UNIVERSITY OF TARTU
Faculty of Mathematics and Computer Science
Institute of Computer Science
Computer Science

Kairi Kangro

On Attribute-Based Encryption for Circuits from Multilinear Maps

Bachelor Thesis (6 EAP)

Supervisors: Helger Lipmaa, PhD
Sven Laur, PhD

Author:	" ......"	May 2013
Supervisor:	" ......"	May 2013
Supervisor:	" ......"	May 2013
Approved for defence		
Professor:	" ......"	May 2013

Tartu 2013

Contents

Introduction

In traditional public key encryption, one can send someone an encrypted message using the other person's public key, which can be looked up in a public database. This ensures that only the intended recipient can read the message, even if someone else manages to intercept the encrypted message. This is used for example by the Estonian police when sending encrypted e-mails - they use the public key associated with the recipient's ID-card to encrypt the message. However, traditional public key encryption also has its drawbacks. One of the main drawbacks is that when sending the same message to many different recipients, it needs to be encrypted for each recipient separately. Another one is that once a message has been encrypted, it is not possible to give someone else the ability to decrypt it without either re-encrypting it or giving the person in question the corresponding private key (which is usually not a good idea). This might be a problem when for example encrypting a file that many people might need to be able to access.

One possible way to solve this problem is to use Attribute-Based Encryption (ABE). Sahai and Waters first introduced the notion of ABE in [SW05]. The variant of ABE used in this work, the Key-Policy ABE (introduced by Goyal *et al* in [GPSW06]) allows an user to define during the encryption of a message a set of so-called attributes to be associated with the resulting ciphertext. The users of the system each receive a secret key associated with some function f defined on the sets of attributes. An user can decrypt a ciphertext only if their function outputs true on the attribute set that the ciphertext is associated with. This can be used for example to implement something like file permissions on an encrypted file system: each file could be associated with the username of its owner and the name(s) of some user group(s), and each user would receive a secret key for a function that would evaluate to true only if either their username or user group belonged to the attributes associated with a file.

Even though there have been many advances in multiple directions since the introduction of ABE, the problem of expanding the set of allowable functions has proved to be hard. The construction in [GPSW06] proved security for the set of functions representable by polynomial-size boolean formulae (alternatively, functions computable by logarithmic depth circuits, see [AB09]), which has remained the best known result until recently, when Garg *et al* presented a construction in [GGH$^+$13] that achieves security for general polynomial size circuits, which are widely believed to be much more powerful than boolean formulae ([AB09]). Garg *et al* use multilinear maps to achieve their construction, citing the existence of an attack called the backtracking attack as a reason why bilinear maps, which are commonly used for ABE, might not be enough.

The aim of this work is to introduce and slightly improve the construction in [GGH$^+$13] while giving most of the background theory needed to understand the construction. Our contribution lies in lessening the amount of key components required and ensuring that decryption works always instead of with high probability. A full security proof is given for the improved construction.

This work consists of four chapters. Chapter 1 gives an overview of some notions together with the relevant results that are necessary for understanding the construction given in later chapters. Topics covered include group theory, boolean circuits and time complexity of algorithms. Chapter 2 introduces the notion of Attribute-Based Encryption and gives an overview of relevant results. An attack on an existing Attribute-Based Encryption scheme called the backtracking attack is also introduced, giving the motivation of the construction presented in the next chapter. In Chapter 3, the construction of [GGH$^+$13] is given along with our improvements. In Chapter 4, a full security proof of our improved construction is given, together with the necessary security definitions.

Chapter 1

Preliminaries

In this chapter, an overview of some important notions for understanding the construction in Chapter 3 is given. All the results and definitions in this chapter are standard and are presented here in order for the thesis to be self-containing. See [Kil05] and [AB09] for more details on the topics presented here.

1.1 Groups and Generators

The definitions and results in this section follow the ones in [Kil05].

Definition 1.1.1. A non-empty set G together with a binary operation $\cdot : G \times G \to G$ (denoted by $(G, \cdot)$) is called a *group* if it satisfies the following properties:

- $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all $a, b, c \in G$ (associativity).
- There exists an element $e \in G$, called the *identity element*, such that $a \cdot e = e \cdot a = a$ for all $a \in G$.
- For every element $a \in G$ there exists an element $b \in G$, called the *inverse element* of a, such that $a \cdot b = b \cdot a = e$.

If the binary operation is clear from the context, the notation „group G” is used instead of „group $(G, \cdot)$”. Note that if $b \in G$ is the inverse element of $a \in G$, then a is also the inverse element of b in group G.

Proposition 1.1.1. *The identity element of a group is unique.*

Proof. Let $(G, \cdot)$ be a group and let $e, e' \in G$ be identity elements of the group. Then, by the identity element property, we have that $e = e \cdot e' = e'$. Therefore, the identity element of a group is unique. $\square$

Proposition 1.1.2. *Every element of a group has exactly one inverse element.*

Proof. Let $(G, \cdot)$ be a group and let $a \in G$ be an element of that group. By the definition of a group, a has to have at least one inverse element. Now suppose that $b, b' \in G$ are both inverse elements of a. Then, using the associativity property, we get $b = b \cdot e = b \cdot (a \cdot b') = (b \cdot a) \cdot b' = e \cdot b' = b'$. Therefore, every element $a \in G$ has exactly one inverse element. $\square$

If the binary operation of a group is multiplication, then the group is called a *multiplicative group* and the unit element is denoted by 1 and the inverse element of element a is denoted by a^{-1}. If the binary operation of a group is addition, then the group is called an *additive group* and the unit element is denoted by 0 and the inverse element of element a is denoted by $-a$.

Example 1.1.1. The set of integers $\mathbb{Z}$ is a group with respect to ordinary addition: $a+(b+c)=(a+b)+c$, $a+0=0+a=a$ and $a+(-a)=(-a)+a=0$ for all integers a,b,c.

Definition 1.1.2. Let G be a multiplicative group. The integer *powers* of an element $g \in G$ are defined as follows:

- $g^0 = 1$
- $g^m = \underbrace{g \cdot g \cdot \ldots \cdot g}_{m \text{ multiplicands}}$ for all positive integers m.
- $g^{-m} = (g^{-1})^m$ for all positive integers m.

Lemma 1.1.1. *Let G be a multiplicative group and $a, b \in G$ elements of that group. Then $(a \cdot b)^{-1} = b^{-1} \cdot a^{-1}$.*

Proof. This follows from the fact that

$$\begin{aligned}(a\cdot b)\cdot(b^{-1}\cdot a^{-1}) &= a\cdot b\cdot b^{-1}\cdot a^{-1} = a\cdot 1\cdot a^{-1} = a\cdot a^{-1} = 1 \text{ and}\\ (b^{-1}\cdot a^{-1})\cdot(a\cdot b) &= b^{-1}\cdot a^{-1}\cdot a\cdot b = b^{-1}\cdot 1\cdot b = b^{-1}\cdot b = 1.\end{aligned}$$

□

Proposition 1.1.3. *Let G be a multiplicative group and $g \in G$ an element of that group. Then $g^{m+n} = g^m \cdot g^n$ and $(g^m)^n = g^{mn}$ for all integers m, n.*

Proof. If $m = 0$ then

$$\begin{aligned}g^{0+n} &= g^n = 1\cdot g^n = g^0\cdot g^n \text{ and}\\ (g^0)^n &= 1^n = 1 = g^{0n}.\end{aligned}$$

If $m = 0$ then

$$\begin{aligned}g^{m+0} &= g^m = g^m\cdot 1 = g^m\cdot g^0 \text{ and}\\ (g^m)^0 &= 1 = g^{0m}.\end{aligned}$$

If $m > 0$ and $n > 0$ then

$$\begin{aligned}g^{m+n} &= \underbrace{g\cdot g\cdots g}_{m+n \text{ multiplicands}} = \underbrace{g\cdot g\cdots g}_{m \text{ multiplicands}} \cdot \underbrace{g\cdot g\cdots g}_{n \text{ multiplicands}} = g^m\cdot g^n \text{ and}\\ (g^m)^n &= \underbrace{g^m\cdot g^m\cdots g^m}_{n \text{ multiplicands}} = \underbrace{(\underbrace{g\cdot g\cdots g}_{m \text{ multiplicands}})\cdot(\underbrace{g\cdot g\cdots g}_{m \text{ multiplicands}})\cdots(\underbrace{g\cdot g\cdots g}_{m \text{ multiplicands}})}_{n \text{ multiplicands}}\\ &= \underbrace{g\cdot g\cdots g}_{mn \text{ multiplicands}} = g^{mn}.\end{aligned}$$

If $m < 0$ and $n < 0$ then

$$g^{m+n} = g^{-(|m|+|n|)} = (g^{-1})^{|m|+|n|} = (g^{-1})^{|m|} \cdot (g^{-1})^{|n|} = g^m \cdot g^n \text{ and}$$

$$(g^m)^n = (((g^{-1})^{|m|})^{-1})^{|n|} = \underbrace{((g^{-1})^{|m|})^{-1} \cdot ((g^{-1})^{|m|})^{-1} \cdots ((g^{-1})^{|m|})^{-1}}_{|n| \text{ multiplicands}}$$

$$= \underbrace{(\underbrace{g^{-1} \cdot g^{-1} \cdots g^{-1}}_{|m| \text{ multiplicands}})^{-1} \cdot (\underbrace{g^{-1} \cdot g^{-1} \cdots g^{-1}}_{|m| \text{ multiplicands}})^{-1} \cdots (\underbrace{g^{-1} \cdot g^{-1} \cdots g^{-1}}_{|m| \text{ multiplicands}})^{-1}}_{|n| \text{ multiplicands}}$$

$$= \underbrace{(\underbrace{g \cdot g \cdots g}_{|m| \text{ multiplicands}}) \cdot (\underbrace{g \cdot g \cdots g}_{|m| \text{ multiplicands}}) \cdots (\underbrace{g \cdot g \cdots g}_{|m| \text{ multiplicands}})}_{|n| \text{ multiplicands}}$$

$$= \underbrace{g \cdot g \cdots g}_{|m| \cdot |n| \text{ multiplicands}} = g^{|mn|} = g^{mn}.$$

The rest of the cases are analogous. □

Definition 1.1.3. A multiplicative group G is called a *cyclic group* if there exists an element $g \in G$, called the *generator* of G, such that $G = \{g^n | n \in \mathbb{Z}\}$ (i.e, all elements of G can be written as g^a for some integer a).

