Attribute Based Encryption for Deterministic Finite Automata from DLIN

Shweta Agrawal [⋆], Monosij Maitra[⋆⋆], and Shota Yamada[⋆⋆⋆]

Abstract. Waters [Crypto, 2012] provided the first attribute based encryption scheme ABE for Deterministic Finite Automata (DFA) from a parametrized or "q-type" assumption over bilinear maps. Obtaining a construction from static assumptions has been elusive, despite much progress in the area of ABE.

In this work, we construct the first attribute based encryption scheme for DFA from static assumptions on pairings, namely, the DLIN assumption. Our scheme supports unbounded length inputs, unbounded length machines and unbounded key requests. In more detail, secret keys in our construction are associated with a DFA M of *unbounded* length, ciphertexts are associated with a tuple $(\mathbf{x}, \mu)$ where $\mathbf{x}$ is a public attribute of *unbounded* length and μ is a secret message bit, and decryption recovers μ if and only if $M(\mathbf{x}) = 1$.

Our techniques are at least as interesting as our final result. We present a simple compiler that combines constructions of unbounded ABE schemes for *monotone span programs* (MSP) in a black box way to construct ABE for DFA. In more detail, we find a way to embed DFA computation into monotone span programs, which lets us compose existing constructions (modified suitably) of unbounded key-policy ABE (kpABE) and unbounded ciphertext-policy ABE (cpABE) for MSP in a simple and modular way to obtain key-policy ABE for DFA. Our construction uses its building blocks in a *symmetric* way – by swapping the use of the underlying kpABE and cpABE, we also obtain a construction of ciphertext-policy ABE for DFA.

Our work extends techniques developed recently by Agrawal, Maitra and Yamada [Crypto 2019], which show how to construct ABE that support unbounded machines and unbounded inputs by combining ABE schemes that are bounded in one co-ordinate. At the heart of our work is the observation that unbounded, multi-use ABE for MSP already achieve most of what we need to build ABE for DFA.

1 Introduction

Attribute based encryption (ABE) [56] is a new paradigm of encryption that enables fine grained access control on encrypted data. In attribute based encryption, a ciphertext of a message m is labelled with a public attribute $\mathbf{x}$ and secret keys are labelled with a function f. Decryption succeeds to yield the hidden message m if and only if the attribute satisfies the function, namely $f(\mathbf{x}) = 1$. ABE schemes have a rich and beautiful

[⋆] IIT Madras, India. shweta.a@cse.iitm.ac.in
[⋆⋆] IIT Madras, India. monosij@cse.iitm.ac.in
[⋆⋆⋆] AIST, Japan. yamada-shota@aist.go.jp

history [56, 41, 19, 16, 43, 50, 3, 57, 37, 17, 38, 39, 21, 8], with constructions for various classes of functions proven secure under diverse assumptions.

Typically, the function f encoded in the secret key is represented as a Boolean circuit, which necessitates issuing different keys to support different input lengths, even to compute the same functionality. In a breakthrough work, Waters [57] provided the first construction of ABE for regular languages: here, the secret key is associated with a deterministic finite automaton (DFA) and ciphertext is associated with attribute $\mathbf{x}$ of *arbitrary* length. The same secret key can directly decrypt ciphertexts that encode inputs of varying lengths, yielding the first ABE that supports a *uniform* model of computation. Since then, other constructions supporting the uniform model of computation were proposed, supporting even Turing machines [34, 9, 4], but all these relied on the powerful machinery of multilinear maps [31], indistinguishability obfuscation [15, 32] or witness encryption [33], none of which are considered standard assumptions.

While the Waters construction relied on the hardness of assumptions over bilinear maps, which are well understood, the assumption is *parametrized* (also known as "q-type"), which means that the size of the assumption depends on the queries made by the adversary. Achieving a construction of ABE for DFA from standard static assumptions over bilinear maps has remained elusive. Very recently, Agrawal, Maitra and Yamada [5] provided an ABE for *nondeterministic* finite automata from the learning with errors assumption. However, their construction makes use of highly lattice specific machinery (such as reusable garbled circuits [35]) and it is unclear how to use these ideas to improve the state of affairs in the world of pairings.

1.1 Our Results.

In this work, we construct the first attribute based encryption scheme for DFA from static assumptions on pairings, namely, the DLIN assumption. Our scheme supports unbounded length inputs as well as unbounded length machines. In more detail, secret keys in our construction are associated with a DFA M of unbounded length, ciphertexts are associated with a tuple $(\mathbf{x}, m)$ where $\mathbf{x}$ is a public attribute of unbounded length and m is a secret message bit, and decryption recovers m if and only if $M(\mathbf{x}) = 1$. Our construction also supports unbounded key requests by the adversary. Additionally, via a simple tweak to our construction, we also obtain the first ciphertext-policy ABE for DFA from the DLIN assumption.

We contrast our results with prior work in Table 1. For brevity, we only compare with constructions of ABE that support uniform models of computation (in particular, handle unbounded input lengths) and rely on standard assumptions. Other relevant work is discussed in Section 1.3.

1.2 Our Techniques.

A natural starting point for constructing (key policy) ABE for DFA is (key policy) ABE for monotone span programs (MSP), which has been studied extensively in the literature. Recall that an MSP is specified by a pair $(\mathbf{L}, \rho)$ of a matrix and a labelling function where $\mathbf{L} \in \mathbb{Z}_p^{\ell \times m}$, $\rho : [\ell] \to \{0,1\}^*$ for some integer ℓ, m. Intuitively, the map ρ labels

Construction	**Model**	KP **or** CP	**Number of Keys**	**Assumption**
Waters [57]	DFA	KP	unbounded	q-type assumption on bilinear maps
Attrapadung [12]	DFA	KP and CP	unbounded	q-type assumption on bilinear maps
Agrawal-Singh [7]	DFA	KP	single	LWE
Agrawal-Maitra-Yamada [5]	NFA	KP	unbounded	LWE
Gong-Waters-Wee [36]	DFA	KP	unbounded	kLIN
This	DFA	KP and CP	unbounded	DLIN

Table 1. Comparison with prior work supporting unbounded input length. KP and CP indicate key-policy and ciphertext-policy respectively.

row i with attribute $\rho(i)$. Given a set of attributes I as input, the MSP accepts the input iff the sub-matrix of $\mathbf{L}$ restricted to attributes selected by I contains a special target vector in its row span (please see Section 2.1 for the precise definition).

Step 1: Leveraging ABE for MSP. Our first observation is that DFA computation is simple enough to be encoded into an MSP. In more detail, given a DFA machine M and an input string $\mathbf{x}$, it is possible to map the DFA M into an MSP $(\mathbf{L}_M, \rho_M)$ and the input $\mathbf{x}$ into a set of attributes $S_\mathbf{x}$ such that the MSP $(\mathbf{L}_M, \rho_M)$ accepts attributes $S_\mathbf{x}$ iff $M(\mathbf{x}) = 1$. We exhibit such a map in Section 4.1 and prove the following theorem:

Theorem 1. *(Informal) Let $(\mathbf{L}_M, \rho_M)$ be the MSP and $S_\mathbf{x}$ be the set of attributes obtained by applying the map specified in Section 4.1 to M and $\mathbf{x}$ respectively. Then, the MSP $(\mathbf{L}_M, \rho_M)$ accepts attributes $S_\mathbf{x}$ iff $M(\mathbf{x}) = 1$.*

This provides a starting point for using ABE for MSP, which can be constructed from static assumptions, as a building block towards constructing ABE for DFA.

Step 2: Handling Unbounded Length. While this seems promising as a first step, the careful reader may have noticed that the above idea fails to address the primary challenge of supporting DFA, namely, that of handling inputs of unbounded length. DFA is a uniform model of computation, which means that the same machine must process inputs of arbitrary length. On the other hand, an MSP can only process inputs of bounded length – in particular, the length of inputs that an MSP can read is clearly bounded above by the number of rows in $\mathbf{L}$.

This appears to make ABE for MSP almost useless for our purposes, since there is no way to guarantee that $|\mathbf{x}|$ is less than the number of rows in $\mathbf{L}$ (denoted by $|\mathbf{x}| \leq |M|$ in the sequel[1]). However, notice that since both the inputs and the machines have unbounded length, it still holds in some cases that $|\mathbf{x}| \leq |M|$, and if we can handle this, it still constitutes progress. More hurdles present themselves – for instance, the syntax of ABE for DFA does not allow the setup algorithm to know the lengths $|\mathbf{x}|$, $|M|$, the key

[1] While imprecise, we use this notation here for intuition. Formally, it will turn out to be sufficient to compare $|\mathbf{x}|$ with $|Q|$, where $|Q|$ is the number of states in M.

generation algorithm cannot know $|\mathbf{x}|$ and the encrypt algorithm cannot know $|M|$. But this challenge can be overcome by making use of the so called *unbounded* ABE schemes, as described next.

Unbounded ABE schemes (for MSP) [54, 23] are those in which the setup algorithm places no restriction on the length of the attributes or the size of the policies that are embedded in the ciphertexts and keys. Moreover, the key generation and encrypt algorithms do not require knowledge of input length or policy size respectively. While significantly more challenging to build than their bounded counterparts, a small number of existing constructions [54, 23] achieve this property while relying on standard assumptions.

We show in Section 3.2 that unbounded key policy ABE schemes for MSP can indeed be used to construct ABE for DFA so long as $|\mathbf{x}| \le |M|$. More formally, we define relation $R^{\mathsf{KP}}(S, (\mathbf{L}, \rho)) = 1$ iff the span program $(\mathbf{L}, \rho)$ accepts the attribute set S and $R^{\mathsf{DFA}\le}(\mathbf{x}, M) = M(\mathbf{x}) \wedge \big(|\mathbf{x}| \overset{?}{\le} |M|\big)$. Then, we have that:

Theorem 2. *(Informal) Let* kpABE *be a secure unbounded ABE for the relation* R^{KP}. *Then, the construction* $\mathsf{dfaABE}^{\le}$ *provided in Section 3.2 is a secure ABE for the relation* $R^{\mathsf{DFA}\le}$.

Step 3: The trick of Agrawal, Maitra and Yamada. To construct a full fledged ABE for DFA, our next tool is a recent trick by Agrawal, Maitra and Yamada [5]. In [5], the authors show how to construct an ABE for nondeterministic finite automata (NFA) that supports unbounded inputs and unbounded machines, by running in parallel two restricted ABE for NFA schemes: one that supports unbounded inputs but bounded machines and one that supports bounded inputs but unbounded machines.

Our goal is to construct an ABE scheme dfaABE for the relation $R^{\mathsf{DFA}}(\mathbf{x}, M) = M(\mathbf{x})$. By using the trick of [5], we can construct our dfaABE from two special ABE schemes as follows:

1. An ABE $\mathsf{dfaABE}^{\le}$ for the relation $R^{\mathsf{DFA}\le}(\mathbf{x}, M) = M(\mathbf{x}) \wedge \big(|\mathbf{x}| \overset{?}{\le} |M|\big)$.
2. An ABE $\mathsf{dfaABE}^{>}$ for the relation $R^{\mathsf{DFA}>}(\mathbf{x}, M) = M(\mathbf{x}) \wedge \big(|\mathbf{x}| \overset{?}{>} |M|\big)$.

It is easy to see that given constructions for the special ABE schemes $\mathsf{dfaABE}^{\le}$ and $\mathsf{dfaABE}^{>}$, we may construct dfaABE simply by running them in parallel. In more detail, the setup algorithm of dfaABE simply runs the setup algorithms of the underlying special ABEs and outputs the public and master secret keys by combining their outputs, the encrypt algorithm encrypts its input $(\mathbf{x}, \mu)$ under both special ABEs, the key generation algorithm produces a key under both special ABEs and the decryption algorithm invokes the decryption of one or the other depending on whether $|\mathbf{x}| \overset{?}{\le} |M|$. This intuition is formalized in Section 3.1, where we prove the following theorem:

Theorem 3. *(Informal) Assume that* $\mathsf{dfaABE}^{\le}$ *and* $\mathsf{dfaABE}^{>}$ *are secure ABE schemes for relations* $R^{\mathsf{DFA}\le}$ *and* $R^{\mathsf{DFA}>}$ *respectively. Then, the scheme* dfaABE *constructed in Section 3.1 is a secure ABE for relation* R^{DFA}.

