires are labeled, we may write the gates as tuples (w_1, w_2, OR, w), (w_1, w_2, AND, w), and $(w, FO, w_1, \ldots, w_j)$, where $j \geq 2$. 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).

Before describing our KP-ABE scheme assume that two multiplicative cyclic groups G_1 and G_2 of prime order p are given, together with a generator g of G_1 and a bilinear map $e : G_1 \times G_1 \to G_2$. As our KP-ABE scheme is based on secret sharing, we will define two procedures, one for secret sharing and the other one for secret reconstruction.

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

1. S assigns to each wire of $\mathcal{C}$ a list of values in $\mathbb{Z}_p$;
2. P assigns to each output wire of a FO-gate a list of pairs in $G_1 \times G_1$.

By a *list of length* n of elements over a set X we understand any vector $L \in X^n$. $|L|$ stands for the length of L, $L_1 L_2$ for the *concatenation* of two lists L_1 and L_2, and $pos(L) = \{1, \ldots, |L|\}$ for the *set of positions* in the list L. $L(i)$ denotes the ith element of L. If L is a list of lists, then $L(i, j)$ denotes the jth element of the list $L(i)$.

Now, the sharing procedure is the following one.

$\underline{Share(y, \mathcal{C})}$

1. Initially, all gates of $\mathcal{C}$ are unmarked;
2. $S(o) := (y)$;
3. If $\Gamma = (w_1, w_2, OR, w)$ is an unmarked OR-gate and $S(w) = L$, then mark Γ and assign $S(w_1) := L$ and $S(w_2) := L$;
4. If $\Gamma = (w_1, w_2, AND, w)$ is an unmarked AND-gate and $S(w) = L$, then mark Γ and do the followings:
   (a) for each $i \in pos(L)$ choose uniformly at random $x_i^1 \in \mathbb{Z}_p$ and compute x_i^2 such that $L(i) = (x_i^1 + x_i^2) \bmod p$;
   (b) compute $L_1 = (x_i^1 | 1 \leq i \leq |L|)$ and $L_2 = (x_i^2 | 1 \leq i \leq |L|)$;
   (c) assign $S(w_1) := L_1$ and $S(w_2) := L_2$;
5. If $\Gamma = (w, FO, w_1, \ldots, w_j)$ is an unmarked FO-gate and $S(w_k) = L_k$ for all $1 \leq k \leq j$, then mark Γ and, for each $1 \leq k \leq j$, do the followings:

(a) for each $i \in pos(L_k)$ choose uniformly at random $a_i \in \mathbb{Z}_p$ and compute b_i such that $L_k(i) = (a_i + b_i) \bmod p$;
(b) compute $L'_k = (a_i | 1 \leq i \leq |L_k|)$ and $P(w_k) := (g^{b_i} | 1 \leq i \leq |L_k|)$;
(c) Assign $S(w) := L'_1 \cdots L'_j$;

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

We will write $(S, P) \leftarrow Share(y, \mathcal{C})$ to denote that (S, P) is an output of the probabilistic algorithm $Share$ on input $(y, \mathcal{C})$. $S(i)$ will be called the *list of shares* of the input wire i associated to the secret y $(1 \leq i \leq n)$. Figure 2a generically illustrates the procedure $Share$.

We define now a reconstruction procedure $Recon(\mathcal{C}, P, V, g^s)$ which reconstructs a "hidden form" of the secret y 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 a list of values in $G_2 \cup \{\perp\}$. The notation and conventions here are as follows:

- $\mathcal{C}$ is a monotone Boolean circuit with n input wires;
- (S, P) is an output of $Share(y, \mathcal{C})$, for some secret y;
- $s \in \mathbb{Z}_p$;
- $V = (V(i) | 1 \leq i \leq n)$, where $V(i)$ is either a list $(e(g, g)^{\alpha_i} | 1 \leq j \leq |S(i)|)$ for some $\alpha_i \in \mathbb{Z}_p$, or a list of $|S(i)|$ undefined values $\perp$, for all $1 \leq i \leq n$;
- $\perp$ is an *undefined value*, not in G_2, for which the following conventions are adopted:
  - $\perp < x$, for all $x \in G_2$;
  - $\perp \cdot z = \perp$, $z/\perp = \perp$, and $\perp^z = \perp$, for all $z \in G_2 \cup \{\perp\}$.

