# FE and iO for Turing Machines from Minimal Assumptions

Shweta Agrawal * Monosij Maitra †

Abstract

We construct Indistinguishability Obfuscation (iO) and Functional Encryption (FE) schemes in the Turing machine model from the minimal assumption of compact FE for circuits (CktFE). Our constructions overcome the barrier of sub-exponential loss incurred by all prior work. Our contributions are:

1. We construct iO in the Turing machine model from the same assumptions as required in the circuit model, namely, sub-exponentially secure FE for circuits. The previous best constructions [KLW15, AJS17] require sub-exponentially secure iO for circuits, which in turn requires sub-exponentially secure FE for circuits [AJ15, BV15].
2. We provide a new construction of single input FE for Turing machines with unbounded length inputs and optimal parameters from polynomially secure, compact FE for circuits. The previously best known construction by Ananth and Sahai [AS16] relies on iO for circuits, or equivalently, sub-exponentially secure FE for circuits.
3. We provide a new construction of multi-input FE for Turing machines. Our construction supports a fixed number of encryptors (say $k$), who may each encrypt a string $\mathbf{x}_i$ of unbounded length. We rely on sub-exponentially secure FE for circuits, while the only previous construction [BGJS15] relies on a strong knowledge type assumption, namely, public coin differing inputs obfuscation.

Our techniques are new and from first principles, and avoid usage of sophisticated iO specific machinery such as positional accumulators and splittable signatures that were used by all relevant prior work [KLW15, AS16, AJS17].

**Note**: The proof of Theorem 3.1 (provided partly in Section 3.3 and in Appendix B respectively) contains a subtle bug that breaks the indistinguishability chain of TMFE. At a high level, this is caused by propagating the data structure $\mathsf{Trap}_b$ (for the challenge bit $b \in \{0,1\}$) through the $\mathsf{1FE}_1$ and $\mathsf{1FE}_2$ ciphertexts across the entire computation chain; this, in turn, fails the analysis to rely on the distributional indistinguishability of $\mathsf{1FE}_2$ (and $\mathsf{1FE}_1$). Nevertheless, our theorem statement still holds true via the work in [KNTY19]. We thank Pratish Datta, Jiaxin Guan and Alexis Korb for pointing out the issue. We will attempt to fix it and post a revised version soon.

## 1 Introduction

The notion of indistinguishability obfuscation (iO) [BGI+01] seeks to garble programs such that the obfuscations of any two functionally equivalent programs are indistinguishable. While non-obvious at first what such a guarantee is good for, iO has emerged as a surprisingly powerful notion in cryptography, leading to many advanced cryptographic applications that

*IIT Madras, India. Email: shweta.a@cse.iitm.ac.in
†IIT Madras, India. Email: monosij@cse.iitm.ac.in

were previously out of reach [GGH+13, SW14, CLTV15, CHJV15, BGL+15, KLW15, BPR15, LPST16, CHN+16, CMR17, LZ17].

Functional encryption (FE) [SW05, BSW11, O'N10] is a generalization of public key encryption that enables fine grained access control on encrypted data. In FE, a secret key corresponds to a function $f$ and ciphertexts correspond to strings from the domain of $f$. Given a function key $\mathsf{SK}_f$ and a ciphertext $\mathsf{CT}_{\mathbf{x}}$, the decryptor learns $f(\mathbf{x})$ and nothing else.

While an important primitive in its own right, FE has also been shown to imply iO, albeit with sub-exponential loss [AJ15, BV15]. Over the last few years, both primitives have received significant attention, with a rich body of work that attempts to support more general models of computation [BGL+15, CHJV15, KLW15, CCHR15, CCC+15, ACC+16, CH16], rely on weaker assumptions [BKS16, GS16, LM16, BNPW16, KS17, AS17b, Lin17, LT17, KNT17, KNT18a, KNT18b], achieve stronger security [ABSV15, BKS16] and greater efficiency [AJS17].

In this work, we make further progress towards the goal of basing iO and FE on minimal assumptions, in the Turing machine model of computation. This question has been studied extensively [GKP+13a, AS16, BGL+15, KLW15, CHJV15, CCC+15, CCHR15, ACC+16, CH16, AJS17] – we refer the reader to [AS16, AJS17] for a detailed discussion. Below, we summarize the state of art:

1. iO for Turing Machines with unbounded memory and bounded inputs are constructed in the works of Koppula et al. and Ananth et al. [KLW15, AJS17]. Both works rely on the existence of sub-exponentially secure iO for circuits along with other standard assumptions. We note that FE for circuits implies iO with sub-exponential loss, so when relying on FE for circuits, these works incur double sub-exponential loss.

2. For single input FE for Turing machines that accept unbounded length inputs and place no restriction on the description size or space complexity of the machine, the state of art is the work of Ananth and Sahai [AS16], which relies on the existence of iO for circuits.

3. For multi-input FE in the Turing machine model, the only known construction is [BGJS15], which relies on the existence of public coin differing inputs obfuscation (diO).

Our Results. We construct Indistinguishability Obfuscation (iO) and Functional Encryption (FE) schemes in the Turing machine model from the minimal assumption of compact FE for circuits (CktFE). Our constructions overcome the barrier of sub-exponential loss incurred by all prior work. Our contributions are:

1. We construct iO for Turing machines with bounded inputs and unbounded memory from the same assumptions as required by iO for circuits, namely, sub-exponentially secure FE for circuits. The previous best constructions [KLW15, AJS17] require sub-exponentially secure iO for circuits, which in turn requires sub-exponentially secure FE for circuits [AJ15, BV15], resulting in double sub-exponential loss.

2. We provide a new construction of single input FE for Turing machines with unbounded inputs, achieving optimal parameters from polynomially secure, compact FE for circuits. The previously best known construction by Ananth and Sahai [AS16] relies on iO for circuits, or equivalently, sub-exponentially secure FE for circuits. We note that iO for circuits implies decomposable compact FE for circuits [GGH+13] (please see Appendix F), so our construction also implies FE for TMs from iO for circuits.

3. We provide a new construction of multi-input FE for Turing machines. Our construction supports a fixed number of encryptors (say $k$), who may each encrypt a string $\mathbf{x}_i$ of

unbounded length. We rely on sub-exponentially secure FE for circuits, while the only previous construction [BGJS15] relies on a strong knowledge type assumption, namely, public coin differing inputs obfuscation. The arity $k$ supported by our scheme depends on the underlying multi-input CktFE scheme, for instance using [KS17], we can support $k = \mathsf{polylog}(\lambda)$.

Our constructions make use of FE for circuits that satisfy a mild property called decomposablity, which in turn can be constructed generically from FE for circuits (please see Appendix F). Decomposable FE, analogously to decomposable randomized encodings [AIK11], roughly posits that a long string be encrypted bit by bit using shared randomness across bits. This property is already satisfied by all known constructions of CktFE in the literature to the best of our knowledge, please see Appendix F.1.

Our techniques are new and from first principles, and avoid usage of sophisticated iO specific machinery such as positional accumulators and splittable signatures that were used by all prior work [KLW15, AS16, AJS17]. Our work leverages the security notion of distributional indistinguishability (DI) for CktFE which was first considered by [GHRW14], who provided a construction for single input FE satisfying DI security assuming the existence of iO. We strengthen this result by constructing DI secure CktFE from standard CktFE. Please see Figure 1 for an overview of our results.

Additional Prior Work. Since iO is considered an inherently sub-exponential assumption and much stronger than the polynomial assumption of compact FE, replacing iO by FE in cryptographic constructions has already been studied extensively, for instance in the context of PPAD hardness [GPS16], multi-input FE for circuits [BKS16, KS17] as well as trapdoor one-way permutations and universal samplers [GPSZ16]. We note that aside from reliance on weaker, better understood assumptions, avoiding sub-exponential loss results in significantly more efficient schemes. We refer the reader to [GPSZ16] for a detailed discussion.

Distributional indistinguishability was also considered in the context of output compressing randomized encodings [LPST16]; indeed, this work implies that achieving DI security for FE for Turing machines with long outputs is impossible in the plain model. We note that our construction sidesteps this lower bound by considering Turing machines with a single output bit.

iO for TMs with unbounded memory has been constructed by [KLW15, AJS17] as discussed above, other prior works were limited to bounded space constraints. We note that [AJS17] additionally achieve constant overhead in the size of the obfuscated program as well as amortization, which we do not consider in this work. We also note that the work of [BGJS15] achieve miFE for TMs where the number of encrypting parties can be arbitrary, whereas we only support a-priori fixed, bounded number of parties.

The approach of using decomposable FE for circuits to construct FE for deterministic finite automata (DFA) in the single key setting was suggested by [AS17a]. In this work we develop and significantly generalize their ideas. In particular, we handle the unbounded key setting in FE for TMs which necessitates dealing with the much more complex indistinguishability style definition, for which we develop new proof techniques which use a novel "sliding trapdoor" approach and leverage distributional indistinguishability. In contrast, since [AS17a] use simulation security for single key FE, their proof must not contend with any of these challenges. Please see below for details.

Our Techniques. We describe an overview of our constructions, starting with single input FE, generalizing to multi-input FE and then building iO. All our constructions support the Turing machine model of computation. Our constructions rely on a single input FE scheme

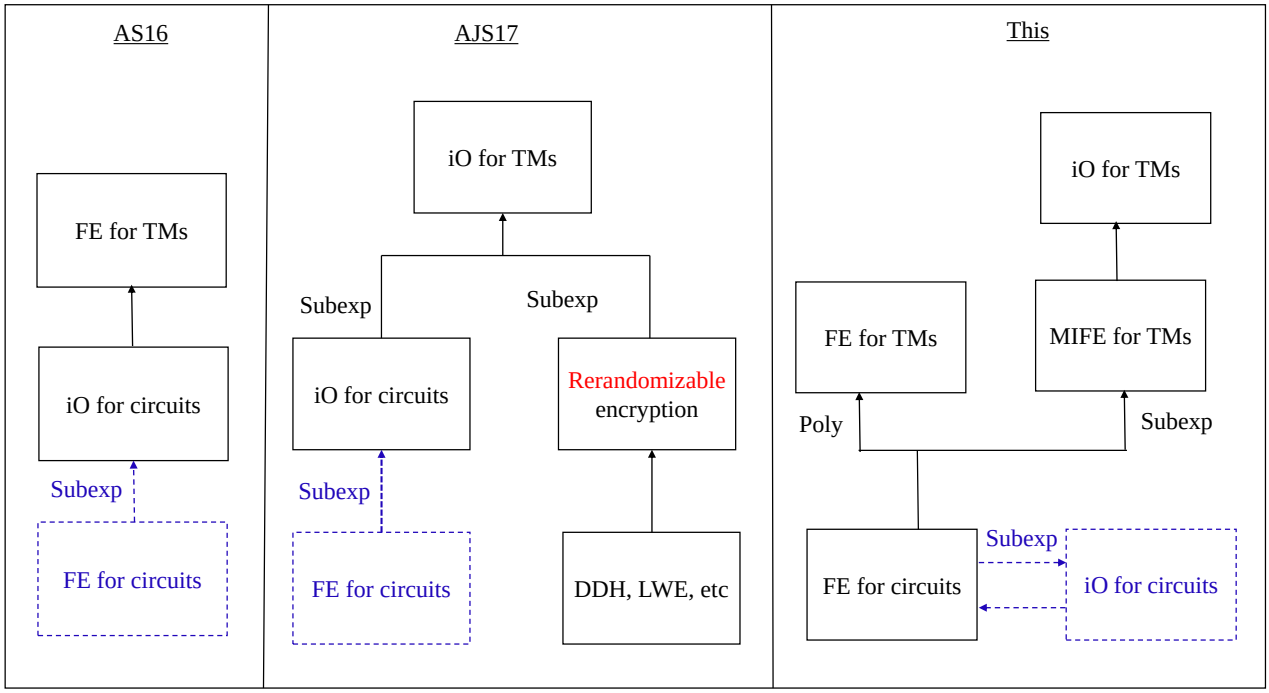


Figure 1: Prior work and our results. The reductions with subexponential loss are specified, no specification implies standard polynomial loss. The dashed blue lines indicate primitives that are not actually used by the work in question; we add these to elucidate the relationship between primitives. We do not include [BGJS15] here since it relies on public coin diO.

for circuits, denoted by CktFE, which satisfies decomposability. In Appendix F, we show that decomposable FE for circuits is implied by FE for circuits. Intuitively, decomposability means that the ciphertext $\mathsf{CT}_{\mathbf{x}}$ for a multi-bit message $\mathbf{x}$ be decomposable into multiple ciphertext components $\mathsf{CT}_i$ for $i \in |\mathbf{x}|$, one for each bit $x_i$ of the message. Moreover, the ciphertext components encoding individual bits of a single input are tied together by common randomness, that is $\mathsf{CT}_i = \mathcal{E}(\mathsf{PK}, r, x_i)$ where $\mathcal{E}$ is an encoding function and $r$ is common randomness used for all $i \in |\mathbf{x}|$[1]. The notion of decomposability has been widely studied and used in the context of randomized encodings, which may be seen as a special case of functional encryption; please see [AIK11] as an example. We note that all known FE schemes in the literature are already decomposable to the best of our knowledge, please see Appendix F.1 for a discussion.

Single Input TMFE. Recall that a Turing machine at any time step reads a symbol, state pair and produces a new symbol which is written to the work tape, a new state and a left or right head movement. By assuming the Turing machine is oblivious, the head movements of the TM may be fixed; thus, at any given time step when a work tape cell is read, we can compute the next time step when the same work tape cell will be accessed. This reduces the output at any time step $t$ to a symbol, state pair, where the state is read in the next time step $t+1$ and the symbol is read at a future (fixed) time step $t' > t$.

Our construction uses two CktFE schemes, $\mathsf{1FE}_1$ and $\mathsf{1FE}_2$, where $\mathsf{1FE}_2$ is decomposable. Intuitively, $\mathsf{1FE}_1$ is used by the encryptor to encode the unbounded length input, while $\mathsf{1FE}_2$ is used to mimic the computation of the Turing machine, as we describe next. The ciphertext

[1]Encoding of each bit may also use additional independent randomness, which is not relevant to the discussion here, and hence omitted.

of $\mathsf{1FE}_2$ is divided into two parts, encoding input components $(t, \sigma)$ and $q$ respectively. Here, $t$ is the current time step in the computation and $\sigma, q$ are the current work-tape symbol and state respectively. We maintain the invariant that at any time step $t$ in the computation, both components of the ciphertext have been computed using common randomness derived from $\mathsf{PRF}_{\mathsf{K}}((t\|\mathsf{salt}))$, where $\mathsf{salt}$ is an input chosen by the key generator and the PRF key $\mathsf{K}$ is chosen by the encryptor.

Now, to mimic the TM computation, we provide a function key for the $\mathsf{Next}$ functionality, that stores the transition table, receives as input the current (symbol, state) pair, computes the symbol to be written on the work tape and the next state using the transition table, derives the randomness using the PRF for the appropriate time step and outputs the encodings of the new (symbol, state) pair. In more detail, say the encryptor provides encodings of each input symbol $x_i$, for $i \in [|\mathbf{x}|]$, in addition to an encoding for the first (fixed) state $\mathsf{q}_{\mathsf{st}}$, where the encodings of $(1, x_1)$ and $\mathsf{q}_{\mathsf{st}}$ share the same randomness so that they may be concatenated to yield a complete ciphertext for $(1, x_1, \mathsf{q}_{\mathsf{st}})$. Now, the function key may read input $(1, x_1, \mathsf{q}_{\mathsf{st}})$, lookup the transition table and produce an encryption of the next state $q_2$ and the symbol to be written $x_2'$. The randomness used to encrypt $q_2$ is derived using a PRF as described above, and is the same as the randomness used by the encryptor to encode $(2, x_2)$. Hence, the two ciphertext components encoding $(2, x_2)$ and $q_2$ may be concatenated to yield a complete $\mathsf{1FE}_2$ ciphertext which may be again decrypted using the function key.

Now consider how to support writing on tape. Say the symbol $x_2'$ will be read at future fixed time step $t'$. Then the function key encodes the tuple $(t', x_2')$ using randomness $\mathsf{PRF}_{\mathsf{K}}((t'\|\mathsf{salt}))$. The state for time step $t'$, say $q'$ is computed at time step $t' - 1$, also using randomness $\mathsf{PRF}_{\mathsf{K}}((t'\|\mathsf{salt}))$. Thus, encodings of $(t', x_2')$ and $q'$ may be joined together to yield a complete $\mathsf{1FE}_2$ ciphertext which may be decrypted to propagate the computation.

A detail brushed away by the above description is that the encryptor, given input $\mathbf{x}$, cannot compute randomness generated by a PRF which has input a value $\mathsf{salt}$ chosen by the key generator. This is handled by making use of an additional scheme $\mathsf{1FE}_1$, which re-encrypts ciphertexts provided by the encryptor via a $\mathsf{ReRand}$ functionality, using the requisite randomness. Note that we support inputs of unbounded length by leveraging the fact that $\mathsf{CktFE}$ schemes $\mathsf{1FE}_1, \mathsf{1FE}_2$ support encryption of unbounded number of inputs, even if each must be of bounded length. Thus, the encryptor provides an unbounded number of $\mathsf{1FE}_1$ ciphertexts which are rerandomized and translated to ciphertexts under $\mathsf{1FE}_2$ using the $\mathsf{ReRand}$ function key provided by the key generator.

Decomposability. The above construction relies on the underlying $\mathsf{CktFE}$ scheme satisfying the property of decomposability. We note that decomposability is a mild assumption and already satisfied by all known $\mathsf{CktFE}$ constructions in the literature to the best of our knowledge (please see Appendix F.1 for a discussion). We can also remove the requirement of decomposability at the expense of making our compiler more complicated[2]. However, a cleaner approach is to build decomposable FE generically from standard FE, by using decomposable randomized encodings, which may be constructed from one way functions. Please see Appendix F for details.

Encoding the PRF key. The above informal description hides an important detail – for the function key to produce ciphertext components using a PRF, it must have the key of the

[2]Intuitively, we use decomposability because the "symbol" and "state" components of the ciphertext are generated during different times in decryption, say $T_1$ and $T_2$. However, since the underlying $\mathsf{CktFE}$ is compact, generating longer outputs comes for free. Hence, we can have the $\mathsf{CktFE}$ generate the complete (symbol, state) CT for the relevant symbol and all possible states at time $T_1$. Given in the clear, this would be insecure but this can be fixed by further nesting these CTs within a symmetric key encryption scheme and outputting them (in randomly permuted order). Later, at time $T_2$, when the state is computed, the decryption can output the SKE key to unlock the appropriate $\mathsf{CktFE}$ CT.

PRF, chosen by the encryptor[3], passed to it as input. Thus the ciphertext must additionally encode the PRF key along with inputs $(t, x, q)$. However, the ciphertext is constructed using randomness derived from the same PRF- resulting in circularity. We resolve this difficulty by using constrained PRFs [BW13, KPTZ13, BGI14], and having a ciphertext encode a PRF key that only allows computation of randomness for time steps of the future; this does not compromise its own security. For this constraint family, we provide a construction of cPRFs from standard assumptions. We believe this construction and the method of its application may be useful elsewhere[4].