Note that by proposition 1.1.3 , a cyclic group is always commutative, since $g^a \cdot g^b = g^{a+b} = g^{b+a} = g^b \cdot g^a$ for all integers a, b.

Definition 1.1.4. A group $(G, \cdot)$ is called *finite* if the number of elements in G is finite. In this case, the number of elements in G is called the *order* of the group.

Proposition 1.1.4. *If G is a finite cyclic group with order n and generator g, then n is the smallest natural number for which $g^n = 1$ and $G = \{1, g, g^2, \ldots, g^{n-1}\}$.*

Proof. Since G is a cyclic group, $G = \{g^t | t \in \mathbb{Z}\}$. Since G is also finite, there have to exist $k, l \in \mathbb{Z}$, $k > l$ such that $g^k = g^l$ (otherwise all powers of g would be different and therefore G would be infinite). Multiplying that equation by g^{-l}, we get $g^{k-l} = 1$, where $k - l > 0$ is a natural number. Now let m be the smallest natural number such that $g^m = 1$. We will show that $G = \{1, g, g^2, \ldots, g^{m-1}\}$. For that, we need to show two things: first, that $g^t \in \{1, g, g^2, \ldots, g^{m-1}\}$ for all integers t, and second, that all the elements in the set are different.

Let us show that $g^t \in \{1, g, g^2, \ldots, g^{m-1}\}$ for all integers t. Dividing t by m, we get $t = q \cdot m + r$ for some integers q and r with $0 \leqslant r < m$. Therefore, $g^t = g^{q \cdot m + r} = (g^m)^q \cdot g^r = 1^q \cdot g^r = g^r \in \{1, g, g^2, \ldots, g^{m-1}\}$.

Now let us show that the elements $1, g, g^2, \ldots, g^{m-1}$ are all different. Suppose that there exist some integers $0 \leqslant t < u < m$ such that $g^t = g^u$. Then, multiplying by g^{-u}, we get $g^{t-u} = 1$, where $0 < t - u < m$, which is contradictory to the choice of m. Therefore $G = \{1, g, g^2, \ldots, g^{m-1}\}$, and since the order of G is n, then $m = n$ and $G = \{1, g, g^2, \ldots, g^{n-1}\}$. □

Example 1.1.2. $\mathbb{Z}_5 \backslash \{0\} = \{1, 2, 3, 4\}$ with multiplication defined modulo 5 is a cyclic finite group with 2 as a generator: $1 = 2^0$, $2 = 2^1$, $3 = 2^3$, $4 = 2^2$.

Proposition 1.1.5. *For any element g^a in a finite cyclic group G with generator g and order p, $(g^a)^{-1} = (g^a)^{p-1}$.*

Proof. Since $g^p = 1$, then $g \cdot g^{p-1} = g^{p-1} \cdot g = 1$, and therefore $g^{-1} = g^{p-1}$. Then, by proposition 1.1.3, $(g^a)^{-1} = g^{-a} = (g^{-1})^a = (g^{p-1})^a = (g^a)^{p-1}$. □

1.2 Circuits

In this section, an overview of circuits is given. Since the construction in Chapter 3 is given for monotone layered circuits, the necessary results for transforming a general circuit into one are given as well. See [AB09] and references therein for more detail.

1.2.1 Definitions and Examples

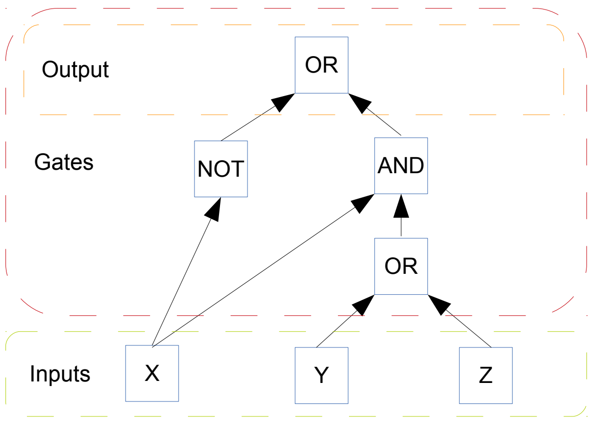

Figure 1.1: A Boolean circuit with three inputs and one output. Corresponds to the boolean formula $X \wedge (Y \vee Z) \vee \neg X$.

Definition 1.2.1. A *Boolean circuit* (Figure 1.1) with $n \in \mathbb{N}$ inputs and one output is a directed acyclic graph that satisfies the following conditions:

- It has n vertices with no incoming edges (called the *inputs*) and one vertex with no outgoing edges (called the *output*)
- All the other vertices except the inputs are called *gates* and labelled with either AND, OR or NOT. The AND and OR gates have exactly two incoming edges, and the NOT gates have exactly one incoming edge.

The *size* of the circuit is said to be the number of gates it has.

In the rest of this work, we will use „circuit" to mean „boolean circuit with one output". Since a circuit is acyclic, there can only be a finite number of different paths from one vertex to another, and therefore we can define the concept of depth as follows.

Definition 1.2.2. The *depth* of an input is defined to be 1. The depth of a gate is equal to the length (number of edges) of the longest path from any input plus 1. The depth of a circuit is equal to the depth of its output.

Definition 1.2.3. The *evaluation* of a circuit f with $n \in \mathbb{N}$ inputs labelled with the numbers $1, \ldots, n$ in some way on the input $x \in \{0,1\}^n$ is defined as follows:

- The output value of input number i is the ith bit of x.
- The output value of an AND gate is defined to be 1 if the output values of the vertices connected to the incoming edges of the AND gate are both 1, and 0 otherwise.

- The output value of an OR gate is defined to be 0 if the output values of the vertices connected to the incoming edges of the AND gate are both 0, and 1 otherwise.
- The output value of a NOT gate is defined to be 1 if the output value of the vertex connected to the incoming edge of the NOT gate is 0, and 0 otherwise.
- The output value of the circuit f on input x is defined to be the output value of the output of f.

Since the circuit does not have cycles, the output values of all vertices are well-defined in the sense that by evaluating the outputs of vertices in order of increasing depth, the output values of the vertices connected to the incoming edges of a gate will be determined by the time the gate itself is evaluated. It is easy to see that in this way, every circuit implements a function from $\{0,1\}^n$ to $\{0,1\}$.

1.2.2 Monotone Circuits

Definition 1.2.4. A circuit is said to be *monotone* if it has only AND and/or OR gates, but no NOT gates.

Since the construction in Chapter 3 is given for monotone circuits, we would like to be able to transform a general circuit into a monotone one. Unfortunately, this is not always possible, but it is possible to convert any circuit to an equivalent circuit that has NOT gates only directly after the inputs:

Theorem 1.2.1. *Every circuit C can be converted into an equivalent (i.e., outputs the same value given the same input) circuit C' that has NOT gates only at depth 2.*

Proof. Let the depth of C be d. First, we will use De Morgan's rule to create a circuit $\tilde{C}$ from C such that every AND and OR gate in C has a corresponding gate in $\tilde{C}$ that outputs the negation of that gate. De Morgan's rule for circuits states that inverting the output of an AND gate is equivalent to replacing the AND gate with an OR gate and inverting both of its inputs, and vice versa. Also, inverting the input of a NOT gate is equivalent to inverting its output. We construct $\tilde{C}$ from C by adding a negation gate after every outgoing edge of every input and replacing all AND gates with OR gates and vice versa (Figure 1.2).

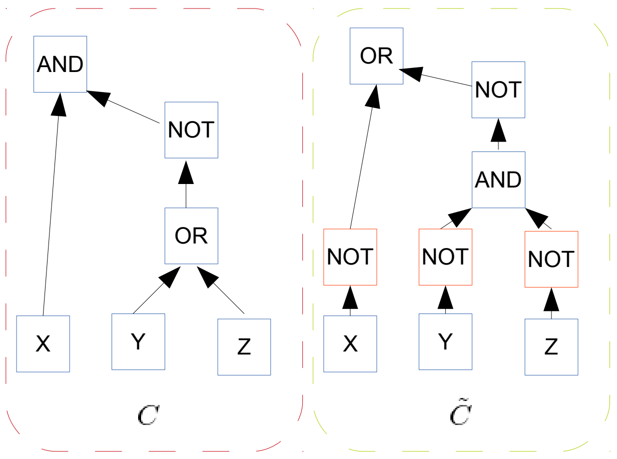

Figure 1.2: Constructing $\tilde{C}$ from C.

Obviously there is a one-to-one correspondence between gates with depth j in C and gates with depth $j+1$ in $\tilde{C}$, $j \in \{2, \dots, d\}$. We will prove by induction on the gate depth j that all gates in $\tilde{C}$ output the negation of the corresponding gate in C.

First, we look at gates at depth 2 in C. Then by De Morgan's rule, the corresponding gates at depth 3 in $\tilde{C}$ will output the negation of the gates in C, because we inverted all of their inputs and replaced AND gates with OR gates and vice versa. Now assume that for gates up to depth $k < d$ in C, the corresponding gates in $\tilde{C}$ output their negation. Then, for gates at depth $k+1$ in C, the corresponding gates at depth $k+2$ in $\tilde{C}$ output their negation as well, since they receive their inputs from gates with depth less than $k+2$, and by the induction assumption, the outputs of these gates are the inverse of the outputs of the corresponding gates in C, and we have changed AND gates to OR gates and vice versa. Therefore for every gate in C there is a corresponding gate in $\tilde{C}$ that computes its negation, and vice versa (excluding the NOT gates at depth 2 in $\tilde{C}$).