The reconstruction procedure can now be described as follows.

$\underline{Recon(\mathcal{C}, P, V, g^s)}$

1. Initially, all gates of $\mathcal{C}$ are unmarked;
2. $R(i) := V(i)$, for all $i \in \mathcal{U}$;
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

$$R(w, i) := sup\{R(w_1, i), R(w_2, i)\},$$

for all $1 \leq i \leq |R(w_1)|$ (remark that $|R(w_1)| = |R(w_2)|$ and, for any i, if $R(w_1, i) \neq R(w_2, i)$ then either $R(w_1, i) = \perp$ or $R(w_2, i) = \perp$);
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 $R(w, i) := R(w_1, i) \cdot R(w_2, i)$, for all $1 \leq i \leq |R(w_1)|$ (remark that $|R(w_1)| = |R(w_2)|$);
5. If $\Gamma = (w, FO, w_1, \ldots, w_j)$ is an unmarked FO-gate and $R(w)$ was defined, then mark Γ and do the followings:
   (a) split $R(w)$ into j lists $R(w) = R_1 \cdots R_j$ with $|R_k| = |P(w_k)|$ for all $1 \leq k \leq j$ (see the sharing algorithm for correctness);
   (b) $R(w_k, i) := R_k(i) \cdot e(P(w_k, i), g^s)$, for all $1 \leq k \leq j$ and for all $1 \leq i \leq |R_k|$;

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

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

KP-ABE_Scheme

Setup(λ, n)**:** the setup algorithm uses the security parameter λ to choose a prime p, two multiplicative groups G_1 and G_2 of prime order p, a generator g of G_1, and a bilinear map $e : G_1 \times G_1 \to G_2$. 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

$$PP = (p, G_1, G_2, g, e, n, Y = e(g, g)^y, (T_i = g^{t_i} | i \in \mathcal{U}))$$

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

Encrypt(m, A, PP)**:** the encryption algorithm encrypts a message $m \in G_2$ by a non-empty set $A \subseteq \mathcal{U}$ of attributes as follows:

- $s \leftarrow \mathbb{Z}_p$;
- output $E = (A, E' = mY^s, (E_i = T_i^s = g^{t_i s} | i \in A), g^s)$;

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 as follows:

- $(S, P) \leftarrow Share(y, \mathcal{C})$;
- output $D = ((D(i) | i \in \mathcal{U}), P)$, where $D(i) = (g^{S(i,j)/t_i} | 1 \leq j \leq |S(i)|)$, 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

$$V_A(i, j) = e(E_i, D(i, j)) = e(g^{t_i s}, g^{S(i,j)/t_i}) = e(g, g)^{S(i,j)s}$$

  for all $i \in A$ and $1 \leq j \leq |S(i)|$, and $V_A(i)$ is a list of $|S(i)|$ symbols $\perp$, for all $i \in \mathcal{U} - A$;
- $R := Recon(\mathcal{C}, P, V_A, g^s)$;
- $m := E'/R(o, 1)$ (recall that o is the output wire of $\mathcal{C}$).

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

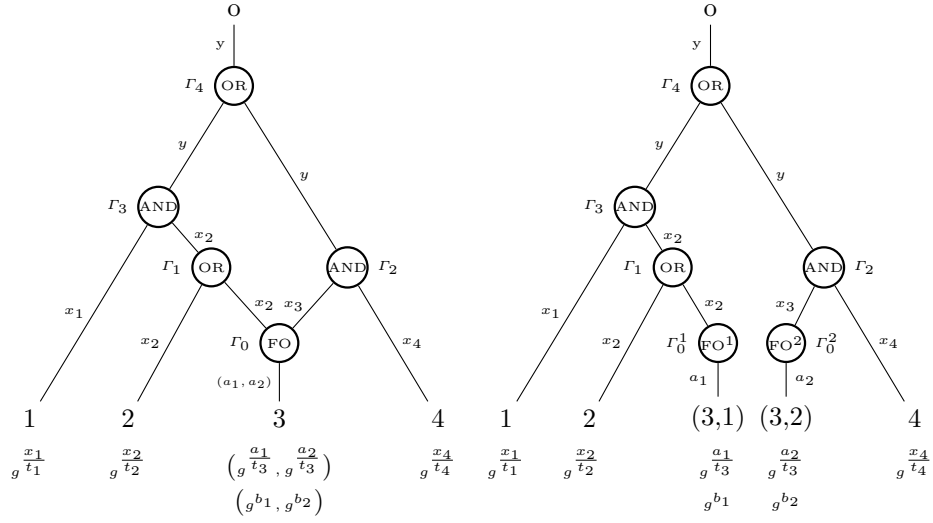
Proof. By a simple inspection of the *Share* and *Recon* procedures. □