Step 4: Plugging the gap with ciphertext policy ABE. We already constructed an ABE for the case of $|\mathbf{x}| \leq |M|$. The case of $|\mathbf{x}| > |M|$ is more challenging, since to use ABE for MSP, it is necessary that the MSP be large enough to read the input as we have discussed above. To handle this, we simply switch the role of key generator and encryptor! In more detail, if the encryptor could instead embed $\mathbf{x}$ into an MSP and the key generator could embed M into a set of attributes, then the dilemma of compatible sizes could be resolved and we would be back in business. We show that this can be done; we provide a maps in Section 4.2 that achieves this embedding. More formally, we prove that:

Theorem 4. *Let $(\mathbf{L}_{\mathbf{x}}, \rho_{\mathbf{x}})$ be the MSP and S_M be the set of attributes obtained by applying the map specified in Section 4.2 to $\mathbf{x}$ and M respectively. Then, the MSP $(\mathbf{L}_{\mathbf{x}}, \rho_{\mathbf{x}})$ accepts attributes S_M iff $M(\mathbf{x}) = 1$.*

In order to support encryption of an MSP $(\mathbf{L}_{\mathbf{x}}, \rho_{\mathbf{x}})$, we now need an unbounded *ciphertext policy* ABE for MSP. In more detail, we define $R^{\mathsf{CP}}((\mathbf{L}, \rho), S) = 1$ iff the span program $(\mathbf{L}, \rho)$ accepts the attribute set S. Recall that $R^{\mathsf{DFA}>}(\mathbf{x}, M) = M(\mathbf{x}) \wedge \big(|\mathbf{x}| \overset{?}{>} |M|\big)$. Then, we show in Section 3.3 that:

Theorem 5. *(Informal.) Let cpABE be a secure unbounded ABE scheme for the relation R^{CP}. Then the construction $\mathsf{dfaABE}^{>}$ provided in Section 3.3 is a secure ABE for the relation $R^{\mathsf{DFA}>}$.*

To summarize, our approach is based on the observation that we must only construct an MSP of length $\max(|\mathbf{x}|, |M|)$, where $|\mathbf{x}|$ is known to the encryptor and $|M|$ is known to the key generator (and neither know the other). When the input vector has size $|\mathbf{x}| \leq |M|$, we embed the DFA into a monotone span program which has number of rows proportional to $|M|$, and the input into a set of attributes – this ensures that the MSP is large enough to support an input of length $|\mathbf{x}|$. We may then leverage an unbounded kpABE scheme to handle this case. On the other hand, when $|\mathbf{x}| > |M|$, we instead embed the input vector into a monotone span program which has number of rows proportional to $|\mathbf{x}|$, and the machine into a set of attributes – this again ensures that the MSP is large enough to support an input of size $|M|$. We may then leverage an unbounded cpABE scheme to handle this case. Of course, neither party knows which case it must support, so it simply provides information for both and leaves it to the decryptor to make the choice!

Step 5: Instantiating the kpABE *and* cpABE. Finally, we must ensure that we can instantiate unbounded ABE schemes kpABE and cpABE for the relations R^{KP} and R^{CP} that we require. While prior work provides constructions of unbounded key policy and ciphertext policy ABE schemes for MSP, these unfortunately cannot be plugged into our compiler out of the box. This is because our construction requires the ABE schemes to support "multi-use" of attributes, i.e. when the map ρ in the MSP is not restricted to be injective. Moreover, the ABE schemes are required to be unbounded, as already discussed above. Finally, we want the schemes to be proven secure from static assumptions such as DLIN, not from q-type assumptions. Schemes achieving all these properties do not exist in the literature to the best of our knowledge.[2] Hence, we must refashion existing schemes to satisfy this. In the full version of our paper [6], we provide constructions for multi-use unbounded key policy and ciphertext policy ABE schemes by modifying the constructions in [23]. Let R^{MUKP} and R^{MUCP} be the same relations as R^{KP} and R^{CP} defined above, but with the requirement that the underlying MSPs in both relations support multi-use of attributes. Then, we obtain the following theorem:

Theorem 6. *(Informal.) The constructions* kpABE *provided in [6] (Section 5.2) and* cpABE *provided in [6] (Section 5.4) are unbounded ABE schemes for the relations R^{MUKP} and R^{MUCP} respectively. Security of* kpABE *relies on the* MDDH *assumption and security of* cpABE *relies on the* DLIN *assumption.*

For both KP and CP-ABE schemes, we simply modify the schemes in [23] so that we allow multi-use of the same attribute in an MSP. However, this simple modification ruins the original security proof given by [23] in both cases. The reason is that the core statistical argument in the security proof does not work any more in the multi-use setting. Intuitively, the problem is that the terms used as "one-time pads" in the single-use setting are used multiple times in the multi-use setting. In both KP and CP cases, we switch to weaker security notions than adaptive security and give security proofs by taking advantage of weaker setting.

For KP-ABE scheme, we prove semi-adaptive security. To prove the security, we first use the handy bilinear entropy expansion lemma [23] to create an instance of a multi-use variant of the KP-ABE scheme by [50] (hereafter denoted by LOSTW) in the semi-functional space. To give a proof, we decompose the LOSTW secret key into smaller pieces and gradually add semi-functional randomness to them through a hybrid argument in a way that their distribution depends on the challenge attribute, in a similar manner to [1]. Since this step requires the knowledge of the challenge attribute, we can only prove semi-adaptive security of the scheme. Intuitively, because of this decomposition, we use the "one-time pad" only single time in one hybrid game and can avoid getting into the aforementioned problem of using one-time pads multiple times. Finally, we can use the core statistical step similarly to the case of single-use setting.

For CP-ABE scheme, we prove the security notion that we call selective* security, where the adversary is forced to choose its key queries and the challenge attribute after seeing the master public key. The first step of the proof is similar to the KP-ABE case. Namely, we first use the bilinear entropy expansion lemma [23] to create an instance of the LOSTW CP-ABE scheme in the semi-functional space. However, in the next step, we cannot use the above decomposition idea due to technical reasons, which in turn prohibits us from using the statistical argument in the core step. We overcome this by using computational argument instead, which uses the DLIN assumption instead. The idea of using computational argument here was taken from some of prior works [51, 12, 13].

Putting together these pieces yields our final result – a key-policy ABE for DFA that supports unbounded inputs, unbounded machines and unbounded key requests.

[2] Only exception is the very recent construction by Kowalczyk and Wee [46]. However, their scheme can only deal with NC_1 circuit instead of general MSP and thus our embedding of DFA into MSP cannot be used.

Ciphertext Policy ABE for DFA. In the above description, note that our construction dfaABE uses the underlying kpABE and cpABE in a symmetric way. Thus, by swapping the use of kpABE and cpABE in our construction, we can equivalently construct ciphertext policy ABE for DFA.

In more detail, we exchange the maps used by KeyGen and Enc in the constructions of $\mathsf{dfaABE}^{\leq}$ and $\mathsf{dfaABE}^{>}$ in Sections 3.2 and 3.3. Please see Section 5 for more details. Thus, we obtain

Theorem 7. *There exists a secure key-policy and ciphertext-policy ABE for R^{DFA} from the* DLIN *assumption.*

1.3 Related Work.

In this section, we discuss the related work in the area, categorized by hardness assumptions. We begin with constructions based on bilinear maps. The first construction of ABE for DFA was given by Waters [57] as discussed above. This scheme achieved selective security, which was improved to adaptive by Attrapadung [12]. For span programs, there have been many constructions [48, 53, 50, 49, 47, 54, 55, 24, 25, 58, 12, 22, 14, 45, 13, 2, 23] that achieve various tradeoffs between security (selective versus adaptive), assumptions (static versus parametrized), underlying mathematical structure (prime versus composite order groups), policy embedding (key versus ciphertext policy) and efficiency. In this work, we are particularly concerned with unbounded ABE schemes, in particular those by [54, 23].

From the Learning With Errors assumption (LWE), Boyen and Li [20] provided a construction of ABE for DFA, but this was restricted to DFAs with *bounded* length inputs, rendering moot the primary advantage of a DFA over circuits. Recently, Ananth and Fan [8] provided an ABE for random access machines from LWE, but this construction is also restricted to inputs of bounded length. Agrawal and Singh [7] constructed a primitive closely related to ABE for DFA, namely *reusable garbled DFA* from LWE, but their construction is only secure in the single key setting, namely, where the adversary is limited to requesting a single function key. In contrast, we support unbounded key requests in this work.

From strong assumptions such as the the existence of multilinear maps [31], witness encryption [34] or indistinguishability obfuscation [15, 32], attribute based encryption (or its more powerful generalization – *functional encryption*) has been constructed even for Turing machines [10, 4, 44], but these are not considered standard assumptions; indeed many candidate constructions have been broken [26, 29, 42, 28, 27, 52, 30, 11].

Also relevant to our work are the constructions of [21, 40], which provide attribute based encryption for the so called “bundling functionalities”. Here, the size of the public parameters does not depend on the length of the input (say ℓ) chosen by the encryptor. However, the key generator must generate a key for a circuit with a fixed input length (say ℓ'), and decryption only succeeds if $\ell = \ell'$. Thus, bundling functionalities do not capture the essential challenge of supporting dynamic data sizes as discussed in [40].

1.4 Concurrent Work.

We note that a concurrent work by Gong et. al. [36] constructs KP-ABE scheme for DFA relying on the k-LIN assumption. Although there is a qualitative overlap in our final results as shown in Table 1, the approaches and techniques in their work are quite different from ours. They construct KP-ABE from scratch imitating the transition function of a DFA using bilinear maps directly. This, in turn, yields a scheme with better concrete efficiency and security than ours. In particular, in the KP-ABE setting, our ciphertexts and keys scale as $O(|\mathbf{x}|^3)$ and $O(|Q|^2)$ respectively while the ciphertexts and keys in [36] scale linearly as $O(|\mathbf{x}|)$ and $O(|Q|)$ respectively. Also, our construction achieves selective* security based on DLIN assumption, while their construction achieves selective security and relies on the slightly weaker k-LIN assumption. On the other hand, our scheme is a generic compiler, and has conceptual advantages: our construction is modular and simpler and yields CP-ABE essentially for free. Further, it reduces the question of adaptive security for DFA for both KP-ABE and CP-ABE to that of adaptive security for unbounded KP-ABE and CP-ABE for MSP from static assumptions.

Organization of the paper. In Section 2, we provide the definitions and preliminaries we require. In Section 3, we provide our ABE for DFA supporting unbounded input and unbounded machines from kpABE and cpABE for monotone span programs. In Section 4, we describe how to encode DFA computation into a monotone span program (MSP): Section 4.1 shows the encoding procedure for any DFA machine to a MSP (and DFA input to attribute set) while Section 4.2 shows the encoding procedure for any input string to a MSP (and DFA machine to attribute set). In the full version of our paper [6], we instantiate our ingredient kpABE and cpABE using techniques from [23]. In Section 5 we put together all ingredients to instantiate our ABE for DFA.

2 Preliminaries

In this section, we define some notation and preliminaries that we require.

Notation. We use bold letters to denote vectors and the notation $[a, b]$ to denote the set of integers $\{k \in \mathbb{N} \mid a \leq k \leq b\}$. We use $[n]$ to denote the set $[1, n]$. Concatenation is denoted by the symbol $\|$.

We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use $\mathrm{negl}(n)$ to denote a negligible function of n. We say $f(n)$ is *polynomial* if it is $O(n^c)$ for some constant $c > 0$, and we use $\mathrm{poly}(n)$ to denote a polynomial function of n. We use the abbreviation PPT for probabilistic polynomial-time. We say an event occurs with *overwhelming probability* if its probability is $1 - \mathrm{negl}(n)$.

