Weighted Attribute-Based Encryption with Parallelized Decryption

Alexandru Ioniţă[1] [0000−1111−2222−3333]

[1] Simion Stoilow Institute of Mathematics of the Romanian Academy, Bucharest, Romania
[2] Department of Computer Science, Alexandru Ioan Cuza University of Iaşi, Iaşi, Romania
alexandru.p.ionita@gmail.com
http://www.students.info.uaic.ro/~alexandru.ionita

Abstract. Unlike conventional ABE systems, which support Boolean attributes (with only 2 states: 1 and 0, or "Present" and "Absent"), weighted Attribute-based encryption schemes also support numerical values attached to attributes, and each terminal node of the access structure contains a threshold for a minimum weight. We propose a weighted ABE system, with access policy of logarithmic expansion, by dividing each weighted attribute in sub-attributes. On top of that, we show that the decryption can be parallelized, leading to a notable improvement in running time, compared to the serial version.

Keywords: Attribute-based Encryption, Bilinear Maps, Public-key Encryption, Access Control, Key Policy

1 Introduction

As interest in Cloud Computing and Internet of Things grew significantly, so did the interest in more expressive encryption and access control possibilities. In this context, Attribute-based encryption (ABE), introduced in (Sahai and Waters, 2005) as an refinement for Identity-based Encryption (Shamir, 1984), witnessed great attention in the past decade.

Depending on how the access policy is linked to the ABE systems, we have two main types:

- *Key-policy* ABE (KP-ABE), first introduced in (Goyal et al., 2006) encrypts a message alongside some attributes; the decryption keys have an access structure (such as a Boolean formula) attached. The decryption is possible if and only if the key's access structure is satisfied with the ciphertext's attributes.
- *Ciphertext-policy* ABE (CP-ABE), in contrast with KP-ABE, links the access structure to the ciphertext, and attributes to the decryption keys. First such system was proposed in (Bethencourt et al., 2007).

Researchers are trying to find more and more flexible access structures that can be used in ABE systems. Starting from well known ABE systems for Boolean Access Trees

(Goyal et al., 2006; Bethencourt et al., 2007) and Linear Secret Sharing Schemes (Waters, 2011), more complex ones are created for Boolean Circuits (Ţiplea and Drăgan, 2014; Hu and Gao, 2017), non-monotonic access structures (Ostrovsky et al., 2007) or compartmented access structures (Tiplea et al., 2020).

While conventional ABE supports only two states for each attribute ("True"/"False" or "Present"/"Absent"), a Weighted ABE system extends the supported access structures to more complex structure: Each attribute can have a value associated to it. For example, in order to describe a role in a software company, we could assign to each position an integer, decreasing according to the company's hierarchy: "ROLE:4" could be a *Junior Developer*, "ROLE:3" a *Senior Developer*, "ROLE:2" - *Manager*, and "ROLE:1" - *Director*. Therefore, different types of ABE were constructed in order to meet these needs, such as ABE with Range Attributes (Gay et al., 2015; Attrapadung et al., 2018), or Weighted ABE (Wang et al., 2016; Li et al., 2021; Liu et al., 2014).

Imagine that in a large company, which uses a Cloud service to share files and important data with customers, all files are encrypted using depictive attributes, such as file type ("TYPE:Java") or last modification date ("YEAR":2020). Then consider a Manager that must be have access to all "SQL" or "CSV" files written (and later encrypted) by developers in the last year. Using KP-ABE with a weighted "ROLE" attribute, this problem could be solved using an access policy describing:

("ROLE>2" AND "YEAR>2020" AND
("TYPE:SQL" OR "TYPE:CSV"))

Without weighted attributes, in order to perform this task, the access policy would need to have an attribute for each year and each role. This will become less and less efficient with the increase of the maximum possible attribute numerical value. Our system provides a more efficient solution to this problem.

1.1 Related Work

The problem of weighted attributes and integer comparisons in the access structure has been a problem of high interest, being addressed even from the first CP-ABE system proposed by Bethencourt et al. (Bethencourt et al., 2007) in 2007. They described a method for realizing integer comparisons using access trees, and by splitting every numerical attribute in $2log(N)$ values, two for each bit of information.

One of the first Weighted ABE was proposed in (Liu et al., 2014), a *key-policy* scheme which used chained components in order to describe a weighted attribute. Thus, their system is inefficient, the length of the chain being equal to the weight of the attribute, resulting in linear number of components for each attribute.

Wang et al. (Wang et al., 2016) proposed in 2016 a weighted CP-ABE system which resolves the key escrow problem for use in Cloud Systems. They support both weighted and binary attributes. However, the size of the ciphertext and the encryption time grow linear on the attribute weight, with each new weighted attribute.

A more efficient solution for the *ciphertext-policy* variant was proposed in (Xue et al., 2017) where the authors achieved logarithmic expansion for each weighted attribute, by using 0- and 1- Encodings of the weights.

A very recent work (Li et al., 2021) presents another Weighted CP-ABE approach using 0- and 1-Encodings, which proves to be the most efficient in practical performance tests among the existing CP-ABE scheme with weighted attribute support. Their system also support online and offline encryption, and it is designed for the *Internet of Health Things*.

Another work in this area was proposed by Attrapadung et al. (Attrapadung et al., 2018) in 2018, which addresses the problem of range attributes. Unlike weighted attributes, which have only a lower bound on the attribute weight, a range attribute can also have an upper bound for it's value. Their system is the first one with sub-linear complexity and no restrictions upon the access tree policy.

