Reusable Garbled Deterministic Finite Automata from Learning With Errors*

Shweta Agrawal[1] and Ishaan Preet Singh[2]

1 **IIT Madras, Chennai, India**
shweta@iitm.ac.in
2 **IIT Delhi, New Delhi, India**
ishaanps92@gmail.com

Abstract

We provide a single-key functional encryption scheme for Deterministic Finite Automata (DFA). The secret key of our scheme is associated with a DFA M, and a ciphertext is associated with an input $\mathbf{x}$ of *arbitrary* length. The decryptor learns $M(\mathbf{x})$ and nothing else. The ciphertext and key sizes achieved by our scheme are optimal – the size of the public parameters is independent of the size of the machine or data being encrypted, the secret key size depends only on the machine size and the ciphertext size depends only on the input size.

Our scheme achieves full functional encryption in the "private index model", namely the entire input $\mathbf{x}$ is hidden (as against $\mathbf{x}$ being public and a single bit b being hidden). Our single key FE scheme can be compiled with symmetric key encryption as in [12] to yield reusable garbled DFAs for arbitrary size inputs, that achieves machine and input privacy along with reusability under a strong simulation based definition of security.

We generalize this to a functional encryption scheme for Turing machines TMFE which has short public parameters that are independent of the size of the machine or the data being encrypted, short function keys, and input-specific decryption time. However, the ciphertext of our construction is large and depends on the worst case running time of the Turing machine (but not its description size). These provide the first FE schemes that support unbounded length inputs, allow succinct public and function keys and rely on LWE.

Our construction relies on a new and arguably natural notion of *decomposable functional encryption* which may be of independent interest.

1998 ACM Subject Classification E.3 Data Encryption

Keywords and phrases Functional Encryption, Learning With Errors, Deterministic Finite Automata, Garbled DFA

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.36

1 Introduction

Functional encryption permits *controlled disclosure* of encrypted data, enabling the evaluator to learn some authorised function of encrypted data in the clear. In functional encryption (FE), a secret key corresponds to a function f and ciphertexts correspond to strings from the domain of f. Given a function SK_f and a ciphertext $\mathsf{CT}_{\mathbf{x}}$, the decryptor learns $f(\mathbf{x})$ and nothing else. Functional encryption has found diverse applications, such as spam filtering on encrypted data [12], online dating [13], delegation of computation [16] and many others.

* A full version of the paper is available at http://www.cse.iitm.ac.in/~shwetaag/papers/dfa.pdf [1].

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 36; pp. 36:1–36:13
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The function embedded within the secret key in FE is typically represented as a circuit. While circuits are a powerful model of computation, the circuit representation has significant drawbacks in practical scenarios. Consider the application of spam filtering on encrypted emails, where the email gateway may be given a key to test the incoming email for spam. Representing the computation as a circuit forces emails to be of a fixed length – a requirement which is ill-fitting and wasteful. Another significant drawback of the circuit model is that it incurs worst case running time on every input.

In practice, most spam filters as well as malware and intrusion detection systems are implemented using pattern matching operations represented as deterministic finite automata (DFA) [19, 14, 5, 10]. Note that in all these applications, the size of the input is highly variable and instance specific, and restricting it to be of fixed length is cumbersome. Therefore a functional encryption scheme for DFAs which supports dynamic data length would be the "right fit" in such situations. However, although functional encryption for circuits has been constructed based on the hardness of Learning With Errors (LWE) in the single key setting, it is unclear how to leverage these techniques to support the arbitrary data length required by DFAs.

1.1 Our Results

In this work, we provide a single-key functional encryption scheme for Deterministic Finite Automata (DFA). The secret key of our scheme is associated with a DFA M, and a ciphertext is associated with an input $\mathbf{x}$ of *arbitrary* length. The decryptor learns $M(\mathbf{x})$ and nothing else. The ciphertext and key sizes achieved by our scheme are optimal[1] – the public key size is independent of the machine and input size, the secret key size depends only on the machine size and the ciphertext size depends only on the input size.

Our scheme achieves full functional encryption in the "private index model", namely the entire input $\mathbf{x}$ is hidden (as against $\mathbf{x}$ being public and a single bit b being hidden). Our single key FE scheme can be compiled with symmetric key encryption as in [12] to yield reusable garbled DFAs for arbitrary size inputs, that achieves machine and input privacy along with reusability under a strong simulation based definition of security.

We generalize this to a functional encryption scheme for Turing machines TMFE which has short public parameters that are independent of the size of the machine or the data being encrypted, short function keys, and input-specific decryption time. However, the ciphertext of our construction is large and depends on the worst case running time of the Turing machine (but not its description size). These provide the first FE schemes that support unbounded length inputs, allow succinct public and function keys and rely on LWE.

Our construction relies on a new and arguably natural notion of *decomposable functional encryption* which may be of independent interest.

1.2 Related Work

Functional encryption for DFAs has received some attention already. Closest to our work is the "Attribute Based Encryption" scheme for DFAs constructed by Waters [20]. In [20], the encrypt algorithm takes as input a pair $(\mathbf{x}, b)$ where $\mathbf{x}$ may be of arbitrary size, and b is a bit. The key corresponds to a DFA machine M so that given a key for M and a ciphertext for $(\mathbf{x}, b)$, the decryptor learns the bit b if and only if M accepts $\mathbf{x}$. Note that in contrast to

[1] Up to logarithmic factors.

our work, the vector $\mathbf{x}$ is not hidden by the construction, neither is the machine M; only the bit b is hidden. On the other hand, the construction [20] can support polynomially many keys, whereas ours can only support a single key. Attrapadung [4] extended the work of Waters [20] to achieve adaptive rather than selective security. Another work that constructs Attribute Based Encryption for DFAs is by Boyen and Li [7]. However, in their construction, the input size to the DFA must be bounded in advance; avoiding this restriction is the main motivation for our work.

There are other known functional encryption systems that support unbounded size inputs, even supporting Turing machines, achieving input specific runtime and dynamic data length [11, 2, 6, 15, 8, 9]. However, the mildest assumption required by this line of work is the existence of indistinguishability obfuscation.

From standard assumptions, single key functional encryption has been constructed for all polynomial sized circuits [18, 12]. A natural approach to construct reusable garbled DFA/TM then, is to convert the machine to a circuit and leverage the constructions of [18, 12]. However, instantiating this compiler with the reusable garbled circuits construction [12] leads to a construction that cannot support dynamic data lengths, which is the main focus of this work. On the other hand, using the construction by Sahai and Seyalioglu [18] leads to a DFA/TM FE construction with large public key and ciphertext size, since the construction by [18] suffers from public key and ciphertext size that depend on the circuit size.

1.3 Our Techniques

To begin, we describe our single key FE scheme for DFA. Next, we describe how this construction may be generalized to Turing machines.