2.1 Definitions: Restricted Monotone Span Programs (MSP)

A monotone span program over $\mathbb{Z}_p$ is specified by a pair $(\mathbf{L}, \rho)$ of a matrix and a labelling function where

$$\mathbf{L} \in \mathbb{Z}_p^{\ell \times m} \qquad\qquad \rho : [\ell] \to \mathbb{Z}$$

for some integer ℓ, m. Intuitively, the map ρ labels row i with attribute $\rho(i)$.

A span program takes as input a set of integers and accepts or rejects an input by the following criterion. Let $S = \{u_1, \ldots, u_t\} \subseteq \mathbb{Z}$ be a set of integers. Intuitively, each u_i represents some attribute. For the set S, we define another set $I \subseteq [\ell]$ as $I = \{i \in [\ell] : \rho(i) \in S\}$ and $\mathbf{L}_I$ as the submatrix of $\mathbf{L}$ restricted to set of rows I, i.e. obtained by removing row j of $\mathbf{L}$ for any $j \notin I$. We say that

$$(\mathbf{L}, \rho) \text{ accepts } S \text{ iff } (1, 0, \ldots, 0) \text{ is in the row span of } \mathbf{L}_I.$$

We can write this also as $\mathbf{e}_1 \in \mathrm{span}(\mathbf{L}_I^\top)$.

2.2 Deterministic Finite Automata

A Deterministic Finite Automaton (DFA) M is represented by the tuple $(Q, \Sigma, T, q_{\mathsf{st}}, F)$ where Q is a finite set of states, Σ is a finite alphabet, $T : \Sigma \times Q \to Q$ is the transition function (stored as a table), q_{st} is the start state, $F \subseteq Q$ is the set of accepting states. We say that M accepts $\mathbf{x} = (x_1, \ldots, x_k) \in \Sigma^k$ if there exists a sequence of states $q_1, \ldots, q_{k+1}$ such that $q_1 = q$, $q_{i+1} \in T(x_i, q_i)$ for $i \in [k]$ and $q_{k+1} \in F$. We assume w.l.o.g. that the states are numbered as 1 to $|Q|$, i.e., $Q = \{1, 2, \ldots, |Q|\}$ with $q_{\mathsf{st}} = 1$ along with $\Sigma = \{0, 1\}$ and $F = \{|Q|\}$. Note that any DFA with many accepting states can be converted to a DFA with a single accepting state [3], and states may be renumbered so that the last state is the accepting one.

2.3 Definition: Attribute-Based Encryption

Syntax. Let $R : A \times B \to \{0, 1\}$ be a relation where A and B denote "ciphertext attribute" and "key attribute" spaces. An attribute based encryption scheme for R is defined by the following PPT algorithms:

$\mathsf{Setup}(1^\lambda) \to (\mathsf{mpk}, \mathsf{msk})$: The setup algorithm takes as input the unary representation of the security parameter λ and outputs a master public key mpk and a master secret key msk.

$\mathsf{Encrypt}(\mathsf{mpk}, \mu, X) \to \mathsf{ct}$: The encryption algorithm takes as input a master public key mpk, the message bit μ, and a ciphertext attribute $X \in A$. It outputs a ciphertext ct.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{mpk}, Y) \to \mathsf{sk}_Y$: The key generation algorithm takes as input the master secret key msk, the master public key mpk, and a key attribute $Y \in B$. It outputs a private key sk_Y.

$\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{ct}, X, \mathsf{sk}_Y, Y) \to \mu$ or $\perp$: We assume that the decryption algorithm is deterministic. The decryption algorithm takes as input the master public key mpk, a ciphertext ct, ciphertext attribute $X \in A$, a private key sk_Y, and private key attribute Y. It outputs the message μ or $\perp$ which represents that the ciphertext is not in a valid form.

[3] In more detail, we may map any input $\mathbf{x} \in \{0, 1\}^*$ to $\mathbf{x} \| \star$, where $\star$ is a special symbol, and modify M so that we change the accepting state to be $\{|Q| + 1\}$ and add edges from the previous accepting state to $|Q| + 1$, where edges are labelled with $\star$.

We require the standard correctness of decryption: for all λ, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $X \in A, Y \in B$ such that $R(X, Y) = 1$, and $\mathsf{sk}_Y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{mpk}, Y)$, we have $\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{Encrypt}(\mathsf{mpk}, \mu, X), X, \mathsf{sk}_Y, Y) = \mu$.

Security. We now define the security for an ABE scheme Π by the following game between a challenger and an attacker $\mathcal{A}$.

- At first, the challenger runs the setup algorithm and gives mpk to $\mathcal{A}$.
- Then $\mathcal{A}$ may adaptively make key-extraction queries. We denote this phase PHASE1. In this phase, if $\mathcal{A}$ submits $Y \in B$ to the challenger, the challenger returns $\mathsf{sk}_Y \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{mpk}, Y)$.
- At some point, $\mathcal{A}$ outputs two equal length messages μ_0 and μ_1 and challenge ciphertext attribute $X^\star \in A$. $X^\star$ cannot satisfy $R(X^\star, Y) = 1$ for any attribute Y such that $\mathcal{A}$ already queried private key for Y.
- Then the challenger flips a random coin $\beta \in \{0, 1\}$, runs $\mathsf{Encrypt}(\mathsf{mpk}, \mu_\beta, X^\star) \rightarrow \mathsf{ct}^\star$ and gives challenge ciphertext $\mathsf{ct}^\star$ to $\mathcal{A}$.
- In PHASE2, $\mathcal{A}$ may adaptively make queries as in PHASE1 with following added restriction: $\mathcal{A}$ cannot make a key-extraction query for Y such that $R(X^\star, Y) = 1$.
- At last, $\mathcal{A}$ outputs a guess β' for β.

We say that $\mathcal{A}$ succeeds if $\beta' = \beta$ and denote the probability of this event by $\Pr^{\mathsf{ABE}}_{\mathcal{A},\Pi}$. The advantage of an attacker $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathsf{ABE}}_{\mathcal{A},\Pi} = |\Pr^{\mathsf{ABE}}_{\mathcal{A},\Pi} - \frac{1}{2}|$. We say that Π is adaptively secure if $\mathsf{Adv}^{\mathsf{ABE}}_{\mathcal{A},\Pi}$ is negligible for all probabilistic polynomial time (PPT) adversary $\mathcal{A}$.

Weaker Security Notions. A weaker notion called selective security can be defined as in the above game with the exception that the adversary $\mathcal{A}$ has to choose the challenge ciphertext attribute $X^\star$ before the setup phase but private key queries $Y_1, \ldots, Y_k$ and choice of (μ_0, μ_1) can still be adaptive. The stronger notion of semi-adaptive security lets the adversary output the challenge ciphertext attribute $X^\star$ after seeing the public key but before making any key requests. The still weaker notion of very selective security requires the adversary to output the challenge ciphertext attribute and private key queries at the very start of the game. An intermediate notion to semi-adaptive and very selective, which we term selective*, allows the adversary to receive the public parameters in the first step, but it must specify the challenge ciphertext attribute and private key queries after this step.

ABE for DFA. We then define ABE for DFA by specifying the relation. We define $A^{\mathsf{DFA}} = \{0, 1\}^*$ and B^{DFA} as the set of all DFA, also represented as strings over $\{0, 1\}^*$. Furthermore, we define the relation $R^{\mathsf{DFA}} = \{A^{\mathsf{DFA}} \times B^{\mathsf{DFA}} \rightarrow \{0, 1\}\}$ as $R^{\mathsf{DFA}}(\mathbf{x}, M) = M(\mathbf{x})$.

An ABE scheme for the relation R^{DFA} is said to be ABE for DFA. We further define $R^{\mathsf{DFA}\leq} = \{A^{\mathsf{DFA}} \times B^{\mathsf{DFA}} \rightarrow \{0, 1\}\}$ as

$$R^{\mathsf{DFA}\leq}(\mathbf{x}, M) = M(\mathbf{x}) \wedge \big(|\mathbf{x}| \overset{?}{\leq} |Q|\big),$$

where $|Q|$ is the number of states in M. We also define $R^{\mathsf{DFA}>}$ analogously.

Unbounded ABE for MSP. Here, we define unbounded ABE for MSP. There are distinctions between "single-use" and "multi-use" as well as "key-policy" and "ciphertext-policy". We first define multi-use key-policy unbounded ABE by specifying the relation R^{MUKP}. To do so, we set $A^{\mathsf{MUKP}} := 2^{\mathbb{Z}}$ (i.e., the set of all subsets of $\mathbb{Z}$) and B^{MUKP} as the set of monotone span programs on $\mathbb{Z}_p$ for some prime p, and $R^{\mathsf{MUKP}}(S, (\mathbf{L}, \rho)) = 1$ iff the span program $(\mathbf{L}, \rho)$ accepts the set $S \in A^{\mathsf{MUKP}}$. An ABE for R^{MUKP} is said to be "multi-use key-policy unbounded ABE".

We also define single-use key-policy unbounded ABE by specifying the relation R^{SUKP}. We set $A^{\mathsf{SUKP}} := 2^{\mathbb{Z}}$ and B^{SUKP} as the set of monotone span programs $(\mathbf{L}, \rho)$ such that ρ is injective. We define $R^{\mathsf{SUKP}}(S, (\mathbf{L}, \rho)) = 1$ iff the span program $(\mathbf{L}, \rho)$ accepts the set S. Finally, we can define the ciphertext variant of the above ABE by specifying R^{SUCP} and R^{MUCP}, where we set $A^{\mathrm{xxCP}} = B^{\mathrm{xxKP}}$ and $B^{\mathrm{xxCP}} = A^{\mathrm{xxKP}}$ for $\mathrm{xx} \in \{\mathsf{SU}, \mathsf{MU}\}$ and define the relation analogously.

Unbounded ABE for MSP with polynomial-valued attributes. We can consider a restricted variant of unbounded ABE for MSP where the value of attributes being used is polynomially bounded. Here, we focus on the case of multi-use and key-policy case. Other cases will be defined similarly. Here, we define $A^{\mathsf{MUKP}'}$ and $B^{\mathsf{MUKP}'}$ as

$$A^{\mathsf{MUKP}'} = \left\{ (S, 1^{s_{\max}}) : S \subseteq \mathbb{Z}, s_{\max} = \max_{s \in S} |s| \right\} \qquad \text{and}$$

$$B^{\mathsf{MUKP}'} = \left\{ ((\mathbf{L}, \rho), 1^{\rho_{\max}}) : (\mathbf{L}, \rho) \text{ is a span program over } \mathbb{Z}_p,\ \rho_{\max} = \max_{i \in [\ell]} |\rho(i)| \right\}$$

We define $R^{\mathsf{MUKP}'}(S, (\mathbf{L}, \rho)) := R^{\mathsf{MUKP}}(S, (\mathbf{L}, \rho))$. Here, the reason why we append $1^{s_{\max}}$ to S is somewhat technical. This is to enforce the adversary in the security definition who declares $S \in A^{\mathsf{MUKP}'}$ as its target to choose attributes with polynomially bounded values. Because of the similar reason, we append $1^{\rho_{\max}}$ to $(\mathbf{L}, \rho)$.

For ease of readability in the remainder of the paper, we will overload notation and denote $A^{\mathsf{MUKP}'}$ and $B^{\mathsf{MUKP}'}$ as A^{MUKP} and B^{MUKP} respectively. However, all our constructions will satisfy the constraint of attribute values being polynomially bounded.