ABE with parallel decryption. Although most ABE systems could be implemented using parallelized algorithms, there are few works addressing this issue. The only relevant previous work we could found being an ABE system for Internet of Vehicles, very recently proposed in (Feng et al., 2020). They describe a general method for outsourcing the decryption over multiple machines for parallel computation in ABE systems with trees as access policy. They ensure that the parallel outsourcing decryption is secure on a honest, but curious outsourcing server. We show that in our system the parallelization of the decryption can be done very easily, without other alterations of the system, but not supporting, out of the box, outsourcing to an external server.

1.2 Our Contribution

Using a similar idea to that described in (Bethencourt et al., 2007) for integer comparisons (using sub-trees in leaf nodes), we have constructed on top of (Goyal et al., 2006) a weighted KP-ABE system. However, this approach works just as good for CP-ABE.

Compared to other Weighted ABE schemes, our system uses a simpler mathematical construction, while having similar performance in terms of algorithms running time.

Our main goal is to show that this simple construction leads to an efficient and versatile weighted ABE system. When compared to existing weighted ABE system, our system will not be the most efficient, but it is not far off either. The theoretical analysis of our schemes compared to the existing ones shows that there not a big difference between them.

The main strength of our scheme is the simplicity of the construction, which opens the possibility of adding with ease new features to our scheme: access revocation, encryption/decryption outsourcing or decentralization.

Furthermore, we have shown that our decryption algorithm can be parallelized in order to make it faster. We have compared the parallelized version with the sequential one, in order to highlight the practical efficiency gain of this optimization.

2 Preliminaries

Notations and abbreviations

Notation	Meaning
Γ	An node in an access tree
k_Γ	threshold value for node Γ
σ_Γ	number of children of node Γ
W_A	weight of attribute A
k_Γ/σ_Γ	A "k out of σ" threshold gate
$attr(\Gamma)$	attribute corresponding to node Γ
ω_Γ	Minimum weight required for $attr(\Gamma)$
In_Γ	Set of input nodes for gate Γ
$\Delta_{i,S}(x)$	Lagrange coefficient: $\prod_{j\in S, j\neq i} \frac{x-j}{i-j}$

Bilinear maps (Goyal et al., 2006) 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.

Decisional Bilinear Diffie-Hellman Assumption Let $a,b,c,z \in \mathbb{Z}_p$ chosen randomly, and g a generator of G_1.

The decisional BDH Assumption (Sahai and Waters, 2005) is that no polynomial-time algorithm $\mathcal{B}$ can distinguish between $(A = g^a, B = g^b, C = g^c, eg, g^{abc})$ and $(A = g^a, B = g^b, C = g^c, eg, g^z)$ with a non-negligible advantage.

The advantage of $\mathcal{B}$ is:

$$|Pr[\mathcal{B}(A,B,C,e(g,g)^{abc})] - Pr[\mathcal{B}(A,B,C,e(g,g)^z)]|$$

where the probability is taken over the random choice of the generator g, the random choice of $a,b,c,z \in \mathbb{Z}_p$, and the random bits consumed by $\mathcal{B}$

Access Structures (Beimel, 2011) Let $p_1, \ldots, p_n$ be a set of parties. A collection $A \subseteq 2^{\{p_1,\ldots,p_n\}}$ is monotone if $B \in A$ and $B \subseteq C$ imply that $C \in A$. An access structure is a monotone collection $A \subseteq 2^{\{p_1,\ldots,p_n\}}$ of non-empty subsets of $\{p_1, \ldots, p_n\}$. Sets in A are called authorized, and sets not in A are called unauthorized.

Weighted Access Tree. A weighted access tree is a tree access structure where

each internal node Γ represents a threshold gate: it has an output wire (which leads to it's parent node in the tree), a number of input wires (σ_Γ) and a threshold value k_Γ, $1 \leq k_\Gamma \leq \sigma_\Gamma$. A node of such type is considered to be satisfied if at least k_Γ of it's σ_Γ children are satisfied.

For every leaf node Γ, there exist a corresponding attribute referred as $attr(\Gamma)$. These gates can be of two types:

- *boolean* - the node is satisfied if the corresponding attribute is present, and it is unsatisfied (evaluated with $\perp$) if the attribute is missing.
- *weighted* - the node has a minimum required weight ω_Γ attached to it. This gate receives as input an attribute $A = attr(\Gamma)$ with an integer weight attached W_A. The gate is satisfied if and only if $W_A \geq \omega_\Gamma$.

The weighted access tree is satisfied, if its root node is satisfied.

KP-ABE Model. A Key-Policy Attribute-Based Encryption scheme, as first described in (Goyal et al., 2006), consists of four algorithms:

setup(λ) A randomized algorithm that takes as input the implicit security parameter λ and return the public and secret keys (MPK and MSK).

encrypt(m, A, MPK) A probabilistic algorithm that encrypts a message m under a set of attributes A with the public key MPK, and outputs the ciphertext E.

keygen($\mathcal{C}, MPK, MSK$) This algorithm receives an access structure, public and master keys, and outputs corresponding decryption keys DK.

decrypt(E, DK, MPK) Given the ciphertext E and the decryption keys DK, the algorithm decrypts the ciphertext and outputs the original message.

Selective-Set Model for ABE. Goyal et al. propose in (Goyal et al., 2006) a Selective-Set Model for ABE. This security model also applies to our system, with the observation that the attributes can be weighted or binary, and the access structure has thresholds in the leaf nodes for the weighted attributes.