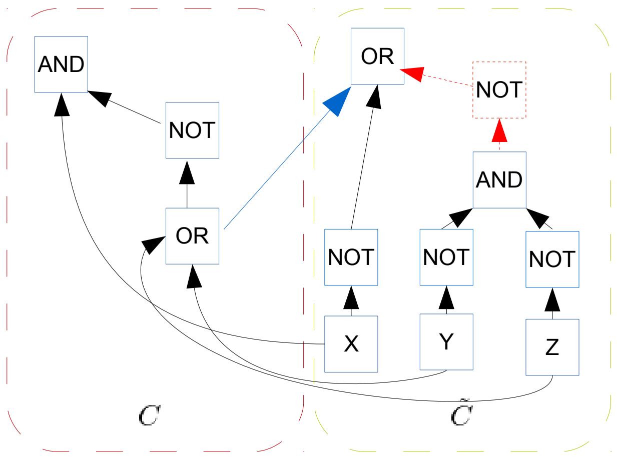

Figure 1.3: Merging inputs and eliminating a NOT gate.

Now we start constructing C'. First, if the output of C (and hence also $\tilde{C}$) is a NOT gate, eliminate the NOT gate from both C and $\tilde{C}$ and designate the gate that was connected to the output in $\tilde{C}$ as the output of C'. Otherwise, designate the output of C as the output of C'. Repeat, swapping the roles of $\tilde{C}$ and C, until the output of C' is not a NOT gate. Second, merge the inputs of C and $\tilde{C}$ (since they have the same inputs). Next, for every NOT gate (in decreasing order of depth) A in either C or $\tilde{C}$ with depth greater than 2 (i.e., the NOT gate gets its input from a gate, not a circuit input), let B be the gate A gets its input from. Find the gate (or the corresponding input, if B happens to be one of the added NOT gates) B' in either C if A is in $\tilde{C}$ or $\tilde{C}$ if A is in C that computes the negation of B. Then remove the A with all of its connecting edges, and create a new edge from B' to every gate that received one of their inputs from A (Figure 1.3). This will remove all NOT gates with depth greater than 2. Finally, remove all gates that are not the output but have no outgoing edges (repeat until the output is the only gate with no outgoing edges). Then C' will have NOT gates only at depth 2, if at all. The equivalence of C and C' follows from the construction of C'. Note that the depth of C' without counting the NOT gates is the same as the depth of C without counting the NOT gates. $\square$

1.2.3 Layered Circuits

Definition 1.2.5. A circuit with depth d is called *layered* if every gate with depth j receives all of its inputs from gates/inputs with depth $j - 1$ for all $j \in \{2, \dots, d\}$.

Since the construction in Chapter 3 uses layered circuits, we show here how to change an arbitrary circuit into a layered one while increasing the size by at most a polynomial factor.

Theorem 1.2.2. *For every circuit C with size s there exists a layered circuit C' that has the same depth and whose size at most $s(s-1)$.*

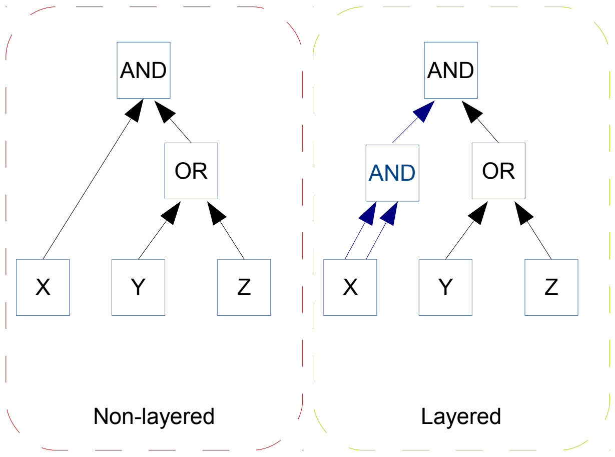

Figure 1.4: Converting a non-layered circuit into a layered one.

Proof. Let the size of C be s and depth of C be d. Consider a gate A that has depth j. Since it has depth j, at most one of the gates/inputs connected to the incoming edges of A can have depth not equal to $j-1$ (if all of them had depth less than $j-1$, the depth of A would be less than j). Suppose that one of them, called B, has depth $k < j-1$. To remedy this, we add $j-1-k$ dummy AND gates between B and A, so that the first AND gate would get both of its inputs from B (and thus have depth $k+1$), the second AND gate would get both of its inputs from the first AND gate (and thus have depth $k+2$),. . . ,A would get its input from the AND gate from the $(j-1-k)$th AND gate, which has depth $j-1$ (Figure 1.4). Therefore, A and all added AND gates satisfy the layered circuit condition. Notice that this does not change the depth of neither A nor B, so the depth of the circuit is still d. We repeat this procedure for every gate in C. Let C' denote the resulting circuit. Since $j \leqslant d$ and $k \geqslant 1$, $j-1-k \leqslant d-2$ for all gates in C, and therefore we added at most $(d-2)s$ gates. This means that the size of C' is at most $(d-2)s+s = (d-1)s < (s-1)s$ (since obviously $d < s$), which is polynomial in s, and has depth d. □

1.3 Complexity of Algorithms

For a cryptographic scheme to be of practical use, it is necessary for it to be efficient in some sense. For this, we need a formal notion of efficiency.

Definition 1.3.1. The *worst case time complexity* $T(n)$ of an algorithm is the maximum number of elementary operations it does on an input with length n.

Exactly what operations are considered elementary operations depends on the computational model, but the class of efficient algorithms as defined below is believed to be the same for all physically realizable computational models (see [AB09] for details). Intuitively, an algorithm is considered efficient if its time complexity is bounded above by some polynomial for large enough inputs.

Definition 1.3.2. An algorithm is said to run in *polynomial time* if there exists a polynomial $p(\cdot)$ and positive constants $c, N \in \mathbb{N}$ such that for all inputs with length $n > N$, $T(n) \leqslant cp(n)$. An algorithm is considered *efficient* if and only if it runs in polynomial time.

For the sake of simplicity, most cryptographic schemes use a *security parameter* $\lambda \in \mathbb{N}$ instead of the input length n to define the complexity of algorithms. In this case, the algorithms of the scheme are provided λ in unary notation (i.e., $\underbrace{11\cdots1}_{\lambda \text{ ones}}$, denoted by 1^λ) as an input.

Chapter 2

Attribute-Based Encryption with Bilinear Maps

In this chapter, the definitions for bi- and multilinear maps are given. The notion of Attribute-Based Encryption and the definition for Key-Policy Attribute-Based Encryption is given. An attack on an existing ABE scheme ([GPSW06]) called the backtracking attack is explained.

2.1 Bi- and Multilinear Maps

In cryptography, bilinear maps are defined on groups instead of on vector spaces as is usual in algebra.

Definition 2.1.1. Let $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_3$ be cyclic multiplicative groups with the same order and g_1, g_2, g_3 be their respective generators. Then $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ is called a *bilinear map* if $e(g_1^a, g_2^b) = g_3^{ab}$ for all $a, b \in \mathbb{Z}$.

In cryptographic applications, it is also assumed that e is efficiently computable. One of the earliest cryptographic schemes based on bilinear maps was an Identity-Based Encryption scheme by Boneh and Franklin in [BF01]. Since then, bilinear maps have been used with great success to construct many different cryptographic schemes, including Attribute-Based Encryption schemes.

A natural extension of bilinear maps are so-called multilinear maps:

Definition 2.1.2. Let $\mathbb{G}_1, \mathbb{G}_2, \ldots, \mathbb{G}_{k+1}$ be cyclic multiplicative groups with the same order and $g_1, g_2, \ldots, g_{k+1}$ be their respective generators. Then $e : \mathbb{G}_1 \times \mathbb{G}_2 \times \ldots \times \mathbb{G}_k \to \mathbb{G}_{k+1}$ is called a *k-multilinear map* if $e(g_1^{a_1}, g_2^{a_2}, \ldots, g_k^{a_k}) = g_{k+1}^{a_1 a_2 \cdots a_k}$ for all $a_1, \ldots, a_k \in \mathbb{Z}$.

Boneh and Silverberg showed in [BS03] that multilinear maps would have many interesting applications in cryptography. However, they also showed that cryptographically useful multilinear maps might be hard to find, and until recently, no such multilinear maps were known. But in [GGH12], Garg Gentry and Halevi presented a mechanism that is the equivalent of multilinear maps for many applications, therefore giving new motivation for using multilinear maps in construction cryptographic schemes. The construction presented in this work can also be directly translated into the framework given in [GGH12] (see [GGH$^+$13] for details).

2.2 Attribute-Based Encryption

The notion of Attribute-Based Encryption (ABE) was introduced by Sahai and Waters in 2005 [SW05] and expanded upon by Goyal *et al* in 2006 [GPSW06], defining two variants of ABE: Key Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE). In a KP-ABE system, ciphertexts are associated with an assignment x of boolean variables (called *attributes*) and users are issued secret keys that are associated with a boolean function f from some set of allowable functions $\mathcal{F}$. An user with a secret key for f should be able to able to decrypt a ciphertext associated with x if and only if $f(x) = 1$. In a CP-ABE system, ciphertexts are associated with a boolean function f from some set of allowable functions $\mathcal{F}$ and users are issued secret keys associated with some assignment x of attributes. An user with a secret key associated with x should be able to decrypt a ciphertext associated with f if and only if $f(x) = 1$.

Example 2.2.1. Suppose the faculty of Mathematics and Computer Science wanted to use KP-ABE to encrypt some files. The allowed set of attributes might be Student, Lecturer, CS, Statistics, Math. A student in computer science would receive the secret key to the access structure "Student AND CS", a lecturer would receive the secret key to the access structure "Lecturer AND {institute to which lecturer belongs}". To encrypt a file so that all lecturers could decrypt it, it would be associated with the attributes {Lecturer, CS, Statistics, Math}. To encrypt a file so that only people belonging to the Institute of Computer Science could decrypt, it would be associated with the attributes {Student, Lecturer, CS}.

