Attribute Based Encryption (and more) for Nondeterministic Finite Automata from LWE

Shweta Agrawal[1], Monosij Maitra[1], and Shota Yamada[2](✉)

[1] IIT Madras, Chennai, India
{shweta.a,monosij}@cse.iitm.ac.in
[2] National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
yamada-shota@aist.go.jp

Abstract. Constructing Attribute Based Encryption (ABE) [56] for uniform models of computation from standard assumptions, is an important problem, about which very little is known. The *only* known ABE schemes in this setting that (i) avoid reliance on multilinear maps or indistinguishability obfuscation, (ii) support *unbounded length inputs* and (iii) permit *unbounded key requests* to the adversary in the security game, are by Waters from *Crypto, 2012* [57] and its variants. Waters provided the first ABE for Deterministic Finite Automata (DFA) satisfying the above properties, from a parametrized or "q-type" assumption over bilinear maps. Generalizing this construction to Nondeterministic Finite Automata (NFA) was left as an explicit open problem in the same work, and has seen no progress to date. Constructions from other assumptions such as more standard pairing based assumptions, or lattice based assumptions has also proved elusive.

In this work, we construct the first symmetric key attribute based encryption scheme for nondeterministic finite automata (NFA) from the learning with errors (LWE) assumption. Our scheme supports unbounded length inputs as well as unbounded length machines. In more detail, secret keys in our construction are associated with an NFA 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 $M(\mathbf{x}) = 1$.

Further, we leverage our ABE to achieve (restricted notions of) attribute hiding analogous to the circuit setting, obtaining the first *predicate encryption* and bounded key *functional encryption* schemes for NFA from LWE. We achieve machine hiding in the single/bounded key setting to obtain the first *reusable garbled NFA* from standard assumptions. In terms of lower bounds, we show that secret key *functional encryption* even for DFAs, with security against unbounded key requests implies indistinguishability obfuscation (iO) for circuits; this suggests a barrier in achieving full fledged functional encryption for NFA.

A. Boldyreva and D. Micciancio (Eds.): CRYPTO 2019, LNCS 11693, pp. 765–797, 2019.
https://doi.org/10.1007/978-3-030-26951-7_26

1 Introduction

Attribute based encryption (ABE) [56] is an emerging 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 Boolean function f. Decryption succeeds to yield the hidden message m if and only if the attribute satisfies the function, namely $f(\mathbf{x}) = 1$. Starting with the seminal work of Sahai and Waters [56], ABE schemes have received a lot of attention in recent years [4,10,20,22,23,26,39–41, 43,45,49,57], yielding constructions for various classes of functions under diverse assumptions.

In most constructions, the function f embedded in the key is represented as a circuit. While powerful, circuits are a *non-uniform* model of computation which necessitates different representations for different input lengths, forcing the scheme to provide multiple function keys for the same functionality as the input length varies. This drawback poses a significant deployment barrier in many practical application scenarios, since data sizes in the real world are rarely of fixed length[1]. Attribute based encryption for uniform models of computation has also been studied, but so far, we have very few constructions from standard assumptions. Waters [57] provided a construction of ABE for Deterministic Finite Automata (DFA) from parametrized or "q-type" assumptions over bilinear maps. Generalizing this construction to Nondeterministic Finite Automata (NFA) was left as an explicit open problem[2] in [57], and has remained open to date. Constructions from other assumptions such as more standard pairing based assumptions, or lattice based assumptions has also proved elusive. Boyen and Li [24] provided a construction of ABE for DFA from the Learning With Errors (LWE) assumption but this was restricted to DFAs with *bounded* length inputs, rendering moot the primary advantage of a DFA over circuits. Agrawal and Singh [8] constructed a primitive closely related to ABE for DFA, namely *reusable garbled DFA* from LWE, but their construction is limited to a security game where the adversary may only request a single function key.

From strong assumptions such as the existence of multilinear maps [33], witness encryption [36] or indistinguishability obfuscation [18,34], attribute based encryption (indeed, even its more powerful generalization – *functional encryption*) has been constructed even for Turing machines [6,14,48], but these are not considered standard assumptions; indeed many candidate constructions have been broken [15,27–29,31,32,44,55]. Very recently, Ananth and Fan [10] constructed ABE for RAM programs from LWE achieving decryption complexity that is sublinear in the database length. However, the key sizes in their

[1] A trivial workaround would be to fix the input length to some fixed upper bound and pad all data to this bound; but this solution incurs substantial overhead (besides being inelegant).

[2] Note that an NFA can be converted to an equivalent DFA but this transformation incurs exponential blowup in machine size.

construction are massive and grow with the size of the entire database as well as with worst case running time of the program on any input. In particular, restricting the construction to any model of computation that reads the entire input string (e.g. DFA, TM) yields a bounded input solution, since the key size depends on the input length. Similarly, [26,42] construct attribute based encryption for "bundling functionalities" where the size of the public parameters does not depend on the size of the input chosen by the encryptor, say ℓ. 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; this was noted explicitly in [42].

Our Results. In this work, we construct the first symmetric key attribute based encryption scheme for nondeterministic finite automata (NFA) from the learning with errors (LWE) assumption. Our scheme supports unbounded length inputs as well as unbounded length machines. In more detail, secret keys in our construction are associated with an NFA 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 $M(\mathbf{x}) = 1$. Moreover our construction achieves succinct parameters, namely, the length of the function key and ciphertext grow only with the machine size and input length respectively (and do not depend on the input length and machine size respectively).

Further, we leverage our ABE to achieve (restricted notions of) attribute hiding analogous to the circuit setting, obtaining the first *predicate encryption* and bounded key *functional encryption* schemes for NFA. We achieve machine hiding in the single key[3] setting to obtain the first *reusable garbled NFA* from standard assumptions. This improves upon the result of [8], which can only support a *single* key request (as against bounded), and only DFAs (as against NFAs).

The above results raise the question of whether full fledged functional encryption, which achieves full attribute hiding for NFAs is possible under standard assumptions. However, we show that secret key functional encryption even for DFA with security against unbounded key requests implies indistinguishability obfuscation (iO) for circuits. Since constructing iO for circuits from standard assumptions is a central challenge in cryptography, this suggests that there is a barrier in further generalizing our result to achieve full attribute hiding.

We summarize our results in Table 1.

[3] This may be generalized to bounded key, for any a-priori fixed (polynomial) bound.

Table 1. Prior work and our results. Above, we say that input length supported by a construction is bounded if the parameters and key lengths depend on the input size. For attribute hiding, yes* indicates hiding in the restricted security games of predicate or bounded key functional encryption.

Construction	Model	Input Length	Number of Keys	Attribute and Function Hiding	Assumption
Waters [57]	DFA	unbounded	unbounded	(no, no)	q-type assumption on bilinear maps
Boyen-Li [24]	DFA	bounded	unbounded	(no, no)	LWE
Agrawal-Singh [8]	DFA	unbounded	single	(yes, yes)	LWE
Ananth-Fan [10]	RAM	bounded	unbounded	(no, no)	LWE
Section 4	NFA	unbounded	unbounded	(no, no)	LWE
Full version	NFA	unbounded	unbounded	(yes*, no)	LWE
Full version	NFA	unbounded	bounded	(yes, yes)	LWE

1.1 Our Techniques

In this section, we provide an overview of our techniques. Before we proceed, we discuss the technical barriers that arise in following the approaches taken by prior work. Since the construction by Waters [57] is the only one that supports unbounded attribute lengths and unbounded key requests by the adversary, [4] it is the most promising candidate for generalization to NFA. However, the challenges in generalizing this construction to support NFAs were explicitly discussed in the same work, and this has seen no progress in the last seven years to the best of our knowledge, despite the significant research attention ABE schemes have received. Moreover, even the solution for DFAs is not fully satisfactory since it relies on a non-standard parametrized or "q-type" assumption.

Boyen and Li [24] attempt to construct ABE for DFAs from the LWE assumption, but their construction crucially requires the key generator to know the length of the attribute chosen by the encryptor, since it must provide a fresh "trapdoor" for each row of the DFA transition table and each input position. Indeed, reusing the same trapdoor for multiple positions in the input leads to trivial "mix and match" attacks against their scheme. Thus, it is not even clear how to obtain ABE for DFA with support for unbounded lengths by following this route. The work of Agrawal and Singh [8] gives a construction of functional encryption for DFA from LWE that does handle unbounded length inputs, but

[4] The construction is later extended to be adaptively secure rather than selectively secure (e.g., [16]), but the basic structure of the construction is unchanged.

only in the limited single key setting. Their construction crucially relies on reusable garbled circuits [37] which is a single key primitive, and natural attempts to generalize their construction to support even two keys fails[5]. Similarly, the very recent construction of Ananth and Fan [10] is also inherently bounded length, for reasons similar as those discussed above for [24].

Thus, the handful of existing results in this domain all appear to pose fundamental barriers to generalization. To overcome this hurdle, we design completely new techniques to handle the challenge of unbounded length; these may be applicable elsewhere. We focus on the symmetric key setting, and proceed in two steps: i) we provide a secret key ABE scheme for NFA that supports unbounded length inputs but only supports bounded size NFA machines, and ii) we "bootstrap" the construction of step (i) to handle unbounded length machines. We additionally achieve various notions of attribute hiding as discussed above, but will focus on the ABE construction for the remainder of this overview. We proceed to describe each of these steps in detail.

Constructing NfaABE *for Bounded Size NFA.* Our first goal is to construct a secret key ABE scheme for NFA that supports unbounded length inputs but only supports bounded size NFA machines from the LWE assumption. Since ABE for circuits has received much success from the LWE assumption [22,39], our first idea is to see if we can run many circuit ABE schemes "in parallel", one for each input length. We refer to our resulting ABE scheme for NFAs as NfaABE and the ABE for circuits scheme simply as ABE, in order to differentiate them.

Naïve Approach: We start with the following naïve construction that uses a (public key) ABE for circuits as an ingredient. The master secret key of the NfaABE scheme is a PRF key K. This PRF key defines a set of key pairs $\{(\mathsf{ABE.mpk}_j, \mathsf{ABE.msk}_j)\}_{j\in[2^\lambda]}$ of the ABE scheme, where each $(\mathsf{ABE.mpk}_j, \mathsf{ABE.msk}_j)$ is sampled using randomness derived from the PRF key K and supports circuits with inputs of length j. When one encrypts a message for a ciphertext attribute $\mathbf{x}$, one chooses the master public key $\mathsf{ABE.mpk}_{|\mathbf{x}|}$ and encrypts the message using the key, where $|\mathbf{x}|$ is the length of $\mathbf{x}$. We can encrypt for $\mathbf{x}$ with length at most 2^λ and therefore can deal with essentially unbounded length strings as ciphertext attributes. In order to generate a secret key for a machine M, one has to convert it into a circuit since our underlying ingredient is an ABE for circuits. The difference between an NFA machine M and a circuit is that while the former takes a string with arbitrary length as an input, the input length for the latter is fixed. To fill the gap, we prepare a circuit version of NFA M for all possible input lengths. Namely, we convert the machine M into an equivalent circuit $\widehat{M}_j$ with input length j for all $j \in [2^\lambda]$. Then, we generate ABE secret key associated with $\widehat{M}_j$ by running the key generation algorithm

[5] For the knowledgeable reader, bounded key variants of reusable garbled circuits exist, for instance by applying the compiler of [38], but using this in the aforementioned construction does not work due to the structure of their construction.

of the ABE for all j to obtain the NfaABE secret key $\{\mathsf{ABE.sk}_j\}_{j\in[2^\lambda]}$. When decrypting a ciphertext associated with $\mathbf{x}$, the decryptor chooses $\mathsf{ABE.sk}_{|\mathbf{x}|}$ and runs the decryption algorithm of the underlying ABE to retrieve the message.

Reducing the Number of Keys : Obviously, there are multiple problems with this approach. The first problem is that there are 2^λ instances of ABE and thus the secret key of NfaABE is exponentially large. To handle this, we thin out most of the instances and change the secret key to be $\{\mathsf{ABE.sk}_{2^j}\}_{j\in[0,\lambda]}$. In order to make sure that the decryption is still possible even with this change, we modify the encryption algorithm. To encrypt a message for an attribute $\mathbf{x}$, one chooses $i \in [0, \lambda]$ such that $2^{i-1} < |\mathbf{x}| \le 2^i$ and uses the i-th instance to encrypt the message, where if the length of $\mathbf{x}$ is not exactly 2^i, it is padded with blank symbols to adjust the length. This change reduces the number of instances down to be polynomial.

Reducing the Size of Keys : However, a bigger problem is that even though we reduced the *number* of secret keys, we did not reduce their size, which is still not polynomial. In particular, there is no guarantee on the size of $\mathsf{ABE.sk}_{2^\lambda}$ since the associated circuit $\widehat{M}_{2^\lambda}$ is of exponential size. Here, we leverage a crucial efficiency property that is enjoyed by the ABE for circuits constructed by Boneh et al. [22], namely, that the secret keys in this scheme are very short. The size of secret keys in their scheme is dependent only on the depth of the circuits being supported and *independent of the input length and size*. Thus, if we can ensure that the depth of $\widehat{M}_{2^\lambda}$ is polynomially bounded (even though the input is exponentially long), we are back in business.