Init The adversary declares the set of attributes (weighted and binary) $\mathcal{A}$, that he wishes to be challenged upon.

Setup The challenger runs the Setup algorithm of ABE and gives the public parameters to the adversary.

Phase 1 The adversary is allowed to issue queries for private keys for many access structures A_j , where $\mathcal{A} \notin A_j$ for all j.

Challenge The adversary submits two equal length messages M_0 and M_1. The challenger flips a random coin b, and encrypts M_b with $\mathcal{A}$. The ciphertext is passed to the adversary.

Phase 2 Phase 1 is repeated.

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

3 Our Construction

We present a concrete KP-ABE construction for our system. We make use of an alteration of the access tree, similar to the one propose in (Bethencourt et al., 2007), in order to support integer comparisons. At each leaf node we incorporate a sub-tree of logarithmic size which simulates the comparison between the attribute weight and the required attribute threshold weight in the access structure.

The construction from (Bethencourt et al., 2007) presumes that for each attribute with values in $\{0 \cdots N\}$ we will have $2log_2(N)$ sub-attribute, two for each bit positions, covering the cases when each bit is either 0, or 1. Our proposal is to have sub-attribute only for the bits that are set to 1. In this way, we slightly reduce the number of attributes needed in the encryption phase: Instead of giving exactly $log(N)$ attributes, one for each bit of information.

In (Bethencourt et al., 2007), in order to model a weighted attribute A with weight $13 = (01101)$ in a system which supports a maximum weight of 31, 5 sub-attributes will

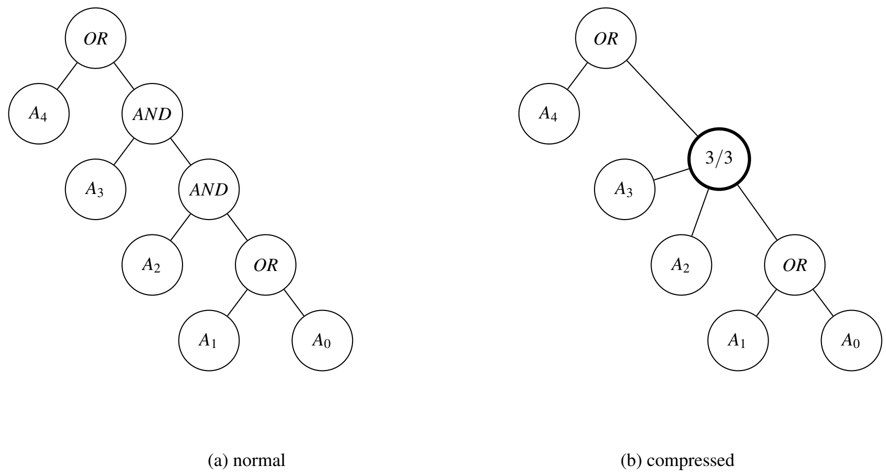


Fig. 1: sub-circuit for comparison "> 13" (13 = $(1101)_2$)

be needed: $A_{0****}, A_{*1***}, A_{**1**}, A_{***0*}, A_{****1}$. Our proposal consists in having sub-attributes only for the bits set to 1 in the weight's binary representation. Thus, in our model, for the weight 13 we will have sub-attributes: A_0, A_2, A_3, since $13 = 2^0 + 2^2 + 2^3$.

However, with this approach, we lose the possibility of creating other type of comparisons except "greater than" (">").

Since we want to check if the attribute's value is greater than the value ω_Γ required in the leaf node Γ, we process ω_Γ's bits $b_\ell \dots b_1 b_1$ in order to create the sub-tree. First, we eliminate the trailing (least significant) zero's from it's binary representation to obtain $\omega'_\Gamma = (b_\ell \dots b_{i+1} b_i)$ such that $b_i = 1$ and $b_{i-1} = \cdots = b_0 = 0$ (These bits are irrelevant when checking if some weight W_A, with $A = attr(\Gamma)$ is greater than ω_Γ). Then, for each bit b_j from the binary representation of ω'_Γ, excluding the last bit i, add a new gate to the system: if the bit is equal to 1, add an AND gate, otherwise add an *OR* gate. This new gate will have as parent the previous created gate (or will be connected to the original tree, if this is the first gate created) and two children:

- the leaf node for the sub-attribute A_j (corresponding to the j-th bit from the weight of attribute A)
- the next internal node (AND or OR gate) to be created.

At the end, create a new leaf node for attribute A_i, corresponding to bit i, and set its parent to the last created node.

For better understanding, an example sub-tree for the comparison ">13" is described in Figure 1 (a)

Comparison sub-tree optimization. We observe that our sub-tree for comparisons are formed out of chained OR and AND gates. Therefore, we can compress this sub-tree, grouping together similar gates:

- each k consecutive *OR* gates can be compressed in one "1 out of $k+1$" threshold gate.
- each k consecutive *AND* gates can be compressed in one "$k+1$ out of $k+1$" threshold gate.

This optimization can be seen in Figure 1 (b). The complete algorithm, including the above-mentioned optimization is described in detail in Algorithm 1.