1.3.1 Single Key FE for DFA

We briefly recall how a DFA works. A DFA machine M is represented by the tuple $(Q, \Sigma, T, q_{st}, F)$ where Q is a finite set of states, Σ is a finite alphabet, $T : \Sigma \times Q \to Q$ is the transition function, q_{st} is the start state, $F \subseteq Q$ is the set of accepting states. Upon input $\mathbf{w} \in \Sigma^k$ for some arbitrary polynomial k, the machine M accepts $\mathbf{w}$ if and only if there exists a sequence of states $q_1, \ldots, q_k$ so that $q_1 = q_{st}$, $T(w_i, q_i) = q_{i+1}$ for $i \in [k-1]$, and $q_k \in F$.

To mimic the DFA computation, a natural starting point is to imagine a function key that stores the transition table of a DFA, receives as input the current (symbol, state) pair and produces as output an encryption of the next state of the computation. In more detail, say the encryptor provides encryptions of each input symbol x_i, for $i \in [|\mathbf{x}|]$, in addition to an encryption for the first (fixed) state q_{st}. Now, the function key could accept 2 inputs (x_1, q_{st}), lookup the transition table and produce an encryption of the next state q_2. Suppose this encryption can only be paired with the encryption of x_2 and none other, then we could provide (x_2, q_2) as input to the function in the next step, thus propagating the computation.

We tie together encryptions of symbol with encryptions of state via the notion of *decomposable functional encryption*. Intuitively, decomposability requires that the public key PK and the ciphertext $\mathsf{CT}_{\mathbf{y}}$ of a functional encryption scheme be decomposable into components PK_j and CT_j for $j \in [|\mathbf{y}|]$, where CT_j depends on a single deterministic bit y_j and the public key component PK_j. All components CT_j are tied together by *common randomness* used for their creation, although each CT_j may use additional independent randomness. Aside from the message dependent components, a ciphertext can contain components that are independent of the message and depend only on the common randomness. The main advantage offered by decomposable functional encryption is that given the common randomness, each ciphertext component CT_j can be constructed independently of the rest. These components can then be joined together to create a complete ciphertext which can then be decrypted successfully. Additionally, only components that were constructed using the same randomness can be "joined", thereby preventing mix and match attacks where an adversary tries to treat mismatched symbol state pairs such (x_3, q_2) as a single legitimate input.

Now, suppose we have a decomposable functional encryption scheme for circuits. Then, we may proceed with the aforementioned strategy and divide the ciphertext into two components – the first encoding the current symbol, and the second encoding the current state. We may use the function key to generate the second component, *using the same common randomness that was used to generate the first component.*

To take this approach forward we must find a suitable decomposable functional encryption scheme for circuits – fortunately most functional encryption schemes in the literature are decomposable. In particular, we show that the the succinct single key FE by Goldwasser et al. [12] is decomposable. This scheme appears suitable for our purposes as the ciphertext and public key in this scheme are independent of circuit size.

However, note that the ciphertext of [12] suffers from *output-size dependence*, i.e. it grows linearly with the output length of the circuit. This implies that the function key may not produce an output that is proportional to the length of the ciphertext. To obtain a (single key) construction from LWE, we resolve this issue by repurposing a classic trick from Yao's garbled circuit construction, so that the output length of the circuit can be made independent of the ciphertext size, at the cost of blowing up the ciphertext size somewhat. More concretely, instead of having the circuit output a new ciphertext, the encryptor provides symmetric key encryptions of CktFE [12] ciphertext components, encrypting *all possible bit values* (nesting CktFE ciphertext inside SKE ciphertext), and the function key outputs the SKE keys to unlock the correct CktFE ciphertext components, corresponding to the bit values chosen by the key. This allows us to *select* the next state with a circuit output length independent of the ciphertext size. For more details, we refer the reader to Section 4. This provides input privacy and reusability but not machine privacy. We achieve machine privacy following ideas of [12] – please see the full version [1] for details.

1.3.2 Single key FE for Turing Machines

To extend the above construction to support Turing Machines, we must address two challenges: a) head movements should not reveal anything about the input and b) we need to write to the tape. Below we describe how to handle each challenge in turn.

To overcome the first challenge, a natural approach is to use oblivious TMs, which fix the head movement of a TM to be independent of the input. Moreover, there exist efficient transformations that convert any Turing machine M that takes time T to decide an input to an oblivious one that takes time $T \log T$ to decide the same input [17]. It remains to address the challenge of handling tape writes. Since the head movements of the TM are now fixed, the only thing that the transition function must specify is the next state, and the symbol that must be written to the current tape cell. We leverage decomposability and have the encryptor provide a ciphertext component encoding state, and another component encoding current work tape symbol *for every step in the computation.* Indeed, this forces our ciphertext to depend (linearly) on worst-case runtime of the Turing machine. All the ciphertext components for a given time step are tied together with common randomness as before. To ensure that the decryptor only learns the ciphertext components corresponding to the particular state and tape symbol that occur during computation, the encryptor encrypts all CktFE ciphertexts with symmetric key encryption SKE. As in the case of DFA, the function key selects the appropriate SKE keys to reveal the CktFE ciphertext encoding next state and symbol to be read.

The careful reader may have noticed that the above description glosses over an important detail: the cell that is written into at step i may be next accessed at any step $j > i$, so the CktFE ciphertext at step i must encode SKE keys for some unknown future step j. Fortunately, the machinery of oblivious TMs comes to our aid again. Since in an oblivious TM, there exists a function t that computes the step that particular cell will be accessed next, in step i, in addition to selecting the state for step $i + 1$ as we did in DFAs, the function key will also select the tape symbol to be read in step $t(i)$. At any step j, the appropriate SKE keys for the state were provided in step $j - 1$ and for tape symbol were provided at step $i < j$ where $t(i) = j$. Hence, the decryptor at step j has the SKE keys to unlock the CktFE CT components for both state and tape symbol, which lets her proceed with the computation. For more details, please see the full version [1].

1.4 Organization of the paper

In Section 2, we define the preliminaries we require for our constructions. In Section 3, we define the notion of decomposable functional encryption. In Section 4, we provide our construction for single key FE for DFAs. In the full version [1], we provide our construction for single key functional encryption for Turing machines.

2 Definitions: FE for Deterministic Finite Automata

In this section we provide some notation and preliminaries that we require.

Functional encryption for deterministic finite automata (DFA) is defined analogously to functional encryption for circuits, except that the public parameters may not depend on the input length, which is unknown a priori. In this section, we will define single key functional encryption for DFAs.