5 Security Issues

We begin by showing that our scheme is resistant to the backtracking attack. This will be achieved by associating an access tree $\mathcal{T}_\mathcal{C}$ to a Boolean circuit $\mathcal{C}$ and showing that the backtracking attack succeeds on $\mathcal{C}$ if and only if it succeeds on

$\mathcal{T}_{\mathcal{C}}$. Then, our claim will follow because the backtracking attack does not work in access trees.

The construction of $\mathcal{T}_{\mathcal{C}}$ consists of multiplying each wire of $\mathcal{C}$ as many times as paths are from the wire to the output wire (the number of such paths gives the numbers of shares the wire receives in the sharing process performed by the procedure *Share*). Then, each wire is labeled in order to distinguish the path associated to it. This is done by splitting up each FO-gate Γ with j ouput wires into j pseudo-gates $\Gamma^1, \ldots, \Gamma^j$, each with one input and one output wire (the kth output wire of Γ is the output wire of Γ^k, for all $1 \leq k \leq j$). If we collect all these exponents on the path from an wire to the output wire, we obtain a label which uniquely identifies the wire. The wires with an empty label have exactly one path from them to the output wire. For instance, the two wires 3 in Figure 2b have, one of them the label 1 and the other one the label 2. The wires 1, 2, and 4 in the same figure have all an empty label.



a) A Boolean circuit $\mathcal{C}$ b) The access tree $\mathcal{T}_{\mathcal{C}}$

Fig. 2. The construction of $\mathcal{T}_{\mathcal{C}}$

If a wire w of $\mathcal{C}$ receives a list $S(w)$ of shares by the *Share* procedure applied to $\mathcal{C}$, then it is straightforward to see that each share $S(w, j)$ uniquely corresponds to some label $u_{w,j}$ as defined above. In the tree $\mathcal{T}_{\mathcal{C}}$, the wire $(w, u_{w,j})$ will receive the share $S(w, j)$.

This technique "unfolds" $\mathcal{C}$ with its lists of shares into a tree $\mathcal{T}_{\mathcal{C}}$ which has exactly one share for each wire. Now, it is straightforward to see that the backtracking attack succeeds on $\mathcal{C}$ if and only if it succeeds on $\mathcal{T}_{\mathcal{C}}$. As the backtracking attack does not work in access trees, our claim above follows.

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

Proof. In Appendix 1. □

6 Complexity of the Construction

We will discuss in this section the complexity of our construction (KP-ABE_Scheme) and we will compare it with the complexity of the construction provided in [5].

As our scheme uses just one bilinear map, the only question we have to answer with respect to the complexity of our construction is about the size of the decryption key. Assume that the Boolean circuit has n input wires and r FO-gates of fanout at most j. Two cases are to be considered:

Case 1: there is no path between any two FO-gates. In this case, by the sharing procedure, exactly r input wires will receive at most j shares (but at least two), and the other input wires will receive exactly one share. This leads to at most $n + r(j-1)$ key components, and this is the minimum size of the decryption key (remark also that $r \leq n$ in this case);

Case 2: there are paths between FO-gates. In this case, the FO-gates on the highest level in the circuit may transmit the j shares collected at their input wires to FO-gates on the previous level. The sharing procedure will associate now at most j^2 shares to the input wires of these gates. This reasoning shows that the maximum number of shares some input wires of the circuit may receive is at most j^α, where α is the number of levels that contain FO-gates (α is less than or equal to miminum of r and the circuit depth).

The approach in [5] 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 FO-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 q is the number of gates. The Boolean circuits in [5] can be transformed into our formalism if we replace each wire which is used by $j \geq 2$ gates by a FO-gate with j outputs. In this way, one can easily remark that q depends on both the number r of FO-gates and on the maximum number j of outputs of these FO-gates.