More formally, our construction makes use of constrained, delegatable PRF for the function family $f_t : \{0,1\}^{2\cdot\lambda} \to \{0,1\}$ defined as follows.

$$\begin{aligned} f_t(x\|z) &= 1 \quad if \;\; x \geq t \\ &= 0 \quad otherwise \end{aligned}$$

We denote the constrained PRF key $\mathsf{K}_{f_t}$ by $\mathsf{K}_t$ for brevity. By the delegation property of constrained PRFs, we have that if $t' \geq t$ then $\mathsf{K}_{t'}$ can be derived from $\mathsf{K}_t$. The proof requires the PRF to be punctured at a fixed point in each hybrid, we provide a construction of delegatable punctured PRF in Appendix D.

Proof Overview. While the above description of single input $\mathsf{TMFE}$ is natural and intuitive, the proof of indistinguishability based security is quite subtle and requires new techniques as we discuss next. For ease of exposition, we describe the proof overview for the case where the adversary makes a single key request corresponding to some TM $M$. We must argue that the challenge ciphertext, which is a sequence of $\mathsf{1FE_1}$ ciphertexts, together with $\mathsf{ReRand}$ and $\mathsf{Next}$ keys corresponding to a TM $M$, do not distinguish the bit $b$.

As discussed above, the $\mathsf{1FE_1}$ ciphertexts are decrypted using the $\mathsf{ReRand}$ key to produce a sequence of $\mathsf{1FE_2}$ ciphertexts, each corresponding to a time step in the TM execution (when the encoded symbol is read), which are in turn decrypted by $\mathsf{Next}$ keys to compute new $\mathsf{1FE_2}$ ciphertexts for future time steps. We may view the $\mathsf{1FE_2}$ ciphertexts as forming a chain, with each link of the chain corresponding to a single step of the TM computation, and each ciphertext producing (via decryption) a new ciphertext for the next time step, finally yielding the output when the TM halts (after $T$ steps, say). Intuitively, since the output of the TM does not distinguish the bit $b$ by admissibility of the $\mathsf{TMFE}$ adversary, we may argue by security of $\mathsf{1FE_2}$ that the ciphertext at the penultimate step $T-1$ also does not distinguish $b$, which implies that the ciphertext at step $T-2$ hides $b$ and so on, ultimately yielding indistinguishability of the entire chain, and hence of the $\mathsf{1FE_1}$ challenge ciphertext.

Formalizing this intuitive argument is quite tricky. A natural approach would be to consider a sequence of hybrids, one corresponding to each link in the chain, and switch the $\mathsf{1FE_2}$ ciphertexts one by one starting from the end of the chain. While intuitive, this idea is misleading – note that a naive implementation of this idea would lead to a chain which is "broken": namely, its first links correspond to $b = 0$, and last links to $b = 1$. Since the ciphertext at a given step is decrypted to compute the ciphertext at the next step, a ciphertext corresponding to $b = 0$ cannot in general output a ciphertext for $b = 1$.

A standard approach to deal with this difficulty is to embed a "trapdoor" mode within the functionality [ABSV15, AJ15, BKS16] which lets us "hardwire" the ciphertexts that must be output by decryption directly in the key, allowing decryption to yield an inconsistent chain.

[3]Note that the PRF key must be encoded in the ciphertext rather than function key since it is required to be hidden.

[4]For instance, a similar situation w.r.t circularity arises in the original garbled RAM construction of Lu and Ostrovsky [LO13].

However, this approach also fails in our case, since the length of the chain is unbounded and there isn't sufficient space in the key to incorporate all its values.

Our Approach: "Sliding" Trapdoors. We deal with this difficulty by designing a novel "sliding-window" trapdoor approach which lets us hardwire the decryption chain "piece by piece". In more detail, we start with the last two time steps $(T, T-1)$, program the key to produce the output corresponding to $b = 1$ for time step $T$ and $b = 0$ for $T-1$, then transition to a world where the output corresponds to $b = 1$ for both $T$ and $T-1$. At this point, the hardwiring of the output for time step $T$ is redundant, since the ciphertext output by the decryption process at time step $T-1$ automatically computes the output coresponding to $b = 1$ at time step $T$. Thus, we may now slide the trapdoor to program to the next pair $(T-1, T-2)$, switching the decryption output at time step $T-2$ to $b = 1$ and so on, until the entire chain corresponds to $b = 1$.

Intuitively, we are "programming" the decryption only for outputs at both ends of the "broken link", so that preceding links are generated using $b = 0$ and subsequent links are generated using $b = 1$. We leverage the fact that the chain links corresponding to future time-steps are encoded implicitly in a given time step – hence if we manage to hide the chain inconsistency at a certain position $i$, this implies that the remainder of the chain is constructed using the bit encoded at step $i$. Formalizing this argument requires a great deal of care, as we must keep track of the "target" time steps corresponding to the two ends of the broken link that are being programmed, the time steps at which the symbol and state ciphertexts are generated to be "consumed" at the target time-steps, the particular values that must be encoded in the symbol, state fields in both cases as well as the key that is being handled at a given time in the proof. For more details, please see Section 3.3.

Generalising to Multi-Input FE for Turing machines. For the $k$ party setting, a natural idea is to have each party encrypt its own input $\mathbf{x}_i$, and use a $k$ input $\mathsf{CktFE}$ scheme $\mathsf{kFE}$ [BKS16, KS17], to "aggregate" these into the "input" ciphertext $\mathsf{CT}(\mathbf{x})$ for one long input $\mathbf{x} = (\mathbf{x}_1 \| \mathbf{x}_2 \| \ldots \| \mathbf{x}_k)$, under a different $\mathsf{CktFE}$ scheme $\mathsf{1FE}$. Note that the length of $\mathbf{x}$ is unknown hence it may not be encoded "all at once" but must be encoded bit by bit as in the previous scheme. Now, by additionally providing the $\mathsf{1FE}$ ciphertext encoding the start state of the Turing machine $\mathsf{CT}(\mathsf{q_{st}})$, and a function key to compute the transition table of the TM as in the previous scheme, we may proceed with the computation exactly as before.

Formalizing this idea must contend with several hurdles. In the multi-input setting, the $i^{th}$ encryptor may encode multiple inputs and functionality permits "mix and match" of ciphertexts in the sense that any input encoded by party $i$ may be combined with any input encoded by parties $j \in [k]$, $j \neq i$. Therefore, if each of $k$ parties encodes $T$ ciphertexts, there are $T^k$ valid input combinations that the TM may execute on. However, when the TM is executing on any input combination, we must ensure that it cannot mix and match symbol, state pairs across different input combinations. Moreover, an encryption for a symbol, state pair produced by some machine $M_i$ should not be decryptable by any machine $M_j$ for $j \neq i$. These issues are handled by careful design of the aggregate functionality to ensure that an execution thread of any input combination by any machine is separate from any other. The proof extends naturally from the single input case. Please see Section 4 for details.

Distributional Indistinguishability. As discussed above, our constructions rely on the security notion of distributional indistinguishability ($\mathsf{DI}$) for functional encryption for circuits [GHRW14]. Intuitively, this notion says that if the outputs produced by a circuit on two input distributions

are merely indistinguishable (as against exactly equal), then the ciphertexts encoding those inputs must also be indistinguishable. In Appendix E we give a construction of DI secure single input FE from standard FE.

Indistinguishability Obfuscation. Constructing iO for TMs given miFE for TM is straightforward, and adapts the miFE to iO circuit compiler by [GGG+14] to the TM setting. As in the circuit case, an miFE for TM that supports two ciphertext queries and single key query suffices for this transformation. Please see Section 5 for details. Since our security proof for miFE for TM is tight, this compiler yields iO for TM from sub-exponentially secure FE for circuits rather than sub-exponentially secure iO for circuits.

Organization of the paper. The paper is organized as follows. In Section 2 we provide the definitions and preliminaries used by our constructions. In Section 3, we provide our construction for single input FE for Turing machines. In Section 4, we provide our construction for multi-input FE for Turing machines for any fixed arity $k$ and in Section 5 we describe the construction of iO for Turing machines for bounded inputs. Our constructions use constrained PRFs which are instantiated in Appendix D and decomposable FE which is constructed in Appendix F.

## 2 Preliminaries

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

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

We say a function $f(n)$ is negligible if it is $O(n^{-c})$ for all $c > 0$, and we use $\mathrm{negl}(n)$ to denote a negligible function of $n$. We say $f(n)$ is polynomial if it is $O(n^c)$ for some $c > 0$, 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 $x$.

### 2.1 Definitions: FE for Circuits

In this section, we define functional encryption for circuits, in both the single and multi-input setting.

#### 2.1.1 Single Input Functional Encryption for Circuits

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 $f \in \mathcal{F}_\lambda$ takes as input a string $\mathbf{x} \in \mathcal{X}_\lambda$ and outputs $f(\mathbf{x}) \in \mathcal{Y}_\lambda$.

A functional encryption scheme CktFE for $\mathcal{F}$ consists of four algorithms CktFE = (CktFE.Setup, CktFE.Keygen, CktFE.Enc, CktFE.Dec) defined as follows.

- $\mathsf{CktFE.Setup}(1^\lambda)$ is a PPT algorithm that takes as input the unary representation of the security parameter and outputs the master public and secret keys $(\mathsf{PK}, \mathsf{MSK})$. Sometimes, the CktFE.Setup algorithm may also accept as input a parameter $1^\ell$, denoting the length of the input. In this case, the input lives in domain $\mathcal{X}^\ell$.

- $\mathsf{CktFE.Keygen}(\mathsf{MSK}, f)$ is a PPT algorithm that takes as input the master secret key $\mathsf{MSK}$ and a circuit $f \in \mathcal{F}_\lambda$ and outputs a corresponding secret key $\mathsf{SK}_f$.
- $\mathsf{CktFE.Enc}(\mathsf{PK}, \mathbf{x})$ is a PPT algorithm that takes as input the master public key $\mathsf{PK}$ and an input message $\mathbf{x} \in \mathcal{X}_\lambda$ and outputs a ciphertext $\mathsf{CT}$.
- $\mathsf{CktFE.Dec}(\mathsf{SK}_f, \mathsf{CT}_\mathbf{x})$ is an (a deterministic) algorithm that takes as input the secret key $\mathsf{SK}_f$ and a ciphertext $\mathsf{CT}_\mathbf{x}$ and outputs $f(\mathbf{x})$.

Definition 2.1 (Correctness). A functional encryption scheme $\mathsf{CktFE}$ is correct if for all $\lambda \in \mathbb{N}$, all $f \in \mathcal{F}_\lambda$ and all $x \in \mathcal{X}_\lambda$,

$$\Pr\left[\begin{array}{l}(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{CktFE.Setup}(1^\lambda); \\ \mathsf{CktFE.Dec}\Big(\mathsf{CktFE.Keygen}(\mathsf{MSK}, f), \mathsf{CktFE.Enc}(\mathsf{PK}, \mathbf{x})\Big) \neq f(\mathbf{x})\end{array}\right] = \mathrm{negl}(\lambda)$$

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

Definition 2.2 ( Compactness [AJ15]). A functional encryption scheme for circuits is said to be compact if for any input message $\mathbf{x}$, the running time of the encryption algorithm is polynomial in the security parameter and the size of $\mathbf{x}$. In particular, it does not depend on the circuit description size or the output length of any function $f$ supported by the scheme.

A weaker version of compactness, known as succinct or semi-compact FE, allows the run time of the encryption algorithm to depend on the output length of the functions. Equivalently, a semi-compact FE scheme is simply a compact FE scheme when we restrict our attention to functions with single-bit outputs.

Distributional Indistinguishability for Circuit FE. In this section we define the notion of distributional indistinguishability for functional encryption for circuits. The notion was first defined by [GHRW14, Sec 3.4] in the context of reusable garbled circuits, i.e. single key functional encryption but may be generalized to the multi-key setting in a straightforward way. Intuitively, this notion says that if the outputs produced by a circuit on two input distributions are indistinguishable, then the ciphertexts encoding those inputs must also be indistinguishable.

Definition 2.3. A functional encryption scheme $\mathcal{F}$ for a circuit family $\mathcal{G}$ is secure in the distributional indistinguishability game, if for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following experiment is negligible in the security parameter $\lambda$:

1. Public Key: Challenger returns $\mathsf{PK}$ to the adversary.
2. Pre-Challenge Key Queries: $\mathcal{A}$ may adaptively request keys for any circuits $g_i \in \mathcal{G}$. In response, $\mathcal{A}$ is given the corresponding keys $\mathsf{SK}_{g_i}$. This step may be repeated any polynomial number of times by the attacker.
3. Challenge Declaration: $\mathcal{A}(1^\lambda, \mathsf{PK})$ outputs two ensembles of challenge distributions $\big(D_0(\lambda), D_1(\lambda)\big)$[5] to the challenger, subject to the restriction that for any $\mathbf{x}_0 \leftarrow D_0, \mathbf{x}_1 \leftarrow D_1$, it holds that $g_i(\mathbf{x}_0) \overset{c}{\approx} g_i(\mathbf{x}_1)$ for all $i$.
4. Challenge CT: $\mathcal{A}$ requests the challenge ciphertext, to which challenger chooses a random bit $b$, samples $\mathbf{x}_b \leftarrow D_b$ and returns the ciphertext $\mathsf{CT}_{\mathbf{x}_b}$.

[5]We omit the parameter $\lambda$ in what follows for brevity of notation.

5. Key Queries: The adversary may continue to request keys for additional functions $g_i$, subject to the same restriction that for any $\mathbf{x}_0 \leftarrow D_0, \mathbf{x}_1 \leftarrow D_1$, it holds that $g_i(\mathbf{x}_0) \overset{c}{\approx} g_i(\mathbf{x}_1)$ for all $i$.

6. $\mathcal{A}$ outputs a bit $b'$, and succeeds if $b' = b$.

The advantage of $\mathcal{A}$ is the absolute value of the difference between its success probability and $1/2$. In the selective game, the adversary is required to declare the challenge distributions in the very first step, without seeing the public key.

Comparison with Standard Indistinguishability. We note that the standard insitinguishability game is implied by the above by restricting the adversary to choose distributions $D_0, D_1$ above to simply be two messages $\mathbf{x}_0, \mathbf{x}_1$ with probability 1 and requesting keys that satisfy $g_i(\mathbf{x}_0) = g_i(\mathbf{x}_1)$ for all $i$, which is a special case of $g_i(\mathbf{x}_0) \overset{c}{\approx} g_i(\mathbf{x}_1)$.

Decomposable functional encryption for circuits In this section, we recall the notion of decomposable functional encryption (DFE) defined by [AS17a]. Decomposable functional encryption is analogous to the notion of decomposable randomized encodings [AIK14]. Intuitively, decomposability requires that the public key $\mathsf{PK}$ and the ciphertext $\mathsf{CT}_{\mathbf{x}}$ of a functional encryption scheme be decomposable into components $\mathsf{PK}_i$ and $\mathsf{CT}_i$ for $i \in [|\mathbf{x}|]$, where $\mathsf{CT}_i$ depends on a single deterministic bit $x_i$ and the public key component $\mathsf{PK}_i$. In addition, the ciphertext may contain components that are independent of the message and depend only on the randomness.

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 $\mathsf{CT}_i$ 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 $|\mathbf{y}| = k$. We assume that given the security parameter, the spaces $\mathcal{P}$, $\mathcal{R}_1$, $\mathcal{R}_2$, $\mathcal{C}$ are fixed, and the length of the message $|\mathbf{x}|$ can be any polynomial.

### 2.1.2 Multi-Input Functional Encryption for Circuits

We define the notion of private-key $t$-input functional encryption for circuits here. Our definition follows that of [KS17].

Let $\forall i \in [t], \mathcal{X}_i = \{(\mathcal{X}_i)\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ be ensembles of finite sets, and let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be an ensemble of finite $t$-ary function families. For each $\lambda \in \mathbb{N}$, each function $f \in \mathcal{F}_\lambda$ takes as input $t$ strings, $\mathbf{x}_1 \in (\mathcal{X}_1)_\lambda, \ldots, \mathbf{x}_t \in (\mathcal{X}_t)_\lambda$, and outputs a value $f(\mathbf{x}_1, \ldots, \mathbf{x}_t) \in \mathcal{Y}_\lambda$.

A private-key $t$-input functional encryption scheme $t$-$\mathsf{CktFE}$ for $\mathcal{F}$ consists of four algorithms $t$-$\mathsf{CktFE} = (t\text{-}\mathsf{CktFE.Setup}, t\text{-}\mathsf{CktFE.Keygen}, t\text{-}\mathsf{CktFE.Enc}, t\text{-}\mathsf{CktFE.Dec})$ defined as follows.

- $t$-$\mathsf{CktFE.Setup}(1^\lambda)$ is a $\mathsf{PPT}$ algorithm that takes as input the unary representation of the security parameter and outputs the master secret key $\mathsf{MSK}$.
- $t$-$\mathsf{CktFE.Keygen}(\mathsf{MSK}, f)$ is a $\mathsf{PPT}$ algorithm that takes as input the master secret key $\mathsf{MSK}$ and a circuit $f \in \mathcal{F}_\lambda$ and outputs a corresponding secret key $\mathsf{SK}_f$.
- $t$-$\mathsf{CktFE.Enc}(\mathsf{MSK}, \mathbf{m}, \mathsf{ind})$ is a $\mathsf{PPT}$ algorithm that takes as input the master secret key $\mathsf{MSK}$, an input message $\mathbf{m} = \mathbf{x}_i \in (\mathcal{X}_i)_\lambda$ if $\mathsf{ind} = i, i \in [t]$, and outputs a ciphertext $\mathsf{CT}_{\mathsf{ind}}$.
- $t$-$\mathsf{CktFE.Dec}(\mathsf{SK}_f, (\mathsf{CT}_1, \ldots, \mathsf{CT}_t))$ is an (a deterministic) algorithm that takes as input the secret key $\mathsf{SK}_f$ and $t$ ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_t$ and outputs a string $y \in \mathcal{Y}_\lambda \cup \perp$.

Definition 2.4 (Correctness). A private-key $t$-input functional encryption scheme $t$-$\mathsf{CktFE}$ is correct if for all $\lambda \in \mathbb{N}$, $f \in \mathcal{F}_\lambda$ and all $(\mathbf{x}_1, \ldots, \mathbf{x}_t) \in (\mathcal{X}_1)_\lambda \times \ldots \times (\mathcal{X}_t)_\lambda$,

$$\Pr\left[\begin{array}{c} t\text{-}\mathsf{CktFE.Dec}\Big(t\text{-}\mathsf{CktFE.Keygen}(\mathsf{MSK}, f), \big(t\text{-}\mathsf{CktFE.Enc}(\mathsf{MSK}, \mathbf{x}_1, 1), \ldots, \\ t\text{-}\mathsf{CktFE.Enc}(\mathsf{MSK}, \mathbf{x}_t, t)\big)\Big) \neq f(\mathbf{x}_1, \ldots, \mathbf{x}_t) \end{array}\right] = \mathrm{negl}(\lambda)$$

Here, $\mathsf{MSK} \leftarrow t\text{-}\mathsf{CktFE.Setup}(1^\lambda)$ and probability is taken over the random coins of $t$-$\mathsf{CktFE.Setup}$, $t$-$\mathsf{CktFE.Enc}$ and $t$-$\mathsf{CktFE.Keygen}$.

