THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 16, Special Issue 2015, pp. 287-298

AN ATTRIBUTE BASED ENCRYPTION SCHEME BASED ON MULTILINEAR MAPS

Alin Silviu BUTNARU, Petru Alexandru IOJA

Department of Computer Science, Alexandru Ioan Cuza University, Iasi, Romania
E-mail: alinnereid@gmail.com

This paper proposes an attribute based encryption scheme based on multilinear maps as a more efficient variant of the scheme of S.Garg et al. [1]. Our scheme also uses secret sharing as a way to share some secret associated to the output gate of the Boolean circuit top down to the leaves.

Key words: Attribute based encryption, Multilinear maps, Secret sharing, Key-Policy ABE.

1. INTRODUCTION

Attribute based encryption (ABE) was introduced by Sahai and Waters in [2] and can be considered a generalisation of identity based encryption. An ABE scheme encrypts a message together with some attributes such that the message can be recovered only in some circumstances, which are described below.

Attribute based encryption schemes are efficient when we want to encrypt an information and only some users can decrypt it. If we use asymmetric schemes, we need to encrypt the message for every user, with his public key. If we have a large number of users, this procedure can cost a lot. Another disadvantage for using public key schemes, is that we can't send encrypted information to users without knowing their identity. Also, using public key schemes, it's hard to implement a system where users have permission to decrypt the ciphertext based on some conditions. If efficient attribute based encryption schemes would be developed, then the encrypted cloud storage would be improved by using them.

There are two variants of ABE: Key-Policy ABE and Ciphertext-Policy ABE. Goyal et al. [3] introduced the idea of KP-ABE systems and Bethencourt et al. [4] introduced the idea of CP-ABE systems. In a Key-Policy ABE system, a ciphertext encrypting a message M is associated with an assignment γ of attributes (they are viewed as Boolean variables). A secret key SK is issued by an authority and is associated with a Boolean function f chosen from some class of allowable functions F. A user with a secret key for f can decrypt a ciphertext associated with γ, if and only if $f(\gamma) = 1$.

In a Ciphertext-Policy system, the role of encryption and key derivation are reversed. A user's private-key is associated with a set of attributes and a ciphertext specifies an access policy over a universe of attributes. A user will be able to decrypt a ciphertext, if and only if his attributes satisfy the policy of the respective ciphertext.

Contributions. We propose and ABE scheme that can be used either as a Key-Policy scheme or a Ciphertext-Policy scheme. We describe only the scheme with Key-Policy, which can be easily transformed in a Cipher-Policy scheme. Our starting point is [1] where an ABE scheme based on multilinear maps was proposed. This scheme uses multilinear maps and works for general Boolean circuits. The main idea is to associate a number of keys to each gate of the Boolean circuit and to reconstruct bottom up the secret. The output gate of the Boolean circuit will receive the secret such constructed, which will be used in order to decrypt the message. The Boolean circuit has Or-gates or And-gates on internal nodes and input-gates on leaves .

Our scheme differs from the scheme of S.Garg et al. [1] in terms of the number of keys used and access structure. In their scheme, where every gate has exactly one output and two inputs, for each Or-gate

they create four key components, for each And-gate they create three and for each wire two. When we deal with this kind of gates, we create only two keys for the Or-gate and no keys for the And-gate or wire.

2. PRELIMINARIES

This section recalls basic concepts of ABE.

Attribute based encryption. An ABE scheme consists of four probabilistic polynomial time algorithms as follows:

Setup($1^\lambda, n, l$): To construct the setup algorithm we require an input parameter λ, the maximum depth l of a circuit and the number of Boolean inputs n (number of attributes in our scheme). This algorithm outputs public parameters PP and a master key MK.

Encrypt(PP, γ, M): To construct the encryption algorithm we require as input the public parameters PP, the set of attributes γ and a message M. This algorithm outputs a ciphertext CT.

KeyGen(MK, Cr): To construct the keygen algorithm we require as input the master key MK and the Boolean circuit Cr. This algorithm outputs a secret key SK that enables the user to decrypt a message under a set of attributes γ if and only if $Cr(\gamma) = 1$.

Decrypt(SK, CT): To construct the decryption algorithm we require as input a secret key SK and ciphertext CT. This algorithm tries to decrypt the ciphertext CT and outputs a message M if successful. Otherwise, it outputs a special symbol $\perp$.

Each ABE scheme should satisfy the following correctness property: Consider all messages M, attributes γ and depth l circuits Cr where $Cr(\gamma) = 1$. If $Encrypt(PP, \gamma, M) \to CT$ and $KeyGen(MK, Cr) \to SK$ where PP, MK were generated from a call to the setup algorithm, then $Decrypt(SK, CT) = M$.