Another main difference between our scheme and the one in [5] consists of the number of bilinear maps used by them. While ours uses just one bilinear map, the one in [5] uses a leveled multilinear map with $\ell(\ell+1)/2$ bilinear map components interrelated with each other (see Section 3), where ℓ is the circuit depth.

7 Extensions and Applications

The above section shows that our KP-ABE_Scheme may be more efficient than the one in [5] when the Boolean circuits do not have FO-gates connected between them by paths. We will show in this section that the Boolean circuits

representing multilevel access structures [10, 12] fulfill this property. For a compact representation of multilevel access structures we need Boolean circuits with *threshold gates* (as in [7]). An (a, b)-*threshold gate*, where a and b are integers satisfying $1 \leq a \leq b$ and $b \geq 2$, is a logic gate with b input wires and one output wire. The output wire of such a gate is evaluated to the truth value 1 whenever at least a input wires of the gate are assigned to the truth value 1. OR-gates are $(1, 2)$-threshold gates, while AND-gates are $(2, 2)$-threshold gates.

A *disjunctive multilevel access structure* [10] over a set $\mathcal{U}$ of attributes is a tuple $(\overline{a}, \overline{\mathcal{U}}, \mathcal{S})$, where $\overline{a} = (a_1, \ldots, a_k)$ is a vector of positive integers satisfying $0 < a_1 < \cdots < a_k$, $\overline{\mathcal{U}} = (\mathcal{U}_1, \ldots, \mathcal{U}_k)$ is a partition of $\mathcal{U}$ (that is, all $\mathcal{U}_i$ are non-empty and their union is $\mathcal{U}$), and $\mathcal{S}$ is defined by:

$$\mathcal{S} = \{A \subseteq \mathcal{U} | (\exists 1 \leq i \leq k)(|A \cap (\cup_{j=1}^{i} \mathcal{U}_j)| \geq a_i)\}.$$

If we replace "$\exists$" by "$\forall$" in the above definition, we obtain the concept of *conjunctive multilevel access structure* [12]. It is well-known, and not difficult to prove (see Appendix 2), that disjunctive and conjunctive multilevel access structures cannot be represented by Boolean formulas. Using Boolean circuits, these access structures can be easily represented as in Figure 3. Moreover, the FO-gates are in between the first two levels.

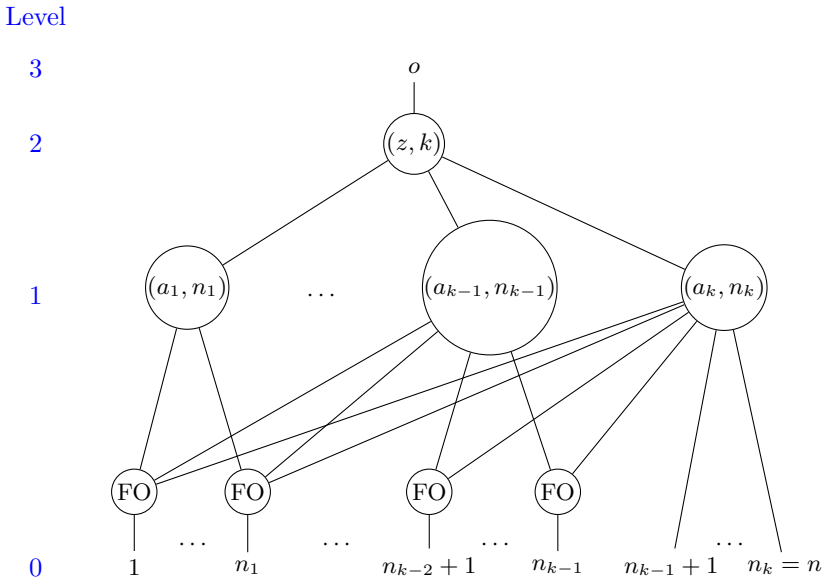


Fig. 3. Boolean circuit representation of multilevel access structure: z is 1 for the disjunctive case, and k for the conjunctive case.