2.4 Embedding Lemma for ABE

Here, we introduce a useful lemma that describes a sufficient criterion for implication from an ABE for a given predicate to an ABE for another predicate. The lemma is introduced in [18] and later formally proven in [14]. The presentation here follows that of [14] with some simplifications. The lemma is applicable to any relation family. We consider two relation families:

$$R^{\mathsf{F}} : A \times B \to \{0, 1\}, \qquad\qquad R^{\mathsf{F}'} : A' \times B' \to \{0, 1\}.$$

Suppose that there exists two efficient mappings $f_{\mathsf{e}} : A' \to A$ and $f_{\mathsf{k}} : B' \to B$ which map parameters, ciphertext attributes, and key attributes, respectively, such that for all $X' \in A', Y' \in B'$,

$$R^{\mathsf{F}'}(X', Y') = 1 \Leftrightarrow R^{\mathsf{F}}(f_{\mathsf{e}}(X'), f_{\mathsf{k}}(Y')) = 1. \tag{2.1}$$

We can then construct an ABE scheme $\Pi' = \{\mathsf{Setup}', \mathsf{Encrypt}', \mathsf{KeyGen}', \mathsf{Decrypt}'\}$ for predicate $R^{\mathsf{F}'}$ from an ABE scheme $\Pi = \{\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt}\}$ for predicate R^{F} as follows. Let $\mathsf{Setup}' = \mathsf{Setup}$ and

$$\mathsf{Encrypt}'(\mathsf{mpk}, \mu, X') = \mathsf{Encrypt}(\mathsf{mpk}, \mu, f_{\mathsf{e}}(X')),$$
$$\mathsf{KeyGen}'(\mathsf{msk}, \mathsf{mpk}, Y') = \mathsf{KeyGen}(\mathsf{msk}, \mathsf{mpk}, f_{\mathsf{k}}(Y')),$$

and $\mathsf{Decrypt}'(\mathsf{mpk}, \mathsf{ct}, X', \mathsf{sk}_{Y'}, Y') = \mathsf{Decrypt}(\mathsf{mpk}, \mathsf{ct}, f_{\mathsf{e}}(X'), \mathsf{sk}_{Y'}, f_{\mathsf{k}}(Y'))$.

Lemma 1 (Embedding lemma [18, 14]). *If Π is correct and secure, then so is Π'. This holds for very selective, selective, selective* and adaptive security.*

Intuitively, the forward and backward direction of Relation (2.1) ensure that the correctness and the security are preserving, respectively.

3 Attribute-based Encryption for DFA

We construct an ABE scheme for DFA denoted by $\mathsf{dfaABE} = (\mathsf{dfaABE.Setup}, \mathsf{dfaABE.KeyGen}, \mathsf{dfaABE.Enc}, \mathsf{dfaABE.Dec})$. Following the notation of Section 2, we achieve this by constructing an ABE scheme for the relation $R^{\mathsf{DFA}} = \{A^{\mathsf{DFA}} \times B^{\mathsf{DFA}} \rightarrow \{0,1\}\}$ which is defined as $R^{\mathsf{DFA}}(\mathbf{x}, M) = M(\mathbf{x})$. Recall that A^{DFA} is the set of all input strings and B^{DFA} is the set of all DFA. Let $|Q|$ be the number of states in M. As described in Section 1, our construction relies on two special ABE for DFA as follows:

1. An ABE denoted by $\mathsf{dfaABE}^{\leq}$ for the relation $R^{\mathsf{DFA}\leq} = \{A^{\mathsf{DFA}} \times B^{\mathsf{DFA}} \rightarrow \{0,1\}\}$ defined as:
$$R^{\mathsf{DFA}\leq}(\mathbf{x}, M) = M(\mathbf{x}) \wedge \left(|\mathbf{x}| \overset{?}{\leq} |Q|\right)$$
2. An ABE denoted by $\mathsf{dfaABE}^{>}$ for the relation $R^{\mathsf{DFA}>} = \{A^{\mathsf{DFA}} \times B^{\mathsf{DFA}} \rightarrow \{0,1\}\}$ defined as:
$$R^{\mathsf{DFA}>}(\mathbf{x}, M) = M(\mathbf{x}) \wedge \left(|\mathbf{x}| \overset{?}{>} |Q|\right)$$

It is easy to see that given constructions for $\mathsf{dfaABE}^{\leq}$ and $\mathsf{dfaABE}^{>}$, we may construct dfaABE simply by running them in parallel. This intuition is formalized in Section 3.1.

Then, it suffices to construct the ingredients $\mathsf{dfaABE}^{\leq}$ and $\mathsf{dfaABE}^{>}$ – we do so by leveraging existing constructions of *unbounded* kpABE and cpABE for monotone span programs. Since the intuition was discussed in Section 1, we directly provide the constructions in Section 3.2 and Section 3.3 respectively.

3.1 Construction of dfaABE

Below, we describe the construction of our ABE for DFA formally. We denote our construction as dfaABE.

$\mathsf{dfaABE.Setup}(1^\lambda)$: On input the security parameter 1^λ, do the following:

1. Invoke $\mathsf{dfaABE}^{\leq}.\mathsf{Setup}(1^\lambda)$ and $\mathsf{dfaABE}^{>}.\mathsf{Setup}(1^\lambda)$ to obtain $(\mathsf{dfaABE}^{\leq}.\mathsf{mpk}, \mathsf{dfaABE}^{\leq}.\mathsf{msk})$ and $(\mathsf{dfaABE}^{>}.\mathsf{mpk}, \mathsf{dfaABE}^{>}.\mathsf{msk})$ respectively.
2. Output $\mathsf{dfaABE}.\mathsf{mpk} = (\mathsf{dfaABE}^{\leq}.\mathsf{mpk}, \mathsf{dfaABE}^{>}.\mathsf{mpk})$ and $\mathsf{dfaABE}.\mathsf{msk} = (\mathsf{dfaABE}^{\leq}.\mathsf{msk}, \mathsf{dfaABE}^{>}.\mathsf{msk})$.

$\mathsf{dfaABE}.\mathsf{Enc}(\mathsf{dfaABE}.\mathsf{mpk}, \mu, \mathbf{x})$: On input the master public key $\mathsf{dfaABE}.\mathsf{mpk}$, a message bit μ, and an attribute $\mathbf{x} \in A^{\mathsf{DFA}}$ of unbounded polynomial length (i.e., bounded by 2^λ), do the following:

1. Compute $\mathsf{ct}_1 = \mathsf{dfaABE}^{\leq}.\mathsf{Enc}(\mathsf{dfaABE}^{\leq}.\mathsf{mpk}, \mu, \mathbf{x})$.
2. Compute $\mathsf{ct}_2 = \mathsf{dfaABE}^{>}.\mathsf{Enc}(\mathsf{dfaABE}^{>}.\mathsf{mpk}, \mu, \mathbf{x})$.
3. Output $(\mathsf{ct}_1, \mathsf{ct}_2)$.

$\mathsf{dfaABE}.\mathsf{KeyGen}(\mathsf{dfaABE}.\mathsf{msk}, \mathsf{dfaABE}.\mathsf{mpk}, M)$: On input the master secret key $\mathsf{dfaABE}.\mathsf{msk}$, the description of a DFA $M \in B^{\mathsf{DFA}}$ do the following:

1. Compute $\mathsf{sk}_1 = \mathsf{dfaABE}^{\leq}.\mathsf{KeyGen}(\mathsf{dfaABE}^{\leq}.\mathsf{msk}, \mathsf{dfaABE}^{\leq}.\mathsf{mpk}, M)$.
2. Compute $\mathsf{sk}_2 = \mathsf{dfaABE}^{>}.\mathsf{KeyGen}(\mathsf{dfaABE}^{>}.\mathsf{msk}, \mathsf{dfaABE}^{>}.\mathsf{mpk}, M)$.
3. Output $(\mathsf{sk}_1, \mathsf{sk}_2)$.

$\mathsf{dfaABE}.\mathsf{Dec}(\mathsf{dfaABE}.\mathsf{mpk}, \mathsf{dfaABE}.\mathsf{ct}, \mathbf{x}, \mathsf{dfaABE}.\mathsf{sk}_M, M)$: On input a ciphertext encoded under attribute $\mathbf{x}$ and a secret key for DFA M, proceed as follows. Let $|Q|$ be the number of states in the machine M.

1. If $|\mathbf{x}| \leq |Q|$, compute $\mu_1 \leftarrow \mathsf{dfaABE}^{\leq}.\mathsf{Dec}(\mathsf{dfaABE}^{\leq}.\mathsf{mpk}, \mathsf{ct}_1, \mathbf{x}, \mathsf{sk}_1, M)$ and output it.
2. If $|\mathbf{x}| > |Q|$, compute $\mu_2 \leftarrow \mathsf{dfaABE}^{>}.\mathsf{Dec}(\mathsf{dfaABE}^{>}.\mathsf{mpk}, \mathsf{ct}_2, \mathbf{x}, \mathsf{sk}_2, M)$ and output it.

Correctness. Correctness follows directly from the correctness of the ingredient schemes $\mathsf{dfaABE}^{\leq}$ and $\mathsf{dfaABE}^{>}$, where the former is invoked for the case that $|\mathbf{x}| \leq |Q|$ and the latter otherwise.

Security. Security of the scheme dfaABE follows directly from the security of $\mathsf{dfaABE}^{\leq}$ and $\mathsf{dfaABE}^{>}$. In more detail, we have:

Theorem 8. *Assume that* $\mathsf{dfaABE}^{\leq}$ *and* $\mathsf{dfaABE}^{>}$ *are ABE schemes for relations* $R^{\mathsf{DFA}\leq}$ *and* $R^{\mathsf{DFA}>}$ *respectively, that satisfy selective/selective*/adaptive security. Then,* dfaABE *is an ABE scheme for relation* R^{DFA} *that satisfies selective/selective*/adaptive security.*

The proof is straightforward: for the case that $|\mathbf{x}| \leq |Q|$, the theorem follows from security of $\mathsf{dfaABE}^{\leq}$, otherwise from the security of $\mathsf{dfaABE}^{>}$.

3.2 Construction of $\mathsf{dfaABE}^{\leq}$

In this section, we construct the ABE scheme $\mathsf{dfaABE}^{\leq}$ for the relation $R^{\mathsf{DFA}\leq} = \{A^{\mathsf{DFA}} \times B^{\mathsf{DFA}} \to \{0,1\}\}$ where $R^{\mathsf{DFA}\leq}(\mathbf{x}, M) = M(\mathbf{x}) \wedge \big(|\mathbf{x}| \overset{?}{\leq} |Q|\big)$. Our construction is built from the following ingredients:

1. An ABE scheme for the relation $R^{\mathsf{MUKP}} : A^{\mathsf{MUKP}} \times B^{\mathsf{MUKP}} \to \{0,1\}$. Recall from Section 2, that $A^{\mathsf{MUKP}} := 2^{\mathbb{Z}}$ is the set of attributes, B^{MUKP} is the set of monotone span programs and $R^{\mathsf{MUKP}}(S, (\mathbf{L}, \rho)) = 1$ iff the span program $(\mathbf{L}, \rho)$ accepts the set $S \in A^{\mathsf{MUKP}}$. We denote such a scheme as kpABE, and construct it in the full version of our paper [6] (Section 5.2).
2. A map $f_{\mathsf{e}}^{\mathsf{KP}} : A^{\mathsf{DFA}} \to A^{\mathsf{MUKP}}$ and a map $f_{\mathsf{k}}^{\mathsf{KP}} : B^{\mathsf{DFA}} \to B^{\mathsf{MUKP}}$ so that $R^{\mathsf{MUKP}}(S_{\mathbf{x}}, (\mathbf{L}_M, \rho_M)) = 1$ iff $R^{\mathsf{DFA}\leq}(\mathbf{x}, M) = 1$, where $S_{\mathbf{x}} = f_{\mathsf{e}}^{\mathsf{KP}}(\mathbf{x})$ and $(\mathbf{L}_M, \rho_M) = f_{\mathsf{k}}^{\mathsf{KP}}(M)$. These maps are constructed in Section 4.1.

The scheme $\mathsf{dfaABE}^{\leq}$ is then defined as follows.