However, converting the NFA to a circuit requires care. We note that implementing the trivial approach of converting an NFA to a circuit by keeping track of all possible states while reading input symbols results in circuit whose depth is linear in input length, which is exponential. To avoid this, we make use of a divide and conquer approach to evaluate the NFA, which makes the circuit depth poly-logarithmic in the input length. As a result, the size of the secret keys can be bounded by a polynomial as desired.

Efficiency of Key Generation : The final and the most difficult problem to be addressed is that even though we managed to make the size of $\{\mathsf{ABE.sk}_{2^j}\}_{j\in[0,\lambda]}$ polynomially bounded, computational time for generating it is still exponentially large, since so is the size of the associated circuits $\{\widehat{M}_{2^j}\}_{j\in[0,\lambda]}$. To resolve the problem, we note that the only algorithm which has the "space" to handle the unbounded input length is the encryption algorithm. Hence, we carefully divide the computation of generating $\{\mathsf{ABE.sk}_{2^j}\}_{j\in[0,\lambda]}$ into pieces so that the key generator only needs to do work proportional to the size of the machine, the encryptor does work proportional to the size of the input and the decryptor computes the requisite key on the fly.

To implement this idea, we use succinct single-key functional encryption (FE), which can be realized from the LWE assumption [2,37]. To support unbounded input length, we generate $\lambda + 1$ instances of the FE scheme to obtain $\{\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j\}_{j\in[0,\lambda]}$. The secret key of NfaABE is $\{\mathsf{FE.ct}_j\}_{j\in[0,\lambda]}$, where $\mathsf{FE.ct}_j = \mathsf{FE.Enc}(\mathsf{FE.mpk}_j, (M, \mathsf{K}))$ is an encryption of a description of the associated NFA M and the PRF key K under the j-th instance of the FE scheme. To provide the matching secret key, the encryptor appends $\mathsf{FE.sk}_i = \mathsf{FE.KeyGen}(\mathsf{FE.msk}_i, C_i)$ to the ciphertext. Here, $\mathbf{x}$ is the attribute vector of unbounded length, i is an integer s.t. $2^{i-1} < |\mathbf{x}| \leq 2^i$ and C_i is a circuit that takes as inputs the machine M and PRF key K and outputs an ABE secret key $\mathsf{ABE.sk}_{2^i}$ associated with M.

We are almost done – the decryptor chooses $\mathsf{FE.ct}_i$ with appropriate i from the received set $\{\mathsf{FE.ct}_j\}_{j\in[0,\lambda]}$ and decrypts it using $\mathsf{FE.sk}_i$ that is appended to the ciphertext to obtain an ABE secret key $\mathsf{ABE.sk}_{2^i}$. Then, it decrypts the ABE ciphertext also provided in the ciphertext to retrieve the message. Note that our construction is carefully designed so that we only require a *single* key of the succinct FE scheme.

Arguing the efficiency of the scheme requires care. In order to make the key generation algorithm run in polynomial time, we rely on the succinctness of the underlying FE. Recall that the succinctness property says that the running time of the encryption algorithm is independent of the size of the circuits being supported and only dependent on the depth and input and output length. In our construction, the computation of $\{\mathsf{FE.ct}_j = \mathsf{FE.Enc}(\mathsf{FE.mpk}_j, (M, \mathsf{K}))\}_{j\in[0,\lambda]}$ can be performed in polynomial time, since the input length $|M| + |\mathsf{K}|$ is bounded by a fixed polynomial[6] and so is the output length $|\mathsf{ABE.sk}_{2^j}|$. Note that we crucially use the succinctness of the FE here, since the size of the circuit C_{2^j}, which is supported by the j-th instance of FE, is polynomial in 2^j and thus exponential for $j = \lambda$.

Security: Our construction of NfaABE satisfies standard (selective) indistinguishability based security. The high level idea of the proof is outlined next. Intuitively, security follows from the security of the single key FE scheme and the underlying circuit ABE scheme. In the first step, we show that even though an adversary can obtain multiple FE ciphertexts and secret keys, it cannot obtain anything beyond their decryption results $\{\mathsf{FE.Dec}(\mathsf{FE.sk}_i, \mathsf{FE.ct}_i) = \mathsf{ABE.sk}_i\}$ by the security of the FE. Then, we leverage the security of the ABE to conclude that the message is indeed hidden. We note that in order to invoke the FE security, we need to ensure that only single secret key is revealed to the adversary for each instance of FE. This property is guaranteed, since the circuit for which a secret key of the j-th instance of FE is generated is fixed (i.e., C_{2^j}). Please see Sect. 3 for details.

Removing the Size Constraint on NFAs. So far, we have constructed NfaABE for NFA that can deal with unbounded input length and bounded size NFAs. Let us call such a scheme (u, b)-NfaABE, where "u" and "b" stand for "unbounded" and "bounded". We define (b, u)-NfaABE and (u, u)-NfaABE analogously, where the first parameter refers to input length and the second to machine size.

[6] Recall that we are only dealing with bounded size NFAs.

Our goal is to obtain (u,u)-NfaABE. At a high level, we compile (u,u)-NfaABE using two pieces, namely (u,b)-NfaABE which we have already constructed, and (b,u)-NfaABE, which we will instantiate next.

To construct (b,u)-NfaABE, our basic idea is to simply convert an NFA into an equivalent circuit and then use existing ABE for circuits schemes [22,39]. This approach almost works, but we need to exercise care to ensure that the depth of these circuits can be bounded since we hope to support NFAs of unbounded size. To fill this gap, we show that an NFA can be converted into an equivalent circuit whose depth is poly-logarithmic in the size of the NFA by again using the divide and conquer approach we discussed previously. This enables us to bound the depth of the circuits by a fixed polynomial, even if the size of corresponding NFA is unbounded and allows us to use existing ABE schemes for circuits to construct (b,u)-NfaABE.

We are ready to construct (u,u)-NfaABE by combining (u,b)-NfaABE and (b,u)-NfaABE. The master secret key of the (u,u)-NfaABE is a PRF key K. This PRF key defines a set of keys $\{(\mathsf{u},\mathsf{b})\text{-}\mathsf{NfaABE.msk}_j\}_{j\in[2^\lambda]}$ of the (u,b)-NfaABE scheme, where each (u,b)-$\mathsf{NfaABE.msk}_j$ supports NFAs with size j. Similarly, the PRF key also defines keys $\{(\mathsf{b},\mathsf{u})\text{-}\mathsf{NfaABE.msk}_j\}_{j\in[2^\lambda]}$ of the (b,u)-NfaABE scheme, where each (b,u)-$\mathsf{NfaABE.msk}_j$ supports input strings with length j. To encrypt a message with respect to a ciphertext attribute $\mathbf{x}$, it encrypts the message for $\mathbf{x}$ using (u,b)-$\mathsf{NfaABE.msk}_j$ to obtain (u,b)-$\mathsf{NfaABE.ct}_j$ for all $j \in [\mathbf{x}]$. Furthermore, it also encrypts the message for $\mathbf{x}$ using (b,u)-$\mathsf{NfaABE.msk}_{|\mathbf{x}|}$ to obtain (b,u)-$\mathsf{NfaABE.ct}_{|\mathbf{x}|}$. The final ciphertext is

$$\left(\ \{(\mathsf{u},\mathsf{b})\text{-}\mathsf{NfaABE.ct}_j\}_{j\in[|\mathbf{x}|]},\ (\mathsf{b},\mathsf{u})\text{-}\mathsf{NfaABE.ct}_{|\mathbf{x}|}\ \right).$$

To generate a secret key for a machine M, we essentially swap the roles of (u,b)-NfaABE and (b,u)-NfaABE. Namely, we generate a secret key (b,u)-$\mathsf{NfaABE.sk}_j$ for M using (b,u)-$\mathsf{NfaABE.msk}_j$ for all $j \in [|M|]$, where $|M|$ is the size of the machine M. We also generate (u,b)-$\mathsf{NfaABE.sk}_{|M|}$ for M using (u,b)-$\mathsf{NfaABE.msk}_{|M|}$. The final secret key is

$$\left(\ (\mathsf{u},\mathsf{b})\text{-}\mathsf{NfaABE.sk}_{|M|},\ \{(\mathsf{b},\mathsf{u})\text{-}\mathsf{NfaABE.sk}_j\}_{j\in[|M|]}\ \right).$$

To decrypt a ciphertext for attribute $\mathbf{x}$ using a secret key for an NFA machine M, we first compare $|\mathbf{x}|$ and $|M|$. If $|\mathbf{x}| > |M|$, it decrypts (u,b)-$\mathsf{NfaABE.ct}_{|M|}$ using (u,b)-$\mathsf{NfaABE.sk}_{|M|}$. Otherwise, it decrypts (b,u)-$\mathsf{NfaABE.ct}_{|\mathbf{x}|}$ using (u,b)-$\mathsf{NfaABE.sk}_{|\mathbf{x}|}$. It is not hard to see that the correctness of the resulting scheme follows from those of the ingredients. Furthermore, the security of the scheme is easily reduced to those of the ingredients, as the construction simply runs them in parallel with different parameters. The proof is by a hybrid argument, where we change the encrypted messages in a instance-wise manner. In Sect. 4, we streamline the construction and directly construct (u,u)-NfaABE from (u,b)-NfaABE and ABE for circuits instead of going through (b,u)-NfaABE.

Generalizations and Lower Bounds. We further generalize our ABE construction to obtain predicate encryption and bounded key functional encryption for

NFAs along with the first construction of resuable garbled NFA. These constructions are obtained by carefully replacing the underlying ABE for circuits with predicate encryption, bounded key functional encryption for circuits or reusable garbled circuits. This compiler requires some care as we need to argue that the delicate balance of efficiency properties that enable our NfaABE construction are not violated, as well as ensure that the constructions and security proofs translate. In the full version, we show that we can indeed ensure this, sometimes by employing additional tricks as required. In Sect. 5 we show that secret key functional encryption (SKFE) for DFA with security against unbounded collusion implies indistinguishability obfuscation for circuits. There, we essentially show that we can convert an SKFE for DFA into an SKFE for NC_1 circuit, which implies indistinguishability obfuscation for circuits by previous results [9,47]. The conversion is by encoding and purely combinatorial – we first convert an NC_1 circuit into an equivalent branching program and then leverage the similarity between the branching program and DFA to obtain the result.

Organization of the Paper. In Sect. 2, we provide the definitions and preliminaries we require. In Sect. 3, we provide our ABE for NFA supporting unbounded input but bounded machine length. In Sect. 4, we enhance the construction to support both unbounded input and unbounded machine length. The extensions of our construction to the setting of bounded key functional encryption and reusable garbled circuits for NFA will appear in the full version. In Sect. 5 we show that secret key functional encryption for DFA with security against unbounded collusion implies indistinguishability obfuscation for circuits. We conclude in Sect. 6.

2 Preliminaries

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

Notation. We begin by defining the notation that we will use throughout the paper. 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 $1 - \mathrm{negl}(n)$. The function $\log x$ is the base 2 logarithm of x. For any finite set S we denote $\mathcal{P}(S)$ to be the power set of S. For a circuit $C : \{0,1\}^{\ell_1+\ell_2} \to \{0,1\}$ and a string $\mathbf{x} \in \{0,1\}^{\ell_1}$, $C[\mathbf{x}] : \{0,1\}^{\ell_2} \to \{0,1\}$ denotes a circuit that takes $\mathbf{y}$ and outputs $C(\mathbf{x}, \mathbf{y})$. We construct $C[\mathbf{x}]$ in the following specified way. Namely, $C[\mathbf{x}]$ is the circuit that takes as input $\mathbf{y}$ and sets

$$z_i = \begin{cases} y_1 \wedge \neg y_1 & \text{if } x_i = 0 \\ y_1 \vee \neg y_1 & \text{if } x_i = 1 \end{cases}$$

and then computes $C(\mathbf{z}, \mathbf{y})$, where x_i, y_i, and z_i are the i-th bit of $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$, respectively. In the above, it is clear that $z_i = x_i$ and we have $C(\mathbf{z}, \mathbf{y}) = C(\mathbf{x}, \mathbf{y})$. Furthermore, it is also easy to see that $\mathsf{depth}(C[\mathbf{x}]) \leq \mathsf{depth}(C) + O(1)$ holds.

2.1 Definitions: Non Deterministic Finite Automata

A Non-Deterministic Finite Automaton (NFA) 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, $T : \Sigma \times Q \to \mathcal{P}(Q)$ is the transition function (stored as a table), q_{st} is the start state, $F \subseteq Q$ is the set of accepting states. For states $q, q' \in Q$ and a string $\mathbf{x} = (x_1, \ldots, x_k) \in \Sigma^k$, we say that q' is reachable from q by reading $\mathbf{x}$ 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 $q_{k+1} = q'$. We say $M(\mathbf{x}) = 1$ iff there is a state in F that is reachable from q_{st} by reading $\mathbf{x}$.

Remark 1. As it is known, we can transform an NFA with ϵ-transitions into a one without them by a simple and efficient conversion. The conversion preserves the size of the NFA. For simplicity and without loss of generality, we do not deal with an NFA with ϵ-transitions in this paper.