A DFA machine M is represented by the tuple $(Q, \Sigma, T, q_{st}, F)$ where Q is a finite set of states, Σ is a finite alphabet, $T : \Sigma \times Q \to Q$ is the transition function (stored as a table), q_{st} is the start state, $F \subseteq Q$ is the set of accepting states. Upon input $\mathbf{w} \in \Sigma^k$ for some arbitrary polynomial k (not known to the setup algorithm), the machine M accepts the input if and only if there exists a sequence of states $q_1, \ldots, q_k$ so that $q_1 = q_{st}$, $T(w_i, q_i) = q_{i+1}$ for $i \in [k-1]$, and $q_k \in F$. We say $M(\mathbf{w}) = 1$ iff M accepts $\mathbf{w}$ and 0 otherwise.

2.1 Definition

Let $\mathcal{M}_\kappa : \mathcal{Q}_\kappa \times \Sigma_\kappa \to \mathcal{Q}_\kappa$ be a DFA family. A functional encryption scheme DfaFE for $\mathcal{M}$ consists of four algorithms DfaFE = (DfaFE.Setup, DfaFE.KeyGen, DfaFE.Enc, DfaFE.Dec) defined as follows.

- $\mathsf{DfaFE.Setup}(1^\kappa)$ is a p.p.t. algorithm takes as input the unary representation of the security parameter and outputs the master public and secret keys $(\mathsf{PK}, \mathsf{MSK})$.
- $\mathsf{DfaFE.KeyGen}(\mathsf{MSK}, \mathsf{M})$ is a p.p.t. algorithm that takes as input the master secret key MSK and a DFA machine M and outputs a corresponding secret key SK_M.
- $\mathsf{DfaFE.Enc}(\mathsf{PK}, \mathbf{w})$ is a p.p.t. algorithm that takes as input the master public key PK and an input message $\mathbf{w}$ and outputs a ciphertext $\mathsf{CT}_\mathbf{w}$.
- $\mathsf{DfaFE.Dec}(\mathsf{SK_M}, \mathsf{CT}_\mathbf{w})$ is a deterministic algorithm that takes as input the secret key SK_M and a ciphertext $\mathsf{CT}_\mathbf{w}$ and outputs $M(\mathbf{w})$.

▶ **Definition 1** (Correctness). A functional encryption scheme DfaFE is correct if for all $M \in \mathcal{M}$ and all $\mathbf{w} \in \Sigma^*$,

$$\Pr\left[\begin{array}{l}(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{DfaFE.Setup}(1^\kappa);\\ \mathsf{DfaFE.Dec}\Big(\mathsf{DfaFE.KeyGen}(\mathsf{MSK}, \mathsf{M}), \mathsf{DfaFE.Enc}(\mathsf{PK}, \mathbf{w})\Big) \neq \mathsf{M}(\mathbf{w})\end{array}\right] = \mathrm{negl}(\kappa)$$

where the probability is taken over the coins of DfaFE.Setup, DfaFE.KeyGen, and DfaFE.Enc.

2.2 Security

In this section, we define simulation based security for single key FE for DFAs.

▶ **Definition 2** (FULL-SIM- Security for DFA-FE). Let $\mathcal{F}_\mathcal{M}$ be a functional encryption scheme for a DFA family $\mathcal{M}$. For every p.p.t. adversary $A = (A_1, A_2)$ and a p.p.t. simulator Sim, consider the following two experiments:

$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathsf{DfaFE,A}}(1^\kappa)\textbf{:}}$	$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{DfaFE},\mathrm{Sim}}(1^\kappa)\textbf{:}}$
1: $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{DfaFE.Setup}(1^\kappa)$	1: $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{DfaFE.Setup}(1^\kappa)$
2: $(M, st_1) \leftarrow A_1(\mathsf{PK})$	2: $(M, st_1) \leftarrow A_1(\mathsf{PK})$
3: $\mathsf{sk}_M \leftarrow \mathsf{DfaFE.KeyGen}(\mathsf{MSK}, \mathsf{M})$	3: $\mathsf{sk}_M \leftarrow \mathsf{DfaFE.KeyGen}(\mathsf{MSK}, \mathsf{M})$
4: $(\mathbf{x}, st) \leftarrow A_2(st_1, \mathsf{PK}, \mathsf{sk}_M)$	4: $(\mathbf{x}, st) \leftarrow A_2(st_1, \mathsf{PK}, \mathsf{sk}_M)$
5: $\mathsf{CT} \leftarrow \mathsf{DfaFE.Enc}(\mathsf{PK}, \mathbf{x})$	5: $\tilde{\mathsf{CT}} \leftarrow \mathrm{Sim}(\mathsf{PK}, \mathsf{sk}_M, M, M(\mathbf{x}), 1^{\lvert\mathbf{x}\rvert})$
6: Output (st, CT)	6: Output $(st, \tilde{\mathsf{CT}})$

The DFA functional encryption scheme $\mathcal{F}_\mathcal{M}$ is then said to be single query FULL-SIM secure if there exists a p.p.t. simulator Sim such that for every p.p.t. adversary $A = (A_1, A_2)$, the following two distributions are computationally indistinguishable:

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

3 Decomposable Functional Encryption for Circuits

In this section, we define the notion of *decomposable functional encryption* (DFE). Decomposable functional encryption is analogous to the notion of decomposable randomized encodings [3]. Intuitively, decomposability requires that the public key PK and the ciphertext $\mathsf{CT}_\mathbf{x}$ of a functional encryption scheme be decomposable into components PK_i and CT_i for $i \in [|\mathbf{x}|]$, where CT_i depends on a single deterministic bit x_i and the public key component PK_i. In addition, the ciphertext may contain components that are independent of the message and depend only on the randomness.

We assume that given the security parameter, the following spaces are fixed: $\mathcal{P}$ containing public key components, $\mathcal{R}_1$, $\mathcal{R}_2$ containing randomness used for encryption and $\mathcal{C}$ containing the encoding of a single message bit. The length of the message $|\mathbf{x}|$ can be any polynomial. Formally, let $\mathbf{x} \in \{0,1\}^k$. A functional encryption scheme is said to be decomposable if there exists a deterministic function $\mathcal{E} : \mathcal{P} \times \{0,1\} \times \mathcal{R}_1 \times \mathcal{R}_2 \to \mathcal{C}$ such that:

1. The public key may be interpreted as $\mathsf{PK} = (\mathsf{PK}_1, \ldots, \mathsf{PK}_k, \mathsf{PK}_{\mathsf{indpt}})$ where $\mathsf{PK}_i \in \mathcal{P}$ for $i \in [k]$. The component $\mathsf{PK}_{\mathsf{indpt}} \in \mathcal{P}^j$ for some $j \in \mathbb{N}$.

2. The ciphertext may be interpreted as $\mathsf{CT}_{\mathbf{x}} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_k, \mathsf{CT}_{\mathsf{indpt}})$, where

$$\mathsf{CT}_i = \mathcal{E}\left(\mathsf{PK}_i, x_i, r, \hat{r}_i\right) \forall i \in [k] \quad \text{and} \quad \mathsf{CT}_{\mathsf{indpt}} = \mathcal{E}\left(\mathsf{PK}_{\mathsf{indpt}}, r, \hat{r}\right).$$

Here $r \in \mathcal{R}_1$ is common randomness used by all components of the encryption. Apart from the common randomness r, each CT_i may additionally make use of independent randomness $\hat{r}_i \in \mathcal{R}_2$.