Since the introduction of ABE, there have been many advances in multiple directions. However, the set of allowable functions $\mathcal{F}$ has remained rather limited: in terms of circuit classes, the best result was achieved by Goyal *et al* in [GPSW06], whose construction achieved security for circuits with depth $\log n$, where n is the number of inputs, which is equivalent to the class of functions representable by polynomial size boolean formulas. Garg *et al* cite in [GGH$^+$13] the existence of a so-called backtracking attack as a possible reason why it might not be possible to achieve (KP-)ABE for general circuits using only one bilinear map .

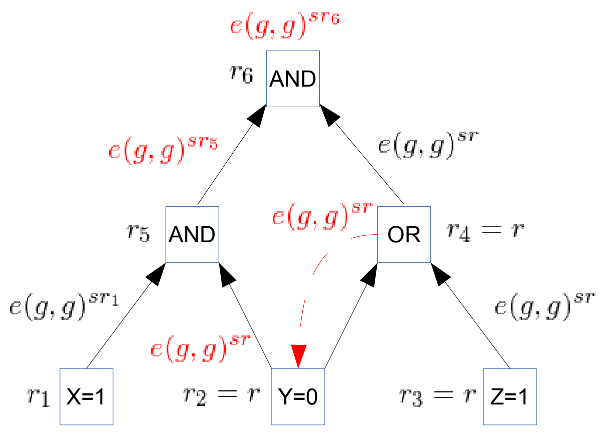

Figure 2.1: Example of the backtracking attack where it enables the attacker to decrypt the ciphertext. The values in red can only be computed due to the attack.

To illustrate the nature of the backtracking attack, we use the construction from [GPSW06] as an example. In their construction, keys are elements in a cyclic group $\mathbb{G}$

with order p, and there exists a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$, where $\mathbb{G}_t$ is also a cyclic group of order p. The access structures are circuits with only one outgoing edge allowed per vertex. Every input w is associated with a value $r_w \in \mathbb{Z}_p$, and the secret keys are chosen in such a way that $e(g, g)^{sr_w}$, where $s \in \mathbb{Z}_p$ is a randomizing factor used to encrypt the ciphertext we wish to decrypt, is computable only if the attribute associated with w in the attribute vector $x \in \{0, 1\}^n$ is set to 1 (note that there may be multiple inputs associated with one attribute, and therefore we can assume that each input has at most one outgoing edge). Every gate A is associated with a value r_A such that $e(g, g)^{sr_A}$ is computable if A outputs 1 on input x, and r_{output} is set to such a value that knowing $e(g, g)^{sr_{output}}$ will enable us to decrypt the ciphertext. This means that if gate A is an OR gate and receives its inputs from gates (or inputs) B and C, it must be possible to compute $e(g, g)^{sr_A}$ knowing either of $e(g, g)^{sr_B}$ or $e(g, g)^{sr_C}$. In [GPSW06], this is done by setting $r_A = r_B = r_C$. But this means that if (say) B outputs 1, then even if C outputs 0, we will still learn the value $e(g, g)^{sr_C} = e(g, g)^{sr_B}$. If every gate and input has exactly one outgoing edge, as is the case with boolean formulas, then this does not have any serious consequences, because C is only connected to A, which evaluated to 1 anyway. But if we allowed our gates to have more than one outgoing edge, and C was connected to another gate D as well, an adversary could use its knowledge of $e(g, g)^{sr_C}$ to pretend in the computation for gate D that C outputs 1, and in the worst case, end up being able to decrypt the ciphertext (Figure 2.1).

The construction in [GGH$^+$13] and therefore the construction in this work aims to expand the allowable set of functions to polynomial size boolean circuits with an arbitrary number of outgoing connections allowed for each vertex. The construction is of the KP-ABE variety, and therefore we will give a formal definition of a KP-ABE scheme for bounded circuits here.

Definition 2.2.1. A *KP-ABE scheme for polynomially bounded circuits* is a set that consists of the following four algorithms

- **Setup**($1^\lambda, n, \ell$). The setup algorithm takes as input the unary representation of the security parameter λ, the length n of the possible attribute vectors and a bound ℓ on the circuit depth, where n and ℓ are polynomially bounded by λ. It outputs the public parameters PP and a master key MSK which is given to a trusted authority.

- **Encrypt**(PP, x, M). The encryption algorithm takes as input the public parameters PP, a bit string $x \in \{0, 1\}^n$ representing the set of attributes the message should be encrypted for, and a message M. It outputs a ciphertext CT.

- **KeyGen**(MSK, f). The key generation algorithm takes as input the master key MSK and a description of a polynomial-size (in the security parameter λ) circuit f, where the depth of f is at most ℓ. The algorithm outputs a private key SK.

- **Decrypt**(SK, CT). The decryption algoritm takes as input a secret key SK and ciphertext CT. The algorithm attempts to decrypt and outputs a message M if successful; otherwise, it outputs a special symbol $\perp$.

that satisfy the following conditions:

- The algorithms are efficiently (i.e., in polynomial time) computable

- For all messages M, strings $x \in \{0, 1\}^n$ and depth ℓ circuits f for which $f(x) = 1$, if $Encrypt$(PP, x, M) outputs CT and $KeyGen$(MSK, f) outputs SK, where PP, MSK were generated by the setup algorithm, then $Decrypt$(SK, CT) $= M$.

The security definition for KP-ABE schemes will be given in Chapter 4.

Chapter 3

The Construction

In this chapter, we first define some necessary assumptions and notations for introducing the KP-ABE construction in [GGH$^+$13]. Then we give a presentation of that construction. Finally, we suggest some modifications that could be made to improve the construction in [GGH$^+$13]. Specifically, we show that it is possible to reduce the amount of necessary secret key components by one for every gate.

3.1 Assumptions and Notations

Since by Theorem 1.2.1 and Theorem 1.2.2, every circuit can be transformed to an equivalent almost monotone layered circuit without increasing the depth of the circuit (if not counting the NOT gates at depth 2), and we can eliminate the NOT gates at depth 2 by adding a new attribute to encode the negation of every existing attribute and instructing honest encryptors to set exactly one of every pair to one, we can assume without loss of generality that all of our circuits are monotone and layered.

We will now define some formal notation to describe a circuit f with n inputs and q gates that will be used in describing the construction in the following section. This notation mostly follows the one used in [GGH$^+$13], which in turn follows the notation used in a paper by Bellare, Hoang and Rogaway ([BHR12]). We number our vertices with numbers 1 to $n+q$, with 1 to n being assigned to inputs, $n+q$ being assigned to the output and the remaining numbers being assigned to the remaining gates so that a gate with greater depth will always have a greater number than a gate with lower depth, and define the sets Inputs $= \{1, \ldots, n\}$, Gates $= \{n+1, \ldots, n+q\}$ and Vertices $=$ Inputs $\cup$ Gates. Let A : Gates $\to$ Vertices$\setminus\{n+q\}$ be a function such that $A(w)$ equals the number of the vertex connected to one incoming edge of the gate w and B : Gates $\to$ Vertices$\setminus\{n+q\}$ be a function such that $B(w)$ equals the number of the vertex connected to the other incoming edge of the gate w, with $A(w) \leqslant B(w)$ for every gate w. Let $\texttt{GateType}$: Gates $\to \{AND, OR\}$ be a function such that $\texttt{GateType}(w)$ returns the type of gate w. Then the circuit f can be described with the five tuple $(n, q, A, B, \texttt{GateType})$. Note that since our circuit is layered, if the depth of a gate w is j, then the depths of the vertices $A(w)$ and $B(w)$ is $j-1$. We will use $f(x)$ to denote the output of the circuit f on input x and $f_w(x)$ to denote the output of vertex w in circuit f on input x.

Since using only one bilinear map does not seem to be enough to avoid the backtracking attack, the construction of Garg *et al* in [GGH$^+$13] uses a set of bilinear maps that can be seen as implementing multilinear maps. Namely, we assume the existence of an efficient group generator $\mathcal{G}$ that takes as input a security parameter λ and a positive integer k, and outputs a sequence of groups $\vec{\mathbb{G}} = \{\mathbb{G}_1, \ldots, \mathbb{G}_k\}$ each of a large prime order $p > 2^\lambda$. For the sake of efficiency, we assume that the length of p in bits is polynomial in λ. Let g_i denote the generator of the group $\mathbb{G}_i$, $i \in \{1, \ldots, k\}$ (we assume this is known from the group description). We then assume that there exists a set of bilinear maps $\{e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j} | i, j \geqslant 1; i + j \leqslant k\}$. When the indices are clear from the context, we will abuse notation and write e instead of $e_{i,j}$. It is easy to see that for example the function $e' : \underbrace{\mathbb{G}_1 \times \ldots \times \mathbb{G}_1}_{k-1} \to \mathbb{G}_k$ defined as

$$e'(g_1^{\alpha_1}, \ldots, g_1^{\alpha_{k-1}}) = e(e(\ldots e(e(g_1^{\alpha_1}, g_1^{\alpha_2}), g_1^{\alpha_3}) \ldots, g_1^{\alpha_{k-2}}), g_1^{\alpha_{k-1}}) = g_k^{\alpha_1 \alpha_2 \cdots \alpha_{k-1}}$$

implements a multilinear map.

3.2 The Garg-Gentry-Halevi-Sahai-Waters Construction

We will now describe the GGHSW KP-ABE scheme of [GGH+13] for polynomially bounded circuits. In the GGHSW construction, it is assumed for ease of exposition that all of the circuits have depth ℓ, where ℓ is the maximal circuit depth defined during setup. We will explain how to handle circuits with depth less than ℓ after the construction.

3.2.1 The Algorithms

Setup$(1^\lambda, n, \ell)$. The setup algorithm takes as an input a security parameter λ, the number of boolean inputs n and the maximum depth ℓ of the circuits. It then runs $\mathcal{G}(1^\lambda, k = \ell + 1)$ to obtain the group descriptions $\vec{\mathbb{G}} = \{\mathbb{G}_1, \ldots, \mathbb{G}_k\}$ of prime order $p > 2^\lambda$ together with their generators $g_1, \ldots, g_k$. We let $g = g_1$. Next, it chooses at random $\alpha \in \mathbb{Z}_p$ and $h_1, \ldots, h_n \in \mathbb{G}_1$. The public parameters PP consist of the group sequence description plus $g_k^\alpha, h_1, \ldots, h_n$. The master secret key MSK is g_{k-1}^α.