Distributional Indistinguishability. We define the notion of distributional indistinguishability for a $t$-input functional encryption scheme for circuits. To begin, we describe a valid $t$-input adversary.

Definition 2.5 (Valid $t$-Input Adversary). A $\mathsf{PPT}$ algorithm $\mathcal{A}$ is a valid $t$-input adversary if for all private-key $t$-input functional encryption schemes over message space $(\mathcal{X}_1)_\lambda \times \ldots \times (\mathcal{X}_t)_\lambda$, and a circuit space $\mathcal{F}$, for any $(f_0, f_1)$ queried by the adversary, and any $t$ pairs of input distribution ensembles $(D_{01}(\lambda), D_{11}(\lambda)), \ldots, (D_{0t}(\lambda), D_{1t}(\lambda))$[6] output by the adversary such that $D_{bj}$ is a distribution over $\mathcal{X}_j$ for $b \in \{0, 1\}$, $j \in [t]$, it holds that

$$f_0(\mathbf{x}_{01}, \ldots, \mathbf{x}_{0t}) \overset{c}{\approx} f_1(\mathbf{x}_{11}, \ldots, \mathbf{x}_{1t}),$$

where $\mathbf{x}_{bj} \leftarrow D_{bj}$ for $b \in \{0, 1\}$, $j \in [t]$.

We define the following game between a challenger and an adversary:

1. Key Queries. $\mathcal{A}$ may adaptively submit key requests for pairs of functions $(f_0, f_1) \in \mathcal{F}$. In response, $\mathcal{A}$ is given the corresponding keys $\mathsf{SK}_{f_b}$ for some random bit $b$ chosen by the challenger. This step may be repeated any polynomial number of times by the attacker.
2. Ciphertext Queries. $\mathcal{A}(1^\lambda)$ submits ciphertext requests for pairs of challenge distribution ensembles $(D_{01}, D_{11}), \ldots, (D_{0t}, D_{1t})$ to the challenger. The challenger samples $\mathbf{x}_j \leftarrow D_{bj}$ for $j \in [t]$ and returns $t$-$\mathsf{CktFE.Enc}(\mathsf{MSK}, \mathbf{x}_j, j), \forall j \in [t]$. This step may be repeated any polynomial number of times by the attacker.

[6] We omit the argument $\lambda$ where it is implicit for notational brevity.

3. Guess. $\mathcal{A}$ outputs a bit $b'$, and succeeds if $b' = b$.

In the above definition, ciphertext and key queries may be interspersed in any order. The advantage of $\mathcal{A}$ is the absolute value of the difference between its success probability and $1/2$. In the selective game, the adversary is required to declare the challenge ciphertext distributions in the very first step, without seeing the public key.

Definition 2.6. A $t$-input functional encryption scheme $t$-$\mathsf{CktFE}$ for a circuit family $\mathcal{F}$ is secure in the distributional indistinguishability game, if for all valid $\mathsf{PPT}$ adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above game is negligible in the security parameter $\lambda$.

We note that the standard indistinguishability game is the special case where the adversary submits challenge messages rather than distributions and all queried functions must output exactly the same rather than indistinguishable values.

## 2.2 Definitions: FE for Turing Machines

In this section, we will define functional encryption for Turing Machines (TM). The definition of Turing machines and oblivious Turing machines is recalled in Appendix A. Functional encryption for TMs is defined analogously to functional encryption for circuits, except that secret keys correspond to TMs rather than circuits. Thus, secret keys can be used to decrypt ciphertexts of messages of arbitrary length and the decryption time depends only the input-specific run time of the TM on the message, not the worst case run time. We denote the runtime of a TM $M$ (i.e. number of steps the head takes) on an input $\mathbf{w}$ by $\mathsf{runtime}(M, \mathbf{w})$.

### 2.2.1 Single Input Functional Encryption for Turing Machines

Let $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of Turing machines with alphabet $\Sigma = \{\Sigma_\lambda\}_{\lambda\in\mathbb{N}}$ and the running time upper-bounded by a polynomial in $\lambda$. A functional encryption scheme $\mathsf{TMFE}$ for a Turing machine family $\mathcal{M}$ consists of four algorithms $\mathsf{TMFE} = (\mathsf{TMFE.Setup}, \mathsf{TMFE.KeyGen}, \mathsf{TMFE.Enc}, \mathsf{TMFE.Dec})$ defined as follows.

- $\mathsf{TMFE.Setup}(1^\lambda)$ is a $\mathsf{PPT}$ algorithm that takes as input the unary representation of the security parameter and outputs the master public and secret keys $(\mathsf{PK}, \mathsf{MSK})$.
- $\mathsf{TMFE.KeyGen}(\mathsf{MSK}, M)$ is a $\mathsf{PPT}$ algorithm that takes as input the master secret key $\mathsf{MSK}$ and a TM $M$ and outputs a corresponding secret key $\mathsf{SK}_M$.
- $\mathsf{TMFE.Enc}(\mathsf{PK}, \mathbf{x})$ is a $\mathsf{PPT}$ algorithm that takes as input the master public key $\mathsf{PK}$, and an input message $\mathbf{x} \in \Sigma_\lambda^*$ of arbitrary length, outputs a ciphertext $\mathsf{CT}_\mathbf{x}$.
- $\mathsf{TMFE.Dec}(\mathsf{SK}_M, \mathsf{CT}_\mathbf{x})$ is an (a deterministic) algorithm that takes as input the secret key $\mathsf{SK}_M$ and a ciphertext $\mathsf{CT}_\mathbf{x}$ and outputs a bit $b$.

Definition 2.7 (Correctness). A functional encryption scheme $\mathsf{TMFE}$ is correct if for all $M \in \mathcal{M}$ and all $\mathbf{x} \in \Sigma^*$,

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

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

Efficiency [AS16]. The efficiency property of a public-key FE scheme for Turing machines says that the algorithm $\mathsf{TMFE.Setup}$ on input $1^\lambda$ should run in time polynomial in $\lambda$, $\mathsf{TMFE.KeyGen}$ on input the Turing machine $M$ and the master key $\mathsf{MSK}$ should run in time polynomial in $(\lambda, |M|)$, $\mathsf{TMFE.Enc}$ on input a message $\mathbf{x}$ and the public key should run in time polynomial in $(\lambda, |\mathbf{x}|)$. Finally, $\mathsf{TMFE.Dec}$ on input a functional key of $M$ and an encryption of $\mathbf{x}$ should run in time polynomial in $(\lambda, |M|, |\mathbf{x}|, \mathsf{runtime}(M, \mathbf{x}))$.

Distributional Indistinguishability for $\mathsf{TMFE}$. In this section we define the notion of distributional indistinguishability based security for functional encryption for Turing machines. This notion was first considered by [GHRW14] in the context of single key FE for circuits.

Definition 2.8. A functional encryption scheme $\mathcal{F}$ for a TM family $\mathcal{M}$ is secure in the distributional indistinguishability game, if for all $\mathsf{PPT}$ adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following experiment is negligible in the security parameter $\lambda$:

1. Public Key: Challenger returns $\mathsf{PK}$ to the adversary.
2. Pre-Challenge Key Queries: $\mathcal{A}$ may adaptively request keys for any TMs $M_i \in \mathcal{M}$. In response, $\mathcal{A}$ is given the corresponding keys $\mathsf{SK}_{M_i}$. This step may be repeated any polynomial number of times by the attacker.
3. Challenge Declaration: $\mathcal{A}(1^\lambda, \mathsf{PK})$ outputs two challenge distribution ensembles $(D_0(\lambda), D_1(\lambda))$[7] to the challenger, subject to the restriction that for any $\mathbf{x}_0 \leftarrow D_0, \mathbf{x}_1 \leftarrow D_1$, it holds that $M_i(\mathbf{x}_0) \overset{c}{\approx} M_i(\mathbf{x}_1)$ for all $i$.
4. Challenge CT: $\mathcal{A}$ requests the challenge ciphertext, to which challenger chooses a random bit $b$, samples $\mathbf{x}_b \leftarrow D_b$ and returns the ciphertext $\mathsf{CT}_{\mathbf{x}_b}$.
5. Key Queries: The adversary may continue to request keys for additional functions, subject to the same restriction that for any $\mathbf{x}_0 \leftarrow D_0, \mathbf{x}_1 \leftarrow D_1$, it holds that $M_i(\mathbf{x}_0) \overset{c}{\approx} M_i(\mathbf{x}_1)$ for all $i$.
6. $\mathcal{A}$ outputs a bit $b'$, and succeeds if $b' = b$.

The advantage of $\mathcal{A}$ is the absolute value of the difference between its success probability and $1/2$. In the selective game, the adversary is required to declare the challenge distributions in the very first step, without seeing the public key.

Comparison with Standard Indistinguishability. We note that the standard indistinguishability game is implied by the above by restricting the adversary to choose distributions $D_0, D_1$ above to simply be two messages $\mathbf{x}_0, \mathbf{x}_1$ with probability 1 and requesting keys that satisfy $M_i(\mathbf{x}_0) = M_i(\mathbf{x}_1)$ for all $i$.

### 2.2.2 Multi-Input Functional Encryption for Turing Machines

In this section, we define multi input functional encryption ($\mathsf{miFE}$) for Turing machines. Our definition generalizes the $\mathsf{CktFE}$ definition of [BKS16]. Our definition supports a fixed number of encryptors, where each of $k$ (say) encryptors is associated with an index $\mathsf{ind} \in [k]$. An encryptor may choose an input string of unbounded length, denoted by $\ell_{\mathsf{ind}}$. We note that our definition is weaker than that of [BGJS15], who allow for unbounded number of encryptors and each encryptor to have a unique encryption key, a subset of which may be requested by the

[7] We omit the argument $\lambda$ where it is implicit for notational brevity.

adversary. By contrast, our definition, following [BKS16], requires all encryptors to use the same $\mathsf{MSK}$ and evidently cannot allow the attacker to request this.

Let $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of Turing machines with alphabet $\Sigma = \{\Sigma_\lambda\}_{\lambda\in\mathbb{N}}$ and the running time upper-bounded by a polynomial in $\lambda$. A multi-input functional encryption scheme for $\mathcal{M}$ consists of a tuple of four algorithms $\mathsf{kTMFE} = (\mathsf{kTMFE.Setup}, \mathsf{kTMFE.KeyGen}, \mathsf{kTMFE.Enc}, \mathsf{kTMFE.Dec})$ defined as follows.

- $\mathsf{kTMFE.Setup}(1^\lambda, 1^k)$ is a $\mathsf{PPT}$ algorithm that takes as input the unary representation of the security parameter and the number of users $k$ and outputs the master secret key $\mathsf{MSK}$.
- $\mathsf{kTMFE.KeyGen}(\mathsf{MSK}, M)$ is a $\mathsf{PPT}$ algorithm that takes as input master secret key $\mathsf{MSK}$ and a TM $M$ and outputs a corresponding secret key $\mathsf{SK}_M$.
- $\mathsf{kTMFE.Enc}(\mathsf{MSK}, \mathbf{x}_{\mathsf{ind}}, \mathsf{ind})$ is a $\mathsf{PPT}$ algorithm that takes as input the master secret key $\mathsf{MSK}$, an index $\mathsf{ind} \in [k]$ denoting the party number, and an input message $\mathbf{x}_{\mathsf{ind}}$ of arbitrary length and outputs a ciphertext $\mathsf{CT}_{\mathbf{w}_{\mathsf{ind}}}$.
- $\mathsf{kTMFE.Dec}(\mathsf{SK}_M, \{\mathsf{CT}_{\mathbf{x}_{\mathsf{ind}}}\}_{\mathsf{ind}\in[\mathsf{k}]})$ is an (a deterministic) algorithm that takes as input the functional secret key $\mathsf{SK}_M$ and $k$ ciphertexts $\mathsf{CT}_{\mathbf{x}_1}, \ldots, \mathsf{CT}_{\mathbf{x}_k}$ and outputs a bit $b$.

Definition 2.9 (Correctness). A functional encryption scheme $\mathsf{kTMFE}$ is correct if for all $M \in \mathcal{M}$ and all $\mathbf{x}_i \in \Sigma^*$ for $i \in [k]$,

$$\Pr\left[\begin{array}{l}\mathsf{kTMFE.Dec}\Big(\mathsf{kTMFE.KeyGen}(\mathsf{MSK}, M), \mathsf{kTMFE.Enc}(\mathsf{MSK}, \mathbf{x}_1, 1),\\ \ldots, \mathsf{kTMFE.Enc}(\mathsf{MSK}, \mathbf{x}_k, k)\Big) \neq \mathsf{M}(\mathbf{x}_1\| \ldots \|\mathbf{x}_k)\end{array}\right] = \mathrm{negl}(\lambda)$$

where $\mathsf{MSK} \leftarrow \mathsf{kTMFE.Setup}(1^\lambda, 1^k)$ and the probability is taken over the coins of $\mathsf{kTMFE.Setup}$, $\mathsf{kTMFE.KeyGen}$, and $\mathsf{kTMFE.Enc}$.

Efficiency is as defined in Section 2.2.

Distributional Indistinguishability for $\mathsf{kTMFE}$. In this section we define the notion of distributional indistinguishability based security for multi-input functional encryption for Turing machines. To begin, we define the notion of a valid $k$-input adversary analogously to the case of circuits [BKS16].

Definition 2.10 (Valid $k$-Input Adversary). A $\mathsf{PPT}$ algorithm $\mathcal{A}$ is a valid $k$-input adversary, if for a Turing machine space $\mathcal{M}$ with alphabet $\Sigma$, for all private key $k$-input functional encryption schemes $\mathsf{kTMFE}$ over message space $\mathcal{X}_1^* \times \ldots \times \mathcal{X}_k^*$ such that $\mathcal{X}_j^* \subset \Sigma^*$ for all $j \in [k]$, for any $M \in \mathcal{M}$ queried by the adversary, and any $k$ pairs of input distribution ensembles $(D_{01}(\lambda), D_{11}(\lambda)), (D_{02}(\lambda), D_{12}(\lambda)), \ldots, (D_{0k}(\lambda), D_{1k}(\lambda))$[8] output by the adversary such that $D_{bj}$ is a distribution over $\mathcal{X}_j^*$ for $b \in \{0,1\}$, $j \in [k]$, it holds that

$$M(\mathbf{x}_{01}\| \ldots \|\mathbf{x}_{0k}) \overset{c}{\approx} M(\mathbf{x}_{11}\| \ldots \|\mathbf{x}_{1k})$$

where $\mathbf{x}_{bj} \leftarrow D_{bj}$ for $b \in \{0,1\}$, $j \in [k]$.

We define the following game between a challenger and an adversary:

[8] We omit the argument $\lambda$ where it is implicit for notational brevity.

1. Key Queries. $\mathcal{A}$ may adaptively submit key requests for TMs $M_i \in \mathcal{M}$. In response, $\mathcal{A}$ is given the corresponding keys $\mathsf{SK}_{M_i}$ for some random bit $b$ chosen by the challenger. This step may be repeated any polynomial number of times by the attacker.

2. Ciphertext Queries. $\mathcal{A}(1^\lambda)$ submits ciphertext requests for $k$ pairs of challenge distribution ensembles $(D_{01}, D_{11}), (D_{02}, D_{12}), \ldots, (D_{0k}, D_{1k})$ to the challenger. The challenger samples $\mathbf{x}_{bj} \leftarrow D_{bj}$ for $j \in [k]$ and returns $\mathsf{kTMFE.Enc}(\mathsf{MSK}, \mathbf{x}_{bj}, j)$ for all $j \in [k]$. This step may be repeated any polynomial number of times by the attacker.

3. Guess. $\mathcal{A}$ outputs a bit $b'$, and succeeds if $b' = b$.

In the above definition, ciphertext and key queries may be interspersed in any order. The advantage of $\mathcal{A}$ is the absolute value of the difference between its success probability and $1/2$. In the selective game, the adversary is required to declare the challenge ciphertext distributions in the very first step, without seeing the public key.

Definition 2.11. A multi input functional encryption scheme $\mathsf{kTMFE}$ for a TM family $\mathcal{M}$ is secure in the distributional indistinguishability game, if for all valid $\mathsf{PPT}$ adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above game is negligible in the security parameter $\lambda$.

We note that the standard indistinguishability game is the special case where the adversary submits challenge messages rather than distributions and all queried machines must output exactly the same rather than indistinguishable values.

### 2.2.3 Indistinguishability Obfuscation for Turing Machines

As in prior work, we construct $\mathsf{iO}$ for Turing machines (TMs) in the setting where the input length is fixed a-priori. A uniform $\mathsf{PPT}$ machine $\mathsf{iO}$ is an indistinguishability obfuscator for a class of Turing machines $\{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ with input length $L$, if the following conditions are satisfied:

1. Correctness. For all security parameters $\lambda \in \mathbb{N}$, for any $M \in \mathcal{M}_\lambda$ and every input $\mathbf{x} \in \{0,1\}^{\leq L}$, we have that:

$$\Pr\left[M' \leftarrow \mathsf{iO}(1^\lambda, M, L) : M'(\mathbf{x}) = M(\mathbf{x})\right] = 1$$

   where the probability is taken over the coin-tosses of the obfuscator $\mathsf{iO}$.

2. Indistinguishability of Equivalent TMs. For every ensemble of pairs of Turing machines $\{M_{0,\lambda}, M_{1,\lambda}\}_{\lambda \in \mathbb{N}}$, such that $M_{0,\lambda}(\mathbf{x}) = M_{1,\lambda}(\mathbf{x})$ for every $\mathbf{x} \in \{0,1\}^{\leq L}$ and $\mathsf{runtime}(M_{0,\lambda}, \mathbf{x}) = \mathsf{runtime}(M_{1,\lambda}, \mathbf{x})$, we have that the following ensembles of pairs of distributions are indistinguishable to any $\mathsf{PPT}$ adversary $\mathsf{Adv}$:

$$\left\{M_{0,\lambda}, M_{1,\lambda}, \mathsf{iO}(1^\lambda, M_{0,\lambda})\right\} \overset{c}{\approx} \left\{M_{0,\lambda}, M_{1,\lambda}, \mathsf{iO}(1^\lambda, M_{1,\lambda})\right\}$$

3. Succinctness. For all security parameters $\lambda \in \mathbb{N}$, for any $M \in \mathcal{M}_\lambda$ , we have that the running time of $\mathsf{iO}(1^\lambda, M, L)$ is $\mathrm{poly}(\lambda, |M|, L)$ and the evaluation time of $\mathsf{iO}(M)$ on input $\mathbf{x}$ where $\mathbf{x} \in \{0,1\}^{\leq L}$, is $\mathrm{poly}(|M|, L, t)$ where $t = \mathsf{runtime}(M, \mathbf{x})$.

## 2.3 Constrained Pseudorandom Functions

Constrained pseudorandom functions (introduced concurrently by Boneh and Waters (CCS 2013), Boyle, Goldwasser, and Ivan (PKC 2014), and Kiayias, Papadopoulos, Triandopoulos,

and Zacharias (CCS 2013)), are pseudorandom functions (PRFs) that allow the owner of the secret key $K$ to compute a constrained key $K_f$ , such that anyone who possesses $K_f$ can compute the output of the PRF on any input $x$ such that $f(x) = 1$ for some predicate $f$. The security requirement of constrained PRFs state that the PRF output must still look indistinguishable from random for any $x$ such that $f(x) = 0$. We will also require the property of delegatability, formalized below.