2.2 Definitions: Secret-Key Attribute Based Encryption for NFA

A secret-key attribute-based encryption (SKABE) scheme NfaABE for a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of four algorithms. In the following, we fix some alphabet $\Sigma = \Sigma_\lambda$ of size $2 \leq |\Sigma| \leq \text{poly}(\lambda)$.

– $\mathsf{NfaABE.Setup}(1^\lambda)$ is a PPT algorithm takes as input the unary representation of the security parameter and outputs the master secret key $\mathsf{NfaABE.msk}$.
– $\mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}, \mathbf{x}, m)$ is a PPT algorithm that takes as input the master secret key $\mathsf{NfaABE.msk}$, a string $\mathbf{x} \in \Sigma^*$ of arbitrary length and a message $m \in \mathcal{M}$. It outputs a ciphertext $\mathsf{NfaABE.ct}$.
– $\mathsf{NfaABE.KeyGen}(\mathsf{NfaABE.msk}, M)$ is a PPT algorithm that takes as input the master secret key $\mathsf{NfaABE.msk}$ and a description of an NFA machine M. It outputs a corresponding secret key $\mathsf{NfaABE.sk}_M$.
– $\mathsf{NfaABE.Dec}(\mathsf{NfaABE.sk}_M, M, \mathsf{NfaABE.ct}, \mathbf{x})$ is a deterministic polynomial time algorithm that takes as input the secret key $\mathsf{NfaABE.sk}_M$, its associated NFA M, a ciphertext $\mathsf{NfaABE.ct}$, and its associated string $\mathbf{x}$ and outputs either a message m' or $\perp$.

Remark 2. In our construction in Sect. 3.2, we will pass an additional parameter $\mathsf{s} = \mathsf{s}(\lambda)$ to the $\mathsf{NfaABE.Setup}$, $\mathsf{NfaABE.Enc}$, $\mathsf{NfaABE.KeyGen}$ algorithms denoting the description size of NFAs that the scheme can deal with. Later we give a construction in Sect. 4 which can support NFAs with arbitrary size.

Definition 1 (Correctness). *An SKABE scheme* NfaABE *is correct if for all NFAs M, all $\mathbf{x} \in \Sigma^*$ such that $M(\mathbf{x}) = 1$ and for all messages $m \in \mathcal{M}$,*

$$\Pr\left[\begin{array}{l}\mathsf{NfaABE.msk} \leftarrow \mathsf{NfaABE.Setup}(1^\lambda)\ ,\\ \mathsf{NfaABE.sk}_M \leftarrow \mathsf{NfaABE.KeyGen}(\mathsf{NfaABE.msk}, M)\ ,\\ \mathsf{NfaABE.ct} \leftarrow \mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}, \mathbf{x}, m)\ :\\ \mathsf{NfaABE.Dec}\big(\mathsf{NfaABE.sk}_M, M, \mathsf{NfaABE.ct}, \mathbf{x}\big) \neq m\end{array}\right] = \mathrm{negl}(\lambda)$$

where the probability is taken over the coins of $\mathsf{NfaABE.Setup}$*,* $\mathsf{NfaABE.KeyGen}$*, and* $\mathsf{NfaABE.Enc}$*.*

Definition 2 (Security for NfaABE). *The SKABE scheme* NfaABE *for a message space $\mathcal{M}$ is said to satisfy selective security if for any stateful PPT adversary* A*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that* $\mathsf{Adv}_{\mathsf{NfaABE,A}}(1^\lambda, \Sigma) :=$

$$\left|\Pr[\mathsf{Exp}^{(0)}_{\mathsf{NfaABE,A}}(1^\lambda) \to 1] - \Pr[\mathsf{Exp}^{(1)}_{\mathsf{NfaABE,A}}(1^\lambda) = 1]\right| \leq \mathrm{negl}(\lambda),$$

where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$, the experiment $\mathsf{Exp}^{(b)}_{\mathsf{NfaABE},A}$*, modeled as a game between the adversary* A *and a challenger, is defined as follows:*

1. **Setup phase:** *At the beginning of the game,* A *takes as input 1^λ and declares its target $X \subset \Sigma^*$, which is a set of strings of arbitrary size. Then the challenger samples* $\mathsf{NfaABE.msk} \leftarrow \mathsf{NfaABE.Setup}(1^\lambda)$*.*
2. **Query phase:** *During the game,* A *adaptively makes the following queries, in an arbitrary order and unbounded many times.*
   - *(a)* **Encryption queries:** A *submits to the challenger an attribute $\mathbf{x} \in X$ and a pair of messages $(m^{(0)}, m^{(1)}) \in (\mathcal{M}_\lambda)^2$. Then, the challenger replies with* $\mathsf{NfaABE.ct} \leftarrow \mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}, \mathbf{x}, m^{(b)})$ *in order.*
   - *(b)* **Key queries:** A *submits to the challenger an NFA M such that $M(\mathbf{x}) = 0$ for all $\mathbf{x} \in X$. Then, the challenger replies with* $\mathsf{NfaABE.sk}_M \leftarrow \mathsf{NfaABE.KeyGen}(\mathsf{NfaABE.msk}, M)$ *in order.*
3. **Output phase:** *A outputs a guess bit b' as the output of the experiment.*

Remark 3. As noted in Remark 2, our construction in Sect. 3.2 is indexed with an additional parameter s that specifies the size of NFAs being dealt with. In that case, the above security definitions are modified so that A chooses 1^{s} in addition to X (or X and S, in the case of very selective security) at the beginning of the game and key generation queries are made only for machines with size s.

2.3 Definitions: Attribute Based Encryption and Functional Encryption for Circuits

Attribute Based Encryption for Circuits. For $\lambda \in \mathbb{N}$, let $\mathcal{C}_{\mathsf{inp,d}}$ denote a family of circuits with inp bit inputs, an a-priori bounded depth d, and binary output and $\mathcal{C} = \{\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}\}_{\lambda \in \mathbb{N}}$. An attribute-based encryption (ABE) scheme ABE for $\mathcal{C}$ over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of four algorithms:

- $\mathsf{ABE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}})$ is a PPT algorithm takes as input the unary representation of the security parameter, the length $\mathsf{inp} = \mathsf{inp}(\lambda)$ of the input and the depth $\mathsf{d} = \mathsf{d}(\lambda)$ of the circuit family $\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}$ to be supported. It outputs the master public key and the master secret key $(\mathsf{ABE.mpk}, \mathsf{ABE.msk})$.
- $\mathsf{ABE.Enc}(\mathsf{ABE.mpk}, \mathbf{x}, m)$ is a PPT algorithm that takes as input the master public key $\mathsf{ABE.mpk}$, a string $\mathbf{x} \in \{0,1\}^{\mathsf{inp}}$ and a message $m \in \mathcal{M}$. It outputs a ciphertext $\mathsf{ABE.ct}$.
- $\mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}, \mathsf{ABE.msk}, C)$ is a PPT algorithm that takes as input the master secret key $\mathsf{ABE.msk}$ and a circuit $C \in \mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}$ and outputs a corresponding secret key $\mathsf{ABE.sk}_C$.
- $\mathsf{ABE.Dec}(\mathsf{ABE.mpk}, \mathsf{ABE.sk}_C, C, \mathsf{ABE.ct}, \mathbf{x})$ is a deterministic algorithm that takes as input the secret key $\mathsf{ABE.sk}_C$, its associated circuit C, a ciphertext $\mathsf{ABE.ct}$, and its associated string $\mathbf{x}$ and outputs either a message m' or $\perp$.

Definition 3 (Correctness). *An ABE scheme for circuits* ABE *is correct if for all* $\lambda \in \mathbb{N}$*, polynomially bounded* inp *and* d*, all circuits* $C \in \mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}$*, all* $\mathbf{x} \in \{0,1\}^{\mathsf{inp}}$ *such that* $C(\mathbf{x}) = 1$ *and for all messages* $m \in \mathcal{M}$*,*

$$\Pr\left[\begin{array}{l}(\mathsf{ABE.mpk}, \mathsf{ABE.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}),\\ \mathsf{ABE.sk}_C \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}, \mathsf{ABE.msk}, C),\\ \mathsf{ABE.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}, \mathbf{x}, m) :\\ \mathsf{ABE.Dec}\Big(\mathsf{ABE.mpk}, \mathsf{ABE.sk}_C, C, \mathsf{ABE.ct}, \mathbf{x}\Big) \neq m\end{array}\right] = \mathrm{negl}(\lambda)$$

where the probability is taken over the coins of $\mathsf{ABE.Setup}$*,* $\mathsf{ABE.KeyGen}$*, and* $\mathsf{ABE.Enc}$.

Definition 4 (Selective Security for ABE**).** *The ABE scheme* ABE *for a circuit family* $\mathcal{C} = \{\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}\}_{\lambda\in\mathbb{N}}$ *and a message space* $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ *is said to satisfy selective security if for any stateful PPT adversary* A*, there exists a negligible function* $\mathrm{negl}(\cdot)$ *such that*

$$\mathsf{Adv}_{\mathsf{ABE},\mathsf{A}}(1^\lambda) = \left|\Pr[\mathsf{Exp}^{(0)}_{\mathsf{ABE},\mathsf{A}}(1^\lambda) = 1] - \Pr[\mathsf{Exp}^{(1)}_{\mathsf{ABE},\mathsf{A}}(1^\lambda) = 1]\right| \leq \mathrm{negl}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$*, where for each* $b \in \{0,1\}$ *and* $\lambda \in \mathbb{N}$*, the experiment* $\mathsf{Exp}^{(b)}_{\mathsf{ABE},\mathsf{A}}$*, modeled as a game between adversary* A *and a challenger, is defined as follows:*

1. **Setup phase:** *On input* $1^\lambda, \mathsf{A}$ *submits* $(1^{\mathsf{inp}}, 1^{\mathsf{d}})$ *and the target* $X \subset \{0,1\}^{\mathsf{inp}}$*,which is a set of binary strings of length* inp*, to the challenger. The challenger samples* $(\mathsf{ABE.mpk}, \mathsf{ABE.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}})$ *and replies to* A *with* $\mathsf{ABE.mpk}$.
2. **Query phase:** *During the game,* A *adaptively makes the following queries, in an arbitrary order and unbounded many times.*
   (a) **Key Queries:** A *chooses a circuit* $C \in \mathcal{C}_{\mathsf{inp},\mathsf{d}}$ *that satisfies* $C(\mathbf{x}) = 0$ *for all* $\mathbf{x} \in X$*. For each such query, the challenger replies with* $\mathsf{ABE.sk}_C \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}, \mathsf{ABE.msk}, C)$.

   (b) **Encryption Queries:** A *submits a string* $\mathbf{x} \in X$ *and a pair of equal length messages* $(m_0, m_1) \in (\mathcal{M})^2$ *to the challenger. The challenger replies to A with* $\mathsf{ABE.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}, \mathbf{x}, m_b)$.

3. **Output phase:** *A outputs a guess bit b' as the output of the experiment.*

Remark 4. The above definition allows an adversary to make encryption queries multiple times. More standard notion of the security for an ABE restricts the adversary to make only a single encryption query. It is well-known that they are actually equivalent, which is shown by a simple hybrid argument. We adopt the above definition since it is convenient for our purpose.

In our construction of SKABE for NFA in Sect. 3.2, we will use the ABE scheme by Boneh et al. [22] as a building block. The following theorem summarizes the efficiency properties of their construction.

Theorem 1 (Adapted from [22]). *There exists a selectively secure ABE scheme* $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$ *with the following properties under the LWE assumption.*

1. *The circuit* $\mathsf{ABE.Setup}(\cdot,\cdot,\cdot;\cdot)$*, which takes as input* $1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}$*, and a randomness* r *and outputs* $\mathsf{ABE.msk} = \mathsf{ABE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}; r)$*, can be implemented with depth* $\mathrm{poly}(\lambda, \mathsf{d})$*. In particular, the depth of the circuit is independent of* inp *and the length of the randomness* r*.*
2. *We have* $|\mathsf{ABE.sk}_C| \le \mathrm{poly}(\lambda, \mathsf{d})$ *for any* $C \in \mathcal{C}_{\mathsf{inp},\mathsf{d}}$*, where* $(\mathsf{ABE.mpk}, \mathsf{ABE.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}})$ *and* $\mathsf{ABE.sk}_C \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}, \mathsf{ABE.msk}, C)$*. In particular, the length of the secret key is independent of the input length* inp *and the size of the circuit* C*.*
3. *Let* $C : \{0,1\}^{\mathsf{inp}+\ell} \to \{0,1\}$ *be a circuit such that we have* $C[v] \in \mathcal{C}_{\mathsf{inp},d}$ *for any* $v \in \{0,1\}^\ell$*. Then, the circuit* $\mathsf{ABE.KeyGen}(\cdot,\cdot,C[\cdot];\cdot)$*, that takes as input* $\mathsf{ABE.mpk}$*,* $\mathsf{ABE.msk}$*,* v*, and randomness* $\widehat{\mathsf{R}}$ *and outputs* $\mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}, \mathsf{ABE.msk}, C[v]; \widehat{\mathsf{R}})$*, can be implemented with depth* $\mathsf{depth}(C) \cdot \mathrm{poly}(\lambda, \mathsf{d})$*.*

Functional Encryption for Circuits. For $\lambda \in \mathbb{N}$, let $\mathcal{C}_{\mathsf{inp},\mathsf{d},\mathsf{out}}$ denote a family of circuits with inp bit inputs, depth d, and output length out and $\mathcal{C} = \{\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda),\mathsf{out}(\lambda)}\}_{\lambda\in\mathbb{N}}$. A functional encryption (FE) scheme $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ for $\mathcal{C}$ consists of four algorithms:

- $\mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}, 1^{\mathsf{out}})$ is a PPT algorithm takes as input the unary representation of the security parameter, the length $\mathsf{inp} = \mathsf{inp}(\lambda)$ of the input, depth $\mathsf{d} = \mathsf{d}(\lambda)$, and the length of the output $\mathsf{out} = \mathsf{out}(\lambda)$ of the circuit family $\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda),\mathsf{out}(\lambda)}$ to be supported. It outputs the master public key $\mathsf{FE.mpk}$ and the master secret key $\mathsf{FE.msk}$.
- $\mathsf{FE.KeyGen}(\mathsf{FE.mpk}, \mathsf{FE.msk}, C)$ is a PPT algorithm that takes as input the master public key $\mathsf{FE.mpk}$, master secret key $\mathsf{FE.msk}$, and a circuit $C \in \mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda),\mathsf{out}(\lambda)}$ and outputs a corresponding secret key $\mathsf{FE.sk}_C$.