Encrypt(PP, $x \in \{0,1\}^n$, $M \in \{0,1\}$). The encryption algorithm takes as input the public parameters PP, an attribute vector $x \in \{0,1\}^n$ and a message bit M. The encryption algorithm chooses a random $s \in \mathbb{Z}_p$. If $M = 1$, it sets C_M to $(g_k^\alpha)^s$. Otherwise, it sets C_M to a random group element in $\mathbb{G}_k$. Next, let S be the set of such i that $x_i = 1$. Then the ciphertext is created as

$$\mathrm{CT} = (C_M, g^s, \forall i \in S \; C_i = h_i^s).$$

KeyGen(MSK, $f = (n, q, A, B, \texttt{GateType})$). The key generation algorithm takes as input the master secret key MSK and the description f of a circuit. The algorithm chooses random $r_1, \ldots, r_{n+q} \in \mathbb{Z}_p$, where we think of r_w as being associated with vertex w. It produces a header component using the master secret key as

$$K_H = g_{k-1}^{\alpha - r_{n+q}}.$$

Next, it generates key components for every vertex w. The structure of the key components depends on whether w is an input, an AND gate or an OR gate. We will describe how the algorithm generates the key components in each case.

- *Input.*
  If $w \in$ Inputs, the algorithm chooses a random $z_w \in \mathbb{Z}_p$ and creates the key components as
  $$K_{w,1} = g^{r_w} h_w^{z_w}, K_{w,2} = g^{-z_w}.$$

- *OR gate.*
  If $w \in$ Gates and $\texttt{Gatetype}(w) =$ OR, then the algorithm chooses random $a_w, b_w \in \mathbb{Z}_p$ and creates the key components as
  $$K_{w,1} = g^{a_w}, K_{w,2} = g^{b_w}, K_{w,3} = g_j^{r_w - a_w r_{A(w)}}, K_{w,4} = g_j^{r_w - b_w r_{B(w)}},$$
  where j is the depth of w.

- *AND gate.*
  If $w \in$ Gates and $\texttt{Gatetype}(w) =$ AND, then the algorithm chooses random $a_w, b_w \in \mathbb{Z}_p$ and creates the key components as
  $$K_{w,1} = g^{a_w}, K_{w,2} = g^{b_w}, K_{w,3} = g_j^{r_w - a_w r_{A(w)} - b_w r_{B(w)}},$$
  where j is the depth of gate w.

The secret key SK consists of the description of f, the header component K_H and the key components for each vertex of f.

Decrypt(SK, CT). Given a secret key SK associated with a circuit $f = (n, q, A, B, \texttt{GateType})$ and a ciphertext associated with the attribute vector x, decryption should be possible if (and only if) $f(x) = 1$. The goal of the decryption algorithm is to compute $g_k^{\alpha s}$, so that it could compare it to C_M. First, it computes $E = e(K_H, g^s) = e(g_{k-1}^{\alpha - r_{n+q}}, g^s) = g_k^{\alpha s} g_k^{-s r_{n+q}}$. Now it would suffice to compute $g_k^{s r_{n+q}}$. It does this by iterating over all $w \in$ Vertices in increasing order of value (this ensures that computations for inputs to a gate are done before trying to do computations on the gate itself) and computing $E_w = g_{j+1}^{s r_w}$ for each vertex w for which $f_w(x) = 1$. The computation steps are again divided by whether w is an input, OR gate or AND gate.

- *Input.*
  If $w \in$ Inputs and $f_w(x) = x_w = 1$, then the ciphertext contains C_w and the algorithm computes
  $$E_w = e(K_{w,1}, g^s) \cdot e(K_{w,2}, C_w) = e(g^{r_w} h_w^{z_w}, g^s) \cdot e(g^{-z_w}, h_w^s) = g_2^{s r_w}.$$

- *OR gate.*
  If $w \in$ Gates, $\texttt{Gatetype}(w) =$ OR and $f_w(x) = 1$, then at least one of $f_{A(w)}(x)$ and $f_{B(w)}(x)$ evaluates to 1. Let j be the depth of gate w. If $f_{A(w)}(x) = 1$, then the algorithm computes
  $$E_w = e(E_{A(w)}, K_{w,1}) \cdot e(K_{w,3}, g^s) = e(g_j^{s r_{A(w)}}, g^{a_w}) \cdot e(g_j^{r_w - a_w r_{A(w)}}, g^s) = g_{j+1}^{s r_w}.$$
  If $f_{A(w)}(x) = 0$, but $f_{B(w)}(x) = 1$, then the algorithm computes
  $$E_w = e(E_{B(w)}, K_{w,2}) \cdot e(K_{w,4}, g^s) = e(g_j^{s r_{B(w)}}, g^{b_w}) \cdot e(g_j^{r_w - b_w r_{B(w)}}, g^s) = g_{j+1}^{s r_w}.$$

- *AND gate.*
  If $w \in$ Gates, $\texttt{Gatetype}(w) = \text{AND}$ and $f_w(x) = 1$, then both $f_{A(w)}(x) = 1$ and $f_{B(w)}(x) = 1$. Let j be the depth of gate w. The algorithm computes

$$\begin{aligned} E_w &= e(E_{A(w)}, K_{w,1}) \cdot e(E_{B(w)}, K_{w,2}) \cdot e(K_{w,3}, g^s) \\ &= e(g_j^{sr_{A(w)}}, g^{a_w}) \cdot e(g_j^{sr_{B(w)}}, g^{b_w}) \cdot e(g_j^{r_w - a_w r_{A(w)} - b_w r_{B(w)}}, g^s) = g_{j+1}^{sr_w}. \end{aligned}$$

If $f(x) = f_{n+q}(x) = 1$, then the algorithm will be able to compute $E_{n+q} = g_k^{sr_{n+q}}$. It then computes $E \cdot E_{n+q} = g_k^{\alpha s}$, compares it to C_M and outputs $M = 1$ if they are equal and $M = 0$ otherwise. Correctness holds with high probability.

There are many ways to handle circuits with depth $d < \ell$. One of the simplest ways would be to add an extra step to the decryption algorithm. Since following the decryption algorithm, we can compute $E_{n+q} = g_{d+1}^{sr_{n+q}}$ (assuming we should be able to decrypt), we can use the multilinear maps to compute $e(E_{n+q}, g_{k-d-1}) = g_k^{sr_{n+q}}$ and proceed with the decryption.

3.2.2 Efficiency

According to the definition of a KP-ABE scheme, all of the algorithms have to be efficient, i.e., computable in time polynomial in the security parameter λ. To show this, we first recall that we assume that k, n and the circuit size are all polynomially bounded by λ, the group generator $\mathcal{G}$ and the bilinear maps are all efficiently computable, and that group multiplication in the generated groups is efficient as well. From this, it follows that exponentiation in the generated groups is efficient as well: even though at first glance, it might seem that we might need more than 2^λ multiplications to compute large powers of group elements (since $p > 2^\lambda$), it is actually possible to use only approximately $\log_2 p$ multiplications (which by assumption is polynomial in λ) using the fact that for any group element a and exponent m we have

$$a^m = \begin{cases} a \cdot (a^{\frac{m-1}{2}})^2, & \text{if } m \text{ is odd} \\ (a^{\frac{m}{2}})^2, & \text{otherwise.} \end{cases}$$

By proposition 1.1.5, this means that finding the inverse element is efficient as well. Also, choosing a random number from $\mathbb{Z}_p$ is equivalent to choosing a sequence of p bits at random, which is obviously efficiently doable (since the length of p is polynomial in λ). Now let us examine the efficiency of the algorithms (note that n and ℓ are constants with respect to λ):

- The $Setup$ algorithm runs $\mathcal{G}$, picks a polynomial (since n is polynomial in λ) number of random elements and does a polynomial number of exponentiations. Since all of these are efficient operations, the $Setup$ algorithm is efficient as well.

- The $Encrypt$ algorithm chooses at most two random elements and then does at most $n+2$ exponentiations (one for each set attribute, one for C_M and one for g^s), therefore it is efficiently computable.

- The $KeyGen$ algorithm chooses a constant number of random elements and does a constant number of exponentiations and multiplications for every vertex of the given circuit. Since the size of the circuit is polynomial in λ, the $KeyGen$ algorithm does a polynomial number of polynomial time operations, and therefore is efficiently computable

- The *Decrypt* algorithm computes a constant number of bilinear maps and does a constant number of group multiplications for every vertex in the given circuit, plus some additional polynomial time computation (the header computation etc). Since we assume that the bilinear maps are efficiently computable, the *Decrypt* algorithm is efficient as well.

3.3 The Modified Construction

We will now explain our modifications to the original scheme. Firstly, in the original scheme, when encrypting the bit 0, decryption will fail with probability $\frac{1}{p}$, since C_M is set to a random group element and therefore might coincide with the encryption of 1. We solve this by encrypting 0 in such a way that it can never collide with the encryption of 1. Secondly, in the original scheme, two random values a_w and b_w are chosen for every gate w during key generation. We will show that it is sufficient to choose only one random value z_w for every gate w and set $a_w = b_w = z_w$ in the key components, thereby eliminating one key component for every gate. The formal definitions of the modified algorithms follow.

Encrypt(PP, $x \in \{0,1\}^n$,$M \in \{0,1\}$). The encryption algorithm takes as input the public parameters PP, a descriptor input $x \in \{0,1\}^n$ and a message bit M. The encryption algorithm chooses a random $s \in \mathbb{Z}_p$. If $M = 1$, it lets $C_M = (g_k^{\alpha})^s$. Otherwise, it chooses a random $y \in \mathbb{Z}_p$ and tests whether $(g_k^{\alpha})^s$ is equal to g_k^y. If it is, the algorithm chooses a random $z \in \mathbb{Z}_p \backslash \{y\}$ and lets $C_M = g_k^z$. Otherwise, it lets $C_M = g_k^y$. Next, let S be the set of such i that $x_i = 1$. Then the ciphertext is created as

$$\text{CT} = (C_M, g^s, \forall i \in S\ C_i = h_i^s)$$

KeyGen(MSK, $f = (n, q, A, B, \texttt{GateType})$). The key generation algorithm takes as input the master secret key MSK and the description f of a circuit. The header component and the keys components for the inputs are computed as in the original scheme.

