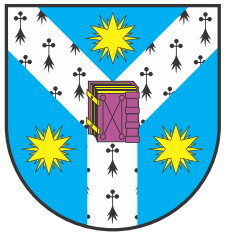
A Hybrid KP-ABE Scheme for Universal Circuits, Based on Monotone Span Programs

Iulian Oleniuc

Coord. Dr. Andrei Arusoaie



Submitted for the degree of Master in Computer Science

Alexandru Ioan Cuza University of Iași

July 2025

Abstract

Many key-policy attribute-based encryption (KP-ABE) schemes, built on top of bilinear pairings, have been developed, specifying access policies by using both trees and circuits, and various kinds of gates, such as `AND`, `OR`, and (t/n)-threshold gates. We take these contributions to the next level, and, building on top of the Hu–Gao scheme for "general circuits" and the Goyal et al. scheme for monotone span programs, we propose the pairing-based KP-ABE scheme with, we argue, the highest degree of generalization. That is, by using our concept of "universal circuits," each internal node of the circuit can compute *any* kind of monotone access structure, by embedding a specific MSP. We argue that our scheme is more efficient than the Hu–Gao scheme for "general circuits" and the Goyal et al. scheme for "access trees." Our research also provides a new monotone span program lower bound, a novel way of proving the nonexistence of ideal linear secret sharing schemes for certain monotone access structures, and a backtracking approach for optimizing a KP-ABE scheme for Boolean circuits.

Contents

Chapter 1

Introduction

Cloud computing has become the backbone of modern enterprise software, enabling flexible, on-demand access to storage and computational resources. Organizations increasingly rely on cloud services for collaborative workflows and centralized data management, reducing infrastructure costs and improving scalability. However, this shift introduces significant privacy concerns. In typical cloud setups, service providers — and sometimes all users within an organization — may have unrestricted access to sensitive data. This lack of fine-grained control over who can access what information poses a serious threat to data confidentiality.

Attribute-based encryption (ABE) addresses this issue by embedding access control directly into the encryption process. In particular, key-policy attribute-based encryption (KP-ABE) allows data to be encrypted under a set of descriptive attributes, while user decryption keys are associated with access policies. A user can decrypt a ciphertext only if the attributes attached to it satisfy the policy in their key. This approach enables rich, fine-grained access control based on flexible relationships between attributes, offering a powerful tool for enforcing privacy in collaborative, cloud-based environments.

KP-ABE schemes can be highly beneficial in the medical system, by enabling fine-grained access control over sensitive patient data. With KP-ABE, medical records can be encrypted with a set of descriptive attributes (e.g., "cardiology," "emergency," or "pediatrics"), while access keys are issued to healthcare professionals based on specific policies, defining which combinations of attributes they are permitted to access. This ensures that only authorized personnel (e.g., doctors, nurses, or researchers with matching policy keys) can decrypt and view relevant data, enhancing privacy, regulatory compliance, and operational efficiency.

1.1 Motivation

Many KP-ABE schemes have already been developed, based on various cryptographic primitives, such as lattices [1], quadratic residues [2], and bilinear pairings [3, 4, 5].

Now, after choosing a certain primitive, one would want to build schemes with the ability of specifying access policies in more and more compact ways. For example, if there is a scheme running on trees of a certain kind, it would be a great improvement

to make it work on *circuits* of said kind, because, in a circuit, the same node can be an input to multiple nodes.

Moreover, the computational models used for specifying access policies do not only need to be more compact, but also more expressive. That is, the gates they use have to be more general (i.e., finer-grained). The trend of improving the gates is clear: from `AND`/`OR`-gates [5], we moved to (t/n)-threshold gates [3], to "CAS-nodes" [6], and so on.

This too helps in reducing the size of the access policies. Specifically, entire parts of the old tree/circuit may be replaced by only one gate. Section 5.2.3 provides a clear example in this sense, where we introduce the "`U` gate," which models the Boolean formula $((A \wedge B) \vee (B \wedge C) \vee (C \wedge D))$. Hence, the 4 double-circle nodes in Fig. 5 can be replaced by only 1 double-circle node, as in Fig. 4. This optimization does not only reduce the circuit size, but also the decryption key size, and, overall, increases the efficiency of the KP-ABE scheme.

Besides finding more compact computational models, we are generally interested in studying circuit schemes, because all state-of-the-art schemes [5, 4] for monotone Boolean circuits (MBCs) yield exponentially large keys, in order not to sacrifice the scheme security. What is more, most existing results regarding ABE, linear secret sharing schemes (LSSSes), and monotone span programs (MSPs) are not clearly connected in the literature, so we were interested in finding these connexions as well.

1.2 Our Contribution

First, we have gently explored the realm of attribute-based encryption, linear secret sharing, and monotone span programs, while systematizing all the relevant results in this research topics and proving a new lower bound.

We have introduced the concept of `U`-gates, while also providing a novel way of proving the nonexistence of ideal linear secret sharing schemes for certain monotone access structures, which was applied specifically on `U`-gates. We have shown the connection between `U`-gates and graph access structures.

We systematized the four main ways of constructing attribute-based encryption schemes from monotone Boolean circuits, while also trying to optimize one of them using backtracking.

That being said, our most important result is definitely the hybrid key-policy attribute-based encryption scheme for circuit access policies, based on bilinear pairings, with, we argue, the highest degree of generalization. That is, by using "universal circuits," each internal node of the circuit can compute *any* kind of monotone access structure (thanks to monotone span programs), thus not being limited anymore to specific, narrow cases, such as `AND`, `OR`, and (t/n)-threshold gates.

Chapter 2

Preliminaries

This chapter presents the cryptographic primitive of bilinear pairings (which are the basis of many ABE schemes), the paradigm of attribute-based encryption (which can be either ciphertext-policy or key-policy), the foundations of linear secret sharing (which is used in virtually every ABE scheme), and the basic computational models for specifying access policies (such as monotone Boolean trees and monotone Boolean circuits).

2.1 Bilinear Pairings

Definition 1 (Bilinear Pairing [7])**.** *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p, and let g be a generator of $\mathbb{G}_1$. A function $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ satisfying all the following conditions is called a* bilinear pairing*:*

1. $e(x^a, y^b) = e(x, y)^{ab}, \forall x, y \in \mathbb{G}_1, \forall a, b \in \mathbb{Z}_p$; *(Bilinearity)*
2. $e(g, g) \neq 1$; *(Non-Degeneracy)*
3. *both e and the group operation are efficiently computable.* *(Computability)*

Bilinear pairings are usually implemented using elliptic curve groups. The most well-known bilinear pairing constructions are the Weil [8] and Tate [9] pairings.

Definition 2 (DBDH [10])**.** *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p, and let g be a generator of $\mathbb{G}_1$. The* Decisional Bilinear Diffie–Hellman (DBDH) *assumption states that, given g, g^a, g^b, g^c, $e(g, g)^{abc}$, and $e(g, g)^z$, with a, b, c, and z randomly chosen from $\mathbb{Z}_p$, there is no efficient algorithm that can distinguish $e(g, g)^{abc}$ from $e(g, g)^z$.*

2.2 Attribute-Based Encryption

Attribute-based encryption (ABE) [10] is a form of public-key encryption that enables fine-grained access control over encrypted data. In ABE, both users and data are associated with sets of attributes, and decryption is possible only when a specified relationship between these attributes is satisfied.

There are two main variants: in ciphertext-policy ABE (CP-ABE), the encryptor defines an access policy, and the decryption key is tied to a set of attributes; in key-policy ABE (KP-ABE), the ciphertext is labeled with attributes, and the decryption key encodes the access policy.

Example 1 (Medical Institution)**.** *This work centers around KP-ABE. To illustrate its applicability, consider a practical use case in the healthcare sector. Assume a hospital maintains numerous records containing patient information. Each document is tagged with specific attributes such as department* $\in \{\texttt{cardio}, \texttt{neuro}, \texttt{ortho}\}$*, sensitivity* $\in \{\texttt{public}, \texttt{confidential}\}$*, and year* $\in \{1917, \ldots, 1953\}$*. A central authority may issue a decryption key governed by the access policy* $((\textit{department} \in \{\texttt{cardio}, \texttt{ortho}\}) \wedge \textit{sensitivity} \neq \texttt{private} \wedge (1922 \leq \textit{year} \leq 1924))$*. With this key, an authorized party could decrypt a file characterized by the attribute values* $\textit{department} = \texttt{ortho}$*,* $\textit{sensitivity} = \texttt{public}$*, and* $\textit{year} = 1923$*. It is important to observe that these attributes might be tied to users (e.g., department) or to the data itself (e.g., sensitivity and year).*

Any KP-ABE scheme defines four main procedures: Setup (which returns the public key and the master key), Encrypt (which, based on the public key and some attribute set, encrypts the given plaintext), KeyGen (which, based on the public key and the master key, generates a decryption key for the given access policy), and Decrypt (which, based on the decryption key, decrypts the given ciphertext if its associated attributes satisfy the access policy embedded in the key; otherwise, it returns $\perp$).

2.3 Linear Secret Sharing

Secret sharing is a field that received a tremendous amount of attention in the last half a century. Secret sharing [11] refers to distributing a *secret* among a group of parties $\mathcal{P}$ in such a way that only certain *authorized* subsets of parties are able to reconstruct the secret based on the *shares* they receive.

Threshold secret sharing is probably the most studied variation, for which many ideal schemes have already been achieved. They are based on Lagrange interpolation [11], linear algebra [12], the Chinese remainder theorem [13, 14], among others.