Algorithm 1: transform($\mathcal{T}$)

```
ℓ_N ← log_2(N);
for every leaf node Γ in T corresponding to a weighted attribute do
    Let ω_Γ = (b_ℓ ··· b_1 b_0)_2 the minimum required weight ;
    Find i such that b_i = 1 and b_{i-1} = ··· = b_0 = 0 ;
    // Lest significant bit from ω_Γ set to 1
    Parent ← Γ ;
    // This is a temporary variable to store the last gate created
    for every j in {ℓ, ··· i+2, i+1} do
        Γ_j ← new leaf node ;
        if b_j = 1 then
            if b_j = b_{j+1} then
                k_Parent ← k_Parent + 1 // increases the threshold, as we will
                    add another child to this node, but we want it to
                    remain an AND node.
            else
                Tmp ← new (2/2)-gate (simple AND gate). ;
                parent(Tmp) ← Parent ;
                Parent ← Tmp ;
        else
            if b_j = b_{j+1} then
                continue ;
            else
                Tmp ← new (1/2)-gate (simple OR gate). ;
                parent(Tmp) ← Parent ;
                Parent ← Tmp ;
        parent(Γ_j) = Parent // Link the leaf node to the last node
            created
    parent(Γ_i) = Parent // Link the last leaf, corresponding to bit i, to
        the last node created
```

3.1 Weighted KP-ABE scheme

We describe further the construction of our Weighted KP-ABE scheme. We consider our attribute universe to be $\mathcal{U} = \{1, 2 \cdots M\}$, each attribute being either a Boolean or a numeric attribute. The numeric attributes can have a maximum value of N. Denote with $\ell = log_2(N)$ the number of bits required to describe these values.

setup(λ) This algorithm receives a security parameter λ, which is used to choose two multiplicative groups G_1 and G_2 of prime order p, g a generator of G_1, and a bilinear map $e : G_1 \times G_1 \rightarrow G_2$.
For each attribute, we have two cases, depending on the attribute type:
- If i it is a weighted attribute, then consider ℓ new sub-attributes: $i.0, i.1, \cdots i.\ell$. For each sub-attribute generate random $t_{i.j}$, $i \in \mathcal{U}, 1 \leq j \leq \ell$
- If i is a Boolean attribute, choose randomly t_i.

Next, choose random $y \in \mathbb{Z}_p$, and then set the public key as:

$$MPK = \langle p, G_1, G_2, e, g, n, Y = e(g,g)^y, T_\alpha = g^{t_\alpha} \rangle$$

and the master key:

$$MSK = \langle y, (t_\alpha) \rangle$$

Note that t_α can be of type t_i or $t_{i.j}$ depending on the attribute type.

encrypt$(m, \mathcal{A}, MPK)$ The encryption algorithm receives a message m, and encrypts it under the set of attributes $\mathcal{A} = \{(A, W_A) \mid A \in \mathcal{U},\ W_A < N\}$, with the public key MPK. Normal (Boolean) attributes, can be considered to have weight 0, or 1.
For each attribute A, it chooses the bits j set to 1 from it's weight W_A binary representation, and computes for them the values $T_{i.j}^s = g^{t_{i.j}s}$, where j is the index of the respective bit, and i the index of the attribute.
Then, generate a random element s, and compute the ciphertext as:

$$E = \langle A, E' = mY^s, T_{i.j}^s = g^{t_{i.j}s}, g^s \rangle, i \in \mathcal{U}, 1 \leq j \leq \ell_i$$

keygen$(MPK, \mathcal{T})$ We first need to modify the access tree $\mathcal{T}$ such that we include at the leaf nodes the sub-trees required to make the comparisons for the weighted attributes, using the function defined in Algorithm 1:

$$\mathcal{T}' = \text{transform}(\mathcal{T})$$

First, it generates a random y, and shares it through the tree, starting from the root node. For each node Γ which has a threshold of k_Γ, it generates a polynomial q_Γ of degree $k_\Gamma - 1$.
For the root node, it sets $q_{root} = y$, and then chooses $k_{root} - 1$ more points randomly to completely define the polynomial. For every internal node Γ, it sets $q_\Gamma(0) = q_{parent}(index(\Gamma))$ and then chooses $k_\Gamma - 1$ more points randomly. Finally, every leaf node Γ should receive a value $q_\Gamma(0)$, which is used to compute the key for the respective node:

$$D_\Gamma = g^{q_\Gamma(0)/t_x}$$

Note that x is of type $i.j$, it is a sub-attribute corresponding for bit j in attribute $A = attr(\Gamma)$.

decrypt(E, DK) This algorithm receives a valid ciphertext and a decryption key, and returns the original message. The simplest form of representation for the decryption algorithm is as an recursive procedure. Let $DecNode(E, D, \Gamma)$ be this algorithm, applied to node Γ with ciphertext E, and decryption key D. For every leaf node:

$$DecNode(E, D, \Gamma) = \begin{cases} e(D_\Gamma, T_x^s) = e(g,g)^{q_\Gamma(0) \cdot s}, \\ \qquad \text{if } x = attr(\Gamma) \in \mathcal{A} \\ \perp, \\ \qquad \text{otherwise} \end{cases}$$

For the recursive case, we will consider an internal node Γ with threshold k_x. Consider the children z of this node such that $DecNode(E, D, z) \neq \perp$. If the number of such nodes is smaller than k_Γ, then return $\perp$, as there is insufficient data to recompute the polynomial. Otherwise, compute the value:

$$\begin{aligned} & DecNode(E, D, \Gamma) = \\ & \quad = \prod_{z \in In_\Gamma} DecNode(E, D, z)^{\Delta_{i, In'_\Gamma}(0)} \\ & \text{where } i = index(z),\ \ In'_\Gamma = \{index(z) | z \in In_\Gamma\} \\ & \quad = \prod_{z \in In_\Gamma} \left(e(g,g)^{s \cdot q_z(0)}\right)^{\Delta_{i, In'_\Gamma}(0)} \\ & \quad = \prod_{z \in In_\Gamma} \left(e(g,g)^{s \cdot q_{parent(z)}(0)}\right)^{\Delta_{i, In'_\Gamma}(0)} \\ & \quad = \prod_{z \in In_\Gamma} \left(e(g,g)^{s \cdot q_x(0)}\right)^{\Delta_{i, In'_\Gamma}(0)} \\ & \quad = e(g,g)^{s \cdot q_\Gamma(0)} \end{aligned}$$