Definition 2.12. [BW13] Let $F : \{0,1\}^{\mathsf{seed}(\lambda)} \times \{0,1\}^{\mathsf{in}(\lambda)} \to \{0,1\}^{\mathsf{out}(\lambda)}$ be an efficient function, where $\mathsf{seed}$, $\mathsf{in}$ and $\mathsf{out}$ are all polynomials in the security parameter $\lambda$. We say that $F$ is a delegatable constrained pseudorandom function with respect to a set system $\mathcal{S} \subseteq 2^{\{0,1\}^{\mathsf{in}(\lambda)}}$ if there exist algorithms $(\mathsf{Setup}, \mathsf{Constrain}, \mathsf{Eval}, \mathsf{KeyDel})$ that satisfy the following:

- $\mathsf{Setup}(1^\lambda, 1^{\mathsf{in}(\lambda)})$ outputs a pair of keys $\mathsf{pk}, \mathsf{sk}$.
- $\mathsf{Constrain}(\mathsf{sk}, S)$ outputs a constrained key $K_S$ which enables evaluation of $F(\mathsf{sk}, \mathbf{x})$ on all $\mathbf{x} \in S$ and no other $\mathbf{x}$.
- $\mathsf{KeyDel}(K_S, S')$ outputs a constrained key $K_{S \cap S'}$ which enables the evaluation of $F(\mathsf{sk}, \mathbf{x})$ for all $\mathbf{x} \in S \cap S'$ and no other $\mathbf{x}$. We note that in systems where $\mathsf{KeyDel}$ is supported, the $\mathsf{Constrain}$ algorithm above can be expressed as a special case of $\mathsf{KeyDel}$ by letting $\mathsf{sk}$ correspond to the set of all inputs, i.e. $\mathsf{sk} = K_{\{0,1\}^{\mathsf{in}(\lambda)}}$.
- $\mathsf{Eval}(K_S, \mathbf{x})$ outputs $F(\mathsf{sk}, \mathbf{x})$ if $\mathbf{x} \in S$, $\perp$ otherwise.

Note that a set system is equivalent to a function family by defining set $S$ as the set of inputs where the function evaluates to 1. For our purposes, it will be more convenient to represent sets as functions.

Security. Constrained security is defined using the following two experiments denoted $\mathsf{EXP}(0)$ and $\mathsf{EXP}(1)$ with an adversary $\mathcal{A}$. For $b \in \{0,1\}$ experiment $\mathsf{EXP}(b)$ proceeds as follows:

First, a random key $k \in \{0,1\}^{\mathsf{seed}(\lambda)}$ is selected and two helper sets $C, V \subseteq \{0,1\}^{\mathsf{in}}$ are initialized to $\varnothing$. The set $V$ will keep track of all the points at which the adversary can evaluate. The set $C$ will keep track of the points where the adversary has been challenged. The sets $C$ and $V$ will ensure that the adversary cannot trivially decide whether challenge values are random or pseudorandom. In particular, the experiments maintain the invariant that $C \cap V = \varnothing$.

The adversary $\mathcal{A}$ is presented with three oracles as follows:

1. $F.\mathsf{Eval}$: Given $\mathbf{x} \in \{0,1\}^{\mathsf{in}}$, if $\mathbf{x} \notin C$, the oracle returns $F(\mathsf{sk}, \mathbf{x})$, else it returns $\perp$. The point $x$ is added to set $V$.
2. $F.\mathsf{Constrain}$: Given a set $S \in \mathcal{S}$ from $\mathcal{A}$, if $S \cap C = \varnothing$ the oracle returns $F.\mathsf{Constrain}(\mathsf{sk}, S)$, otherwise returns $\perp$. The set $V$ is updated to contain $S$.
3. $\mathsf{Challenge}$: Given $\mathbf{x}$ from $\mathcal{A}$, where $\mathbf{x} \notin V$, if $b = 0$, the adversary is given $F(\mathsf{sk}, \mathbf{x})$, else a random (consistent) element $y$. The set $C$ is updated to contain $\mathbf{x}$.

When the adversary is done interrogating the oracles, it outputs a bit $b'$. Let $W_b$ be the event that $b' = 1$ in $\mathsf{EXP}(b)$. The adversary's advantage is defined as $|\Pr[W_0] - \Pr[W_1]|$. We say that the PRF $F$ is a secure constrained PRF with respect to a set system $\mathcal{S}$ if all $\mathsf{PPT}$ adversaries $\mathcal{A}$ have negligible advantage in the above game.

# 3 Construction: Single Input FE for Turing Machines

In this section, we construct a single input functional encryption scheme for Turing machines, denoted by $\mathsf{TMFE}$ from the following ingredients:

1. Two compact functional encryption schemes for circuits, $\mathsf{1FE_1}$ and $\mathsf{1FE_2}$. We will assume that the scheme $\mathsf{1FE_2}$ is decomposable as defined in Section 2.
2. A symmetric encryption scheme $\mathsf{SKE} = (\mathsf{SKE.KeyGen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$.
3. A delegatable constrained pseudorandom function ($\mathsf{cPRF}$), denoted by $\mathsf{F}$ which supports $T$ delegations for the function family $f_t : \{0,1\}^{2\cdot\lambda} \to \{0,1\}$ defined as follows. Let $x, t$ denote integers whose binary representations are $\mathbf{x}, \mathbf{t}$ of $\lambda$ bits. Then,

$$f_t(\mathbf{x}\|\mathbf{z}) = 1, \text{if } \; x \geq t \text{ and } 0 \text{ otherwise}$$

Intuitively, the function is parametrized by a value $t$ and evaluates to 1 if the first half of its input, $x \geq t$. We will denote the constrained PRF key $\mathsf{K}_{f_t}$ corresponding to function $f_t$ by $\mathsf{K}_t$ for ease of notation. By the delegation property of constrained PRFs (Section 2.3), we have that if $t' \geq t$ then $\mathsf{K}_{t'}$ can be derived from $\mathsf{K}_t$. In our construction the parameter $t$ will represent the time step in the computation, which means that a PRF key of the current time step can be used to derive PRF keys for future time steps. We will denote a PRF for this functionality by $\mathsf{F}$. The security proof makes use a punctured version of the above $\mathsf{cPRF}$, please see Sections 3.3 and D for details.

## 3.1 Construction

Below we provide our construction for single input $\mathsf{FE}$ for Turing machines.

Notation. Note that since $\mathsf{1FE_2}$ is decomposable, there exists an encoding function $\mathcal{E}$ which encodes each bit of the input and since it is compact, the output length of $\mathcal{E}$ is independent of the circuit class supported by $\mathsf{1FE_2}$. Thus, by choosing the encoding function first, the $\mathsf{CktFE}$ scheme may support a circuit class that outputs its own ciphertext components. We denote by $\bar{\mathcal{E}}$ the encoding function $\mathcal{E}$ applied bitwise to a vector, i.e. $\bar{\mathcal{E}}(\mathbf{w}) = \mathcal{E}(w_1) \ldots \mathcal{E}(w_n)$.

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

1. Let $(\mathsf{1FE_2.PK}, \mathsf{1FE_2.MSK}) \leftarrow \mathsf{1FE_2.Setup}(1^\lambda)$, where $\mathsf{1FE_2}$ is a decomposable functional encryption scheme for the circuit family

   $$\mathsf{Next} : \Big( \Big( \{\mathsf{SYM}\} \times \{0,1\}^{4\lambda} \times \Sigma \times \mathsf{Trap} \Big) \times \Big( \{\mathsf{ST}\} \times \mathcal{Q} \Big) \Big) \to \Big( \mathcal{C}^{\mathsf{1FE_2}} \Big)^2 \cup \{\mathsf{ACC}, \mathsf{REJ}, \perp\}$$

   Here, $\Sigma$ and $\mathcal{Q}$ are the alphabet and state space respectively of the Turing machine family. The tokens $\mathsf{SYM}$ and $\mathsf{ST}$ are flags denoting a symbol and a state respectively. The set $\{0,1\}^{4\lambda}$ encodes in order, a random value $\mathsf{key\text{-}id}$ associated with a TM $M$, a cPRF key, the current time step in the computation and the length of the input string, each of $\lambda$ bits. Here, $\mathsf{Trap}$ is a data structure of fixed polynomial length which will be used in the proof. Since we do not need it in the construction, we do not discuss it here, please see Figure 6 for its definition. $\mathcal{C}^{\mathsf{1FE_2}}$ denotes the ciphertext space of $\mathsf{1FE_2}$, and $\mathsf{ACC}$ and $\mathsf{REJ}$ are bits indicating accepting and a rejecting states of a TM respectively.

2. Let $(\mathsf{1FE_1.PK}, \mathsf{1FE_1.MSK}) \leftarrow \mathsf{1FE_1.Setup}(1^\lambda)$, where $\mathsf{1FE_1}$ is a compact, public-key $\mathsf{CktFE}$ scheme for the circuit family

$$\mathsf{ReRand} : \Big(\{0,1\}^{3\lambda} \times \Sigma \times \mathsf{Trap}\Big) \to \mathcal{C}^{\mathsf{1FE_2}} \times \Big(\mathcal{C}^{\mathsf{1FE_2}} \cup \{\bot\}\Big)$$

Again, $\{0,1\}^{3\lambda}$ encodes in order, a root cPRF key, a time step and the length of the input string respectively, while $\Sigma$, $\mathsf{Trap}$ and $\mathcal{C}^{\mathsf{1FE_2}}$ are as described above.

3. Output $\mathsf{PK} = \mathsf{1FE_1.PK}$ and $\mathsf{MSK} = (\mathsf{1FE_1.MSK}, (\mathsf{1FE_2.PK}, \mathsf{1FE_2.MSK}))$.

$\mathsf{TMFE.Enc}(\mathsf{PK}, \mathbf{w})$: Upon input the public key $\mathsf{PK}$, and message $\mathbf{w}$ of arbitrary length $\ell = |\mathbf{w}|$, do the following:

1. Sample the root key $\mathsf{K}_0$ for function $f_t$ where $t = 0$ for the cPRF $\mathsf{F}$ described above.
2. For $i \in [\ell]$, let $\mathsf{CT}_i = \mathsf{1FE_1.Enc}(\mathsf{PK}, (\mathsf{K}_0, i, \ell, w_i, \mathsf{Trap}))$, where $\mathsf{Trap}$ is a data structure which is only relevant in the proof. Here, all fields of $\mathsf{Trap}$ are set to $\bot$ except a flag $\mathsf{Trap.mode\text{-}real} = 1$ which indicates that we are in the real world. Please see Figure 6 for the definition of $\mathsf{Trap}$.
3. Output $\mathsf{CT}_{\mathbf{w}} = \{\mathsf{CT}_i\}_{i \in [\ell]}$.

$\mathsf{TMFE.KeyGen}(\mathsf{MSK}, M)$: Upon input the master secret key $\mathsf{MSK}$ and the description of a Turing machine $M$, do the following. We will assume, w.l.o.g. that the TM is oblivious (see Appendix A for a justification) and $\mathsf{q_{st}} \in \mathcal{Q}$ is the start state of $M$.

1. Sample a random value $\mathsf{salt} \leftarrow \{0,1\}^\lambda$.
2. Interpret $\mathsf{MSK} = (\mathsf{1FE_1.MSK}, (\mathsf{1FE_2.PK}, \mathsf{1FE_2.MSK}))$.
3. Let $\mathsf{SK_{ReRand}} = \mathsf{1FE_1.KeyGen}(\mathsf{1FE_1.MSK}, \mathsf{ReRand}_{\mathsf{1FE_2.PK},\mathsf{salt},\mathsf{q_{st}},\bot,\bot})$ where Figure 2 defines the circuit $\mathsf{ReRand}_{\mathsf{1FE_2.PK},\mathsf{salt},\mathsf{q_{st}},\bot,\bot}$.
4. Let $\mathsf{SK_{Next}} = \mathsf{1FE_2.KeyGen}(\mathsf{1FE_2.MSK}, \mathsf{Next}_{\mathsf{1FE_2.PK},\mathsf{salt},M,\bot,\bot})$ where Figure 4 defines the circuit $\mathsf{Next}_{\mathsf{1FE_2.PK},\mathsf{salt},M,\bot,\bot}$.
5. Output $\mathsf{SK}_M = (\mathsf{SK_{ReRand}}, \mathsf{SK_{Next}})$.

$\mathsf{TMFE.Dec}(\mathsf{SK}_M, \mathsf{CT}_{\mathbf{w}})$: Upon input secret key $\mathsf{SK}_M$ and ciphertext $\mathsf{CT}_{\mathbf{w}}$, do the following:

1. Interpret $\mathsf{SK}_M = (\mathsf{SK_{ReRand}}, \mathsf{SK_{Next}})$ and $\mathsf{CT}_{\mathbf{w}} = \big(\mathsf{CT}_1, \ldots, \mathsf{CT}_{|\mathbf{w}|}\big)$.
2. For $i \in [|\mathbf{w}|]$, do the following:
   (a) If $i = 1$, invoke $\mathsf{1FE_1.Dec}(\mathsf{SK_{ReRand}}, \mathsf{CT}_1)$ to obtain $(\mathsf{CT}_{\mathsf{sym},1}, \mathsf{CT}_{\mathsf{st},1})$.
   (b) Else, invoke $\mathsf{1FE_1.Dec}(\mathsf{SK_{ReRand}}, \mathsf{CT}_i)$ to obtain $(\mathsf{CT}_{\mathsf{sym},i}, \bot)$.
3. Denote $\big((\mathsf{CT}_{\mathsf{sym},1}, \mathsf{CT}_{\mathsf{st},1}), \mathsf{CT}_{\mathsf{sym},2}, \ldots, \mathsf{CT}_{\mathsf{sym},|\mathbf{w}|}\big)$ as the new sequence of ciphertexts obtained under the $\mathsf{Next}$ scheme.
4. Let $t = 1$. While the Turing machine does not halt, do:
   (a) Invoke $\mathsf{1FE_2.Dec}\big(\mathsf{SK_{Next}}, (\mathsf{CT}_{\mathsf{sym},t}, \mathsf{CT}_{\mathsf{st},t})\big)$ to obtain:
      - $\mathsf{ACC}$ or $\mathsf{REJ}$. In this case, output "Accept" or "Reject" respectively, and exit the loop.
      - $\big(\mathsf{CT}_{\mathsf{sym},t'}, \mathsf{CT}_{\mathsf{st},t+1}\big)$.

      Note that $t'$ is the next time step that the work tape cell accessed at time step $t$ will be accessed again.
   (b) Let $t = t + 1$ and go to start of loop.

Function $\mathsf{ReRand}_{\mathsf{1FE_2.PK},\mathsf{salt},\mathsf{q_{st}},\mathcal{C}_1,\mathcal{C}_2}\big((\mathsf{K}_0, i, \ell, w_i, \mathsf{Trap})\big)$

(a) Initialization and Choosing Real or Trapdoor mode.

Initialize an input vector $\mathsf{inp} = (w_i, \mathsf{q_{st}})$. If $\mathsf{Trap.mode\text{-}real} = 1$, set $\mathsf{out} = (c_1, c_2)$, where $c_1 = c_2 = \perp$. If $i \neq 1$, set $\mathsf{inp} = (w_i, \perp)$. Else invoke $\mathsf{Trap\text{-}Mode}_{\mathsf{ReRand}}\big(\mathsf{Trap}, \mathsf{inp}, \mathsf{salt}, \ell, \mathcal{C}_1, \mathcal{C}_2, i\big)$ as described in Figure 3 to obtain $\big(\mathsf{inp}=(u_1, u_2), \mathsf{out}=(c_1, c_2)\big)$.

(b) Computing Encrypted Symbols using randomness derived from cPRF. If $\mathsf{out}.c_1 = \perp$, do the following.

i. Noting that $i > 0$, derive delegated cPRF key $\mathsf{K}_i$ from $\mathsf{K}_0$ as $\mathsf{K}_i = \mathsf{F.KeyDel}(\mathsf{K}_0, f_i)$. Compute randomness for encryption as $r_i = \mathsf{F.Eval}(\mathsf{K}_i, (i \| \mathsf{salt}))$.

ii. Derive delegated cPRF key $\mathsf{K}_{i+1} = \mathsf{F.KeyDel}(\mathsf{K}_i, f_{i+1})$. Set $\mathsf{key\text{-}id} = \mathsf{salt}$.

iii. Compute the $\mathsf{1FE_2}$ ciphertext component encoding $w_i = \mathsf{inp}.u_1$ for time step $i$ as

$$\mathsf{CT}_{\mathsf{sym},i} = \bar{\mathcal{E}}\big(\mathsf{1FE_2.PK}_1, (\mathsf{SYM}, \mathsf{key\text{-}id}, \mathsf{K}_{i+1}, i, \ell, w_i, \mathsf{Trap}); r_i\big)$$

iv. Set $\mathsf{out}.c_1 = \mathsf{CT}_{\mathsf{sym},i}$.

(c) Computing Encrypted State for First Time Step. If $\big((\mathsf{out}.c_2 = \perp) \wedge (i = 1)\big)$, do the following.

i. Compute $\mathsf{1FE_2}$ ciphertext component to encode the starting state $\mathsf{q_{st}} = \mathsf{inp}.u_2$ as

$$\mathsf{CT}_{\mathsf{st},1} = \bar{\mathcal{E}}\big(\mathsf{1FE_2.PK}_2, (\mathsf{ST}, \mathsf{q_{st}}); r_1\big)$$

ii. Set $\mathsf{out}.c_2 = \mathsf{CT}_{\mathsf{st},1}$.

(d) Output. If $i = 1$, output $\mathsf{out} = (\mathsf{CT}_{\mathsf{sym},1}, \mathsf{CT}_{\mathsf{st},1})$, else output $\mathsf{out} = (\mathsf{CT}_{\mathsf{sym},i}, \perp)$.

Figure 2: This circuit re-randomizes the ciphertexts provided during encryption to use randomness derived from a cPRF. The seed for the cPRF is specified in the ciphertext and the input is specified by the key. This ensures that each ciphertext, key pair form a unique "thread" of execution.

### 3.2 Correctness and Efficiency of single input TMFE

We now argue that the above scheme is correct. The TMFE.Dec algorithm takes as input a secret key $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ and a ciphertext $\mathsf{CT}_{\mathbf{w}} = \big(\mathsf{CT}_1, \ldots, \mathsf{CT}_{|\mathbf{w}|}\big)$ under the $\mathsf{1FE_1}$ scheme supporting the functionality $\mathsf{ReRand} := \mathsf{ReRand}_{\mathsf{1FE_2.PK},\mathsf{salt},\mathsf{q_{st}},\mathcal{C}_2,\mathcal{C}_2}$. Firstly, note that given a secret key $\mathsf{SK}_{\mathsf{ReRand}}$ along with a ciphertext $\mathsf{CT}_{\mathbf{w}}$, we have as follows.

1. Since $\mathsf{CT}_1$ encodes Trap with $\mathsf{Trap.mode\text{-}real} = 1$, hence by the correctness of the $\mathsf{1FE_1}$ scheme, we get $\mathsf{1FE_1.Dec}(\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{CT}_1) = (\mathsf{CT}_{\mathsf{sym},1}, \mathsf{CT}_{\mathsf{st},1})$ as output.