Definition 3 (MAS [15])**.** *A collection* $\mathbb{A} \subseteq 2^{\mathcal{P}}$ *over a set of parties* $\mathcal{P}$ *is called a* monotone access structure (MAS) *if* $\mathcal{X} \in \mathbb{A} \wedge \mathcal{X} \subseteq \mathcal{Y} \rightarrow \mathcal{Y} \in \mathbb{A}$*.*

In other words, the monotonicity of an access structure consists in the fact that every superset of an authorized subset is also authorized. Intuitively, there are no secret sharing schemes realizing non-monotone access structures. It is natural that, by adding more parties to an already authorized subset, they will not break its already achieved ability to reconstruct the secret.

Example 2. *For* $\mathcal{P} = \{A, B, C\}$*, one possible MAS is*

$$\mathbb{A} = \{\{A\}, \{B, C\}, \{A, B\}, \{A, C\}, \{A, B, C\}\}.$$

For instance, $\{A, C\} \in \mathbb{A}$ *and* $\{A, C\} \subseteq \{A, B, C\}$ *lead to* $\{A, B, C\} \in \mathbb{A}$*.*

Definition 4 (MSP [15])**.** *A* monotone span program (MSP) *over a finite field K is a labeled matrix $\hat{M}(M, \rho)$, where M is an $m \times d$ matrix over K, and ρ is a labeling of the rows of M by parties in $\mathcal{P}$.*

It is said that $\hat{M}$ computes *an MAS $\mathbb{A}$ if $\vec{1} \in \text{span}(M_{\mathcal{X}}) \leftrightarrow \mathcal{X} \in \mathbb{A}$, where $\vec{1} = (1, \ldots, 1)$ is called the* objective *vector, and $M_{\mathcal{X}}$ denotes the submatrix of M consisting exclusively of the rows labeled by parties in $\mathcal{X}$.*

The size *of $\hat{M}$ is defined simply as m.*

The objective vector can be replaced by any other non-zero vector via a change of basis. Many times, $(1, 0, \ldots, 0)$ is used instead.

Example 3. *The following MSP $\hat{M}$ computes $\mathbb{A} = \{\{A, B\}, \{B, C\}, \{A, B, C\}\}$, for $\vec{1} = (1, 0, 0)$:*

$$\hat{M} = \left[\begin{array}{c|ccc} A & 1 & 1 & 0 \\ B & 0 & -1 & 0 \\ B & 1 & 0 & 1 \\ C & 0 & 0 & -1 \end{array}\right].$$

i) For $\{A, B\} \in \mathbb{A}$, we can state that $\langle M_1, M_2, M_3\rangle$ do indeed span $\vec{1}$, since $1 \cdot M_1 + 1 \cdot M_2 = (1, 0, 0)$.

ii) For $\{B\} \notin \mathbb{A}$, we can state that $\langle M_2, M_3\rangle$ do not span $\vec{1}$, since the coefficient of M_3 must be 1 to achieve 1 on the first coordinate of the result, but we also need to cancel the 1 it adds on the third coordinate, which is not possible.

Theorem 1 ([16])**.** *It is always possible to restrict the matrix of an MSP to a set of linearly independent columns without changing the MAS computed by the program. Therefore, it is not necessary to use more columns than rows.*

Hence the above definition of size. Taking into account the number of columns in an MSP would only quadratically increase its size.

Definition 5 (LSSS [15])**.** *Let K be a finite field, and let Π be a secret sharing scheme with domain of secrets $S \subseteq K$* realizing *an MAS $\mathbb{A}$. It is said that Π is a* linear secret sharing scheme (LSSS) *over K if it satisfies the following conditions:*

1. *The share of each party is a vector over K. That is, for every i, there exists a constant d_i such that the share of P_i is taken from K^{d_i}. We denote by $\Pi_{i,j}(s, r)$ the j-th coordinate in the share of P_i, where $s \in S$ is a secret, and $r \in R$ is the random input given by the dealer.*

2. *For every authorized subset, the reconstruction function of the secret from the shares is linear. That is, for every $\mathcal{X} \in \mathbb{A}$, there exist constants $\{\alpha_{i,j} \mid P_i \in \mathcal{X}, 1 \leq j \leq d_i\}$ such that, for every secret $s \in S$ and every choice of random input $r \in R$,*

$$s = \sum_{P_i \in \mathcal{X}} \sum_{1 \leq j \leq d_i} \alpha_{i,j} \cdot \Pi_{i,j}(s, r).$$

The size *of Π is defined as $d = \sum_{i=1}^{n} d_i$.*
The scheme is said to be ideal *if $d = |\mathcal{P}|$.*

Theorem 2 ([17]). *Let $\mathbb{A}$ be an MAS, and let K be a finite field. There exists an LSSS of size d over K realizing $\mathbb{A}$ if and only if there exists an MSP of size d over K computing $\mathbb{A}$.*

The proof of this theorem provides an algorithm for constructing LSSSes from MSPs.

2.4 Computational Models

In an ABE scheme, each ciphertext (in the case of CP-ABE) or each decryption key (in the case of KP-ABE) is bound to an access policy, which can be viewed as a program computing a specific MAS. In a KP-ABE scheme, a key can decrypt only those ciphertexts whose associated attributes form an authorized subset under that MAS.

These access policies can be expressed through various computational models. The most common approach is the monotone Boolean tree, where internal nodes represent logical operations (either `AND` or `OR`), and leaves correspond to individual attributes. Note that `AND` and `OR`-gates are computational models in themselves. More expressive models, such as monotone Boolean *circuits* and the previously introduced MSPs, enable even richer and more flexible policy definitions.

Definition 6 (MBC). *A* monotone Boolean circuit (MBC) *is a directed acyclic graph (DAG) such that, for each node v in the graph, either v has in-degree 0 and is labeled with a party $X \in \mathcal{P}$ that no other such node is labeled with, or v has in-degree $n > 1$ and is labeled with one of the* `AND` *and* `OR` *Boolean operators.*

Definition 7 (MBT). *An MBC where each gate has out-degree less than or equal to 1 is called a* monotone Boolean tree (MBT). *It can also be viewed as the abstract syntax tree (AST) of a logical formula containing only the $\wedge$ and $\vee$ operators.*

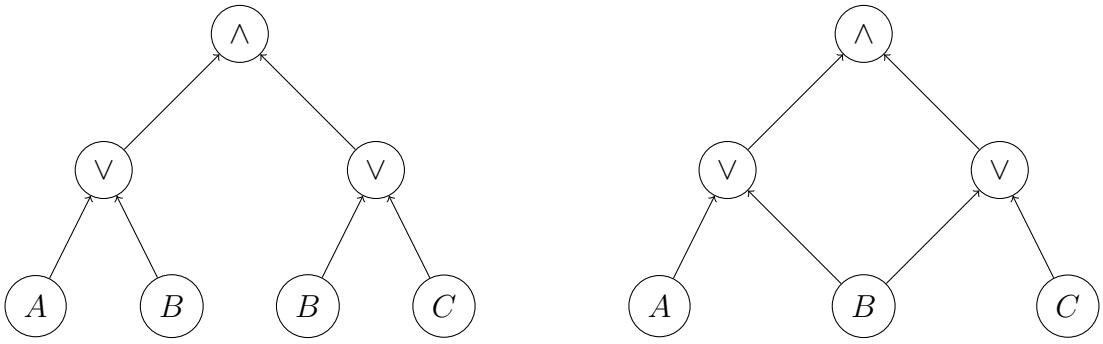
The term "monotone" refers to the absence of negations: these trees and circuits use only `AND` and `OR`-gates, but never `NOT`-gates. This is because an access structure is monotone if and only if it can be computed by a *monotone* Boolean tree (or circuit).

Example 4. *Take, for instance, the MBT in Fig. 1a and the equivalent MBC in Fig. 1b. They both model the access policy $((A \vee B) \wedge (B \vee C))$. However, note that, in the second model, the out-degree of input B is 2, hence making it a circuit, indeed.*

Definition 8 (Threshold Tree). *A* threshold tree *(referred to as "access tree" in [3]) only uses threshold gates as operators. Such a gate is characterized by the threshold t, and is satisfied only when at least t out of its n input nodes are also satisfied.*

Example 5. *Let T be a (2/3)-threshold gate of inputs A, B, and C.*

i) If only A and C are satisfied, then T is satisfied too.
ii) If all A, B, and C are satisfied, then T is satisfied too.
iii) If only B is satisfied, then T is not satisfied.



(a) An MBT computing $\mathbb{A}$. **(b)** An MBC computing $\mathbb{A}$.

Figure 1: An MBT and, respectively, an equivalent MBC, both of them computing the same MAS $\mathbb{A} = \{\{B\}, \{A, B\}, \{A, C\}, \{B, C\}, \{A, B, C\}\}$.

Note that an `AND`-gate can be viewed as an (n/n)-threshold gate, and an `OR`-gate can be viewed as a $(1/n)$-threshold gate.

Definition 9 (Compartmented Tree)**.** *A* compartmented tree *(referred to as "CAS tree" in [6]) only uses compartmented gates as operators. A compartmented gate divides its n input wires into k disjoint compartments, such that $n = n_1 + \cdots + n_k$, where n_i is the number of input wires of compartment i. Each compartment i has a threshold $t_i \leq n_i$, while the entire gate also has its own threshold t, such that $t_1 + \cdots + t_k \leq t \leq n$. Indeed, a compartmented gate is satisfied only when all its $k + 1$ thresholds are satisfied.*

Example 6. *Let T be a compartmented gate with $n = 6$ inputs A, B, C, D, E, and F, respectively, a threshold of $t = 4$, and $k = 2$ compartments with $(t_1, n_1) = (1, 2)$ and $(t_2, n_2) = (2, 4)$, respectively.*

i) If only A, B, D, and F are satisfied, then all compartments, including the main one, are satisfied, and thus the entire gate T is satisfied.
ii) If only B, C, and D are satisfied, then both small compartments are satisfied, but the main one is not, and thus the entire gate T is not satisfied.
iii) If only C, D, E, and F are satisfied, then the main compartment is satisfied, but among the small ones, the first compartment is not, and thus the entire gate T is not satisfied.
iv) If only B and E are satisfied, then only the first small compartment is satisfied, and thus the entire gate T is not satisfied.