Calling the function on the root of the tree, we obtain:

$$\begin{aligned} R = DecNode(E, D, root) = e(g,g)^{s \cdot q_{root}(0)} \\ = e(g,g)^{ys} \end{aligned}$$

Finally, we can recover the message by computing:

$$m = E'/R = m \cdot e(g,g)^{ys} / e(g,g)^{ys}$$

3.2 Security & Extensions

Our system is, actually, an instance of Goyal's KP-ABE system (Goyal et al., 2006) with some attribute relabeling. The only concrete change is in the structure of the access tree. Therefore, it inherits the latter's security properties. If an attacker would have a non-negligible advantage against our scheme, then an attacker with non-negligible advantage against (Goyal et al., 2006) would also exist. Any access tree with comparison sub-trees in the leaf nodes is also a valid input for Goyal's KP-ABE system (Goyal et al., 2006). (We can simply relabel the sub-attributes of form $i.j$ to a single integer $\alpha_{i.j}$).

Since Goyal's KP-ABE system (Goyal et al., 2006) is secure in the Selective Set Model for ABE, under the decisional Bilinear Diffie-Hellman Problem, this also proves that our system is secure in the Selective Set Model for ABE, under the same hardness assumption.

Theorem 1. *The Weighted KP-ABE system is secure in the Key-Policy Attribute-based Selective-Set Model under the bilinear Decisional Diffie-Hellman problem.*

Proof. A formal proof is provided in Appendix.

OR and AND gates optimization. In most previous ABE system, *OR* and *AND* gates are simply treated as general threshold gates, k out of n: For *OR* gate $k = 1$, and for *AND*, $k = n$. Thus, the secret sharing for these gates is realized in the same manner as for regular threshold gates, by using Shamir's secret sharing technique (Shamir, 1979). While applied to ABE systems, this requires the computation of expensive exponentiations in G_T during the reconstruction phase. More exactly, the *OR* gate requires one exponentiation, but the *AND* gate requires n exponentiations. Therefore, we will use in our system a more efficient method of secret sharing though *AND* and *OR* gates, which was proposed by Tiplea-Dragan in (Ţiplea and Drăgan, 2014). Their method works as follows:

– *OR* gates: For this gate, simply forward the value received at the output node to all children, as each of them should be able to decrypt using it's own secret.
– *AND* gates: For this gate, generate for each child node a secret value, such that the sum of those values equal the value from the output wire of the *AND* gate.

Since our scheme uses many *AND* and *OR* gates for the comparison sub-trees, this optimization should have a noticeable effect on the running time of our scheme. Concrete test results can be seen in Section 5

This secret sharing method for *OR* and *AND* gates has been proven to be secure and successfully used in previous ABE constructions, such as (Ţiplea and Drăgan, 2014).

Parallelized decryption During the decryption phase, we can observe that the sub-trees referring to attribute comparisons are independent one of each other. This means that the decryption can be done simultaneously on these parts of the access structure, by creating a new thread for each sub-tree. When the execution of the sub-threads is finished, the algorithm may resume and compute the reconstruction of the secret on the rest of the tree.

A graphical representation of the parallelization computation over the comparison sub-trees can be seen in Figure 2: For each sub-tree corresponding to a weighted attribute we create a new thread which reconstructs the secret for that sub-tree.

Outsourced parallelized decryption Since every comparison sub-structure can be see as an access tree by itself, we can consider that we have $p+1$ distinct access trees, for which we can outsource the decryption (in parallel) of the first p, and then join the rest with the tree. However, this would require a more complex solution, in order to securely outsource the decryption on the respective nodes, on untrusted cloud servers.

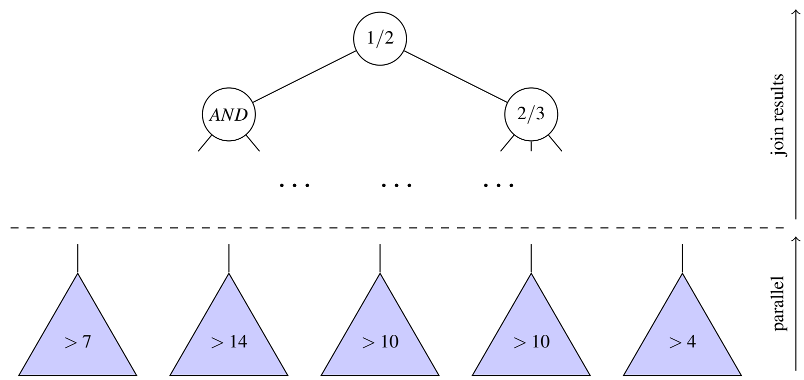


Fig. 2: Parallel decryption

3.3 Other Extensions

The tree transformation method can be applied to any CP-ABE or KP-ABE scheme that has an access tree as policy. Therefore, many existing systems can be extended to support weighted attributes alongside other features, such as: encryption and decryption outsourcing (Asim et al., 2014), multi-authority ABE (Chase, 2007), revocation in a multi-authority system (Qian et al., 2015).

