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

Shweta Agrawal [*] Monosij Maitra [†] Shota Yamada [‡]

Abstract

Constructing Attribute Based Encryption (ABE) [SW05] 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* [Wat12] 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.

[*] IIT Madras, Chennai, India. Email: shweta.a@cse.iitm.ac.in
[†] IIT Madras, Chennai, India. Email: monosij@cse.iitm.ac.in
[‡] National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan. Email: yamada-shota@aist.go.jp

Contents

1 Introduction

Attribute based encryption (ABE) [SW05] 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 [SW05], ABE schemes have received a lot of attention in recent years [GPSW06, BW07, BSW07, KSW08, LOS$^+$10, AFV11, Wat12, GVW13, BGG$^+$14, GVW15, GV15, BV16, AF18], 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 [Wat12] 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 [Wat12], 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 [BL15] 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 [AS17a] 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 the existence of multilinear maps [GGH13a], witness encryption [GTKP$^+$13] or indistinguishability obfuscation [BGI$^+$01, GGH$^+$13b], attribute based encryption (indeed, even its more powerful generalization – *functional encryption*) has been constructed even for Turing machines [AS17b, AM18, KNTY18], but these are not considered standard assumptions; indeed many candidate constructions have been broken [CHL$^+$15, CGH$^+$15, HJ15, CJL, CFL$^+$, MSZ16, CLLT16, ADGM16]. Very recently, Ananth and Fan [AF18] constructed ABE for RAM programs from LWE achieving decryption complexity that is sublinear in the database length. However, the key sizes in their 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, [BV16, GKW16] 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 [GKW16].

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

[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	Model	Input Length	Number of Keys	Attribute and Function Hiding	Assumption
Waters [Wat12]	DFA	unbounded	unbounded	(no, no)	q-type assumption on bilinear maps
Boyen-Li [BL15]	DFA	bounded	unbounded	(no, no)	LWE
Agrawal-Singh [AS17a]	DFA	unbounded	single	(yes, yes)	LWE
Ananth-Fan [AF18]	RAM	bounded	unbounded	(no, no)	LWE
Section 4	NFA	unbounded	unbounded	(no, no)	LWE
Appendix B	NFA	unbounded	unbounded	(yes*, no)	LWE
Appendix C	NFA	unbounded	bounded	(yes, yes)	LWE

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.

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 [AS17a], 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 (please see Appendix A.3 for the formal definition), 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.

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 [Wat12] 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

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

[4]The construction is later extended to be adaptively secure rather than selectively secure (e.g., [Att14]), but the basic structure of the construction is unchanged.

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 [BL15] 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 [AS17a] gives a construction of functional encryption for DFA from LWE that does handle unbounded length inputs, but only in the limited single key setting. Their construction crucially relies on reusable garbled circuits [GKP$^+$13] 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 [AF18] is also inherently bounded length, for reasons similar as those discussed above for [BL15].

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 [GVW13, BGG$^+$14], 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 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.

[5]For the knowledgeable reader, bounded key variants of reusable garbled circuits exist, for instance by applying the compiler of [GVW12], but using this in the aforementioned construction does not work due to the structure of their construction.

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. [BGG+14], 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 [GKP+13, Agr17]. 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. $2^{i-1} < |\mathbf{x}| \le 2^i$ 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 $j = \lambda$.

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

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 Section 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. 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 [GVW13, BGG$^+$14]. 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 Sec. 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 Appendix B and Appendix C, we show that we can indeed ensure this, sometimes (see for instance, the construction in Appendix C) by employing additional tricks as required. In Section 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 [ABSV15, KNT18]. 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 Section 2, we provide the definitions and preliminaries we require. In Section 3, we provide our ABE for NFA supporting unbounded input but bounded machine length. In Section 4, we enhance the construction to support both unbounded input and unbounded machine length. In Appendix B we leverage our ABE to construct the first predicate and bounded key functional encryption schemes for NFA. In Appendix C, we provide the first construction of reusable garbled NFA. In Section 5 we show that secret key functional encryption for DFA with security against unbounded collusion implies indistinguishability obfuscation for circuits. We conclude in Section 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 $\{k \in \mathbb{N} \mid a \le k \le b\}$. 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 $\mathsf{depth}(C[\mathbf{x}]) \le \mathsf{depth}(C) + O(1)$ 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 2.1. 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 $2 \leq |\Sigma| \leq \mathrm{poly}(\lambda)$.

- $\mathsf{NfaABE.Setup}(1^\lambda)$ 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 2.2. In our construction in Sec. 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 Sec. 4 which can support NFAs with arbitrary size.

Definition 2.3 (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 2.4 (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

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

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 2.5. As noted in Remark 2.2, our construction in Sec. 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

2.3.1 Attribute based Encryption for Circuits

For $\lambda \in \mathbb{N}$, let $\mathcal{C}_{\mathsf{inp},\mathsf{d}}$ 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 2.6 (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 2.7 (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

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

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^λ, 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 2.8. 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 Sec. 3.2, we will use the ABE scheme by Boneh et al. [BGG+14] as a building block. The following theorem summarizes the efficiency properties of their construction.

Theorem 2.9 (Adapted from [BGG+14])**.** *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* $\mathsf{depth}(C) \cdot \mathrm{poly}(\lambda, \mathsf{d})$.

Proof. We show that the construction proposed by Boneh et al. [BGG+14] satisfies the properties. We only focus on the third item of the theorem, as the first one is easy to observe and the second one is explicitly mentioned in the paper.

To give the proof, we briefly recall the setup and key generation algorithms of their scheme. The setup algorithm prepares a set of matrices $(\mathbf{A}, \mathbf{A}_1, \ldots, \mathbf{A}_{\mathsf{inp}})$ and a vector $\mathbf{u}$, whose sizes only depend on λ and d. To generate a secret key for a circuit C (without a hardwired value), the key generation algorithm homomorphically evaluates the circuit on matrices $(\mathbf{A}_1, \ldots, \mathbf{A}_{\mathsf{inp}})$ in a gate by gate manner. In more details, it assigns $\mathbf{A}_i$ to the wire corresponding to the i-th bit of the input and computes a matrix for each internal wire of the circuit. The size of the matrices will be the same for all wires. In more details, let g be a gate with incoming wires w_1 and w_2 and output wire w_3. Then, the matrix corresponding to w_3 is computed from the matrices corresponding to w_1 and w_2, where the computation applied to the matrices depends on the type of the gate g. In the end, it obtains the matrix $\mathbf{A}_C$ corresponding to the output wire. Then, it generates a short vector $\mathbf{e}$ such that $[\mathbf{A} \| \mathbf{A}_C]\mathbf{e} = \mathbf{u}$ using the trapdoor for $\mathbf{A}$ and outputs $\mathbf{e}$ as a secret key.

We first show that in the case of $\ell = 0$, or equivalently in the case where a circuit $C : \{0,1\}^{\mathsf{inp}} \to \{0,1\}$ is not hardwired any value, the statement holds. To see this, we first observe that the last operation in which short vector $\mathbf{e}$ is sampled can be implemented by a circuit with size $\mathrm{poly}(\lambda, \mathsf{d})$, since the sizes of $\mathbf{A}$, $\mathbf{A}_C$, and $\mathbf{u}$ are bounded by $\mathrm{poly}(\lambda, \mathsf{d})$. We then focus on the computational cost of homomorphic operation on matrices. We can implement the circuit that performs this step with depth $\mathsf{depth}(C) \cdot \mathrm{poly}(\lambda, \mathsf{d})$ by just replacing each gate of the circuit C with a circuit that performs the homomorphic matrix operation corresponding to this gate.

We then consider the general case where $\ell \neq 0$. In this case, we first construct a circuit that performs homomorphic operations given matrices $\mathbf{B}_1, \ldots, \mathbf{B}_\ell, \mathbf{A}_1, \ldots, \mathbf{A}_{\mathsf{inp}}$ and $C(\cdot, \cdot)$, where $\mathbf{B}_1, \ldots, \mathbf{B}_\ell$ will correspond to the hardwired value. By the above discussion, such a circuit can be implemented with depth $\mathsf{depth}(C) \cdot \mathrm{poly}(\lambda, \mathsf{d})$. It remains to show that it is possible to construct a circuit that takes as input $\mathbf{A}_1, \ldots, \mathbf{A}_{\mathsf{inp}}$ and the hardwired value v and outputs matrices $\mathbf{B}_1, \ldots, \mathbf{B}_\ell$. Such a circuit can be implemented with depth $\mathrm{poly}(\lambda, \mathsf{d})$ by computing $\mathbf{B}^{(0)}$ and $\mathbf{B}^{(1)}$ that correspond to $0 = x_1 \wedge (\neg x_1)$ and $1 = x_1 \vee (\neg x_1)$ from $\mathbf{A}_1$ and then outputting $\mathbf{B}^{(v_1)}, \cdots, \mathbf{B}^{(v_\ell)}$, where v_i is the i-th bit of v. This completes the proof of the theorem. $\square$

Remark 2.10. As we mentioned, we use ABE for circuits with the above efficiency properties to construct ABE for NFA in Sec. 3.2. Since we only have selectively secure ABE scheme satisfying the above properties, the resulting construction of ABE for NFA will only have selective security. One could consider that by applying the standard complexity leveraging argument to the selectively secure ABE scheme by Boneh et al. [BGG+14] to obtain an adaptively secure scheme and then using the resultant scheme in the construction in Sec. 3.2, we can obtain adaptively secure ABE scheme for NFA. This is not true because the resulting ABE scheme obtained by the complexity leveraging have secret keys whose size is polynomially dependent on the input length $\mathsf{inp}(\lambda)$ of the circuits and does not satisfy the second efficiency property in Theorem 2.9, which is crucial for the construction in Sec. 3.2 to work.

2.3.2 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 $\mathsf{FE.ct}$.
- $\mathsf{FE.Dec}(\mathsf{FE.mpk}, \mathsf{FE.sk}_C, C, \mathsf{FE.ct})$ is a deterministic algorithm that takes as input the master public key $\mathsf{FE.mpk}$, a secret key $\mathsf{FE.sk}_C$, corresponding circuit C, and a ciphertext $\mathsf{FE.ct}$ and outputs $C(\mathbf{x})$.

Definition 2.11 (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 $\mathbf{x} \in \{0,1\}^{\mathsf{inp}(\lambda)}$,

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

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 [GKP+13, Defn 2.13].

Definition 2.12 (FULL-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 FULL-SIM secure if there exists a PPT simulator Sim such that for every PPT adversary A, the following two distributions are computationally indistinguishable:

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

Remark 2.13. The above definition allows an adversary to make encryption queries multiple times. In the security notion defined in [GKP+13], 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 2.8). We adopt the above definition since it is convenient for our purpose.

Remark 2.14 (Selective Simulation Security.). We can consider a weaker version of the above security notion where A outputs a set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|X|}\} \subset \Sigma^*$ along with $(1^{\mathsf{inp}}, 1^{\mathsf{d}}, 1^{\mathsf{out}})$ at the beginning of the game and A is only allowed to query $\mathbf{x} \in X$ to $\mathsf{FE.Enc}(\mathsf{FE.mpk}, \cdot)$ and $\mathsf{O}(\cdot)$. We call this security notion selective simulation security.

In our construction of SKABE for NFA in Sec. 3.2, we will use the FE scheme by Goldwasser et al. [GKP$^+$13] as a building block. The following theorem summarizes the efficiency properties of their construction.

Theorem 2.15 ([GKP$^+$13])**.** *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 [AJ15, BV15].

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 3.1. *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* $\mathbf{x} \in \Sigma^{\le \ell}$*, 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* $\mathrm{poly}(\mathsf{s}, \ell)$.

Proof. We define the circuit $\mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}$ as in Figure 1. There, we introduce a circuit M_{2^j} that takes as input $\mathbf{x} \in (\Sigma \cup \{\perp\})^{2^j}$ and outputs $\{b_{q,q',\mathbf{x}}\}_{(q,q') \in Q \times Q}$, where the boolean value $b_{q,q',\mathbf{x}} \in \{0,1\}$ is set to 1 if the state q' is reachable from q by reading $\mathbf{x}$ and 0 otherwise. Here, we augment the transition function T so that it works on the extended alphabet $\Sigma \cup \{\perp\}$, where we define $T(\perp, q) = \{q\}$ for all $q \in Q$. It is easily seen that the padding with $\perp$ and the augmentation of the transition function T we introduce here do not change the value of $M(\mathbf{x})$. We refer to Figure 2 for the concrete way of constructing M_{2^j}. It is not hard to see that $\mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}$ defined as in Figure 1 satisfies Item 1 of the theorem.

Circuit $\mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}(M)$

1. Compute $b_{q,q',x}$ for all $(q,q',x) \in Q \times Q \times (\Sigma \cup \{\perp\})$ in parallel from M.
2. Then, construct the circuit M_1 from $\{b_{q,q',x}\}_{(q,q',x)\in Q\times Q\times(\Sigma\cup\{\perp\})}$, which takes $y \in (\Sigma \cup \{\perp\})$ as input, checks whether $y{=}x$ for all $x \in (\Sigma \cup \{\perp\})$ in parallel, and outputs $\{b_{q,q',x}\}_{(q,q')\in Q\times Q}$ such that $x = y$.
3. Compute M_{2^j} for $j \in [i]$ in the ascending order, where M_{2^j} is constructed from $M_{2^{j-1}}$ as in Figure 2 and $i = \lceil \log \ell \rceil$.
4. Compute $\widehat{M_\ell}$ defined as in Figure 3 from M_{2^i} and output $\widehat{M_\ell}$.

Figure 1 : The Circuit $\mathsf{To\text{-}Circuit}$.

Circuit $M_{2^j}(\mathbf{x})$

1. Parse the input $\mathbf{x} = \mathbf{x}_0 \| \mathbf{x}_1$, where $\mathbf{x}_0, \mathbf{x}_1 \in (\Sigma \cup \{\perp\})^{2^{j-1}}$.
2. Compute $M_{2^{j-1}}(\mathbf{x}_0) = \{b_{q,q',\mathbf{x}_0}\}_{(q,q')\in Q\times Q}$ and $M_{2^{j-1}}(\mathbf{x}_1) = \{b_{q,q',\mathbf{x}_1}\}_{(q,q')\in Q\times Q}$ in parallel.
3. Compute $b_{q,q',\mathbf{x}}$ for all $(q,q') \in Q \times Q$ in parallel by executing the following:
   (a) Compute $(b_{q,q'',\mathbf{x}_0} \wedge b_{q'',q',\mathbf{x}_1})$ for all $q'' \in Q$ in parallel.
   (b) Compute $b_{q,q',\mathbf{x}} := \vee_{q''\in Q}(b_{q,q'',\mathbf{x}_0} \wedge b_{q'',q',\mathbf{x}_1})$.
4. Output $\{b_{q,q',\mathbf{x}}\}_{(q,q')\in Q\times Q}$.

Figure 2 : The Circuit $M_{2^j}(\mathbf{x})$.

Circuit $\widehat{M_\ell}(\hat{\mathbf{x}})$

1. Pad the input $\hat{\mathbf{x}} \in (\Sigma \cup \{\perp\})^\ell$ to obtain $\tilde{\mathbf{x}} := \hat{\mathbf{x}} \| \perp^{2^i-\ell} \in (\Sigma \cup \{\perp\})^{2^i}$.
2. Compute $M_{2^i}(\tilde{\mathbf{x}}) = \{b_{q,q',\tilde{\mathbf{x}}}\}_{(q,q')\in Q\times Q}$.
3. Compute $b = \vee_{q\in F} b_{q_{\mathsf{st}},q,\tilde{\mathbf{x}}}$ and output b.

Figure 3 : The Circuit $\widehat{M_\ell}(\hat{\mathbf{x}})$.

To finish the proof, we have to show Item 2 of the theorem. We first bound the size of $\widehat{M_\ell}$. To do this, we first observe that $\mathsf{size}(M_1) \le \mathrm{poly}(|\Sigma|,|Q|)$ holds. Furthermore, we have

$$\mathsf{size}(M_{2^j}) \le 2 \cdot \mathsf{size}(M_{2^{j-1}}) + \mathrm{poly}(|\Sigma|,|Q|) \quad \text{and} \quad \mathsf{depth}(\widehat{M_\ell}) \le \mathsf{depth}(M_{2^i}) + \mathrm{poly}(|\Sigma|,|Q|).$$

From the above, we have

$$\mathsf{size}(\widehat{M_\ell}) \le 2^i \,\mathrm{poly}(|\Sigma|,|Q|) \le \mathrm{poly}(\mathsf{s},\ell) \tag{3.1}$$

as desired. We then bound the depth of $\widehat{M_\ell}$. We first observe $\mathsf{depth}(M_1) = \mathrm{poly}(\log|\Sigma|, \log|Q|)$. Furthermore, we have

$$\mathsf{depth}(M_{2^j}) \le \mathsf{depth}(M_{2^{j-1}}) + \mathrm{poly}(\log|\Sigma|, \log|Q|)$$

and

$$\mathsf{depth}(\widehat{M_\ell}) \le \mathsf{depth}(M_{2^i}) + \mathrm{poly}(\log|\Sigma|, \log|Q|).$$

From the above, we have

$$\mathsf{depth}(\widehat{M_\ell}) \le i \cdot \mathrm{poly}(\log|\Sigma|, \log|Q|) \le \mathrm{poly}(\log \mathsf{s}, \log \ell)$$

as desired.