Obviously, any threshold tree can be simulated by a compartmented tree, since a threshold gate can be replaced by a similar compartmented gate of only one compartment.

Chapter 3

Monotone Span Program Limitations

We have shown, thanks to Theorem 2, that MSPs are a very powerful computational model for building ABE schemes. However, unlike MBTs and MBCs, MSPs do not encode MASes in a human-readable manner. Thus, we are concerned with finding ways of efficiently converting access policies to MSPs, in order to then be able to pass those as inputs to the ABE schemes. But this, of course, depends on the type of the access policy (i.e., the computational model used for its MAS). In this chapter, we go over some constructions from particular MASes to MSPs, we review the known lower bounds of the MSP dimension in regard to the MAS size (while also proving a new one, in regard to the MBC size), and we finally introduce the concept of `U`-gates (while also giving a novel way to prove the nonexistence of ideal LSSSes for them, and presenting their application to graph access structures).

3.1 Operations and Constructions

In order to help in designing MSP construction algorithms, various operations over MSPs [18] have been developed, which let us form "bigger" MSPs from "smaller" ones. For example, by already knowing two MSPs $\hat{M}_1$ and $\hat{M}_2$ computing the MASes given by $(A \wedge B)$ and, respectively, $(C \vee D \vee E)$, we should be able to somehow combine them in order to form an MSP $\hat{M}_3$ computing $(A \wedge (C \vee D \vee E))$. Luckily, this can be achieved by doing an *insertion* [18] of $\hat{M}_2$ at B in $\hat{M}_1$. That is, by, conceptually, replacing B by $(C \vee D \vee E)$.

Moreover, there are known efficient constructions from MBTs of in-degree 2 [19] and from threshold trees [20] (using insertions). Since any MBT is just a threshold tree with only "1 out of n" (i.e., `OR`) and "n out of n" (i.e., `AND`) gates, the latter construction also translates to an efficient construction from *general* MBTs. These two constructions are linear in the number of leaves of the given tress.

Example 7. *The MSP construction from MBTs of in-degree* 2 *[19] works as in*

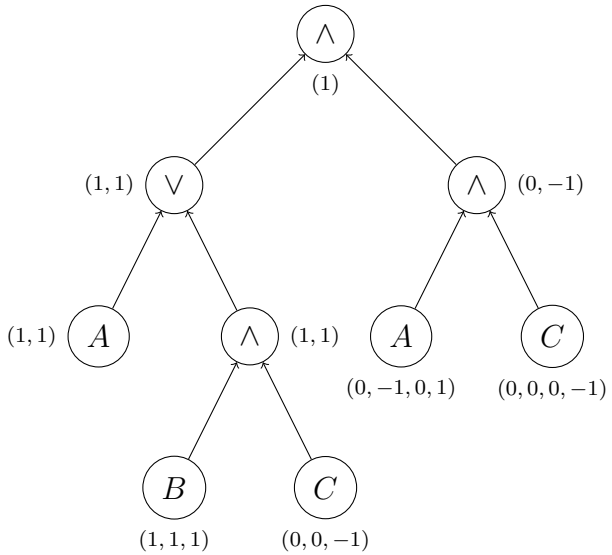


Figure 2: An MBT of in-degree 2 modelling the access policy $((A \vee (B \wedge C)) \wedge (A \wedge C))$. Each node is labeled with its corresponding vector at the moment it was traveled. The travel order is root–left–right.

Fig. 2. Therefore, the resulting MSP $\hat{M}$ *computing* $((A \vee (B \wedge C)) \wedge (A \wedge C))$ *is*

$$\hat{M} = \left[\begin{array}{c|cccc} A & 1 & 1 & 0 & 0 \\ B & 1 & 1 & 1 & 0 \\ C & 0 & 0 & -1 & 0 \\ A & 0 & -1 & 0 & 1 \\ C & 0 & 0 & 0 & -1 \end{array}\right].$$

For instance, if only A *and* C *are satisfied, then the entire access policy is satisfied as well, which is fine because* $\langle M_1, M_3, M_4, M_5 \rangle$ *spans* $\vec{1} = (1, 0, 0, 0)$ *as* $1 \cdot M_1 + 1 \cdot M_4 + 1 \cdot M_5$.

3.2 General Lower Bounds

Wegener authored a comprehensive study [21] on the complexity of Boolean functions, which comes in handy when it is to decide what specific computational model should be used to represent a particular Boolean function. Moreover, some lower bounds for the minimum size of MSPs computing peculiar Boolean functions were proven in the literature [17, 22].

For example, take the construction [17] for the NonBipartite$_m$ access structure, which only needs m rows when working in GF(2). The input of this function is a binary array of length $m = \binom{n}{2}$, encoding the edges of a graph G with n vertices, and its output is 1 if and only if G is not bipartite.

Another such function [22] is PerfectMatching$_n$, which tests whether a graph contains a perfect matching. An MSP can compute this function using a number of rows that is polynomial in n.

3.3 Monotone Boolean Circuit Lower Bounds

MBTs are the most compact human-readable computational model for access policies, so they are of big interest for us. The correlation between MSPs, LSSSes, and Boolean circuits has been studied in various combinations, and for various computational models. However, the existing literature on these problems is unorganized and studies these concepts in isolation from one another. Also, there is a lack of correlation between circuit theory and MSPs.

When it comes to MBCs, designing an MSP construction algorithm directly operating on the circuit, without first converting it to a tree, is challenging, if not impossible. Therefore, taking inspiration from Theorem 3, we narrow our research focus to finding MSP lower bounds.

Theorem 3 ([23])**.** *A lower bound for the size of an MSP computing a particular MAS is* $2^{\Omega(n)}$*, where* n *is the number of parties.*

Let us introduce the concept of "MAS/MBC construction," referring to a mapping from the number of parties n to an MAS/MBC over $\mathcal{P} = \{P_1, \ldots, P_n\}$. Hence, the above theorem can be rephrased, more intuitively, as "there exists an MAS construction requiring an MSP of size at least $2^{\Omega(n)}$."

This result, while of a significant theoretical importance, is lacking a practical character, since the input to the LSSS is not the set of parties, but rather the MAS itself. Since the MAS, as a set of authorized parties, is esentially a DNF formula, its actual size may be exponential in n. Thus, we should also search for MSP lower bounds expressed in the size of the input sent to the LSSS, be it an MAS or an MBC.

Theorem 4. *A lower bound for the size of an MSP computing the MAS computed by a particular polynomial MBC is* $2^{\Omega(m)}$*, where* m *is the MBC size.*

Proof. According to the proof of Theorem 3, there is a so-called `GEN` function (i.e., MAS construction) requiring MSPs of size at least $2^{\Omega(n)}$. Since the `GEN` function is computable in monotone $\mathbb{P}$, and hence by a polynomial MBC, it follows that there are polynomial (in n) MBCs requiring exponential (in m) MSPs. □

Only polynomial MBCs are relevant, since they are inherently trying to be efficient, whereas the exponential MBCs can employ an exponential amount of redundant gates.

3.4 `U`-Gates and Ideal Schemes

First, let us introduce the concept of `U`-gates, which will further be used throughout this paper, for multiple proofs and experiments.

Definition 10 (`U`-Gate)**.** *A* `U`-gate *is an MBT/MBC gate, of inputs A, B, C, and D, embedding the MAS represented by* $((A \wedge B) \vee (B \wedge C) \vee (C \wedge D))$*, as in Fig. 3.*

They will be of great use especially because they do not admit ideal LSSSes. So, let us prove it.

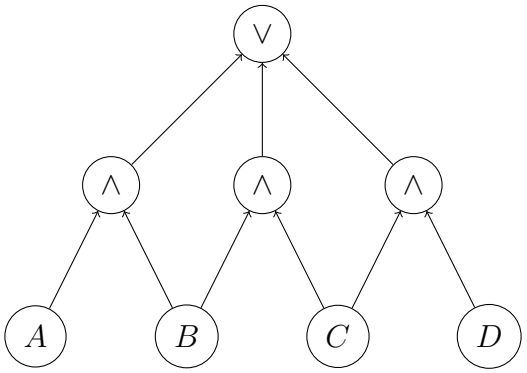


Figure 3: The U-gate [24], representing the access policy $((A \wedge B) \vee (B \wedge C) \vee (C \wedge D))$.

Lemma 1. *The MAS $\mathbb{A}$ of the* U*-gate does not admit ideal LSSSes.*

Proof. Let us assume that $\mathbb{A}$ admits an MSP with

$$\rho = \{(1, A), (2, B), (3, C), (4, D)\}$$

and let its rows be $\vec{A}$, $\vec{B}$, $\vec{C}$, and $\vec{D}$, respectively. Thus, there exist $a, b, c, d, e, f \in \mathbb{Z}_p$ such that

$$a\vec{A} + b\vec{B} = \vec{1},$$
$$c\vec{B} + d\vec{C} = \vec{1},$$
$$e\vec{C} + f\vec{D} = \vec{1}.$$

First, notice that no coefficient can be zero. Take, for instance, $a = 0$. Then, from the first equation it would result that $b\vec{B} = \vec{1}$. This implies that $\{B\} \in \mathbb{A}$, which violates the hypothesis.