Our proposed alteration for access trees can also be made to Boolean circuits, in order to add support for weighted attributes, one example of such scheme being (Ţiplea and Drăgan, 2014) or (Hu and Gao, 2017). The idea is the same as for access trees: Replacing terminal nodes with small sub-circuits for comparisons.

4 Theoretical analysis

We stress that our construction could easily be applied to any CP-ABE supporting access trees. We will thus consider in our comparisons Bethencourt et al.'s - BSW(Bethencourt et al., 2007) with our construction for weighted attributes.

Key-Policy Schemes The only weighted KP-ABE system we have identified is the one in (Liu et al., 2014), which has liner expansion in key and encryption/decryption time per attribute. Some of the CP-ABE systems proposed do have a KP variant, but we have compared them with our variant of CP-ABE.

Ciphertext-Policy Schemes Although we have not given a proper definition for a CP-ABE system, it is easy to observe that the tree transformation algorithm can be applied to the Bethencourt et al.'s system (Bethencourt et al., 2007).

We can also modify the scheme from (Hu and Gao, 2017) in order to obtain a weighted CP-ABE system for Boolean circuits, but not efficient enough to be used in practice. However, if we consider the subset of Boolean circuits representing Boolean

Table 1: Notations used for theoretical analysis

Notation	Meaning
N	Bit-length of attribute weights
S_u	Number of attributes in user set.
S_{u_m}	Number of weighted attributes in user set.
S_{u_n}	Number of Boolean attributes in user set.
S_t	Number of attributes in access policy.
S_{t_m}	Number of weighted attributes in AP.
S_{t_n}	Number of Boolean attributes in AP.
E_p	cost of a pairing operation
E_{G_1}	One exponentiation in G_1
E_{G_T}	One exponentiation in G_T
T	Number of interior nodes satisfying the AP
$hw(x)$	Hamming weight

formulas (which can be represented as access trees), we can slightly reduce the complexity compared to the variant in which we used Bethencourt et al.'s system by a constant factor of 2.

This is due to the fact that (Hu and Gao, 2017), when limited to Boolean formulae, offers a CP-ABE system for access trees, which is more efficient than (Bethencourt et al., 2007).

As shown from the theoretical and experimental analysis from (Li et al., 2021), we can observe that the best weighted CP-ABE systems by a considerable margin are LYL+ (Li et al., 2021) and CABE (Xue et al., 2017). These systems are also the only ones with logarithmic expansion per weighted attribute. Therefore, we will compare our two variants which rely on BSW(Bethencourt et al., 2007) and HG(Hu and Gao, 2017) with these two weighted CP-ABE schemes (CABE(Xue et al., 2017) and LYL+(Li et al., 2021)).

In Table 2 we have listed the theoretical cost of key generation encryption and decryption of the systems described above. The only algorithm with notable theoretical difference is the decryption algorithm. For the key generation and encryption algorithm, our scheme has a similar computational overhead compared to CABE and LYL+.

The key generation and decryption algorithm add a computational overhead of $hw(N)$ per attribute to our scheme, where $hw(x)$ is the Hamming weight (the number of ones in the binary representation) of x. This is slightly better than $log(N)$ in the average case, but in worst case, still logarithmic.

5 Experimental results

The single weighted KP-ABE scheme that we have found, is clearly more slow compared to our solution, as it can be seen from the theoretical analysis, excluding the need of an experimental comparison between the two of them.

Table 2: Theoretical analysis of weighted CP-ABE schemes

Scheme	KeyGeneration
Ours (BSW(Bethencourt et al., 2007) variant)	$[2(hw(N)\cdot S_{u_m}+S_{u_n})+1]E_{G_1}$
Ours (HG (Hu and Gao, 2017) variant)	$(hw(N)\cdot S_{u_m}+S_{u_n}+1)E_{G_1}$
CABE(Xue et al., 2017)	$[2(log_2(N)\cdot S_{u_m}+S_{u_n})+1]E_{G_1}$
LYL+ (Li et al., 2021)	$(log_2(N)\cdot S_{u_m}+S_{u_n}+2)E_{G_1}$
Scheme	**Encryption**
Ours (BSW(Bethencourt et al., 2007) variant)	$[2log_2(N)\cdot(S_{t_m}+S_{t_n})+1]E_{G_1}+E_{G_T}$
Ours (HG (Hu and Gao, 2017) variant)	$[log_2(N)\cdot(S_{t_m}+S_{t_n})+1]E_{G_1}+E_{G_T}$
CABE(Xue et al., 2017)	$(log_2(N)\cdot S_{t_m}+2S_{t_n}+1)E_{G_1}+E_{G_T}$
LYL+ (Li et al., 2021)	$((log_2(N)+2)\cdot S_{t_m}+2S_{t_n}+1)E_{G_1}+E_{G_T}$
Scheme	**Decryption**
Ours (BSW(Bethencourt et al., 2007) variant)	$2(log_2(N)\cdot S_{u_m}+S_{u_n})E_p+T\cdot E_{G_T}$
Ours (HG(Hu and Gao, 2017) variant)	$(log_2(N)\cdot S_{u_m}+S_{u_n})E_p+T\cdot E_{G_T}$
CABE(Xue et al., 2017)	$(2S_u+1)E_p+TE_{G_T}$
LYL+ (Li et al., 2021)	$(2S_u+1)E_p+TE_{G_T}$