We note that if a scheme is decomposable "bit by bit", i.e. into k components for inputs of size k, it is also decomposable into components corresponding to any partition of the interval $[k]$. Thus, we may decompose the public key and ciphertext into any $i \leq k$ components of length k_i each, such that $\sum k_i = k$. We will sometimes use $\bar{\mathcal{E}}(\mathbf{y})$ to denote the tuple of function values obtained by applying $\mathcal{E}$ to each component of a vector, i.e. $\bar{\mathcal{E}}(\mathsf{PK}, \mathbf{y}, r) \triangleq \Big(\mathcal{E}(\mathsf{PK}_1, y_1, r, \hat{r}_1), \ldots, \mathcal{E}(\mathsf{PK}_k, y_k, r, \hat{r}_k)\Big)$, where $|\mathbf{y}| = k$.

4 Single-Key Succinct FE for DFAs from LWE

In this section, we will construct a single key (public key) functional encryption scheme for deterministic finite automata (DFA). Our construction makes use a decomposable single key FE scheme for circuits, CktFE. In the full version [1], we show that:

▶ **Lemma 3.** *The single key, succinct functional encryption scheme for circuits by Goldwasser et al. [12], based on LWE is decomposable.*

Conceptually, we decompose the input into two components of size ℓ_1 and ℓ_2 each, where the second component is further decomposed bit by bit. We will use the first component to encrypt the current input symbol in the DFA computation and keys to select the next state in the computation, and the second component to encrypt the current state in the DFA computation. While the input symbol encoded in the first component can be treated as a unit of size ℓ_1, it will be helpful for us to represent the encoded input of size ℓ_2 bit by bit.

Thus, we have,

$$\mathsf{CktFE.PK} = (\mathsf{PK}_1, \mathsf{PK}_2, \mathsf{PK}_{\mathsf{indpt}}) \text{ and } \mathsf{CktFE.CT} = (\mathsf{CT}_1, \mathsf{CT}_2, \mathsf{CT}_{\mathsf{indpt}}).$$

Now let

$$\begin{aligned}\mathsf{CktFE.Enc}(\mathsf{PK}, \mathbf{x} \| \mathbf{y}) &= (\mathsf{CT}_1, \mathsf{CT}_2, \mathsf{CT}_{\mathsf{indpt}}) \\ &= \Big(\bar{\mathcal{E}}(\mathsf{PK}_1, \mathbf{x}, r, \hat{\mathbf{r}}_1),\ \bar{\mathcal{E}}(\mathsf{PK}_2, \mathbf{y}, r, \hat{\mathbf{r}}_2),\ \bar{\mathcal{E}}(\mathsf{PK}_{\mathsf{indpt}}, r, \hat{\mathbf{r}}_3)\Big).\end{aligned}$$

We decompose

$$\bar{\mathcal{E}}(\mathsf{PK}_2, \mathbf{y}, r, \hat{\mathbf{r}}_2) = \Big(\mathcal{E}(\mathsf{PK}_{2,1}, y_1, r, \hat{r}_{2,1}), \ldots, \mathcal{E}(\mathsf{PK}_{2,\ell_2}, y_{\ell_2}, r, \hat{r}_{2,\ell_2})\Big).$$

Recall that $\mathcal{E} : \mathcal{P} \times \{0,1\} \times \mathcal{R}_1 \times \mathcal{R}_2 \to \mathcal{C}$ and $\bar{\mathcal{E}}(\mathbf{x})$ denotes the tuple of function values obtained by applying $\mathcal{E}$ to each coordinate. Then,

$$\text{Let} \quad |\mathbf{x}| = \ell_1, \quad |\mathbf{y}| = \ell_2, \quad \mathsf{PK}_1 \in \mathcal{P}^{\ell_1}, \quad \mathsf{PK}_2 \in \mathcal{P}^{\ell_2},$$
$$j \in \mathbb{N}, \quad \mathsf{PK}_{\mathsf{indpt}} \in \mathcal{P}^j, \quad r \in \mathcal{R}_1, \quad \hat{\mathbf{r}}_1 \in \mathcal{R}_2^{\ell_1}, \quad \hat{\mathbf{r}}_2 \in \mathcal{R}_2^{\ell_2}, \quad \hat{\mathbf{r}}_3 \in \mathcal{R}_2^{j}.$$

In what follows, we abuse notation slightly and omit mention of the independent, fresh randomness from $\mathcal{R}_2$ needed for each invocation for $\mathcal{E}$. For convenience, we club the message independent component $\mathsf{CT}_{\mathsf{indpt}}$ with CT_1 and let

$$\mathbf{c} = (\mathsf{CT}_1, \mathsf{CT}_{\mathsf{indpt}}) \text{ and } \mathbf{d} = \mathsf{CT}_2 = (\mathsf{CT}_{2,1}, \ldots, \mathsf{CT}_{2,\ell_2}).$$

Let $\mathcal{M}_\kappa : \mathcal{Q}_\kappa \times \Sigma_\kappa \to \mathcal{Q}_\kappa$ be a DFA family. For notational convenience, we will drop the subscript κ here on. Let $Q = |\mathcal{Q}|$, the size of the state space of the DFA family. Then, the single key functional encryption scheme for DFAs is constructed as follows.

DfaFE.Setup(1^κ):

Upon input the security parameter 1^κ, do:

1. Choose a symmetric key encryption scheme SKE with key space $\mathcal{K}$.
2. Define a circuit family as follows. Let $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X} = (\Sigma \times \mathcal{K}^{2\log Q} \times \{0,1\}) \times \mathcal{Q}$ and $\mathcal{Y} = \mathcal{K}^{\log Q}$. We set
$$\ell_1 = \lfloor \Sigma \rfloor + \lfloor \mathcal{K}^{2\log Q} \rfloor + 1, \quad \ell_2 = \lfloor \mathcal{Q} \rfloor = \log Q$$
where $\lfloor \cdot \rfloor$ denotes size in bits. Let $\ell = \ell_1 + \ell_2$.
3. Invoke $\mathsf{CktFE.Setup}(1^\kappa, 1^\ell)$ to obtain $\mathsf{PK} = \big(\mathsf{PK}_1, (\mathsf{PK}_{2,1}, \ldots, \mathsf{PK}_{2,\log Q}), \mathsf{PK}_{\mathsf{indpt}}\big)$ and MSK.
4. Output (PK, MSK).