Multiplying the first equation by c, the second one by b, and then subtracting them leads to

$$ac\vec{A} - bd\vec{C} = (c - b)\vec{1}.$$

As explained above, it is safe to divide this by bd. After some algebraic manipulation, we get

$$ace\vec{A} + bdf\vec{D} = (bd + ce - be)\vec{1}.$$

First Case If $bd + ce - be \neq 0$, then

$$\frac{ace}{bd + ce - be}\vec{A} + \frac{bdf}{bd + ce - be}\vec{D} = \vec{1},$$

and therefore $\{A, D\} \in \mathbb{A}$, which cannot be true.

Second Case If $bd + ce - be = 0$, then

$$\vec{A} = -\frac{bdf}{ace}\vec{D}.$$

Now, in any equation, $\vec{A}$ can be replaced by $(-bdf/ace)\vec{D}$ and $\vec{D}$ can be replaced by $(-ace/bdf)\vec{A}$. Therefore, the first equation (i.e., $\{A, B\} \in \mathbb{A}$) implies that $\{D, B\} \in \mathbb{A}$ and the third one (i.e., $\{C, D\} \in \mathbb{A}$) implies that $\{C, A\} \in \mathbb{A}$. Both findings violate the given access structure.

Therefore, $\mathbb{A}$ does not admit any MSP of 4 rows (i.e., ideal LSSS). □

However, the U-gate admits the following MSP $\hat{M}$ of size 5, where $\vec{1} = (1, 0, 0)$:

$$\hat{M} = \left[\begin{array}{c|ccc} A & 0 & -1 & 0 \\ B & 1 & 1 & 0 \\ C & 0 & -1 & 0 \\ C & 1 & 0 & 1 \\ D & 0 & 0 & -1 \end{array}\right].$$

Hence, the lowest number of rows $\hat{M}$ needs is 5.

3.5 Graph Access Structures and Ideal Schemes

As an application of the ideas presented in the previous proof, we look into graph access structures and we provide a necessary condition for them to admit ideal LSSSes. This condition was already proved to also be *sufficient* [25].

Definition 11 (GAS [24])**.** *Let $\mathcal{P}$ be the set of parties and let $\mathbb{A}$ be an MAS over $\mathcal{P}$. By $\mathbb{A}^+$, we denote the set of the minterms of $\mathbb{A}$. If $\mathbb{A}^+ \subseteq \{\{U, V\} \mid U, V \in \mathcal{P}, U \neq V\}$, then $\mathbb{A}$ is a* graph access structure (GAS).

Lemma 2. *Let $A, B, C \in \mathcal{P}$. If $\{A, B\} \in \mathbb{A}$, $\{B, C\} \in \mathbb{A}$, $\{A, C\} \notin \mathbb{A}$, and $\{X \mid \{A, X\} \in \mathbb{A}\} \neq \{X \mid \{C, X\} \in \mathbb{A}\}$ (i.e., the adjacency lists of A and C are different), then $\mathbb{A}$ does not admit an ideal LSSS.*

Proof. Some details were already touched in the previous proof, and thus are ommited. Since sets $\{A, B\}$ and $\{B, C\}$ are authorized, then there exist $a, b, c, d \in \mathbb{Z}_p^*$ such that

$$a\vec{A} + b\vec{B} = \vec{1},$$
$$c\vec{B} + d\vec{C} = \vec{1}.$$

Through some algebraic manipulation we get

$$ac\vec{A} - bd\vec{C} = (c - b)\vec{1}.$$

Case $b \neq c$ would lead to $\{A, C\} \in \mathbb{A}$, which is false. Thus, $b = c$ and, consequently, $\vec{A} = (bd/ac)\vec{C}$. Therefore, any $\{A, X\} \in \mathbb{A}$ also implies that $\{C, X\} \in \mathbb{A}$ and any $\{C, X\} \in \mathbb{A}$ also implies that $\{A, X\} \in \mathbb{A}$. □

Lemma 2 shows that, if a graph $\mathbb{A}$ contains any subgraph $\{A, B, C\}$ with said property, then $\mathbb{A}$ does not admit an ideal LSSS. The full implication of this result [25] is that a graph admits an ideal LSSS if and only if it is multipartite.

In regard to GASes, an interesting open problem is finding the minimum number of rows an MSP needs in order to compute them (i.e., the smallest LSSS). For all graphs on 6 nodes, the answer has already been found. However, when it comes to general secret sharing schemes, the answer remained unknown for two GASes [26].

Chapter 4

From Monotone Boolean Circuits to KP-ABE Schemes

One open problem is to construct an efficient KP-ABE system for MBCs using bilinear maps. The already existing schemes have exponential expansion in the key size [4, 5]. However, recent works try to improve these results, either by directly optimizing the circuit structure using heuristics [27], or by constructing more efficient LSSSes for some particular MBCs, such as compartmented groups [6]. This chapter presents the two main approaches of designing such KP-ABE schemes, while going in-depth into our backtracking solution.

4.1 Two Approaches

There are two main approaches when it comes to building KP-ABE schemes from MBCs. The **first** approach uses a KP-ABE construction directly based on an MBC as input [4]. You may either *a priori* optimize the given circuit (i.e., reduce its size) or not. This size reduction can be done using nature-inspired methods, like simulated annealing [27]:

1. MBC $\xrightarrow{[4]}$ LSSS;
2. MBC $\xrightarrow{\text{NIM optimization [27]}}$ MBC $\xrightarrow{[4]}$ LSSS.

Unfortunately, the resulting scheme may still get exponential.

The **second** approach consists in first converting the MBC to an MSP and then applying a scheme designed for MSPs [3]. Now, the conversion procedure may take place in two ways. You either convert the MBC to an MBT of in-degree 2 and then apply the already existing conversion method [19], or you view the MBC as an MAS and generate suitable MSPs from scratch, with as few rows as possible, through backtracking:

3. MBC $\rightarrow$ MBT $\rightarrow$ 2-MBT $\xrightarrow{[19]}$ MSP $\xrightarrow{[3]}$ LSSS;
4. MBC $\rightarrow$ MBT $\rightarrow$ DNF-tree $\equiv$ MAS $\xrightarrow{\text{backtracking}}$ MSP $\xrightarrow{[3]}$ LSSS.

The downside of both variants is the conversion to MBT potentially leading to an exponential conversion time. Moreover, the conversion to a disjunctive normal form (DNF) tree may also lead to an exponentially larger tree.

For instance, it is a well-known fact that the conjunctive normal form (CNF) formula $((X_1 \vee Y_1) \wedge \cdots \wedge (X_n \vee Y_n))$ requires an exponential number of literals in order to be rewritten in DNF. Therefore, the MAS induced by it will *directly* generate a very inneficient MSP.

That being said, if using *backtracking* and constructing the MSP *from scratch*, its size *may* break this exponential gap, so its worth trying to design a backtracking approach as good as possible.

4.2 Motivation for Improvements

Let us take the policy $((A \wedge B) \vee (B \wedge C))$, defined over $\mathcal{P} = \{A, B, C\}$, as an example. It induces the MAS $\mathbb{A} = \{\{A, B\}, \{B, C\}, \{A, B, C\}\}$.

Applying the conversion method described previously leads us to $\hat{M}_1$, of size 4. However, one may remark that a better — actually, an *ideal* — MSP $\hat{M}_2$, of size 3, can be obtained:

$$\hat{M}_1 = \left[\begin{array}{c|ccc} A & 1 & 1 & 0 \\ B & 0 & -1 & 0 \\ B & 1 & 0 & 1 \\ C & 0 & 0 & -1 \end{array}\right]; \quad \hat{M}_2 = \left[\begin{array}{c|cc} A & 0 & -1 \\ B & 1 & 1 \\ C & 0 & -1 \end{array}\right].$$

Hence, we are inclined to believe that there exist certain classes of MASes which admit better MSPs than those obtained throught the straightforward conversion.

4.3 Backtracking Solution

We tried [28] a backtracking approach that generates candidates for matrix M. The inputs are:

- $\mathbb{A}$ — the given MAS;
- $m \times d$ — the dimensions of the matrix;
- ρ — the labeling of the rows of the matrix;
- p — the matrix is defined over $\mathbb{Z}_p$;
- k — the elements of the matrix take values from $\{0, 1, \ldots, k-1\}$.

The matrix is generated row by row and element by element. When a row i is finished, every submatrix M_I induced by a subset of rows $I \subseteq \{1, 2, \ldots, i\}$, with $i \in I$, is analyzed in order to abort this branch of execution if possible. Thus, the following Boolean values are computed:

- `full` — there is no row $j \notin I$ with $\rho(j) \in \{\rho(i) \mid i \in I\}$;
- `auth` — $\{\rho(i) \mid i \in I\}$ is an authorized subset;
- `span` — $\vec{1} \in \text{span}(M_I)$.

Therefore, we can backtrack when `auth` $\wedge$ `full` $\wedge \neg$`span` or $\neg$`auth` $\wedge$ `span`.

We mention that testing whether $\vec{1} \in \text{span}(M_I)$ is done using the Rouché-Capelli theorem. That is, M_I spans $\vec{1} = (1, \ldots, 1)$ if and only if

$$\text{rank}(M_I) = \text{rank}\left(\begin{bmatrix} M_I \\ \vec{1} \end{bmatrix}\right).$$

Moreover, if the set is authorized but it is not a minterm (i.e., it is a superset of an authorized set), then we may skip the entire step.

4.4 Backtracking Results

We tried to find ideal MSPs for various MAS classes, using $k = 5$ and $p = 10^9 + 7$. The latter is a big prime number, hence the results do show relevance in the context of cryptography.