We next bound the size of the circuit $\mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}(\cdot)$. It is easy to see that Step 1 and 2 of $\mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}(\cdot)$ can be implemented by circuits of size $\mathrm{poly}(|\Sigma|, |Q|)$. We also observe that j-th repetition in Step 3 can be implemented by a circuit of size $\mathrm{poly}(\mathsf{size}(M_{2^{j-1}}), |\Sigma|, |Q|) \le \mathrm{poly}(\mathsf{size}(\widehat{M_\ell}), |\Sigma|, |Q|)$. We can also see that Step 4 can be implemented by a circuit of size $\mathrm{poly}(\mathsf{size}(\widehat{M_\ell}), |\Sigma|, |Q|)$. Therefore, we have

$$\mathsf{size}(\mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}) \le i \cdot \mathrm{poly}(\mathsf{size}(\widehat{M_\ell}), |\Sigma|, |Q|) \le \mathrm{poly}(\mathsf{s}, \ell)$$

as desired, where the second inequality follows from Eq. (3.1).

We finally bound the depth of the circuit $\mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}(\cdot)$. It is easy to see that Step 1, 2, and 4 of $\mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}(\cdot)$ can be implemented by circuits of depth $\mathrm{poly}(\log|\Sigma|, \log|Q|)$. We also observe that each repetition in Step 3 can be implemented with depth $\mathrm{poly}(\log|\Sigma|, \log|Q|)$, since it just copies $M_{2^{j-1}}$ and adds a fixed circuit to it that performs Item 3 and 4 of M_{2^j}. Therefore, Step 3 of $\mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}(\cdot)$ can be implemented by a circuit of depth $i \cdot \mathrm{poly}(\log|\Sigma|, \log|Q|)$. To sum up, we have

$$\mathsf{depth}(\mathsf{To\text{-}Circuit}_{\mathsf{s},\ell}) \le i \cdot \mathrm{poly}(\log|\Sigma|, \log|Q|) \le \mathrm{poly}(\log \mathsf{s}, \log \ell)$$

as desired. This completes the proof of the theorem. □

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 2.15. We can instantiate FE with the scheme proposed by Goldwasser et al. [GKP+13].

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 2.9. We can instantiate ABE with the scheme proposed by Boneh et al. [BGG+14].

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 $\mathsf{depth}(U) \le O(\mathsf{depth}(C))$. For construction of such a universal circuit, we refer to [CH85].

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 2.9 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 all $j \in [0, \lambda]$, sample PRF keys $\widehat{\mathsf{K}}_j, \mathsf{R}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
2. For all $j \in [0, \lambda]$, sample $(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{out}(\lambda)}, 1^{\mathsf{d}(\lambda)})$.

   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 Figure 4.
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 3.1 for the definition of To-Circuit.)
4. Compute and output $\mathsf{ABE.sk}_{U[\widehat{M}_{2^i}]} = \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}, \mathsf{ABE.msk}, U[\widehat{M}_{2^i}]; \widehat{\mathsf{R}})$.

Figure 4

$\mathsf{NfaABE.KeyGen}(\mathsf{NfaABE.msk}, M, 1^{\mathsf{s}})$: 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 $\perp$ and abort. Else, proceed as follows.

1. Parse the master secret key as $\mathsf{NfaABE.msk} \to (\{\widehat{\mathsf{K}}_j, \mathsf{R}_j, \mathsf{FE.mpk}_j, \mathsf{FE.msk}_j\}_{j\in[0,\lambda]})$.
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 $y = \mathsf{FE.Dec}(\mathsf{FE.mpk}_i, \mathsf{FE.sk}_i, C_{\mathsf{s},2^i}, \mathsf{FE.ct}_i)$.
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 $\hat{\mathbf{x}} = \mathbf{x}||\perp^{2^i-\ell}$.

3.3 Correctness of NfaABE

The following theorem asserts that our scheme is efficient.

Theorem 3.2. *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.*

Proof. It is easy to see that the $\mathsf{NfaABE.Setup}$ and $\mathsf{NfaABE.KeyGen}$ run in polynomial time.

We then show that $\mathsf{NfaABE.Enc}$ runs in polynomial time. By the efficiency of ABE, it suffices to show that $C_{\mathsf{s},2^i}$ can be computed in polynomial time. To see this, we bound the time for constructing each step of the circuit. Trivially, Step 1 can be implemented with no cost. We first observe that Step 2 of the circuit can be implemented by a circuit with size $\mathrm{poly}(\lambda, 2^i\eta, \hat{\mathsf{d}}) = \mathrm{poly}(\lambda, |\mathbf{x}|)$ by the efficiency of $\mathsf{ABE.KeyGen}$. We then observe that Step 3 of the circuit can be implemented with size $\mathrm{poly}(\mathsf{s}, 2^i) \leq \mathrm{poly}(\mathsf{s}, |\mathbf{x}|)$ by Item 2 of Theorem 3.1. Finally, Step 4 of the circuit can be implemented with size $\mathrm{poly}(\lambda, 2^i\eta, \hat{\mathsf{d}}, \mathsf{s}) \leq \mathrm{poly}(\lambda, \mathsf{s}, |\mathbf{x}|)$ by the efficiency of the universal circuit and ABE and Item 2 of Theorem 3.1.

We finally bound the running time of $\mathsf{NfaABE.Dec}$. Trivially, Step 1 and 2 of $\mathsf{NfaABE.Dec}$ can be implemented with no cost. By the efficiency of FE, the running time of Step 3 is also bounded by $\mathrm{poly}(\lambda, \mathsf{s}, |\mathbf{x}|)$. Finally, to bound the running time of Step 4, it suffices to bound the time for constructing $U[\widehat{M}_{\mathsf{s},2^i}]$ by the efficiency of ABE. Since $\widehat{M}_{\mathsf{s},2^i}$ can be constructed in time $\mathrm{poly}(\mathsf{s}, 2^i) \leq \mathrm{poly}(\mathsf{s}, |\mathbf{x}|)$ by Item 2 of Theorem 3.1, so is $U[\widehat{M}_{\mathsf{s},2^i}]$. This completes the proof of the theorem. □

The following theorem addresses the correctness of the scheme.

Theorem 3.3. *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 4 of the decryption algorithm. To show this, let us set $\hat{\mathsf{d}}(\lambda) = \Omega(\lambda)$ and assume that

$$y = \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i, U[\widehat{M}_{2^i}]; \widehat{\mathsf{R}}_i) \tag{3.2}$$

holds for the moment, where y is the value retrieved in Step 3 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 3.1. 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

$$\begin{aligned} \mathsf{depth}(U[\widehat{M}_{2^i}]) &\leq \mathsf{depth}(U(\cdot,\cdot)) + O(1) \\ &\leq O(1) \cdot \mathsf{depth}(\widehat{M}_{2^i}) + O(1) \\ &\leq \mathrm{poly}(\log \mathsf{s}, \log 2^i) \\ &\leq \mathrm{poly}(\log \lambda) \\ &\leq \hat{\mathsf{d}} \end{aligned} \tag{3.3}$$

holds, where the second inequality follows from the property of the depth preserving universal circuit U and the third from Item 2 of Theorem 3.1.

It remains to prove that Eq. (3.2) 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. (3.2) 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 2.9. 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 3.1. 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 2.9 and the second from Eq. (3.3). 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 2.9. 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 3.4. *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 E_{xxx} denote the probability that A outputs 1 in $\mathbf{Game}_{\text{xxx}}$. It suffices to prove $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_{\lambda+1}]| = \text{negl}(\lambda)$, 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 $|\Pr[\mathsf{E}_i] - \Pr[\mathsf{E}_{i+1}]| = \text{negl}(\lambda)$ 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 $|\Pr[\mathsf{E}_i] - \Pr[\mathsf{E}_{i+1}]| = \text{negl}(\lambda)$ 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

$$\mathsf{FE.ct}_j \leftarrow \begin{cases} \mathsf{FE.Enc}(\mathsf{FE.mpk}_j, (M, \widehat{\mathsf{K}}_j, \widehat{\mathsf{R}}_j)) & \text{If } j \in [0, \lambda] \backslash \{i\} \\ \mathrm{Sim}(\mathsf{FE.mpk}_i, \mathsf{FE.sk}_i, C_{\mathsf{s},2^i}, \mathsf{ABE.sk}_{U[\widehat{M}_{2^i}]}, 1^{\mathsf{inp}(\lambda)}) & \text{If } j = i. \end{cases} \tag{3.4}$$

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.

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

Since we have

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

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 3.5. *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 3.6. *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 $\mathbf{Game}_{i,1}$. Therefore, the lemma follows by a straightforward reduction to the security of PRF. □

Lemma 3.7. *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

$$\mathsf{FE.ct} \leftarrow \begin{cases} \mathsf{FE.Enc}(\mathsf{FE.mpk}, (M, \widehat{\mathsf{K}}_i, \widehat{\mathsf{R}}_i)) & \text{If } \mathsf{B} \text{ is in } \mathsf{Exp}^{\mathsf{real}}_{\mathsf{FE,B}}(1^\lambda) \\ \mathrm{Sim}(\mathsf{FE.mpk}, \mathsf{FE.sk}, C_{\mathsf{s},2^i}, C_{\mathsf{s},2^i}(M, \widehat{\mathsf{K}}_i, \widehat{\mathsf{R}}_i), 1^{\mathsf{inp}(\lambda)}) & \text{If } \mathsf{B} \text{ is in } \mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{FE},\mathrm{Sim}}(1^\lambda) \end{cases} \quad (3.5)$$

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. (3.4) while in the simulation above, it is generated as Eq. (3.5) (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. $\square$

Lemma 3.8. *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. $\square$

Lemma 3.9. *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 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 $i' \ne i$, 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. (3.4) 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 3.1. Therefore, B is a legitimate adversary as long as so is A. This completes the proof of the lemma. □

Lemma 3.10. *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 3.4.

□

3.5 Extensions

In Appendix B, 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. In Appendix D, we additionally achieve machine privacy, improving the result of [AS17a]. 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 2.9) are not violated. Please see appendices B,D 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 Section 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 2.9. We can instantiate ABE with the scheme proposed by Boneh et al. [BGG+14].
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 $\mathsf{K}_{\mathsf{NfaABE}} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$, $\mathsf{K}_{\mathsf{ABE}} \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
2. Output $\mathsf{uNfaABE.msk} = (\mathsf{K}_{\mathsf{NfaABE}}, \mathsf{K}_{\mathsf{ABE}})$.

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

1. Parse the master secret key as $\mathsf{uNfaABE.msk} = (\mathsf{K}_{\mathsf{NfaABE}}, \mathsf{K}_{\mathsf{ABE}})$. 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 $r_i = \mathsf{PRF.Eval}(\mathsf{K}_{\mathsf{NfaABE}}, i)$.

   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 $r_\ell = \mathsf{PRF.Eval}(\mathsf{K}_{\mathsf{ABE}}, \ell)$.

   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 $M = (Q, \Sigma, T, q_{\mathsf{st}}, F)$, proceed as follows.

1. Parse the master secret key as $\mathsf{uNfaABE.msk} = (\mathsf{K}_{\mathsf{NfaABE}}, \mathsf{K}_{\mathsf{ABE}})$. 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 3.1 for the definition of To-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 $r_i = \mathsf{PRF.Eval}(\mathsf{K}_{\mathsf{ABE}}, i)$.

(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 $r_\mathsf{s} = \mathsf{PRF.Eval}(\mathsf{K}_{\mathsf{NfaABE}}, \mathsf{s})$.

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 $\widehat{M}_{|\mathbf{x}|} = \mathsf{To\text{-}Circuit}_{|M|,|\mathbf{x}|}(M)$.

4.1 Correctness of uNfaABE

The following theorem asserts that our scheme is efficient.

Theorem 4.1. *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. .

Proof. It is easy to see that the $\mathsf{uNfaABE.Setup}$ runs in polynomial time.

We start with showing that $\mathsf{uNfaABE.KeyGen}$ runs in polynomial time. Note that for any NFA M with size $\mathsf{s} = |M|$ and any input length $i < [\mathsf{s}]$, Item 2 of Theorem 3.1 asserts that the sizes and depths of $\mathsf{To\text{-}Circuit}_{\mathsf{s},i}$ and $\widehat{M_i} = \mathsf{To\text{-}Circuit}_{\mathsf{s},i}(M)$ are both bounded by $\mathrm{poly}(\mathsf{s}, i)$ and $\mathrm{poly}(\log \mathsf{s}, \log i)$ respectively. Hence, the efficiency requirements met by [BGG+14] discussed in Theorem 2.9 which is used to instantiate ABE implies that $\mathsf{ABE.Setup}$ and $\mathsf{ABE.KeyGen}$ always runs in time polynomial in λ, s and $\hat{\mathsf{d}}$. Further, the efficiency of NfaABE discussed in Theorem 3.2 which is used to instantiate NfaABE implies that $\mathsf{NfaABE.Setup}$ and $\mathsf{NfaABE.KeyGen}$ runs in polynomial time.

Next, we show that $\mathsf{uNfaABE.Enc}$ runs in polynomial time. We have a similar reasoning as above to argue the following: for any input $\mathbf{x}$ with length $\ell = |\mathbf{x}|$ and any NFA size $i \in [\ell]$, the efficiency of NfaABE discussed in Theorem 3.2 implies that $\mathsf{NfaABE.Setup}$ and $\mathsf{NfaABE.Enc}$ run in polynomial time.

We finally bound the running time of $\mathsf{uNfaABE.Dec}$. It is easy to see that the efficiency guarantees given by the underlying schemes NfaABE and ABE along with Theorem 3.1 implies that $\mathsf{uNfaABE.Dec}$ runs in polynomial time. □

The following theorem addresses the correctness of the scheme.

Theorem 4.2. *For appropriately chosen* $\hat{\mathsf{d}} = \hat{\mathsf{d}}(\lambda)$*, our scheme* $\mathsf{uNfaABE}$ *is correct for any NFA.*

Proof. As long as $\hat{\mathsf{d}}$ is chosen appropriately, we can argue the correctness of uNfaABE as follows. The uNfaABE.Dec algorithm takes as input

$$\begin{aligned}\mathsf{uNfaABE.sk}_M &= \big(\mathsf{NfaABE.sk}_{|M|}, \{\mathsf{ABE.mpk}_i, \mathsf{ABE.sk}_i\}_{i\in[|M|]}\big) \text{ and}\\ \mathsf{uNfaABE.ct} &= \big(\{\mathsf{NfaABE.ct}_i\}_{i\in[|\mathbf{x}|]}, \mathsf{ABE.mpk}_{|\mathbf{x}|}, \mathsf{ABE.ct}_{|\mathbf{x}|}\big)\end{aligned}$$

as a secret key for an NFA M and a ciphertext under an attribute $\mathbf{x}$ respectively.

By our definition, for any $i \in \mathbb{N}$, $\mathsf{NfaABE.msk}_i$ supports NFA machines of size bounded above by i with unbounded input length. Thus, in the case when $|\mathbf{x}| \geq |M|$, the correctness of NfaABE scheme implies the correctness of the uNfaABE scheme.

Alternatively, we have by Theorem 2.9 that for any $i \in \mathbb{N}$, $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i)$ supports a circuit class of input length i and depth $\hat{\mathsf{d}} = \hat{\mathsf{d}}(\lambda)$ while Theorem 3.1 implies that $\mathsf{depth}(\widehat{M_i})$ is always bounded above by $\mathrm{poly}(\log i, \log \mathsf{s}) = \mathrm{poly}(\log\lambda) < \lambda$. In particular, we have that $\mathsf{depth}(\widehat{M_i}) < \hat{\mathsf{d}}$, where we set $\hat{\mathsf{d}} = \lambda$ and thus the circuit $\widehat{M_i}$ is supported by the ABE scheme. Therefore, in the case when $|\mathbf{x}| < |M|$, the correctness of ABE scheme implies the correctness of the uNfaABE scheme. $\square$

4.2 Proof of Security for uNfaABE

Next, we prove that the above uNfaABE scheme is secure, as long as the underlying NfaABE and ABE schemes are secure.

Theorem 4.3. *Assume that* NfaABE *and* ABE *both satisfy selective indistinguishability based security and* PRF *is a secure pseudorandom function. Then,* uNfaABE *satisfies selective security.*

Proof. To show that any PPT adversary A succeeds with only negligible probability in the selective security game of uNfaABE as stated in Definition 2.4, let us introduce the following sequence of games $\{\mathbf{Game}_k\}_{k\in[0,6]}$ between the uNfaABE challenger and A.

$\mathbf{Game}_0$**:** For any challenge message query $(m^{(0)}, m^{(1)})$ with respect to any attribute $\mathbf{x}$, this game corresponds to the real experiment where the uNfaABE challenger encrypts messages corresponding to challenge bit $b = 0$.

$\mathbf{Game}_1$**:** In this game, we change the setup phase as follows.

Setup phase. Given $X \subset \Sigma^*$ from A, the challenger chooses $\mathsf{uNfaABE.msk} \leftarrow \mathsf{NfaABE.Setup}(1^\lambda)$ as in the previous game. In addition, it precomputes all relevant master keys as follows which are later used to answer encryption as well as key queries.

$$\begin{aligned}\mathsf{NfaABE.msk}_i &\leftarrow \mathsf{NfaABE.Setup}(1^\lambda, 1^i; r_i), \text{ where } r_i = \mathsf{PRF.Eval}(\mathsf{K}_{\mathsf{NfaABE}}, i)\\ (\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) &\leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^i, 1^{\hat{\mathsf{d}}}; r'_i), \text{ where } r'_i = \mathsf{PRF.Eval}(\mathsf{K}_{\mathsf{ABE}}, i)\end{aligned}$$

for all $i \in [T], T = \max(\ell_{\max}, \mathsf{s}_{\max})$, where we define $\ell_{\max} := \max\{|\mathbf{x}| : \mathbf{x} \in X\}$ as per Definition 2.4 and $\mathsf{s}_{\max}$ as the maximum size of any NFA queried by A. Note that both $\ell_{\max}$ and $\mathsf{s}_{\max}$ and hence T are bounded by the running time of A and thus are polynomial in λ.