Multilinear maps. We say that a map $e: G_1^n \to G_2$ is an $n-multilinear\ map$ if it satisfied the following properties:

1. G_1 and G_2 are groups of the same prime order.
2. If $a_1, a_2, ..., a_n \in Z$ and $x_1, x_2, ..., x_n \in G_1$ then

$$e(x_1^{a_1}, x_2^{a_2}, ..., x_n^{a_n}) = e(x_1, x_2, ..., x_n)^{a_1 a_2 \cdots a_n}.$$

3. The map e is non-degenerate: if $g \in G_1$ is a generator of G_1 then $e(g, g, ..., g)$ is a generator of G_2.

A multilinear map generator $G(1^\lambda, k)$ is a randomized algorithm that runs in a polynomial time and takes as input a security parameter λ, a positive integer k and outputs a sequence of groups $\vec{G} = (G_1, G_2, ..., G_k)$ each of large prime order $p > 2^\lambda$. Let g_i be a canonical generator of G_i and $g = g_1$.

In this paper, we sometimes abuse notation and drop the subscripts i,j writing, for example : $e(g_i^a, g_j^b) = g_{i+j}^{ab}$.

The multilinear Diffie-Hellman assumption. This assumption states that given $g, g^{a_1}, g^{a_2}, ..., g^{a_{n+1}} \in G_1$ it is hard to compute $e(g, g, .., g)^{a_1 a_2 \cdots a_{n+1}}$ in G_2. In other words, we say that the multilinear map generator G satisfies the multilinear Diffie-Hellman assumption if for all polynomial time randomized algorithms A, the probability of being able to compute $e(g, g, .., g)^{a_1 a_2 \cdots a_{n+1}}$ from $g, g^{a_1}, g^{a_2}, ..., g^{a_{n+1}}$ is negligible.

Boolean Circuit. For Boolean circuits we used the standard notation and concepts as in [1]. That is, we deal with monotone Boolean circuits having only input gates, or-gates, and-gates and just one output. The logic gates have at least two inputs and one or more outputs.

Circuit Notation. We need to modify the circuit notation from [1] because we use more than two inputs for and/or-gates. Our circuits will be a quadruple $Cr = (n, q, l, Gates)$. We let n be the number of inputs, q the number of gates and l the number of levels in the circuit. Let $Gates$ be a list of $circuit-gates$, where a $circuit-gate = (type, id, level, input-wires)$. The $type$ can be $Input, And, Or$, id is a unique natural number for each gate, $level$ is the level where the gate is found and $input-wires$ is a list of id's for the inputs ($wires-id$) of the gate. If the $type$ is $Input$ the $input-wires$ will be empty. The id's will be set from left to right, from bottom to top in an ascending order, starting from 1 to the number of nodes in the circuit. Because of this, the input-gates will have as id's $= \{1,...,n\}$. The $wires-id$ is a list of natural numbers set like id's for the gates, except they are now set for the wires in the circuit. Let $input_wires$ for a node be a list of $wires-id$, for the wires that are input for this node, and $output_wires$ a list for the wires that are output for this node. We also define a function $depth(w)$ where if $w \in input\ depth(w) = 1$ and in general $depth(w)$ of wire w is equal to the shortest path to an input wire plus 1. Because we use layered circuits, if $depth(w) = j$, all the inputs for w will have $depth = j - 1$.

3. OUR CONSTRUCTION

We propose the following ABE scheme.

Setup($1^\lambda, n, l$): To construct the setup algorithm we require an input parameter λ, the maximum depth l of a circuit and the number of Boolean inputs n (number of attributes in our scheme).

Using a group generator $G(1^\lambda, k = l+1)$ we produce groups $\vec{G} = (G_1, G_2, ..., G_k)$ of prime order p, with generators $g_1, ..., g_k$. Next we chose a random $\alpha \in Z_p$ and for each attribute we choose a random number $t_i \in G_1$. Let $G = G_1$ and $g = g_1$.

The public parameters PP are $g_k^\lambda, t_1, ..., t_n$ and the group sequence.

The master key MK is $(g_{k-1})^\lambda$.

Encrypt(PP, γ, M): To construct the encryption algorithm we require as input the public parameters PP, the set of attributes γ and a message M.

The encryption algorithm chooses a random $s \in Z_p$ and outputs the ciphertext $CT = (C_M, g^s, \forall i \in \gamma\ C_i = t_i^s)$ where $C_M = M(g_k^\alpha)^s$.

KeyGen($MK, Cr = (n, q, l, Gates)$): To construct the keygen algorithm we require as input the master key MK and the Boolean circuit Cr. This algorithm outputs a secret key SK that enables the user to decrypt a message under a set of attributes γ if and only if $Cr(\gamma) = 1$. The key generation algorithm chooses a random $r \in Z_p$. The algorithm then produces a header component : $K_H = (g_{k-1})^{(\alpha - r)}$.