However, we do want to find out how useful is parallel decryption and the optimization of *AND* and *OR* gates in practice. We have thus implemented and ran performance tests over three variants of our scheme:

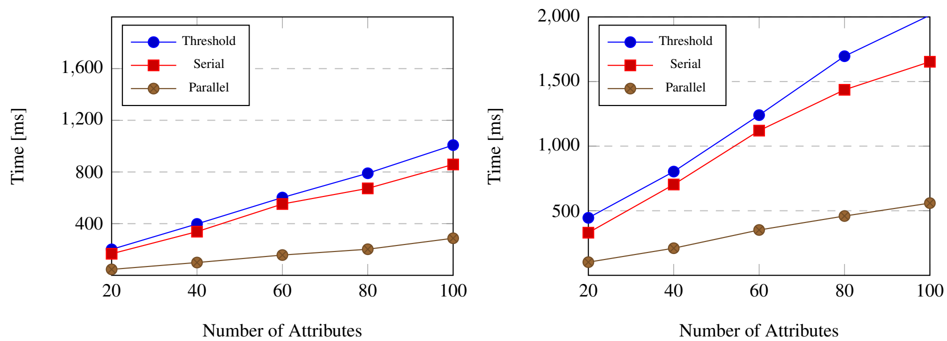
- "Threshold": this variant uses threshold gates instead of *AND* and *OR* gates. No parallelization is present in this implementation
- "Serial": this variant uses improved secret sharing through *AND* and *OR* gates, but still no parallelization was performed.
- "Parallel": this variant computes in parallel the secret over each comparison subtree.

The implementation was made in C++, using the *Pairing Based Cryptography* Library (Lynn, 2010), and the tests were performed under a Debian 10 system, with 16GB of RAM and an Intel Core i7-3630QM Processor.

We have divided our tests in two scenarios, depending on the attribute weight dimension: 8 bits and 16 bits. On each type, we have tested our system against an access structure with variable number of weighted attribute, ranging from 20 to 100. Each attribute was split, according to our scheme description, in 8 or 16 sub-attributes, depending on the chosen scenario. The access tree was formed mostly by AND gates, and the threshold weight from the leaf nodes was the maximum possible - it was requiring 2^8-1 (and $2^{16}-1$ for the 16 bit variant) weight for each attribute. For each weighted attribute, our program created a new thread which computed the result of the sub-tree corresponding to that attribute.

Our results can be see in Figure 3: The decryption algorithm works in less than a second up to 40-50 attributes of 16 bits for the normal version, while the parallel version takes roughly 500 milliseconds for 100 weighted attributes.

In both scenarios, we can see that the decryption time for the parallelized algorithm is roughly $\frac{1}{3}$ of the normal version, while the *AND* and *OR* gates optimizations also offer some smaller improvements in running time.



(a) 8 bits (b) 16 bits

Fig. 3: Performance tests

6 Conclusions

While this approach is most likely not the most efficient for Weighted ABE systems, it is not far away from the best existing solution in terms of efficiency.

However, our variant provides a more simpler mathematical construction, which lead to more versatility, inheriting all the properties of the emblematic KP-ABE (Goyal et al., 2006) and CP-ABE (Bethencourt et al., 2007) systems: security, fast secret-sharing for *OR* and *AND* gates, and various extensions, such as: access revocation, outsourcing and multi-authority.

On top of that, this weighted ABE system proves to be very suitable for parallelized decryption, in order to make it more efficient: It is both easy to implement and offers great practical time benefit, without any mathematical alteration of the system.

The performance tests show that this simple approach is suitable for practical use. While for the normal version we could use access policies up to 40-50 attributes, for the parallel one, this number will greatly increase to around 100.

Bibliography

[Asim et al., 2014]Asim, M., Petkovic, M., and Ignatenko, T. (2014). Attribute-based encryption with encryption and decryption outsourcing.

[Attrapadung et al., 2018]Attrapadung, N., Hanaoka, G., Ogawa, K., Ohtake, G., Watanabe, H., and Yamada, S. (2018). Attribute-based encryption for range attributes. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 101(9):1440–1455.

[Beimel, 2011]Beimel, A. (2011). Secret-sharing schemes: a survey. In *International conference on coding and cryptology*, pages 11–46. Springer.

[Bethencourt et al., 2007]Bethencourt, J., Sahai, A., and Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)*, pages 321–334. IEEE.

[Chase, 2007]Chase, M. (2007). Multi-authority attribute based encryption. In *Theory of Cryptography Conference*, pages 515–534. Springer.

[Feng et al., 2020]Feng, C., Yu, K., Aloqaily, M., Alazab, M., Lv, Z., and Mumtaz, S. (2020). Attribute-based encryption with parallel outsourced decryption for edge intelligent iov. *IEEE Transactions on Vehicular Technology*, 69(11):13784–13795.

[Gay et al., 2015]Gay, R., Méaux, P., and Wee, H. (2015). Predicate encryption for multi-dimensional range queries from lattices. In *IACR International Workshop on Public Key Cryptography*, pages 752–776. Springer.

[Goyal et al., 2006]Goyal, V., Pandey, O., Sahai, A., and Waters, B. (2006). Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98.

[Hu and Gao, 2017]Hu, P. and Gao, H. (2017). Ciphertext-policy attribute-based encryption for general circuits from bilinear maps. *Wuhan University Journal of Natural Sciences*, 22(2):171–177.

[Li et al., 2021]Li, H., Yu, K., Liu, B., Feng, C., Qin, Z., and Srivastava, G. (2021). An efficient ciphertext-policy weighted attribute-based encryption for the internet of health things. *IEEE Journal of Biomedical and Health Informatics*.