For the Boolean tree $\mathbb{A}_1 \equiv ((A \wedge B) \vee (B \wedge C))$, for the threshold tree $\mathbb{A}_2 \equiv (3/ABCDE)$, and for the compartmented tree $\mathbb{A}_3 \equiv (3; 1, 1/AB, CDE)$, the MSPs found by the backtracking algorithm are, respectively:

$$\hat{M}_1 = \left[\begin{array}{c|cc} A & 0 & 1 \\ B & 1 & 0 \\ C & 0 & 1 \end{array}\right]; \quad \hat{M}_2 = \left[\begin{array}{c|ccc} A & 0 & 0 & 1 \\ B & 0 & 1 & 0 \\ C & 1 & 0 & 0 \\ D & 1 & 2 & 3 \\ E & 1 & 3 & 2 \end{array}\right]; \quad \hat{M}_3 = \left[\begin{array}{c|ccccc} A & 0 & 0 & 0 & 0 & 1 \\ B & 0 & 0 & 0 & 1 & 0 \\ C & 1 & 1 & 1 & 0 & 0 \\ D & 1 & 1 & 1 & 2 & 3 \\ E & 4 & 4 & 4 & 2 & 3 \end{array}\right].$$

Note that these MSPs are ideal indeed.

However, $\mathbb{A}_4 \equiv ((A \wedge B) \vee (A \wedge C) \vee (C \wedge D))$ (i.e., the $\mathbb{U}$-gate) did not yield any MSP for the given constraints (i.e., with values less than 5), and certainly not an ideal one. This was expected as well, according to Lemma 1.

According to our tests, the backtracking algorithm can easily prove that a given MAS, with a small number of literals (i.e., up to 5), does not admit ideal MSPs with values less than a very small k (i.e., up to 5). Thus, we believe that it can be a starting point in proving the "general" (i.e., without a restriction on k) nonexistence of ideal MSPs for a given MAS.

4.5 Maximum Value Reduction

We will briefly explain how the maximum value in an MSP can be reduced to a smaller one:

$$\hat{M} = \left[\begin{array}{c|cc} A & 4 & 6 \\ B & 6 & 4 \end{array}\right]; \quad \hat{M}' = \left[\begin{array}{c|cc} A & 2 & 3 \\ B & 3 & 2 \end{array}\right].$$

Let us take $p = 7$ and the MSP $\hat{M}$, inducing the MAS $\mathbb{A} = \{\{A, B\}\}$ over $\mathcal{P} = \{A, B\}$, as an example. It can easily be seen that $\hat{M}$ induces $\mathbb{A}$, since

$$5 \cdot M_1 + 5 \cdot M_2 = (6, 2) + (2, 6) = (1, 1) = \vec{1}.$$

One may notice that, when multiplying any row i of M by a non-zero value α, a new MSP $\hat{M}'$ inducing $\mathbb{A}$ is obtained. That is because a linear combination of the rows of M, equal to $\vec{1}$, can be transformed into a linear combination of the rows of M', equal to $\vec{1}$, when multiplying coefficient i by the multiplicative inverse of α modulo p.

In our case, we can multiply both rows of M by $4 = 5^{-1}$, obtaining M'. This new MSP $\hat{M}'$ does indeed induce $\mathbb{A}$, since

$$3 \cdot M'_1 + 3 \cdot M'_2 = (6, 2) + (2, 6) = (1, 1) = \vec{1}.$$

Notice how the maximum value in the MSP happened to change from 6 to 3. Therefore, now we know that if we were to run the backtracking algorithm on $\mathbb{A}$ with $k = 4$ instead of $k = 7$, it would have found a solution much faster.

This method of reducing the maximum value in the MSP may be useful in finding a way to determine the minimum value of k needed to be passed as input to the backtracking algorithm in order for it to be able to find an ideal MSP fo the given MAS.

Chapter 5

A Hybrid KP-ABE Scheme for Universal Circuits

Building on top of the Hu–Gao scheme for "general circuits" and the Goyal et al. scheme for MSPs, we have developed the pairing-based KP-ABE scheme with, we argue, the highest degree of generalization. That is, by using our concept of "universal circuits," each internal node of the circuit can compute *any* kind of monotone access structure (thanks to MSPs), thus not being limited anymore to specific, narrow cases, such as `AND`, `OR`, and (t/n)-threshold gates. Therefore, the search for pairing-based KP-ABE schemes, running over more expressive computational models, is over, since now every circuit node is as expressive as possible.

We have argued that our scheme is more efficient than the Hu–Gao scheme for "general circuits" and the Goyal et al. scheme for "access trees," since universal circuits are more expressive, they use more compact gates, and thus they lead to shorter decryption key sizes. We have implemented our scheme in `C++` and made it publicly available on GitHub [29]. This chapter presents all these contributions.

5.1 Preliminary KP-ABE Schemes

Goyal et al. proposed two important KP-ABE schemes [3] in the same paper: one where the decryption keys are linked to so-called "access trees" (i.e., trees made up of only (t/n)-threshold gates), and another one using MSPs. Both of them are based on bilinear pairings, and use similar techniques.

In the Hu–Gao KP-ABE scheme [5], based as well on bilinear pairings, the keys are linked to so-called "general circuits," which are made up of only `AND` and `OR`-gates, both of in-degree 2. During KeyGen and Decrypt, the circuit is traveled in a depth-first fashion such that, conceptually, it is rewritten as a monotone Boolean tree, over which a simplified version of the Goyal et al. scheme for "access trees" is applied. As an example, for the monotone Boolean circuit $\tilde{\mathcal{C}}'$ (see Fig. 5), the corresponding monotone Boolean tree, formed during the depth-first travel, is $\tilde{\mathcal{T}}'$ (see Fig. 6).

5.2 Universal Circuits

This section introduces the concept of universal circuits — a flexible and expressive computational model for specifying access control policies. Formally, a *universal circuit* is defined as a directed acyclic graph consisting of:

1. *input* (i.e., of in-degree zero) nodes, which embed Boolean variables indicating the presence or absence of specific attributes;
2. *internal* (i.e., of in-degree non-zero) nodes, which embed MSPs computing specific local MASes (i.e., such a node is satisfied only when its satisfied inputs form an authorized subset of said MAS);
3. exactly one *output* (i.e., of out-degree zero) node, indicating whether the overall access policy is fulfilled.

To demonstrate that universal circuits are at least as expressive as existing circuit models commonly found in the literature — typically composed of `AND`, `OR`, and/or (t/n)-threshold gates — we will next show how each of these standard gates can easily be represented using MSPs within our computational model.

Moreover, once we will present our proposed scheme, it will be clear that its implementation does not actually rely on the entire matrix of each MSP $\hat{M}$, but rather on two custom functions related to it:

1. one that computes the dot product between M and a given vector $\vec{r}$;
2. one that finds solutions to the equation $\vec{\alpha} \cdot M_{\mathcal{X}} = \vec{1}$ (i.e., what linear combination $\tilde{\vec{\alpha}}$ of the rows of M labeled with parties in $\mathcal{X}$ gives the objective vector).

First, it is not trivial to efficiently find these solutions without prior knowledge of the MSP structure. Second, since it is known, both functions can sometimes be optimized. This is the case for the classic gates covered next.

5.2.1 Classic Gates

The `AND`-Gate An MSP computing the MAS of an `AND`-gate with input nodes $I_1, I_2, \ldots, I_n$ is

$$\hat{M}_{\texttt{AND}(n)} = \left[\begin{array}{c|cccc} I_1 & 1 & 0 & \cdots & 0 \\ I_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I_n & 0 & 0 & \cdots & 1 \end{array}\right].$$

Thus, $M \cdot \vec{r} = \vec{r}$, and the only solution to $\vec{\alpha} \cdot M_{\mathcal{X}} = \vec{1}$ is $\tilde{\vec{\alpha}} = (1, 1, \ldots, 1)$, which requires all inputs to be satisfied.

The OR-Gate An MSP computing the MAS of an OR-gate with input nodes $I_1, I_2, \ldots, I_n$ is

$$\hat{M}_{\mathtt{OR}(n)} = \left[\begin{array}{c|cccc} I_1 & 1 & 1 & \cdots & 1 \\ I_2 & 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I_n & 1 & 1 & \cdots & 1 \end{array}\right].$$

Thus, $M \cdot \vec{r} = \sum_{i=1}^{n} \vec{r}_i$, and one possible solution to $\vec{\alpha} \cdot M_{\mathcal{X}} = \vec{1}$, when at least one input I_i is satisfied, is $\tilde{\vec{\alpha}}$, where $\tilde{\vec{\alpha}}_j = 1$ for $j = i$, and $\tilde{\vec{\alpha}}_j = 0$ for $j \neq i$, while there are no solutions when no input is satisfied.

The (t/n)-Threshold Gate An MSP computing the MAS of a (t/n)-threshold gate with input nodes $I_1, I_2, \ldots, I_n$ is

$$\hat{M}_{(t/n)} = \left[\begin{array}{c|ccccc} I_1 & 1 & 1 & 1 & \cdots & 1 \\ I_2 & 1 & 2 & 2^2 & \cdots & 2^{t-1} \\ I_2 & 1 & 3 & 3^2 & \cdots & 3^{t-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ I_n & 1 & n & n^2 & \cdots & n^{t-1} \end{array}\right].$$

First, $M \cdot \vec{r}$ can be computed without storing the entire matrix M, since its elements can be deduced on the fly. Second [20], when it comes to solving $\vec{\alpha} \cdot M_{\mathcal{X}} = \vec{1}$, there is no solution when less than t inputs are satisfied, while, otherwise, it is enough to look at only the first t satisfied inputs, since a square matrix $M_{\mathcal{X}'}$ will be formed, and one possible solution will be $\tilde{\vec{\alpha}} = \vec{1} \cdot M_{\mathcal{X}'}^{-1}$, which is computable in $\mathcal{O}(t^3)$.