$\mathsf{dfaABE}^{\leq}.\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, do the following:
1. Invoke $\mathsf{kpABE.Setup}(1^\lambda)$ to obtain $(\mathsf{kpABE.mpk}, \mathsf{kpABE.msk})$.
2. Output $\mathsf{dfaABE}^{\leq}.\mathsf{mpk} = \mathsf{kpABE.mpk}$ and $\mathsf{dfaABE}^{\leq}.\mathsf{msk} = \mathsf{kpABE.msk}$.

$\mathsf{dfaABE}^{\leq}.\mathsf{Enc}(\mathsf{dfaABE}^{\leq}.\mathsf{mpk}, \mu, \mathbf{x})$: On input the master public key $\mathsf{dfaABE}^{\leq}.\mathsf{mpk}$, a message bit μ, and an attribute $\mathbf{x} \in A^{\mathsf{DFA}}$ of unbounded polynomial length (i.e. length at most 2^λ), do the following:
1. Convert $\mathbf{x}$ to attribute $S_{\mathbf{x}}$ by computing $S_{\mathbf{x}} = f_{\mathsf{e}}^{\mathsf{KP}}(\mathbf{x})$ as described in Section 4.1.
2. Compute $\mathsf{ct} = \mathsf{kpABE.Enc}(\mathsf{kpABE.mpk}, \mu, S_{\mathbf{x}})$ and output it.

$\mathsf{dfaABE}^{\leq}.\mathsf{KeyGen}(\mathsf{dfaABE}^{\leq}.\mathsf{msk}, \mathsf{dfaABE}^{\leq}.\mathsf{mpk}, M)$: On input the master secret key $\mathsf{dfaABE}^{\leq}.\mathsf{msk}$, the description of a DFA $M \in B^{\mathsf{DFA}}$ do the following:
1. Convert M into an MSP $(\mathbf{L}_M, \rho_M)$ by computing $(\mathbf{L}_M, \rho_M) = f_{\mathsf{k}}^{\mathsf{KP}}(M)$ as described in Section 4.1.
2. Compute $\mathsf{sk}_M = \mathsf{kpABE.KeyGen}\big(\mathsf{kpABE.msk}, \mathsf{kpABE.mpk}, (\mathbf{L}_M, \rho_M)\big)$ and output it.

$\mathsf{dfaABE}^{\leq}.\mathsf{Dec}(\mathsf{dfaABE}^{\leq}.\mathsf{mpk}, \mathsf{dfaABE}^{\leq}.\mathsf{ct}, \mathbf{x}, \mathsf{dfaABE}^{\leq}.\mathsf{sk}_M, M)$: On input a ciphertext encoded under attribute $\mathbf{x}$ and a secret key for DFA M:
1. Compute $S_{\mathbf{x}} = f_{\mathsf{e}}^{\mathsf{KP}}(\mathbf{x})$ and $(\mathbf{L}_M, \rho_M) = f_{\mathsf{k}}^{\mathsf{KP}}(M)$ as described in Section 4.1.
2. Compute $\mu \leftarrow \mathsf{kpABE.Dec}\big(\mathsf{kpABE.mpk}, \mathsf{kpABE.ct}, S_{\mathbf{x}}, \mathsf{sk}_M, (\mathbf{L}_M, \rho_M)\big)$ and output it.

Correctness and Security. Correctness and security follow directly from the "embedding lemma" (Lemma 1) provided in Section 2 by setting

$$A' = A^{\mathsf{DFA}}, \quad B' = B^{\mathsf{DFA}}, \quad R^{F'} = R^{\mathsf{DFA}\leq},$$
$$A = A^{\mathsf{MUKP}}, \quad B = B^{\mathsf{MUKP}}, \quad R^{F} = R^{\mathsf{MUKP}}$$

In more detail, we have the following theorem.

Theorem 9. *Assume that* kpABE *is an ABE scheme for relation* R^{MUKP} *satisfying selective/selective*/adaptive security. Then,* $\mathsf{dfaABE}^{\leq}$ *is an ABE scheme for relation* $R^{\mathsf{DFA}\leq}$ *satisfying selective/selective*/adaptive security.*

3.3 Construction of $\mathsf{dfaABE}^{>}$

In this section, we construct the ABE scheme $\mathsf{dfaABE}^{>}$ for the relation $R^{\mathsf{DFA}>} = \{A^{\mathsf{DFA}} \times B^{\mathsf{DFA}} \to \{0,1\}\}$ where $R^{\mathsf{DFA}>}(\mathbf{x}, M) = M(\mathbf{x}) \wedge \big(|\mathbf{x}| \overset{?}{>} |Q|\big)$. Our construction is built from the following ingredients:

1. An ABE scheme for the relation $R^{\mathsf{MUCP}} : A^{\mathsf{MUCP}} \times B^{\mathsf{MUCP}} \to \{0,1\}$. Recall from Section 2, that A^{MUCP} is the set of all monotone span programs, B^{MUCP} is the set of attributes and $R^{\mathsf{MUCP}}((\mathbf{L}, \rho), S) = 1$ iff the span program $(\mathbf{L}, \rho) \in A^{\mathsf{MUCP}}$ accepts the set $S \in B^{\mathsf{MUCP}}$. We denote such a scheme as cpABE, and construct it in the full version of our paper [6] (Section 5.4).
2. A map $f_{\mathsf{e}}^{\mathsf{CP}} : A^{\mathsf{DFA}} \to A^{\mathsf{MUCP}}$ and a map $f_{\mathsf{k}}^{\mathsf{CP}} : B^{\mathsf{DFA}} \to B^{\mathsf{MUCP}}$ so that $R^{\mathsf{MUCP}}((\mathbf{L}_{\mathbf{x}}, \rho_{\mathbf{x}}), S_M) = 1$ iff $R^{\mathsf{DFA}>}(\mathbf{x}, M) = 1$, where $(\mathbf{L}_{\mathbf{x}}, \rho_{\mathbf{x}}) = f_{\mathsf{e}}^{\mathsf{CP}}(\mathbf{x})$ and $S_M = f_{\mathsf{k}}^{\mathsf{CP}}(M)$. These maps are constructed in Section 4.2.

The scheme $\mathsf{dfaABE}^{>}$ is then defined as follows.

$\mathsf{dfaABE}^{>}.\mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, do the following:
1. Invoke $\mathsf{cpABE}.\mathsf{Setup}(1^\lambda)$ to obtain $(\mathsf{cpABE}.\mathsf{mpk}, \mathsf{cpABE}.\mathsf{msk})$.
2. Output $\mathsf{dfaABE}^{>}.\mathsf{mpk} = \mathsf{cpABE}.\mathsf{mpk}$ and $\mathsf{dfaABE}^{>}.\mathsf{msk} = \mathsf{cpABE}.\mathsf{msk}$.

$\mathsf{dfaABE}^{>}.\mathsf{Enc}(\mathsf{dfaABE}^{>}.\mathsf{mpk}, \mu, \mathbf{x})$: On input the master public key $\mathsf{dfaABE}^{>}.\mathsf{mpk}$, a message μ, and an attribute $\mathbf{x} \in A^{\mathsf{DFA}}$ of unbounded polynomial length (i.e. length at most 2^λ), do the following:
1. Convert $\mathbf{x}$ to MSP $(\mathbf{L}_{\mathbf{x}}, \rho_{\mathbf{x}})$ by computing $(\mathbf{L}_{\mathbf{x}}, \rho_{\mathbf{x}}) = f_{\mathsf{e}}^{\mathsf{CP}}(\mathbf{x})$ as described in Section 4.2.
2. Compute $\mathsf{ct} = \mathsf{cpABE}.\mathsf{Enc}(\mathsf{cpABE}.\mathsf{mpk}, \mu, (\mathbf{L}_{\mathbf{x}}, \rho_{\mathbf{x}}))$ and output it.

$\mathsf{dfaABE}^{>}.\mathsf{KeyGen}(\mathsf{dfaABE}^{>}.\mathsf{msk}, \mathsf{dfaABE}^{>}.\mathsf{mpk}, M)$: On input the master secret key $\mathsf{dfaABE}^{>}.\mathsf{msk}$, the description of a DFA M do the following:
1. Convert M into an attribute S_M by computing $S_M = f_{\mathsf{k}}^{\mathsf{CP}}(M)$ as described in Section 4.2.
2. Compute $\mathsf{sk} = \mathsf{cpABE}.\mathsf{KeyGen}(\mathsf{cpABE}.\mathsf{msk}, \mathsf{cpABE}.\mathsf{mpk}, S_M)$ and output it.

$\mathsf{dfaABE}^{>}.\mathsf{Dec}(\mathsf{dfaABE}^{>}.\mathsf{mpk}, \mathsf{dfaABE}^{>}.\mathsf{ct}, \mathbf{x}, \mathsf{dfaABE}^{>}.\mathsf{sk}_M, M)$: On input a ciphertext encoded under attribute $\mathbf{x}$ and a secret key sk_M for DFA M:
1. Compute $(\mathbf{L}_{\mathbf{x}}, \rho_{\mathbf{x}}) = f_{\mathsf{e}}^{\mathsf{CP}}(\mathbf{x})$ and $S_M = f_{\mathsf{k}}^{\mathsf{CP}}(M)$ as described in Section 4.2.
2. Compute $\mu \leftarrow \mathsf{cpABE}.\mathsf{Dec}(\mathsf{cpABE}.\mathsf{mpk}, \mathsf{cpABE}.\mathsf{ct}, (\mathbf{L}_{\mathbf{x}}, \rho_{\mathbf{x}}), \mathsf{sk}_M, S_M)$ and output it.

Correctness and Security. Correctness and security follow exactly as in Section 3.2, by considering the maps defined in Section 4.2 instead of Section 4.1. In more detail, we have the following theorem:

Theorem 10. *Assume that* cpABE *is an ABE scheme for relation* R^{MUCP} *satisfying selective/selective*/adaptive security. Then,* $\mathsf{dfaABE}^{>}$ *is an ABE scheme for relation* $R^{\mathsf{DFA}>}$ *satisfying selective/selective*/adaptive security.*

4 Mapping DFA Computation to Monotone Span Programs

In this section we will describe how to encode DFA computation over a binary alphabet $\Sigma = \{0,1\}$ into a monotone span program (MSP). Section 4.1 shows the encoding procedure for any DFA machine to a MSP and further how to encode its input to a set of attributes associated with the MSP. In a dual view, Section 4.2 shows the encoding procedure for any input string to a MSP while encoding the DFA machine itself as a set of attributes associated with the MSP. For both sections, we denote any DFA machine as $M = (Q, \Sigma, T, q_{\mathsf{st}}, F)$ and $\mathbf{x} \in \Sigma^*$ as its input of arbitrary (polynomial) length.

4.1 Encoding Deterministic Finite Automata to Monotone Span Programs

In this section, we construct two efficiently computable functions (please see Section 2 for the notation):

1. $f_{\mathsf{e}}^{\mathsf{KP}} : A^{\mathsf{DFA}} \to A^{\mathsf{MUKP}}$ to encode $\mathbf{w} \in A^{\mathsf{DFA}}$ as a set of attributes $S_{\mathbf{w}} \in A^{\mathsf{MUKP}}$, and
2. $f_{\mathsf{k}}^{\mathsf{KP}} : B^{\mathsf{DFA}} \to B^{\mathsf{MUKP}}$ to encode $M \in B^{\mathsf{DFA}}$ into a MSP $(\mathbf{L}_M, \rho_M) \in B^{\mathsf{MUKP}}$.

We argue that $R^{\mathsf{MUKP}}(S_{\mathbf{w}}, (\mathbf{L}_M, \rho_M)) = 1$ iff $R^{\mathsf{DFA}\leq}(\mathbf{w}, M) = 1$, where $S_{\mathbf{w}} = f_{\mathsf{e}}^{\mathsf{KP}}(\mathbf{w})$ and $(\mathbf{L}_M, \rho_M) = f_{\mathsf{k}}^{\mathsf{KP}}(M)$.