In the secret sharing procedure, we share g_k^{rs} starting from the output wire of the output gate of the Boolean circuit. The secret sharing is:

- **Input-wire:** Suppose that the input-wire has the output g_2^{xs}. We will generate a random $z \in Z_p$ and create two key components: $b_1 = g^x t_i^z$ and $b_2 = g^{-z}$, where i is the id of this gate.

• **And-gate:** Consider first the case with exactly one output wire and q input wires. Assume that the output wire of the gate is g_j^{xs}. First, we generate $q-1$ random numbers $a_1,...,a_{q-1} \in Z_p$ and then assign $g_j^{(x-a_1-a_2-...-a_{q-1})s}$ to the first input wire and $g_j^{a_1 s},...,g_j^{a_{q-1}s}$ to the next $q-1$ input wires.

This construction is illustrated in Fig. 1.

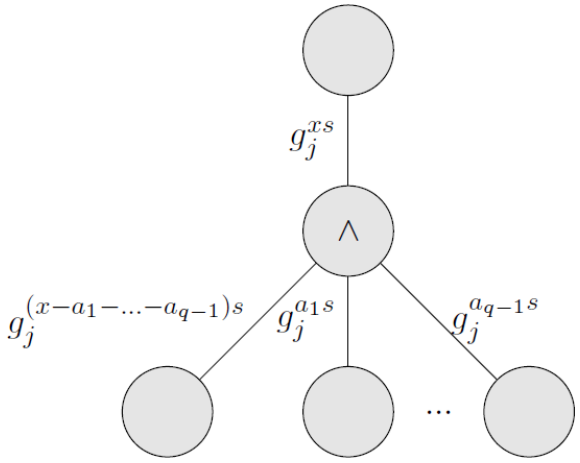

Fig. 1 – And-gate with exactly one output.

If the And-gate has more than one output, let $g_{j+1}^{x_1 s}, g_{j+1}^{x_2 s},...g_{j+1}^{x_o s}$ be the values associated to the output wires. Like in the first case, we generate $q-1$ random numbers $a_1,...,a_{q-1} \in Z_p$ and one random number $x \in Z_p$. We assign $g_j^{(x-a_1-a_2-...-a_{q-1})s}$ to the first input wire and $g_j^{a_1 s},...,g_j^{a_{q-1}s}$ to the next $q-1$ input wires. We then generate o random numbers $b_1, b_2,..., b_o$ such that $x_i = xb_i$ for all $1 \le i \le o$. The key components: $g^{b_1},...,g^{b_o}$ are also assigned to the gate.

This construction is illustrated in Fig. 2.

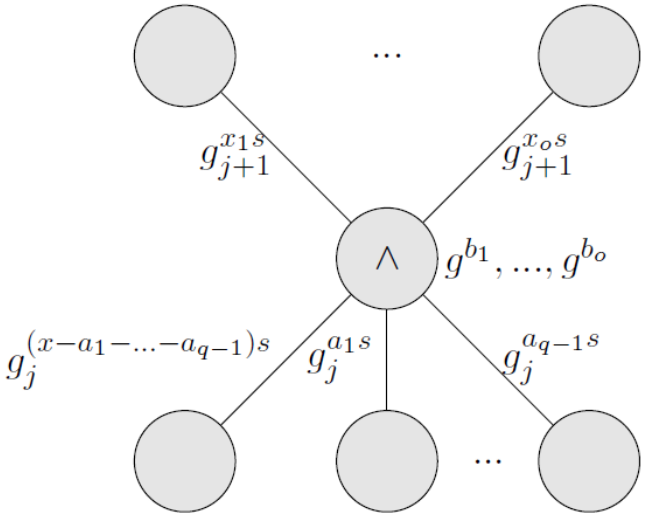

Fig. 2 – And-gate with more than one output.

• **Or-gate:** Suppose that the Or-gate has q input wires and o output wires. Let $g_{j+1}^{x_1 s}, g_{j+1}^{x_2 s}, ..., g_{j+1}^{x_o s}$ be the values associated to the output wires. We generate q random numbers $a_1, a_2, ..., a_q \in Z_p$ and assign $g_j^{a_i s}$ to the i^{th} input wire. The key components $g^{b_{i,j}}$ are also assigned to the gate where $x_i = a_j \cdot b_{i,j}$ for all $1 \le i \le o$ and $1 \le j \le q$.

This construction is illustrated in Fig. 3.

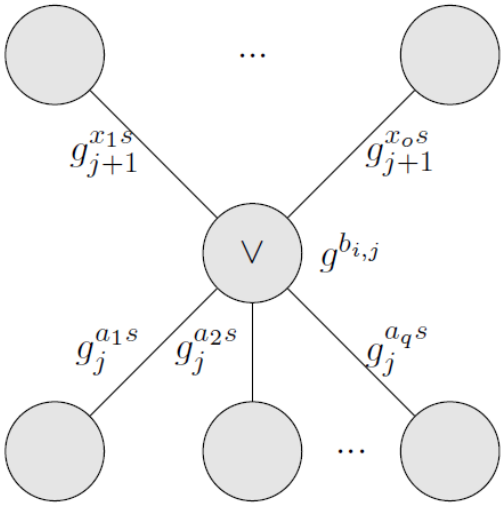

Fig. 3 – Or-gate.