5.2.2 Custom Gates

Now that we have covered how the well-known AND, OR, and (t/n)-threshold gates are to be converted into equivalent MSP-embedded gates, hence making them usable in our proposed KP-ABE scheme, let us introduce a new type of gate, namely the U-gate, which highlightes the efficiency of using universal circuits.

A U-gate of inputs I_1, I_2, I_3, and I_4 is satisfied if and only if $((I_1 \wedge I_2) \vee (I_2 \wedge I_3) \vee (I_3 \wedge I_4))$, and

$$\hat{M}_{\mathtt{U}} = \left[\begin{array}{c|ccccc} I_1 & 0 & 0 & 0 & 0 & 1 \\ I_2 & 1 & 1 & 1 & 1 & 0 \\ I_3 & 0 & 0 & 0 & 0 & 1 \\ I_3 & 0 & 0 & 0 & 1 & 0 \\ I_4 & 1 & 1 & 1 & 0 & 0 \end{array}\right].$$

Note that, in the case of $\hat{M}_{\mathtt{U}}$, the following linear combinations of the rows of M are enough to solve $\vec{\alpha} \cdot M_{\mathcal{X}} = \vec{1}$ for any authorized subset $\mathcal{X}$:

$$(1, 1, 0, 0, 0),$$
$$(0, 1, 1, 0, 0),$$
$$(0, 0, 1, 1, 1).$$

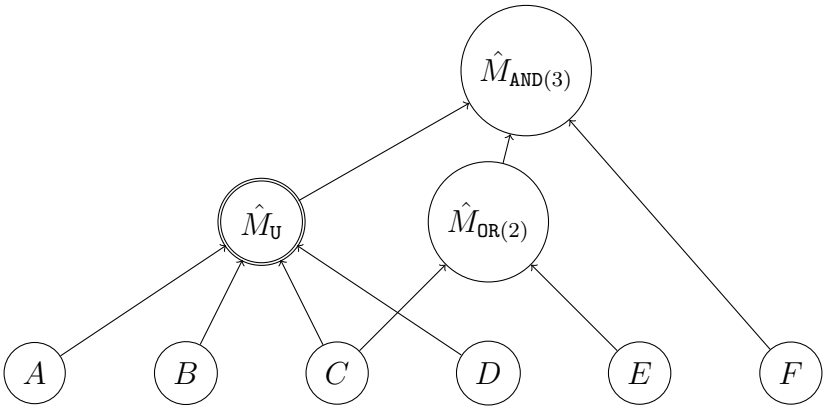


Figure 4: Example of a universal circuit $\tilde{\mathcal{C}}$ using `AND`, `OR`, and `U`-gates.

That is, for any authorized subset $\mathcal{X}$, there exists a linear combination $\tilde{\vec{\alpha}}$ among the aforementioned ones, solving the equation, such that the only rows i it involves (i.e., with $\tilde{\vec{\alpha}}_i \neq 0$) are rows with $\rho(i) \in \mathcal{X}$.

It can be proven [24] that an MSP of smaller size cannot be constructed for this MAS.

5.2.3 Circuit Example

Let us take as an example the universal circuit $\tilde{\mathcal{C}}$ (see Fig. 4), which uses `AND`, `OR`, and `U`-gates. Note that the input node C is of out-degree 2, so $\tilde{\mathcal{C}}$ cannot be considered a tree, but strictly a circuit.

For instance, a key embedding $\tilde{\mathcal{C}}$ is able to decrypt a cyphertext with attributes $\{A, C, D, F\}$, while it is not able to decrypt a ciphertext with attributes $\{B, C, E\}$.

To illustrate that our scheme is more expressive than Hu–Gao, it is enough to look at how the same universal circuit $\tilde{\mathcal{C}}$ would look if it was made up of only classic gates, that is, if it was a monotone Boolean circuit $\tilde{\mathcal{C}}'$ (see Fig. 5). In order to compute $((A \wedge B) \vee (B \wedge C) \vee (C \wedge D))$, it uses 4 gates, whereas $\tilde{\mathcal{C}}$ uses only 1. No combination of `AND`, `OR`, and (t/n)-threshold gates can beat, or even reach, the compactness of $\tilde{\mathcal{C}}$. This translates to an even bigger impact regarding the performance of our proposed KP-ABE scheme, but this will be more thoroughly analyzed in Section 5.5.

5.3 Construction

Our proposed KP-ABE scheme for universal circuits combines the key ideas of the Hu–Gao and Goyal et al. constructions. Like Hu–Gao, we structure the decryption process over a circuit, even though its gates are not `AND` and `OR`-gates, but actual MSPs, and, over each such MSP, we apply the Goyal et al. scheme. By integrating these two approaches, and building on bilinear pairings, our scheme efficiently supports complex, general-purpose access policies in a compact and flexible way.

$\mathsf{Setup}(\lambda, n)$ Based on the security parameter λ and the number of attributes n, it chooses a prime p on λ bits and the attribute universe $\mathcal{U} = \{0, 1, \ldots, n-1\}$. Then,

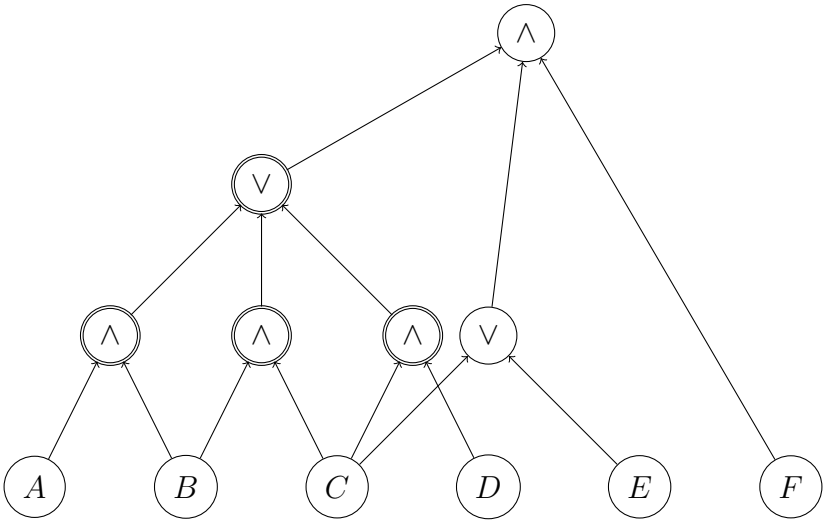


Figure 5: The circuit $\tilde{\mathcal{C}}$ rewritten as an MBC $\tilde{\mathcal{C}}'$.

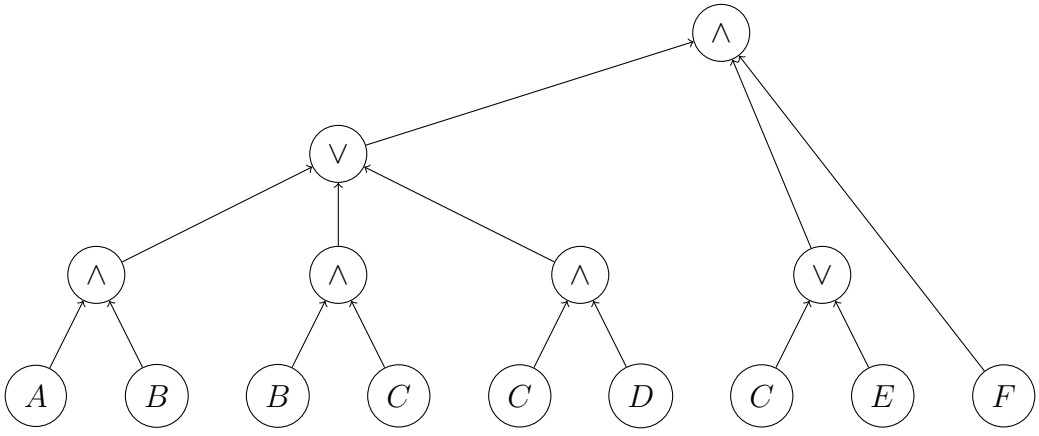


Figure 6: The MBC $\tilde{\mathcal{C}}'$ rewritten as an MBT $\tilde{\mathcal{T}}'$.

it generates two bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order p, a generator g of $\mathbb{G}_1$, and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.

Afterwards, it randomly chooses $y \in \mathbb{Z}_p$ and $t_i \in \mathbb{Z}_p, \forall i \in \mathcal{U}$. Then, it computes $Y = e(g,g)^y$ and $T_i = g^{t_i}, \forall i \in \mathcal{U}$. Finally, it returns the public key $pk = (g, Y, T)$ and the master key $mk = (g, y, t)$.

$\mathsf{Encrypt}(pt, A, pk)$ Based on the plaintext $pt \in \mathbb{G}_2$, the attribute set $A \subseteq \mathcal{U}$, and the public key $pk = (g, Y, T)$, it randomly chooses $s \in \mathbb{Z}_p$, and then it returns the ciphertext $ct = (A, g^s, E', E)$, where $E' = ptY^s$ and $E_i = T_i^s, \forall i \in A$.

$\mathsf{KeyGen}(\mathcal{C}, mk)$ It travels the given circuit $\mathcal{C}$ in a top–down manner, conceptually converting it into a tree $\mathcal{T}$, of root node r, and associating each node v of $\mathcal{T}$ with a value $K(v)$. Note that the children array of some node in $\mathcal{T}$ does not have to match the children array of its corresponding node $\mathcal{C}[v]$ in $\mathcal{C}$, but rather the labeling ρ of the rows in $\hat{M}$, where $\hat{M}$, of size $m \times d$, is the MSP of $\mathcal{C}[v]$.