For ease of exposition, we represent the universe of attributes in the following form:

$$A^{\mathsf{MUKP}} := \{\text{“}x_i = b\text{”} \mid i \in [2^\lambda], b \in \{0,1\}\} \cup \{\text{“String length} = i\text{”} \mid i \in [2^\lambda]\} \cup \{\text{“Dummy”}\}.$$

We assume that these attributes are embedded into $\mathbb{Z}$ via an injective mapping such as

$$\text{“Dummy”} \mapsto 0, \quad \text{“}x_i = b\text{”} \mapsto 3i + b \quad \text{“String length} = i\text{”} \mapsto 3i + 2.$$

However, for maintaining intuitive notation, we make the mapping implicit. An input string $\mathbf{w} = (w_1, \ldots, w_\ell) \in A^{\mathsf{DFA}}$ of length ℓ is encoded to a set of attributes given by $f_{\mathsf{e}}^{\mathsf{KP}}(\mathbf{w}) = S_{\mathbf{w}} \in A^{\mathsf{MUKP}}$ as:

$$S_{\mathbf{w}} := \{\text{“Dummy”}\} \cup \{\text{“}x_i = w_i\text{”} \mid i \in [\ell]\} \cup \{\text{“String length} = \ell\text{”}\}.$$

When we represent $S_{\mathbf{w}}$ as a set of integers, we have $S_{\mathbf{w}} \subseteq [4\ell]$ and thus in particular, all the values in $S_{\mathbf{w}}$ are bounded by $\mathrm{poly}(\ell)$.

A DFA machine $M = (Q, \Sigma, T, q_{\mathsf{st}}, F) \in B^{\mathsf{DFA}}$ is encoded into a MSP given by $f_{\mathsf{k}}^{\mathsf{KP}}(M) = (\mathbf{L}_M, \rho_M) \in B^{\mathsf{MUKP}}$. Here $\mathbf{L}_M \in \{0, \pm 1\}^{\mathcal{R} \times \mathcal{C}}$ with $\mathcal{R} = 1 + (2 \cdot |Q| + 1) \cdot |Q|$ and $\mathcal{C} = 1 + |Q| + |Q|^2$. The label map ρ_M will be implicit in the description of the matrix $\mathbf{L}_M$. Before providing the construction of $\mathbf{L}_M$, we define the following sub-matrices useful in the construction:

- matrix $\mathbf{I}_Q$ denoting the $|Q| \times |Q|$ identity matrix, and

– matrices $\mathbf{Y}^{(b)} \in \{0,-1\}^{|Q|\times|Q|}, \forall b \in \{0,1\}$ defined as $\mathbf{Y}^{(b)} := \left[y_{i,j}^{(b)}\right]$ such that:

$$\begin{aligned} y_{i,j}^{(b)} &= -1, \text{ if } T(i,b) = j \text{ (i.e. there is a transition from state } i \text{ to state } j \text{ upon input } b) \\ &= \ 0, \text{ otherwise} \end{aligned}$$

We also denote $\mathbf{0}_{Q\times Q}$ to be the all-zero matrix of size $|Q| \times |Q|$ and $\mathbf{0}_Q$ as the column-vector of size $|Q|$ containing all 0s.

We define $\mathbf{L}_M$ and the map ρ_M in Table 2.

"Dummy" $\mapsto$	1	$-10\ldots0$	$0\ldots0$	$0\ldots0$	$\ldots$	$0\ldots0$	$0\ldots0$
"$x_1 = 0$" $\mapsto$	$\mathbf{0}_Q$	$\mathbf{I}_Q$	$\mathbf{Y}^{(0)}$	$\mathbf{0}_{Q\times Q}$	$\ldots$	$\mathbf{0}_{Q\times Q}$	$\mathbf{0}_{Q\times Q}$
"$x_1 = 1$" $\mapsto$	$\mathbf{0}_Q$	$\mathbf{I}_Q$	$\mathbf{Y}^{(1)}$	$\mathbf{0}_{Q\times Q}$	$\ldots$	$\mathbf{0}_{Q\times Q}$	$\mathbf{0}_{Q\times Q}$
"$x_2 = 0$" $\mapsto$	$\mathbf{0}_Q$	$\mathbf{0}_{Q\times Q}$	$\mathbf{I}_Q$	$\mathbf{Y}^{(0)}$	$\ldots$	$\mathbf{0}_{Q\times Q}$	$\mathbf{0}_{Q\times Q}$
"$x_2 = 1$" $\mapsto$	$\mathbf{0}_Q$	$\mathbf{0}_{Q\times Q}$	$\mathbf{I}_Q$	$\mathbf{Y}^{(1)}$	$\ldots$	$\mathbf{0}_{Q\times Q}$	$\mathbf{0}_{Q\times Q}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
"$x_{\|Q\|} = 0$" $\mapsto$	$\mathbf{0}_Q$	$\mathbf{0}_{Q\times Q}$	$\mathbf{0}_{Q\times Q}$	$\mathbf{0}_{Q\times Q}$	$\ldots$	$\mathbf{I}_Q$	$\mathbf{Y}^{(0)}$
"$x_{\|Q\|} = 1$" $\mapsto$	$\mathbf{0}_Q$	$\mathbf{0}_{Q\times Q}$	$\mathbf{0}_{Q\times Q}$	$\mathbf{0}_{Q\times Q}$	$\ldots$	$\mathbf{I}_Q$	$\mathbf{Y}^{(1)}$
"String length $= 1$" $\mapsto$	0	$0\ldots0$	$0\ldots01$				
"String length $= 2$" $\mapsto$	0		$0\ldots00$	$0\ldots01$			
$\vdots$	$\vdots$				$\ddots$		
"String length $= \|Q\|$" $\mapsto$	0					$0\ldots00$	$0\ldots01$

Table 2. Encoding a DFA M to matrix $\mathbf{L}_M$

We observe that $\max_i \rho_M(i) \leq 4|Q|$, where we regard the attributes as integers through the aforementioned injective mapping. In particular, $\mathbf{L}_M$ is associated with attributes bounded by $\text{poly}(|Q|)$.

The last $|Q|$ rows pertaining to attributes "String length $= i$", $i \in [|Q|]$ is a $|Q| \times \mathcal{C}$ submatrix containing all zeros except specific locations filled with 1s in a diagonal form as shown. We prove the following theorem.

Theorem 11. *Let $\mathbf{L}_{M,\mathbf{w}}$ be the submatrix of $\mathbf{L}_M$ restricted to the rows selected by attribute set $S_\mathbf{w}$ (please see Definition 2.1). Then, for any DFA $M = (Q, \Sigma, T, q_{\mathsf{st}}, F) \in B^{\mathsf{DFA}}$ and any input $\mathbf{w} \in A^{\mathsf{DFA}}$ we have $\mathbf{e}_1 \in \text{span}(\mathbf{L}_{M,\mathbf{w}}^\top)$ iff $(M(\mathbf{w}) = 1 \wedge |\mathbf{w}| \leq |Q|)$.*

Proof. We first prove "if" direction. For any $\mathbf{w} \in A^{\mathsf{DFA}}$ with $|\mathbf{w}| = \ell \leq |Q|$, the submatrix $\mathbf{L}_{M,\mathbf{w}}$ of $\mathbf{L}_M$ restricted by $S_\mathbf{w}$ is shown in Table 3.

Since M is a DFA, the matrix $\mathbf{Y}^{(b)}$ will always have exactly one "-1" in each of its rows. Let $\mathbf{w} = (w_1, \ldots, w_\ell)$. To prove the theorem, we give an algorithm which

“Dummy” $\mapsto$	1	-10 $\ldots$ 0	0 $\ldots$ 0	0 $\ldots$ 0	$\ldots$	0 $\ldots$ 0	0 $\ldots$ 0
“$x_1 = w_1$” $\mapsto$	$\mathbf{0}_Q$	$\mathbf{I}_Q$	$\mathbf{Y}^{(w_1)}$				
“$x_2 = w_2$” $\mapsto$	$\mathbf{0}_Q$		$\mathbf{I}_Q$	$\mathbf{Y}^{(w_2)}$			
$\vdots$	$\vdots$				$\ddots$		
“$x_\ell = w_\ell$” $\mapsto$	$\mathbf{0}_Q$					$\mathbf{I}_Q$	$\mathbf{Y}^{(w_\ell)}$
“String length $= \ell$” $\mapsto$	0					0 $\ldots$ 0	0 $\ldots$ 01

Table 3. Submatrix $\mathbf{L}_{M,\mathbf{w}}$ defined by $S_\mathbf{w}$ and $\mathbf{L}_M$

constructs a subset of rows $\widehat{\mathbf{L}}_{M,\mathbf{w}}$ of $\mathbf{L}_{M,\mathbf{w}}$ inductively that sums up to $\mathbf{e}_1$ iff $M(\mathbf{w}) = 1$. The algorithm proceeds as follows:

On input $(M, \mathbf{w}, \mathbf{L}_{M,\mathbf{w}})$, it does the following:

1. Initialize $\widehat{\mathbf{L}}_{M,\mathbf{w}}$ with the first row of $\mathbf{L}_{M,\mathbf{w}}$ labelled with attribute “Dummy”.
2. For $i \in [\ell]$, do the following:
   (a) If $i = 1$, populate $\widehat{\mathbf{L}}_{M,\mathbf{w}}$ with *second* row of $\mathbf{L}_{M,\mathbf{w}}$ labelled with “$x_1 = w_1$”. Discard the remaining $|Q| - 1$ rows in the block labelled with “$x_1 = w_1$”.

   For the chosen row, let $k_1 \in Q$ be such that $T(1, w_1) = k_1$. By construction this implies $y^{(w_1)}_{1,k_1} = -1$ in $\mathbf{Y}^{(w_1)}$.

   (b) If $i \in [2, \ell]$, choose the k_{i-1}-th row in the block labelled with “$x_i = w_i$” and add it to $\widehat{\mathbf{L}}_{M,\mathbf{w}}$. Discard the remaining $|Q| - 1$ rows in the block labelled with “$x_i = w_i$”.
   For the chosen row, let $k_i \in Q$ be such that $T(k_{i-1}, w_i) = k_i$. By construction this implies $y^{(w_i)}_{k_{i-1},k_i} = -1$ in $\mathbf{Y}^{(w_i)}$.
3. Add the row labelled “String length $= \ell$” to $\widehat{\mathbf{L}}_{M,\mathbf{w}}$. Output $\widehat{\mathbf{L}}_{M,\mathbf{w}}$ and terminate.

It is easy to see that the above algorithm always terminates. The first two rows of $\mathbf{L}_{M,\mathbf{w}}$ labelled with attributes “Dummy” and “$x_1 = w_1$” are chosen in Step 1 and Step $2(a)$ of the above algorithm respectively. The last row is chosen in a natural way in Step 3 based on the length of the input string.

Aside from these, note that the way the remaining rows are added to $\widehat{\mathbf{L}}_{M,\mathbf{w}}$ is governed by the transition function T of the DFA M. Essentially, the computation of $\widehat{\mathbf{L}}_{M,\mathbf{w}}$ mirrors the computation of M on input $\mathbf{w}$. In particular, the *order* in which the rows are selected iteratively in Step 2 always follow a loop invariant: at the end of the i-th iteration the chosen rows sum to a vector $\mathbf{v}_i = (1, 0, \ldots, 0, -1, 0, \ldots, 0)$, where -1 appears exactly at the k_i-th position associated with the $|Q| \times |Q|$-sized block matrix $\mathbf{Y}^{(w_i)}$. Hence, when $M(\mathbf{w}) = 1$ with $|\mathbf{w}| = \ell$, the vectors in $\widehat{\mathbf{L}}_{M,\mathbf{w}}$ at the end of the Step 2 sum to $\mathbf{v}_\ell = (1, 0, \ldots, 0, -1)$. Here -1 is at position $|Q|$ associated with $\mathbf{Y}^{(w_\ell)}$ and is also the final state of M. By construction of $\mathbf{L}_{M,\mathbf{w}}$, it follows that the last row selected in Step 3 labelled with "$\mathsf{String\ length} = \ell$" when added to $\mathbf{v}_\ell$ results to $\mathbf{e}_1$, as intended.