In order to create this sharing scheme, first we need to create a circuit like the initial one where we put coefficients on wires which will allow us to create the key components.

Decrypt(SK, CT **):** Suppose that we are evaluating decryption for a secret key SK associated with a circuit Cr and a ciphertext with input x. We will be able to decrypt it only if $Cr(x) = 1$.

The goal of decryption is to compute $g_k^{\alpha s}$ (M can be recovered by computing $M = C_M / g_k^{\alpha s}$). First, we compute $K'_H = e(K_H, g^s) = e(g_{k-1}^{\alpha - r}, g^s) = g_k^{\alpha s} g_k^{-rs}$. Our goal is now reduced to computing g_k^{rs}.

Next we evaluate the circuit bottom up. If $Cr_w(x) = 1$ then, our algorithm can compute the information in this node, like below. Our algorithm starts from the leaves and tries to go up to the last output and if the user is allowed to decrypt the ciphertext, in the last output will be g_k^{rs}.

We may assume that all inputs of an And-gate have the same generator g_j. If, for instance, some inputs have the generator g_u, with $u < j$, then we may apply $e(g_u^x, g)$ to obtain g_{u+1}^x. This procedure can be repeated until we obtain g_j^x.

For each gate, we will do the following :

• **Input-wire:** If $Cr_i(x) = 1$, the ciphertext contains C_i and we can calculate $e(b_1, g^s) \cdot e(b_2, C_i) = e(g^x t_i^z, g^s) \cdot e(g^{-z}, t_i^s) = g_2^{xs}$.

• **And-gate:** Consider first the case with exactly one output wire and q input wires, where all inputs have $Cr(x) = 1$, like we described earlier, the first input wire contains $g_j^{(x - a_1 - a_2 - ... - a_{q-1})s}$ and the next $q-1$

inputs contain $g_j^{a_i s}$. Multiplying the information from these inputs we get g_j^{xs}, which will be contained by the output wire.

If the And-gate has o output wires and q input wires, where all inputs have $Cr(x)=1$, like we described earlier, the first input contains $g_j^{(x-a_1-a_2-\ldots-a_{q-1})s}$ and the next $q-1$ inputs contain $g_j^{a_i s}$. Multiplying the information from these inputs we will get g_j^{xs}. Using key components $g^{b_1},\ldots,g^{b_o}$, where $b_i = x_i x^{-1}$ for all $1 \le i \le o$, we will compute for each key b_i, $e(g_j^{xs}, g^{b_i}) = g_{j+1}^{xsx_i x^{-1}} = g_{j+1}^{x_i s}$, then we assign $g_{j+1}^{x_i s}$ to the i^{th} output wire.

- **Or-gate:** Suppose that the Or-gate has o output wires and q input wires and at least one input has $Cr(x)=1$, like we described earlier, the i^{th} input contains $g_j^{a_i s}$. Using the key components $g^{b_{i,j}} = g^{x_i a_j^{-1}}$ from any input wire l we can compute, for each output wire i, $e(g_j^{a_l s}, g^{b_{i,l}}) = e(g_j^{a_l s}, g^{x_i a_l^{-1}}) = g_{j+1}^{x_i s}$. This value, $g_{j+1}^{x_i s}$, will be assigned to the i^{th} output wire.

If $Cr(x)=1$ then the algorithm will output g_k^{rs}, this is the value assigned to the last output wire. Finally, multiplying K'_H with g_k^{rs} we obtain $g_k^{\alpha s}$ and dividing C_M by $g_k^{\alpha s}$ we obtain the message M.

When the Or-gates have many output wires we may try to change the generator two times in order to reduce the number of keys. We describe below this new construction indicating only the encryption and decryption for Or-gates.

Encryption part: Suppose that the Or-gate has q input wires and o output wires.

Let $g_{j+2}^{x_1 s}, g_{j+2}^{x_2 s},\ldots,g_{j+2}^{x_o s}$ be the values associated to the outputs. We generate q random numbers $a_1, a_2,\ldots,a_q \in Z_p$ and assign $g_j^{a_i s}$ to the i^{th} input wire. We then generate a random number $x \in Z_p$ and publish q key components g^{b_i}, where $x = a_i b_i$ for all $1 \le i \le q$. For each output wire, we publish o key components g^{c_k}, where $x = x_k c_k$ for all $1 \le k \le o$.

This construction is illustrated in Fig. 4.