2. For $i \in [2, |\mathbf{w}|]$, since $\mathsf{CT}_i$ encodes Trap with $\mathsf{Trap.mode\text{-}real} = 1$, hence by the correctness of the $\mathsf{1FE_1}$ scheme, we get $\mathsf{1FE_1.Dec}(\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{CT}_i) = (\mathsf{CT}_{\mathsf{sym},i}, \perp)$ as the correct output.

The new sequence of $\mathsf{1FE_2}$ ciphertexts output by ReRand are now sequenced as $\big((\mathsf{CT}_{\mathsf{sym},1}, \mathsf{CT}_{\mathsf{st},1}), \mathsf{CT}_{\mathsf{sym},2}, \ldots, \mathsf{CT}_{\mathsf{sym},|\mathbf{w}|}\big)$. The $\mathsf{1FE_2}$ scheme supports the functionality $\mathsf{Next} := \mathsf{Next}_{\mathsf{1FE_2.PK},\mathsf{salt},M,\mathcal{C}_1,\mathcal{C}_2}$. Throughout the $\mathsf{1FE_2}$ decryption, we maintain the invariant that at any time step $t$, apart from a secret key $\mathsf{SK}_{\mathsf{Next}}$, the input to the $\mathsf{1FE_2.Dec}$ algorithm is an entire $\mathsf{1FE_2}$ ciphertext decomposed into two components corresponding to a symbol and a state ciphertext both of which are computed with the same randomness, which is computed as $\mathsf{F.Eval}(\mathsf{K}_0, (\mathsf{t} \| \mathsf{salt}))$[9].

[9]We do not explicitly construct ciphertext components corresponding to blank tape cells in the Next functionality for ease of exposition; we assume w.l.o.g that any non-input cell that is accessed by the OTM has been written to by the Next functionality in a previous step, thus generating the requisite symbol ciphertext.

Subroutine $\mathsf{Trap\text{-}Mode}_{\mathsf{ReRand}}\big(\mathsf{Trap}, \mathsf{inp}, \mathsf{salt}, \ell, \mathcal{C}_1, \mathcal{C}_2, i\big)$

Interpret $\mathsf{inp} = (u_1, u_2) = (w_i, \mathsf{q}_{\mathsf{st}})$ and initialize $\mathsf{out} = (c_1, c_2)$, where $c_1 = c_2 = \perp$.

If $\mathsf{Trap.key\text{-}id} = \mathsf{salt}$, do the following.

(a) If $\mathsf{Trap.mode\text{-}trap}_3 = 1$, do the following:

i. If $\big((\mathsf{Trap.Sym\ TS} = i) \wedge (i \leq \ell)\big)$, compute the $\mathsf{1FE}_2$ ciphertext $\mathsf{CT}_{\mathsf{sym},i} = \mathsf{SKE.Dec}(\mathsf{Trap.SKE.K}, \mathcal{C}_1)$ and set $\mathsf{out}.c_1 = \mathsf{CT}_{\mathsf{sym},i}$.

ii. If $\big((\mathsf{Trap.ST\ TS} = i) \wedge (i = 1)\big)$, compute the $\mathsf{1FE}_2$ ciphertext $\mathsf{CT}_{\mathsf{st},i} = \mathsf{SKE.Dec}(\mathsf{Trap.SKE.K}, \mathcal{C}_2)$ and set $\mathsf{out}.c_2 = \mathsf{CT}_{\mathsf{st},1}$.

(b) If $\mathsf{Trap.mode\text{-}trap}_1 = 1$, do the following:

i. If $\big((\mathsf{Trap.Sym\ TS}_1 = i) \wedge (i \leq \ell)\big)$, set $\mathsf{inp}.u_1 = \mathsf{Trap.Sym\ val}_1$ with the symbol to be encrypted and output at time step $i$.

ii. If $\big((\mathsf{Trap.ST\ TS}_1 = i) \wedge (i = 1)\big)$, set $\mathsf{inp}.u_2 = \mathsf{Trap.ST\ val}_1$ with the start state to be encrypted and output at time step 1.

(c) If $\mathsf{Trap.mode\text{-}trap}_2 = 1$, do the following:

i. If $\big((\mathsf{Trap.Sym\ TS}_2 = i) \wedge (i \leq \ell)\big)$, set $\mathsf{inp}.u_1 = \mathsf{Trap.Sym\ val}_2$ with the symbol to be encrypted and output at time step $i$.

ii. If $\big((\mathsf{Trap.ST\ TS}_2 = i) \wedge (i = 1)\big)$, set $\mathsf{inp}.u_2 = \mathsf{Trap.ST\ val}_2$ with the start state to be encrypted and output at time step 1.

If $\mathsf{Trap.key\text{-}id} \neq \mathsf{salt}$, do the following.

(a) If $\mathsf{salt} > \mathsf{Trap.key\text{-}id}$ set $b = 0$; else set $b = 1$[a].

(b) If $i \neq 1$, update $\mathsf{inp} = (\mathsf{Trap.val}_b, \perp)$; else update $\mathsf{inp} = (\mathsf{Trap.val}_b, \mathsf{q}_{\mathsf{st}})$.

Output. Return $(\mathsf{inp}, \mathsf{out})$.

[a] We assume a lexicographic ordering on the salt values and a generalized comparison operator.

Figure 3: Subroutine handling the trapdoor modes in ReRand. This is "active" only in the proof.

We show that given a secret key $\mathsf{SK}_{\mathsf{Next}}$ and the sequence of ciphertexts $\big((\mathsf{CT}_{\mathsf{sym},1}, \mathsf{CT}_{\mathsf{st},1}), \mathsf{CT}_{\mathsf{sym},2}, \ldots, \mathsf{CT}_{\mathsf{sym},|\mathbf{w}|}\big)$ generated from the outputs of the $\mathsf{1FE}_1.\mathsf{Dec}$ algorithm, $\mathsf{1FE}_2.\mathsf{Dec}$ correctly computes the decomposed ciphertext components of a symbol and a state that occur along the computation path and finally outputs the value of machine $M$ on the sequenced input. Define $\tau = \mathsf{runtime}(M, \mathbf{w})$. Formally, by the correctness of $\mathsf{1FE}_2$ scheme, at any time step $t \in [\tau - 2]$, $\mathsf{1FE}_2.\mathsf{Dec}(\mathsf{SK}_{\mathsf{Next}}, (\mathsf{CT}_{\mathsf{sym},t}, \mathsf{CT}_{\mathsf{st},t}))$ correctly outputs either $(\mathsf{CT}_{\mathsf{sym},t'}, \mathsf{CT}_{\mathsf{st},t+1})$ with $t < t' \leq \tau - 1$. Further, for any time step $t \in [\tau - 2]$, we have:

1. Let $t \in [\tau - 2] \setminus [\ell]$. If the current work tape cell was accessed[10], at some time step $\tilde{t} < t$, then $\mathsf{CT}_{\mathsf{sym},t}$ encoding $(\mathsf{SYM}, \mathsf{key\text{-}id}, \mathsf{K}_{t+1}, t, \ell, \sigma_t, \mathsf{Trap})$ was constructed at time step $\tilde{t}$. Note that $\sigma_t$ may be the blank symbol $\beta$. When $t \in [\ell]$, $\mathsf{CT}_{\mathsf{sym},t}$ is constructed at time step $t$ via the ReRand circuit.

[10] We assume that every time a cell is accessed, it is written to, by writing the same symbol again if no change is made.

Function $\mathsf{Next}_{1\mathsf{FE}_2.\mathsf{PK},\mathsf{salt},M,\mathcal{C}_1,\mathcal{C}_2}\big((\mathbf{z}_1,\mathbf{z}_2)\big)$

(a) Reading Current (Symbol, State) Pair and Looking up Transition Table.

i. Interpret $\mathbf{z}_1 = (\mathsf{type}, \mathsf{key\text{-}id}, \mathsf{K}_{t+1}, t, \ell, s, \mathsf{Trap})$, $\mathbf{z}_2 = (\mathsf{type}, s)$. If $((\mathbf{z}_1.\mathsf{type} \neq \mathsf{SYM}) \vee (\mathbf{z}_2.\mathsf{type} \neq \mathsf{ST}) \vee (\mathbf{z}_1.\mathsf{key\text{-}id} \neq \mathsf{salt}))$, output $\perp$ and abort.

ii. Interpret $(\mathbf{z}_1.s, \mathbf{z}_2.s) = (\sigma_t, \mathsf{q}_t)$ as the symbol, state pair for the current time step $t = \mathbf{z}_1.t$, input $\mathsf{K}_{t+1} = \mathbf{z}_1.\mathsf{K}_{t+1}$ as the constrained PRF key for future time steps. Denote $\mathsf{key\text{-}id} = \mathbf{z}_1.\mathsf{key\text{-}id}$, $\ell = \mathbf{z}_1.\ell$ and $\mathsf{Trap} = \mathbf{z}_1.\mathsf{Trap}$. Using the transition table of the machine $M$, look up the next state $\mathsf{q}_{t+1}$ as well as the symbol $\sigma_{t'}$ to be written on the work-tape, where $t'$ is the time step the current work tape cell will next be read by $M$. If $\mathsf{q}_{t+1}$ is an accept or reject state, then output $\mathsf{ACC}$ or $\mathsf{REJ}$ and exit.

iii. Initialize $\mathsf{inp} = (\sigma_{t'}, \mathsf{q}_{t+1})$.

(b) Choosing Real or Trapdoor mode. If $\mathsf{Trap.mode\text{-}real} = 1$, initialize an output vector $\mathsf{out} = (c_1, c_2)$, where $c_1 = c_2 = \perp$. Else invoke $\mathsf{Trap\text{-}Mode}_{\mathsf{Next}}\big(\mathsf{Trap}, \mathsf{inp}, \mathsf{salt}, \ell, \mathcal{C}_1, \mathcal{C}_2, t, t'\big)$ as described in Figure 5 to obtain $\big(\mathsf{inp}=(u_1,u_2), \mathsf{out}=(c_1,c_2)\big)$.

(c) Computing Next Encrypted Symbol. If $\mathsf{out}.c_1 = \perp$, do the following.

i. Noting that $t' > t$, derive the randomness at time step $t'$ using the delegated key $\mathsf{K}_{t+1}$ as $r_{t'} = \mathsf{F.Eval}(\mathsf{K}_{t+1}, (t' \| \mathsf{salt}))$. Compute the delegated PRF key $\mathsf{K}_{t'+1} = \mathsf{F.KeyDel}(\mathsf{K}_{t+1}, f_{t'+1})$.

ii. Compute the $1\mathsf{FE}_2$ ciphertext component encoding the symbol $\sigma_{t'} = \mathsf{inp}.u_1$ for time step $t'$ as

$$\mathsf{CT}_{\mathsf{sym},t'} = \bar{\mathcal{E}}\big(1\mathsf{FE}_2.\mathsf{PK}_1, (\mathsf{SYM}, \mathsf{key\text{-}id}, \mathsf{K}_{t'+1}, t', \ell, \sigma_{t'}, \mathsf{Trap}); r_{t'}\big)$$

iii. Set $\mathsf{out}.c_1 = \mathsf{CT}_{\mathsf{sym},t'}$.

(d) Computing Next Encrypted State. If $\mathsf{out}.c_2 = \perp$, do the following.

i. Derive the randomness at time step $t+1$ as $r_{t+1} = \mathsf{F.Eval}(\mathsf{K}_{t+1}, (t+1 \| \mathsf{salt}))$ and compute the $1\mathsf{FE}_2$ ciphertext component encoding the state $\mathsf{q}_{t+1} = \mathsf{inp}.u_2$ for time step $t+1$ as

$$\mathsf{CT}_{\mathsf{st},t+1} = \bar{\mathcal{E}}\big(1\mathsf{FE}_2.\mathsf{PK}_2, (\mathsf{ST}, \mathsf{q}_{t+1}); r_{t+1}\big)$$

ii. Set $\mathsf{out}.c_2 = \mathsf{CT}_{\mathsf{st},t+1}$.

(e) Output : $\mathsf{out} = \big(\mathsf{CT}_{\mathsf{sym},t'}, \mathsf{CT}_{\mathsf{st},t+1}\big)$

Figure 4: Function to mimic TM computation. It reads the current symbol, state pair and outputs an encryption of the new state and symbol to be written under the appropriate randomness generated using a $\mathsf{cPRF}$.

2. The ciphertext component $\mathsf{CT}_{\mathsf{st},t}$ encoding $(\mathsf{ST}, \mathsf{q}_t)$ at time step $t$ was constructed at time step $t-1$ for $t > 1$ and at time step 1, when $t = 1$.

3. The randomness $r_t = \mathsf{F.Eval}(\mathsf{K}_{\tilde{t}+1}, (\mathsf{t} \| \mathsf{salt})) = \mathsf{F.Eval}(\mathsf{K}_{\mathsf{t}}, (\mathsf{t} \| \mathsf{salt}))$ binds the components $\mathsf{CT}_{\mathsf{sym},t}$ and $\mathsf{CT}_{\mathsf{st},t}$.

Thus, at any given time step $t \in [\tau - 2]$, we have a complete ciphertext of $1\mathsf{FE}_2$ which may be fed again with $\mathsf{SK}_{\mathsf{Next}}$ to $1\mathsf{FE}_2.\mathsf{Dec}$ in order to proceed with the computation. Thus, the execution of $1\mathsf{FE}_2.\mathsf{Dec}$ at the $(\tau-2)^{\text{th}}$ time step provides the complete pair $(\mathsf{CT}_{\mathsf{sym},\tau-1}, \mathsf{CT}_{\mathsf{st},\tau-1})$. By the correctness of $1\mathsf{FE}_2$ scheme again, at time step $t = \tau - 1$, invoking $1\mathsf{FE}_2.\mathsf{Dec}(\mathsf{SK}_{\mathsf{Next}}, (\mathsf{CT}_{\mathsf{sym},\tau-1}, \mathsf{CT}_{\mathsf{st},\tau-1}))$ outputs either "Accept" or "Reject" by simulating the execution of $M$ for the final time step $\tau$ inside the function $\mathsf{Next}$, thus correctly outputting $M(\mathbf{w})$.

Subroutine $\mathsf{Trap\text{-}Mode}_{\mathsf{Next}}(\mathsf{Trap}, \mathsf{inp}, \mathsf{salt}, \ell, \mathcal{C}_1, \mathcal{C}_2, t, t')$

Interpret the input vector $\mathsf{inp} = (u_1, u_2) = (\sigma_{t'}, \mathsf{q}_{t+1})$ and initialize the output vector $\mathsf{out} = (c_2, c_2)$, where $c_1 = c_2 = \bot$.

(a) If $\big((\mathsf{Trap.key\text{-}id} = \mathsf{salt}) \wedge (\mathsf{Trap.mode\text{-}trap}_3 = 1)\big)$, do the following.

i. If $\big((\mathsf{Trap.Sym\ TS} = t) \wedge (\mathsf{Trap.Target\ TS} = t') \wedge (t > \ell)\big)$, compute the $\mathsf{1FE}_2$ symbol ciphertext $\mathsf{CT}_{\mathsf{sym},t'} = \mathsf{SKE.Dec}(\mathsf{Trap.SKE.K}, \mathcal{C}_1)$ and set $\mathsf{out}.c_1 = \mathsf{CT}_{\mathsf{sym},t'}$.

ii. If $\big((\mathsf{Trap.ST\ TS} = t) \wedge (\mathsf{Trap.Target\ TS} = t+1) \wedge (t > 1)\big)$, compute the $\mathsf{1FE}_2$ state ciphertext $\mathsf{CT}_{\mathsf{st},t+1} = \mathsf{SKE.Dec}(\mathsf{Trap.SKE.K}, \mathcal{C}_2)$ and set $\mathsf{out}.c_2 = \mathsf{CT}_{\mathsf{st},t+1}$.

(b) If $\big((\mathsf{Trap.key\text{-}id} = \mathsf{salt}) \wedge (\mathsf{Trap.mode\text{-}trap}_1 = 1)\big)$, do the following:

i. If $\big((\mathsf{Trap.Sym\ TS}_1 = t) \wedge (\mathsf{Trap.Target\ TS}_1 = t') \wedge (t > \ell)\big)$, set $\mathsf{inp}.u_1 = \mathsf{Trap.Sym\ val}_1$ with the symbol $\sigma_{t'} = \mathsf{Trap.Sym\ val}_1$ to be encrypted and given as output for time step $t'$.

ii. If $\big((\mathsf{Trap.ST\ TS}_1 = t) \wedge (\mathsf{Trap.Target\ TS}_1 = t+1) \wedge (t > 1)\big)$, set $\mathsf{inp}.u_2 = \mathsf{Trap.ST\ val}_1$ with the state $\mathsf{q}_{t+1} = \mathsf{Trap.ST\ val}_1$ to be encrypted and given as output for time step $t+1$.

(c) If $\big((\mathsf{Trap.key\text{-}id} = \mathsf{salt}) \wedge (\mathsf{Trap.mode\text{-}trap}_2 = 1)\big)$, do the following:

i. If $\big((\mathsf{Trap.Sym\ TS}_2 = t) \wedge (\mathsf{Trap.Target\ TS}_2 = t') \wedge (t > \ell)\big)$, set $\mathsf{inp}.u_1 = \mathsf{Trap.Sym\ val}_2$ with the symbol $\sigma_{t'} = \mathsf{Trap.Sym\ val}_2$ to be encrypted and given as output for time step $t'$.

ii. If $\big((\mathsf{Trap.ST\ TS}_2 = t) \wedge (\mathsf{Trap.Target\ TS}_2 = t+1) \wedge (t > 1)\big)$, set $\mathsf{inp}.u_2 = \mathsf{Trap.ST\ val}_2$ with the state $\mathsf{q}_{t+1} = \mathsf{Trap.ST\ val}_2$ to be encrypted and given as output for time step $t+1$.

(d) Exit the subroutine returning $(\mathsf{inp}, \mathsf{out})$.

Figure 5: Subroutine handling the trapdoor modes in Next. This is "active" only in the proof.

Efficiency. The TMFE construction described above inherits its efficiency from the underlying CktFE constructions. Note that the ciphertext is compact and is of size $\mathrm{poly}(\lambda, |\mathbf{w}|)$. Also, the running time of the decryption procedure is input specific since it mimics the computation of $M$ on $\mathbf{w}$ using secret key encoding $M$ and ciphertext encoding all the intermediate states of the computation. Additionally, the public parameters are short $\mathrm{poly}(\lambda)$, since these are just the public parameters of a compact CktFE scheme. The function keys are also short, since they are CktFE function keys for circuits ReRand and Next which are of size $\mathrm{poly}(\lambda)$ and $\mathrm{poly}(|M|, \lambda)$ respectively.

### 3.3 Proof of Security for Single Input TMFE

Next, we prove that the above TMFE scheme satisfies distributional indistinguishability (DI) for single (or constant) length outputs, as long as the underlying CktFE scheme satisfies distributional indistinguishability for any output length. In Appendix E, we provide an instantiation of a CktFE scheme satisfying distributional indistinguishability.

Theorem 3.1. Assume that the functional encryption schemes for circuits $\mathsf{1FE}_1$ and $\mathsf{1FE}_2$ are DI secure (according to definition 2.3) and that F is a secure cPRF for the function family defined above (according to definition 2.3). Then, the construction of functional encryption for Turing machines TMFE is selective DI secure for single bit outputs (according to definition 2.8).