- $\mathsf{FE.Enc}(\mathsf{FE.mpk}, \mathbf{x})$ is a PPT algorithm that takes as input the master public key $\mathsf{FE.mpk}$ and an input message $\mathbf{x} \in \{0,1\}^{\mathsf{inp}(\lambda)}$ and outputs a ciphertext $\mathsf{FE.ct}$.
- $\mathsf{FE.Dec}(\mathsf{FE.mpk}, \mathsf{FE.sk}_C, \mathsf{FE.ct})$ is a deterministic algorithm that takes as input the master public key $\mathsf{FE.mpk}$, a secret key $\mathsf{FE.sk}_C$ and a ciphertext $\mathsf{FE.ct}$ and outputs $C(\mathbf{x})$.

Definition 5 (Correctness). *A functional encryption scheme* FE *is correct if for all* $C \in \mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda),\mathsf{out}(\lambda)}$ *and all* $\mathbf{x} \in \{0,1\}^{\mathsf{inp}(\lambda)}$,

$$\Pr\left[\begin{array}{l}(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{d}(\lambda)}, 1^{\mathsf{out}(\lambda)});\\ \mathsf{FE.sk}_C \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.mpk}, \mathsf{FE.msk}, C);\\ \mathsf{FE.Dec}\Big(\mathsf{FE.mpk}, \mathsf{FE.sk}_C, \mathsf{FE.Enc}(\mathsf{FE.mpk}, \mathbf{x})\Big) \neq C(\mathbf{x})\end{array}\right] = \mathrm{negl}(\lambda)$$

where the probability is taken over the coins of $\mathsf{FE.Setup}$, $\mathsf{FE.KeyGen}$, $\mathsf{FE.Enc}$ *and,* $\mathsf{FE.Dec}$).

We then define full simulation based security for single key FE as in [37, Definition 2.13].

Definition 6 ($\mathsf{FULL\text{-}SIM}$ Security). *Let* FE *be a functional encryption scheme for a circuits. For a stateful PPT adversary* A *and a stateless PPT simulator* Sim, *consider the following two experiments:*

```
Exp^{real}_{FE,A}(1^λ):
1: $(1^{\mathsf{inp}}, 1^{\mathsf{d}}, 1^{\mathsf{out}}) \leftarrow \mathsf{A}(1^\lambda)$
2: $(\mathsf{FE.mpk}, \mathsf{FE.msk})$
   $\leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}, 1^{\mathsf{out}})$
3: $C \leftarrow \mathsf{A}(\mathsf{FE.mpk})$
4: $\mathsf{FE.sk}_C$
   $\leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.mpk}, \mathsf{FE.msk}, C)$
5: $\alpha \leftarrow \mathsf{A}^{\mathsf{FE.Enc}(\mathsf{FE.mpk},\cdot)}(\mathsf{FE.mpk}, \mathsf{FE.sk}_C)$

Exp^{ideal}_{FE,Sim}(1^λ):
1: $(1^{\mathsf{inp}}, 1^{\mathsf{d}}, 1^{\mathsf{out}}) \leftarrow \mathsf{A}(1^\lambda)$
2: $(\mathsf{FE.mpk}, \mathsf{FE.msk})$
   $\leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}, 1^{\mathsf{out}})$
3: $C \leftarrow \mathsf{A}(\mathsf{FE.mpk})$
4: $\mathsf{FE.sk}_C$
   $\leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.mpk}, \mathsf{FE.msk}, C)$
5: $\alpha \leftarrow \mathsf{A}^{\mathsf{O}(\cdot)}(\mathsf{FE.mpk}, \mathsf{FE.sk}_C)$
```

Here, $\mathsf{O}(\cdot)$ *is an oracle that on input* $\mathbf{x}$ *from* A, *runs* Sim *with inputs* $(\mathsf{FE.mpk}, \mathsf{sk}_C, C, C(\mathbf{x}), 1^{\mathsf{inp}})$ *to obtain a ciphertext* $\mathsf{FE.ct}$ *and returns it to the adversary* A.

The functional encryption scheme FE *is then said to be single query* $\mathsf{FULL\text{-}SIM}$ *secure if there exists a PPT simulator* Sim *such that for every PPT adversary* A, *the following two distributions are computationally indistinguishable:*

$$\left\{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE},\mathsf{A}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathrm{Sim}}(1^\lambda)\right\}_{\lambda\in\mathbb{N}}$$

Remark 5. The above definition allows an adversary to make encryption queries multiple times. In the security notion defined in [37], the adversary is allowed to make only a single encryption query. Similarly to the case of ABE, it is easy to see that these definitions are actually equivalent (See Remark 4). We adopt the above definition since it is convenient for our purpose.

In our construction of SKABE for NFA in Sect. 3.2, we will use the FE scheme by Goldwasser et al. [37] as a building block. The following theorem summarizes the efficiency properties of their construction.

Theorem 2 ([37]). *There exists an FE scheme* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *with the following properties.*

1. *For any polynomially bounded* $\mathsf{inp}(\lambda), \mathsf{d}(\lambda), \mathsf{out}(\lambda)$*, all the algorithms in* FE *run in polynomial time. Namely, the running time of* $\mathsf{FE.Setup}$ *and* $\mathsf{FE.Enc}$ *do not depend on the size of circuit description to be supported by the scheme.*
2. *Assuming the subexponential hardness of the LWE problem, the scheme satisfies full-simulation-based security.*

We note that the first property above is called succinctness or semi-compactness of FE. A stronger version of the efficiency property called compactness requires the running time of the encryption algorithm to be dependent only on the length of input message $\mathbf{x}$. An FE with compactness is known to imply indistinguishability obfuscation [12,21].

3 Attribute-Based Encryption for NFA

3.1 NFA as NC Circuit

Here, we introduce a theorem that provides an efficient algorithm that converts an NFA into an equivalent circuit with shallow depth. The shallowness of the circuit will play a crucial role in our construction of SKABE for NFA. In the following, for ease of notation, we often input a string in Σ^* to a circuit with the understanding that the input is actually a binary string encoding a string in Σ^*. To do so, we set $\eta := \lceil \log(|\Sigma|+1) \rceil$ and regard a symbol in Σ as a binary string in $\{0,1\}^\eta$ by a natural injection map from Σ to $\{0,1\}^\eta$. Furthermore, we also introduce a special symbol $\perp$ that is not in Σ and assign an unused symbol in $\{0,1\}^\eta$ to it. Intuitively, $\perp$ represents a blank symbol that will be used to adjust the length of a string. We will use alphabets $\{0,1\}^\eta$ and $\Sigma \cup \{\perp\}$ interchangeably.

Theorem 3. *Let Σ be an alphabet for NFAs. Then we have the following:*

1. *There exists a family of circuits* $\{\mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}\}_{\mathsf{s},\ell \in \mathbb{N}}$ *where the circuit* $\mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}$ *takes as input an NFA M with size* s *and outputs a circuit*

$\widehat{M_\ell} : (\Sigma \cup \{\perp\})^\ell \to \{0,1\}$. *Furthermore, for all* $\ell, \mathsf{s} \in \mathbb{N}$, *all string* $\mathbf{x} \in \Sigma^{\leq \ell}$, *and all NFA* M *with size* s, *we have*

$$\widehat{M_\ell}(\hat{\mathbf{x}}) = M(\mathbf{x}),$$

where $\widehat{M_\ell} = \mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}(M)$ *and* $\hat{\mathbf{x}} = \mathbf{x} \| \perp^{\ell - |\mathbf{x}|}$.

2. *The depths of the circuits* $\mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}$ *and* $\widehat{M_\ell} = \mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}(M)$ *for an NFA* M *of size* s *are bounded by* $\mathrm{poly}(\log \mathsf{s}, \log \ell)$. *Furthermore, the sizes of these circuits are bounded by* $\mathrm{poly}(\mathsf{s}, \ell)$.

The proof is by divide and conquer and will appear in the full version.

3.2 Construction: SKABE for Bounded Size NFA

We construct an SKABE scheme for NFA denoted by $\mathsf{NfaABE} = (\mathsf{NfaABE.Setup}, \mathsf{NfaABE.KeyGen}, \mathsf{NfaABE.Enc}, \mathsf{NfaABE.Dec})$ from the following ingredients:

1. $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval})$: a pseudorandom function, where a PRF key $\mathsf{K} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ defines a function $\mathsf{PRF.Eval}(\mathsf{K}, \cdot) : \{0,1\}^\lambda \to \{0,1\}$. We denote the length of K by $|\mathsf{K}|$.
2. $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$: a functional encryption scheme for circuit with the efficiency property described in Item 1 of Theorem 2. We can instantiate FE with the scheme proposed by Goldwasser et al. [37].
3. $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$: An ABE scheme that satisfies the efficiency properties described in Theorem 1. We can instantiate ABE with the scheme proposed by Boneh et al. [22].
4. $U(\cdot,\cdot)$: a universal circuit that takes as input a circuit C of fixed depth and size and an input $\mathbf{x}$ to the circuit and outputs $C(\mathbf{x})$. We often denote by $U[C](\cdot) = U(C, \cdot)$ a universal circuit U with the first input C being hardwired. We need to have $\mathsf{depth}(U) \leq O(\mathsf{depth}(C))$. For construction of such a universal circuit, we refer to [30].

Below we provide our construction for SKABE for NFA. In the description below, we abuse notation and denote as if the randomness used in a PPT algorithm was a key K of the pseudorandom function PRF. Namely, for a PPT algorithm (or circuit) A that takes as input x and a randomness $r \in \{0,1\}^\ell$ and outputs y, $\mathsf{A}(x; \mathsf{K})$ denotes an algorithm that computes $r := \mathsf{PRF}(\mathsf{K}, 1) \| \mathsf{PRF}(\mathsf{K}, 2) \| \cdots \| \mathsf{PRF}(\mathsf{K}, \ell)$ and runs $\mathsf{A}(x; r)$. Note that if A is a circuit, this transformation makes the size of the circuit polynomially larger and adds a fixed polynomial overhead to its depth. In particular, even if we add this change to $\mathsf{ABE.Setup}$ and $\mathsf{ABE.KeyGen}$, the efficiency properties of ABE described in Theorem 1 is preserved.

$\mathsf{NfaABE.Setup}(1^\lambda, 1^{\mathsf{s}})$: On input the security parameter 1^λ and a description size s of an NFA, do the following:

1. For $j \in [0, \lambda]$, sample PRF keys $\widehat{\mathsf{K}}_j, \mathsf{R}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.

2. For $j \in [0, \lambda]$, sample $(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{out}(\lambda)}, 1^{\mathsf{d}(\lambda)})$.
   Here, we generate $\lambda + 1$ instances of FE. Note that all instances support a circuit class with input length $\mathsf{inp}(\lambda) = \mathsf{s} + 2|\mathsf{K}|$, output length $\mathsf{out}(\lambda)$, and depth $\mathsf{d}(\lambda)$, where $\mathsf{out}(\lambda)$ and $\mathsf{d}(\lambda)$ are polynomials in the security parameter that will be specified later.
3. Output $\mathsf{NfaABE.msk} = (\{\widehat{\mathsf{K}}_j, \mathsf{R}_j, \mathsf{FE.mpk}_j, \mathsf{FE.msk}_j\}_{j\in[0,\lambda]})$.