DfaFE.Enc(PK, w):

Let $|\mathbf{w}| = k$. Note that k is arbitrary, and unknown to DfaFE.Setup. Do the following:

1. Sample randomness $r_i \leftarrow \mathcal{R}_1$ for $i \in [k]$.
2. Sample SKE keys as follows. We define
$$\mathbf{K}_{i+1} = \Big((K^0_{(i+1,1)}, K^1_{(i+1,1)}), \ldots, (K^0_{(i+1,\log Q)}, K^1_{(i+1,\log Q)})\Big)$$
where $K^b_{i+1,j} \leftarrow \mathcal{K}$ for $i \in [k-1]$, $j \in [\log Q]$, $b \in \{0,1\}$.
3. Define message $\mathbf{y}_i = (w_i, \mathbf{K}_{i+1}, 0)$ for $i \in [k-1]$ and $\mathbf{y}_k = (w_k, \perp, 1)$.
4. For $i \in [k]$, we define:
$$\mathbf{c}_{i,1} = \bar{\mathcal{E}}(\mathsf{PK}_1, \mathbf{y}_i, r_i), \quad \mathbf{c}_{i,2} = \bar{\mathcal{E}}(\mathsf{PK}_{\mathsf{indpt}}, r_i), \quad \mathbf{c}_i = (\mathbf{c}_{i,1}, \mathbf{c}_{i,2}).$$
5. Let $\mathbf{d}_1 = \bar{\mathcal{E}}\ (\mathsf{PK}_2, q_{st}, r_1)$. Here q_{st} denotes the start state of the DFA. Further, let
$$\mathbf{d}^b_{i,j} = \mathcal{E}\ (\mathsf{PK}_{2,j}, b, r_i)\ \ \forall\, i \in [2,k],\ j\ \in [\log Q],\ b \in \{0,1\}.$$
$$\mathbf{d}_{i,q} \triangleq (\mathbf{d}^{q_j}_{i,j})\ \ \forall j \in [\log Q] \text{ where } q_j \text{ is the } j\text{th bit of } q.$$
6. For $i \in [2,k]$, $j \in [\log Q]$, $b \in \{0,1\}$ encrypt each $\mathbf{d}^b_{i,j}$ using the corresponding key $K^b_{i,j}$ as:
$$\hat{\mathbf{d}}^b_{i,j} = \mathsf{SKE.Enc}(K^b_{i,j}, \mathbf{d}^b_{i,j}).$$
7. Choose $b_{i,j} \leftarrow \{0,1\}$ randomly for $i \in [2,k]$, $j \in [\log Q]$ and define:
$$\hat{\mathbf{D}}_{i,j} = \big(\ \hat{\mathbf{d}}^{b_{i,j}}_{i,j}, \hat{\mathbf{d}}^{\bar{b}_{i,j}}_{i,j}\ \big), \quad \hat{\mathbf{D}}_i = (\hat{\mathbf{D}}_{i,j}), \quad \hat{\mathbf{D}}_1 = \mathbf{d}_1.$$
8. Output $\mathsf{CT}_{\mathbf{w}} = \{\mathbf{c}_i, \hat{\mathbf{D}}_i\}$ for $i \in [k]$.

DfaFE.KeyGen(MSK, M):

Let M denote a DFA machine and T denote its transition matrix. Let T_i denote the i^{th} row of T, with format $\big((\sigma, q) \to q'\big)$ indicating that the machine enters state q' upon input symbol σ and input state q. Let $\mathsf{SK}_M = \mathsf{CktFE.Keygen}(\mathsf{MSK}, f)$ where f is defined below in Figure 1.

Function $f\big((\sigma, \mathbf{K}, \mathsf{flag}), q\big)$

1. Lookup table T for (σ, q). Say that $(\sigma, q) \to q'$. If no entry is found, output $\perp$ and exit.
2. If $\mathsf{flag} = 1$, check if q' is an accepting state. If yes, output 1, else output 0 and exit.
3. If $\mathsf{flag} = 0$, parse $\mathbf{K}$ as $\{(K_j^0, K_j^1)\}$ for $j \in [\log Q]$, $b \in \{0,1\}$. Choose the $\log Q$ keys $K_j^{q'_j}$ (for $j \in [\log Q]$), corresponding to the bits of q' and output these.

Figure 1 Function to provide keys for next state in DFA computation.

$\mathsf{DfaFE.Dec}(\mathsf{SK}_f, \mathsf{CT}_\mathbf{w})$:

Interpret $\mathsf{CT}_\mathbf{w} = (\mathbf{c}_i, \hat{\mathbf{D}}_i)_{i\in[k]}$ and let $\mathbf{d}_{1,q_1} = \hat{\mathbf{D}}_1$.

Initialize $i = 1$. While $i \le k$, do the following:

- Let $\mathsf{CT}'_i = (\mathbf{c}_i, \mathbf{d}_{i,q_i})$. Recall that $\mathbf{d}_{i,q_i} = (\mathbf{d}_{i,j}^{q_{i,j}})$ for $j \in [\log Q]$. If $i = k$, let $b \leftarrow \mathsf{CktFE.Dec}(\mathsf{SK}_f, \mathsf{CT}'_k)$. Output b and exit.
- Else let $(K_{i+1,1}, \ldots, K_{i+1,\log Q}) = \mathsf{CktFE.Dec}(\mathsf{SK}_f, \mathsf{CT}'_i)$.
- For $j \in [\log Q]$, try to decrypt each value in $\hat{\mathbf{D}}_{i+1,j}$ using obtained key $K_{i+1,j}$. Exactly one of the two ciphertexts per bit position will be decrypted, say $\hat{\mathbf{d}}^{b_j}_{i+1,j}$. Set

$$\mathbf{d}_{i+1,q_{i+1}} = \Big(\ \mathsf{SKE.Dec}(K_{i+1,j}, \hat{\mathbf{d}}^{b_j}_{i+1,j})\ \Big) \quad \forall j \in [\log Q]\,.$$

- Increment i.

4.1 Correctness

In this section, we establish correctness of the above construction. Before we proceed with the formal argument, we provide some intuition. Note that in the encryption, the first component $\mathbf{c}_i$ encrypts message $\mathbf{y}_i$, which contains the ith input symbol, along with the set of all $2\log Q$ symmetric keys used to construct SKE encryptions of the $(i+1)^{th}$ state. In the second component, the element $\mathbf{d}^b_{i,j}$ in tuple $(\ \mathbf{d}^b_{i,j}\ )$ for $j \in [\log Q]$ and $b \in \{0,1\}$, contains an encryption of bit b, corresponding to the event that the j^{th} bit of i^{th} state is b. The set $\hat{\mathbf{D}}_i$ contains $2\log Q$ SKE encryptions of $\mathbf{d}^b_{i,j}$ under keys $K^b_{i,j}$, shuffled for each position j. Decryption at step $i-1$ provides the level i symmetric keys $K^{b_j}_{i,j}$ to unlock the $\mathbf{d}^{b_j}_{i,j}$ for the correct next state of the computation q', i.e. $b_j = q'_{i,j}$. Thus, the decryptor recovers exactly the components $\mathbf{d}^{b_j}_{i,j}$ which may be combined to create the ciphertext $\mathbf{d}_{i,q_i}$. Put together with $\mathbf{c}_i$ we get an encryption of $(w_i, \mathbf{K}_{i+1}, q_i)$ which may again be decrypted with the function key to obtain the appropriate keys to decrypt the correct $\mathbf{d}_{i+1,q_{i+1}}$.