Since the intuition was discussed in Section 1, we proceed to the formal proof.

The Trapdoor Data Structure. To implement the approach discussed in Section 1, we will make use of a data-structure Trap that lets us store all the requisite trapdoor information needed for the security proof within the ciphertext. In our construction, decryption of a particular input by a particular function key results in a chain of ciphertexts, each of which contain the trapdoor data structure. In the real world, this information is not used but as we progress through the proof, different fields become relevant. The data structure is outlined in Figure 6.

| mode-real | key-id | $\mathsf{val}_0$ | $\mathsf{val}_1$ | SKE.K | $\perp$ |
|---|---|---|---|---|---|
| $\mathsf{mode\text{-}trap}_1$ | Target $\mathsf{TS}_1$ | Sym $\mathsf{TS}_1$ | Sym $\mathsf{val}_1$ | ST $\mathsf{TS}_1$ | ST $\mathsf{val}_1$ |
| $\mathsf{mode\text{-}trap}_2$ | Target $\mathsf{TS}_2$ | Sym $\mathsf{TS}_2$ | Sym $\mathsf{val}_2$ | ST $\mathsf{TS}_2$ | ST $\mathsf{val}_2$ |
| $\mathsf{mode\text{-}trap}_3$ | Target TS | Sym TS | $\perp$ | ST TS | $\perp$ |

Figure 6: Data Structure Trap used for Proof

Row 1. Above, key-id refers to the particular function key being considered and we switch the execution chain from $b = 0$ to $b = 1$ key by key. All the ciphertexts in a given execution chain share the key-id value. We assume a lexicographic order on the key-id fields, this can be easily ensured by having a counter as part of the key-id field. We do not make this explicit below for notational brevity. If $\mathsf{key\text{-}id}^*$ is the key identity programmed in a particular execution chain, then all keys with values smaller than $\mathsf{key\text{-}id}^*$ will decrypt the chain using the input bit $b = 1$, and all keys with values larger than $\mathsf{key\text{-}id}^*$ will use $b = 0$. Hence, the $\mathsf{1FE}_1$ ciphertexts provided by the encryptor must encode messages corresponding to both values of $b$, the fields $\mathsf{val}_0$ and $\mathsf{val}_1$ are designed for this purpose[11]. Note that $\mathsf{1FE}_2$ ciphertexts computed by decryption need not track messages corresponding to both values of $b$, since the "chain is extended" via decryption corresponding to exactly one of $b = 0$ or $b = 1$ depending on the relation between the key identities in the ciphertext and the function key. The field SKE.K refers to the key of a symmetric key encryption scheme, which is used to decrypt some encrypted value embedded in the function key. This is a standard trick when the key must hide something in the public key setting. The flag mode-real means the scheme operates in the real world mode and the trapdoor information is not used.

Rows 2 and 3. The fields Target $\mathsf{TS}_1$ and Target $\mathsf{TS}_2$ refer to the time steps corresponding to the "broken link" in the decryption chain, namely the two time steps for which the ciphertext and function key are being programmed so as to switch from $b = 0$ to $b = 1$. The fields Sym $\mathsf{TS}_1$ and ST $\mathsf{TS}_1$ are the time steps when the symbol and state ciphertexts for time step Target $\mathsf{TS}_1$ are generated; for instance ST $\mathsf{TS}_1$ = Target $\mathsf{TS}_1 - 1$ since the state ciphertext for a given time step is always generated in the previous time step, while the symbol ciphertext for a given time step may be generated much earlier. Sym $\mathsf{TS}_2$ and ST $\mathsf{TS}_2$ are defined analogously. The fields Sym $\mathsf{val}_1$ and ST $\mathsf{val}_1$ contain the symbol and state values which will be encrypted in the hybrid at the time steps Sym $\mathsf{TS}_1$ and ST $\mathsf{TS}_1$ when $\mathsf{mode\text{-}trap}_1$ is set; Sym $\mathsf{val}_2$ and ST $\mathsf{val}_2$ are defined analogously.

Row 4. When $\mathsf{mode\text{-}trap}_3$ is set, the symbol and state values are set to $\perp$, and the values hard coded in the function key are used for the target time step. In more detail, the function key contains SKE encryptions of symbol and state ciphertexts corresponding to time step Target TS hard-coded within itself. If $\mathsf{key\text{-}id}^* = \mathsf{key\text{-}id}$, where $\mathsf{key\text{-}id}^*$ is the key identity programmed in a particular execution chain and key-id is the key identity of the function

[11] For the knowledgeable reader, this is similar to what was done by [AJ15].

key in question, and $\mathsf{mode\text{-}trap}_3 = 1$, then at time steps $\mathsf{SYM\ TS}$ and $\mathsf{ST\ TS}$ the $\mathsf{SKE}$ secret key in row 1 of the $\mathsf{Trap}$ data structure is used to decrypt the $\mathsf{SKE}$ encryptions and output the encrypted values.

The Hybrids. We now proceed to describe our hybrids. For simplicity we first describe the hybrids for a single function request, for some Turing machine $M$. We denote by $T$ the time taken by $M$ to run on the challenge messages. Since the proof is very involved, we describe it first for the weak selective game, where the adversary specifies the challenge vectors and machine at the same time. We discuss how to remove this restriction to obtain selective security at the end of the detailed proof.

$\mathcal{H}(0)$: This is the real world, when $\mathsf{mode\text{-}real} = 1$ and $\mathsf{mode\text{-}trap}_1 = \mathsf{mode\text{-}trap}_2 = \mathsf{mode\text{-}trap}_3 = \perp$.

$\mathcal{H}(1,1)$: In this world, all ciphertexts (constructed by the encryptor as well as function keys) have $\mathsf{mode\text{-}real} = \perp$, $\mathsf{mode\text{-}trap}_1 = 1$, $\mathsf{mode\text{-}trap}_2 = 1$, $\mathsf{mode\text{-}trap}_3 = \perp$. We program the last link in the decryption chain for switching bit $b$ by setting:

$$\mathsf{Target\ TS}_1 = T - 1, \mathsf{Target\ TS}_2 = T - 2$$

The fields $\mathsf{Sym\ TS}_1$ and $\mathsf{ST\ TS}_1$ contain the time steps when the symbol and state ciphertext pieces are generated for time step $T-1$, and the fields $\mathsf{Sym\ val}_1$ and $\mathsf{ST\ val}_1$ contain the symbol and state values which must be encrypted by the function key in the above time steps when $\mathsf{mode\text{-}trap}_1$ is set. Note that these fields exactly mimic the behaviour in the real world, namely the time steps and values are set to be exactly what the real world decryption would output. The fields corresponding to $\mathsf{TS}_2$ are defined analogously.

Indistinguishability follows from security of $\mathsf{1FE}_1$, since the decryption values in both hybrids are exactly the same.

$\mathcal{H}(1,2)$: Hardwire the key with an $\mathsf{SKE}$ encryption of symbol and state ciphertexts output at step $T-1$ for $b = 0$. Use the same ciphertexts as would be generated in the previous hybrid.

Indistinguishability follows from security of $\mathsf{SKE}$, since the only difference is the value of the message encrypted using $\mathsf{SKE}$ which is embedded in the key.

$\mathcal{H}(1,3)$: Set $\mathsf{mode\text{-}trap}_1 = \perp$, $\mathsf{mode\text{-}trap}_2 = 1$, $\mathsf{mode\text{-}trap}_3 = 1$ and $\mathsf{Target\ TS} = T - 1$. In this hybrid the hardwired value in the key is used to be output as step $T-1$ ciphertext.

Indistinguishability follows from security of $\mathsf{1FE}_1$, since the decryption values in both hybrids are exactly the same.

$\mathcal{H}(1,4)$: Change normal root key $\mathsf{K}_0$ to punctured root key $\mathsf{K}_0^{T-1}$ which punctures all delegated keys at point $(T-1\|\mathsf{key\text{-}id})$.

Indistinguishability follows from security of $\mathsf{1FE}_1$. Note that we evaluate the $\mathsf{cPRF}$ at point $(T-1\|\mathsf{key\text{-}id})$ only to construct the $\mathsf{1FE}_2$ ciphertext output at time step $T-1$ identified with $\mathsf{key\text{-}id}$. This ciphertext is currently hardwired in the function key, and is computed exactly the same way in both hybrids. Thus, the $\mathsf{cPRF}$ key is only required to compute randomness of points $\neq (T-1\|\mathsf{key\text{-}id})$, for which the punctured key suffices, and which moreover evaluates to the same value as the normal key on all such points. Hence, we have that the decryption values in both hybrids are exactly the same. Note that the punctured key is not used to evaluate on the punctured points.

$\mathcal{H}(1,5)$: Switch the randomness in the $\mathsf{1FE_2}$ ciphertexts for time step $T-1$ which are hardwired in the key to true randomness.

Indistinguishability follows from security of punctured $\mathsf{cPRF}$ for the aforementioned function family, since the remainder of the distribution only uses the punctured key.

$\mathcal{H}(1,6)$: Switch the value encoded in the $\mathsf{1FE_2}$ ciphertexts for time step $T-1$ which are hardwired in the key to correspond to $b=1$.

Indistinguishability follows from security of $\mathsf{1FE_2}$. Formally, we do a reduction which plays the security game against the $\mathsf{1FE_2}$ challenger and simulates the $\mathsf{TMFE}$ adversary. The reduction simulates $\mathsf{1FE_1}$ itself and receives the $\mathsf{1FE_2}$ public and function keys from the challenger. The only difference between the two hybrids is the $\mathsf{1FE_2}$ ciphertext for time step $T-1$ which is embedded in the function key as received from the $\mathsf{1FE_2}$ challenger.

$\mathcal{H}(1,7)$: Switch randomness back to PRF randomness in the ciphertext hardwired in key, using the punctured key for all but the hardwired ciphertext.

Indistinguishability follows from security of $\mathsf{cPRF}$ as discussed above.

$\mathcal{H}(1,8)$: Switch the punctured root key to the normal root key.

Indistinguishability follows from security of $\mathsf{1FE_1}$ as discussed above.

$\mathcal{H}(2,1)$: Switch ciphertext in slot 1 for target $T-1$ to be for $b=1$. Slot 2 remains $b=0$. Set $\mathsf{mode\text{-}trap}_3=\perp$ and $\mathsf{mode\text{-}trap}_1=\mathsf{mode\text{-}trap}_2=1$.

Indistinguishability follows from security of $\mathsf{1FE_1}$, since the decryption values in both hybrids are exactly the same.

$\mathcal{H}(2,2)$: Hardwire key with $\mathsf{SKE}$ encryption of $\mathsf{1FE_2}$ ciphertext for time step $T-2$ and bit $b=0$ (same as hybrid $(1,2)$ but for $T-2$).

Indistinguishability follows from security of $\mathsf{SKE}$ as above.

$\mathcal{H}(2,3)$: Set $\mathsf{mode\text{-}trap}_1=1$ with target $T-1$, $\mathsf{mode\text{-}trap}_2=\perp$, and $\mathsf{mode\text{-}trap}_3=1$ with target $T-2$.

Indistinguishability follows from security of $\mathsf{1FE_1}$, since the decryption values in both hybrids are exactly the same.

$\mathcal{H}(2,4)$: Switch normal root key to punctured key at point $(T-2\|\mathsf{key\text{-}id})$.

Indistinguishability follows from security of $\mathsf{1FE_1}$ as discussed above.

$\mathcal{H}(2,5)$: Switch randomness to true in the ciphertext hardwired in key.

Indistinguishability follows from security of $\mathsf{cPRF}$ as discussed above.

$\mathcal{H}(2,6)$: Switch hardwired $\mathsf{1FE_2}$ ciphertext for step $T-2$ to correspond to bit $b=1$.

Indistinguishability follows from security of $\mathsf{1FE_2}$.

$\mathcal{H}(2,7)$: Switch randomness back to use the $\mathsf{PRF}$ in the ciphertext hardwired in key.

Indistinguishability follows from security of $\mathsf{cPRF}$ as discussed above.

$\mathcal{H}(2,8)$: Switch punctured root key to normal root key.

Indistinguishability follows from security of $\mathsf{1FE_1}$ as discussed above.

$\mathcal{H}(3,1)$: Intuitively, we slide the trapdoor left by one step, i.e. change target time-steps to $T-2$ and $T-3$ in the ciphertext. Now slot 1 for $T-2$ corresponds to $b=1$ and slot 2 for $T-3$ to $b=0$. Set $\mathsf{mode\text{-}real} = \mathsf{mode\text{-}trap}_3 = \perp$ and $\mathsf{mode\text{-}trap}_1 = \mathsf{mode\text{-}trap}_2 = 1$.

Indistinguishability follows from security of $\mathsf{1FE}_1$, since the decryption values in both hybrids are exactly the same. Note that now slot $T-1$ is redundant, since $T-2$ ciphertext is already switched to $b=1$.

Hybrid $\mathcal{H}(3,i)$ will be analogous to $\mathcal{H}(2,i)$ for $i \in [8]$.

As we proceed left in the execution chain one step at a time, we reach step $\ell$ where $\ell = |\mathbf{w}|$, i.e. time steps for which $\mathsf{1FE}_1$ ciphertexts are provided by the encryptor. At this point we will hardwire the $\mathsf{ReRand}$ key with symbol ciphertexts for $\ell$ time steps, one at a time, and the $\mathsf{Next}$ key for the state ciphertexts[12]. Moreover, we must now add an additional hybrid in which the challenge $\mathsf{1FE}_1$ ciphertext at position $\ell$ contains the message bit corresponding to $b=1$; intuitively, we must switch the bit before we slide the trapdoor since the ciphertext for this position is not generated by decrypting the previous ciphertext. In more detail, in $\mathcal{H}(T-\ell,8)$, analogously to hybrid $(1,8)$, the $T-(T-\ell)=\ell^{th}$ bit hard-wired in the trapdoor is changed to 1. We now add one more hybrid, namely:

$\mathcal{H}(T-\ell,9)$ : In this hybrid, we modify the $\mathsf{1FE}_1$ challenge ciphertext in position $\ell$ as follows: the encoded message is changed corresponding to $b=1$ and flag $\mathsf{mode\text{-}real}=1$. The other flags $\mathsf{mode\text{-}trap}_1 = \mathsf{mode\text{-}trap}_2 = \mathsf{mode\text{-}trap}_3 = \perp$.

Note that all ciphertexts previous to time step $\ell$ remain unchanged, and output their corresponding symbol ciphertexts correctly. The $\mathsf{Next}$ circuit outputs the state ciphertext for time step $\ell$ corresponding to bit $b=1$. The only difference between this hybrid and the previous one is that here we use the real mode to output the symbol ciphertext for $b=1$ whereas previously we used the trapdoor mode to output the same symbol ciphertext. Hence, decryption values in both hybrids are exactly the same, and indistinguishability follows from security of $\mathsf{1FE}_1$.

Finally in $\mathcal{H}(T-1,9)$, the entire chain has been replaced to use $b=1$ and all the challenge $\mathsf{1FE}_1$ ciphertexts have encoded messages corresponding to $b=1$ with $\mathsf{mode\text{-}real}=1$.

$\mathcal{H}(T)$: In this hybrid, all the other fields in the trapdoor data structure, excepting $\mathsf{mode\text{-}real}$ are disabled and set to $\perp$. This is the real world with $b=1$.

Since all the encoded messages use $b=1$, decryption values are all exactly the same as in $\mathcal{H}(T-1,9)$, hence indistinguishability follows from security of $\mathsf{1FE}_1$.

The formal reductions are provided in Appendix B.

Multiple Keys. We handle multiple keys by repeating the above set of hybrids key by key. Each key carries within it an identifier $\mathsf{key\text{-}id}$, and if this is less than the key identifier encoded in the ciphertext, the bit $b=1$ is used, if it is greater then the bit $b=0$ is used and if it is equal, then the above sequence of hybrids is performed to switch from $b=0$ to $b=1$. To support this, the $\mathsf{1FE}_1$ ciphertexts provided by the encryptor must encode messages corresponding to both values of $b$, the fields $\mathsf{val}_0$ and $\mathsf{val}_1$ in the trapdoor data structure of Figure 6 are provided for this purpose. Security follows by a standard hybrid argument as in [AJ15].

[12]There is an exception at time step 1 when both the symbol ciphertext and the start state ciphertexts are hardwired in the $\mathsf{ReRand}$ key

### 3.4 Constructing the cPRF.

In Appendix D, we provide a construction for a cPRF $\mathsf{F}$ which supports puncturing and delegation as required; the $T$ cPRFs $\mathsf{F}_i$ for $i \in [T]$ may each be constructed similarly. To begin, note that we require the root key of $\mathsf{F}$ to be punctured at a point $i^*$ (say). The cPRF construction for punctured PRF [BW13, KPTZ13, BGI14](which is in turn inherited from the standard PRG based GGM [GGM86]) immediately satisfies this constraint, so we are left with the question of delegation.

Recall that we are required to delegate $T$ times, where $T$ is the (polynomial) runtime of the Turing machine on the encrypted input (please see Section 3), and the $j^{th}$ delegated key must support evaluation of points $\{(k\|z) : z \in \{0,1\}^\lambda\}$ for $k \geq j$, except when $(k\|z) = i^*$. This may be viewed as the $j^{th}$ key being punctured on points $[1, j-1] \cup i^*$. We show that the GGM based construction for puncturing a single point can be extended to puncturing an interval (plus an extra point). Intuitively, puncturing an interval corresponds to puncturing at most $\lambda$ internal nodes in the GGM tree. In more detail, we show that regardless of the value of $j$, it suffices to puncture at most $\lambda$ points in the GGM tree to achieve puncturing of the entire interval $[1, j-1]$. Please see Appendix D for details.

## 4 Construction: Multi-Input FE for Turing Machines

In this section we construct a multi-input functional encryption scheme for Turing machines. Our construction supports a fixed number of encryptors (say $k$), who may each encrypt a string $\mathbf{w}_i$ of unbounded length. Function keys may be provided for Turing machines, so that given $k$ ciphertexts for $\mathbf{w}_i$ and a function key for TM $M$ , decryption reveals $M(\mathbf{w}_1\| \ldots \|\mathbf{w}_k)$ and nothing else. We use the following ingredients for our construction:

1. A compact, $k$-input functional encryption scheme for circuits, $\mathsf{kFE}$ and a compact, public-key functional encryption scheme $\mathsf{1FE}$. As before, we will assume that the scheme $\mathsf{1FE}$ is decomposable as defined in Section 2.

2. A symmetric encryption scheme $\mathsf{SKE} = (\mathsf{SKE.KeyGen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$.