- *OR gate*
  If $w \in$ Gates and $\texttt{Gatetype}(w) =$ OR, then the algorithm chooses a random $z_w \in \mathbb{Z}_p$ and creates the key components as

  $$K_{w,1} = g^{z_w}, K_{w,2} = g_j^{r_w - z_w r_{A(w)}}, K_{w,3} = g_j^{r_w - z_w r_{B(w)}},$$

  where j is the depth of gate w.

- *AND gate*
  If $w \in$ Gates and $\texttt{Gatetype}(w) =$ AND, then the algorithm chooses a random $z_w \in \mathbb{Z}_p$ and creates the key components as

  $$K_{w,1} = g^{z_w}, K_{w,2} = g_j^{r_w - z_w(r_{A(w)} + r_{B(w)})},$$

  where j is the depth of gate w.

Decryption works analogously to the original scheme.

For an intuition why this construction might be secure (at least against the backtracking attack), consider an OR gate w. Suppose $f_{A(w)}(x) = 1$, but $f_{B(w)}(x) = 0$ for some input

$x \in \{0,1\}^n$. Then, using the bilinear maps, it is possible to compute $g_j^{sr_{A(w)}}$, $g_{j+1}^{sr_w}$ and $g_{j+1}^{sr_w - sz_w r_{B(w)}}$, from which it is possible to compute $g_{j+1}^{sz_w r_{B(w)}}$, but since our bilinear maps are not invertible (this follows from the security assumption given in Chapter 4), it is not possible to even compute $g_j^{sz_w r_{B(w)}}$. The formal proof of security will be given in Chapter 4.

Chapter 4

Security

In this chapter, the security of a KP-ABE scheme and the security assumption under which we prove security are defined. These are exactly the same as in [GGH$^+$13]. Then we give the proof of security for our modified scheme.

4.1 Security Definition

Definition 4.1.1. A function $\mu : \mathbb{N} \to \mathbb{R}$ is called *negligible*, if for every positive polynomial p (i.e., $p(n) > 0 \; \forall n \in \mathbb{N}$) there exists a $N \in \mathbb{N}$ such that for all $n > N$, $|\mu(n)| < \frac{1}{p(n)}$.

Definition 4.1.2. Define the selective security game for KP-ABE for bounded circuits with maximum depth ℓ and input number n between an adversary and a challenger as follows:

- **Init.** The adversary chooses a challenge attribute set x^* and sends it to the challenger.
- **Setup.** The challenger runs the setup algorithm with inputs 1^λ, n, ℓ, where λ is the security parameter, and sends the public parameters PP to the adversary.
- **Phase 1.** The adversary makes up to a polynomial number of queries for private keys for any circuit f with $f(x^*) = 0$. The challenger responds with $KeyGen(MSK, f)$.
- **Challenge.** The adversary sends two equal length messages M_0 and M_1 to the challenger. The challenger chooses $b \in \{0, 1\}$ at random and sends $Encrypt(PP, x^*, M_b)$ to the adversary.
- **Phase 2.** Phase 1 is repeated.
- **Guess.** The adversary outputs a guess b' of b.

An encryption scheme for ABE for polynomially bounded circuits is said to be *selectively secure* if for all polynomial time adversaries $\mathcal{A}$ the advantage of $\mathcal{A}$ with respect to the security parameter λ $Adv_{\mathcal{A}}(\lambda) = \Pr[b' = b] - \frac{1}{2}$ is negligible (an encryption scheme for ABE for polynomially bounded circuits is said to be *(non-selectively) secure*, if in the security game above the adversary sends the challenge attribute set x^* not during the Init phase, but during the Challenge phase).

Intuitively, Phase 1 and Phase 2 represent the situation where many dishonest users try to work together to break a ciphertext that none of them would be able to decrypt on their own. This security definition ensures that the scheme remains secure even if such a situation should occur.

4.2 Proof of Security

For the proof of selective security for the original construction, please see [GGH$^+$13]. In this section, we will give a proof that our modified scheme remains selectively secure. As is done in [GGH$^+$13], we prove the selective security of the modified construction under the following assumption, called the k-Multilinear Decisional Diffie-Hellman assumption.

Assumption (k-MDDH). *A challenger runs the group generator $\mathcal{G}(1^\lambda, k)$, where λ is the security parameter, to generate k groups and generators of order $p > 2^\lambda$. Then it picks random $s, c_1, c_2, \ldots, c_k$, and chooses $b \in \{0, 1\}$ at random. If $b = 1$, it sets T equal to $g^{sc_1c_2\cdots c_k}$, otherwise it sets T equal to a random element of $\mathbb{G}_k$. Then it sends the group descriptions $\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ and $p, g = g_1, g^s, g^{c_1}, \ldots, g^{c_k}, T$ to the adversary, and the adversary (possibly using the set of bilinear maps) outputs its guess b' of b. The assumption states that for all polynomial time adversaries $\mathcal{A}$ the advantage of $\mathcal{A}$ with respect to the security parameter λ $Adv_{\mathcal{A}}(\lambda) = Pr[b' = b] - \frac{1}{2}$ is negligible.*

Theorem 4.2.1. *The modified construction given in the previous chapter achieves selective security for circuits of depth $k-1$ under the k-MDDH assumption.*

Proof. The proof is almost identical to the one in [GGH$^+$13], with the exception of the $KeyGen$ phase for AND and OR gates and the fact that the probability computations are explicit.

We will show that if there exists a polynomial time adversary $\mathcal{A}$ that has a non-negligible advantage in the selective security game for circuits of depth $k-1$ and inputs of length n, then there exists a polynomial time adversary $\mathcal{B}$ that breaks the k-MDDH security assumption. In the following, we describe how $\mathcal{B}$ can use $\mathcal{A}$ to break the security assumption by simulating the challenger in the selective security game.

Init. $\mathcal{B}$ first receives the group descriptions $\vec{\mathbb{G}} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ and $p, g = g_1, g^s, g^{c_1}, \ldots, g^{c_k}, T$ from the challenger of the k-MDDH assumption, where T is either $g_k^{sc_1c_2\cdots c_k}$ or a random group element of $\mathbb{G}_k$ with probability $\frac{1}{2}$. $\mathcal{A}$ chooses the challenge attribute set $x^* \in \{0,1\}^n$ and sends it to $\mathcal{B}$.

Setup. $\mathcal{B}$ chooses random $y_1, \ldots, y_n \in \mathbb{Z}_p$ and sets

$$h_i = \begin{cases} g^{y_i} & \text{if } x_i^* = 1 \\ g^{c_1+y_i} & \text{if } x_i^* = 0 \end{cases}, i \in \{1, 2, \ldots, n\}.$$

Note that the choices of h_i above are distributed identically to the h_i in the actual construction: for a fixed $a \in \mathbb{Z}_p$ and $i \in \{1, 2, \ldots, n\}$, $\Pr[h_i = g^a] = \frac{1}{p}$ in the actual construction, and in for the choices of h_i above $\Pr[h_i = g^a] = \Pr[g^{y_i} = g^a] = \frac{1}{p}$ if $x_i^* = 1$ and $\Pr[h_i = g^a] = \Pr[g^{c_1+y_i} = g^a] = \Pr[g^{y_i} = g^{a-c_1}] = \frac{1}{p}$ if $x_i^* = 0$.

Next, B sets $g_k^\alpha = g_k^{\xi + c_1 \cdots c_k}$, where $\xi \in \mathbb{Z}_p$ is chosen randomly (this can be computed by using the pairing function on $g^{c_1}, \ldots, g^{c_k}$ repeatedly). Note that again, the distribution is the as the distribution of g_k^α in the actual construction. $\mathcal{B}$ then sends $g_k^\alpha, h_1, \ldots, h_n$ to $\mathcal{A}$.

KeyGen Phase. Both key generation phases will be executed in the same way by $\mathcal{B}$, so they will be described here only once. $\mathcal{B}$ will receive a circuit description $f = (n, q, A, B, \texttt{Gatetype})$ from $\mathcal{A}$ such that $f(x^*) = 0$. Note that $\mathcal{B}$ cannot just use the actual construction to generate the secret keys, since $\mathcal{B}$ does not know the actual value of α and therefore cannot (or at least we cannot assume that it can) compute the master secret key $MSK = g_{k-1}^{\alpha}$ and therefore is unable to directly compute the header component $K_H = g_{k-1}^{\alpha - r_{n+q}}$ just from knowing r_{n+q}. On the other hand, the distribution of keys and the values of r_w has to be identical to the one in the actual construction for the success probability of $\mathcal{A}$ to remain the same.

To overcome these problems, $\mathcal{B}$ does the following for every vertex w with depth j. If $f_w(x^*) = 1$, it views r_w as a random element in $\mathbb{Z}_p$ (as in the actual construction). If $f_w(x^*) = 0$, it views r_w as $c_1 \cdots c_{j+1} + \eta_w$, where η_w is a random element in $\mathbb{Z}_p$ known by $\mathcal{B}$. Then the distribution of r_w is the same as in the actual construction, and since $f_{n+q}(x^*) = f(x^*) = 0$, $\mathcal{B}$ will view r_{n+q} as $c_1 \cdots c_k + \eta_{n+q}$ and know how to compute $g_{k-1}^{\eta_{n+q}}$, which will allow it to compute the header component by cancellation as

$$K_H = g_{k-1}^{\alpha - r_{n+q}} = g_{k-1}^{\xi + c_1 \cdot \ldots \cdot c_k - (c_1 \cdot \ldots \cdot c_k + \eta_{n+q})} = g_{k-1}^{\xi} \cdot (g_{k-1}^{\eta_{n+q}})^{-1}.$$

The details of the key generation for each vertex w follow.