We then prove "only if" direction. For any $\mathbf{w} = (w_1, \ldots, w_\ell) \in \Sigma^\ell$ such that $M(\mathbf{w}) \neq 1$ and $\ell \leq |Q|$, note that the description of $\mathbf{L}_{M,\mathbf{w}}$ forces the first two rows corresponding to attributes "Dummy" and "$x_1 = w_1$" to be chosen to build $\mathbf{e}_1$ progressively. For $i \in [2, \ell - 1]$, let $k_{i-1}, k_i \in Q$ be such that $y^{(w_i)}_{k_{i-1},k_i} = -1$ in $\mathbf{Y}^{(w_i)}$. Consequently, the only choice left for selecting the next row further to nullify the -1 in $y^{(w_i)}_{k_{i-1},k_i}$ is restricted to the k_i-th row in the block labelled with "$x_{i+1} = w_{i+1}$" which again forces the emulation of M's computation on input $\mathbf{w}$. Since $M(\mathbf{w}) \neq 1$, the sum of all the rows at the end of the ℓ-th iteration cannot have a "-1" in its $|Q|^{th}$ position. When added to the row labelled "$\mathsf{String\ length} = \ell$", this does not yield $\mathbf{e}_1$ as desired.

We then consider $\mathbf{w} = (w_1, \ldots, w_\ell) \in \Sigma^\ell$ such that $\ell > |Q|$. In this case, the matrix $\mathbf{L}_{M,\mathbf{w}}$ does not have the last row in Table 3. Therefore, we cannot nullify "-1" that appears in the rightmost block as a result of enforced emulation of M's computation. Therefore, we cannot obtain $\mathbf{e}_1$ as desired.

4.2 Encoding DFA Input Strings to Monotone Span Programs

In this case the DFA machine M is encoded into a set of attributes S_M from an appropriately defined attribute universe while the input string $\mathbf{x} \in \Sigma^*$ will be encoded to a MSP $(\mathbf{L}_\mathbf{x}, \rho_\mathbf{x})$.

We construct two efficiently computable functions:

1. $f^{\mathsf{CP}}_{\mathsf{e}} : A^{\mathsf{DFA}} \to A^{\mathsf{MUCP}}$ to encode $\mathbf{x} \in A^{\mathsf{DFA}}$ into a MSP $(\mathbf{L}_\mathbf{x}, \rho_\mathbf{x}) \in A^{\mathsf{MUCP}}$.
2. $f^{\mathsf{CP}}_{\mathsf{k}} : B^{\mathsf{DFA}} \to B^{\mathsf{MUCP}}$ to encode $M \in B^{\mathsf{DFA}}$ as a set of attributes $S_M \in B^{\mathsf{MUCP}}$.

We argue that $R^{\mathsf{MUCP}}(S_M, (\mathbf{L}_\mathbf{x}, \rho_\mathbf{x})) = 1$ iff $R^{\mathsf{DFA}>}(\mathbf{x}, M) = 1$, where $S_M = f^{\mathsf{CP}}_{\mathsf{k}}(M)$ and $(\mathbf{L}_\mathbf{x}, \rho_\mathbf{x}) = f^{\mathsf{CP}}_{\mathsf{e}}(\mathbf{x})$.

For ease of exposition, we represent the universe of attributes as follows:

$$B^{\mathsf{MUCP}} := \{(b, i, j) \mid b \in \{0, 1\}, i, j \in [2^\lambda]\} \cup \{\text{“}\mathsf{Size} = \mathsf{s}\text{”} \mid \mathsf{s} \in [2^\lambda]\} \cup \{\text{“}\mathsf{Dummy}\text{”}\}.$$

We assume that these attributes are embedded into $\mathbb{Z}$ via an injective mapping such as

$$\text{“}\mathsf{Dummy}\text{”} \mapsto 0, \quad \text{“}(b, i, j)\text{”} \mapsto 4((i + j)^2 + j) + 2b \quad \text{“}\mathsf{Size} = \mathsf{s}\text{”} \mapsto 2\mathsf{s} + 1,$$

But for maintaining intuitive notation, we make the mapping implicit.

A DFA $M = (Q, \Sigma, T, q_{\mathsf{st}}, F) \in B^{\mathsf{DFA}}$ is encoded as a set of attributes given by $f^{\mathsf{CP}}_{\mathsf{k}}(M) = S_M \in B^{\mathsf{MUCP}}$ as:

$$S_M := \{\text{“}\mathsf{Dummy}\text{”}\} \cup \{(b, i, j) \in \Sigma \times Q^2 \mid T(i, b) = j\} \cup \{\text{“}\mathsf{Size} = |Q|\text{”}\}.$$

When we represent S_M as a set of integers, we have $S_M \subseteq [20|Q|^2]$ and thus in particular, all the values in S_M are bounded by $\mathrm{poly}(|Q|)$.

An input string $\mathbf{x} = (x_1, \dots, x_\ell) \in A^{\mathsf{DFA}}$ of length ℓ is encoded into a MSP given by $f_{\mathsf{e}}^{\mathsf{CP}}(\mathbf{x}) = (\mathbf{L}_\mathbf{x}, \rho_\mathbf{x}) \in A^{\mathsf{MUCP}}$. Here $\mathbf{L}_\mathbf{x} \in \{0, \pm 1\}^{\mathcal{R} \times \mathcal{C}}$ with $\mathcal{R} = 1 + \ell^3 + \ell$ and $\mathcal{C} = 1 + \ell + \ell^2$. The label map $\rho_\mathbf{x}$ will be implicit in the description of the matrix $\mathbf{L}_\mathbf{x}$. Before providing the construction of $\mathbf{L}_\mathbf{x}$, we define the following sub-matrices useful in the construction:

– matrix $\mathbf{I}_\ell$ denoting the $\ell \times \ell$ identity matrix and a column-vector $\mathbf{g}_\ell = \underbrace{(1, \dots, 1)}_{\ell}^\top$

– matrices $\mathbf{S}_\ell$ and $\mathbf{T}_\ell$ such that

$$\mathbf{S}_\ell := \mathbf{I}_\ell \otimes \mathbf{g}_\ell = \begin{bmatrix} \mathbf{g}_\ell & \mathbf{0}_\ell & \dots & \mathbf{0}_\ell \\ \mathbf{0}_\ell & \mathbf{g}_\ell & \dots & \mathbf{0}_\ell \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_\ell & \mathbf{0}_\ell & \dots & \mathbf{g}_\ell \end{bmatrix}_{\ell^2 \times \ell} \text{, where } \mathbf{0}_\ell \text{ is the all-zero column-vector of size } \ell$$

and $\mathbf{T}_\ell = -\mathbf{g}_\ell \otimes \mathbf{I}_\ell = [-\mathbf{I}_\ell \| \dots \| - \mathbf{I}_\ell]^\top$ of size $\ell^2 \times \ell$.

For a fixed $b \in \{0, 1\}$, we say "*associate* $[\mathbf{S}_\ell \| \mathbf{T}_\ell]$ *with* b"[4] when we label the rows of $[\mathbf{S}_\ell \| \mathbf{T}_\ell]$ as shown in Table 4.

$$\begin{array}{r|cccc|c|}
(b,1,1) \mapsto & & & & & \\
\vdots & \mathbf{g}_\ell & \mathbf{0}_\ell & \dots & \mathbf{0}_\ell & -\mathbf{I}_\ell \\
(b,1,\ell) \mapsto & & & & & \\
(b,2,\ell) \mapsto & & & & & \\
\vdots & \mathbf{0}_\ell & \mathbf{g}_\ell & \dots & \mathbf{0}_\ell & -\mathbf{I}_\ell \\
(b,2,\ell) \mapsto & & & & & \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
(b,\ell,1) \mapsto & & & & & \\
\vdots & \mathbf{0}_\ell & \mathbf{0}_\ell & \dots & \mathbf{g}_\ell & -\mathbf{I}_\ell \\
(b,\ell,\ell) \mapsto & & & & &
\end{array}$$

Table 4. Submatrix $[\mathbf{S}_\ell \| \mathbf{T}_\ell]$ with its row label map

We also denote $\mathbf{0}_{\ell^2}$, $\mathbf{0}_{\ell^2 \times \ell}$ and $\mathbf{0}_{\ell \times \ell}$ to be all-zero column-vector of size ℓ^2 and all-zero matrices of size $\ell^2 \times \ell$ and $\ell \times \ell$ respectively. We now define $\mathbf{L}_\mathbf{x}$ with its rows labelled with attributes as specified in Table 5.

We observe that we have $\max_i \rho_\mathbf{x}(i) \leq 20\ell^2$, where we regard the attributes as integers through the aforementioned injective mapping. In particular, $\mathbf{L}_\mathbf{x}$ is associated with attributes bounded by $\mathrm{poly}(\ell)$.

The last ℓ rows pertaining to attributes "$\mathsf{Size} = i$", $i \in [\ell]$ is a $\ell \times \mathcal{C}$ submatrix containing all zeros except an identity matrix block $\mathbf{I}_\ell$ located under the rightmost $\mathbf{T}_\ell$ with its i-th row labelled with attribute "$\mathsf{Size} = i$", $\forall i \in [\ell]$. We show the following.

[4] For brevity, we express this as $b \Leftrightarrow [\mathbf{S}_\ell \| \mathbf{T}_\ell]$ in the final description of $\mathbf{L}_\mathbf{x}$.

"Dummy" $\mapsto$	1	-10 . . . 0	0 . . . 0	0 . . . 0	. . .	0 . . . 0	0 . . . 0
$x_1 \Leftrightarrow$	$\mathbf{0}_{\ell^2}$	$\mathbf{S}_\ell$	$\mathbf{T}_\ell$	$\mathbf{0}_{\ell^2\times\ell}$	. . .	$\mathbf{0}_{\ell^2\times\ell}$	$\mathbf{0}_{\ell^2\times\ell}$
$x_2 \Leftrightarrow$	$\mathbf{0}_{\ell^2}$	$\mathbf{0}_{\ell^2\times\ell}$	$\mathbf{S}_\ell$	$\mathbf{T}_\ell$	. . .	$\mathbf{0}_{\ell^2\times\ell}$	$\mathbf{0}_{\ell^2\times\ell}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$x_\ell \Leftrightarrow$	$\mathbf{0}_{\ell^2}$	$\mathbf{0}_{\ell^2\times\ell}$	$\mathbf{0}_{\ell^2\times\ell}$	$\mathbf{0}_{\ell^2\times\ell}$	. . .	$\mathbf{S}_\ell$	$\mathbf{T}_\ell$
"Size $= 1$" $\mapsto$	0	$\mathbf{0}_{\ell\times\ell}$	$\mathbf{0}_{\ell\times\ell}$	$\mathbf{0}_{\ell\times\ell}$	. . .	$\mathbf{0}_{\ell\times\ell}$	$\mathbf{I}_\ell$
$\vdots$	$\vdots$						
"Size $= \ell$" $\mapsto$	0						

Table 5. Encoding a string $\mathbf{x}$ to matrix $\mathbf{L}_\mathbf{x}$

Theorem 12. *Let $\mathbf{L}_{M,\mathbf{x}}$ be the submatrix of $\mathbf{L}_\mathbf{x}$ restricted to the rows selected by the set S_M (please see Definition 2.1). Then, for any DFA $M = (Q, \Sigma, T, q_{\mathsf{st}}, F) \in B^{\mathsf{DFA}}$ and any input $\mathbf{x} \in A^{\mathsf{DFA}}$ we have $\mathbf{e}_1 \in \mathrm{span}(\mathbf{L}_{M,\mathbf{x}}^\top)$ iff $\big(M(\mathbf{x}) = 1 \wedge |\mathbf{x}| \geq |Q|\big)$.*

Proof. We first remove all the all-zero columns from $\mathbf{L}_{M,\mathbf{x}}$ and call the remaining matrix as $\mathbf{L}_{M,\mathbf{x}}$ w.l.o.g. since these columns do not influence on whether $\mathbf{e}_1 \in \mathrm{span}(\mathbf{L}_{M,\mathbf{x}}^\top)$ or not. This simplification ensures that $\mathbf{L}_{M,\mathbf{x}}$ is given as shown in Table 6. Note that the rows present in $\mathbf{L}_{M,\mathbf{x}}$ is governed by the transition function, T of M (via the row labels in $\mathbf{L}_\mathbf{x}$). We also note that the last row in Table 6 will be missing if we have $|\mathbf{x}| < |Q|$. Therefore, the matrix $\mathbf{Y}^{(b)}$ here is the same as that was defined in Section 4.1. Hence,

"Dummy" $\mapsto$	1	-10 . . . 0	0 . . . 0	0 . . . 0	. . .	0 . . . 0	0 . . . 0
$x_1 \Leftrightarrow$	$\mathbf{0}_Q$	$\mathbf{I}_Q$	$\mathbf{Y}^{(x_1)}$				
$x_2 \Leftrightarrow$	$\mathbf{0}_Q$		$\mathbf{I}_Q$	$\mathbf{Y}^{(x_2)}$			
$\vdots$	$\vdots$				$\ddots$		
$x_\ell \Leftrightarrow$	$\mathbf{0}_Q$					$\mathbf{I}_Q$	$\mathbf{Y}^{(x_\ell)}$
"Size $= \vert Q\vert$" $\mapsto$	0					0 . . . 0	0 . . . 01

Table 6. Submatrix $\mathbf{L}_{M,\mathbf{x}}$ defined by S_M and $\mathbf{L}_\mathbf{x}$

the proof follows identically to that of Theorem 11.