Formally, let k denote the length of input $\mathbf{w}$ and let $q_1, \ldots, q_k$ denote the states visited by the DFA during computation. We have by correctness of decomposable functional encryption that:

$$\forall i \in [k-1],\ \mathsf{CktFE.Enc}\big(\ \mathsf{PK}, (w_i, \mathbf{K}_{i+1}, 0, q_i)\ \big) = (\mathbf{c}_i, \mathbf{d}_{i,q_i}) \quad \text{where}$$
$$\mathbf{c}_k = \Big(\bar{\mathcal{E}}\big(\mathsf{PK}_1, (w_i, \mathbf{K}_{i+1}, 0), r_i, \big), \bar{\mathcal{E}}\big(\mathsf{PK}_{\mathsf{indpt}}, r_i\big)\Big), \quad \mathbf{d}_{i,q_i} = \big(\mathcal{E}(\mathsf{PK}_{2,j}, q_{i,j}, r_i)\big)_{j\in[\log Q]}$$
$$\text{s.t. } \mathsf{CktFE.Dec}\big(\mathsf{SK}_f, (\mathbf{c}_i, \mathbf{d}_{i,q_i})\big) = \mathbf{K}_{q_{i+1}} \triangleq (K^{b_1}_{i+1,1}, \ldots, K^{b_{\log Q}}_{i+1,\log Q}) \ \text{ where } \ b_j = q_{i+1,j}.$$

Now, both elements of $\hat{\mathbf{D}}_{i+1,j}$ are attempted for decryption by $K^{b_j}_{i+1,j}$, of which only the element encoding the correct bit $q_{i+1,j}$ is recovered. Formally, we have:

$$\hat{\mathbf{D}}_{i+1,j} = \left(\hat{\mathbf{d}}^0_{i+1,j}, \hat{\mathbf{d}}^1_{i+1,j} \right) \quad \text{and}$$
$$\mathsf{SKE.Dec}\left(K^{b_j}_{i+1,j},\ \hat{\mathbf{d}}^b_{i+1,j} \right) = \perp \ \text{ if } \ b_j \neq b, \ \text{and } \mathbf{d}^{q_{i+1,j}}_{i+1,j} \quad \text{otherwise.}$$

By putting together all the components, we get by decomposability:

$$\mathbf{d}_{i+1,q_{i+1}} = \left(\mathbf{d}^{q_{i+1,j}}_{i+1,j} \right) \ \forall\, j \in [\log Q]$$

Also, since each component of $\mathbf{d}_{i+1,q_{i+1}}$ uses the same common randomness r_{i+1} as is used by $\mathbf{c}_{i+1}$, we have that $\mathsf{CT}_{i+1} = (\mathbf{c}_{i+1}, \mathbf{d}_{i+1,q_{i+1}})$, hence we may repeat while $i < k$. Finally for $i = k$,

$$\mathsf{CktFE.Enc}\big(\mathsf{PK}, (w_k, \perp, 1, q_k) \big) = (\mathbf{c}_i, \mathbf{d}_{k,q_k})$$
so that $\mathsf{CktFE.Dec}\big(\mathsf{SK}_f, (\mathbf{c}_k, \mathbf{d}_{k,q_k})\big) = 1$ iff q_k is an accepting state, 0 otherwise.

Efficiency. We note that the public key size of our scheme is the public key size of CktFE [12] with message length $\ell = O(\log Q + \log|\Sigma| + \log Q \cdot \log|\mathcal{K}|$ which is polynomial in the security parameter κ. The ciphertext size is $O(|\mathbf{w}| \cdot \log Q)$ and the secret key size is $O(|M|)$ (ignoring polynomials in the security parameter).

4.2 Proof of Security

We proceed to show that our construction is secure. Formally:

▶ **Theorem 4.** *Assume that the underlying* CktFE *scheme satisfies* $\mathsf{FULL\text{-}SIM}$ *security according to definition (please see* [1]*). Then the construction for* DfaFE *achieves* $\mathsf{FULL\text{-}SIM}$ *security as defined in Definition 2.*

Proof. We proceed to construct a simulator $\mathsf{DfaFE.Sim}$ as required by Definition 2. The simulator receives $(\mathsf{PK}, \mathsf{SK}_M, M, M(\mathbf{w}), 1^{|\mathbf{w}|})$ and does the following:

1. Assign the bit $b = M(\mathbf{w})$, and construct the circuit f corresponding to M as defined in Figure 1 in the description of $\mathsf{DfaFE.KeyGen}$.
2. Let $\mathsf{CktFE.SK_f} = \mathsf{SK_M}$ and invoke $\mathsf{CktFE.Sim}\big(\mathsf{PK}, f, \mathsf{CktFE.SK}_f, b \big)$ to receive $\tilde{\mathsf{CT}}_k$ where we may express $\tilde{\mathsf{CT}}_k = (\tilde{\mathbf{c}}_k, \tilde{\mathbf{d}}_{k,q_k})$ and $\tilde{\mathbf{d}}_{k,q_k} = (\tilde{\mathbf{d}}_{k,j})$ for $j \in [\log Q]$.
3. For $(i = k, i \geq 1, i--)$, do:
   - **a.** If $i = 1$, set $\mathsf{Sim}.\hat{\mathbf{D}}_1 = \tilde{\mathbf{d}}_{1,q_1}$ and exit.
   - **b.** Sample key $\mathbf{K}^*_i = (K^*_{i,1}, \ldots, K^*_{i,\log Q}) \leftarrow \mathcal{K}^{\log Q}$ and let
   $$\mathsf{Sim}.\hat{\mathbf{d}}_{i,j} = \mathsf{SKE.Enc}(K^*_{i,j}, \tilde{\mathbf{d}}_{i,j}) \ \ \forall j \in [\log Q]\,.$$
   - **c.** Sample $\tilde{b}_{i,j} \leftarrow \{0,1\}$ and assign $\mathsf{Sim}.\hat{\mathbf{d}}^{\tilde{b}_{i,j}}_{i,j} = \mathsf{Sim}.\hat{\mathbf{d}}_{i,j}$.
   - **d.** Choose another $\log Q$ keys $\tilde{K}_{i,1}, \ldots \tilde{K}_{i,\log Q} \leftarrow \mathcal{K}^{\log Q}$ and compute
   $$\mathsf{Sim}.\hat{\mathbf{d}}^{\bar{\tilde{b}}_{ij}}_{i,j} = \mathsf{SKE.Enc}(\tilde{K}_{i,j}, 0^{|\tilde{\mathbf{d}}_{i,j}|}) \ \ \forall\, j \in [\log Q]\,.$$
   - **e.** Let $\mathsf{Sim}.\hat{\mathbf{D}}_{i,j} = \big(\mathsf{Sim}.\hat{\mathbf{d}}^{\tilde{b}_{i,j}}_{i,j},\ \mathsf{Sim}.\hat{\mathbf{d}}^{\bar{\tilde{b}}_{i,j}}_{i,j}\big)$ and $\mathsf{Sim}.\hat{\mathbf{D}}_i = \big(\mathsf{Sim}.\hat{\mathbf{D}}_{i,j} \big)$ for $j \in [\log Q]$.
   - **f.** Let $(\tilde{\mathbf{c}}_{i-1}, \tilde{\mathbf{d}}_{i-1,q_{i-1}}) = \mathsf{CktFE.Sim}(\mathsf{PK}, f, \mathsf{SK}_f, \mathbf{K}^*_i)$.
4. Output the ciphertext as $\mathsf{CT}_\mathbf{w} = (\tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2, \ldots, \tilde{\mathbf{c}}_k, \mathsf{Sim}.\hat{\mathbf{D}}_1, \ldots, \mathsf{Sim}.\hat{\mathbf{D}}_k)$.