$\mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}, \mathbf{x}, m, 1^{\mathsf{s}})$: On input the master secret key $\mathsf{NfaABE.msk}$, an attribute $\mathbf{x} \in \Sigma^*$ of length at most 2^λ, a message m and the description size s of NFA, do the following:
1. Parse the master secret key as $\mathsf{NfaABE.msk} \rightarrow (\{\widehat{\mathsf{K}}_j, \mathsf{R}_j, \mathsf{FE.mpk}_j, \mathsf{FE.msk}_j\}_{j\in[0,\lambda]})$.
2. Set $\hat{\mathbf{x}} = \mathbf{x} \| \bot^{2^i - \ell}$, where $\ell = |\mathbf{x}|$ and $i = \lceil \log \ell \rceil$.
3. Compute an ABE key pair $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) = \mathsf{ABE.Setup}(1^\lambda, 1^{2^i\eta}, 1^{\hat{\mathsf{d}}}; \widehat{\mathsf{K}}_i)$ with $\widehat{\mathsf{K}}_i$ as the randomness.
   Here, we generate an instance of ABE that supports a circuit class with input domain $\{0,1\}^{2^i\eta} \supseteq (\Sigma \cup \{\bot\})^{2^i}$ and depth $\hat{\mathsf{d}}$.
4. Compute $\mathsf{ABE.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, \hat{\mathbf{x}}, m)$ as an ABE ciphertext for the message m under attribute $\hat{\mathbf{x}}$.
5. Obtain $\mathsf{FE.sk}_i = \mathsf{FE.KeyGen}(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i, C_{\mathsf{s},2^i}; \mathsf{R}_i)$, where $C_{\mathsf{s},2^i}$ is a circuit described in Fig. 1.
6. Output $\mathsf{NfaABE.ct} = (\mathsf{FE.sk}_i, \mathsf{ABE.mpk}_i, \mathsf{ABE.ct})$.

Function $C_{\mathsf{s},2^i}$

1. Parse the input $\mathbf{w} = (M, \widehat{\mathsf{K}}, \widehat{\mathsf{R}})$, where M is an NFA and $\widehat{\mathsf{K}}$ and $\widehat{\mathsf{R}}$ are PRF keys.
2. Compute $(\mathsf{ABE.mpk}, \mathsf{ABE.msk}) = \mathsf{ABE.Setup}(1^\lambda, 1^{2^i\eta}, 1^{\hat{\mathsf{d}}}; \widehat{\mathsf{K}})$.
3. Compute $\widehat{M_{2^i}} = \mathsf{To\text{-}Circuit}_{\mathsf{s},2^i}(M)$. (See Theorem 3 for the definition of To-Circuit.)
4. Compute and output $\mathsf{ABE.sk}_{U[\widehat{M_{2^i}}]} = \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}, \mathsf{ABE.msk}, U[\widehat{M_{2^i}}]; \widehat{\mathsf{R}})$.

Fig. 1. The description of the circuit.

$\mathsf{NfaABE.KeyGen}(\mathsf{NfaABE.msk}, M, 1^{\mathsf{s}})$: On input the master secret key $\mathsf{NfaABE.msk}$, the description of an NFA M and a size s of the NFA, if $|M| \neq \mathsf{s}$, output $\bot$ and abort. Else, proceed as follows.
1. Parse the master secret key as $\mathsf{NfaABE.msk} \rightarrow (\{\widehat{\mathsf{K}}_j, \mathsf{R}_j, \mathsf{FE.mpk}_j, \mathsf{FE.msk}_j\}_{j\in[0,\lambda]})$.
2. Sample $\widehat{\mathsf{R}}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for all $j \in [0, \lambda]$.
3. Compute $\mathsf{FE.ct}_j = \mathsf{FE.Enc}(\mathsf{FE.mpk}_j, (M, \widehat{\mathsf{K}}_j, \widehat{\mathsf{R}}_j))$ for all $j \in [0, \lambda]$.
4. Output $\mathsf{NfaABE.sk}_M = \{\mathsf{FE.ct}_j\}_{j\in[0,\lambda]}$.

$\mathsf{NfaABE.Dec}(\mathsf{NfaABE.sk}_M, M, \mathsf{NfaABE.ct}, \mathbf{x})$: On input a secret key for NFA M and a ciphertext encoded under attribute $\mathbf{x}$, proceed as follows:

1. Parse the secret key as $\mathsf{NfaABE.sk}_M \to \{\mathsf{FE.ct}_j\}_{j\in[0,\lambda]}$ and the ciphertext as $\mathsf{NfaABE.ct} \to (\mathsf{FE.sk}_i, \mathsf{ABE.mpk}_i, \mathsf{ABE.ct})$.
2. Set $\ell = |\mathbf{x}|$ and choose $\mathsf{FE.ct}_i$ from $\mathsf{NfaABE.sk}_M = \{\mathsf{FE.ct}_j\}_{j\in[0,\lambda]}$ such that $i = \lceil \log \ell \rceil < \lambda$.
3. Compute $y = \mathsf{FE.Dec}(\mathsf{FE.mpk}_i, \mathsf{FE.sk}_i, \mathsf{FE.ct}_i)$.
4. Compute and output $z = \mathsf{ABE.Dec}(\mathsf{ABE.mpk}_i, y, U[\widehat{M}_{2^i}], \mathsf{ABE.ct}_i, \hat{\mathbf{x}})$, where we interpret y as an ABE secret key and $\hat{\mathbf{x}} = \mathbf{x}\|\bot^{2^i-\ell}$.

3.3 Correctness of NfaABE

The following theorem asserts that our scheme is efficient. This directly follows from Theorem 3 and the efficiency of the underlying scheme NfaABE. We refer to full version for the formal proof.

Theorem 4. *Let* $|\Sigma|$, $\mathsf{d}(\lambda)$, $\hat{\mathsf{d}}(\lambda)$, *and* $\mathsf{out}(\lambda)$, *be polynomials in* λ. *Then,* $\mathsf{NfaABE} = (\mathsf{NfaABE.Setup}, \mathsf{NfaABE.KeyGen}, \mathsf{NfaABE.Enc}, \mathsf{NfaABE.Dec})$ *defined above runs in polynomial time.*

The following theorem addresses the correctness of the scheme.

Theorem 5. *For appropriately chosen* $\hat{\mathsf{d}}(\lambda)$, $\mathsf{out}(\lambda)$, *and* $\mathsf{d}(\lambda)$, *our scheme* NfaABE *is correct for any polynomially bounded* $\mathsf{s}(\lambda)$.

Proof. We have to show that if we set $\hat{\mathsf{d}}(\lambda)$, $\mathsf{out}(\lambda)$, and $\mathsf{d}(\lambda)$ appropriately, we have $z = m$ when $M(\mathbf{x}) = 1$, where z is the value retrieved in Step 3.2 of the decryption algorithm. To show this, let us set $\hat{\mathsf{d}}(\lambda) = \Omega(\lambda)$ and assume that

$$y = \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i, U[\widehat{M}_{2^i}]; \widehat{\mathsf{R}}_i) \tag{3.1}$$

holds for the moment, where y is the value retrieved in Step 3.2 of the decryption algorithm. Then, we have $z = m$ by the correctness of ABE if $U[\widehat{M}_{2^i}]$ is supported by the scheme, since we have

$$U[\widehat{M}_{2^i}](\hat{\mathbf{x}}) = \widehat{M}_{2^i}(\hat{\mathbf{x}}) = M(\mathbf{x}) = 1$$

by Item 1 of Theorem 3. We claim that the depth of $U[\widehat{M}_{2^i}]$ is at most $\hat{\mathsf{d}}$ and therefore $U[\widehat{M}_{2^i}]$ is indeed supported by the scheme. To see this, we observe that

$$\begin{aligned}
\mathsf{depth}(U[\widehat{M}_{2^i}]) &\le \mathsf{depth}(U(\cdot,\cdot)) + O(1) \\
&\le O(1)\cdot \mathsf{depth}(\widehat{M}_{2^i}) + O(1) \\
&\le \mathrm{poly}(\log \mathsf{s}, \log 2^i) \\
&\le \mathrm{poly}(\log \lambda) \\
&\le \hat{\mathsf{d}}
\end{aligned} \tag{3.2}$$

holds, where the second inequality follows from the property of the depth preserving universal circuit U and the third from Item 2 of Theorem 3.

It remains to prove that Eq. (3.1) holds if we set $\mathsf{d}(\lambda)$ and $\mathsf{out}(\lambda)$ appropriately. To do so, we show that the depth and the output length of $C_{\mathsf{s},2^i}$ are bounded by some fixed polynomials. By taking $\mathsf{d}(\lambda)$ and $\mathsf{out}(\lambda)$ larger than these polynomials, we can ensure that the circuit $C_{\mathsf{s},2^i}$ is supported by the FE scheme and thus Eq. (3.1) follows from the correctness of the FE, since we have

$$C_{\mathsf{s},2^i}(M, \widehat{\mathsf{K}}_i, \widehat{\mathsf{R}}_i) = \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i, U[\widehat{M_{2^i}}]; \widehat{\mathsf{R}}_i),$$

where $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) = \mathsf{ABE.Setup}(1^\lambda, 1^{2^i\eta}, 1^{\hat{\mathsf{d}}}; \widehat{\mathsf{K}}_i)$ by the definition of $C_{\mathsf{s},2^i}$. We first bound the depth of $C_{\mathsf{s},2^i}$. To do so, we first observe that Step 2 of $C_{\mathsf{s},2^i}$ can be implemented by a circuit of depth $\mathrm{poly}(\lambda, \hat{\mathsf{d}}) = \mathrm{poly}(\lambda)$ by Item 1 of Theorem 1. We then observe that Step 3 of $C_{\mathsf{s},2^i}$ can be implemented by a circuit of depth $\mathrm{poly}(\log \mathsf{s}, \log 2^i) = \mathrm{poly}(\log \lambda)$ by Item 2 of Theorem 3. We then bound the depth of the circuit that implements Step 4 of $C_{\mathsf{s},2^i}$. This step is implemented by the circuit $\mathsf{ABE.KeyGen}(\cdot, \cdot, U[\cdot]; \cdot)$ that takes as input $\mathsf{ABE.mpk}_i$, $\mathsf{ABE.msk}_i$, $U[\widehat{M_{2^i}}]$ constructed in the previous step, and $\widehat{\mathsf{R}}$ and returns $\mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i, U[\widehat{M_{2^i}}]; \widehat{\mathsf{R}})$. We have

$$\begin{aligned}\mathsf{depth}(\mathsf{ABE.KeyGen}(\cdot, \cdot, U[\cdot]; \cdot)) &\leq \mathrm{poly}(\lambda, \hat{\mathsf{d}}) \cdot \mathsf{depth}(U(\cdot, \cdot)) \\ &\leq \mathrm{poly}(\lambda, \hat{\mathsf{d}}) \cdot \hat{\mathsf{d}} \\ &\leq \mathrm{poly}(\lambda),\end{aligned}$$

where the first inequality follows from Item 3 of Theorem 1 and the second from Eq. (3.2). To sum up, we have that the depth of the circuit $C_{\mathsf{s},2^i}$ is bounded by some fixed polynomial.

We next bound the output length of $C_{\mathsf{s},2^i}$. Since the output of the circuit is $\mathsf{ABE.sk}_{U[\widehat{M_{2^i}}]} = \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i, U[\widehat{M_{2^i}}]; \widehat{\mathsf{R}})$, we bound the length of the ABE secret key. We have

$$|\mathsf{ABE.sk}_{U[\widehat{M_{2^i}}]}| \leq \mathrm{poly}(\lambda, \hat{\mathsf{d}}) \leq \mathrm{poly}(\lambda, \mathrm{poly}(\lambda)) \leq \mathrm{poly}(\lambda)$$

as desired, where the first inequality follows from the Item 2 of Theorem 1. This completes the proof of the theorem.

3.4 Proof of Security for NfaABE

Here, we prove that NfaABE defined above is secure, if so are FE and ABE. Formally, we have the following theorem.

Theorem 6. *Assume that* FE *satisfies full simulation based security,* ABE *is selectively secure, and that* PRF *is a secure pseudorandom function. Then,* NfaABE *satisfies selective security.*

Proof. To prove the theorem, let us fix a PPT adversary A and introduce the following game $\mathbf{Game}_i$ between the challenger and A for $i \in [0, \lambda]$.

$\mathbf{Game}_i$**:** The game proceeds as follows.

Setup phase. At the beginning of the game, A takes 1^λ as input and submits 1^{s} and the set of its target $X \subset \Sigma^*$ to the challenger. Then, the challenger chooses $\mathsf{NfaABE.msk} \leftarrow \mathsf{NfaABE.Setup}(1^\lambda, 1^{\mathsf{s}})$

The challenger answers the encryption and key queries made by A as follows.

Encryption queries. Given two messages $(m^{(0)}, m^{(1)})$ and $\mathbf{x} \in X$ from A, the challenger sets $\ell := |\mathbf{x}|$ and computes

$$\mathsf{NfaABE.ct} = \begin{cases} \mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}, \hat{\mathbf{x}}, m^{(0)}) & \text{If } \lceil \log \ell \rceil \geq i \\ \mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}, \hat{\mathbf{x}}, m^{(1)}) & \text{If } \lceil \log \ell \rceil \leq i-1. \end{cases}$$

Then, it returns $\mathsf{NfaABE.ct}$ to A.

Key queries. Given an NFA M from A, the challenger runs $\mathsf{NfaABE.sk}_M \leftarrow \mathsf{NfaABE.KeyGen}(\mathsf{NfaABE.msk}, M)$ and returns $\mathsf{NfaABE.sk}_M$ to A.