[Liu et al., 2014]Liu, X., Zhu, H., Ma, J., Ma, J., and Ma, S. (2014). Key-policy weighted attribute based encryption for fine-grained access control. In *2014 IEEE International Conference on Communications Workshops (ICC)*, pages 694–699. IEEE.

[Lynn, 2010]Lynn, B. (2010). The pairing-based cryptography (pbc) library. https://crypto.stanford.edu/pbc/.

[Ostrovsky et al., 2007]Ostrovsky, R., Sahai, A., and Waters, B. (2007). Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 195–203.

[Qian et al., 2015]Qian, H., Li, J., Zhang, Y., and Han, J. (2015). Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation. *International Journal of Information Security*, 14(6):487–497.

[Sahai and Waters, 2005]Sahai, A. and Waters, B. (2005). Fuzzy identity-based encryption. In *EuroCrypt*, pages 457–473. Springer.

[Shamir, 1979]Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22(11):612–613.

[Shamir, 1984]Shamir, A. (1984). Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques*, pages 47–53. Springer.

[Ţiplea and Drăgan, 2014]Ţiplea, F. L. and Drăgan, C. C. (2014). Key-policy attribute-based encryption for boolean circuits from bilinear maps. In *BalkanCryptSec*, pages 175–193. Springer.

[Tiplea et al., 2020]Tiplea, F. L., Ionita, A., and Nica, A.-M. (2020). Practically efficient attribute-based encryption for compartmented access structures. In *ICETE (2)*, pages 201–212.

[Wang et al., 2016]Wang, S., Liang, K., Liu, J. K., Chen, J., Yu, J., and Xie, W. (2016). Attribute-based data sharing scheme revisited in cloud computing. *IEEE Transactions on Information Forensics and Security*, 11(8):1661–1673.

[Waters, 2011]Waters, B. (2011). Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *International Workshop on Public Key Cryptography*, pages 53–70. Springer.

[Xue et al., 2017]Xue, K., Hong, J., Xue, Y., Wei, D. S., Yu, N., and Hong, P. (2017). Cabe: A new comparable attribute-based encryption construction with 0-encoding and 1-encoding. *IEEE Transactions on Computers*, 66(9):1491–1503.

Appendix

Theorem 1. *The Weighted KP-ABE system is secure in the Key-Policy Attribute-based Selective-Set Model under the bilinear Decisional Diffie-Hellman problem.*

Proof. First, denote with $W-KP-ABE$ our scheme, and with $GPSW$ Goyal's KP-ABE (Goyal et al., 2006).

Then, we will show that if there exists a non-negligible advantage adversary for $W-KP-ABE$, then we can also construct an adversary with non-negligible advantage for $GPSW$.

Suppose there exists an adversary $\mathcal{A}$ with non-negligible advantage against $KP-W-ABE$. Then, construct an adversary $\mathcal{A}'$, using $\mathcal{A}$ as challenger for $KP-W-ABE$.

Setup. The challenger Ch runs the Setup algorithm and gives the public parameters, mpk to $\mathcal{A}'$. Then, $\mathcal{A}'$ forwards them to $\mathcal{A}$.
In the next steps we will need more attributes in the $GPSW$ scheme. Therefore we will create, for each weighted attribute A_i from $W-KP-ABE$, $\ell = log(N)$ corresponding attributes in $GPSW$: A_{i_1}, A_{i_ℓ}.

Phase 1. $\mathcal{A}$ makes repeated decryption keys inquiries for the sets of (possibly weighted) attributes $S_1, ..., S_{q_1}$. For each set of attributes S_i, $\mathcal{A}'$ generates a valid answer, by querying Ch with the corresponding set of attributes from $KP-W-ABE$: For each weighted A_i $\mathcal{A}'$ will require from the challenger $\mathcal{C}$ decryption keys corresponding to A_{i_1}, A_{i_ℓ}.

The decryption keys for A_j will have the form $D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j}$, we will simply hide from $\mathcal{A}_i$ the decryption keys for the newly added attributes ($D_{j'}$ and $D'_{j'}$ for every j')

Then respond to $\mathcal{A}$ by simply forwarding the decryption keys.

Challenge. $\mathcal{A}$ submits two equal length messages M_0 and M_1. In addition $\mathcal{A}$ gives a challenge access structure $\mathcal{C}$ (a weighted access tree) such that none of the sets of attributes $S_1, ..., S_{q_1}$ from *Phase 1* satisfy the access structure. $\mathcal{A}'$ will transform the access structure using the transformation algorithm from Algorithm 1, into a valid one for $GPSW$: $\mathcal{C}^*$ will be a simple access tree.

Then, $\mathcal{C}^*$ will be sent to the challenger along with the message M for encryption. Ch flips a random coin b, and encrypts M_b under the new access structure. The ciphertext CT is given to $\mathcal{A}'$.

$\mathcal{A}'$ then simply forwards the ciphertext to $\mathcal{A}$.

Phase 2. *Phase 1* is repeated with the restriction that none of sets of attributes $S_{q_1+1}, ..., S_q$ satisfy the access structure corresponding to the challenge.

Guess. $\mathcal{A}$ outputs a guess b' of b, which is then forwarded by $\mathcal{A}'$ to Ch.

It is clearly that the advantage of $\mathcal{A}$ against $KP-W-ABE$ is the same as the advantage of $\mathcal{A}'$ against $GPSW$.