$\mathbf{Game}_2$**:** This game is exactly the same as $\mathbf{Game}_1$ except that now the challenger samples the master keys in setup phase for *both* the underlying schemes NfaABE and ABE using true randomness instead of using PRF randomness as

$$\begin{aligned}\{\mathsf{NfaABE.msk}_i &\leftarrow \mathsf{NfaABE.Setup}(1^\lambda, 1^i)\}_{i\in[T]},\\ \{(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) &\leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^i, 1^{\hat{\mathsf{d}}})\}_{i\in[T]}\end{aligned}$$

Game$_3$: In this game, for any challenge message query $(m^{(0)}, m^{(1)})$ with respect to any attribute $\mathbf{x}$, the challenger generates NfaABE ciphertexts corresponding to challenge bit $b = 1$ while the ABE ciphertexts are generated corresponding to challenge bit $b = 0$.

Game$_4$: In this game, for any challenge message query $(m^{(0)}, m^{(1)})$ with respect to any attribute $\mathbf{x}$, the challenger generates both NfaABE and ABE ciphertexts corresponding to challenge bit $b = 1$.

Game$_5$: This game is exactly the same as **Game$_4$** except that now the challenger samples the master keys in the setup phase for *both* the underlying schemes NfaABE and ABE using PRF randomness again instead of using true randomness.

Game$_6$: For any challenge message query $(m^{(0)}, m^{(1)})$ with respect to any attribute $\mathbf{x}$, this game corresponds to the real experiment where the $\mathsf{uNfaABE}$ challenger encrypts messages corresponding to challenge bit $b = 1$.

In the following, let E_k denote the event that A outputs 1 in **Game$_k$**. To prove the theorem, we will show that $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_6]| = \mathrm{negl}(\lambda)$, since **Game$_0$** (resp., **Game$_6$**) corresponds to the selective security game with $b = 0$ (resp., $b = 1$). Since we have

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_6]| \leq \sum_{k\in[0,5]} |\Pr[\mathsf{E}_k] - \Pr[\mathsf{E}_{k+1}]|$$

by the triangle inequality, it suffices to show $|\Pr[\mathsf{E}_k] - \Pr[\mathsf{E}_{k+1}]| = \mathrm{negl}(\lambda)$ for all $k \in [0, 5]$. In order to prove indistinguishability between **Game$_2$** and **Game$_3$** (resp., **Game$_3$** and **Game$_4$**), we further introduce a sequence of games $\{\mathbf{Game}_{2,i}\}_{i\in[0,\ell_{\max}]}$ (resp., $\{\mathbf{Game}_{3,i}\}_{i\in[0,\ell_{\max}]}$), where $\ell_{\max}$ was defined in **Game$_1$**. We then describe the two games **Game$_{2,i}$** and **Game$_{3,i}$** for any $i \in [0, \ell_{\max}]$ as follows.

Game$_{2,i}$: The game proceeds as follows.

Setup phase. At the beginning of the game, A takes 1^λ as input and submits the set of its target $X \subset \Sigma^*$ to the $\mathsf{uNfaABE}$ challenger. The challenger then generates the master keys as before.

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

Encryption queries. For any message pair $(m^{(0)}, m^{(1)})$ and $\mathbf{x} \in X$ from A, the challenger sets $\ell = |\mathbf{x}|$, chooses the appropriate NfaABE and ABE keys from the set of precomputed keys and computes

$$\mathsf{NfaABE.ct}_j \leftarrow \begin{cases} \mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}_j, \mathbf{x}, m^{(0)}, 1^j) & \text{If } j > i \\ \mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}_j, \mathbf{x}, m^{(1)}, 1^j) & \text{If } j \leq i \end{cases}$$

for $j \in [\ell]$. It also computes $\mathsf{ABE.ct}_\ell \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_\ell, \mathbf{x}, m^{(0)})$ and returns $\mathsf{uNfaABE.ct} \leftarrow (\{\mathsf{NfaABE.ct}_j\}_{j\in[\ell]}, \mathsf{ABE.mpk}_\ell, \mathsf{ABE.ct}_\ell)$ to A.

Key queries. Given an NFA M from A, the challenger sets $\mathsf{s} = |M|$ and then uses the precomputed master keys to compute and returns $\mathsf{uNfaABE.sk}_M = (\mathsf{NfaABE.sk}_\mathsf{s}, \{\mathsf{ABE.mpk}_j, \mathsf{ABE.sk}_j\}_{j\in[\mathsf{s}]})$ to A.

Finally, A outputs its guess b'.

Game$_{3,i}$: The game proceeds as follows.

Setup phase. At the beginning of the game, A takes 1^λ as input and submits the set of its target $X \subset \Sigma^*$ to the $\mathsf{uNfaABE}$ challenger. The challenger then generates the master keys as before.

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

Encryption queries. For any message pair $(m^{(0)}, m^{(1)})$ and $\mathbf{x} \in X$ from A, the challenger sets $\ell = |\mathbf{x}|$, chooses the appropriate NfaABE and ABE from the set of precomputed keys and computes

$$\mathsf{ABE.ct}_\ell \leftarrow \begin{cases} \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_\ell, \mathbf{x}, m^{(0)}) & \text{If } \ell > i \\ \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_\ell, \mathbf{x}, m^{(1)}) & \text{If } \ell \leq i. \end{cases}$$

It also computes $\{\mathsf{NfaABE.ct}_j \leftarrow \mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}_j, \mathbf{x}, m^{(1)}, 1^j)\}_{j\in[\ell]}$ and returns $\mathsf{uNfaABE.ct} = (\{\mathsf{NfaABE.ct}_j\}_{j\in[\ell]}, \mathsf{ABE.mpk}_\ell, \mathsf{ABE.ct}_\ell)$ to A.

Key queries. Given an NFA M from A, the challenger sets $\mathsf{s} = |M|$ and then uses the precomputed master keys to compute and returns $\mathsf{uNfaABE.sk}_M = (\mathsf{NfaABE.sk_s}, \{\mathsf{ABE.mpk}_j, \mathsf{ABE.sk}_j\}_{j\in[\mathsf{s}]})$ to A.

Finally, A outputs its guess b'.

We first note that $\forall k \in [2,3]$, $\mathbf{Game}_k = \mathbf{Game}_{k,0}$ and $\mathbf{Game}_{k,\ell_{\max}} = \mathbf{Game}_{k+1}$. Since we have

$$|\Pr[\mathsf{E}_k] - \Pr[\mathsf{E}_{k+1}]| = |\Pr[\mathsf{E}_{k,0}] - \Pr[\mathsf{E}_{k,\ell_{\max}}]| \leq \sum_{i=0}^{(\ell_{\max}-1)} |\Pr[\mathsf{E}_{k,i}] - \Pr[\mathsf{E}_{k,i+1}]|$$

by the triangle inequality, it suffices to show $|\Pr[\mathsf{E}_{k,i}] - \Pr[\mathsf{E}_{k,i+1}]| = \mathrm{negl}(\lambda)$, for all $k \in [2,3]$ and for all $i \in [0, \ell_{\max} - 1]$. To complete the proof of the theorem, it remains to prove the following lemmas.

Lemma 4.4. *We have* $\Pr[\mathsf{E}_0] = \Pr[\mathsf{E}_1]$.

Proof. The change introduced here is only conceptual, where all the master keys $\{\mathsf{NfaABE.msk}_i\}_{i\in[T]}$ and $\{(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i)\}_{i\in[T]}$ are computed a-priori. The lemma trivially follows. □

Lemma 4.5. *We have* $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| = \mathrm{negl}(\lambda)$.

Proof. Note that the PRF keys $\mathsf{K}_{\mathsf{NfaABE}}$ and $\mathsf{K}_{\mathsf{ABE}}$ are only used for generating randomness in order to precompute the NfaABE and ABE master keys and are not used anywhere else in $\mathbf{Game}_1$. Further, note that since the two PRF keys $\mathsf{K}_{\mathsf{NfaABE}}$ and $\mathsf{K}_{\mathsf{ABE}}$ are *independent* of each other we can replace them at once with true randomness in $\mathbf{Game}_2$. Therefore, the lemma follows by a straightforward reduction to the security of PRF with respect to the two independently drawn keys $\mathsf{K}_{\mathsf{NfaABE}}$ and $\mathsf{K}_{\mathsf{ABE}}$. □

Lemma 4.6. *We have* $|\Pr[\mathsf{E}_{2,i}] - \Pr[\mathsf{E}_{2,i+1}]| = \mathrm{negl}(\lambda), \forall i \in [0, \ell_{\max} - 1]$.

Proof. To prove the lemma, let us assume that $|\Pr[\mathsf{E}_{2,i}] - \Pr[\mathsf{E}_{2,i+1}]|$ is non-negligible and construct an adversary B that breaks the selective security of NfaABE using A. B proceeds as follows.

Setup phase. At the beginning of the game, B inputs 1^λ to A and obtains $X \subset \Sigma^*$ from A. Then, B submits $(1^\lambda, 1^i)$ and X_i to the NfaABE challenger where $X_i := \{\mathbf{x} \mid \mathbf{x} \in X \text{ and } |\mathbf{x}| = i\}$ and precomputes $\{(\mathsf{ABE.mpk}_j, \mathsf{ABE.msk}_j)\}_{j\in[T]}$ and $\{\mathsf{NfaABE.msk}_j\}_{j\in[T]\setminus\{i\}}$. It then implicitly sets $\mathsf{NfaABE.msk}_i := \mathsf{NfaABE.msk}$ without knowing its value, where $\mathsf{NfaABE.msk}$ is chosen by the NfaABE challenger and further handles the encryption and key queries as follows.

Encryption queries. Given a message pair $(m^{(0)}, m^{(1)})$ and $\mathbf{x} \in X$ from A, B sets $\ell = |\mathbf{x}|$, chooses $(\mathsf{ABE.mpk}_\ell, \mathsf{ABE.msk}_\ell)$ and $\{\mathsf{NfaABE.msk}_j\}_{j\in[\ell]\setminus\{i\}}$ from the set of precomputed ABE and NfaABE keys to answer any encryption query as follows.

1. B computes $\{\mathsf{NfaABE.ct}_j \leftarrow \mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}_j, \mathbf{x}, m^{(0)}, 1^j)\}_{j\in[i+1,\ell]}$.
2. B makes an encryption query $\big((m^{(0)}, m^{(1)}), \mathbf{x}\big)$ to the NfaABE challenger. The challenger computes

$$\mathsf{NfaABE.ct} \leftarrow \begin{cases} \mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}, \mathbf{x}, m^{(0)}, 1^i) & \text{If } b = 0 \\ \mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}, \mathbf{x}, m^{(1)}, 1^i) & \text{If } b = 1 \end{cases}$$

   and returns NfaABE.ct to B. B then sets $\mathsf{NfaABE.ct}_i := \mathsf{NfaABE.ct}$.
3. B computes $\{\mathsf{NfaABE.ct}_j \leftarrow \mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}_j, \mathbf{x}, m^{(1)}, 1^j)\}_{j\in[1,i-1]}$.

B also computes by itself $\mathsf{ABE.ct}_\ell \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_\ell, \mathbf{x}, m^{(0)})$ and returns $\mathsf{uNfaABE.ct} = (\{\mathsf{NfaABE.ct}_j\}_{j\in[\ell]}, \mathsf{ABE.mpk}_\ell, \mathsf{ABE.ct}_\ell)$ to A.

Key queries. Given an NFA M from A, B first sets $\mathsf{s} = |M|$, computes $\{\widehat{M_j} = \mathsf{To\text{-}Circuit}_{\mathsf{s},j}(M)\}_{j\in[\mathsf{s}]}$ and then chooses $\{(\mathsf{ABE.mpk}_j, \mathsf{ABE.msk}_j)\}_{j\in[\mathsf{s}]}$ from the set of precomputed ABE keys. It then proceeds as follows.

1. If $\mathsf{s} \neq i$, B chooses $\mathsf{NfaABE.msk}_\mathsf{s}$ from the set of precomputed NfaABE keys to compute $\mathsf{NfaABE.sk}_\mathsf{s} \leftarrow \mathsf{NfaABE.KeyGen}(\mathsf{NfaABE.msk}_\mathsf{s}, M)$.
2. If $\mathsf{s} = i$, B submits M to the NfaABE challenger upon which the challenger computes and returns $\mathsf{NfaABE.sk} \leftarrow \mathsf{NfaABE.KeyGen}(\mathsf{NfaABE.msk}, M)$. B then sets $\mathsf{NfaABE.sk}_\mathsf{s} := \mathsf{NfaABE.sk}$.

B also computes by itself $\{\mathsf{ABE.sk}_j \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}_j, \mathsf{ABE.msk}_j, \widehat{M_j})\}_{j\in[\mathsf{s}]}$ and returns $\mathsf{uNfaABE.sk}_M = \big(\mathsf{NfaABE.sk}_\mathsf{s}, \{\mathsf{ABE.mpk}_j, \mathsf{ABE.sk}_j\}_{j\in[\mathsf{s}]}\big)$ to A.

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

It is easy to see that B simulates $\mathbf{Game}_{2,i}$ if $b = 0$ and $\mathbf{Game}_{2,i+1}$ if $b = 1$. Therefore, B breaks the security of NfaABE 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 $\mathbf{x}$ for which B makes an encryption query and for any key query for an NFA M, we have that $M(\mathbf{x}) = 0$, by the legitimacy of adversary A.

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

Lemma 4.7. *We have* $|\Pr[\mathsf{E}_{3,i}] - \Pr[\mathsf{E}_{3,i+1}]| = \mathrm{negl}(\lambda), \forall i \in [0, \ell_{\max} - 1]$.

Proof. To prove the lemma, let us assume that $|\Pr[\mathsf{E}_{3,i}] - \Pr[\mathsf{E}_{3,i+1}]|$ 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 $X \subset \Sigma^*$. Then, B submits $(1^\lambda, 1^i, 1^{\hat{\mathsf{d}}})$ and X_i to the ABE challenger, where $X_i := \{\mathbf{x} \mid \mathbf{x} \in X \text{ and } |\mathbf{x}| = i\}$. The ABE challenger chooses (ABE.mpk, ABE.msk) and replies to B with ABE.mpk upon which B precomputes $\{(\mathsf{ABE.mpk}_j, \mathsf{ABE.msk}_j)\}_{j\in[T]\setminus\{i\}}$ and $\{\mathsf{NfaABE.msk}_j\}_{j\in[T]}$ and then implicitly sets $(\mathsf{ABE.mpk}_i, \mathsf{ABE.msk}_i) := (\mathsf{ABE.mpk}, \mathsf{ABE.msk})$ without knowing the value of ABE.msk. It then handles the encryption and key queries as follows.

Encryption queries. Given a message pair $(m^{(0)}, m^{(1)})$ and $\mathbf{x} \in X$ from A, B sets $\ell = |\mathbf{x}|$, chooses $\{\mathsf{NfaABE.msk}_j\}_{j \in [\ell]}$ and $\mathsf{ABE.mpk}_\ell$ from the set of precomputed NfaABE and ABE keys to answer any encryption query as follows.

1. If $\ell > i$, B computes $\mathsf{ABE.ct}_\ell \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_\ell, \mathbf{x}, m^{(0)})$.
2. If $\ell = i$, B makes a challenge ciphertext query $\big((m^{(0)}, m^{(1)}), \mathbf{x}\big)$ to the ABE challenger. The challenger computes

$$\mathsf{ABE.ct} \leftarrow \begin{cases} \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_\ell, \mathbf{x}, m^{(0)}) & \text{If } b = 0 \\ \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_\ell, \mathbf{x}, m^{(1)}) & \text{If } b = 1 \end{cases}$$

   and returns $\mathsf{ABE.ct}$ to B. B then sets $\mathsf{ABE.ct}_\ell := \mathsf{ABE.ct}$.
3. If $\ell < i$, B computes $\mathsf{ABE.ct}_\ell \leftarrow \mathsf{ABE.Enc}(\mathsf{ABE.mpk}_\ell, \mathbf{x}, m^{(1)})$.

B also computes $\{\mathsf{NfaABE.ct}_j \leftarrow \mathsf{NfaABE.Enc}(\mathsf{NfaABE.msk}_j, \mathbf{x}, m^{(1)}, 1^j)\}_{j \in [\ell]}$ and returns to A $\mathsf{uNfaABE.ct} = (\{\mathsf{NfaABE.ct}_j\}_{j \in [\ell]}, \mathsf{ABE.mpk}_\ell, \mathsf{ABE.ct}_\ell)$.

Key queries. Given an NFA M from A, B first sets $\mathsf{s} = |M|$, computes $\{\widehat{M_j} = \mathsf{To\text{-}Circuit}_{\mathsf{s},j}(M)\}_{j \in [\mathsf{s}]}$ and then chooses $\mathsf{NfaABE.msk}_\mathsf{s}$ from the set of precomputed NfaABE keys. It then proceeds as follows.

1. B chooses $\{(\mathsf{ABE.mpk}_j, \mathsf{ABE.msk}_j)\}_{j \in [\mathsf{s}] \setminus \{i\}}$ from the set of precomputed ABE keys to compute $\{\mathsf{ABE.sk}_j \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}_j, \mathsf{ABE.msk}_j, \widehat{M_j})\}_{j \in [\mathsf{s}] \setminus \{i\}}$.
2. B submits $\widehat{M_i}$ to the ABE challenger upon which the challenger computes and returns $\mathsf{ABE.sk} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{ABE.mpk}, \mathsf{ABE.msk}, \widehat{M_i})$. B then sets $\mathsf{ABE.sk}_i := \mathsf{ABE.sk}$.