Finally, A outputs its guess b'.

In the following, let $\mathsf{E}_{\mathrm{xxx}}$ denote the probability that A outputs 1 in $\mathbf{Game}_{\mathrm{xxx}}$. It suffices to prove $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_{\lambda+1}]| = \mathrm{negl}(\lambda)$, since $\mathbf{Game}_0$ (resp., $\mathbf{Game}_{\lambda+1}$) corresponds to the selective security game with $b = 0$ (resp., $b = 1$). Since we have

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_{\lambda+1}]| \leq \sum_{i \in [0,\lambda]} |\Pr[\mathsf{E}_i] - \Pr[\mathsf{E}_{i+1}]|$$

by the triangle inequality, it suffices to show $|\Pr[\mathsf{E}_i] - \Pr[\mathsf{E}_{i+1}]| = \mathrm{negl}(\lambda)$ for $i \in [0, \lambda]$. Let us define $\ell_{\max}$ and $i_{\max}$ as

$$\ell_{\max} := \max\{|\mathbf{x}| : \mathbf{x} \in X\} \qquad \text{and} \qquad i_{\max} := \lceil \log \ell_{\max} \rceil.$$

Note that $\ell_{\max}$ is bounded by the running time of A and thus is polynomial in λ. We then observe that for $i > i_{\max}$, we have $\mathbf{Game}_i = \mathbf{Game}_{\lambda+1}$ and thus $\Pr[\mathsf{E}_i] - \Pr[\mathsf{E}_{i+1}] = 0$. Therefore, in the following, we will show that $|\Pr[\mathsf{E}_i] - \Pr[\mathsf{E}_{i+1}]| = \mathrm{negl}(\lambda)$ holds for $i \leq i_{\max}$. To do so, we further introduce the following sequence of games for $i \in [0, i_{\max}]$:

$\mathbf{Game}_{i,0}$**:** The game is the same as $\mathbf{Game}_i$.

$\mathbf{Game}_{i,1}$**:** In this game, we change the setup phase and the way encryption queries are answered as follows.

Setup phase. Given $X \subset \Sigma^*$ from A, the challenger chooses $\mathsf{NfaABE.msk} \leftarrow \mathsf{NfaABE.Setup}(1^\lambda, 1^{\mathsf{s}})$ as in the previous game. In addition, it computes

$$(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^{2^i \eta}, 1^{\hat{\mathsf{d}}}; \widehat{\mathsf{K}}_i)$$

and

$$\mathsf{FE.sk}_i \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i, C_{\mathsf{s},2^i}; \mathsf{R}_i).$$

Encryption queries. Given two messages $(m^{(0)}, m^{(1)})$ and $\mathbf{x} \in X$ from A, the challenger sets $\ell := |\mathbf{x}|$ and computes $\mathsf{NfaABE.ct}$ as in the previous game if $\lceil \log \ell \rceil \neq i$. Otherwise, it computes

$$\mathsf{ABE.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, \hat{\mathbf{x}}, m^{(0)})$$

and returns $\mathsf{NfaABE.ct} = (\mathsf{FE.sk}_i, \mathsf{ABE.mpk}_i, \mathsf{ABE.ct})$ to A, where $\mathsf{FE.sk}_i$ and $\mathsf{ABE.mpk}_i$ are the values that are computed in the setup phase.

Game$_{i,2}$: In this game, the challenger samples $\mathsf{FE.sk}_i$ as

$$\mathsf{FE.sk}_i \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i, C_{\mathsf{s},2^i})$$

in the setup phase. Namely, it is sampled using true randomness instead of the pseudorandom bits derived from the PRF key R_i.

Game$_{i,3}$: We change the way key queries are answered as follows:

Key queries. Given an NFA M of size s from A, the challenger answers the query as follows. It first chooses $\widehat{\mathsf{R}}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $j \in [0, \lambda]$ and computes

$$\mathsf{ABE.sk}_{U[\widehat{M}_{2^i}]} = \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i, U[\widehat{M}_{2^i}]; \widehat{\mathsf{R}}_i),$$

where $\mathsf{ABE.mpk}_i$ and $\mathsf{ABE.msk}_i$ are the values that are computed in the setup phase. It then computes $\mathsf{FE.ct}_j \leftarrow$

$$\begin{cases} \mathsf{FE.Enc}(\mathsf{FE.mpk}_j, (M, \widehat{\mathsf{K}}_j, \widehat{\mathsf{R}}_j)) & \text{If } j \in [0, \lambda] \backslash \{i\} \\ \mathrm{Sim}(\mathsf{FE.mpk}_i, \mathsf{FE.sk}_i, C_{\mathsf{s},2^i}, \mathsf{ABE.sk}_{U[\widehat{M}_{2^i}]}, 1^{\mathsf{inp}(\lambda)}) & \text{If } j = i. \end{cases} \tag{3.3}$$

Then, it returns $\mathsf{NfaABE.sk}_M := \{\mathsf{FE.ct}_j\}_{j \in [0,\lambda]}$ to A.

Game$_{i,4}$: In this game, the challenger samples $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i)$ in the setup phase as

$$(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^{2^i \eta}, 1^{\hat{\mathsf{d}}}).$$

It also generates $\mathsf{ABE.sk}_{U[\widehat{M}_{2^i}]}$ as

$$\mathsf{ABE.sk}_{U[\widehat{M}_{2^i}]} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i, U[\widehat{M}_{2^i}]).$$

when answering a key query. Namely, they are sampled using true randomness instead of the pseudorandom bits derived from the PRF keys $\widehat{\mathsf{K}}_i$ and $\widehat{\mathsf{R}}_i$.

Game$_{i,5}$: In this game, we change the way the encryption queries are answered as follows.

Encryption queries. Given two messages $(m^{(0)}, m^{(1)})$ and $\mathbf{x} \in X$ from A, the challenger sets $\ell := |\mathbf{x}|$ and computes $\mathsf{NfaABE.ct}$ as in the previous game if $\lceil \log \ell \rceil \neq i$. Otherwise, it computes

$$\mathsf{ABE.ct} = \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, \hat{\mathbf{x}}, m^{(1)})$$

and returns $\mathsf{NfaABE.ct} = (\mathsf{FE.sk}_i, \mathsf{ABE.mpk}_i, \mathsf{ABE.ct})$ to A, where $\mathsf{FE.sk}_i$ and $\mathsf{ABE.mpk}_i$ are the values that are computed in the setup phase.

Game$_{i,6}$: The game is the same as **Game$_{i+1}$**.

Since we have

$$|\Pr[\mathsf{E}_i] - \Pr[\mathsf{E}_{i+1}]| \le \sum_{j \in [6]} |\Pr[\mathsf{E}_{i,j-1}] - \Pr[\mathsf{E}_{i,j}]|$$

by the triangle inequality, it suffices to show $|\Pr[\mathsf{E}_{i,j-1}] - \Pr[\mathsf{E}_{i,j}]| = \mathrm{negl}(\lambda)$ for $j \in [6]$. To complete the proof of the theorem, it remains to prove the following lemmas.

Lemma 1. *We have* $\Pr[\mathsf{E}_{i,0}] = \Pr[\mathsf{E}_{i,1}]$.

Proof. The change introduced here is only conceptual, where $\mathsf{ABE.mpk}_i$ and $\mathsf{FE.sk}_i$ are computed beforehand. The lemma trivially follows.

Lemma 2. *We have* $|\Pr[\mathsf{E}_{i,1}] - \Pr[\mathsf{E}_{i,2}]| = \mathrm{negl}(\lambda)$.

Proof. We observe that R_i is used only when generating $\mathsf{FE.sk}_i$ in **Game$_{i,1}$**. Therefore, the lemma follows by a straightforward reduction to the security of PRF.

Lemma 3. *We have* $|\Pr[\mathsf{E}_{i,2}] - \Pr[\mathsf{E}_{i,3}]| = \mathrm{negl}(\lambda)$.

Proof. To prove the lemma, let us assume that $|\Pr[\mathsf{E}_{i,2}] - \Pr[\mathsf{E}_{i,3}]|$ is non-negligible and construct an adversary B that breaks the full simulation security of FE using A. B proceeds as follows.

Setup phase. At the beginning of the game, B inputs 1^λ to A and obtains 1^{s} and $X \subset \Sigma^*$ from A. Then B submits its target $(1^\lambda, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{out}(\lambda)})$. Then, the experiment samples

$$(\mathsf{FE.mpk}, \mathsf{FE.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{out}(\lambda)})$$

and returns $\mathsf{FE.mpk}$ to B. B then sets $\mathsf{FE.mpk}_i := \mathsf{FE.mpk}$. In the rest of the simulation, it implicitly sets $\mathsf{FE.msk}_i := \mathsf{FE.msk}$ without knowing the value. B then chooses $(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{out}(\lambda)}, 1^{\mathsf{d}(\lambda)})$ for $j \in [0, \lambda] \backslash \{i\}$. It also chooses $\widehat{\mathsf{K}}_j, \mathsf{R}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $j \in [0, \lambda]$ and $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^{2^i \eta}, 1^{\hat{\mathsf{d}}}; \widehat{\mathsf{K}}_i)$. Finally, it declares $C_{\mathsf{s},2^i}$ as a circuit for which it request a secret key. Then, the experiment runs

$$\mathsf{FE.sk} \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.mpk}, \mathsf{FE.msk}, C_{\mathsf{s},2^i})$$

and returns $\mathsf{FE.sk}$ to B. B sets $\mathsf{FE.sk}_i := \mathsf{FE.sk}$.

B then handles the encryption and key queries as follows.

Encryption queries. Given two messages $(m^{(0)}, m^{(1)})$ and $\mathbf{x} \in X$ from A, $\mathcal{B}$ sets $\ell := |\mathbf{x}|$ and $i' = \lceil \log \ell \rceil$. If $i' \neq i$, B answers the query using $(\widehat{\mathsf{K}}_{i'}, \mathsf{R}_{i'}, \mathsf{FE.mpk}_{i'}, \mathsf{FE.msk}_{i'})$. Otherwise, it computes $\mathsf{ABE.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_i, \hat{\mathbf{x}}, m^{(0)})$ and returns $\mathsf{NfaABE.ct} = (\mathsf{FE.sk}_i, \mathsf{ABE.mpk}_i, \mathsf{ABE.ct})$ to A, where $\mathsf{ABE.mpk}_i$ (resp., $\mathsf{FE.sk}_i$) is the value sampled by itself (resp., by the experiment) in the setup phase.

Key queries. Given an NFA M of size s from A, B first chooses $\widehat{\mathsf{R}}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $j \in [0, \lambda]$ and computes $\mathsf{FE.ct}_j = \mathsf{FE.Enc}(\mathsf{FE.mpk}_j, (M, \widehat{\mathsf{K}}_j, \widehat{\mathsf{R}}_j))$ for $j \in [0, \lambda] \backslash \{i\}$. B then submits $(M, \widehat{\mathsf{K}}_i, \widehat{\mathsf{R}}_i)$ to its encryption oracle. Then, the experiment computes $\mathsf{FE.ct} \leftarrow$

$$\begin{cases} \mathsf{FE.Enc}(\mathsf{FE.mpk}, (M, \widehat{\mathsf{K}}_i, \widehat{\mathsf{R}}_i)) & \text{If } \mathsf{B} \text{ is in } \mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE,B}}(1^\lambda) \\ \mathrm{Sim}(\mathsf{FE.mpk}, \mathsf{FE.sk}, C_{\mathsf{s},2^i}, C_{\mathsf{s},2^i}(M, \widehat{\mathsf{K}}_i, \widehat{\mathsf{R}}_i), 1^{\mathsf{inp}(\lambda)}) & \text{If } \mathsf{B} \text{ is in } \mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathrm{Sim}}(1^\lambda) \end{cases} \tag{3.4}$$

and returns $\mathsf{FE.ct}$ to B. B then sets $\mathsf{FE.ct}_i := \mathsf{FE.ct}$ and returns $\mathsf{NfaABE.sk}_M := \{\mathsf{FE.ct}_j\}_{j \in [0,\lambda]}$ to A.

Output phase: B outputs the same bit as A as its guess.

It is easy to see that B simulates $\mathbf{Game}_{i,2}$ if B is in the real game. We then claim that B simulates $\mathbf{Game}_{i,3}$ if B is in the simulated game. The only difference between these games is the way $\mathsf{FE.ct}_i$ is computed. In $\mathbf{Game}_{i,3}$, it is generated as Eq. (3.3) while in the simulation above, it is generated as Eq. (3.4) (with B being in $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathrm{Sim}}$). However, they are equivalent because $\mathcal{B}$ has set $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) := (\mathsf{FE.mpk}, \mathsf{FE.msk})$ and $\mathsf{FE.sk}_i := \mathsf{FE.sk}$ and we have

$$C_{\mathsf{s},2^i}(M, \widehat{\mathsf{K}}_i, \widehat{\mathsf{R}}_i) = \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i, U[\widehat{M}_{2^i}]; \widehat{\mathsf{R}}_i) = \mathsf{ABE.sk}_{U[\widehat{M}_{2^i}]}.$$

From the above observation, we can see that B breaks the security of FE if A distinguishes the two games. This completes the proof of the lemma.