Our KP-ABE_Scheme can be easily adapted to accommodate threshold gates. Assume that LSSS is a probabilistic linear secret sharing scheme [7] such that,

given a and b as above, and given a master secret $x \in \mathbb{Z}_p$, the scheme outputs b shares $x_1, \ldots, x_b$ such that x can be uniquely reconstructed from any a shares. Moreover, assume that there exists an associated and efficient deterministic procedure $LSSS^{-1}$ such that

$$LSSS^{-1}(e(g,g)^{x_{i_1}s}, \ldots, e(g,g)^{x_{i_a}s}) = e(g,g)^{xs},$$

for any a shares $x_{i_1}, \ldots, x_{i_a}$ and any $s \in \mathbb{Z}_p$.

Shamir's threshold secret sharing scheme satisfies this property and it was used in [7] exactly with this purpose.

Define now a procedure $Share'$ obtained from $Share$ by replacing the steps 3 and 4 by just one step:

3'. If $\Gamma = (w_1, \ldots, w_b, (a,b), w)$ is an unmarked (a,b)-threshold gate and $S(w) = L$, then mark Γ and do the followings:
   (a) for each $i \in pos(L)$, run $LSSS$ and obtain the shares $x_i^1, \ldots, x_i^b$;
   (b) for each $1 \leq j \leq b$ compute the list L_j from L by replacing $L(i)$ by x_i^j;
   (c) assign $S(w_j) := L_j$, for all $1 \leq j \leq b$;

The corresponding reconstruction procedure $Recon'$ is obtained by replacing the steps 3 and 4 in $Recon$ by just one step:

3'. If $\Gamma = (w_1, \ldots, w_b, (a,b), w)$ is an unmarked (a,b)-threshold gate and $R(w_j)$ was defined for all $1 \leq j \leq b$, then mark Γ and assign $R(w)$ by

$$R(w,i) := sup\{LSSS^{-1}(R(w_{i_1}, i), \ldots, R(w_{i_a}, i)) | i_1, \ldots, i_a \in \{1, \ldots, b\}\},$$

for all $1 \leq i \leq |R(w_1)|$;

The new ABE scheme, denoted KP-ABE_Scheme', is obtained from KP-ABE_Scheme by replacing $Share$ and $Recon$ by $Share'$ and $Recon'$, respectively. Its security can be proved as for the KP-ABE_Scheme.

The number of key components distributed by our KP-ABE_Scheme' when applied to a multilevel access structure as in Figure 3 is

$$n_1 \cdot k + (n_2 - n_1) \cdot (k-1) + \cdots + (n_k - n_{k-1}) \cdot 1$$

If we approximate n_1 and $n_i - n_{i-1}$ by the average value n/k, for all $2 \leq i \leq k$, the average number of the decryption key components is $n(k+1)/2$.

The KP-ABE scheme in [5] can be easily adapted to accommodate $(1,b)$- and (b,b)-threshold gates. In the first case $2b$ key components are associated to the gate, while in the second case $b+1$ key components are associated to the gate. However, there is no direct way to accommodate (a,b)-threshold gates when $1 < a < b$. The indirect way is to consider $C(b,a)$ threshold gates of type (a,a) and one threshold gate of type $(1, C(b,a))$ ($C(b,a)$ stands for the number of combinations of b taken a). Therefore, the number of decryption key components in case of a multilevel access structure can be approximated as follows:

Case 1: $a_i = n_i$, for all i. In this case, the number of key components is

$$2n + \sum_{i=1}^{k}(C(n_i, n_i) + 1) + (2k + 1 - z) = 2n + \sum_{i=1}^{k}(n_i + 1) + (2k + 1 - z)$$

If we write each n_i in the form $n_i = n_1 + (n_2 - n_1) + \cdots + (n_i - n_{i-1})$ and approximate n_1 and $n_j - n_{j-1}$ by the average value n/k, for all $2 \leq j \leq i$, we obtain the average number of the decryption key components as being $n(k+5)/2 + (3k+1-z)$, where $z = 1$ for the disjunctive case and $z = k$ for conjunctive case;

Case 2: $a_i < n_i$, for all i. In this case, $a_i \geq i$ for all i. Using the inequality $C(n_i, a_i) \geq n_i$, we can bound from below the number of key components by

$$2n + \sum_{i=1}^{k}(a_i + 1)C(n_i, a_i) + \sum_{i=1}^{k} 2C(n_i, a_i) + (2k + 1 - z)$$

$$\geq 2n + \sum_{i=1}^{k}(i + 3)n_i + (2k + 1 - z).$$

If we apply the same reasoning as in the previous case to the right hand side of this inequality, we obtain the average estimate $(2 + (k+1)(k+5)/3)n + (2k + 1 - z)$ (z is as above).