Based on the master key $mk = (g, y, t)$, it initially sets $K(r) = y$. During the travel, for each internal node v, it randomly chooses $\vec{r} \in \mathbb{Z}_p^d$ such that $\sum_{i=1}^{d} \vec{r}_i = K(v)$, and then it sets $K(w_i) = (M \cdot \vec{r})_i$, for each child w_i of v.

When reaching the leaf node v, with a being the attribute of $\mathcal{C}[v]$, it sets $\tilde{K}(v) = g^{K(v)/t_a}$.

Finally, it returns the decryption key $dk = (\mathcal{C}, \tilde{K})$.

$\mathsf{Decrypt}(ct, dk)$ Based on the ciphertext $ct = (A, g^s, E', E)$ and the decryption key $dk = (\mathcal{C}, \tilde{K})$, it travels the circuit $\mathcal{C}$ in a bottom–up manner, conceptually converting it into the same tree $\mathcal{T}$ as before, and associating each node v of $\mathcal{T}$ with a value $D(v)$.

When reaching the leaf node v of $\mathcal{T}$, with a being the attribute of $\mathcal{C}[v]$, it sets $D(v) = e(\tilde{K}(v), E_a)$ if $a \in A$, or $D(v) = \perp$ otherwise.

During the travel, for each internal node v, with children array w and with $\hat{M}$, of size $m \times d$, being the MSP of $\mathcal{C}[v]$, it begins by solving the equation $\vec{\alpha} \cdot M_{\mathcal{X}} = \vec{1}$, where $\mathcal{X}$ is the subset of nodes $\mathcal{C}[w_i]$ such that $D(w_i) \neq \perp$. If there is no solution, then it sets $D(v) = \perp$. Otherwise, denoting the found solution as $\tilde{\vec{\alpha}}$, it sets $D(v) = \prod_{i=1}^{m} D(w_i)^{\tilde{\vec{\alpha}}_i}$.

Finally, it returns $\tilde{D} = E'/D(r)$ if $D(r) \neq \perp$, or $\tilde{D} = \perp$ otherwise.

5.4 Security Proof

This section presents the formal proof of security of our KP-ABE scheme under the selective set security model, which can be viewed as a game between a challenger and an attacker. At the end of the game, the attacker tries to guess which plaintext, among pt_0 and pt_1, was encrypted as ct. If the guess is right, then the attacker wins the game. Otherwise, they lose the game.

We prove that, if there exists a polynomial-time attacker who can break our KP-ABE scheme with advantage ϵ, then the challenger can solve the DBDH problem with advantage $\epsilon/2$.

First, the challenger chooses the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order p, the bilinear pairing e, and the generator g of $\mathbb{G}_1$. Second, they flip a fair binary coin μ. If $\mu = 0$, then they set $(A, B, C, Z) = (g^a, g^b, g^c, e(g,g)^{abc})$. Otherwise, they set $(A, B, C, Z) = (g^a, g^b, g^c, e(g,g)^z)$, where a, b, c, and z are chosen at random from $\mathbb{Z}_p$ (but not by the challenger).

Init The attacker chooses the challenge attribute set A.

Setup The challenger sets the public parameters $Y = e(A, B) = e(g,g)^{ab}$ and T, where $T_i = g^{r_i}$ if $i \in A$, or $T_i = B^{r_i} = g^{br_i}$ otherwise, with r_i chosen at random from $\mathbb{Z}_p$.

Phase 1 The attacker adaptively sends a polynomial number of circuits to the challenger, such that A does not satisfy any of them. For each circuit $\mathcal{C}$, the challenger generates the decryption key for $y = ab$ (without knowing the value itself), and sends it to the attacker.

We will make use of two procedures in our proof, MSPsat and MSPunsat, which will share a value from $\mathbb{Z}_p$ and $\mathbb{G}_1$, respectively, over the respective satisfied or unsatisfied node. Their description is as follows.

MSPsat(a, v) It shares the value a from $\mathbb{Z}_p$ thorugh the subtree rooted in v. This is similar to the regular secret sharing in the key generation process. More specifically, it randomly chooses $\vec{r} \in \mathbb{Z}_p^d$ such that $\sum_{i=1}^d \vec{r}_i = K(v)$, and then it sets $K(w_i) = (M \cdot \vec{r})_i$, for each child w_i of v.

MSPunsat(g^a, v) It shares the value g^a over the node v. This procedure is very similar to the one from [3], but adapted in such a way that it can be applied to a tree of MSPs.

This procedure simulates the secret sharing of a over the MSP $\hat{M}$ corresponding to the gate v, but using values from $\mathbb{G}_1$ (i.e., g^a instead of a).

To generate the secret key, the challenger has to fix a random vector $\vec{r} = (r_1, r_2, \ldots, r_d)$ such that $\vec{1} \cdot \vec{r} = a$. To do that, first define $\vec{v} = (v_1, v_2, \ldots, v_d)$ such that $v_i = b\lambda_i$, for λ_i randomly chosen from $\mathbb{Z}_p$.

Remark 1 ([3])**.** *A vector $\vec{\pi}$ is independent of a set of vectors represented by a matrix N if and only if there exists a vector $\vec{w}$ such that $N \cdot \vec{w} = \vec{0}$, while $\vec{\pi} \cdot \vec{w} \neq 0$.*

Since $\vec{1}$ is independent of M, there exists a vector $\vec{w}$ such that $M \cdot \vec{w} = \vec{0}$ and $\vec{1} \cdot \vec{w} \neq 0$; let $h = \vec{1} \cdot \vec{w}$. Such a vector can be efficiently computed. Let $\vec{w} = (w_1, w_2, \ldots, w_d)$. Finally, define the vector $\vec{u}$ as $\vec{u} = \vec{v} + \psi \cdot \vec{w}$, where

$$\psi = \frac{a - \sum_{k=1}^d \lambda_k}{h}.$$

Note that

$$u_i = v_i + \psi w_i = \lambda_i + \frac{a - \sum_{k=1}^d \lambda_k}{h} \cdot w_i.$$

Let $M_j = (x_{j1}, x_{j2}, \ldots, x_{jd})$. Give the secret key for the row M_j as follows.

If node j is a satisfied child, then

$$K(j) = \sum_{i=1}^{d} x_{ji}\lambda_j.$$

Note that $K(j)$ is completely known in this case and forms a legitimate key because

$$\begin{aligned} M_j \cdot \vec{u} &= M_j \cdot (\vec{v} + \psi\vec{w}) \\ &= M_j \cdot \vec{v} + \psi(M_j \cdot \vec{w}) \\ &= M_j \cdot \vec{v} + \psi \cdot 0 \\ &= \textstyle\sum_{i=1}^{d} x_{ji}\lambda_i \\ &= K(j). \end{aligned}$$

If node j is not a satisfied child, then

$$g^{K(j)} = A^{\phi_2} g^{\phi_3} \text{ where } \phi_2 = \frac{\sum_{i=1}^{d} x_{ji}}{h} \text{ and } \phi_3 = \frac{\sum_{i=1}^{d} x_{ji}(h\lambda_i - \sum_{k=1}^{d} \lambda_k)}{h}.$$

Note that ϕ_2 and ϕ_3 are completely known and a legitimate key is formed because

$$\begin{aligned} M_j \cdot \vec{u} &= \textstyle\sum_{i=1}^{d} x_{ji} \left(\lambda_i + \frac{a - \sum_{k=1}^{d} \lambda_k}{h}\right) \\ &= a \textstyle\sum_{i=1}^{d} x_{ji} + \left(\sum_{i=1}^{d} x_{ji}(h\lambda_i - \sum_{k=1}^{d} \lambda_k)\right) \\ &= a\phi_2 + \phi_3. \end{aligned}$$

Now, for each child, forward the value computed above and apply MSPsat and MSPunsat according to whether the child is satisfied or not.

Using the aforementioned procedures, we will share the value $A = g^a$ in a top–down manner over the tree starting with the MSPunsat procedure. This results in having at every leaf node v of the tree values of type $K(v) \in \mathbb{Z}_p$ or $g^{K(v)} \in \mathbb{G}_1$, depending on whether the attribute for the respective leaf node is present or not.

Then, for each leaf node v, of attribute i, set

$$\tilde{K}(v) = \begin{cases} B^{K(v)/r_i} = g^{bK(v)/t_i} & \text{if } i \in A, \\ g^{K(v)/r_i} = g^{bK(v)/br_i} = g^{bK(v)/t_i} & \text{if } i \notin A. \end{cases}$$

This simulates as if ab was shared through the circuit, since $\tilde{K}(v) = g^{bK(v)/t_i}$ in both cases.

Challenge The attacker sends two challenge plaintexts pt_0 and pt_1 to the challenger. The latter flips a fair binary coin ν, and returns the encryption of m_ν, namely $ct = (A, pt_\nu Z, E)$, where $E_i = C^{r_i}, \forall i \in A$.

Phase 2 Exactly as Phase 1.

Guess The attacker makes a guess ν' of ν. If $\nu' = \nu$, then the challenger returns $\mu' = 0$ to indicate that $Z = e(g,g)^{abc}$. Otherwise, they return $\mu' = 1$ to indicate that $Z = e(g,g)^z$.

If $\mu = 1$, then the attacker gains no information about ν. Thus, $\Pr\{\nu \neq \nu' \mid \mu = 1\} = 1/2$. Since they guess $\mu' = 1$ when $\nu \neq \nu'$, it follows that $\Pr\{\mu' = \mu \mid \mu = 1\} = 1/2$.