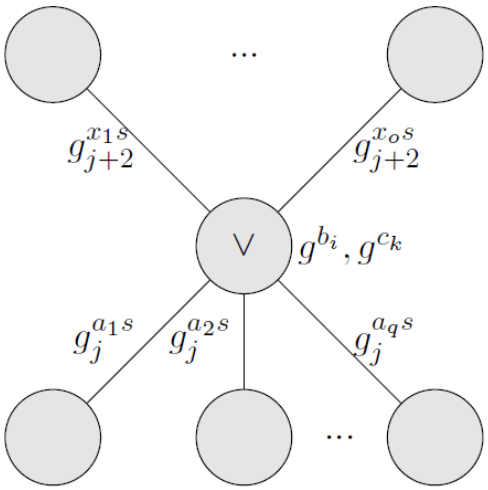

Fig. 4 – Or-gate, switching generator two times.

Decryption part: Suppose that the Or-gate has o output wires and q input wires and at least one input has $Cr(x)=1$, like we described above, the i^{th} input contains $g_j^{a_i s}$. Using key components g^{b_i}, starting from any input wire, we can compute $e(g_j^{a_i s}, g^{b_i}) = g_{j+1}^{xs}$. Using key components g^{c_k} and g_{j+1}^{xs} we compute $e(g_{j+1}^{xs}, g^{c_k}) = g_{j+2}^{x_k s}$ for all $1 \le k \le o$ and assign $g_{j+2}^{x_k s}$ to the $k^t h$ output wire.

Using this method we only need $q+o$ key components but we need to change the generator two times.

Examples of sharing on different gates. For each And-gate with exactly one output wire we share the output to the input gates without using any key components. For example, if we have g_j^{xs} in the output and we have three inputs, we generate two random numbers a,b and we assign $g_j^{(x-a-b)s}$ to the first input wire, g_j^{as} to the second input wire and g_j^{bs} to the last one. If all the inputs are satisfied, all the values from the inputs are the ones from above and we can multiply them in order to obtain the value associated to the output wire, which is $g_j^{(x-a-b)s} \cdot g_j^{as} \cdot g_j^{bs} = g_j^{xs}$.

For each And-gate with more than one output we share the outputs to the input wires using a number of key components equal to the number of output wires. For example, if we have two outputs with values $g_{j+1}^{x_1 s}$ and $g_{j+1}^{x_2 s}$ and three inputs, first we generate a random number x and two other random numbers a,b such that $x_1 = xa$ and $x_2 = xb$ and we assign $g_j^{(x-a-b)s}$ to the first input, g_j^{as} to the second input and g_j^{bs} to the last one. For this gate, we attach the key components g^a, g^b. Multiplying the values from the input gates we get g_j^{xs} and using the key components and applying the multilinear function we get the values from the outputs, which are $e(g_j^{xs}, g^a) = g_{j+1}^{x_1 s}$ and $e(g_j^{xs}, g^b) = g_{j+1}^{x_2 s}$.

For each Or-gate, if we change the group generator once, the number of key components will be equal to the number of inputs multiplied by the number of outputs. If we change the group generator twice, the number of key components will be equal to the number of inputs plus the number of outputs.

4. SECURITY OF OUR CONSTRUCTION

We prove selective security of our scheme under the k-MDDH assumption, that given $g = g_1, g^s, g^{c_1}, \ ..., g^{c_k} \in Z$ it is hard to distinguish $T = g_k^{s \prod_{j \in [1,k]} c_j}$. Recall first the security game.

In the game-based security definition, as in other similar systems (e.g.[1], [2], [3]), an attacker is able to query for multiple keys, but not the ones that can easily be used to decrypt the ciphertext. The adversary may perform a polynomial number of requests for private keys corresponding to any circuit Cr, but must encrypt some string x^* (the x^* is a string of length n with elements $0,1$ and it shows what attributes adversary has) such that every circuit Cr for which a private key was requested has $Cr(x^*) = 0$. The security game has the following steps:

Setup. The challenger runs the setup algorithm and gives the public parameters PP to the adversary and keeps the MK.

Phase 1. The adversary makes any polynomial number of queries in order to obtain private keys for a circuit description f of its choice. The challenger returns for each query the output of $KeyGen(MK, Cr)$.

Challenge. The adversary sends two messages M_0 and M_1 having the same length. He also gives a challenge string x^* such that for all f requested in Step 1 we have $f(x^*)=0$. The challenger then chooses a random $b \in \{0,1\}$, and computes $Encrypt(PP, x^*, M_b) \to CT^*$ and sends CT^* to the adversary.

Phase 2. Phase 1 can be repeated with the restriction that for all f requested $f(x^*)=0$.

Guess. The adversary outputs $b' \in \{0,1\}$.

The advantage of an adversary A in this game is defined as $Pr[b'=b]-\frac{1}{2}$.