Moreover, the leveled multilinear map has six bilinear map components, but only three of them are used (the approach in [5] counts three levels).

Our discussion so far, summarized in Table 1 below, shows clearly that our approach is more efficient than the one in [5] for multilevel access structures.

Scheme	Average no. of keys	No. of bilinear maps
KP-ABE scheme in [5]	Case 1: $n\frac{k+5}{2} + 3k + 1 - z$ Case 2: $\geq \left(2 + \frac{(k+1)(k+5)}{3}\right) n + 2k + 1 - z$	3
Our KP-ABE_Scheme′	$n\frac{k+1}{2}$	1

Table 1. Comparisons between the scheme in [5] and our scheme for multilevel access structures

8 Conclusions

We have proposed in this paper a KP-ABE scheme for monotone Boolean circuits. The scheme is based on secret sharing and just one bilinear map, and can be viewed as an extension of the scheme in [7]. It is in fact the first KP-ABE scheme for monotone circuits based on bilinear maps.

The efficiency of our scheme depends on the number of FO-gates and their positions in the circuit. Thus, for Boolean circuits representing multilevel access structures our scheme performs better than the one in [5]. For more "complex" Boolean circuits, the KP-ABE scheme in [5] may have a better complexity than ours with respect to the number of decryption keys. However, it faces the problem of computing leveled multilinear maps. Although some progress has recently been achieved along this direction [4, 3], working with leveled multilinear maps is far more expensive than working with just one bilinear map.

Our KP-ABE_Scheme associates at least two keys to each input wire of a FO-gate. Finding ways to reduce the number of keys would be extremely helpful in order to reduce the complexity of the scheme.

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. Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013 - 33rd Annual Cryptology Conference Advances in Cryptology*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493, Santa-Barbara, États-Unis, August 2013. Springer. Preprint on IACR ePrint 2013/183.
4. 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.
5. 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.
6. 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.
7. 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.

8. Ehud D. Karnin, J. W. Greene, and Martin E. Hellman. On secret sharing systems. *IEEE Transactions on Information Theory*, 29(1):35–41, 1983.
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. G. J. Simmons. How to (Really) Share a Secret. In Shafi Goldwasser, editor, *8th Annual International Cryptology Conference on Advances in Cryptology (CRYPT '88)*, volume 403 of *Lecture Notes in Computer Science*, pages 390–448. Springer, 1988.
11. D.R. Stinson. *Cryptography: Theory and Practice*. Chapman and Hall/CRC, 3 edition, 2005.
12. T. Tassa. Hierarchical Threshold Secret Sharing. *Journal of Cryptology*, 20(2):237–264, 2007.
13. T. Tassa and N. Dyn. Multipartite Secret Sharing by Bivariate Interpolation. *Journal of Cryptology*, 22(2):227–258, 2008.

Appendix 1

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 bilinear Diffie-Hellman 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 DBDH problem. The algorithm $\mathcal{B}$ plays the role of challenger for $\mathcal{A}$ in the selective game for KP-ABE_Scheme.

The algorithm $\mathcal{B}$ is given an instance of the DBDH problem, that is: two groups G_1 and G_2 of prime order p, a generator g of G_1, a bilinear map $e : G_1 \times G_1 \to G_2$, the values g^a, g^b, g^c, and $Z_v \leftarrow \{Z_0, Z_1\}$, where $Z_0 = e(g,g)^{abc}$, $Z_1 = e(g,g)^z$, and $a, b, c, z \leftarrow \mathbb{Z}_p$.

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 $Y = e(g^a, g^b) = e(g,g)^{ab}$ and $T_i = g^{t_i}$ for all $i \in \mathcal{U}$, where

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

($\mathcal{B}$ can compute T_i because it knows r_i and g^b). Then, $\mathcal{B}$ publishes the public parameters

$$PP = (p, G_1, G_2, g, e, n, Y, (T_i | i \in \mathcal{U})).$$

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 $\mathcal{C}(A) = 0$. Given such a query, the decryption key is computed as follows. The algorithm $\mathcal{B}$ uses first a procedure $FakeShare$ which will share g^a as the procedure $Share$ shares $y = ab$ (remark that $\mathcal{B}$ does not know ab). Then, $\mathcal{B}$ delivers decryption keys based on g^b. The following 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 $e(g,g)^{abc}$.