4.2.1 Analysis of Simulator

Correctness of the simulator DfaFE.Sim can be easily established using correctness of the simulator CktFE.Sim and the semantic security of SKE. Let us say that the DFA M visits states $q_1, \ldots, q_k$ while computing on input $\mathbf{w}$ where $|\mathbf{w}| = k$.

1. We have by correctness of CktFE.Sim that:

$$\Big\{ \mathsf{CT}_k \leftarrow \mathsf{CktFE.Enc}\big(\mathsf{PK}, (w_k, \perp, 1, q_k)\big) \overset{c}{\approx} \tilde{\mathsf{CT}}_k \leftarrow \mathsf{CktFE.Sim}\big(\mathsf{PK}, f_M, \mathsf{SK}_f, b\big) \Big\}.$$

   By decomposability, $\mathsf{CT}_k = (\mathbf{c}_k, \mathbf{d}_{k,q_k})$ where $\mathbf{d}_{k,q_k} = (\mathbf{d}^{b_j}_{k,j})$ for $j \in [\log Q]$ and $b_j = q_{k,j}$ defined as the j^{th} bit of state q_k. Similarly, $\tilde{\mathsf{CT}}_k = (\tilde{\mathbf{c}}_k, \tilde{\mathbf{d}}_{k,q_k})$ where $\tilde{\mathbf{d}}_{k,q_k}$ may be decomposed as $(\tilde{\mathbf{d}}_{k,j})$ for $j \in [\log Q]$. Let $i = k$.

2. We now establish that $(\hat{\mathbf{d}}^{b_j}_{i,j} \overset{c}{\approx} \mathsf{Sim}.\hat{\mathbf{d}}_{i,j})$ where $b_j = q_{i,j}$ and $j \in [\log Q]$.
   a. We have that in algorithm DfaFE.Enc,

$$\mathbf{K}_i = \Big((K^0_{(i,1)}, K^1_{(i,1)}), \ldots, (K^0_{(i,\log Q)}, K^1_{(i,\log Q)}) \Big)$$

   where $K^b_{i,j} \leftarrow \mathcal{K}$ for $j \in [\log Q]$. We also have, for $j \in [\log Q]$, $b \in \{0,1\}$:

$$\hat{\mathbf{d}}^b_{i,j} = \mathsf{SKE.Enc}(K^b_{i,j}, \mathbf{d}^b_{i,j}) \tag{4.1}$$

   b. In simulator DfaFE.Sim:

$$\mathbf{K}^*_i = (K^*_{i,1}, \ldots, K^*_{i,\log Q}) \leftarrow \mathcal{K}^{\log Q} \quad \text{and}$$

$$\mathsf{Sim}.\hat{\mathbf{d}}_{i,j} = \mathsf{SKE.Enc}(K^*_{i,j}, \tilde{\mathbf{d}}_{i,j}) \ \ \forall j \in [\log Q]\,.$$

   Hence, since $\mathbf{d}^{b_j}_{i,j} \overset{c}{\approx} \tilde{\mathbf{d}}_{i,j}$ and the symmetric keys are picked using the same distribution in each case, we have that $(\hat{\mathbf{d}}^{b_j}_{i,j} \overset{c}{\approx} \mathsf{Sim}.\hat{\mathbf{d}}_{i,j})$ where $b_j = q_{i,j}$ and $j \in [\log Q]$.

3. We now establish that $(\hat{\mathbf{d}}^{\bar{b}_j}_{i,j} \overset{c}{\approx} \mathsf{Sim}.\hat{\mathbf{d}}^{\bar{\tilde{b}}_j}_{i,j})$ where $j \in [\log Q]$.
   a. Construction of $\hat{\mathbf{d}}^{\bar{b}_j}_{i,j}$ is described in Equation 4.1.
   b. For the latter, DfaFE.Sim samples $\tilde{b}_j$ and sets $\mathsf{Sim}.\hat{\mathbf{d}}^{\tilde{b}_j}_{i,j} = \mathsf{Sim}.\hat{\mathbf{d}}_{i,j}$. Next, it samples another $\log Q$ keys $\tilde{K}_{i,1}, \ldots \tilde{K}_{i,\log Q} \leftarrow \mathcal{K}^{\log Q}$ and computes

$$\mathsf{Sim}.\hat{\mathbf{d}}^{\bar{\tilde{b}}_j}_{i,j} = \mathsf{SKE.Enc}(\tilde{K}_{i,j}, 0^{|\tilde{\mathbf{d}}_{i,j}|}) \ \ \forall\, j \in [\log Q]\,.$$

   By semantic security of SKE, we have that $(\hat{\mathbf{d}}^{\bar{b}_j}_{i,j} \overset{c}{\approx} \mathsf{Sim}.\hat{\mathbf{d}}^{\bar{\tilde{b}}_j}_{i,j})$.