B also computes by itself $\mathsf{NfaABE.sk}_\mathsf{s} \leftarrow \mathsf{NfaABE.KeyGen}(\mathsf{NfaABE.msk}_\mathsf{s}, M)$ and returns to A $\mathsf{uNfaABE.sk}_M = \big(\mathsf{NfaABE.sk}_\mathsf{s}, \{\mathsf{ABE.mpk}_j, \mathsf{ABE.sk}_j\}_{j \in [\mathsf{s}]}\big)$.

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

It is easy to see that B simulates $\mathbf{Game}_{3,i}$ if $b = 0$ and $\mathbf{Game}_{3,i+1}$ 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). Note that any key query issued by B is a circuit $\widehat{M_i}$ against a key query for an NFA M issued by A. Further, Theorem 3.1 implies that for any NFA $M, M(\mathbf{x}) = \widehat{M_i}(\mathbf{x}), \forall \mathbf{x} \in \Sigma^{\leq i}$. Thus, for any attribute $\mathbf{x}$ of length i for which B makes an encryption query and for any NFA key query M issued by A, we have $\widehat{M_i}(\mathbf{x}) = M(\mathbf{x}) = 0$, by the legitimacy of A.

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

Lemma 4.8. *We have* $|\Pr[\mathsf{E}_4] - \Pr[\mathsf{E}_5]| = \mathrm{negl}(\lambda)$.

Proof. The proof follows similarly to Lemma 4.5. □

Lemma 4.9. *We have* $\Pr[\mathsf{E}_5] = \Pr[\mathsf{E}_6]$.

Proof. The proof follows similarly to Lemma 4.4. □

This concludes the proof of Theorem 4.3. □

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 [GVW13, GGH+13c, GGH+13b, GVW15, GVW12, AR17, GKP+13, GVW15, Agr17, ABCP15, ALS16, Lin17, BCFG17, AJ15, BV15, AJS15, Lin16, LV16, Lin17, AS17a, LT17, AJS18, LM18, Agr18].

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 [BV14].

Definition 5.1 (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

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

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 5.2 (Barrington's Theorem [Bar89])**.** *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 and prove the following theorem.

Theorem 5.3. *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.

Proof. We define Encode and ToDFA as follows:

- $\mathsf{Encode}(\mathbf{x})$ outputs $\mathbf{y} := (\mathbf{x}\|0)^L$, where $(\mathbf{x}\|0)^L$ denotes the L-times repetition of the same string $\mathbf{x}\|0$ and $L = 4^d$.
- $\mathsf{ToDFA}(C)$ first converts C into a branching program $\mathsf{BP} = \{(\mathsf{var}(t), \sigma_{t,0}, \sigma_{t,1})\}_{t\in[L]}$ of length $L = 4^d$ by using the algorithm in Theorem 5.2. It then outputs DFA $M = (Q, \Sigma, T, q_{\mathsf{st}}, F)$, where

$$Q = \{q_{i,j}^{(k)}\}_{i\in[L],j\in[0,\ell],k\in[5]} \cup \{q_{\mathsf{ed}}^{(k)}\}_{k\in[5]}, \quad \Sigma = \{0,1\}, \quad q_{\mathsf{st}} = q_{1,0}^{(1)}, \quad F = \{q_{\mathsf{ed}}^{(1)}\},$$

and

$$T(b, q_{i,j}^{(k)}) = \begin{cases} q_{i,\mathsf{var}(i)}^{\sigma_{i,b}(k)} & \text{if } j = \mathsf{var}(i) - 1 \\ q_{i+1,0}^{(k)} & \text{if } j = \ell \wedge i \neq L \\ q_{\mathsf{ed}}^{(k)} & \text{if } j = \ell \wedge i = L \\ q_{i,j+1}^{(k)} & \text{if } j \neq \mathsf{var}(i) - 1,\ j \neq \ell, \end{cases} \qquad T(b, q_{\mathsf{ed}}^{(k)}) = q_{\mathsf{ed}}^{(k)}.$$

In the following, we often denote $q_{L+1,0}^{(k)} := q_{\mathsf{ed}}^{(k)}$ for notational convenience.

It is easy to see that the running time of the algorithms is bounded by $\mathrm{poly}(\ell, 2^d)$. We then prove $M(\mathbf{y}) = C(\mathbf{x})$. Since we have $C(\mathbf{x}) = \mathsf{BP}(\mathbf{x})$ by Theorem 5.2, it suffices to show $\mathsf{BP}(\mathbf{x}) = M(\mathbf{y})$. To show this, we prove the following claim.

Claim 5.4. *Let t be an integer $t \in [0, L]$. Then, state q_{st} goes to state $q_{t+1,0}^{\zeta_t}$ on input $(\mathbf{x}\|0)^t$, where ζ_t is defined as Eq. (5.1).*

Proof. The proof is by induction on t. For the base case of $t = 0$, the statement holds because $q_{\mathsf{st}} = q_{1,0}^{(1)}$ and $\zeta_0 = 1$. Next, we prove the statement for $t = t^*$ assuming that the statement for $t = t^* - 1$. By the assumption, state q_{st} goes to state $q_{t^*,0}^{\zeta_{t^*-1}}$ on input $(\mathbf{x}\|0)^{t^*-1}$. It suffices to prove that state $q_{t^*,0}^{\zeta_{t^*-1}}$ goes to state $q_{t^*+1,0}^{\zeta_{t^*}}$ on input $\mathbf{x}\|0$. By inspection, it can be seen that state $q_{t^*,0}^{\zeta_{t^*-1}}$ goes to state $q_{t^*,\mathsf{var}(t^*)-1}^{\zeta_{t^*-1}}$ on input $x_1, \ldots, x_{\mathsf{var}(t^*)-1}$. We also observe that state $q_{t^*,\mathsf{var}(t^*)-1}^{\zeta_{t^*-1}}$ goes to state

$$q_{t^*,\mathsf{var}(t^*)}^{\sigma_{t^*,\mathsf{var}(t^*)}(\zeta_{t^*-1})} = q_{t^*,\mathsf{var}(t^*)}^{\zeta_{t^*}}$$

on $x_{\mathsf{var}(t^*)}$, where the above equality follows from the definition of ζ_{t^*}. Again by inspection, it can be seen that state $q_{t^*,\mathsf{var}(t^*)}^{\zeta_{t^*}}$ goes to state $q_{t^*,\ell}^{\zeta_{t^*}}$ on $x_{\mathsf{var}(t^*)+1}, \ldots, x_\ell$ and $q_{t^*,\ell}^{\zeta_{t^*}}$ goes state $q_{t^*+1,0}^{\zeta_{t^*}}$ on input 0. This completes the proof of the claim. □

By setting $t = L$, the claim implies that state q_{st} goes to state $q_{\mathsf{ed}}^{\zeta_L}$ on input $\mathbf{y}$. Thus, we have $M(\mathbf{y}) = 1$ iff $\zeta_1 = 1$, which implies $M(\mathbf{y}) = 1$ iff $\mathsf{BP}(\mathbf{x}) = 1$. This completes the proof of the theorem. □

We then discuss that if there exists subexponentially secure SKFE (please see Appendix A.3 for a formal definition) 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 5.3 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 5.3 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 5.3.

Then, we apply the conversion given by [AJ15, BV15] to the SKFE for NC^1 to obtain SKFE for all circuits. We then further apply the conversion by Kitagawa et al. [KNT17, KNT18] to the SKFE for all circuits to obtain iO. Note that while the former conversion incurs only polynomial loss, the latter conversion incurs sub-exponential security loss.

In summary, we obtain the following theorem.

Theorem 5.5. *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

[ABCP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. Cryptology ePrint Archive, Report 2015/017, 2015. http://eprint.iacr.org/ To appear in PKC'15.

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *CRYPTO*, 2015.

[ADGM16] Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee. Cryptanalysis of indistinguishability obfuscations of circuits over ggh13. eprint 2016, 2016.

[AF18] Prabhanjan Ananth and Xiong Fan. Attribute based encryption with sublinear decryption from lwe. Cryptology ePrint Archive, Report 2018/273, 2018. https://eprint.iacr.org/2018/273.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Asiacrypt*, 2011.

[Agr17] Shweta Agrawal. Stronger security for reusable garbled circuits, new definitions and attacks. In *Crypto*, 2017.

[Agr18] Shweta Agrawal. Indistinguishability obfuscation minus multilinear maps: New methods for bootstrapping and instantiation, 2018.

[AGVW13] Shweta Agrawal, Sergey Gurbanov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *Crypto*, 2013.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 308–326, 2015.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Achieving compactness generically: Indistinguishability obfuscation from non-compact functional encryption. IACR Cryptology ePrint Archive, 2015:730, 2015.

[AJS18] Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: io from lwe, bilinear maps, and weak pseudorandomness. Cryptology ePrint Archive, Report 2018/615, 2018.

[ALS16] Shweta Agrawal, Benoit Libert, and Damien Stehle. Fully secure functional encryption for linear functions from standard assumptions, and applications. In *Crypto*, 2016.

[AM18] Shweta Agrawal and Monosij Maitra. Fe and io for turing machines from minimal assumptions. In *TCC*, 2018.

[AR17] Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In *TCC*, 2017.

[AS17a] Shweta Agrawal and Ishaan Preet Singh. Reusable garbled deterministic finite automata from learning with errors. In *ICALP*, volume 80. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[AS17b] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *EUROCRYPT*, 2017.

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *Eurocrypt*, 2014.

[Bar89] David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

[BCFG17] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *Crypto*, 2017.

[BGG$^+$14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT*, pages 533–556, 2014.

[BGI$^+$01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.

[BL15] Xavier Boyen and Qinyi Li. Attribute-based encryption for finite automata from lwe. In *ProvSec*, 2015.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ITCS '14, 2014.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *FOCS*, 2015:163, 2015.

[BV16] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-abe from lwe: Unbounded attributes and semi-adaptive security. In *Crypto*, 2016.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.

[CFL$^+$] Jung Hee Cheon, Pierre-Alain Fouque, Changmin Lee, Brice Minaud, and Hansol Ryu. Cryptanalysis of the new clt multilinear map over the integers. Eprint 2016/135.

[CGH$^+$15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrede Lepoint, Hemanta K Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New mmap attacks and their limitations. In *Advances in Cryptology–CRYPTO 2015*, pages 247–266. Springer, 2015.

[CH85] Stephen A. Cook and H. James Hoover. A depth-universal circuit. *SIAM J. Comput.*, 14(4):833–839, 1985.

[CHL$^+$15] J.-H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé. Cryptanalysis of the multilinear map over the integers. In *Proc. of EUROCRYPT*, volume 9056 of *LNCS*, pages 3–12. Springer, 2015.

[CJL] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low level encoding of zero. Eprint 2016/139.

[CLLT16] Jean-Sébastien Coron, Moon Sung Lee, Tancrède Lepoint, and Mehdi Tibouchi. Zeroizing attacks on indistinguishability obfuscation over clt13. Eprint 2016, 2016.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, 2013.

[GGH$^+$13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013. `http://eprint.iacr.org/`.

[GGH$^+$13c] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO*, 2013.

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.

[GKW16] Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In *TCC*, 2016.

[Gol08] Oded Goldreich. Computational complexity: a conceptual perspective. *ACM Sigact News*, 39(3):35–39, 2008.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.

[GTKP$^+$13] S. Goldwasser, Y. Tauman Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run turing machines on encrypted data. In *CRYPTO (2)*, pages 536–553, 2013.

[GV15] Sergey Gorbunov and Dhinakaran Vinayagamurthy. Riding on asymmetry: Efficient abe for branching programs. In *Proceedings, Part I, of the 21st International Conference on Advances in Cryptology – ASIACRYPT 2015 - Volume 9452*, 2015.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions from multiparty computation. In *CRYPTO*, 2012.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute based encryption for circuits. In *STOC*, 2013.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from lwe. In *Crypto*, 2015.

[HJ15] Y. Hu and H. Jia. Cryptanalysis of GGH map. Cryptology ePrint Archive: Report 2015/301, 2015.

[KNT17] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Indistinguishability obfuscation for all circuits from secret-key functional encryption. *IACR Cryptology ePrint Archive*, 2017:361, 2017.

[KNT18] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Obfustopia built on secret-key functional encryption. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 603–648, 2018.

[KNTY18] Fuyuki Kitagawa, Ryo Nishimaki, Keisuke Tanaka, and Takashi Yamakawa. Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. Cryptology ePrint Archive, Report 2018/974, 2018. https://eprint.iacr.org/2018/974.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 28–57, 2016.

[Lin17] Huijia Lin. Indistinguishability obfuscation from sxdh on 5-linear maps and locality-5 prgs. In *Crypto*, 2017.

[LM18] Huijia Lin and Christian Matt. Pseudo flawed-smudging generators and their application to indistinguishability obfuscation. Cryptology ePrint Archive, Report 2018/646, 2018.

[LOS$^+$10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local prgs. In *Crypto*, 2017.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *FOCS*, 2016.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13. In *Crypto*, 2016.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[Wat12] Brent Waters. Functional encryption for regular languages. In *Crypto*, 2012.

APPENDICES

A Definitions: Predicate and Functional Encryption

A.1 Predicate and Bounded Key Functional Encryption for Circuits

We present the definition of predicate and bounded key functional encryption for general circuits similarly to [GVW15, GKP+13, Agr17]. We follow the notation of [Agr17] which provides a single definition for predicate encryption and succinct bounded key functional encryption. Since this primitive interpolates predicate and functional encryption, we denote it by PE^+. We note that this definition achieves input privacy but not machine privacy. Machine privacy is considered in the notion of reusable garbled circuits, defined in Section **??**.

For $\lambda \in \mathbb{N}$, let $\mathcal{C}_{\mathsf{inp},\mathsf{d}}$ denote a family of circuits with inp bit inputs, an a-priori bounded depth d, and binary output and $\mathcal{C} = \{\mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}\}_{\lambda\in\mathbb{N}}$. A predicate encryption scheme PE^+ for $\mathcal{C}$ over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of four algorithms:

- $\mathsf{PE}^+.\mathsf{Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}})$ 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 $(\mathsf{PE}^+.\mathsf{mpk}, \mathsf{PE}^+.\mathsf{msk})$.

- $\mathsf{PE}^+.\mathsf{Enc}(\mathsf{PE}^+.\mathsf{mpk}, \mathbf{x}, m)$ is a PPT algorithm that takes as input the master public key $\mathsf{PE}^+.\mathsf{mpk}$, a string $\mathbf{x} \in \{0,1\}^{\mathsf{inp}}$ and a message $m \in \mathcal{M}$. It outputs a ciphertext $\mathsf{PE}^+.\mathsf{ct}$.

- $\mathsf{PE}^+.\mathsf{KeyGen}(\mathsf{PE}^+.\mathsf{mpk}, \mathsf{PE}^+.\mathsf{msk}, C)$ is a PPT algorithm that takes as input the master public key $\mathsf{PE}^+.\mathsf{mpk}$, master secret key $\mathsf{PE}^+.\mathsf{msk}$, and a circuit $C \in \mathcal{C}_{\mathsf{inp}(\lambda),\mathsf{d}(\lambda)}$ and outputs a corresponding secret key $\mathsf{PE}^+.\mathsf{sk}_C$.

- $\mathsf{PE}^+.\mathsf{Dec}(\mathsf{PE}^+.\mathsf{mpk}, \mathsf{PE}^+.\mathsf{sk}_C, C, \mathsf{PE}^+.\mathsf{ct})$ is a deterministic algorithm that takes as input the master public key $\mathsf{PE}^+.\mathsf{mpk}$, the secret key $\mathsf{PE}^+.\mathsf{sk}_C$, its associated circuit C, and a ciphertext $\mathsf{PE}^+.\mathsf{ct}$ and outputs either a message m' or $\perp$.

Definition A.1 (Correctness)**.** A predicate encryption scheme for circuits PE^+ 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 $m \in \mathcal{M}$,

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

where the probability is taken over the coins of $\mathsf{PE}^+.\mathsf{Setup}$, $\mathsf{PE}^+.\mathsf{KeyGen}$, and $\mathsf{PE}^+.\mathsf{Enc}$.

Security. Next, we define simulation based security for PE^+. Note that simulation based security is impossible for functional encryption against an adversary that requests even a *single* key after seeing the challenge ciphertext [BSW11], or an unbounded number of keys before seeing the challenge ciphertext [AGVW13]. However, against an adversary who only requests an a-priori bounded number of keys *before* seeing the challenge ciphertext, simulation based security is possible but causes the ciphertext size to grow with the number of requested keys [AGVW13].

Following [Agr17], we provide a definition that subsumes single (or bounded) key functional encryption as well as predicate encryption; namely where an attacker can make an unbounded number of function queries C_i so long as it holds that the function keys do not decrypt the challenge ciphertext $\mathsf{PE}^+.\mathsf{Enc}(\mathsf{PE}^+.\mathsf{mpk}, \mathbf{x}, m)$ to recover m. Thus, it holds that $C_i(\mathbf{x}) = 0$ for all requested C_i. We shall refer to such C_i as 0-keys, and any C such that $C(\mathbf{x}) = 1$ as a 1-key. In our definition, the adversary can request a single arbitrary (i.e. 0 or 1) key followed by an unbounded polynomial number of 0-keys. As in [Agr17], we refer to this security notion as $(1, \mathrm{poly})$ simulation security.