In order to easily describe the procedure $FakeShare$ we adopt the notation $\mathcal{C}_w(A)$ for the truth value at the wire 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 $\Gamma = (w_1, w_2, X, w)$ 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^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 Γ, 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^{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:

$FakeShare(g^a, \mathcal{C})$

1. Initially, all gates of $\mathcal{C}$ are unmarked;
2. $S(o) := (g^a)$;
3. If $\Gamma = (w_1, w_2, OR, w)$ is an unmarked OR-gate and $S(w) = L$, then mark Γ and do the followings:
   (a) if $\mathcal{C}_w(A) = \mathcal{C}_{w_1}(A) = \mathcal{C}_{w_2}(A)$, then $S(w_1) := L$ and $S(w_2) := L$;
   (b) if $\mathcal{C}_w(A) = 1 = \mathcal{C}_{w_1}(A)$ and $\mathcal{C}_{w_2}(A) = 0$, then $S(w_1) := L$ and $S(w_2) := (g^{L(i)} | 1 \leq i \leq |L|)$;
   (c) if $\mathcal{C}_w(A) = 1 = \mathcal{C}_{w_2}(A)$ and $\mathcal{C}_{w_1}(A) = 0$, then $S(w_2) := L$ and $S(w_1) := (g^{L(i)} | 1 \leq i \leq |L|)$.
   Remark that, in the last two cases (b) and (c), all the elements in L are from $\mathbb{Z}_p$;
4. If $\Gamma = (w_1, w_2, AND, w)$ is an unmarked AND-gate and $S(w) = L$, then mark Γ and do the followings:
   (a) if $\mathcal{C}_w(A) = 1$, then:
      i. for each $i \in pos(L)$ choose x_i^1 uniformly at random from $\mathbb{Z}_p$ and compute $x_i^2 = (L(i) - x_i^1) \bmod p$. Define L_1 (L_2, resp.) as being the list obtained from L by replacing $L(i)$ by x_i^1 (x_i^2, resp.), for all $i \in pos(L)$;

ii. assign $S(w_1) := L_1$ and $S(w_2) := L_2$;

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

i. for each $i \in pos(L)$ choose x_i^1 uniformly at random from $\mathbb{Z}_p$ and compute $g^{x_i^2} = L(i)/g^{x_i^1}$. Define L_1 (L_2, resp.) as being the list obtained from L by replacing $L(i)$ by x_i^1 ($g^{x_i^2}$, resp.), for all $i \in pos(L)$;

ii. assign $S(w_1) := L_1$ and $S(w_2) := L_2$;

(c) 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;

(d) if $\mathcal{C}_w(A) = \mathcal{C}_{w_1}(A) = \mathcal{C}_{w_2}(A) = 0$ then:

i. for each $i \in pos(L)$ choose x_i^1 uniformly at random from $\mathbb{Z}_p$ and compute $g^{x_i^2} = L(i)/g^{x_i^1}$. Define L_1 (L_2, resp.) as being the list obtained from L by replacing $L(i)$ by $g^{x_i^1}$ ($g^{x_i^2}$, resp.), for all $i \in pos(L)$;

ii. assign $S(w_1) := L_1$ and $S(w_2) := L_2$;

5. If $\Gamma = (w, FANOUT, w_1, \ldots, w_j)$ is an unmarked FANOUT-gate and $S(w_k) = L_k$ for all $1 \leq k \leq j$, then mark Γ and do the followings:

(a) if $\mathcal{C}_w(A) = \mathcal{C}_{w_1}(A) = \cdots = \mathcal{C}_{w_j}(A) = 1$ then

i. for each $i \in pos(L_1)$ choose uniformly at random $a_i \in \mathbb{Z}_p$ and compute b_i such that $L_1(i) = (a_i + b_i) \bmod p$;

ii. compute $L_1' = (a_i | 1 \leq i \leq |L_1|)$ and $P(w_1) := (g^{b_i} | 1 \leq i \leq |L_1|)$;

iii. compute L_k' and $P(w_k)$ in a similar way to L_1' and $P(w_1)$, for all $2 \leq k \leq j$;

iv. Assign $S(w) := L_1' \cdots L_j'$;