We say that an attribute-based encryption scheme for circuits is **secure** if all polynomial time adversaries have a negligible advantage in the described game.

We say that a system is **selectively secure** if the system is secure in a game where we add a new step before setup called Init, where the adversary can send to the challenger the string x^*.

Now, we can prove the following : The construction given in the previous section achieves selective security for arbitrary circuits of depth $k-1$ in the KP-ABE security game under the k-MDDH assumption.

Proof. The init, setup and challenge ciphertext are the same as in [1].

Init. B first receives the $l+1$-multilinear problem where it is given the group description $\vec{G}=(G_1,G_2,...,G_k)$ and a problem instance $g, g^s, g^{c_1},...,g^{c_k}, T$. T is either $g_k^{s\prod_{j\in[1,k]}c_j}$ or a random group element in G_k.

Next, the attacker declares the challenge input $x^* \in \{0,1\}^n$.

Setup. B chooses random $y_1,...y_n \in Z_p$. For $i \in [1,n]$ set

$$t_i = \begin{cases} g^{y_i} & \text{if} x_i^* = 1 \\ g^{y_i+c_1} & \text{if} x_i^* = 0 \end{cases}$$

Remark. We need g^{y_i} to be either statistically close to or indistinguishable from $g^{y_i+c_1}$.

Next, B sets $g_k^{\alpha} = g_k^{\varepsilon+\prod_{i\in[1,k]}c_i}$ where ε is chosen randomly. It computes this using $g^{c_1},...,g^{c_k}$ from the assumption by means of the iterated use of the pairing function.

Remark. We need $g_k^{\alpha} = g_k^{\varepsilon+\prod_{i\in[1,k]}c_i}$ to be either statistically close to or indistinguishable from g_k^{ε}.

Challenge Ciphertext. Let $S^* \subseteq [1,n]$ be the set of input indices where $x_i^*=1$. The reduction algorithm receives two messages M_0, M_1 and flips a coin b. B creates the challenge ciphertext as: $CT=(M_b \cdot T \cdot g_k^{s\varepsilon}, g^s, \forall i \in \gamma\ C_i=(g^s)^{y_i})$.

If $T = g_k^{s\prod_{j\in[1,k]}c_j}$, then this is an encryption of M_b; otherwise if T was chosen random in G_k then the challenge ciphertext contains no information about the message from the attacker's view.

KeyGen Phase. Both key generation phases are executed in the same manner by the reduction algorithm. Therefore, we describe them once here. The attacker will give a circuit Cr to the reduction algorithm such that $Cr(x^*)=0$. Consider a gate w at depth j and the simulators viewpoint of r_w. If

$Cr_w(x^*) = 0$ then the simulator will view r_w as the term $c_1 \cdot c_2 \cdot ... \cdot c_{j+1}$ plus some additional known randomization terms. If $Cr_w(x^*) = 1$ then the simulator will view r_w as 0 plus some additional known randomization terms. If we can keep this property intact for simulating the keys up the circuit, the simulator will view r as $c_1 \cdot c_2 \cdot ... \cdot c_k$. This will allow for it to simulate the header component K_H by cancellation.

We describe how to create the key components for each gate:

- **Input-wire**: Suppose that $w \in [1, n]$ and therefore is an input wire.

If $(x^*)_w = 1$ then we choose z random and the key components will be: $b_1 = g^x t_w^z$ and $b_2 = g^{-z}$.

If $(x^*)_w = 0$ then we let $x = c_1 c_2 + \eta_w$ (the output coefficient) and $z = -c_2 + \nu_w$, where η_w and ν_w are randomly chosen elements. The key components are: $(b_1 = g^{c_1 c_2 + \eta_w} t_w^{-c_2 + \nu_w}$, $b_2 = g^{c_2 - \nu_w}) = (g^{-c_2 y_w + \eta_w + (y_w + c_1)\nu_w}, g^{c_2 - \nu_w})$.

Remark. Here we need $g^{-c_2 y_w + \eta_w + (y_w + c_1)\nu_w}$ to be appropriately close to a randomly chosen element.

- **And-gate**: Let $j = depth(w)$. For the first case, where there is exactly one output wire (g_j^{xs}) and q input wires (let them be $g_j^{r_1 s}, g_j^{r_2 s}, ..., g_j^{r_q s}$).

If $(x^*)_w = 1$ then we generate $q-1$ random numbers $a_1, ..., a_{q-1}$ and assign $g_j^{(x - a_1 - a_2 - ... - a_{q-1})s} = g_j^{r_1 s}$ to the first input wire and $g_j^{a_1 s} = g_j^{r_2 s}, ..., g_j^{a_{q-1} s} = g_j^{r_q s}$ to the next $q-1$ input wires.