Definition A.2 ($(1, \mathrm{poly})$-Sel-SIM Security)**.** Let PE^+ be a predicate encryption scheme for a Boolean circuit family $\mathcal{C}$. For a stateful PPT adversary A and a stateful PPT simulator Sim, consider the following two experiments:

```
$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{PE}^+,\mathsf{A}}(1^\lambda)}$:

1: $(X, \mathsf{Msg}, C^*) \leftarrow \mathsf{A}(1^\lambda)$
2: $(\mathsf{PE}^+.\mathsf{mpk}, \mathsf{PE}^+.\mathsf{msk}) \leftarrow \mathsf{PE}^+.\mathsf{Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}})$
3: For $\mathbf{x}_i \in X$, let $b_i = m_i$.
4: let $\mathsf{CT}_X :=$
       $\{\mathsf{PE}^+.\mathsf{ct}_{\mathbf{x}_i} \leftarrow \mathsf{PE}^+.\mathsf{Enc}(\mathsf{PE}^+.\mathsf{mpk}, \mathbf{x}_i, b_i)\}_{i \in [|X|]}$
5: $\mathsf{PE}^+.\mathsf{sk}_{C^*} \leftarrow \mathsf{PE}^+.\mathsf{KeyGen}(\mathsf{PE}^+.\mathsf{mpk}, \mathsf{PE}^+.\mathsf{msk}, C^*)$
6: $\alpha \leftarrow \mathsf{A}^{\mathsf{PE}^+.\mathsf{KeyGen}(\mathsf{PE}^+.\mathsf{mpk}, \mathsf{PE}^+.\mathsf{msk}, \cdot)}(\mathsf{CT}_X, \mathsf{PE}^+.\mathsf{sk}_{C^*})$
7: Output $(X, \mathsf{Msg}, \alpha)$
```

```
$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{PE}^+,\mathrm{Sim}}(1^\lambda)}$:

1: $(X, \mathsf{Msg}, C^*) \leftarrow \mathsf{A}(1^\lambda)$
2: $\mathsf{PE}^+.\mathsf{mpk} \leftarrow \mathrm{Sim}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}, C^*)$
3: For $\mathbf{x}_i \in X$, let $b_i = m_i$ if $C^*(\mathbf{x}_i) = 1$,
   $\perp$ otherwise.
4: Let $\mathsf{CT}_X :=$
   $\{\mathsf{PE}^+.\mathsf{ct}_{\mathbf{x}_i} \leftarrow \mathrm{Sim}(1^{|\mathbf{x}_i|}, C^*, b_i)\}_{i \in [|X|]}$
5: $\mathsf{PE}^+.\mathsf{sk}_{C^*} \leftarrow \mathrm{Sim}(C^*)$
6: $\alpha \leftarrow \mathsf{A}^{\mathrm{Sim}}(\mathsf{CT}_X, \mathsf{PE}^+.\mathsf{sk}_{C^*})$
7: Output $(X, \mathsf{Msg}, \alpha)$
```

Here, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|X|}\}$ is the target set of attributes of length inp, and let $\mathsf{Msg} = \{m_1, \ldots, m_{|X|}\}$ be the corresponding set of messages in $\mathcal{M}$. We say an adversary A is admissible if:

1. For a single query C^*, it may hold that $C^*(\mathbf{x}) = 1$ or $C^*(\mathbf{x}) = 0$ for any $\mathbf{x} \in X$.

2. For all other queries $C_i \neq C^*$, it holds that $C_i(\mathbf{x}) = 0$ for any $\mathbf{x} \in X$.

The functional encryption scheme FE is then said to be $(1, \mathrm{poly})$-Sel-SIM-secure if there is an admissible stateful PPT simulator Sim such that for every admissible PPT adversary A, the following two distributions are computationally indistinguishable.

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

For the (Q, poly) version of the above game, we replace each occurrence of C^* with a tuple $C_1^*, \ldots, C_Q^*$ and set $b_i = m_i$ if there is $j \in [Q]$ such that $C_j^*(\mathbf{x}_i) = 1$ and otherwise $b_i = \perp$.

Remark A.3. Note that the above definition is a multi-challenge one, where the adversary can obtain multiple challenge ciphertexts. While this security notion implies more standard notion of single-challenge security, the latter may not imply the former since the simulator is stateful and may use some secret information to simulate the game. Because of the reason, it is impossible to perform the hybrid argument to prove the implication. However, in the special case of Agrawal's PE^+ scheme [Agr17], which is only proven secure in the single-challenge setting, actually satisfies the above stronger notion, since the simulator for the ciphertext does not use any secret information not known to the adversary in her construction.

In our construction of PE^+ for NFA in Appendix B, we will use the scheme by Agrawal [Agr17] as a building block. The following theorem summarizes the efficiency properties of her construction.

Theorem A.4 (Adapted from [Agr17])**.** *There exists a selectively secure FE scheme* $\mathsf{PE}^+ = (\mathsf{PE}^+.\mathsf{Setup}, \mathsf{PE}^+.\mathsf{KeyGen}, \mathsf{PE}^+.\mathsf{Enc}, \mathsf{PE}^+.\mathsf{Dec})$ *with the following properties under the LWE assumption.*

1. *The circuit* $\mathsf{PE}^+.\mathsf{Setup}(\cdot,\cdot,\cdot;\cdot)$*, which takes as input* $1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}$*, and a randomness* r *and outputs* $\mathsf{PE}^+.\mathsf{msk} = \mathsf{PE}^+.\mathsf{Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}}; r)$*, 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* $|\mathsf{PE}^+.\mathsf{sk}_C| \leq \mathrm{poly}(\lambda, \mathsf{d})$ *for any* $C \in \mathcal{C}_{\mathsf{inp},\mathsf{d}}$*, where* $(\mathsf{PE}^+.\mathsf{mpk}, \mathsf{PE}^+.\mathsf{msk}) \leftarrow \mathsf{PE}^+.\mathsf{Setup}(1^\lambda, 1^{\mathsf{inp}}, 1^{\mathsf{d}})$ *and* $\mathsf{PE}^+.\mathsf{sk}_C \leftarrow \mathsf{PE}^+.\mathsf{KeyGen}(\mathsf{PE}^+.\mathsf{mpk}, \mathsf{PE}^+.\mathsf{msk}, C)$*. In particular, the length of the secret key is independent of the input length* inp *and the size of the circuit* C*.*

3. *Let* $C : \{0,1\}^{\mathsf{inp}+\ell} \rightarrow \{0,1\}$ *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* $\mathsf{PE}^+.\mathsf{KeyGen}(\cdot,\cdot,C[\cdot];\cdot)$*, that takes as input* $\mathsf{PE}^+.\mathsf{mpk}$*,* $\mathsf{PE}^+.\mathsf{msk}$*,* v*, and randomness* r *and outputs* $\mathsf{PE}^+.\mathsf{KeyGen}(\mathsf{PE}^+.\mathsf{mpk}, \mathsf{PE}^+.\mathsf{msk}, C[v]; r)$*, can be implemented with depth* $\mathsf{depth}(C) \cdot \mathrm{poly}(\lambda, \mathsf{d})$*.*

Proof. The proof follows from the proof of Theorem 2.9 and the structure of the PE^+ scheme of [Agr17]. We note that the PE^+ scheme of [Agr17] uses fully homomorphic encryption (as in [GVW15]), to hide the attributes in the ABE scheme of [BGG$^+$14], and reverses the FHE encryption by augmenting the circuit in the function key with an FHE decryption circuit. Thus, if a key is requested for a circuit C, then the PE^+ scheme must construct a function key for a circuit $\hat{C} \circ \mathsf{IP}$ where $\hat{C}$ is the FHE evaluation circuit corresponding to circuit C, and IP is the FHE decryption procedure, which in turn, involves computing an inner product followed by a modular reduction [BGV12, GSW13]. Since the FHE evaluation circuit corresponding to some circuit C only causes constant polynomial blowup in depth [BGV12, GSW13] and the FHE decryption circuit is in NC_1, we have that the depth of the augmented circuit $\hat{C} \circ \mathsf{IP}$ is $\mathrm{poly}(\lambda) \cdot \mathsf{depth}(C)$. The setup and encryption algorithms of the PE^+ scheme of [Agr17] are the same as that of [BGG$^+$14]. Hence, the theorem follows from the proof of Theorem 2.9. $\square$

A.2 Predicate Encryption and Bounded Key Functional Encryption for NFA

A secret-key functional encryption scheme NfaPE^+ 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 $2 \leq |\Sigma| \leq \mathrm{poly}(\lambda)$.

- $\mathsf{NfaPE}^+.\mathsf{Setup}(1^\lambda)$ is a PPT algorithm takes as input the unary representation of the security parameter and outputs the master secret key $\mathsf{NfaPE}^+.\mathsf{msk}$.

- $\mathsf{NfaPE}^+.\mathsf{Enc}(\mathsf{NfaPE}^+.\mathsf{msk}, \mathbf{x}, m)$ is a PPT algorithm that takes as input the master secret key $\mathsf{NfaPE}^+.\mathsf{msk}$, a string $\mathbf{x} \in \Sigma^*$ of arbitrary length and a message $m \in \mathcal{M}$. It outputs a ciphertext $\mathsf{NfaPE}^+.\mathsf{ct}$.

- $\mathsf{NfaPE}^+.\mathsf{KeyGen}(\mathsf{NfaPE}^+.\mathsf{msk}, M)$ is a PPT algorithm that takes as input the master secret key $\mathsf{NfaPE}^+.\mathsf{msk}$ and a description of an NFA machine M. It outputs a corresponding secret key $\mathsf{NfaPE}^+.\mathsf{sk}_M$.

- $\mathsf{NfaPE}^+.\mathsf{Dec}(\mathsf{NfaPE}^+.\mathsf{sk}_M, M, \mathsf{NfaPE}^+.\mathsf{ct})$ is a deterministic polynomial time algorithm that takes as input the secret key $\mathsf{NfaPE}^+.\mathsf{sk}_M$, its associated NFA M, and a ciphertext $\mathsf{NfaPE}^+.\mathsf{ct}$ and outputs either a message m' or $\perp$.

Remark A.5. As in the construction in Sec. 3.2, we will pass an additional parameter $\mathsf{s} = \mathsf{s}(\lambda)$ to the $\mathsf{NfaPE}^+.\mathsf{Setup}$, $\mathsf{NfaPE}^+.\mathsf{Enc}$, $\mathsf{NfaPE}^+.\mathsf{KeyGen}$ algorithms denoting the description size of NFAs that the scheme can deal with. The construction in Sec. 4 can be adapted in a straightforward way to support NFAs with arbitrary size.

Definition A.6 (Correctness)**.** A scheme NfaPE^+ is correct if for all NFAs M, all $\mathbf{x} \in \Sigma^*$ such that $M(\mathbf{x}) = 1$ and for all messages $m \in \mathcal{M}$,

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

where the probability is taken over the coins of $\mathsf{NfaPE}^+.\mathsf{Setup}$, $\mathsf{NfaPE}^+.\mathsf{KeyGen}$, and $\mathsf{NfaPE}^+.\mathsf{Enc}$.

Definition A.7 ($(1, \mathrm{poly})$-Sel-SIM Security)**.** The definition is adapted from Defn A.2 in the symmetric key setting. For a stateful PPT adversary A and a stateful PPT simulator Sim, consider the following two experiments:

$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{NfaPE}^+,\mathsf{A}}(1^\lambda)}$**:**

```
1: $(X, \mathsf{Msg}, M^*) \leftarrow \mathsf{A}(1^\lambda)$
2: For $\mathbf{x}_i \in X$, let $b_i = m_i$.
   Let $\mathsf{NfaPE}^+.\mathsf{msk} \leftarrow \mathsf{NfaPE}^+.\mathsf{Setup}(1^\lambda)$.
3: Let $\mathsf{CT}_X :=$
   $\{\mathsf{PE}^+.\mathsf{ct}_{\mathbf{x}_i} \leftarrow \mathsf{NfaPE}^+.\mathsf{Enc}(\ \mathsf{NfaPE}^+.\mathsf{msk}, \mathbf{x}_i, b_i\ )\}_{i \in [|X|]}$
4: $\mathsf{PE}^+.\mathsf{sk}_{M^*} \leftarrow \mathsf{NfaPE}^+.\mathsf{KeyGen}(\mathsf{NfaPE}^+.\mathsf{msk}, M^*)$
5: $\alpha \leftarrow \mathsf{A}^{\mathsf{NfaPE}^+.\mathsf{KeyGen}(\mathsf{NfaPE}^+.\mathsf{msk},\cdot)}(\mathsf{CT}_X, \mathsf{PE}^+.\mathsf{sk}_{M^*})$
6: Output $(X, \mathsf{Msg}, \alpha)$
```

$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{NfaPE}^+,\mathrm{Sim}}(1^\lambda)}$**:**

```
1: $(X, \mathsf{Msg}, M^*) \leftarrow \mathsf{A}(1^\lambda)$
2: For $\mathbf{x}_i \in X$, let $b_i = m_i$ if $M^*(\mathbf{x}_i) = 1$,
   $\perp$ otherwise.
3: let $\mathsf{CT}_X :=$
   $\{\mathsf{PE}^+.\mathsf{ct}_{\mathbf{x}_i} \leftarrow \mathrm{Sim}(1^{|\mathbf{x}_i|}, M^*, b_i)\}_{i \in [|X|]}$
4: $\mathsf{PE}^+.\mathsf{sk}_{M^*} \leftarrow \mathrm{Sim}(M^*)$
5: $\alpha \leftarrow \mathsf{A}^{\mathrm{Sim}}(\mathsf{CT}_X, \mathsf{PE}^+.\mathsf{sk}_{M^*})$
6: Output $(X, \mathsf{Msg}, \alpha)$
```

Here, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|X|}\}$ is the target set of attributes over Σ^*, and let $\mathsf{Msg} = \{m_1, \ldots, m_{|X|}\}$ be the corresponding set of messages in $\mathcal{M}$. We say an adversary A is admissible if:

1. For a single query M^*, it may hold that $M^*(\mathbf{x}_i) = 1$ or $M^*(\mathbf{x}_i) = 0$ for any $\mathbf{x}_i \in X$.
2. For all other queries $M_j \neq M^*$, it holds that $M_j(\mathbf{x}_i) = 0$ for any $\mathbf{x}_i \in X$.

The PE^+ scheme NfaPE^+ is then said to be $(1, \mathrm{poly})$-Sel-SIM-secure if there is an admissible stateful PPT simulator Sim such that for every admissible PPT adversary A, the following two distributions are computationally indistinguishable.

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

For the (Q, poly) version of the above game, we replace each occurrence of M^* with a tuple $M_1^*, \ldots, M_Q^*$ and set $b_i = m_i$ if there is $j \in [Q]$ such that $M_j^*(\mathbf{x}_i) = 1$ and otherwise $b_i = \perp$.

A.3 Symmetric Key Functional Encryption

Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ denote ensembles where each $\mathcal{X}_\lambda$ and $\mathcal{Y}_\lambda$ is a finite set. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ denote an ensemble where each $\mathcal{F}_\lambda$ is a finite collection of circuits, and each circuit $g \in \mathcal{F}_\lambda$ takes as input a string $x \in \mathcal{X}_\lambda$ and outputs $g(x) \in \mathcal{Y}_\lambda$.

A symmetric key functional encryption scheme SKFE for $\mathcal{F}$ consists of four algorithms $\mathsf{SKFE} = (\mathsf{SKFE.Setup}, \mathsf{SKFE.KeyGen}, \mathsf{SKFE.Enc}, \mathsf{SKFE.Dec})$ defined as follows.

- $\mathsf{SKFE.Setup}(1^\lambda)$ is a PPT algorithm takes as input the unary representation of the security parameter and outputs the master secret key msk.

- $\mathsf{SKFE.KeyGen}(\mathsf{msk}, g)$ is a PPT algorithm that takes as input the master secret key msk and a circuit $g \in \mathcal{F}_\lambda$ and outputs a corresponding secret key sk_g.
- $\mathsf{SKFE.Enc}(\mathsf{msk}, x)$ is a PPT algorithm that takes as input the master secret key msk and an input message $x \in \mathcal{X}_\lambda$ and outputs a ciphertext ct.
- $\mathsf{SKFE.Dec}(\mathsf{sk}_g, \mathsf{ct}_x)$ is a deterministic algorithm that takes as input the secret key sk_g and a ciphertext ct_x and outputs $g(x)$.

Definition A.8 (Correctness)**.** A symmetric key functional encryption scheme SKFE is correct if for all $g \in \mathcal{F}_\lambda$ and all $x \in \mathcal{X}_\lambda$,

$$\Pr\left[\begin{array}{l}\mathsf{msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda);\\ \mathsf{SKFE.Dec}\Big(\mathsf{SKFE.KeyGen}(\mathsf{msk}, g), \mathsf{SKFE.Enc}(\mathsf{msk}, x)\Big) \neq g(x)\end{array}\right] = \mathrm{negl}(\lambda)$$

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

Security. In this paper we will consider the standard indistinguishability based definition.