If $\mu = 0$, then the attacker sees an encryption of pt_ν. Their advantage becomes ϵ. Hence, $\Pr\{\nu = \nu' \mid \mu = 0\} = 1/2 + \epsilon$. Since the guess is $\mu' = 0$ when $\nu = \nu'$, it follows that $\Pr\{\mu' = \mu \mid \mu = 0\} = 1/2 + \epsilon$.

Therefore, the overall advantage of the attacker in this game is $(1/2)\Pr\{\mu' = \mu \mid \mu = 0\} + (1/2)\Pr\{\mu' = \mu \mid \mu = 1\} - 1/2 = (1/2)(1/2+\epsilon) + (1/2)(1/2) - 1/2 = (1/2)\epsilon$.

5.5 Efficiency

For universal circuits that happen to be, in particular, "general circuits," our proposed scheme is as efficient as Hu–Gao, since it does the exact same operations, even though they are disguised as MSP operations. The same can be said when comparing our scheme to that of Goyal et al. for "access trees." Here, it is worth noting the optimizations made for the classic gates play a crucial role, because they do not store the MSP matrix, thus acting almost like no MSPs are involved at all.

However, for universal circuits which really take advantage of the ability to use any kind of MSP-embedded gate, like the `U`-gate, our scheme is more efficient than both Hu–Gao and Goyal et al. for "access trees," because they would need to be provided with heavier circuits, which are not universal. As we have already seen in Section 5.2.3, $\tilde{\mathcal{C}}'$ requires more nodes than $\tilde{\mathcal{C}}$. This translates to the depth-first search tree used during KeyGen and Decrypt having more leaf nodes, thus extending the key generation and decryption times.

In order to get a better grasp on the impact of using `U`-gates directly, like in Fig. 4, instead of representing them using `AND` and `OR`-gates, like in Fig. 5 (see the double circles), let us compute the number of leaves, when the tree conversion happens, that are generated in each case.

Let us denote by $f(X)$ the number of leaves generated by the subcircuit rooted in X. In the first case, if the inputs to the `U`-gate are subcircuits rooted in A, B, C, and D, then it contributes with $f(A) + f(B) + 2f(C) + f(D)$ to the final number of leaves. In the second case, if the inputs to the `AND`-gates are A and B, B and C, and, respectively, C and D, then their contribution is $f(A)+2f(B)+2f(C)+f(D)$, which is greater than the previous one.

That being said, our scheme still inherits the same downside as Hu–Gao. Since the decryption key size is equal to the number of leaves in the depth-first search tree, their size may become exponential for many circuits.

5.6 Implementation

The implementation of our proposed KP-ABE scheme is openly accessible on GitHub [29]. The code was mainly developed in `C++`, chosen for its high execution speed

and efficient memory management, which are essential for cryptographic operations involving large-scale policy evaluation and key generation. For bilinear pairing operations, the `C++` wrapper for `PBC` (pairing-based cryptography) library was used. To compute $M_{\mathcal{X}'}^{-1}$ for (t/n)-threshold gates, the `C++` code defers to running a `Python` script, which uses the `Galois` and `NumPy` libraries. Developers can define fully-customizable MSP classes, that is, with custom `dot` and `solve` functions, by just extending the `MSP` class. This open-source release is intended to promote transparency, facilitate independent verification, and encourage further research and development in the field of KP-ABE.

Chapter 6

Conclusion

First, we have gently explored the realm of attribute-based encryption, linear secret sharing, and monotone span programs, while systematizing all the relevant results in this research topics and proving a new lower bound.

We have introduced the concept of `U`-gates, while also providing a novel way of proving the nonexistence of ideal linear secret sharing schemes for certain monotone access structures, which was applied specifically on `U`-gates. We have shown the connection between `U`-gates and graph access structures.

We systematized the four main ways of constructing attribute-based encryption schemes from monotone Boolean circuits, while also trying to optimize one of them using backtracking. The relevant code is publicly available on GitHub [28].

That being said, our most important result is definitely the hybrid key-policy attribute-based encryption scheme for circuit access policies, based on bilinear pairings, with, we argue, the highest degree of generalization. That is, by using "universal circuits," each internal node of the circuit can compute *any* kind of monotone access structure (thanks to monotone span programs), thus not being limited anymore to specific, narrow cases, such as `AND`, `OR`, and (t/n)-threshold gates.

We have argued that our scheme is more efficient than the Hu–Gao scheme for "general circuits" and the Goyal et al. scheme for "access trees," since universal circuits are more expressive, they use more compact gates, and thus they lead to shorter decryption key sizes. We have implemented our scheme in `C++` and made it publicly available on GitHub [29].

6.1 Future Work

The search for KP-ABE schemes, based on bilinear pairings, running over more expressive computational models, is over, since now every circuit node is as expressive as possible. However, the issue of the decryption keys possibly getting exponential for some universal circuits still leaves room for improvement, that is, for optimizing the already existing circuit schemes, and specifically ours.

Moreover, new MSP-embedded gates can be created and implemented (e.g., for "CAS-nodes" [6], which most probably admit efficient MSPs). Also, the implementation of our proposed KP-ABE scheme can be improved by transpiling the `Python` code for inverting a `Galois` matrix into `C++`.

Bibliography

[1] Wei Dai, Yarkın Doröz, Yuriy Polyakov, Kurt Rohloff, Hadi Sajjadpour, Erkay Savaş, and Berk Sunar. Implementation and evaluation of a lattice-based key-policy abe scheme. *IEEE Transactions on Information Forensics and Security*, 13(5):1169–1184, 2017.

[2] Balaji Chandrasekaran and Ramadoss Balakrishnan. Attribute based encryption using quadratic residue for the big data in cloud environment. In *Proceedings of the International Conference on Informatics and Analytics*, pages 1–4, 2016.

[3] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. 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, 2006.

[4] Ferucio Laurențiu Țiplea and Constantin Cătălin Drăgan. Key-policy attribute-based encryption for boolean circuits from bilinear maps. In *Cryptography and Information Security in the Balkans: First International Conference, BalkanCryptSec 2014, Istanbul, Turkey, October 16-17, 2014, Revised Selected Papers 1*, pages 175–193. Springer, 2015.

[5] Peng Hu and Haiying Gao. A key-policy attribute-based encryption scheme for general circuit from bilinear maps. *Int. J. Netw. Secur.*, 19(5):704–710, 2017.

[6] Alexandru Ionita. Optimizing attribute-based encryption for circuits using compartmented access structures. *Cryptology ePrint Archive*, 2023.

[7] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

[8] André Weil. Sur les fonctions algébriques à corps de constantes fini. *Comptes Rendus de l'Académie des Sciences de Paris*, 210:592–594, 1940.

[9] John Tate. Wc-groups over p-adic fields. In *Séminaire Bourbaki, Vol. 1959/60, Exposé 190*, volume 7 of *Lecture Notes in Mathematics*, pages 265–277. Springer, 1961.

[10] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24*, pages 457–473. Springer, 2005.

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

[12] George Robert Blakley. Safeguarding cryptographic keys. In *Managing requirements knowledge, international workshop on*, pages 313–313. IEEE Computer Society, 1979.

[13] Maurice Mignotte. How to share a secret. In *Cryptography: Proceedings of the Workshop on Cryptography Burg Feuerstein, Germany, March 29–April 2, 1982 1*, pages 371–375. Springer, 1983.

[14] Charles Asmuth and John Bloom. A modular approach to key safeguarding. *IEEE transactions on information theory*, 29(2):208–210, 1983.

[15] Amos Beimel. Secure schemes for secret sharing and key distribution. *PhD thesis, Israel Institute of Technology, Technion*, 1996.

[16] Anna Gál. A characterization of span program size and improved lower bounds for monotone span programs. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 429–437, 1998.

[17] Amos Beimel, Anna Gál, and Mike Paterson. Lower bounds for monotone span programs. *Computational Complexity*, 6:29–45, 1996.

[18] Ventzislav Nikov and Svetla Nikova. New monotone span programs from old. *Cryptology ePrint Archive*, 2004.

[19] Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 568–588. Springer, 2011. See Appendix G.

[20] Zhen Liu, Zhenfu Cao, and Duncan S Wong. Efficient generation of linear secret sharing scheme matrices from threshold access trees. *Cryptology ePrint Archive*, 2010.

[21] Ingo Wegener. *The complexity of Boolean functions*. John Wiley & Sons, Inc., 1987.

[22] László Babai, Anna Gál, and Avi Wigderson. Superpolynomial lower bounds for monotone span programs. *Combinatorica*, 19(3):301–319, 1999.

[23] Robert Robere, Toniann Pitassi, Benjamin Rossman, and Stephen A Cook. Exponential lower bounds for monotone span programs. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 406–415. IEEE, 2016.

[24] Iulian Oleniuc and Alexandru Ioniță. Secret sharing limitations over boolean circuits. *Computer Science*, 33(1):97, 2025.

[25] Ernest F Brickell and Daniel M Davenport. On the classification of ideal secret sharing schemes. *Journal of Cryptology*, 4(2):123–134, 1991.

[26] Motahhareh Gharahi and Shahram Khazaei. Optimal linear secret sharing schemes for graph access structures on six participants. *Theoretical Computer Science*, 771:1–8, 2019.

[27] Alexandru Ioniță, Denis-Andrei Banu, and Iulian Oleniuc. Heuristic optimizations of boolean circuits with application in attribute-based encryption. *Procedia Computer Science*, 225:3173–3182, 2023.

[28] Iulian Oleniuc. Backtracking algorithm for finding MSPs for access structures. `https://github.com/gareth618/mspbkt`, 2024.

[29] Anonymous. A hybrid KP-ABE scheme for universal circuits. `https://anonymous.4open.science/r/kp-abe-universal-circuits/`, 2025.