4. Next, we show that $\hat{\mathbf{D}}_i \overset{c}{\approx} \mathsf{Sim}.\hat{\mathbf{D}}_i$. For $i = 1$, we have by definitions of $\hat{\mathbf{D}}_1$ and $\mathsf{Sim}.\hat{\mathbf{D}}_1$, that the above holds. For $i > 1$, in DfaFE.Enc, we have $b_{i,j} \leftarrow \{0,1\}$ and

$$\hat{\mathbf{D}}_{i,j} = \big(\ \hat{\mathbf{d}}^{b_{i,j}}_{i,j}, \hat{\mathbf{d}}^{\bar{b}_{i,j}}_{i,j}\ \big)\,.$$

   In DfaFE.Sim, we have $\tilde{b}_{i,j} \leftarrow \{0,1\}$ and

$$\mathsf{Sim}.\hat{\mathbf{D}}_{i,j} = \big(\mathsf{Sim}.\hat{\mathbf{d}}^{\tilde{b}_{i,j}}_{i,j},\ \mathsf{Sim}.\hat{\mathbf{d}}^{\bar{\tilde{b}}_{i,j}}_{i,j}\big)\,.$$

   Since $\hat{\mathbf{D}}_i = (\hat{\mathbf{D}}_{i,j})$ and $\mathsf{Sim}.\hat{\mathbf{D}}_i = \big(\ \mathsf{Sim}.\hat{\mathbf{D}}_{i,j}\ \big)$ for $j \in [\log Q]$, we have that $\hat{\mathbf{D}}_i \overset{c}{\approx} \mathsf{Sim}.\hat{\mathbf{D}}_i$.

5. Let $i = i - 1$. Now, we have by correctness of $\mathsf{CktFE.Sim}$,

$$\Big\{\mathsf{CT}_i \leftarrow \mathsf{CktFE.Enc}\big(\mathsf{PK}, (w_i, \mathbf{K}_{i+1}, 0, q_i)\big) \stackrel{c}{\approx} \tilde{\mathsf{CT}}_i \leftarrow \mathsf{CktFE.Sim}\big(\mathsf{PK}, f_M, \mathsf{SK}_f, \mathbf{K}^*_{i+1}\big)\Big\}.$$

By decomposability, $\mathsf{CT}_i = (\mathbf{c}_i, \mathbf{d}_{i,q_i})$ where $\mathbf{d}_{i,q_i} = (\mathbf{d}_{i,j}^{q_{i,j}})$ for $j \in [\log Q]$. Also, $\tilde{\mathsf{CT}}_i = (\tilde{\mathbf{c}}_i, \tilde{\mathbf{d}}_{i,q_i})$ where $\tilde{\mathbf{d}}_{i,q_i} = (\tilde{\mathbf{d}}_{i,j})$ for $j \in [\log Q]$. If $i > 1$, go to step 2. For $i = 1$, we have by definitions of $\hat{\mathbf{D}}_1$ and $\mathsf{Sim}.\hat{\mathbf{D}}_1$, that $(\mathbf{c}_1, \hat{\mathbf{D}}_1) \stackrel{c}{\approx} (\tilde{\mathbf{c}}_1, \mathsf{Sim}.\hat{\mathbf{D}}_1)$.

6. Now, a straightforward hybrid argument yield that:

$$\Big\{(\mathbf{c}_1, \hat{\mathbf{D}}_1), (\mathbf{c}_2, \hat{\mathbf{D}}_2), \ldots, (\mathbf{c}_k, \hat{\mathbf{D}}_k)\Big\} \stackrel{c}{\approx} \Big\{(\tilde{\mathbf{c}}_1, \mathsf{Sim}.\hat{\mathbf{D}}_1), (\tilde{\mathbf{c}}_2, \mathsf{Sim}.\hat{\mathbf{D}}_2), \ldots, (\tilde{\mathbf{c}}_k, \mathsf{Sim}.\hat{\mathbf{D}}_k)\Big\}$$

as desired.

◀

Reusable Garbled DFA. In the full version [1] we show how to compile the above construction with symmetric key encryption to obtain the first construction of reusable garbled DFAs from standard assumptions.

5 Single Key Functional Encryption for Turing Machines

In the full version [1], we provide the construction of single key functional encryption for Turing machines. Our construction has short public parameters that are independent of the size of the machine or the data being encrypted, short function keys, and input-specific decryption time. However, the ciphertext of our construction is large and depends on the worst case running time of the Turing machine (but not its description size).

While the large ciphertext size of our TMFE construction restricts its utility for practical applications, we emphasize that the parameters obtained by our TMFE construction are not implied by previous work to the best of our knowledge (please see the full version [1] for a detailed discussion about previous work). To improve the ciphertext size of our construction, while allowing succinct keys, dynamic data length and input specific run time is an interesting open problem.

References

1 Shweta Agrawal and Ishaan Preet Singh. Reusable garbled deterministic finite automata from learning with errors, full version. `http://www.cse.iitm.ac.in/~shwetaag/papers/dfa.pdf`, 2017.

2 Prabhanjan Ananth and Amit Sahai. Functional encryption for turing machines. In *Theory of Cryptography*, pages 125–153. Springer, 2016.

3 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. *SIAM J. Comput.*, 43(2):905–929, 2014.

4 Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT*, 2014.

5 Domagoj Babić, Daniel Reynaud, and Dawn Song. Malware analysis with tree automata inference. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, 2011.

6 Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In *STOC*, 2015.

7 Xavier Boyen and Qinyi Li. Attribute-Based Encryption for Finite Automata from LWE. In *Provable Security*, pages 247–267. Springer, 2015.

8 Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and ram programs. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC'15, 2015.

9 Yu-Chi Chen, Sherman S. M. Chow, Kai-Min Chung, Russell W. F. Lai, Wei-Kai Lin, and Hong-Sheng Zhou. Computation-trace indistinguishability obfuscation and its applications. *IACR Cryptology ePrint Archive*, 2015, 2015.

10 Sanjeev Das, Hao Xiao, Yang Liu, and Wei Zhang. Online malware defense using attack behavior model. In *IEEE International Symposium on Circuits and Systems, ISCAS 2016, Montréal, QC, Canada, May 22-25, 2016*, pages 1322–1325, 2016.

11 Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *CRYPTO (2)*, pages 536–553, 2013.

12 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.

13 Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate Encryption for Circuits from LWE. In *Crypto*, 2015.

14 Christopher L. Hayes and Yan Luo. Dpico: A high speed deep packet inspection engine using compact finite automata. In *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems*, ANCS'07, pages 195–203, 2007.

15 Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC'15, 2015.

16 Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In Ronald Cramer, editor, *Theory of Cryptography: 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, 2012.

17 Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *Journal of the ACM (JACM)*, 26(2):361–381, 1979.

18 Amit Sahai and Hakan Seyalioglu. Worry-free encryption: Functional encryption with public keys. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS'10, 2010.

19 Daniele Paolo Scarpazza, Oreste Villa, and Fabrizio Petrini. Peak-performance dfa-based string matching on the cell processor. In *21th International Parallel and Distributed Processing Symposium (IPDPS), Proceedings, 26-30 March 2007, Long Beach, California, USA*, pages 1–8, 2007.

20 Brent Waters. Functional encryption for regular languages. In *Crypto*, 2012.