Definition A.9. A symmetric key functional encryption scheme SKFE for a function family $\mathcal{F}$ is very selectively secure under the unbounded collusion, if for all PPT adversaries A, the advantage of A in the following experiment is negligible in the security parameter λ:

1. **Key Queries and Challenge Queries.** Given the security parameter 1^λ, A submits key queries $g_1, \ldots, g_q \in \mathcal{F}_\lambda$ and ciphertext queries $(x_1^{(0)}, \ldots, x_{q'}^{(0)}), (x_1^{(1)}, \ldots, x_{q'}^{(1)})$ to the challenger. Here, the number of key queries and the challenge queries can be arbitrarily large polynomial. These queries should satisfy $g_i(x_j^{(0)}) = g_i(x_j^{(1)})$ for all $i \in [q]$ and $j \in [q']$.
2. **Challenge.** Then, the challenger runs $\mathsf{msk} \leftarrow \mathsf{SKFE.Setup}(1^\lambda)$ and chooses random bit b. Then it computes $\mathsf{sk}_i \leftarrow \mathsf{SKFE.KeyGen}(\mathsf{msk}, g_i)$ for $i \in [q]$ and $\mathsf{ct}_j \leftarrow \mathsf{SKFE.Enc}(\mathsf{msk}, x_j^{(b)})$ for $j \in [q']$ and gives $\{\mathsf{sk}_i\}_{i\in[q]}$ and $\{\mathsf{ct}_j\}_{j\in[q']}$ to the adversary.
3. **Guess.** A outputs a bit b', and succeeds if $b' = b$.

The *advantage* of A is $|\Pr[b = b'] - 1/2|$ in the above game.

Function Class. In this paper, we consider two function classes for SKFE. The first one is circuits, namely we set $\mathcal{X}_\lambda = \{0,1\}^{\mathsf{inp}(\lambda)}$ and $\mathcal{Y}_\lambda$ is the circuits of input length $\mathsf{inp}(\lambda)$ and fixed depth $\mathsf{depth}(\lambda)$ and output length $\mathsf{out}(\lambda)$. The second class is DFAs, namely we set $\mathcal{X}_\lambda = \Sigma^*$ and $\mathcal{Y}_\lambda$ is DFA with alphabet Σ.

B Construction: Predicate and Bounded Key Functional Encryption for NFA

We construct a secret key predicate and bounded key FE scheme for NFA denoted by $\mathsf{NfaPE}^+ = (\mathsf{NfaPE}^+.\mathsf{Setup}, \mathsf{NfaPE}^+.\mathsf{KeyGen}, \mathsf{NfaPE}^+.\mathsf{Enc}, \mathsf{NfaPE}^+.\mathsf{Dec})$ 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 2.15. We can instantiate FE with the scheme proposed by Goldwasser et al. [GKP+13].

3. $\mathsf{PE}^+ = (\mathsf{PE}^+.\mathsf{Setup}, \mathsf{PE}^+.\mathsf{KeyGen}, \mathsf{PE}^+.\mathsf{Enc}, \mathsf{PE}^+.\mathsf{Dec})$: A PE^+ scheme for circuits that satisfies the efficiency properties described in Theorem A.4. We can instantiate PE^+ with the scheme proposed by Agrawal [Agr17].

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 $\mathsf{depth}(U) \le O(\mathsf{depth}(C))$. For construction of such a universal circuit, we refer to [CH85].

Below we provide our construction for secret key PE^+ for NFA, where size of machines is s. This restriction can be removed by the same trick as in Sec. 4. 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 $\mathsf{PE}^+.\mathsf{Setup}$ and $\mathsf{PE}^+.\mathsf{KeyGen}$, the efficiency properties of PE^+ described in Theorem A.4 is preserved.

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

1. For all $j \in [0,\lambda]$, sample PRF keys $\widehat{\mathsf{K}}_j, \mathsf{R}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$.
2. For all $j \in [0,\lambda]$, sample $(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{d}(\lambda)}, 1^{\mathsf{out}(\lambda)})$.

   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 $\mathsf{NfaPE}^+.\mathsf{msk} = (\{\widehat{\mathsf{K}}_j, \mathsf{R}_j, \mathsf{FE.mpk}_j, \mathsf{FE.msk}_j\}_{j\in[0,\lambda]})$.

$\mathsf{NfaPE}^+.\mathsf{Enc}(\mathsf{NfaPE}^+.\mathsf{msk}, \mathbf{x}, m, 1^{\mathsf{s}})$: On input the master secret key $\mathsf{NfaPE}^+.\mathsf{msk}$, an attribute $\mathbf{x} \in \Sigma^*$ of length at most 2^λ, a message m and the bound s on NFA size, do the following:

1. Parse the master secret key as $\mathsf{NfaPE}^+.\mathsf{msk} \to (\{\widehat{\mathsf{K}}_j, \mathsf{R}_j, \mathsf{FE.mpk}_j, \mathsf{FE.msk}_j\}_{j\in[0,\lambda]})$.
2. Set $\hat{\mathbf{x}} = \mathbf{x}\|\bot^{2^i-\ell}$, where $\ell = |\mathbf{x}|$ and $i = \lceil \log \ell \rceil$.
3. Compute a PE^+ key pair $(\mathsf{PE}^+.\mathsf{mpk}_i, \mathsf{PE}^+.\mathsf{msk}_i) = \mathsf{PE}^+.\mathsf{Setup}(1^\lambda, 1^{2^i\eta}, 1^{\hat{\mathsf{d}}}; \widehat{\mathsf{K}}_i)$ with $\widehat{\mathsf{K}}_i$ as the randomness.

   Here, we generate an instance of PE^+ 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 $\mathsf{PE}^+.\mathsf{ct} \leftarrow \mathsf{PE}^+.\mathsf{Enc}(\mathsf{PE}^+.\mathsf{mpk}_i, \hat{\mathbf{x}}, m)$ as an PE^+ 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 Figure 5.
6. Output $\mathsf{NfaPE}^+.\mathsf{ct} = (\mathsf{FE.sk}_i, \mathsf{PE}^+.\mathsf{mpk}_i, \mathsf{PE}^+.\mathsf{ct})$.

Without loss of generality, we assume that i is revealed from $\mathsf{PE}^+.\mathsf{ct}$.

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

(a) 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.

(b) Compute $(\mathsf{PE}^+.\mathsf{mpk}, \mathsf{PE}^+.\mathsf{msk}) = \mathsf{PE}^+.\mathsf{Setup}(1^\lambda, 1^{2^i \eta}, 1^{\hat{\mathsf{d}}}; \widehat{\mathsf{K}})$.

(c) Compute $\widehat{M_{2^i}} = \mathsf{To\text{-}Circuit}_{\mathsf{s},2^i}(M)$. (See Theorem 3.1 for the definition of To-Circuit.)

(d) Compute and output $\mathsf{PE}^+.\mathsf{sk}_{U[\widehat{M_{2^i}}]} = \mathsf{PE}^+.\mathsf{KeyGen}(\mathsf{PE}^+.\mathsf{mpk}, \mathsf{PE}^+.\mathsf{msk}, U[\widehat{M_{2^i}}]; \widehat{\mathsf{R}})$.

Figure 5

$\mathsf{NfaPE}^+.\mathsf{KeyGen}(\mathsf{NfaPE}^+.\mathsf{msk}, M, 1^{\mathsf{s}})$: On input the master secret key $\mathsf{NfaPE}^+.\mathsf{msk}$, the description of an NFA M and a size s of the NFA, if $|M| \neq \mathsf{s}$, output $\perp$ and abort. Else, proceed as follows.

1. Parse the master secret key as $\mathsf{NfaPE}^+.\mathsf{msk} \to (\{\widehat{\mathsf{K}}_j, \mathsf{R}_j, \mathsf{FE.mpk}_j, \mathsf{FE.msk}_j\}_{j\in[0,\lambda]})$.
2. Sample $\widehat{\mathsf{R}}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for all $j \in [0, \lambda]$.
3. Compute $\mathsf{FE.ct}_j \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_j, (M, \widehat{\mathsf{K}}_j, \widehat{\mathsf{R}}_j))$ for all $j \in [0, \lambda]$.
4. Output $\mathsf{NfaPE}^+.\mathsf{sk}_M = \{\mathsf{FE.ct}_j\}_{j\in[0,\lambda]}$.

$\mathsf{NfaPE}^+.\mathsf{Dec}(\mathsf{NfaPE}^+.\mathsf{sk}_M, M, \mathsf{NfaPE}^+.\mathsf{ct})$: On input a secret key for NFA M and a ciphertext, proceed as follows:

1. Parse the secret key as $\mathsf{NfaPE}^+.\mathsf{sk}_M \to \{\mathsf{FE.ct}_j\}_{j\in[0,\lambda]}$ and the ciphertext as $\mathsf{NfaPE}^+.\mathsf{ct} \to (\mathsf{FE.sk}_i, \mathsf{PE}^+.\mathsf{mpk}_i, \mathsf{PE}^+.\mathsf{ct})$.
2. Learn i from $\mathsf{PE}^+.\mathsf{ct}$ and choose $\mathsf{FE.ct}_i$ from $\mathsf{NfaPE}^+.\mathsf{sk}_M = \{\mathsf{FE.ct}_j\}_{j\in[0,\lambda]}$.
3. Compute $y = \mathsf{FE.Dec}(\mathsf{FE.mpk}_i, \mathsf{FE.sk}_i, \mathsf{FE.ct}_i)$.
4. Compute and output $z = \mathsf{PE}^+.\mathsf{Dec}(\mathsf{PE}^+.\mathsf{mpk}_i, y, U[\widehat{M_{2^i}}], \mathsf{PE}^+.\mathsf{ct}_i)$, where we interpret y as a secret key.

Correctness follows as in Section 3.3 by appropriately setting $\mathsf{out}(\lambda)$ and $\mathsf{d}(\lambda)$. For security, we prove the following theorem.

Theorem B.1. *Assume that* FE *satisfies full simulation based security,* PE^+ *is* $\mathsf{Sel\text{-}SIM}$ *secure, and that* PRF *is a secure pseudorandom function. Then,* NfaPE^+ *satisfies selective simulation based security.*

We note that the structure of hybrids is almost the same as in the proof of Theorem 3.4. The difference is that instead of changing ciphertexts corresponding to each instance of PE^+ from an encryption of m_0 to m_1, we change honest setup, key generation, and encryption of each instance of PE^+ to the simulated ones. To do so, we must replace the reliance on ABE for circuits with PE^+ for circuits. In more detail, we consider $\mathbf{Game}_i$ for $i \in [0, i_{\max} + 1]$ as follows, where $i_{\max}$ is defined as in the proof of Theorem 3.4.

$\mathbf{Game}_i$: The game proceeds as follows. In the following $\mathsf{FE}.\mathrm{Sim}$ and $\mathsf{PE}^+.\mathrm{Sim}$ are the simulators for FE and PE^+, respectively.

Setup phase. At the beginning of the game, A takes 1^λ as input and submits 1^{s} and (X, Msg, M^*) to the challenger. Then, the challenger chooses $\{\widehat{\mathsf{K}}_j, \mathsf{R}_j\}_{j\in[0,\lambda]}$ and $\{\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j\}_{j\in[0,\lambda]}$. It further samples $\mathsf{PE}^+.\mathsf{mpk}_j \leftarrow \mathsf{PE}^+.\mathrm{Sim}(1^\lambda, 1^{2^j\eta}, 1^{\hat{\mathsf{d}}}, U[\widehat{M^*_{2^j}}])$ and $\mathsf{FE.sk}_j \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j, C_{\mathsf{s},2^j})$ for $j \leq i - 1$.

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

Simulating Keys. During the game, secret key for M (which is possibly M^*) is answered as follows. For $j \in [0, \lambda]$, the challenger computes

$$\mathsf{FE.ct}_j \leftarrow \begin{cases} \mathsf{FE.Enc}(\mathsf{FE.mpk}_j, (M, \widehat{\mathsf{K}}_j, \widehat{\mathsf{R}}_j)) & \text{If } \lambda \geq j \geq i \\ \mathsf{FE.}\mathrm{Sim}(\mathsf{FE.mpk}_j, \mathsf{FE.sk}_j, C_{\mathsf{s},2^j}, \mathsf{PE}^+.\mathsf{sk}_{U[\widehat{M}_{2^j}]}, 1^{\mathsf{inp}(\lambda)}) & \text{If } j \leq i-1, \end{cases} \tag{B.1}$$

where $\mathsf{PE}^+.\mathsf{sk}_{U[\widehat{M}_{2^j}]} \leftarrow \mathsf{PE}^+.\mathsf{KeyGen}(\mathsf{PE}^+.\mathsf{mpk}_j, \mathsf{PE}^+.\mathsf{msk}_j, U[\widehat{M}_{2^j}])$ and returns $\{\mathsf{FE.ct}_j\}$ to A.

Simulating Ciphertexts. To generate a ciphertext for $m \in \mathsf{Msg}$ associated with $\mathbf{x} \in X$, the challenger sets $j := \lceil \log |\mathbf{x}| \rceil$ and computes

$$\mathsf{PE}^+.\mathsf{ct} \leftarrow \begin{cases} \mathsf{PE}^+.\mathsf{Enc}(\mathsf{PE}^+.\mathsf{mpk}_j, \hat{\mathbf{x}}, m) & \text{If } i_{\max} \geq j \geq i \\ \mathsf{PE}^+.\mathrm{Sim}(1^{2^j \eta}, \widehat{M}^*_{2^j}, b) & \text{If } j \leq i-1, \end{cases}$$

where $b = m$ if $M^*(\mathbf{x}) = 1$ and $\perp$ otherwise. It also computes $\mathsf{FE.sk}_j = \mathsf{FE.KeyGen}(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j, C_{\mathsf{s},2^j}; \mathsf{R}_j)$ if $j \geq i$. The ciphertext is $\mathsf{NfaPE}^+.\mathsf{ct} = (\mathsf{FE.sk}_j, \mathsf{PE}^+.\mathsf{mpk}_j, \mathsf{PE}^+.\mathsf{ct})$.

Finally, A outputs its guess b'.

It is easy to see that $\mathbf{Game}_0$ is the same as $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{NfaPE}^+,\mathsf{A}}(1^\lambda)$. Furthermore, we can construct a simulator for NfaPE^+ from the challenger in $\mathbf{Game}_{i_{\max}+1}$ appropriately, since the challenger in $\mathbf{Game}_{i_{\max}+1}$ only uses b and $|\mathbf{x}|$ to simulate the ciphertext. Therefore, it suffices to show the indistinguishability between $\mathbf{Game}_i$ and $\mathbf{Game}_{i+1}$. To do so, we further consider following sequence of games, which closely follows the proof of Theorem 3.4 except that we do not need counterpart of $\mathbf{Game}_{i,6}$, where we undo changes we made from $\mathbf{Game}_{i,1}$ to $\mathbf{Game}_{i,5}$. We directly go to $\mathbf{Game}_{i+1,0}$ from $\mathbf{Game}_{i,5}$.

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

$\mathbf{Game}_{i,1}$**:** The game is the same as the previous game except that $(\mathsf{PE}^+.\mathsf{mpk}_i, \mathsf{PE}^+.\mathsf{msk}_i)$ and $\mathsf{FE.sk}_i$ are computed at the setup phase using $\widehat{\mathsf{K}}_i$ and R_i.

$\mathbf{Game}_{i,2}$**:** The game is the same as the previous game except that $\mathsf{FE.sk}_i$ is generated using true randomness instead of using the PRF keys.

$\mathbf{Game}_{i,3}$**:** In this game, to answer a key query, $\mathsf{FE.ct}_i$ is computed using $\mathsf{FE.}\mathrm{Sim}$ on input $\mathsf{PE}^+.\mathsf{sk}_{U[\widehat{M}_{2^i}]} = \mathsf{PE}^+.\mathsf{KeyGen}(\mathsf{PE}^+.\mathsf{mpk}_i, \mathsf{PE}^+.\mathsf{msk}_i, U[\widehat{M}_{2^i}]; \widehat{\mathsf{R}}_i)$.

$\mathbf{Game}_{i,4}$**:** In this game, to answer a key query, $\mathsf{PE}^+.\mathsf{sk}_{U[\widehat{M}_{2^i}]}$ is generated using true randomness instead of using the PRF key.

$\mathbf{Game}_{i,5}$**:** In this game, we generate $\mathsf{PE}^+.\mathsf{mpk}_i$ using $\mathsf{PE}^+.\mathrm{Sim}$. We also generate $\mathsf{PE}^+.\mathsf{ct}$ using $\mathsf{PE}^+.\mathrm{Sim}$ when $\lceil \log |\mathbf{x}| \rceil = i$.

Note that we have $\mathbf{Game}_{i,5} = \mathbf{Game}_{i+1,0}$. Furthermore, indistinguishability between $\mathbf{Game}_{i,j-1}$ and $\mathbf{Game}_{i,j}$ for $j \in [4]$ can be established in exactly the same manners as in Theorem 3.4. For $\mathbf{Game}_{i,4}$ and $\mathbf{Game}_{i,5}$, we give the proof in the following.

Lemma B.2. *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 simulation security of PE^+ using A. B proceeds as follows.

Setup phase. At the beginning of the game, B inputs 1^λ to A and gets (X, Msg, M^*) and 1^{s} from A. Then PE^+ challenger chooses $\mathsf{PE}^+.\mathsf{mpk}$ and sends it to B, where $\mathsf{PE}^+.\mathsf{mpk}$ is honestly chosen or simulated, depending on whether B is playing the real game or simulated game. B then sets $\mathsf{PE}^+.\mathsf{mpk}_i := \mathsf{PE}^+.\mathsf{mpk}$ and chooses $\widehat{\mathsf{K}}_j, \mathsf{R}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ 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 $\mathsf{FE.sk}_i \leftarrow \mathsf{FE.KeyGen}(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i, C_{\mathsf{s},2^i})$. Finally, B sets

$$X_i := \{\hat{\mathbf{x}} = \mathbf{x} \| \perp^{2^i - |\mathbf{x}|} : \mathbf{x} \in X, 2^{i-1} < |\mathbf{x}| \leq 2^i\}$$

and

$$\mathsf{Msg}_i := \{m_j \in \mathsf{Msg} : 2^{i-1} < |\mathbf{x}_j| \leq 2^i\}$$

and submits its target as $(X_i, \mathsf{Msg}_i, \widehat{M}^*_{2^i})$ and $(1^{2^i \eta}, 1^{\hat{\mathsf{d}}}, 1^{\mathsf{s}})$ to the PE^+ challenger, where $\widehat{M}^*_{2^i} = \mathsf{To\text{-}Circuit}_{|M^*|, 2^i}(M^*)$.