- *Input*
  Consider $w \in$ Inputs. If $x_w^* = 1$, then $\mathcal{B}$ chooses $r_w, z_w \in \mathbb{Z}_p$ at random and sets the key components as is done in the actual construction. If $x_w^* = 0$, $\mathcal{B}$ chooses $\eta_w, \nu_w \in \mathbb{Z}_p$ at random and views r_w as $c_1 c_2 + \eta_w$ and z_w as $-c_2 + \nu_w$ (therefore the distributions of r_w and z_w are the same as in the actual construction). Then the key components are

  $$K_{w,1} = g^{r_w} h_w^{z_w} = g^{c_1 c_2 + \eta_w} (g^{c_1 + y_w})^{-c_2 + \nu_w} = g^{\eta_w} (g^{c_2})^{-y_w} (g^{c_1})^{\nu_w} g^{\nu_w y_w}$$

  and
  $$K_{w,2} = g^{z_w} = g^{-c_2 + \nu_w} = (g^{c_2})^{-1} g^{\nu_w}.$$

- *OR gate*
  Consider $w \in$ Gates with $\texttt{Gatetype}(w) =$ OR. Let j be the depth of gate w. If $f_w(x^*) = 1$, then $\mathcal{B}$ behaves exactly as the $KeyGen$ in the actual construction (note that this can be done due to the fact that $g_j^{r_{A(w)}}$ and $g_j^{r_{B(w)}}$ can always be computed no matter what the output values of the vertices $A(w)$ and $B(w)$ were: if they were 1, $\mathcal{B}$ knows $r_{A(w)}/r_{B(w)}$, and if say $f_{A(w)}(x^*) = 0$, then $\mathcal{B}$ can knows $\eta_{A(w)}$ and can compute $g_j^{r_{A(w)}} = g_j^{c_1 \cdots c_j} \cdot g^{\eta_{A(w)}}$ using the multilinear maps). If $f_w(x^*) = 0$, then $\mathcal{B}$ chooses random $\eta_w, \phi_w \in \mathbb{Z}_p$, views z_w as $c_{j+1} + \phi_w$ and r_w as $c_1 \cdots c_{j+1} + \eta_w$. Note that if $f_w(x^*) = 0$, then $f_{A(w)}(x^*) = 0$ and $f_{B(w)}(x^*) = 0$ as well, and therefore $r_{B(w)} = c_1 \cdots c_j + \eta_{B(w)}$ and $r_{A(w)} = c_1 \cdots c_j + \eta_{A(w)}$. $\mathcal{B}$ sets the key components to

  $$K_{w,1} = g^{z_w} = g^{c_{j+1} + \phi_w},$$

  $$\begin{aligned} K_{w,2} = g_j^{r_w - z_w r_{A(w)}} &= g_j^{c_1 \cdots c_{j+1} + \eta_w - (c_{j+1} + \phi_w)(c_1 \cdots c_j + \eta_{A(w)})} \\ &= g_j^{\eta_w - c_{j+1} \eta_{A(w)} - \phi_w c_1 \cdots c_j - \phi_w \eta_{A(w)}} \\ &= g_j^{\eta_w - \phi_w \eta_{A(w)}} \cdot e(g^{c_{j+1}}, g_{j-1}^{-\eta_{A(w)}}) \cdot (g_j^{c_1 \cdots c_j})^{-\phi_w}, \end{aligned}$$

  $$\begin{aligned} K_{w,3} = g_j^{r_w - z_w r_{B(w)}} &= g_j^{c_1 \cdots c_{j+1} + \eta_w - (c_{j+1} + \phi_w)(c_1 \cdots c_j + \eta_{B(w)})} \\ &= g_j^{\eta_w - c_{j+1} \eta_{B(w)} - \phi_w c_1 \cdots c_j - \phi_w \eta_{B(w)}} \\ &= g_j^{\eta_w - \phi_w \eta_{B(w)}} \cdot e(g^{c_{j+1}}, g_{j-1}^{-\eta_{B(w)}}) \cdot (g_j^{c_1 \cdots c_j})^{-\phi_w} \end{aligned}$$

$\mathcal{B}$ can compute $K_{w,2}$ and $K_{w,3}$, because it knows $\eta_w, \phi_w, \eta_{B(w)}$ and $\eta_{A(w)}$ and can compute $g_j^{c_1 \cdots c_j}$ by using e repeatedly. Since the distributions of z_w and r_w are the same as in the actual construction, the distribution of the simulated keys is also the same as in the actual construction.

- *AND gate*
  Consider $w \in$ Gates with $\texttt{Gatetype}(w) =$ AND. Let j be the depth of gate w. If $f_w(x^*) = 1$, then $\mathcal{B}$ behaves exactly as the honest $KeyGen$. If $f_w(x^*) = 0$, then $\mathcal{B}$ chooses random $\phi_w, \eta_w \in \mathbb{Z}_p$ and views r_w as $c_1 \cdots c_{j+1} + \eta_w$. Since $f_w(x^*) = 0$, either $f_{A(w)}(x^*) = 0$ or $f_{B(w)}(x^*) = 0$ or both. If $f_{A(w)}(x^*) = 0$ and $f_{B(w)}(x^*) = 1$, then $r_{A(w)} = c_1 \cdots c_j + \eta_{A(w)}$, $\mathcal{B}$ views z_w as $c_{j+1} + \phi_w$ and sets the key components as

$$K_{w,1} = g^{z_w} = g^{c_{j+1}+\phi_w},$$

$$\begin{aligned} K_{w,2} &= g_j^{r_w - z_w(r_{A(w)}+r_{B(w)})} \\ &= g_j^{c_1\cdots c_{j+1}+\eta_w-(c_{j+1}+\phi_w)(c_1\cdots c_j+\eta_{A(w)}+r_{B(w)})} \\ &= g_j^{\eta_w - c_{j+1}(\eta_{A(w)}+r_{B(w)})-\phi_w c_1\cdots c_j - \phi_w(\eta_{A(w)}+r_{B(w)})} \\ &= g_j^{\eta_w-\phi_w(\eta_{A(w)}+r_{B(w)})} \cdot e\big(g^{c_{j+1}}, g_{j-1}^{-(\eta_{A(w)}+r_{B(w)})}\big) \cdot (g_j^{c_1\cdots c_j})^{-\phi_w}. \end{aligned}$$

  $\mathcal{B}$ can compute $K_{w,2}$, because it knows $\eta_w, \phi_w, r_{B(w)}$ and $\eta_{A(w)}$ and can compute $g_j^{c_1\cdots c_j}$ by using e repeatedly. The case $f_{A(w)}(x^*) = 1$ and $f_{B(w)}(x^*) = 0$ is analogous. If both $f_{A(w)}(x^*) = 0$ and $f_{B(w)}(x^*) = 0$, then both $r_{A(w)} = c_1 \cdots c_j + \eta_{A(w)}$ and $r_{B(w)} = c_1 \cdots c_j + \eta_{B(w)}$, $\mathcal{B}$ views z_w as $2^{-1}c_{j+1} + \phi_w$ and sets the key components as

$$K_{w,1} = g^{z_w} = g^{2^{-1}c_{j+1}+\phi_w},$$

$$\begin{aligned} K_{w,2} &= g_j^{r_w - z_w(r_{A(w)}+r_{B(w)})} \\ &= g_j^{c_1\cdots c_{j+1}+\eta_w-(2^{-1}c_{j+1}+\phi_w)(2c_1\cdots c_j+\eta_{A(w)}+\eta_{B(w)})} \\ &= g_j^{\eta_w - 2^{-1}c_{j+1}(\eta_{A(w)}+\eta_{B(w)})-2\phi_w c_1\cdots c_j - \phi_w(\eta_{A(w)}+\eta_{B(w)})} \\ &= g_j^{\eta_w-\phi_w(\eta_{A(w)}+\eta_{B(w)})} \cdot e\big(g^{c_{j+1}}, g_{j-1}^{-2^{-1}(\eta_{A(w)}+\eta_{B(w)})}\big) \cdot (g_j^{c_1\cdots c_j})^{-2\phi_w}. \end{aligned}$$

  $\mathcal{B}$ can compute $K_{w,2}$ for the same reasons as above. It remains to show that the distribution of z_w is the same as in the actual construction. For a fixed element $a \in \mathbb{Z}_p$, the probability that a randomly chosen element of $\mathbb{Z}_p$ is equal to $\frac{1}{p}$, and therefore in the actual construction, Pr$[z_w = a] = \frac{1}{p}$. In the simulated construction,

  - If $f_{A(w)}(x^*) = 1$ and $f_{B(w)}(x^*) = 1$, then $f_w(x^*) = 1$ and $\mathcal{B}$ behaves exactly as in the actual construction, and therefore Pr$[z_w = a] = \frac{1}{p}$.
  - If $f_{A(w)}(x^*) = 1$ and $f_{B(w)}(x^*) = 0$ or $f_{A(w)}(x^*) = 0$ and $f_{B(w)}(x^*) = 1$, then Pr$[z_w = a] =$ Pr$[\phi_w = a - c_{j+1}] = \frac{1}{p}$
  - If $f_{A(w)}(x^*) = 0$ and $f_{B(w)}(x^*) = 0$, then Pr$[z_w = a] =$ Pr$[\phi_w = a - 2^{-1}c_{j+1}] = \frac{1}{p}$

Therefore the distribution of the simulated z_w is the same as the z_w in the actual construction in all possible cases.

$\mathcal{B}$ sends the secret key SK to $\mathcal{A}$.