If $(x^*)_w = 0$ then exist at least one input wire that are not satisfied. Because of this, exists one r_i that is view as $c_1 \cdot c_2 \cdot ... \cdot c_j + \eta_{r_i}$ where η_{r_i} is a random number.

Because at the sharing part we chose random number $a_1, ..., a_{q-1} \mathbb{Z}$, when we have at least one wire not satisfied, when we multiply all the input wires values we will obtain $g_j^{c_1 \cdot c_2 \cdot ... \cdot c_j + \eta}$ where η it's a sum of some η_{r_i}.

Remark that $g_j^{c_1 \cdot c_2 \cdot ... \cdot c_j + \eta}$ is appropriately close to a randomly chosen element.

For the case where there are o output wires ($g_{j+1}^{x_1 s}, g_{j+1}^{x_2 s}, ..., g_{j+1}^{x_o s}$) and q input wires (let them be $g_j^{r_1 s}, g_j^{r_2 s}, ..., g_j^{r_q s}$):

If $(x^*)_w = 1$ then we generate $q-1$ random numbers $x, a_1, ..., a_{q-1}$ and assign $g_j^{(x - a_1 - a_2 - ... - a_{q-1})s} = g_j^{r_1 s}$ to the first input wire and $g_j^{a_1 s} = g_j^{r_2 s}, ..., g_j^{a_{q-1} s} = g_j^{r_q s}$ to the next $q-1$ input wires. Then we generate o random numbers $b_1, b_2, ..., b_o$ such that $x_i = x b_i$ for all $1 \le i \le o$. The key components: $g^{b_1}, ..., g^{b_o}$ are also assigned to the gate.

If $(x^*)_w = 0$ then at least one input wire that is not satisfied exists. Because of this, one r_i exists that is viewed as $c_1 \cdot c_2 \cdot ... \cdot c_j + \eta_{r_i}$ where η_{r_i} is a random number. Also x_i is viewed as $c_1 \cdot c_2 \cdot ... \cdot c_{j+1} + \eta_{x_i}$ for all $i \in [1, o]$. For the key components, b_i is viewed as $c_{j+1} + \eta_{b_i}$ where η_{b_i} is a random number.

Remark that $g_j^{c_1 \cdot c_2 \cdot ... \cdot c_j + \eta}$ is appropriately close to a randomly chosen element. Applying $e(g_{j+1}^{(c_1 \cdot c2 \cdot ... \cdot c_j + \eta)s}, g^{c_{j+1} + \eta_{b_i}})$ we obtain $g_{j+1}^{(c_1 \cdot c_2 \cdot ... \cdot c_{j+1} + c_1 \cdot c_2 \cdot ... \cdot c_j \cdot \eta_{b_i} + c_{j+1} \cdot \eta + \eta \cdot \eta_{b_i})s}$ which is also appropriately close to a randomly chosen element.

- **Or-gate**: Let $g_j^{r_1 s}, g_j^{r_2 s}, ..., g_j^{r_q s}$ be the values on the input wires, $g_{j+1}^{x_1 s}, g_{j+1}^{x_2 s}, ..., g_{j+1}^{x_o s}$ be the values on the output wires and $j = depth(w)$.

If $(x^*)_w = 1$ we generate q random numbers $a_1, a_2, ..., a_q$ and assign $g_j^{a_i s}$ to the i^{th} input wire. The key components $g^{b_{i,j}}$ are also assigned to the gate where $x_i = a_j \cdot b_{i,j}$ for all $1 \le i \le o$ and $1 \le j \le q$.

If $(x^*)_w = 0$ none of the input wires are satisfied. Because of this, for all $i \in [1, q]$ r_i is viewed as $c_1 \cdot c_2 \cdot ... \cdot c_j + \eta_{r_i}$ where η_{r_i} is a random number. Also, for all $i \in [1, o]$ x_i is view as $c_1 \cdot c_2 \cdot ... \cdot c_{j+1} + \eta_{x_i}$. For the key components, $b_{i,j}$ is viewed as $c_{j+1} + \eta_{b_{i,j}}$ where $\eta_{b_{i,j}}$ is a random number.

From any input wire l, applying for each output wire i, $e(g_j^{(c_1 \cdot c_2 \cdot ... \cdot c_j + \eta_{r_l}) s}, g^{c_{j+1} + \eta_{b_{i,j}}})$ we will get $g_{j+1}^{(c_1 \cdot c_2 \cdot ... \cdot c_{j+1} + c_1 \cdot c_2 \cdot ... \cdot c_j \cdot \eta_{b_{i,j}} + c_{j+1} \cdot \eta_{r_l} + \eta_{r_l} \cdot \eta_{b_{i,j}}) s}$.

Remark that $g_{j+1}^{(c_1 \cdot c_2 \cdot ... \cdot c_{j+1} + c_1 \cdot c_2 \cdot ... \cdot c_j \cdot \eta_{b_{i,j}} + c_{j+1} \cdot \eta_{r_l} + \eta_{r_l} \cdot \eta_{b_{i,j}}) s}$ is appropriately close to a randomly chosen element.