After B specifies its target, it is given a secret key $\mathsf{PE}^+.\mathsf{sk}_{U[\widehat{M}^*_{2^i}]}$ and ciphertexts $\{\mathsf{PE}^+.\mathsf{ct}_{\hat{\mathbf{x}}}\}_{\hat{\mathbf{x}} \in X_i}$. They are honestly generated or simulated depending on whether B is playing the real game or simulated game. B then simulates the ciphertexts $\{\mathsf{NfaPE}^+.\mathsf{ct}_{\mathbf{x}}\}_{\mathbf{x} \in X}$ as follows.

Simulating Ciphertexts. To generate a ciphertext for $\mathbf{x} \in X$ associated with $m \in \mathsf{Msg}$, B computes the ciphertext $\mathsf{NfaPE}^+.\mathsf{ct}_{\mathbf{x}}$ as follows. For the case of $\lceil \log |\mathbf{x}| \rceil \neq i$, it proceeds as in the previous game. Otherwise, it retrieves corresponding component $\mathsf{PE}^+.\mathsf{ct}_{\hat{\mathbf{x}}}$ to $\hat{\mathbf{x}}$ from $\{\mathsf{PE}^+.\mathsf{ct}_{\hat{\mathbf{x}}}\}_{\hat{\mathbf{x}} \in X_i}$ and sets $\mathsf{NfaPE}^+.\mathsf{ct}_{\mathbf{x}} = (\mathsf{FE.sk}_i, \mathsf{PE}^+.\mathsf{mpk}_i, \mathsf{PE}^+.\mathsf{ct}_{\hat{\mathbf{x}}})$, where B uses $\mathsf{PE}^+.\mathsf{mpk}_i$ and $\mathsf{FE.sk}_i$ that are sampled in the setup phase.

B also simulates the 1-key $\mathsf{NfaPE}^+.\mathsf{sk}_{M^*} := \{\mathsf{FE.ct}_j\}_{j \in [0,\lambda]}$ from $\mathsf{PE}^+.\mathsf{sk}_{U[\widehat{M}^*_{2^i}]}$ as in Eq. (B.1). B then gives $\mathsf{NfaPE}^+.\mathsf{sk}_{M^*}$ and $\{\mathsf{NfaPE}^+.\mathsf{ct}_{\mathbf{x}}\}_{\mathbf{x} \in X}$ to A and answers the key queries as follows.

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 $j \in [0, \lambda]$. It then queries a secret key for $U[\widehat{M}_{2^i}]$ to its challenger. Then, the challenger returns $\mathsf{PE}^+.\mathsf{sk}_{U[\widehat{M}_{2^i}]}$ to B. The key is honestly generated or simulated depending on whether B is playing the real game or simulated game. It then computes $\mathsf{FE.ct}_j$ for $j \in [0, \lambda]$ as in Eq. (B.1) and returns $\mathsf{NfaPE}^+.\mathsf{sk}_M := \{\mathsf{FE.ct}_j\}_{j \in [0,\lambda]}$ to A.

It is easy to see that B simulates $\mathbf{Game}_{i,4}$ if it is in the real game and $\mathbf{Game}_{i,5}$ if it is in the simulated game. Therefore, B breaks the security of PE^+ 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 3.1. Therefore, B is a legitimate adversary as long as so is A. This completes the proof of the lemma. $\square$

C Reusable Garbled Nondeterministic Finite Automata

In this section, we provide the definition of reusable garbled NFAs.

C.1 Reusable Garbled NFA

In this section, we will define garbled NFAs and notions of input and function privacy, adapting corresponding definitions from [AS17a]. We further show how to construct garbled NFAs (with unbounded inputs) that can be used to evaluate multiple inputs (of possibly varying size).

Definition C.1. (Garbled NFA scheme) A garbling scheme for a family of NFAs $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ with M_λ a family of NFAs $\Sigma_\lambda \times \mathcal{Q}_\lambda \to \mathcal{Q}_\lambda$, is a tuple of PPT algorithms $\mathsf{RGbNFA} = (\mathsf{RGNfa.Garble}$, $\mathsf{RGNfa.Encode}, \mathsf{RGNfa.Eval})$ such that

- $\mathsf{RGbNFA.Setup}(1^\lambda)$ takes as input the security parameter λ and outputs a secret key gsk.
- $\mathsf{RGNfa.Garble}(\mathsf{gsk}, M)$ takes as input a secret key gsk and an NFA $M \in \mathcal{M}_\lambda$ and outputs the garbled NFA M_G.
- $\mathsf{RGNfa.Encode}(\mathsf{gsk}, \mathbf{w})$ takes as input the vector $\mathbf{w} \in \Sigma^*$, the secret key gsk and outputs an encoding c.
- $\mathsf{RGNfa.Eval}(M_G, c)$ takes as input a garbled NFA M_G, an encoding c and outputs 1 iff M accepts $\mathbf{w}$, 0 otherwise.

Definition C.2. (Correctness). For all sufficiently large security parameters λ, for all NFAs $M \in \mathcal{M}_\lambda$ and all $\mathbf{w} \in \Sigma^*$, we have:

$$\Pr\left[\begin{array}{l}\mathsf{gsk} \leftarrow \mathsf{RGbNFA.Setup}(1^\lambda), M_G \leftarrow \mathsf{RGNfa.Garble}(\mathsf{gsk}, M);\\ c \leftarrow \mathsf{RGNfa.Encode}(\mathsf{gsk}, \mathbf{w}); b \leftarrow \mathsf{RGNfa.Eval}(M_G, c) : M(\mathbf{w}) = b\end{array}\right] = 1 - \mathrm{negl}(\lambda)$$

Definition C.3. (Efficiency) There exist universal polynomials $p_1 = p_1(\lambda)$ and $p_2 = p_2(\lambda, \cdot)$ such that for all security parameters λ, for all NFAs $M \in M_\lambda$, for all $\mathbf{w} \in \Sigma^*$,

$$\Pr\left[\begin{array}{l}\mathsf{gsk} \leftarrow \mathsf{RGbNFA.Setup}(1^\lambda) :\\ |\mathsf{gsk}| \le p_1(\lambda) \text{ and runtime } (\mathsf{RGNfa.Encode}(\mathsf{gsk}, \mathbf{w})) \le p_2(\lambda, |\mathbf{w}|)\end{array}\right] = 1.$$

Definition C.4. (Input and machine privacy with reusability) Let RGbNFA be a garbling scheme for a family of NFAs $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$. For a pair of stateful PPT algorithms A and a PPT simulator Sim, consider the following two experiments:

```
Exp^{real}_{RGbNFA,A}(1^λ):
1: $M \leftarrow \mathsf{A}(1^\lambda)$
2: $\mathsf{gsk} \leftarrow \mathsf{RGbNFA.Setup}(1^\lambda)$
   $M_G \leftarrow \mathsf{RGNfa.Garble}(\mathsf{gsk}, M)$
3: $\alpha \leftarrow \mathsf{A}^{\mathsf{RGNfa.Encode}(\mathsf{gsk},\cdot)}(M, M_G)$
4: Output $\alpha$

Exp^{ideal}_{RGbNFA,A,Sim}(1^λ):
1: $M \leftarrow \mathsf{A}(1^\lambda)$
2: $\tilde{M}_G \leftarrow \mathrm{Sim}(1^\lambda, 1^{|M|})$
3: $\alpha \leftarrow \mathsf{A}^{\mathsf{O}(\cdot,M)}(M, \tilde{M}_G)$
4: Output $\alpha$
```

Here, $\mathsf{O}(\cdot, M)$ is an oracle that on input $\mathbf{w}$ from A, runs Sim with inputs $M(\mathbf{w}), 1^{|\mathbf{w}|}$, and the latest state of Sim; it returns the output of Sim (Note that Sim updates and maintains its internal states upon its invocation, since it is a stateful algorithm.).

A garbling scheme RGbNFA is input and machine private with reusability if there exists a PPT simulator Sim such that for all pairs of PPT adversaries A, the following two distributions are computationally indistinguishable:

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

Selective Simulation Security. We can consider a weaker version of the above security notion where $\mathcal{A}$ outputs a set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|X|}\} \subset \Sigma^*$ along with M at the beginning of the game and $\mathcal{A}$ is only allowed to query $\mathbf{x} \in X$ to $\mathsf{RGNfa.Encode}(\mathsf{gsk}, \cdot)$ and $\mathsf{O}(\cdot, M)$. We call this security notion selective simulation security.

D Construction: Reusable Garbled NFA

We first note that one could consider replacing the underlying PE^+ scheme in Section B with a Reusable Garbled circuit scheme, RGC to obtain a construction of Reusable Garbled NFA. Following the previous template of our constructions from Sec. 3.2 and Sec. B, this makes the circuit supported by the underlying FE scheme to work as follows.

1. Convert the input NFA machine M to an equivalent circuit $\widehat{M}_{2^i}$ that can handle inputs of length 2^i.
2. Compute an RGC secret key to encode $\widehat{M}_{2^i}$ and output this as its garbled version.

This intuitive conversion fails to work since the resulting encoding as the garbled machine cannot be shorter than the circuit description length $|\widehat{M}_{2^i}|$, while we need it to be independent of equivalent circuit size. At a high level, to avoid this problem and still ensure machine privacy, we will encrypt M under a symmetric key encryption scheme, SKE. Further, we will use another FE scheme (detailed below) with suitable efficiency guarantees to decrypt this SKE ciphertext encoding M and then run its equivalent circuit description on the encoded input to obtain the desired output.

To this end, we use 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. $\mathsf{SKE} = (\mathsf{SKE.Setup}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$: a secret key encryption, where the length of a secret key $\mathsf{S} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ is denoted by $|\mathsf{S}|$. For the sake of concreteness and simplicity, we assume that the ciphertext length encrypting a message of length s is $\mathsf{s} + \lambda$. Furthermore, we assume that the depth of the decryption circuit is bounded by some fixed polynomial $\mathrm{poly}(\lambda)$ that is independent of the length of the message. These properties can be easily achieved by using PRF for instance.
3. $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$: a functional encryption scheme for circuit with the efficiency property described in Item 1 of Theorem 2.15. We can instantiate FE by the scheme proposed by Goldwasser et. al [GKP+13].
4. $\widehat{\mathsf{FE}} = (\widehat{\mathsf{FE}}.\mathsf{Setup}, \widehat{\mathsf{FE}}.\mathsf{KeyGen}, \widehat{\mathsf{FE}}.\mathsf{Enc}, \widehat{\mathsf{FE}}.\mathsf{Dec})$: a functional encryption scheme for circuit with the efficiency properties described in Theorem A.4. We can instantiate $\widehat{\mathsf{FE}}$ with the PE^+ scheme proposed by Agrawal [Agr17].[7] Alternatively, we can instantiate $\widehat{\mathsf{FE}}$ by the scheme proposed by Goldwasser et. al with the underlying ABE scheme used in their construction being instantiated by the scheme by Boneh et. al [BGG+14]. In both cases, the scheme will only have selective simulation security rather than full simulation security. See Remark 2.14 for the definition of selective simulation security and Remark 2.10 for the reason why we only have *selective* simulation secure FE scheme with the required efficiency properties.

The reusable garbled NFA scheme is defined as follows. The encode algorithm here needs the size s of NFA. This requirement will be removed later using similar technique to Sec. 4.

[7]Though PE^+ has different syntax from functional encryption, it is easy to convert the former into the latter. For example, we encrypt a random message under an attribute $\mathbf{x}$ using PE^+ and append the message to the PE^+ ciphertext to form a functional encryption of a message $\mathbf{x}$. To decrypt the ciphertext, we use PE^+ secret key to decrypt the PE^+ ciphertext and output 1 if it corresponds to the appended message and 0 otherwise.

$\mathsf{RGbNFA.Setup}(1^\lambda)$: Upon input the security parameter, sample PRF keys $\mathsf{K}_j, \widehat{\mathsf{K}}_j, \mathsf{R}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ and SKE key $\mathsf{S}_j \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ for all $j \in [0,\lambda]$. Then, output $\mathsf{gsk} = (\{\widehat{\mathsf{K}}_j, \mathsf{K}_j, \mathsf{R}_j, \mathsf{S}_j\}_{j\in[0,\lambda]})$.

$\mathsf{RGNfa.Garble}(\mathsf{gsk}, M)$: Upon input a secret key gsk and an NFA machine M of size $|M| = \mathsf{s}$, do the following:

1. For all $j \in [0,\lambda]$, sample $(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{d}(\lambda)}, 1^{\mathsf{out}(\lambda)}; \mathsf{K}_j)$.

   Here, we generate $\lambda + 1$ instances of FE. Note that all instances support a circuit class with input length $\mathsf{inp}(\lambda) = \mathsf{s} + \lambda + 2|\mathsf{K}|$, output length $\mathsf{out}(\lambda)$, and depth $\mathsf{d}(\lambda)$, where $\mathsf{out}(\lambda)$ and $\mathsf{d}(\lambda)$ are polynomials in the security parameter as in Section 3.
2. Sample $\widehat{\mathsf{R}}_j \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ for all $j \in [0,\lambda]$.
3. Compute $\mathsf{SKE.ct}_j \leftarrow \mathsf{SKE.Enc}(\mathsf{S}_j, M)$ for all $j \in [0,\lambda]$.
4. Compute $\mathsf{FE.ct}_j \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_j, (\mathsf{SKE.ct}_j, \widehat{\mathsf{K}}_j, \widehat{\mathsf{R}}_j))$ for all $j \in [0,\lambda]$.
5. Output garbled NFA $M_G = (\{\mathsf{SKE.ct}_j, \mathsf{FE.mpk}_j, \mathsf{FE.ct}_j\}_{j\in[0,\lambda]})$.

$\mathsf{RGNfa.Encode}(\mathsf{gsk}, \mathbf{x}, 1^{\mathsf{s}})$: Upon input a secret key gsk, a vector $\mathbf{x}$, and the size of NFA 1^{s} do the following:

1. Parse gsk as $\mathsf{gsk} \rightarrow (\{\widehat{\mathsf{K}}_j, \mathsf{K}_j, \mathsf{R}_j, \mathsf{S}_j\}_{j\in[0,\lambda]})$.
2. Set $\hat{\mathbf{x}} = \mathbf{x} \| \bot^{2^i - \ell}$, where $\ell = |\mathbf{x}|$ and $i = \lceil \log \ell \rceil$.
3. Sample $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{d}(\lambda)}, 1^{\mathsf{out}(\lambda)}; \mathsf{K}_i)$.
4. Sample $(\widehat{\mathsf{FE}}.\mathsf{mpk}_i, \widehat{\mathsf{FE}}.\mathsf{msk}_i) = \widehat{\mathsf{FE}}.\mathsf{Setup}(1^\lambda, 1^{2^i\eta}, 1^{\hat{\mathsf{d}}(\lambda)}, 1^1; \widehat{\mathsf{K}}_i)$. Note that this FE supports a circuit class with input domain $\{0,1\}^{2^i\eta} \supseteq (\Sigma \cup \{\bot\})^{2^i}$, single bit output, and depth $\hat{\mathsf{d}}$.
5. Compute $\widehat{\mathsf{FE}}.\mathsf{ct} \leftarrow \widehat{\mathsf{FE}}.\mathsf{Enc}(\widehat{\mathsf{FE}}.\mathsf{mpk}_i, (\mathsf{S}_i, \hat{\mathbf{x}}))$.
6. 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 Figure 6.

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

(a) Parse the input $\mathbf{w} = (\mathsf{SKE.ct}, \widehat{\mathsf{K}}, \widehat{\mathsf{R}})$, where SKE.ct is an SKE ciphertext and $\widehat{\mathsf{K}}$ and $\widehat{\mathsf{R}}$ are PRF keys.

(b) Compute $(\widehat{\mathsf{FE}}.\mathsf{mpk}, \widehat{\mathsf{FE}}.\mathsf{msk}) = \widehat{\mathsf{FE}}.\mathsf{Setup}(1^\lambda, 1^{2^i\eta}, 1^{\hat{\mathsf{d}}}; \widehat{\mathsf{K}})$

(c) Compute and output $\widehat{\mathsf{FE}}.\mathsf{sk} = \widehat{\mathsf{FE}}.\mathsf{KeyGen}(\widehat{\mathsf{FE}}.\mathsf{mpk}, \widehat{\mathsf{FE}}.\mathsf{msk}, D_{\mathsf{s},2^i}[\mathsf{SKE.ct}]; \widehat{\mathsf{R}})$. (See Figure 7 for the definition of $D_{\mathsf{s},2^i}[\mathsf{SKE.ct}]$)

Figure 6

7. Output $c = (\widehat{\mathsf{FE}}.\mathsf{mpk}_i, \mathsf{FE.sk}_i, \widehat{\mathsf{FE}}.\mathsf{ct})$.

$\mathsf{RGNfa.Eval}(M_G, c)$: Upon input the reusable garbled NFA M_G and the input encoding c, do the following:

1. Parse the garbled machine as $M_G \rightarrow (\{\mathsf{SKE.ct}_j, \mathsf{FE.mpk}_j, \mathsf{FE.ct}_j\}_{j\in[0,\lambda]})$ and the encoding as $c \rightarrow (\widehat{\mathsf{FE}}.\mathsf{mpk}_i, \mathsf{FE.sk}_i, \widehat{\mathsf{FE}}.\mathsf{ct})$.
2. Set $\ell = |\mathbf{x}|$ and choose $\mathsf{FE.ct}_i$ such that $i = \lceil \log \ell \rceil < \lambda$.
3. Compute $y = \mathsf{FE.Dec}(\mathsf{FE.mpk}_i, \mathsf{FE.sk}_i, C_{\mathsf{s},2^i}, \mathsf{FE.ct}_i)$.