5 Putting it all together: ABE for DFA

In this section, we discuss instantiation of our generic construction of ABE for DFA by putting together all the ingredients developed so far.

As we have seen in Sec. 3.1, ABE for R^{DFA} (i.e., ABE for DFA) can be constructed from ABE for $R^{\mathsf{DFA}\geq}$ and ABE for $R^{\mathsf{DFA}\leq}$. Furthermore, as we have seen in Theorem 10 (resp., Theorem 9), ABE for $R^{\mathsf{DFA}>}$ (resp., ABE for $R^{\mathsf{DFA}\leq}$) is implied by ABE for R^{MUCP} (resp., R^{MUKP}).

To instantiate the ABE for R^{MUKP}, we use the construction in the full version of our paper [6] (Section 5.2). As was shown in [6] (in Theorem 13), this construction is semi-adaptively secure under the MDDH_k assumption. To instantiate the ABE for R^{MUCP}, we use the construction in the full version of our paper [6] (Section 5.4). As was shown in [6] (in Theorem 14), this construction satisfies selective* security under the DLIN assumption. Putting all pieces together, we obtain the following theorem.

Theorem 13. *There exists selective* secure key-policy ABE for R^{DFA} from the* DLIN *assumption.*

Ciphertext Policy ABE for DFA. We observe that our construction dfaABE uses the underlying kpABE and cpABE in a symmetric way. Thus, by swapping the use of kpABE and cpABE in our construction, we can equivalently construct ciphertext-policy ABE for DFA. Recall that analogous to ABE for MSP (Section 2), the ciphertext-policy variant of ABE for DFA is defined simply by swapping the order of the domains in the relation R^{DFA}. In more detail, we set $A^{\mathsf{CPDFA}} = B^{\mathsf{DFA}}$ and $B^{\mathsf{CPDFA}} = A^{\mathsf{DFA}}$ and define the relation R^{CPDFA} analogously for a ciphertext policy scheme for DFA. Thus, in a ciphertext-policy scheme, the encryptor to encrypt a machine and the key generator to compute a key for an input $\mathbf{x}$.

To modify dfaABE to be ciphertext-policy, we exchange the maps used by KeyGen and Enc in the constructions of $\mathsf{dfaABE}^{\leq}$ and $\mathsf{dfaABE}^{>}$ in Sections 3.2 and 3.3 respectively. For instance, to construct a ciphertext-policy variant of $\mathsf{dfaABE}^{\leq}$, we modify the encrypt and key generation algorithms so that:

1. The key generation algorithm receives as input an attribute $\mathbf{x}$, converts it to attributes $S_{\mathbf{x}}$ using the map defined in Section 4.1 and computes cpABE key for $S_{\mathbf{x}}$.
2. The encryption algorithm receives as input an MSP M, converts it to an MSP $(\mathbf{L}_M, \rho_M)$ using the map defined in Section 4.1 and computes cpABE encryption for policy $(\mathbf{L}_M, \rho_M)$.

The modification to $\mathsf{dfaABE}^{>}$ is analogous. The compiler dfaABE remains the same.

Thus, we additionally obtain the following theorem:

Theorem 14. *There exists selective* secure ciphertext-policy ABE for R^{DFA} from the* DLIN *assumption.*

References

1. Agrawal, S., Chase, M.: A study of pair encodings: Predicate encryption in prime order groups. In: TCC 2016-A, Part II. pp. 259–288 (2016)

2. Agrawal, S., Chase, M.: Fame: Fast attribute-based message encryption. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. CCS '17 (2017)
3. Agrawal, S., Freeman, D.M., Vaikuntanathan, V.: Functional encryption for inner product predicates from learning with errors. In: Asiacrypt (2011)
4. Agrawal, S., Maitra, M.: Fe and io for turing machines from minimal assumptions. In: TCC (2018)
5. Agrawal, S., Maitra, M., Yamada, S.: Attribute based encryption (and more) for nondeterministic finite automata from learning with errors. In: Crypto (2019)
6. Agrawal, S., Maitra, M., Yamada, S.: Attribute based encryption for deterministic finite automata from dlin. Cryptology ePrint Archive, Report 2019/645 (2019), https://eprint.iacr.org/2019/645
7. Agrawal, S., Singh, I.P.: Reusable garbled deterministic finite automata from learning with errors. In: ICALP. vol. 80. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
8. Ananth, P., Fan, X.: Attribute based encryption with sublinear decryption from lwe. Cryptology ePrint Archive, Report 2018/273 (2018), https://eprint.iacr.org/2018/273
9. Ananth, P., Sahai, A.: Functional encryption for turing machines. In: Kushilevitz, E., Malkin, T. (eds.) Theory of Cryptography (2016)
10. Ananth, P., Sahai, A.: Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In: EUROCRYPT (2017)
11. Apon, D., Döttling, N., Garg, S., Mukherjee, P.: Cryptanalysis of indistinguishability obfuscations of circuits over ggh13. eprint 2016 (2016)
12. Attrapadung, N.: Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In: EUROCRYPT. pp. 557–577 (2014)
13. Attrapadung, N.: Dual system encryption framework in prime-order groups via computational pair encodings. In: Proceedings, Part II, of the 22Nd International Conference on Advances in Cryptology — ASIACRYPT 2016 - Volume 10032 (2016)
14. Attrapadung, N., Hanaoka, G., Yamada, S.: Conversions among several classes of predicate encryption and applications to abe with various compactness tradeoffs. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 575–601. Springer (2015)
15. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. In: CRYPTO (2001)
16. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy. pp. 321–334 (2007)
17. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: EUROCRYPT. pp. 533–556 (2014)
18. Boneh, D., Hamburg, M.: Generalized identity based and broadcast encryption schemes. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 455–470. Springer (2008)
19. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: TCC. pp. 535–554 (2007)
20. Boyen, X., Li, Q.: Attribute-based encryption for finite automata from lwe. In: ProvSec (2015)
21. Brakerski, Z., Vaikuntanathan, V.: Circuit-abe from lwe: Unbounded attributes and semi-adaptive security. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016 (2016)
22. Chen, J., Gay, R., Wee, H.: Improved dual system abe in prime-order groups via predicate encodings. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 (2015)

23. Chen, J., Gong, J., Kowalczyk, L., Wee, H.: Unbounded abe via bilinear entropy expansion, revisited. In: EUROCRYPT (1). pp. 503–534 (2018)
24. CHEN, J., Wee, H.: Fully, (almost) tightly secure ibe and dual system groups. In: CRYPTO (2013)
25. Chen, J., Wee, H.: Semi-adaptive attribute-based encryption and improved delegation for boolean formula. In: Security and Cryptography for Networks (2014)
26. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: Proc. of EUROCRYPT. LNCS, vol. 9056, pp. 3–12. Springer (2015)
27. Cheon, J.H., Fouque, P.A., Lee, C., Minaud, B., Ryu, H.: Cryptanalysis of the new clt multilinear map over the integers. Eprint 2016/135
28. Cheon, J.H., Jeong, J., Lee, C.: An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low level encoding of zero. Eprint 2016/139
29. Coron, J.S., Gentry, C., Halevi, S., Lepoint, T., Maji, H.K., Miles, E., Raykova, M., Sahai, A., Tibouchi, M.: Zeroizing without low-level zeroes: New mmap attacks and their limitations. In: Advances in Cryptology–CRYPTO 2015, pp. 247–266. Springer (2015)
30. Coron, J.S., Lee, M.S., Lepoint, T., Tibouchi, M.: Zeroizing attacks on indistinguishability obfuscation over clt13. Eprint 2016 (2016)
31. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: EUROCRYPT (2013)
32. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013), http://eprint.iacr.org/
33. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: STOC (2013)
34. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run turing machines on encrypted data. In: CRYPTO (2). pp. 536–553 (2013)
35. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: STOC. pp. 555–564 (2013)
36. Gong, J., Waters, B., Wee, H.: Abe for dfa from k-lin. In: Annual International Cryptology Conference. pp. 732–764. Springer (2019)
37. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute based encryption for circuits. In: STOC (2013)
38. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from lwe. In: Crypto (2015)
39. Gorbunov, S., Vinayagamurthy, D.: Riding on asymmetry: Efficient abe for branching programs. In: Proceedings, Part I, of the 21st International Conference on Advances in Cryptology – ASIACRYPT 2015 - Volume 9452 (2015)
40. Goyal, R., Koppula, V., Waters, B.: Semi-adaptive security and bundling functionalities made generic and easy. In: TCC (2016)
41. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security. pp. 89–98 (2006)
42. Hu, Y., Jia, H.: Cryptanalysis of GGH map. Cryptology ePrint Archive: Report 2015/301 (2015)
43. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: EUROCRYPT. pp. 146–162 (2008)
44. Kitagawa, F., Nishimaki, R., Tanaka, K., Yamakawa, T.: Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. Cryptology ePrint Archive, Report 2018/974 (2018), https://eprint.iacr.org/2018/974
45. Kowalczyk, L., Lewko, A.B.: Bilinear entropy expansion from the decisional linear assumption. In: CRYPTO (2015)

46. Kowalczyk, L., Wee, H.: Compact adaptively secure abe for nc1 from k-lin. In: EUROCRYPT, Part I. pp. 3–33 (2019)
47. Lewko, A.: Tools for simulating features of composite order bilinear groups in the prime order setting. In: Proceedings of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques. EUROCRYPT'12 (2012)
48. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure hibe with short ciphertexts. In: Theory of Cryptography - 7th Theory of Cryptography Conference, TCC 2010, Proceedings (2010)
49. Lewko, A., Waters, B.: Unbounded hibe and attribute-based encryption. In: Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology. EUROCRYPT'11 (2011)
50. Lewko, A.B., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: EUROCRYPT (2010)
51. Lewko, A.B., Waters, B.: New proof methods for attribute-based encryption: Achieving full security through selective techniques. In: CRYPTO. pp. 180–198 (2012)
52. Miles, E., Sahai, A., Zhandry, M.: Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13. In: Crypto (2016)
53. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Proceedings of the 30th Annual Conference on Advances in Cryptology. CRYPTO'10 (2010)
54. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012 (2012)
55. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security. CCS '13 (2013)
56. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: EUROCRYPT. pp. 457–473 (2005)
57. Waters, B.: Functional encryption for regular languages. In: Crypto (2012)
58. Wee, H.: Dual system encryption via predicate encodings. In: TCC (2014)