For the output gate we chose η_w at random. At the end we have $r = \prod_{i \in [1,k]} c_i + \eta_r$ for the output gate. Thus, the header component of the key is computed as $K_H = (g_{k-1})^{\alpha - r} = (g_{k-1})^{\varepsilon - \eta_w}$.

Guess Phase. The challenger receives back the guess $M' \in \{0,1\}$ from the adversary. If $M' = 1$ it guesses that T is a tuple; otherwise, it guesses that it's random.

This shows that any adversary that runs in polynomial time with non-trivial advantage in the KP-ABE selective security game will have an identical advantage in breaking the k-MDDH assumption.

For the Or-gate where we change the generator two times, in the KeyGen Phase we will create the key components as follows:

Let $g_j^{r_1 s}, g_j^{r_2 s}, ..., g_j^{r_q s}$ be the values on the input wires, $g_{j+2}^{x_1 s}, g_{j+2}^{x_2 s}, ..., g_{j+2}^{x_o s}$ be the values on the output wires and $j = depth(w)$.

If $(x^*)_w = 1$ we generate q random numbers $a_1, a_2, ..., a_q$ and assign $g_j^{a_i s}$ to the i^{th} input wire. Next we will generate a random number x and attach the key components g^{b_i} where $x = a_i b_i$ for all $1 \le i \le q$. We also publish o key components g^{d_k}, where $x = x_k d_k$ for all $1 \le k \le o$.

If $(x^*)_w = 0$ none of the input wires are satisfied. Because of this, for all $1 \in [1, q]$ r_i is viewed as $c_1 \cdot c_2 \cdot ... \cdot c_j + \eta_{r_i}$ where η_{r_i} is a random number. Also x_i is viewed as $c_1 \cdot c_2 \cdot ... \cdot c_{j+2} + \eta_{x_i}$ for all $i \in [1, o]$. For the key components, b_i is viewed as $c_{j+1} + \eta_{b_i}$ where η_{b_i} is a random number and d_i is viewed as $c_{j+2} + \eta_{d_i}$ where η_{d_i}.

From any input wire l, applying $e(g_j^{(c_1 \cdot c_2 \cdot ... \cdot c_j + \eta_{r_l}) s}, g^{c_{j+1} + \eta_{b_l}})$ we will get $g_{j+1}^{(c_1 \cdot c_2 \cdot ... \cdot c_{j+1} + c_1 \cdot c_2 \cdot ... \cdot c_j \cdot \eta_{b_l} + c_{j+1} \cdot \eta_{r_l} + \eta_{r_l} \cdot \eta_{b_l}) s}$.

Remark that $g_{j+1}^{(c_1 \cdot c_2 \cdot ... \cdot c_{j+1} + c_1 \cdot c_2 \cdot ... \cdot c_j \cdot \eta_{b_l} + c_{j+1} \cdot \eta_{r_l} + \eta_{r_l} \cdot \eta_{b_l}) s}$ is appropriately close to a randomly chosen element.

Because of this, if we apply $e(g_{j+1}^{(c_1 \cdot c_2 \cdot ... \cdot c_{j+1} + c_1 \cdot c_2 \cdot ... \cdot c_j \cdot \eta_{b_l} + c_{j+1} \cdot \eta_{r_l} + \eta_{r_l} \cdot \eta_{b_l}) s}, g^{c_{j+2} + \eta_{d_i}})$ for this element we also get an element that is appropriately close to a random chosen element.

REFERENCES

1. S. GARG, C. GENTRY, S. HALEVİ, A. SAHAİ, B. WATERS, *Attribute-Based Encryption for Circuits from Multilinear Maps*, Lecture Notes in Computer Science, **8043**, 2013, pp. 479-499, CRYPTO 2013.
2. A. SAHAİ, B. WATERS, *Fuzzy Identity Based Encryption*, Lecture Notes in Computer Science, **3494**, 2005, pp. 457-473, Eurocrypt 2005.
3. V. GOYAL, O. PANDEY, A. SAHAİ, B. WATERS, *Attribute-based encryption for fine-grained access control of encrypted data*, Proceedings of the 13th ACM conference on Computer and communications security, pp. 89 - 98, New York, NY, USA, 2006.
4. J. BETHENCOURT, A. SAHAİ, B. WATERS. *Ciphertext-policy attribute-based encryption*, Proceedings of the 2007 IEEE Symposium on Security and Privacy, IEEE Computer Society, pp. 321 - 334, Washington, DC, USA, 2007.
5. D. BONEH, A. SİLVERBERG, *Applications of Multilinear Forms to Cryptography*, Contemporary Mathematics, **324**, pp. 71-90, 2003.