Challenge. We can assume without loss of generality that the two messages that $\mathcal{A}$ sends to $\mathcal{B}$ are 0 and 1, since there are only two possible messages in our construction and sending two equal messages would only decrease the likelihood of $\mathcal{A}$ guessing correctly. Therefore $\mathcal{B}$ has to create a ciphertext that is an encryption of either 0 or 1 with equal probability. Let S^* be the set on indices i for which $x_i^* = 1$. $\mathcal{B}$ creates the challenge ciphertext as

$$\text{CT} = (T \cdot g_k^{s\xi}, g^s, \forall j \in S^* C_j = (g^s)^{y_j}).$$

If $T = g_k^{sc_1c_2\cdots c_k}$, then this in an encryption of 1. Otherwise, if T was chosen randomly in $\mathbb{G}_k$, this is an encryption of 0 with probability $\frac{p-1}{p}$ and an encryption of 1 with probability $\frac{1}{p}$. Therefore, $\mathcal{B}$ sends $\mathcal{A}$ the encryption of 1 with probability $\frac{1}{2} + \frac{1}{2p}$ and the encryption of 0 with probability $\frac{1}{2} - \frac{1}{2p}$. This somewhat skews the probability of $\mathcal{A}$ winning the selective security game, fortunately by a small enough amount (this will be proved later).

After repeating the $KeyGen$ phase, $\mathcal{A}$ outputs a guess b'' of whether the encrypted message was 1 or 0, and $\mathcal{B}$ outputs $b' = b''$ as its guess of b.

It remains to be shown that if $\mathcal{A}$ has a non-negligible advantage in the selective security game for circuits of maximum depth $k-1$ and n inputs, then $\mathcal{B}$ has a non-negligible advantage in the k-MDDH assumption game. Denote by $a(\lambda)$ the advantage of $\mathcal{A}$ in the selective security game. Let C be the event that $\mathcal{A}$ wins in the selective security game, D be the event that $\mathcal{A}$ is sent an encryption of 1, and E be the event that $\mathcal{A}$ is sent an encryption of 0. Let $a_1(\lambda) = \Pr[C|D] - \frac{1}{2}$ and $a_2(\lambda) = \Pr[C|E] - \frac{1}{2}$. Then from the normal selective security game, we get

$$\begin{aligned}\frac{1}{2} + a(\lambda) &= \Pr[C] = \Pr[D]\Pr[C|D] + \Pr[E]\Pr[C|E] \\ &= \frac{1}{2}\left(\frac{1}{2} + a_1(\lambda)\right) + \frac{1}{2}\left(\frac{1}{2} + a_2(\lambda)\right) = \frac{1}{2} + \frac{a_1(\lambda) + a_2(\lambda)}{2},\end{aligned}$$

from which

$$a(\lambda) = \frac{a_1(\lambda) + a_2(\lambda)}{2}.$$

Since $a(\lambda)$ is not negligible, then by the definition of negligible functions, there exists a positive polynomial p_1 such that for every $N \in \mathbb{N}$ there exists a $\lambda > N$ such that $|a(\lambda)| > \frac{1}{p_1(\lambda)}$. It remains to show that such a polynomial exists for the advantage of $\mathcal{B}$ $b(\lambda)$ in the k-MDDH security game as well. As shown in the description of the Challenge phase, in our simulated selective security game, we have $\Pr[D|b = 1] = 1$, $\Pr[D|b = 0] = \frac{1}{p}$, $\Pr[E|b = 1] = 0$ and

$\Pr[E|b=0] = 1 - \frac{1}{p}$. Therefore,

$$\begin{aligned}
\frac{1}{2} + b(\lambda) &= \Pr[\mathcal{B} \text{ wins}] \\
&= \Pr[b=1]\Pr[D|b=1]\Pr[C|D] + \Pr[b=1]\Pr[E|b=1]\left(1 - \Pr[C|E]\right) \\
&+ \Pr[b=0]\Pr[D|b=0]\left(1 - \Pr[C|D]\right) + \Pr[b=0]\Pr[E|b=0]\Pr[C|E] \\
&= \frac{1}{2}\left(\frac{1}{2} + a_1(\lambda)\right) + 0 \\
&+ \frac{1}{2} \cdot \frac{1}{p} \cdot \left(1 - \frac{1}{2} - a_1(\lambda)\right) + \frac{1}{2}\left(1 - \frac{1}{p}\right)\left(\frac{1}{2} + a_2(\lambda)\right) \\
&= \frac{1}{2} + \frac{a_1(\lambda) + a_2(\lambda)}{2} - \frac{a_1(\lambda) + a_2(\lambda)}{2p} \\
&= \frac{1}{2} + a(\lambda)\left(1 - \frac{1}{p}\right)
\end{aligned}$$

From the fact that $p > 2^\lambda$, we get that for every λ for which $|a(\lambda)| > \frac{1}{p_1(\lambda)}$,

$$\begin{aligned}
|b(\lambda)| = |a(\lambda)\left(1 - \frac{1}{p}\right)| &= |a(\lambda)|\left(1 - \frac{1}{p}\right) \\
&> |a(\lambda)|\left(1 - \frac{1}{2^\lambda}\right) \geqslant \frac{1}{2}|a(\lambda)| > \frac{1}{2p_1(\lambda)}.
\end{aligned}$$

Since $2p_1(\lambda)$ is a polynomial as well, it follows that $b(\lambda)$ is not negligible.

$\mathcal{B}$ is a polynomial time algorithm, because $\mathcal{A}$ is a polynomial time algorithm, all the algorithms in the actual construction are polynomial time algorithms, and $\mathcal{B}$ only a constant amount of extra polynomial time computations for every gate compared to the actual construction.

□

Summary

In this work, an overview of Attribute-Based Encryption was given. A recent construction for Key-Policy Attribute-Based Encryption which expands the allowable class of functions from boolean formulae to polynomial size circuits that was presented in [GGH$^+$13] was introduced. Some suggestions for improving the original construction were given, and the security of the resulting scheme was proved.

There are many possible research problems remaining in connection with the KP-ABE constructions given in this work. Some examples include proving non-selective security and encrypting messages longer than one bit. Whether it is possible to further decrease the necessary number of key components also remains an interesting open question.

Multilineaarsetel kujutustel baseeruvast atribuudipõhisest krüpteerimisest loogikaskeemide jaoks

Bakalaureusetöö

Kairi Kangro

Resümee

Tänapäeval on avaliku võtme krüptograafia laialdaselt kasutuses, näiteks krüpteeritud meilide saatmisel. Sellel on aga üks oluline puudus: saajaid saab olla ainult üks. Üks võimalik lahendus sellele probleemile on Sahai ja Watersi poolt välja töötatud atribuudipõhine krüpteerimine (*Attribute-Based Encryption*,[SW05]), mis võimaldab paindlikumalt määratleda, kes milliseid krüpteeritud andmeid lahti krüpteerida saab.

Käesolevas töös käsitletakse põhjalikumalt võtmepoliitika atribuudipõhist krüpteerimist (*Key-Policy Attribute-Based Encryption*,[GPSW06]), mis võimaldab siduda andmed krüpteerimisel teatud hulga atribuutidega ja väljastada kasutajatele võtmeid, mis on seotud teatud funktsioonidega. Kasutaja saab šifferteksti lahti krüpteerida ainult siis, kui tema võtmega seotud funktsioon on šiffertekstiga seotud atribuutide korral tõene.

Töös tutvustatakse hiljuti Garg *et al* poolt avaldatud artiklis [GGH$^+$13] toodud võtmepoliitika atribuudipõhise krüpteerimise skeemi, mis laiendab lubatavate funktsioonide klassi seni parimalt polünomiaalse suurusega loogiliste avaldiste klassilt polünomiaalse suurusega loogika-skeemide klassile, mida peetakse oluliselt suuremaks klassiks ([AB09]). Skeemi loomiseks kasutatakse atribuudipõhises krüpteerimises üldlevinud bilineaarsete kujutuste asemel multilineaarseid kujutusi, ja Garg *et al* toovad artiklis [GGH$^+$13] esile ühe teatud tüüpi ründe, nn tagurdusründe, võimaliku põhjusena, miks bilineaarsete kujutustega pole seni õnnestunud lubatavate funktsioonide klassi laiendada. Töös tuuakse ära ka skeemist arusaamiseks vajalikud põhimõisted ja -tulemused, nende hulgas olulisemateks on rühmad, loogikaskeemid ja algoritmide efektiivsus.

Lisaks eelmainitud tulemuste ja mõistete tutvustamisele pakutakse töö autori poolt välja üks moodus skeemis vajaminevate võtmekomponentide arvu vähendamiseks, ja esitatakse tulemusena saadud skeemi täielik turvatõestus koos vajalike turvalisuse definitsioonide ja eeldustega.

Bibliography

[GPSW06] V. Goyal, O. Pandey, A. Sahai, B. Waters. *Attribute-based encryption for fine-grained access control of encrypted data*, ACM Conference on Computers and Communications Security, pages 89-98, 2006.

[AB09] S. Arora, B. Barak. *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009. 594 pages.

[BHR12] M. Bellare, V. T. Hoang, P. Rogaway. *Foundations of garbled circuits*, Cryptology ePrint Archive, Report 2012/265, 2012. http://eprint.iacr.org/2012/265.pdf.

[BF01] D. Boneh, M. K. Franklin. *Identity-based encryption from the weil pairing*, CRYPTO, volume 2139 of Lecture Notes in Computer Science, pages 213-229, 2001.

[BS03] D. Boneh, A. Silverberg. *Applications of multilinear forms to cryptography*, Contemporary Mathematics, 324:71-90, 2003.

[GGH12] S. Garg, C. Gentry, and S. Halevi. *Candidate multilinear maps from ideal lattices and applications*, Cryptology ePrint Archive, Report 2012/610, 2012.http://eprint.iacr.org/2012/610.pdf.

[GGH$^+$13] S. Garg, C.Gentry, S.Halevi, A. Sahai, B. Waters. *Attribute-Based Encryption for Circuits from Multilinear Maps*, Cryptology ePrint Archive, Report 2013/128, 2013. http://eprint.iacr.org/2013/128.pdf. (accepted for publication in CRYPTO 2013)

[Kil05] M. Kilp. *Algebra I*, Eesti Matemaatika Selts, 2005. 311 pages.

[SW05] A. Sahai, B. Waters. *Fuzzy identity-based encryption*, EUROCRYPT, pages 457-473, 2005.