Lemma 4. *We have* $|\Pr[\mathsf{E}_{i,3}] - \Pr[\mathsf{E}_{i,4}]| = \mathrm{negl}(\lambda)$.

Proof. Due to the change we introduced, $\widehat{\mathsf{K}}_i$ is not used to answer the encryption queries any more and used only when generating $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i)$ in $\mathbf{Game}_{i,3}$. We also observe that $\widehat{\mathsf{R}}_i$ is used only when generating $\mathsf{ABE.sk}_{U[\widehat{M}_{2^i}]}$. Therefore, the lemma follows by straightforward reductions to the security of PRF.

Lemma 5. *We have* $|\Pr[\mathsf{E}_{i,4}] - \Pr[\mathsf{E}_{i,5}]| = \mathrm{negl}(\lambda)$.

Proof. To prove the lemma, let us assume that $|\Pr[\mathsf{E}_{i,4}] - \Pr[\mathsf{E}_{i,5}]|$ is non-negligible and construct an adversary B that breaks the selective security of ABE using A. B proceeds as follows.

Setup phase. At the beginning of the game, B inputs 1^λ to A and obtains 1^{s} and $X \subset \Sigma^*$ from A. Then, B sets $X_i := \{\hat{\mathbf{x}} = \mathbf{x} \| \bot^{2^i - |\mathbf{x}|} : \mathbf{x} \in X,\ 2^{i-1} < |\mathbf{x}| \le 2^i\}$ and submits its target X_i and $(1^\lambda, 1^{2^i \eta}, 1^{\hat{\mathsf{d}}})$ to its challenger. Then, the challenger samples

$$(\mathsf{ABE.mpk}, \mathsf{ABE.msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^{2^i \eta}, 1^{\hat{\mathsf{d}}})$$

and returns $\mathsf{ABE.mpk}$ to B. B then sets $\mathsf{ABE.mpk}_i := \mathsf{ABE.mpk}$. In the rest of the simulation, it implicitly sets $\mathsf{ABE.msk}_i := \mathsf{ABE.msk}$ without knowing the value. It then chooses $\widehat{\mathsf{K}}_j, \mathsf{R}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $j \in [0, \lambda] \backslash \{i\}$ and $(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{out}(\lambda)}, 1^{\mathsf{d}(\lambda)})$ for $j \in [0, \lambda]$. It also computes $\mathsf{FE.sk}_i \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i, C_{\mathsf{s}, 2^i})$.

B then handles the encryption and key queries as follows.

Encryption queries. Given two messages $(m^{(0)}, m^{(1)})$ and $\mathbf{x} \in X$ from A, $\mathcal{B}$ sets $\ell := |\mathbf{x}|$ and $i' = \lceil \log \ell \rceil$. If $i' \neq i$, B answers the encryption query using $(\widehat{\mathsf{K}}_{i'}, \mathsf{R}_{i'}, \mathsf{FE.mpk}_{i'}, \mathsf{FE.msk}_{i'})$. Otherwise, $\mathcal{B}$ makes an encryption query for the attribute $\hat{\mathbf{x}} = \mathbf{x} \| \bot^{2^i - \ell}$ and messages $(m^{(0)}, m^{(1)})$ to its challenger. Then, the challenger runs

$$\mathsf{ABE.ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}, \hat{\mathbf{x}}, m^{(b)})$$

and returns a ciphertext $\mathsf{ABE.ct}$ to B. Then, it returns $\mathsf{NfaABE.ct} = (\mathsf{FE.sk}_i, \mathsf{ABE.mpk}_i, \mathsf{ABE.ct})$ to A. Here, B uses $\mathsf{FE.sk}_i$ that is sampled in the setup phase.

Key queries. Given an NFA M of size s from A, B first chooses $\widehat{\mathsf{R}}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for $j \in [0, \lambda] \backslash \{i\}$. It then queries a secret key for $U[\widehat{M}_{2^i}]$ to its challenger. Then, the challenger runs

$$\mathsf{ABE.sk}_{U[\widehat{M}_{2^i}]} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}, \mathsf{ABE.msk}, U[\widehat{M}_{2^i}])$$

and returns $\mathsf{ABE.sk}_{U[\widehat{M}_{2^i}]}$ to B. It then computes $\mathsf{FE.ct}_j$ for $j \in [0, \lambda]$ as Eq. (3.3) and returns $\mathsf{NfaABE.sk}_M := \{\mathsf{FE.ct}_j\}_{j \in [0, \lambda]}$ to A.

Output phase: B outputs the same bit as A as its guess.

It is easy to see that B simulates $\mathbf{Game}_{i,4}$ if $b = 0$ and $\mathbf{Game}_{i,5}$ if $b = 1$. Therefore, B breaks the security of ABE if A distinguishes the two games. It remains to prove that B is a legitimate adversary (i.e., it does not make any prohibited key queries). For any attribute $\hat{\mathbf{x}}$ for which B makes an encryption query and for any circuit $U[\widehat{M}_{2^i}]$ for which B makes a key query, we have

$$U[\widehat{M}_{2^i}](\hat{\mathbf{x}}) = \widehat{M}_{2^i}(\hat{\mathbf{x}}) = M(\mathbf{x}),$$

where the second equality above follows from Item 1 of Theorem 3. Therefore, B is a legitimate adversary as long as so is A. This completes the proof of the lemma.

Lemma 6. *We have* $|\Pr[\mathsf{E}_{i,5}] - \Pr[\mathsf{E}_{i,6}]| = \mathrm{negl}(\lambda)$.

Proof. This follows as in the indistinguishability of $\mathbf{Game}_{i,0}$ and $\mathbf{Game}_{i,4}$, but in the reverse order. That is, we first change the random bits used in $\mathsf{ABE.KeyGen}$ to a pseudorandom one by invoking the security of PRF. We then generate $\mathsf{FE.ct}_i$ by using $\mathsf{FE.Enc}$ instead of Sim by invoking the full-simulation security of FE. Finally, we change the random bits used in $\mathsf{ABE.KeyGen}$ to a pseudorandom one by invoking the security of PRF again.

This concludes the proof of Theorem 6.

3.5 Extensions

In the full version, we adapt our ABE construction to achieve (restricted versions of) attribute privacy. In more detail, we construct secret key predicate encryption and bounded key functional encryption for nondeterministic finite automata. We also additionally achieve machine privacy, improving the result of [8]. Intuitively, these results proceed by replacing the "inner" circuit ABE scheme in our compiler by predicate encryption or bounded key functional encryption scheme and arguing that the requisite efficiency requirements (Theorem 1) are not violated. We again refer to the full version for details.

4 Attribute Based Encryption for NFA with Unbounded Size Machines and Inputs

In this section we construct a secret-key attribute-based encryption scheme (SKABE) for nondeterministic finite automata of arbitrary sizes supporting inputs of arbitrary length. We denote our scheme by $\mathsf{uNfaABE} = (\mathsf{uNfaABE.Setup}, \mathsf{uNfaABE.KeyGen}, \mathsf{uNfaABE.Enc}, \mathsf{uNfaABE.Dec})$ and its construction uses the following two ingredients.

1. $\mathsf{NfaABE} = (\mathsf{NfaABE.Setup}, \mathsf{NfaABE.KeyGen}, \mathsf{NfaABE.Enc}, \mathsf{NfaABE.Dec})$: An SKABE for NFA supporting inputs of *unbounded length* but for *bounded size* machines. We instantiate NfaABE from our construction in Sect. 3.2.
2. $\mathsf{ABE} = (\mathsf{ABE.Setup}, \mathsf{ABE.KeyGen}, \mathsf{ABE.Enc}, \mathsf{ABE.Dec})$: An ABE scheme for circuits that satisfies the efficiency properties described in Theorem 1. We can instantiate ABE with the scheme proposed by Boneh et al. [22].
3. $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval})$: a pseudorandom function, where a PRF key $\mathsf{K} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ defines a function $\mathsf{PRF.Eval}(\mathsf{K}, \cdot) : \{0,1\}^\lambda \to \mathcal{R}$, where we assume $\mathcal{R}$ to be the randomness space of *both* $\mathsf{NfaABE.Setup}$ and $\mathsf{ABE.Setup}$ algorithms. Note that without loss of generality, we may assume $\mathcal{R} = \{0,1\}^{p(\lambda)}$ for some sufficiently large polynomial $p(\lambda)$.

Below we provide our construction for SKABE for NFA.

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

1. Sample two PRF keys $\mathsf{K_{NfaABE}} \leftarrow \mathsf{PRF.Setup}(1^\lambda), \mathsf{K_{ABE}} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
2. Output $\mathsf{uNfaABE.msk} = (\mathsf{K_{NfaABE}}, \mathsf{K_{ABE}})$.

$\mathsf{uNfaABE.Enc}(\mathsf{uNfaABE.msk}, \mathbf{x}, m)$: On input the master secret key $\mathsf{uNfaABE.msk}$, an attribute as $\mathbf{x} \in \varSigma^*$ of length at most 2^λ and a message $m \in \mathcal{M}$, do the following:

1. Parse the master secret key as $\mathsf{uNfaABE.msk} = (\mathsf{K_{NfaABE}}, \mathsf{K_{ABE}})$. Denote $\ell = |\mathbf{x}|$.
2. For all $i \in [\ell]$, do the following:
   (a) Sample $\mathsf{NfaABE.msk}_i \leftarrow \mathsf{NfaABE.Setup}(1^\lambda, 1^i; r_i)$ as an NfaABE master secret key, where $r_i = \mathsf{PRF.Eval}(\mathsf{K_{NfaABE}}, i)$.

   Note that i denotes the size of the NFAs that are supported by $\mathsf{NfaABE.msk}_i$.

   (b) Compute $\mathsf{NfaABE.ct}_i = \mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}_i, \mathbf{x}, m, 1^i)$.
3. Sample $(\mathsf{ABE.mpk}_\ell, \mathsf{ABE.msk}_\ell) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^\ell, 1^{\hat{\mathsf{d}}}; r_\ell)$ as an ABE key pair, where $r_\ell = \mathsf{PRF.Eval}(\mathsf{K_{ABE}}, \ell)$.
   Note that ℓ and $\hat{\mathsf{d}}$ denotes the input length and the depth of the circuit respectively that $(\mathsf{ABE.mpk}_\ell, \mathsf{ABE.msk}_\ell)$ supports.
4. Compute $\mathsf{ABE.ct}_\ell = \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_\ell, \mathbf{x}, m)$.
5. Output $\mathsf{uNfaABE.ct} = (\{\mathsf{NfaABE.ct}_i\}_{i\in[\ell]}, \mathsf{ABE.mpk}_\ell, \mathsf{ABE.ct}_\ell)$.

$\mathsf{uNfaABE.KeyGen}(\mathsf{uNfaABE.msk}, M)$: On input the master secret key $\mathsf{uNfaABE.msk}$ and the description of a NFA $M = (Q, \varSigma, T, q_{\mathsf{st}}, F)$, proceed as follows.

1. Parse the master secret key as $\mathsf{uNfaABE.msk} = (\mathsf{K_{NfaABE}}, \mathsf{K_{ABE}})$. Denote $\mathsf{s} = |M|$.
2. For all $i \in [\mathsf{s}]$, do the following:
   (a) Let $\widehat{M_i} = \mathsf{To\text{-}Circuit}_{\mathsf{s},i}(M)$. (See Theorem 3 for the definition of $\mathsf{To\text{-}Circuit}$.)

   (b) Sample $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^i, 1^{\hat{\mathsf{d}}}; r_i)$ as an ABE key pair, where $r_i = \mathsf{PRF.Eval}(\mathsf{K_{ABE}}, i)$.

   (c) Compute $\mathsf{ABE.sk}_i = \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i, \widehat{M_i})$.
   Note that $\forall i \in [\mathsf{s}]$, i and $\hat{\mathsf{d}}$ denotes the input length and the depth of the circuit respectively that $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i)$ supports.
3. Sample $\mathsf{NfaABE.msk}_\mathsf{s} \leftarrow \mathsf{NfaABE.Setup}(1^\lambda, 1^\mathsf{s}; r_\mathsf{s})$ as an NfaABE master secret key, where $r_\mathsf{s} = \mathsf{PRF.Eval}(\mathsf{K_{NfaABE}}, \mathsf{s})$.
4. Compute $\mathsf{NfaABE.sk}_\mathsf{s} = \mathsf{NfaABE.KeyGen}(\mathsf{NfaABE.msk}_\mathsf{s}, M)$.
5. Output $\mathsf{uNfaABE.sk}_M = (\mathsf{NfaABE.sk}_\mathsf{s}, \{\mathsf{ABE.mpk}_i, \mathsf{ABE.sk}_i\}_{i\in[\mathsf{s}]})$.