(b) if $\mathcal{C}_w(A) = \mathcal{C}_{w_1}(A) = \cdots = \mathcal{C}_{w_j}(A) = 0$ then

i. for each $i \in pos(L_1)$ choose uniformly at random $a_i \in \mathbb{Z}_p$ and compute $g^{b_i} = L_1(i)/g^{a_i}$;

ii. compute $L_1' = (g^{a_i} | 1 \leq i \leq |L_1|)$ and $P(w_1) := (g^{b_i} | 1 \leq i \leq |L_1|)$;

iii. compute L_k' and $P(w_k)$ in a similar way to L_1' and $P(w_1)$, for all $2 \leq k \leq j$;

iv. Assign $S(w) := L_1' \cdots L_2'$;

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

Let $(S, P) \leftarrow FakeShare(g^a, \mathcal{C})$. The algorithm $\mathcal{B}$ will deliver to $\mathcal{A}$ the decryption key $D = ((D(i) | i \in \mathcal{U}), P')$, where

$$D(i) = \begin{cases} \big((g^b)^{S(i,j)/r_i} | 1 \leq j \leq |S(i)|\big), & \text{if } i \in A \\ \big(S(i,j)^{1/r_i} | 1 \leq j \leq |S(i)|\big), & \text{if } i \notin A \end{cases}$$

for any $i \in \mathcal{U}$. Remark that the key component $D(i)$ for $i \in A$ is of the form

$$\big(g^{bS(i,j)/r_i} | 1 \leq j \leq |S(i)|\big)$$

while for $i \notin A$ it is of the form

$$\big(g^{y_{i,j}/r_i} | 1 \leq j \leq |S(i)|\big) = \big(g^{by_{i,j}/br_i} | 1 \leq j \leq |S(i)|\big)$$

(for some $y_{i,j} \in \mathbb{Z}_p$) because the shares of i are all powers of g.

The distribution of this decryption key is identical to that in the original scheme. Moreover, it is easy to see that the reconstruction procedure *Recon*, applied to $V_A(i,j) = e(g,g)^{S(i,j)bc}$ for all $i \in A$ and $1 \leq j \leq |S(i)|$, returns $e(g,g)^{abc}$.

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

$$E = (A, E' = m_u Z_v, \{E_i = T_i^c = g^{cr_i}\}_{i \in A})$$

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 $\mathcal{A}$'s guess. 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 $P(v' = v) - \frac{1}{2} = \frac{1}{2}\eta$. □

Appendix 2

We will show here, by means of an example, that disjunctive multilevel access structures cannot be represented by Boolean formulas (Boolean circuits without FANOUT-gates).

Let $\mathcal{U} = \{1,2,3,4\}$, $\mathcal{U}_1 = \{1,2\}$, $\mathcal{U}_2 = \{3,4\}$, $a_1 = 2$, and $a_2 = 3$. The minimal authorized sets are $\{1,2\}$, $\{1,3,4\}$, and $\{2,3,4\}$. If this disjunctive multilevel access structure would be representable by a Boolean formula, then the following would hold:

1. 1 and 2 cannot be connected by an OR-gate because then $\{1\}$ would be authorized;
2. 1 and 2 cannot be connected by an AND-gate because $\{1,3,4\}$ is authorized and $\{3,4\}$ would become authorized too, which is a contradiction;

3. 1 and 3 cannot be connected by an OR-gate because $\{1, 2, 3\}$ is authorized and $\{2, 3\}$ would become authorized too, which is a contradiction. Similarly, 1 and 4 cannot be connected by an OR-gate;
4. 1 and 3 cannot be connected by an AND-gate because $\{1, 2\}$ is authorized and $\{2, 3\}$ would become authorized too, which is a contradiction. Similarly, 1 and 4 cannot be connected by an AND-gate;
5. 2 and 3 (2 and 4) cannot be connected by OR- or AND-gates by similar reasons as above;
6. 3 and 4 cannot be connected by an OR-gate because $\{1, 3, 4\}$ is authorized and $\{1, 3\}$ would become authorized too, which is a contradiction;
7. according to the above items, 3 and 4 can be connected only by an AND-gate Γ. But then, it is easy to see that there is no way to connect 1, 2, and Γ to obtain this access structure (the discussion is similar to the one above).

Similarly, conjunctive multilevel access structures cannot be represented by Boolean formulas.