3. A delegatable constrained pseudorandom function ($\mathsf{cPRF}$), denoted by $\mathsf{F}$ which supports $T$ delegations for the function family $f_t : \{0,1\}^{(k+2)\cdot\lambda} \to \{0,1\}$ defined as follows. Let $x, t$ denote integers whose binary representations are $\mathbf{x}, \mathbf{t}$ of $\lambda$ bits. Then,

$$f_t(\mathbf{x}\|\mathbf{z}) = 1, \text{if } \; x \geq t \text{ and } 0 \text{ otherwise}$$

The functionalities supported by $\mathsf{kFE}$ and $\mathsf{1FE}$ are called $\mathsf{Agg}$ and $\mathsf{Next}$ respectively, described next. $\mathsf{Agg}$ aggregates the inputs $\mathbf{w}_1, \ldots, \mathbf{w}_k$ of all $k$ parties into one long "global" string $(\mathbf{w}_1\| \ldots \|\mathbf{w}_k)$, encrypted under the scheme $\mathsf{1FE}$. Since the length of this aggregate string is unbounded, a single invocation of $\mathsf{Agg}$ produces an encryption of a single symbol in the string, and the function is invoked repeatedly to produce ciphertexts for the entire string. Each ciphertext output by the $\mathsf{Agg}$ scheme contains a symbol $w_i$ as well as the position of the symbol within the global string. The encryption of the symbols (and the initial state) also contains a global salt which $\mathsf{Agg}$ computes from the random salts provided in the ciphertexts under the $\mathsf{kFE}$ scheme by the individual encryptors. The global salt identifies the particular input combination that is aggregated, and serves as input to the PRF in the $\mathsf{Next}$ functionality.

Our $k$-input $\mathsf{CktFE}$ scheme may be either private or public key, and will result in the corresponding notion for $k$-input $\mathsf{TMFE}$. Since the multi input setting for $\mathsf{FE}$ is considered

more interesting in the symmetric key setting (see [BKS16] for a discussion), we present our construction in the symmetric key setting – the public key adaptation is straightforward.

We note that ciphertexts output by Agg, which are encryptions of the symbols in the aggregate string under the 1FE scheme, are exactly the same as the output of the ReRand function in the single input scheme of Section 3. Therefore, as before, we may have the functionality Next of the 1FE scheme mimic the computation of the Turing machine on the global string $(\mathbf{w}_1 \| \ldots \| \mathbf{w}_k)$. As in the previous construction, 1FE.Dec accepts as its inputs a ciphertext decomposed into two components encoding the current symbol on the worktape and the current state in the computation, both of which have been encrypted using the same randomness, and outputs a ciphertext component corresponding to the symbol written on the tape, as well as the next state. The global salt in the ciphertext, along with a random nonce chosen by KeyGen are used as input to a cPRF as before, to compute the randomness used to generate ciphertexts. This ensures that the execution of a given machine on a given input combination is maintained separate from any other execution, and thwarts "mix and match" attacks, where, for instance, an attacker may try to combine a state generated at some time step $t$ in one execution with a symbol generated at time step $t$ from a different execution.

If we instantiate the underlying multi-input CktFE by the construction of [KS17], we may let the arity $k$ be poly-logarithmic in the security parameter. If we instantiate multi-input CktFE by the construction of [GGG+14], we may support fixed polynomial arity at the cost of worsening the assumption. Note that [GGG+14] rely on iO while [KS17] rely on compact FE. Note that [BGJS15] support unbounded polynomial arity, but from public coin DiO as discussed in Section 1.

### 4.1 Construction of multi-input TMFE

In the following, we denote a $k$-input, private-key CktFE scheme by $k$-CktFE and a decomposable, public key CktFE scheme by 1FE. Since our scheme supports an a-priori fixed number of parties, say $k$, we assume that every user is pre-assigned an index $\mathsf{ind} \in [k]$.

$\mathsf{kTMFE.Setup}(1^\lambda, 1^k)$: Upon input the security parameter $1^\lambda$ and the bound $1^k$, do the following:

1. Choosing the functionality for 1FE. Let 1FE be a decomposable, public-key CktFE for the following circuit family.

$$\mathsf{Next}: \Big( (\{\mathsf{SYM}\} \times \{0,1\}^{(k+4)\lambda} \times \Sigma \times \mathsf{Trap}) \times (\{\mathsf{ST}\} \times \mathcal{Q} \times \{0,1\}^{k \cdot \lambda}) \Big) \rightarrow \Big(\mathcal{C}^{\mathsf{1FE}}\Big)^2 \cup \{\mathsf{ACC}, \mathsf{REJ}, \perp\}$$

   The tokens SYM and ST are flags denoting a symbol and a state respectively of a Turing machine $M$ which has $\Sigma$ and $\mathcal{Q}$ as the alphabet and state space respectively. The set $\{0,1\}^{(k+4)\lambda}$ encodes in order, a random value key-id associated with a TM $M$, a constrained PRF key, the current time step in the computation, the length of the input string, each of $\lambda$ bits and a string of length $k \cdot \lambda$ bits encoding a random value gsalt. Here, Trap is a data structure of fixed polynomial length which will be used in the proof. Since we do not need it in the construction, we do not discuss it here, please see Figure 17 for its definition. The set $\{0,1\}^{k \cdot \lambda}$ encodes again a random value gsalt associated with the message component for state. $\mathcal{C}^{\mathsf{1FE}}$ is the ciphertext space of 1FE. ACC and REJ denote tokens when $M$ reaches an accepting state and a rejecting state respectively.

2. Choosing the functionality for $\mathsf{kFE}$. Let $\mathsf{kFE}$ be a $k$-$\mathsf{CktFE}$ for the following circuit family.

$$\mathsf{Agg} : (\{\mathsf{SYM}, \mathsf{SP}\} \times \{0,1\}^{4\lambda} \times [k] \times \Sigma \times \mathsf{Trap})^k \to \mathcal{C}^{\mathsf{1FE}} \times \left(\mathcal{C}^{\mathsf{1FE}} \cup \{\bot\}\right)$$

The special token $\mathsf{SP}$ denotes an encryption of the length of an input string corresponding to any user. The set $\{0,1\}^{4\lambda}$ encodes in order, a constrained PRF key, the time step of the current symbol, the input length and a random salt each of $\lambda$ bits. $\Sigma, \mathsf{Trap}$ and $\mathcal{C}^{\mathsf{1FE}}$ are as described above.

3. Choosing keys for $\mathsf{kFE}$ and $\mathsf{1FE}$.

$$\text{Let } \mathsf{kFE.MSK} \leftarrow \mathsf{kFE.Setup}(1^\lambda, 1^k), (\mathsf{1FE.PK}, \mathsf{1FE.MSK}) \leftarrow \mathsf{1FE.Setup}(1^\lambda, 1^k)$$

4. Output $\mathsf{MSK} = (\mathsf{kFE.MSK}, (\mathsf{1FE.PK}, \mathsf{1FE.MSK}))$.

$\mathsf{kTMFE.Enc}(\mathsf{MSK}, \mathbf{w}_{\mathsf{ind}}, \mathsf{ind})$: Upon input the master key $\mathsf{MSK}$, and message $\mathbf{w}_{\mathsf{ind}}$ of arbitrary length $\ell_{\mathsf{ind}}$ and an index $\mathsf{ind} \in [k]$, do the following:

1. Interpret the input $\mathsf{MSK} = (\mathsf{kFE.MSK}, (\mathsf{1FE.PK}, \mathsf{1FE.MSK}))$.
2. Let $\mathbf{w}_{\mathsf{ind}} = w_1 w_2 \ldots w_{\ell_{\mathsf{ind}}}$. Sample $\mathsf{salt}_{\mathsf{ind}} \leftarrow \{0,1\}^\lambda$.
3. Construct the data structure $\mathsf{Trap}$ and set all its fields to $\bot$ except a flag $\mathsf{Trap.mode\text{-}real} = 1$ which indicates that we are in the real world. The data structure $\mathsf{Trap}$ is only relevant in the proof. Please see Figure 6 for the definition of $\mathsf{Trap}$.

- Encoding Input String and Its Length

4. If $\mathsf{ind} = 1$, do the following:
   (a) Sample a root key for the constrained PRF $\mathsf{F}$ as $\mathsf{K}_0 \leftarrow \mathsf{F.Setup}(1^\lambda)$.
   (b) Construct the input message $\mathsf{len}_1 = (\mathsf{SP}, \mathsf{K}_0, \bot, \ell_1, \mathsf{salt}_1, 1, \bot, \mathsf{Trap})$.
   (c) Encrypt $\ell_1$ as a special ciphertext $\mathsf{CT}_{1,\mathsf{SP}} = \mathsf{kFE.Enc}(\mathsf{kFE.MSK}, \mathsf{len})$.
   (d) For $i \in [\ell_1]$ do the following:
      i. Construct the input message $\mathbf{y}_{1,i} = (\mathsf{SYM}, \mathsf{K}_0, i, \ell_1, \mathsf{salt}_1, 1, w_i, \mathsf{Trap})$.
      ii. Compute the ciphertext $\mathsf{CT}_{1,\mathsf{SYM},i} = \mathsf{kFE.Enc}(\mathsf{kFE.MSK}, \mathbf{y}_i)$.
5. If $\mathsf{ind} \in [2, k]$, do the following:
   (a) Construct the input message $\mathsf{len}_{\mathsf{ind}} = (\mathsf{SP}, \bot, \bot, \ell_{\mathsf{ind}}, \mathsf{salt}_{\mathsf{ind}}, \mathsf{ind}, \bot, \mathsf{Trap})$.
   (b) Encrypt $\ell_{\mathsf{ind}}$ as a special ciphertext $\mathsf{CT}_{\mathsf{ind},\mathsf{SP}} = \mathsf{kFE.Enc}(\mathsf{kFE.MSK}, \mathsf{len})$.
   (c) For $i \in [\ell_{\mathsf{ind}}]$ do the following:
      i. Construct the input message $\mathbf{y}_{\mathsf{ind},i} = (\mathsf{SYM}, \bot, i, \ell_{\mathsf{ind}}, \mathsf{salt}_{\mathsf{ind}}, \mathsf{ind}, w_i, \mathsf{Trap})$.
      ii. Compute the ciphertext $\mathsf{CT}_{\mathsf{ind},\mathsf{SYM},i} = \mathsf{kFE.Enc}(\mathsf{kFE.MSK}, \mathbf{y}_i)$.
6. Output $\mathsf{CT}_{\mathbf{w}_{\mathsf{ind}}} = \left(\mathsf{CT}_{\mathsf{ind},\mathsf{SP}}, \{\mathsf{CT}_{\mathsf{ind},\mathsf{SYM},i}\}_{i \in [\ell_{\mathsf{ind}}]}\right)$.

$\mathsf{kTMFE.KeyGen}(\mathsf{MSK}, M)$: Upon input the master secret key $\mathsf{MSK}$ and the description of a Turing machine $M$, do the following. We will assume, w.l.o.g. that the TM is oblivious (see Appendix A for a justification) and $\mathsf{q}_{\mathsf{st}} \in \mathcal{Q}$ is the start state of $M$.

1. Sample a random value $\mathsf{rand} \leftarrow \{0,1\}^\lambda$.
2. Interpret $\mathsf{MSK} = (\mathsf{kFE.MSK}, (\mathsf{1FE.PK}, \mathsf{1FE.MSK}))$.
3. Let $\mathsf{SK}_{\mathsf{Agg}} = \mathsf{kFE.KeyGen}(\mathsf{kFE.MSK}, \mathsf{Agg}_{\mathsf{1FE.PK},\mathsf{rand},\mathsf{q}_{\mathsf{st}},\perp,\perp})$, where Figure 7 defines the circuit $\mathsf{Agg}_{\mathsf{1FE.PK},\mathsf{rand},\mathsf{q}_{\mathsf{st}},\perp,\perp}$.
4. Let $\mathsf{SK}_{\mathsf{Next}} = \mathsf{1FE.KeyGen}(\mathsf{1FE.MSK}, \mathsf{Next}_{\mathsf{1FE.PK},\mathsf{rand},M,\perp,\perp})$, where Figure 9 defines the circuit $\mathsf{Next}_{\mathsf{1FE.PK},\mathsf{rand},M,\perp,\perp}$.
5. Output the secret key as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{Agg}}, \mathsf{SK}_{\mathsf{Next}})$.

Function $\mathsf{Agg}_{\mathsf{1FE.PK},\mathsf{rand},\mathsf{q}_{\mathsf{st}},\mathcal{C}_1,\mathcal{C}_2}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$

(a) Interpret $\mathbf{x}_i = (\mathsf{type}, \mathsf{K}, t, \ell, \mathsf{salt}, \mathsf{ind}, s, \mathsf{Trap})$, for $i \in [k]$ and set a flag $\mathsf{proceed}_1 = \mathsf{proceed}_2 = 0$.

(b) For all $i, j \in [k]$, if $\mathbf{x}_i.\mathsf{ind} \neq \mathbf{x}_j.\mathsf{ind}$ for $i \neq j$, set $\mathsf{proceed} = 1$. If there exists exactly one $i \in [k]$ for which $\mathbf{x}_i.\mathsf{type} = \mathsf{SYM}$ and $\mathbf{x}_j.\mathsf{type} = \mathsf{SP}, \forall j \in [k] \setminus \{i\}$ and $\mathsf{proceed}_1 = 1$, set $\mathsf{proceed}_2 = 1$. If $\mathsf{proceed}_2 = 0$, output $\perp$ and abort.

(c) Initialization and Choosing Real or Trapdoor mode.
Let $i \in [k]$ be such that $\mathbf{x}_i.\mathsf{type} = \mathsf{SYM}$. Initialize an input vector $\mathsf{inp} = (\sigma, \mathsf{q}_{\mathsf{st}})$, where $\sigma = \mathbf{x}_i.s$. Let $\mathsf{gsalt} = (\mathbf{x}_1.\mathsf{salt} \| \mathbf{x}_2.\mathsf{salt} \| \ldots \| \mathbf{x}_k.\mathsf{salt})$ and $\ell = \sum_{i=1}^{k} \mathbf{x}_i.\ell$ denote the global salt and the aggregate input length respectively. Denote $\mathsf{pos} = \mathbf{x}_i.t$ and do the following:

i. Computing Global Symbol Position : If $1 < \mathbf{x}_i.\mathsf{ind} \leq k$, compute the new position of the symbol as $\mathsf{pos} = \mathsf{pos} + \sum_{r \in \mathcal{S}} \mathbf{x}_r.\ell$, where the set $\mathcal{S} = \{r \mid \mathbf{x}_r.\mathsf{ind} < \mathbf{x}_i.\mathsf{ind}\} \subset [k]$.

ii. If $\mathsf{Trap.mode\text{-}real} = 1$, set $\mathsf{out} = (c_1, c_2)$, where $c_1 = c_2 = \perp$. If $\mathsf{pos} \neq 1$, set $\mathsf{inp} = (\sigma, \perp)$.

iii. Else obtain $\big(\mathsf{inp} = (u_1, u_2), \mathsf{out} = (c_1, c_2)\big) = \mathsf{Trap\text{-}Mode}_{\mathsf{Agg}}\big(\mathsf{Trap}, \mathsf{inp}, \mathsf{rand}, \mathsf{gsalt}, \ell, \mathcal{C}_1, \mathcal{C}_2, \mathsf{pos}\big)$ as described in Figure 8.

(d) If $((\mathsf{out}.c_1 = \perp) \vee (\mathsf{out}.c_2 = \perp))$, do the following.

i. Let $p \in [k]$ be such that $\mathbf{x}_p.\mathsf{ind} = 1$ and denote $\mathsf{K}_0 = \mathbf{x}_p.\mathsf{K}$ as the root key for cPRF.

ii. Derive the randomness for encryption at time step pos as $r_{\mathsf{pos}} = \mathsf{F.Eval}(\mathsf{K}_0, (\mathsf{pos} \| \mathsf{rand} \| \mathsf{gsalt}))$.

iii. Computing Encrypted Symbols using randomness derived from cPRF. If $\mathsf{out}.c_1 = \perp$, do the following.

- Compute the delegated PRF key $\mathsf{K}_{\mathsf{pos}+1} = \mathsf{F.KeyDel}(\mathsf{K}_0, f_{\mathsf{pos}+1})$. Set $\mathsf{key\text{-}id} = \mathsf{rand}$.
- Compute the 1FE symbol ciphertext encoding $\sigma = \mathsf{inp}.u_1$ as $\mathsf{CT}_{\mathsf{sym},\mathsf{pos}} = \bar{\mathcal{E}}\left(\mathsf{1FE.PK}_1, \mathbf{y}_1; r_{\mathsf{pos}}\right)$,
  where $\mathbf{y}_1 = (\mathsf{SYM}, \mathsf{key\text{-}id}, \mathsf{K}_{\mathsf{pos}+1}, \mathsf{pos}, \ell, \mathsf{gsalt}, \sigma, \mathsf{Trap})$.

iv. Computing Encrypted State for First Time Step. If $((\mathsf{out}.c_2 = \perp) \wedge (\mathsf{pos} = 1))$, do the following.

- Compute the 1FE state ciphertext encoding $\mathsf{q}_{\mathsf{st}} = \mathsf{inp}.u_2$ as $\mathsf{CT}_{\mathsf{st},1} = \bar{\mathcal{E}}(\mathsf{1FE.PK}_2, \mathbf{y}_2; r_1)$, where $\mathbf{y}_2 = (\mathsf{ST}, \mathsf{q}_{\mathsf{st}}, \mathsf{gsalt})$. Set $\mathsf{out}.c_2 = \mathsf{CT}_{\mathsf{st},1}$.

(e) If $\mathsf{pos} = 1$, output $\mathsf{out} = (\mathsf{CT}_{\mathsf{sym},1}, \mathsf{CT}_{\mathsf{st},1})$. Otherwise, output $\mathsf{out} = (\mathsf{CT}_{\mathsf{sym},\mathsf{pos}}, \perp)$.

Figure 7: This circuit aggregates and re-randomizes the ciphertexts provided during encryption to use randomness derived from a cPRF. The seed for the cPRF is specified in the ciphertext for first party and the input is specified by the key. This ensures that each ciphertext, key pair form a unique "thread" of execution.

Subroutine $\mathsf{Trap\text{-}Mode}_{\mathsf{Agg}}(\mathsf{Trap}, \mathsf{inp}, \mathsf{rand}, \mathsf{gsalt}, \ell, \mathcal{C}_1, \mathcal{C}_2, \mathsf{pos})$

Interpret $\mathsf{inp} = (u_1, u_2) = (w_i, \mathsf{q_{st}})$ and initialize $\mathsf{out} = (c_1, c_2)$, where $c_1 = c_2 = \perp$.

If Trap.key-id $=$ rand, do the following.

(a) If $\big((\mathsf{Trap.global\text{-}salt} = \mathsf{gsalt}) \wedge (\mathsf{Trap.mode\text{-}trap}_3 = 1)\big)$, do the following:

i. If $\big((\mathsf{Trap.Sym\ TS} = \mathsf{pos}) \wedge (\mathsf{pos} \leq \ell)\big)$, compute $\mathsf{CT}_{\mathsf{sym},\mathsf{pos}} = \mathsf{SKE.Dec}(\mathsf{Trap.SKE.K}, \mathcal{C}_1)$ and set $\mathsf{out}.c_1 = \mathsf{CT}_{\mathsf{sym},\mathsf{pos}}$.

ii. If $\big((\mathsf{Trap.ST\ TS} = \mathsf{pos}) \wedge (\mathsf{pos} = 1)\big)$, compute $\mathsf{CT}_{\mathsf{st},\mathsf{pos}} = \mathsf{SKE.Dec}(\mathsf{Trap.SKE.K}, \mathcal{C}_2)$ and set $\mathsf{out}.c_2 = \mathsf{CT}_{\mathsf{st},1}$.