$\mathsf{uNfaABE.Dec}(\mathsf{uNfaABE.sk}_M, M, \mathsf{uNfaABE.ct}, \mathbf{x})$: On input a secret key for NFA M and a ciphertext encoded under some attribute $\mathbf{x}$, proceed as follows:

1. Parse the secret key as $\mathsf{uNfaABE.sk}_M = (\mathsf{NfaABE.sk}_{|M|}, \{\mathsf{ABE.mpk}_i, \mathsf{ABE.sk}_i\}_{i \in [|M|]})$ and the ciphertext as $\mathsf{uNfaABE.ct} = (\{\mathsf{NfaABE.ct}_i\}_{i\in[|\mathbf{x}|]}, \mathsf{ABE.mpk}_{|\mathbf{x}|}, \mathsf{ABE.ct}_{|\mathbf{x}|})$.
2. If $|\mathbf{x}| \geq |M|$, compute and output $\mathsf{NfaABE.Dec}(\mathsf{NfaABE.sk}_{|M|}, M, \mathsf{NfaABE.ct}_{|M|}, \mathbf{x})$.
3. Otherwise, compute and output $\mathsf{ABE.Dec}(\mathsf{ABE.mpk}_{|\mathbf{x}|}, \mathsf{ABE.sk}_{|\mathbf{x}|}, \widehat{M}_{|\mathbf{x}|}, \mathsf{ABE.ct}_{|\mathbf{x}|}, \mathbf{x})$, where $\widehat{M}_{|\mathbf{x}|} = \mathsf{To\text{-}Circuit}_{|M|,|\mathbf{x}|}(M)$.

The following theorems assert that our scheme is efficient, satisfies correctness, and is secure, as long as so are the underlying NfaABE and ABE schemes. Intuitively, these theorems follow since we simply run these underlying schemes in parallel. We refer to the full version for the formal proofs.

Theorem 7. *The scheme* $\mathsf{uNfaABE} = (\mathsf{uNfaABE.Setup}, \mathsf{uNfaABE.KeyGen}, \mathsf{uNfaABE.Enc}, \mathsf{uNfaABE.Dec})$ *defined above runs in polynomial time, as long as* $\hat{\mathsf{d}}$ *and* $|\Sigma|$ *are polynomials in* λ.

Theorem 8. *For appropriately chosen* $\hat{\mathsf{d}} = \hat{\mathsf{d}}(\lambda)$*, our scheme* uNfaABE *is correct for any NFA.*

Theorem 9. *Assume that* NfaABE *and* ABE *both satisfy selective indistinguishability based security and* PRF *is a secure pseudorandom function. Then,* uNfaABE *satisfies selective security.*

5 FE for DFA Implies iO

Here, we show that secret key functional encryption (SKFE) for DFA with security against unbounded collusion implies indistinguishability obfuscation (iO). This result illuminates the difficulty of constructing such SKFE from a standard assumption, since no construction of iO from standard assumption is known despite the significant research effort in recent years [1–3,5,7,8,11–13,17,21,34,35,37–40,40,50,51,51–54].

5.1 Preliminaries on DFA and Branching Programs

Here, we first recall that a deterministic finite automaton (DFA) is a special case of NFA where for the transition function T, $T(\sigma, q)$ consists of a single element in Q for any $\sigma \in \Sigma$ and $q \in Q$. We then define branching program similarly to [25].

Definition 7 (Branching Programs). *A width-5 permutation branching program* BP *of length* L *with input space* $\{0,1\}^\ell$ *is a sequence of* L *tuples of the form* $(\mathsf{var}(t), \sigma_{t,0}, \sigma_{t,1})$ *where*

- $\mathsf{var} : [L] \to [\ell]$ *is a function that associates the* t*-th tuple with an input bit* $x_{\mathsf{var}(t)}$.
- $\sigma_{j,0}$ *and* $\sigma_{j,1}$ *are permutations on* 5 *elements. We will think of* $\sigma_{j,0}$ *and* $\sigma_{j,1}$ *as bijective functions from the set* $\{1,2,3,4,5\}$ *to itself.*

The computation of the program BP *on input* $\mathbf{x} = (x_1, \ldots, x_\ell)$ *proceeds as follows. The state of the computation at any point in time* t *is a number* $\zeta_t \in \{1,2,3,4,5\}$. *Computation starts with the initial state* $\zeta_0 = 1$. *The state* ζ_t *is computed recursively as*

$$\zeta_t = \sigma_{t, x_{\mathsf{var}(t)}}(\zeta_{t-1}). \tag{5.1}$$

Finally, after L *steps, our state is* ζ_L. *The output of the computation* $\mathsf{BP}(\mathbf{x})$ *is* 1 *if* $\zeta_L = 1$ *and* 0 *otherwise.*

We will use the following theorem, which essentially says that an NC^1 circuit can be converted into an equivalent branching program.

Theorem 10 (Barrington's Theorem [19]). *Every Boolean NAND circuit* C *that acts on* ℓ *inputs and has depth* d *can be computed by a width-5 permutation branching program* BP *of length* 4^d. *Given the description of the circuit* BP*, the description of the branching program* BP *can be computed in* $\mathrm{poly}(\ell, 4^d)$ *time. In particular, if* C *is a polynomial-sized circuit with logarithmic depth (i.e., if the circuit is in* NC^1*),* BP *can be computed in polynomial time.*

5.2 SKFE for DFA Implies iO

We first state the following theorem, which will be useful for our purpose. We refer to the full version for the proof.

Theorem 11. *Let* $d = d(\lambda)$ *and* $\ell = \ell(\lambda)$ *be integers. There exist deterministic algorithms* Encode *and* ToDFA *with the following properties.*

- $\mathsf{Encode}(\mathbf{x}) \to \mathbf{y} \in \{0,1\}^n$, *where* $\mathbf{x} \in \{0,1\}^\ell$ *and* n *is a parameter determined by* d *and* ℓ.
- $\mathsf{ToDFA}(C) \to M$, *where* $C : \{0,1\}^\ell \to \{0,1\}$ *is a circuit with depth bounded by* d *and* M *is a DFA over alphabet* $\Sigma = \{0,1\}$.

We have that $M(\mathbf{y}) = 1$ *if and only if* $C(\mathbf{x}) = 1$. *We also have that the running time of* Encode *and* ToDFA *is* $\mathrm{poly}(\ell, 2^d)$. *In particular, if* C *is a polynomial-sized circuit with logarithmic depth (i.e., if the circuit is in* NC^1*),* Encode *and* $\mathsf{ToDFA}(C)$ *run in polynomial time.*

We then discuss that if there exists subexponentially secure SKFE for DFA that is very selectively secure against unbounded collusion, it can be converted into a secure indistinguishability obfuscation.

To do so, we first convert an SKFE for DFA into an SKFE for NC^1 circuits. The latter SKFE has the same setup algorithm as the former, but when generating a secret key for a circuit C, it first converts C into a DFA M using the algorithm in Theorem 11 and then invoke the key generation algorithm of the SKFE for DFA on input M. Similarly, when encrypting a message $\mathbf{x}$, it computes $\mathbf{y}$ as in Theorem 11 and then invoke the encryption algorithm of the SKFE for DFA on input $\mathbf{y}$. The decryption algorithm is defined naturally. It is easy to see that this conversion preserves the correctness and the security since we have $M(\mathbf{y}) = C(\mathbf{x})$ by Theorem 11.

Then, we apply the conversion given by [12,21] to the SKFE for NC^1 to obtain SKFE for all circuits. We then further apply the conversion by Kitagawa et al. [46,47] to the SKFE for all circuits to obtain iO. Note that while the former conversion incurs only polynomial loss, the latter conversion incurs subexponential security loss.

In summary, we obtain the following theorem.

Theorem 12. *If there exists a subexponentially secure SKFE scheme for DFA that is very selectively secure against unbounded collusion, then there exists an indistinguishability obfuscation.*

6 Conclusions

Several interesting questions arise from our work. The first is whether we may generalize our techniques to support more advanced models of computation. For the moment, we are restricted to NFAs, since we must bound the depth of the equivalent circuits by a fixed polynomial and this step fails for more general models such as Turing machines. Second, it would be interesting to design a public key variant of our scheme. Improving the security proof to satisfy adaptive rather than selective security is also a useful direction. Finally, it would be nice to find other applications for our techniques.

Acknowledgement. We thank anonymous reviewers of Crypto 2019 for their helpful comments. The third author is supported by JST CREST Grant Number JPMJCR19F6 and JSPS KAKENHI Grant Number 16K16068.

References

1. Abdalla, M., Bourse, F., Caro, A.D., Pointcheval, D.: Simple functional encryption schemes for inner products. Cryptology ePrint Archive, Report 2015/017 (2015). http://eprint.iacr.org/ To appear in PKC'15
2. Agrawal, S.: Stronger security for reusable garbled circuits, general definitions and attacks. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 3–35. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_1
3. Agrawal, S.: Indistinguishability obfuscation minus multilinear maps: new methods for bootstrapping and instantiation (2018)
4. Agrawal, S., Freeman, D.M., Vaikuntanathan, V.: Functional encryption for inner product predicates from learning with errors. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 21–40. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_2
5. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_12
6. Agrawal, S., Maitra, M.: FE and iO for turing machines from minimal assumptions. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018. LNCS, vol. 11240, pp. 473–512. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03810-6_18
7. Agrawal, S., Rosen, A.: Functional encryption for bounded collusions, revisited. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 173–205. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_7
8. 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)
9. Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: From selective to adaptive security in functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 657–677. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_32
10. 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
11. Ananth, P., Jain, A., Sahai, A.: Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. Cryptology ePrint Archive, Report 2018/615 (2018)
12. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_15
13. Ananth, P., Jain, A., Sahai, A.: Achieving compactness generically: indistinguishability obfuscation from non-compact functional encryption. IACR Cryptology ePrint Archive 2015/730 (2015)
14. Ananth, P., Sahai, A.: Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 152–181. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_6
15. Apon, D., Döttling, N., Garg, S., Mukherjee, P.: Cryptanalysis of indistinguishability obfuscations of circuits over GGH13. eprint 2016 (2016)

16. Attrapadung, N.: Dual system encryption via doubly selective security: framework, fully secure functional encryption for regular languages, and more. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 557–577. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_31
17. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 67–98. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_3
18. Barak, B., et al.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
19. Barrington, D.A.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC1. J. Comput. Syst. Sci. **38**(1), 150–164 (1989)
20. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
21. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: FOCS 2015, 163 (2015). http://eprint.iacr.org/2015/163
22. Boneh, D., et al.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 533–556. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_30
23. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_29
24. Boyen, X., Li, Q.: Attribute-based encryption for finite automata from LWE. In: Au, M.-H., Miyaji, A. (eds.) ProvSec 2015. LNCS, vol. 9451, pp. 247–267. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26059-4_14
25. Brakerski, Z., Vaikuntanathan, V.: Lattice-based FHE as secure as PKE. In: Proceedings of the 5th Conference on Innovations in Theoretical Computer Science, ITCS 2014 (2014)
26. Brakerski, Z., Vaikuntanathan, V.: Circuit-ABE from LWE: unbounded attributes and semi-adaptive security. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 363–384. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_13
27. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 3–12. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_1
28. 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
29. 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 (2016)
30. Cook, S.A., Hoover, H.J.: A depth-universal circuit. SIAM J. Comput. **14**(4), 833–839 (1985). https://doi.org/10.1137/0214058
31. Coron, J.-S., et al.: Zeroizing without low-level zeroes: new MMAP attacks and their limitations. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 247–266. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_12
32. Coron, J.S., Lee, M.S., Lepoint, T., Tibouchi, M.: Zeroizing attacks on indistinguishability obfuscation over CLT13. Eprint 2016 (2016)

33. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_1
34. 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/
35. Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_27
36. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run turing machines on encrypted data. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 536–553. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_30
37. 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)
38. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_11
39. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute based encryption for circuits. In: STOC (2013)
40. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 503–523. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_25
41. Gorbunov, S., Vinayagamurthy, D.: Riding on asymmetry: efficient abe for branching programs. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 550–574. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_23
42. Goyal, R., Koppula, V., Waters, B.: Semi-adaptive security and bundling functionalities made generic and easy. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 361–388. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_14
43. 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)
44. Hu, Y., Jia, H.: Cryptanalysis of GGH map. Cryptology ePrint Archive: Report 2015/301 (2015)
45. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_9
46. Kitagawa, F., Nishimaki, R., Tanaka, K.: Indistinguishability obfuscation for all circuits from secret-key functional encryption. IACR Cryptology ePrint Archive 2017/361 (2017)
47. Kitagawa, F., Nishimaki, R., Tanaka, K.: Obfustopia built on secret-key functional encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 603–648. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_20

48. 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
49. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_4
50. Lin, H.: Indistinguishability obfuscation from constant-degree graded encoding schemes. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 28–57. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_2
51. Lin, H.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 599–629. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_20
52. Lin, H., Matt, C.: Pseudo flawed-smudging generators and their application to indistinguishability obfuscation. Cryptology ePrint Archive, Report 2018/646 (2018)
53. Lin, H., Tessaro, S.: Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 630–660. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_21
54. Lin, H., Vaikuntanathan, V.: Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In: FOCS (2016)
55. Miles, E., Sahai, A., Zhandry, M.: Annihilation attacks for multilinear maps: cryptanalysis of indistinguishability obfuscation over GGH13. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 629–658. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_22
56. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
57. Waters, B.: Functional encryption for regular languages. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 218–235. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_14