Function $D_{\mathsf{s},2^i}[\mathsf{SKE.ct}]$

(a) Parse the input $\mathbf{w} = (\mathsf{S}, \hat{\mathbf{x}})$, where S is an SKE secret key.

(b) Compute $M = \mathsf{SKE.Dec}(\mathsf{S}, \mathsf{SKE.ct})$.

(c) Compute $\widehat{M}_{2^i} = \mathsf{To\text{-}Circuit}_{\mathsf{s},2^i}(M)$. (See Theorem 3.1 for the definition of To-Circuit.)

(d) Compute and output $\widehat{M}_{2^i}(\hat{\mathbf{x}})$. (This part of the computation is implemented by $U[\widehat{M}_{2^i}]$ where the universal circuit U is instantiated by [CH85].)

Figure 7

4. Construct $D_{\mathsf{s},2^i}[\mathsf{SKE.ct}_i]$ from $\mathsf{SKE.ct}_i$.
5. Compute and output $z = \mathsf{FE.Dec}(\widehat{\mathsf{FE}}.\mathsf{mpk}_i, y, D_{\mathsf{s},2^i}[\mathsf{SKE.ct}_i], \widehat{\mathsf{FE}}.\mathsf{ct})$, where we interpret y as a secret key of the underlying FE.

Correctness. Correctness of the scheme follows from the correctness of FE and $\widehat{\mathsf{FE}}$ as in Section 3.4 if we appropriately set $\hat{\mathsf{d}}$, out, and d. Here, we give a brief explanation. We first observe that $D_{\mathsf{s},2^i}[\mathsf{SKE.ct}_i]$ can be implemented by combining the decryption circuit of SKE, $\mathsf{To\text{-}Circuit}_{\mathsf{s},2^i}$, and $U[\widehat{M}_{2^i}]$. The depth of the first circuit (resp., the latter two circuits) can be bounded by some fixed polynomial, which is in particular independent of 2^i, by our assumption on SKE (resp., by Theorem 3.1). Therefore, the depth of the overall circuit $D_{\mathsf{s},2^i}[\mathsf{SKE.ct}_i]$ can be bounded by some fixed polynomial. We would set $\hat{\mathsf{d}}(\lambda)$ to be larger than this polynomial so that we can invoke the correctness of $\widehat{\mathsf{FE}}$. By our assumption on the secret key size of $\widehat{\mathsf{FE}}$, we can bound the length of $\widehat{\mathsf{FE}}.\mathsf{sk}$ that is output by $C_{\mathsf{s},2^i}$ by a polynomial in $\hat{\mathsf{d}}$ and λ, which can be bounded by some fixed polynomial in λ. We would set $\mathsf{out}(\lambda)$ to be larger than this polynomial. Furthermore, we can see that the depth of $C_{\mathsf{s},2^i}$ can be bounded by some fixed polynomial by the assumptions we posed on the depth of the setup and key generation circuits of $\widehat{\mathsf{FE}}$ and by the fact that the depth of $D_{\mathsf{s},2^i}[\mathsf{SKE.ct}]$ can be bounded by some fixed polynomial. We would set $\mathsf{d}(\lambda)$ to be larger than this polynomial. Since the depth and output length of $C_{\mathsf{s},2^i}$ are bounded by d and out respectively, the circuit $C_{\mathsf{s},2^i}$ is supported by the scheme and thus we can invoke the correctness of FE. We therefore have $y = C_{\mathsf{s},2^i}(\mathsf{SKE.ct}_i, \mathsf{K}_i, \mathsf{R}_i) = \widehat{\mathsf{FE}}.\mathsf{KeyGen}(\widehat{\mathsf{FE}}.\mathsf{msk}_i, D_{\mathsf{s},2^i}[\mathsf{SKE.ct}_i]; \widehat{\mathsf{R}}_i)$ by the correctness of FE and $z = D_{\mathsf{s},2^i}[\mathsf{SKE.ct}_i](\mathsf{S}_i, \hat{\mathbf{x}}) = \widehat{M}_{2^i}(\hat{\mathbf{x}}) = M(\mathbf{x})$ by the correctness of $\widehat{\mathsf{FE}}$ and SKE.

Security. Here, we prove that RGbNFA defined above is secure, if so is FE. Formally, we have the following theorem.

Theorem D.1. *Assume that* FE *satisfies full simulation based security,* $\widehat{\mathsf{FE}}$ *satisfies selective simulation security, and that* PRF *is a secure pseudorandom function. Then,* NfaABE *satisfies selective simulation security.*

Proof. Security follows analogously to that of Theorem B.1. The main difference is that the PE^+ simulator is replaced by the FE simulator and we additionally invoke the security of SKE to achieve the function privacy. Other differences are that we have to change $(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j)$ to be sampled using true randomness instead of K_j using the security of the PRF and we introduce the step where we undo the changes added during the hybrid games similarly to the proof of Theorem 3.4. Below, we give the hybrid games to prove the security. In more detail, we consider $\mathbf{Game}_i$ for $i \in [0, \lambda + 1]$ as follows.

$\mathbf{Game}_i$**:** The game proceeds as follows. In the following FE.Sim is the simulator for FE.

Setup. At the beginning of the game, the challenger takes 1^λ as input and samples $\{\mathsf{K}_j, \widehat{\mathsf{K}}_j, \mathsf{R}_j, \mathsf{S}_j\}_{j\in[0,\lambda]}$. It also computes $(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{\mathsf{inp}(\lambda)}, 1^{\mathsf{d}(\lambda)}, 1^{\mathsf{out}(\lambda)}; \mathsf{K}_j)$ for $j \in [0, \lambda]$.

The challenger answers the queries made by A as follows.

Garbling the NFA. A takes 1^λ as input and submits an NFA machine M of size s and a set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|X|}\} \subset \Sigma^*$ to the challenger. The challenger computes M_G as follows. The challenger computes $\mathsf{SKE.ct}_j$ and $\mathsf{FE.ct}_j$ for $j \in [0, \lambda]$ as

$$\mathsf{FE.ct}_j \leftarrow \mathsf{FE.Enc}(\mathsf{FE.mpk}_j, (\mathsf{SKE.ct}_j, \widehat{\mathsf{K}}_j, \widehat{\mathsf{R}}_j)), \quad \mathsf{SKE.ct}_j \leftarrow \begin{cases} \mathsf{SKE.Enc}(\mathsf{S}_j, M) & \text{If } \lambda \geq j \geq i \\ \mathsf{SKE.Enc}(\mathsf{S}_j, 0^{\mathsf{s}}) & \text{If } j \leq i-1, \end{cases}$$

and returns $M_G = (\{\mathsf{SKE.ct}_j, \mathsf{FE.mpk}_j, \mathsf{FE.ct}_j\}_{j\in[0,\lambda]})$ to A.

Simulating Encodings. To generate an encoding for $\mathbf{x} \in X$, the challenger sets $j := \lceil \log |\mathbf{x}| \rceil$ and generates $(\widehat{\mathsf{FE}}.\mathsf{mpk}_j, \widehat{\mathsf{FE}}.\mathsf{msk}_j) = \widehat{\mathsf{FE}}.\mathsf{Setup}(1^\lambda, 1^{2^j\eta}, 1^{\hat{\mathsf{d}}(\lambda)}, 1^1; \widehat{\mathsf{K}}_j)$. It then computes

$$\widehat{\mathsf{FE}}.\mathsf{ct} \leftarrow \begin{cases} \widehat{\mathsf{FE}}.\mathsf{Enc}(\widehat{\mathsf{FE}}.\mathsf{mpk}_j, (\mathsf{S}_j, \hat{\mathbf{x}})) & \text{If } \lambda \geq j \geq i \\ \widehat{\mathsf{FE}}.\mathrm{Sim}(\widehat{\mathsf{FE}}.\mathsf{mpk}_j, \widehat{\mathsf{FE}}.\mathsf{sk}_{D_{\mathsf{s},2^j}[\mathsf{SKE.ct}_j]}, D_{\mathsf{s},2^j}[\mathsf{SKE.ct}_j], M(\mathbf{x})) & \text{If } j \leq i-1 \end{cases}, \tag{D.1}$$

where $\widehat{\mathsf{FE}}.\mathsf{sk}_{D_{\mathsf{s},2^j}[\mathsf{SKE.ct}_j]} \leftarrow \widehat{\mathsf{FE}}.\mathsf{KeyGen}(\widehat{\mathsf{FE}}.\mathsf{msk}_j, D_{\mathsf{s},2^j}[\mathsf{SKE.ct}_j]; \widehat{\mathsf{R}}_j)$. It also computes $\mathsf{FE.sk}_j = \mathsf{FE.KeyGen}(\mathsf{FE.mpk}_j, \mathsf{FE.msk}_j, C_{\mathsf{s},2^j}; \mathsf{R}_j)$. The encoding of $\mathbf{x}$ is $c = (\widehat{\mathsf{FE}}.\mathsf{mpk}_j, \mathsf{FE.sk}_j, \widehat{\mathsf{FE}}.\mathsf{ct})$.

Finally, A outputs its guess b'.

It is easy to see that $\mathbf{Game}_0$ is the same as $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{RGbNFA},\mathsf{A}}(1^\lambda)$. Furthermore, we can construct a simulator for RGbNFA from the challenger in $\mathbf{Game}_{\lambda+1}$ appropriately, since the challenger in $\mathbf{Game}_{\lambda+1}$ only uses s and $|\mathbf{x}|$ to simulate a garbled NFA and an encoding, respectively. Therefore, it suffices to show the indistinguishability between $\mathbf{Game}_i$ and $\mathbf{Game}_{i+1}$. To do so, we consider two cases separately depending on whether $i \leq i_{\max}$ or not, where $i_{\max}$ is defined as in the proof of Theorem 3.4.

We first consider the case of $i \geq i_{\max} + 1$. In this case, the only difference between $\mathbf{Game}_i$ and $\mathbf{Game}_{i+1}$ is the way $\mathsf{SKE.ct}_i$ is generated, since the upper branch of Equation (D.1) is never triggered when answering the encoding query because of the definition of $i_{\max}$. The indistinguishability of the two games immediately follows from the security of SKE since S_i is never used except when generating $\mathsf{SKE.ct}_i$.

We then consider the case of $i \leq i_{\max}$. The proof for this case closely follows the proof of Theorem B.1 except for we some changes that we explained at the beginning of the proof.

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

$\mathbf{Game}_{i,1}$: The game is the same as the previous game except that $\mathsf{FE.sk}_i = \mathsf{FE.KeyGen}(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i, C_{\mathsf{s},2^i}; \mathsf{R}_i)$ and $(\widehat{\mathsf{FE}}.\mathsf{mpk}_i, \widehat{\mathsf{FE}}.\mathsf{msk}_i) = \widehat{\mathsf{FE}}.\mathsf{Setup}(1^\lambda, 1^{2^i\eta}, 1^{\hat{\mathsf{d}}(\lambda)}, 1^1; \widehat{\mathsf{K}}_i)$ are computed at the setup phase.

$\mathbf{Game}_{i,2}$: The game is the same as the previous game except that $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i)$ and $\mathsf{FE.sk}_i$ are generated using true randomness instead of using the PRF keys.

$\mathbf{Game}_{i,3}$: In this game, to answer a garbling query, $\mathsf{FE.ct}_i$ is computed as

$$\mathsf{FE.ct}_i \leftarrow \mathsf{FE.Sim}(\mathsf{FE.mpk}_i, \mathsf{FE.sk}_i, C_{\mathsf{s},2^i}, \widehat{\mathsf{FE}}.\mathsf{sk}_{D_{\mathsf{s},2^i}[\mathsf{SKE.ct}_i]}, 1^{\mathsf{inp}(\lambda)})$$

where $\widehat{\mathsf{FE}}.\mathsf{sk}_{D_{\mathsf{s},2^i}[\mathsf{SKE.ct}_i]} \leftarrow \widehat{\mathsf{FE}}.\mathsf{KeyGen}(\widehat{\mathsf{FE}}.\mathsf{msk}_i, D_{\mathsf{s},2^i}[\mathsf{SKE.ct}_i]; \widehat{\mathsf{R}}_i)$.

$\mathbf{Game}_{i,4}$: In this game, to answer a garbling query, $(\widehat{\mathsf{FE}}.\mathsf{mpk}_i, \widehat{\mathsf{FE}}.\mathsf{msk}_i)$ and $\widehat{\mathsf{FE}}.\mathsf{sk}_{D_{\mathsf{s},2^i}[\mathsf{SKE.ct}_i]}$ are generated using true randomness instead of using the PRF keys $\widehat{\mathsf{K}}_i$ and $\widehat{\mathsf{R}}_i$.

$\mathbf{Game}_{i,5}$: In this game, to answer an encoding query, we generate

$$\widehat{\mathsf{FE}}.\mathsf{ct} \leftarrow \widehat{\mathsf{FE}}.\mathrm{Sim}(\widehat{\mathsf{FE}}.\mathsf{mpk}_i, \widehat{\mathsf{FE}}.\mathsf{sk}_{D_{\mathsf{s},2^i}[\mathsf{SKE.ct}_i]}, D_{\mathsf{s},2^i}[\mathsf{SKE.ct}_i], M(\mathbf{x}))$$

instead of honestly generating it.

$\mathbf{Game}_{i,6}$: In this game, $\mathsf{SKE.ct}_i$ is changed to be $\mathsf{SKE.Enc}(\mathsf{S}_i, 0^{|M|})$.

$\mathbf{Game}_{i,7}$: In this game, we undo the changes we added from $\mathbf{Game}_{i,0}$ to $\mathbf{Game}_{i,4}$. Namely, we generate $(\widehat{\mathsf{FE}}.\mathsf{mpk}_i, \widehat{\mathsf{FE}}.\mathsf{msk}_i)$ by using $\widehat{\mathsf{K}}_i, \widehat{\mathsf{R}}_i$, $\mathsf{FE.ct}_i$ by honestly encrypting $(\mathsf{SKE.ct}_i, \widehat{\mathsf{K}}_i, \widehat{\mathsf{R}}_i)$ (but $\mathsf{SKE.ct}_i$ is still an encryption of $0^{|M|}$), and $(\mathsf{FE.mpk}_i, \mathsf{FE.msk}_i)$ by using $\mathsf{K}_i, \mathsf{R}_i$.

The indistinguishability between $\mathbf{Game}_{i,j-1}$ and $\mathbf{Game}_{i,j}$ for $j \in [5]$ follows similarly to the proof of Theorem B.1, except that we use the security of $\widehat{\mathsf{FE}}$ rather than PE^+ when moving from $\mathbf{Game}_{i,4}$ to $\mathbf{Game}_{i,5}$. The indistinguishability between $\mathbf{Game}_{i,5}$ and $\mathbf{Game}_{i,6}$ follows from the security of SKE, since S_i is never used except when generating $\mathsf{SKE.ct}_i$ in these games even if the upper branch of Equation (D.1) is triggered, due to the change we added in $\mathbf{Game}_{i,5}$. The indistinguishability between $\mathbf{Game}_{i,6}$ and $\mathbf{Game}_{i,7}$ can be shown by repeating the same argument for showing $\mathbf{Game}_{i,0} \overset{c}{\approx} \mathbf{Game}_{i,4}$ in the reverse order. Finally, we note that $\mathbf{Game}_{i,7}$ is equivalent to $\mathbf{Game}_{i+1}$. These imply that $\mathbf{Game}_i$ and $\mathbf{Game}_{i+1}$ are indistinguishable, which completes the proof of the theorem. □

Efficiency. In the above construction, the efficiency requirement (Definition C.3) is not satisfied, since the encoding algorithm constructs $C_{\mathsf{s},2^i}$ whose size is polynomially dependent on the size of NFA M. This is problematic when $|M| \gg |\mathbf{x}|$. To resolve the issue, we use the same idea as that we used in Sec. 4. Namely, we combine our RGC construction above with the one that poses upper-bound on the input length [GKP+13]. Namely, when we garble an NFA with size $|M|$, we convert M into an equivalent circuit $\widehat{M}_{|M|,j}$ with input length j for all $j \in [|M|]$ and then garble all of them using [GKP+13]. In addition, we garble M with the above scheme with $\mathsf{s} = |M|$, which supports unbounded length. We do this by deriving randomness from a single PRF key and therefore gsk is compact. To encode $\mathbf{x}$, we encode it with the above scheme with different s in parallel. Namely, we encode $\mathbf{x}$ by the above scheme with all of $\mathsf{s} \leq [|\mathbf{x}|]$. Furthermore, we also choose $|\mathbf{x}|$-th instance of the RGC of [GKP+13] and encodes it. Similarly to the construction in Sec. 4, evaluation algorithm first sees if $|\mathbf{x}| > |M|$ and uses the above scheme to decode if so. Otherwise, it uses the bounded input scheme [GKP+13]. It can be seen that the encoding algorithm runs in polynomial time in $|\mathbf{x}|$ independent of $|M|$. Furthermore, the security of the scheme is preserved, since the construction simply runs secure schemes in parallel.

Generalizing to Bounded Keys. We note that by replacing the inner scheme with a bounded key FE scheme [GVW12, AR17], the construction immediately generalizes to support bounded number of (reusable) garbled circuits.