(b) If $\big((\mathsf{Trap.global\text{-}salt} = \mathsf{gsalt}) \wedge (\mathsf{Trap.mode\text{-}trap}_1 = 1)\big)$, do the following:

i. If $\big((\mathsf{Trap.Sym\ TS}_1 = \mathsf{pos}) \wedge (\mathsf{pos} \leq \ell)\big)$, set $\mathsf{inp}.u_1 = \mathsf{Trap.Sym\ val}_1$ with the symbol to be encrypted and output at time step pos.

ii. If $\big((\mathsf{Trap.ST\ TS}_1 = \mathsf{pos}) \wedge (\mathsf{pos} = 1)\big)$, set $\mathsf{inp}.u_2 = \mathsf{Trap.ST\ val}_1$ with the start state to be encrypted and output at time step 1.

(c) If $\big((\mathsf{Trap.global\text{-}salt} = \mathsf{gsalt}) \wedge (\mathsf{Trap.mode\text{-}trap}_2 = 1)\big)$, do the following:

i. If $\big((\mathsf{Trap.Sym\ TS}_2 = \mathsf{pos}) \wedge (\mathsf{pos} \leq \ell)\big)$, set $\mathsf{inp}.u_1 = \mathsf{Trap.Sym\ val}_2$ with the symbol to be encrypted and output at time step pos.

ii. If $\big((\mathsf{Trap.ST\ TS}_2 = \mathsf{pos}) \wedge (\mathsf{pos} = 1)\big)$, set $\mathsf{inp}.u_2 = \mathsf{Trap.ST\ val}_2$ with the start state to be encrypted and output at time step 1.

(d) If Trap.global-salt $<$ gsalt, set $b = 0$, if Trap.global-salt $>$ gsalt, set $b = 1$.

i. If $\mathsf{pos} \neq 1$, update $\mathsf{inp} = (\mathsf{Trap.val}_b, \perp)$; else update $\mathsf{inp} = (\mathsf{Trap.val}_b, \mathsf{q_{st}})$.

If Trap.key-id $>$ rand, set $b = 1$, if Trap.key-id $<$ rand set $b = 0$.

(a) If $\mathsf{pos} \neq 1$, update $\mathsf{inp} = (\mathsf{Trap.val}_b, \perp)$; else update $\mathsf{inp} = (\mathsf{Trap.val}_b, \mathsf{q_{st}})$.

Output. Return $(\mathsf{inp}, \mathsf{out})$.

Figure 8: Subroutine handling the trapdoor modes in Agg. This is "active" only in the proof.

$\mathsf{kTMFE.Dec}(\mathsf{SK}_M, \{\mathsf{CT}_{\mathbf{w}_i}\}_{i\in[k]})$: Upon input secret key $\mathsf{SK}_M$ and $k$ ciphertexts $\mathsf{CT}_{\mathbf{w}_1}, \ldots, \mathsf{CT}_{\mathbf{w}_k}$, do the following:

1. Interpret the secret key as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{Agg}}, \mathsf{SK}_{\mathsf{Next}})$.
2. Parse $\mathsf{CT}_{\mathbf{w}_{\mathsf{ind}}} = (\mathsf{CT}_{\mathsf{ind},\mathsf{SP}}, (\mathsf{CT}_{\mathsf{ind},\mathsf{SYM},1}, \ldots, \mathsf{CT}_{\mathsf{ind},\mathsf{SYM},\ell_{\mathsf{ind}}}))$ for all $\mathsf{ind} \in [k]$.

- Aggregate the ciphertexts of all users.

3. For $i = 1$ to $k$, do the following:
   (a) For $j = 1$ to $\ell_i$, do the following:
      i. If $((i = 1) \wedge (j = 1))$, invoke $\mathsf{kFE.Dec}\big(\mathsf{SK}_{\mathsf{Agg}}, (\mathsf{CT}_{1,\mathsf{SYM},1}, \{\mathsf{CT}_{n,\mathsf{SP}}\}_{n\in[k]\setminus\{1\}})\big)$ to obtain $(\mathsf{CT}_{\mathsf{sym},1}, \mathsf{CT}_{\mathsf{st},1})$.
      ii. If $((i = 1) \wedge (j > 1))$, invoke $\mathsf{kFE.Dec}\big(\mathsf{SK}_{\mathsf{Agg}}, (\mathsf{CT}_{1,\mathsf{SYM},j}, \{\mathsf{CT}_{n,\mathsf{SP}}\}_{n\in[k]\setminus\{1\}})\big)$ to obtain $(\mathsf{CT}_{\mathsf{sym},j}, \perp)$.

Function $\mathsf{Next}_{\mathsf{1FE.PK},\mathsf{rand},\mathsf{M},\mathcal{C}_1,\mathcal{C}_2}\big((\mathbf{z}_1,\mathbf{z}_2)\big)$

(a) Reading Current (Symbol, State) Pair and Looking up Transition Table.

i. Interpret $\mathbf{z}_1 = (\mathsf{type}, \mathsf{key\text{-}id}, \mathsf{K}_{t+1}, t, \ell, \mathsf{gsalt}, s, \mathsf{Trap}), \mathbf{z}_2 = (\mathsf{type}, s, \mathsf{gsalt})$. If $((\mathbf{z}_1.\mathsf{type} \neq \mathsf{SYM}) \vee (\mathbf{z}_2.\mathsf{type} \neq \mathsf{ST}) \vee (\mathbf{z}_1.\mathsf{key\text{-}id} \neq \mathsf{rand}) \vee \wedge(\mathbf{z}_1.\mathsf{gsalt} \neq \mathbf{z}_2.\mathsf{gsalt}))$, output $\perp$ and abort.

ii. Interpret $(\mathbf{z}_1.s, \mathbf{z}_2.s) = (\sigma_t, q_t)$ as the symbol, state pair for the current time step $\mathbf{z}_1.t = t$, input $\mathbf{z}_1.\mathsf{K}_{t+1} = \mathsf{K}_{t+1}$ as the constrained PRF key for future time steps. Denote $\mathsf{key\text{-}id} = \mathbf{z}_1.\mathsf{key\text{-}id}, \ell = \mathbf{z}_1.\ell, \mathsf{gsalt} = \mathbf{z}_1.\mathsf{gsalt}$ and $\mathsf{Trap} = \mathbf{z}_1.\mathsf{Trap}$. Using the transition table of the machine $M$, look up the next state $q_{t+1}$ as well as the symbol $\sigma_{t'}$ to be written on the work-tape, where $t'$ is the time step the current work tape cell will next be read by $M$. If $q_{t+1}$ is an accept or reject state, then output ACC or REJ and exit.

iii. Initialize $\mathsf{inp} = (\sigma_{t'}, \mathsf{q}_{t+1})$.

(b) Choosing Real or Trapdoor mode. If Trap.mode-real $= 1$, initialize an output vector $\mathsf{out} = (c_1, c_2)$, where $c_1 = c_2 = \perp$. Else invoke $\mathsf{Trap\text{-}Mode}_{\mathsf{Next}}\big(\mathsf{Trap}, \mathsf{inp}, \mathsf{rand}, \mathsf{gsalt}, \ell, \mathcal{C}_1, \mathcal{C}_2, t, t'\big)$ as described in Figure 10 to obtain $\big(\mathsf{inp}=(u_1,u_2), \mathsf{out}=(c_1,c_2)\big)$.

(c) Computing Next Encrypted Symbol. If $\mathsf{out}.c_1 = \perp$, do the following.

i. Noting that $t' > t$, derive the randomness at time step $t'$ using the delegated key $\mathsf{K}_{t+1}$ as $r_{t'} = \mathsf{F.Eval}(\mathsf{K}_{t+1}, (t' \| \mathsf{rand} \| \mathsf{gsalt}))$. Compute the delegated PRF key $\mathsf{K}_{t'+1} = \mathsf{F.KeyDel}(\mathsf{K}_{t+1}, f_{t'+1})$.

ii. Compute the 1FE ciphertext component encoding the symbol $\sigma_{t'} = \mathsf{inp}.u_1$ for time step $t'$ as

$$\mathsf{CT}_{\mathsf{sym},t'} = \bar{\mathcal{E}}\big(\mathsf{1FE.PK}_1, (\mathsf{SYM}, \mathsf{key\text{-}id}, \mathsf{K}_{t'+1}, t', \ell, \mathsf{gsalt}, \sigma_{t'}, \mathsf{Trap}); r_{t'}\big)$$

iii. Set $\mathsf{out}.c_1 = \mathsf{CT}_{\mathsf{sym},t'}$.

(d) Computing Next Encrypted State. If $\mathsf{out}.c_2 = \perp$, do the following.

i. Derive the randomness at time step $t+1$ as $r_{t+1} = \mathsf{F.Eval}(\mathsf{K}_{t+1}, (t+1 \| \mathsf{rand} \| \mathsf{salt}))$ and compute the $\mathsf{1FE}_2$ ciphertext component encoding the state $\mathsf{q}_{t+1} = \mathsf{inp}.u_2$ for time step $t+1$ as

$$\mathsf{CT}_{\mathsf{st},t+1} = \bar{\mathcal{E}}\big(\mathsf{1FE}_2.\mathsf{PK}_2, (\mathsf{ST}, \mathsf{q}_{t+1}, \mathsf{gsalt}); r_{t+1}\big)$$

ii. Set $\mathsf{out}.c_2 = \mathsf{CT}_{\mathsf{st},t+1}$.

(e) Output : $\mathsf{out} = \big(\mathsf{CT}_{\mathsf{sym},t'}, \mathsf{CT}_{\mathsf{st},t+1}\big)$

Figure 9: Function to mimic TM computation. It reads the current symbol, state pair and outputs an encryption of the new state and symbol to be written under the appropriate randomness generated using a cPRF.

iii. Else, invoke $\mathsf{kFE.Dec}\left(\mathsf{SK}_{\mathsf{Agg}}, \left(\mathsf{CT}_{i,\mathsf{SYM},j}, \{\mathsf{CT}_{n,\mathsf{SP}}\}_{n\in[k]\setminus\{i\}}\right)\right)$ to obtain $(\mathsf{CT}_{\mathsf{sym},\widetilde{L}_i+j}, \perp)$, where $\widetilde{L}_i = \sum_{m=1}^{i-1} \ell_m$.

- Execute the TM on aggregated input.

4. The aggregated sequence of ciphertexts under the Next scheme, of length $L_k = \sum_{j=1}^{k} \ell_j$ computed above is expressed as:
$((\mathsf{CT}_{\mathsf{sym},1}, \mathsf{CT}_{\mathsf{st},1}), \mathsf{CT}_{\mathsf{sym},2}, \ldots, \mathsf{CT}_{\mathsf{sym},\ell_1}, \mathsf{CT}_{\mathsf{sym},\ell_1+1}, \ldots, \mathsf{CT}_{\mathsf{sym},L_k})$.
5. Let $t = 1$. While the Turing machine does not halt, do:
   (a) Invoke $\mathsf{1FE.Dec}\big(\mathsf{SK}_{\mathsf{Next}}, (\mathsf{CT}_{\mathsf{sym},t}, \mathsf{CT}_{\mathsf{st},t})\big)$ to obtain:
   - ACC or REJ. In this case, output "Accept" or "Reject" respectively, and exit

Subroutine $\mathsf{Trap\text{-}Mode}_{\mathsf{Next}}\big(\mathsf{Trap}, \mathsf{inp}, \mathsf{rand}, \mathsf{gsalt}, \ell, \mathcal{C}_1, \mathcal{C}_2, t, t'\big)$

Interpret the input vector $\mathsf{inp} = (u_1, u_2) = (\sigma_{t'}, \mathsf{q}_{t+1})$ and initialize the output vector $\mathsf{out} = (c_2, c_2)$, where $c_1 = c_2 = \bot$.

1. If $\big((\mathsf{Trap.key\text{-}id} = \mathsf{salt}) \wedge (\mathsf{Trap.global\text{-}salt} = \mathsf{gsalt}) \wedge (\mathsf{Trap.mode\text{-}trap}_3 = 1)\big)$, do the following.
   (a) If $\big((\mathsf{Trap.Sym\ TS} = t) \wedge (\mathsf{Trap.Target\ TS} = t') \wedge (t > \ell)\big)$, compute the $\mathsf{1FE}_2$ symbol ciphertext $\mathsf{CT}_{\mathsf{sym},t'} = \mathsf{SKE.Dec}(\mathsf{Trap.SKE.K}, \mathcal{C}_1)$ and set $\mathsf{out}.c_1 = \mathsf{CT}_{\mathsf{sym},t'}$.
   (b) If $\big((\mathsf{Trap.ST\ TS} = t) \wedge (\mathsf{Trap.Target\ TS} = t+1) \wedge (t > 1)\big)$, compute the $\mathsf{1FE}_2$ state ciphertext $\mathsf{CT}_{\mathsf{st},t+1} = \mathsf{SKE.Dec}(\mathsf{Trap.SKE.K}, \mathcal{C}_2)$ and set $\mathsf{out}.c_2 = \mathsf{CT}_{\mathsf{st},t+1}$.
2. If $\big((\mathsf{Trap.key\text{-}id} = \mathsf{salt}) \wedge (\mathsf{Trap.global\text{-}salt} = \mathsf{gsalt}) \wedge (\mathsf{Trap.mode\text{-}trap}_1 = 1)\big)$, do the following:
   (a) If $\big((\mathsf{Trap.Sym\ TS}_1 = t) \wedge (\mathsf{Trap.Target\ TS}_1 = t') \wedge (t > \ell)\big)$, set $\mathsf{inp}.u_1 = \mathsf{Trap.Sym\ val}_1$ with the symbol $\sigma_{t'} = \mathsf{Trap.Sym\ val}_1$ to be encrypted and given as output for time step $t'$.
   (b) If $\big((\mathsf{Trap.ST\ TS}_1 = t) \wedge (\mathsf{Trap.Target\ TS}_1 = t+1) \wedge (t > 1)\big)$, set $\mathsf{inp}.u_2 = \mathsf{Trap.ST\ val}_1$ with the state $\mathsf{q}_{t+1} = \mathsf{Trap.ST\ val}_1$ to be encrypted and given as output for time step $t+1$.
3. If $\big((\mathsf{Trap.key\text{-}id} = \mathsf{salt}) \wedge (\mathsf{Trap.global\text{-}salt} = \mathsf{gsalt}) \wedge (\mathsf{Trap.mode\text{-}trap}_2 = 1)\big)$, do the following:
   (a) If $\big((\mathsf{Trap.Sym\ TS}_2 = t) \wedge (\mathsf{Trap.Target\ TS}_2 = t') \wedge (t > \ell)\big)$, set $\mathsf{inp}.u_1 = \mathsf{Trap.Sym\ val}_2$ with the symbol $\sigma_{t'} = \mathsf{Trap.Sym\ val}_2$ to be encrypted and given as output for time step $t'$.
   (b) If $\big((\mathsf{Trap.ST\ TS}_2 = t) \wedge (\mathsf{Trap.Target\ TS}_2 = t+1) \wedge (t > 1)\big)$, set $\mathsf{inp}.u_2 = \mathsf{Trap.ST\ val}_2$ with the state $\mathsf{q}_{t+1} = \mathsf{Trap.ST\ val}_2$ to be encrypted and given as output for time step $t+1$.
4. Exit the subroutine returning $(\mathsf{inp}, \mathsf{out})$.

Figure 10: Subroutine handling the trapdoor modes in Next. This is "active" only in the proof.

the loop.
- $\big(\mathsf{CT}_{\mathsf{sym},t'}, \mathsf{CT}_{\mathsf{st},t+1}\big)$.

Note that $t'$ is the next time step that the work tape cell accessed at time step $t$ will be accessed again.

(b) Let $t = t+1$ and go to start of loop.

## 4.2 Correctness of Multi-Input TMFE

The proof of correctness is split into two parts. In the first part we argue that, given as input the secret key $\mathsf{SK}_{\mathsf{Agg}}$ along with $k$ ciphertexts under the kFE scheme, exactly one of which encodes a symbol and the other $(k-1)$ encode the individual input lengths, the kFE.Dec algorithm computes a 1FE ciphertext component of the symbol with its updated position in the global string. By repeating this process for all symbols encoded by all users, we obtain a sequence of 1FE ciphertext components, each containing its updated position in the aggregated string. Additionally, each of these ciphertext components contains a global/aggregate salt that is generated from concatenating each individual encryptor's randomly generated salts. This global salt identifies the particular input combination being aggregated.

Correctness of the second part corresponds to the correct execution of the Turing machine on the aggregate sequence of ciphertexts, and this is exactly the same as in Section 3. As before, we maintain the invariant that at any time step $t$, the input to the 1FE.Dec algorithm

is a complete 1FE ciphertext decomposed into two components corresponding to symbol and state (along with additional auxiliary inputs), both computed with the same randomness $\mathsf{F.Eval}(\mathsf{K}_0, (\mathsf{t}\|\mathsf{rand}\|\mathsf{gsalt}))$.

In more detail, we have the following. Correctness of Aggregation. Formally, let there be $k$ users so that $k$ ciphertexts $\{\mathsf{CT}_{\mathbf{w}_{\mathsf{ind}}}\}_{\mathsf{ind}\in[k]}$ are given as input to kTMFE.Dec algorithm. For all $\mathsf{ind} \in [k]$, let $\ell_{\mathsf{ind}}$ be the length of input string of user ind. Each ciphertext $\mathsf{CT}_{\mathbf{w}_{\mathsf{ind}}}$ is a sequence $(\mathsf{CT}_{\mathsf{ind},\mathsf{SP}}, (\mathsf{CT}_{\mathsf{ind},\mathsf{SYM},1}, \ldots, \mathsf{CT}_{\mathsf{ind},\mathsf{SYM},\ell_{\mathsf{ind}}}))$ of ciphertexts, where the first component $\mathsf{CT}_{\mathsf{ind},\mathsf{SP}}$ encodes the input string length of user ind and the second component $\{\mathsf{CT}_{\mathsf{ind},\mathsf{SYM},i}\}_{i\in[\ell_{\mathsf{ind}}]}$ encodes in order the $i$-th symbol $w_i$ of the actual input string $\mathbf{w}_{\mathsf{ind}} = (w_1, w_2, \ldots, w_{\ell_{\mathsf{ind}}})$ of the same user. These ciphertexts are generated under the kFE scheme with the master secret key kFE.MSK which supports a $k$-input functionality $\mathsf{Agg} := \mathsf{Agg}_{\mathsf{1FE.PK},\mathsf{rand},\mathsf{q}_{\mathsf{st}},\perp,\perp}$. Therefore, given secret key $\mathsf{SK}_{\mathsf{Agg}}$, we have:

1. Invoking kFE.Dec on the ciphertext $\mathsf{CT}_{1,\mathsf{SYM},1}$ encoding the first symbol of $\mathbf{w}_1$ along with the special ciphertexts $\mathsf{CT}_{\mathsf{ind},\mathsf{SP}}$ encoding $|\mathbf{w}_{\mathsf{ind}}|$ for $\mathsf{ind} \neq 1$ gives $(\mathsf{CT}_{\mathsf{sym},1}, \mathsf{CT}_{\mathsf{st},1})$. By correctness of kFE decryption, we have: $\mathsf{kFE.Dec}\left(\mathsf{SK}_{\mathsf{Agg}}, \left(\mathsf{CT}_{1,\mathsf{SYM},1}, \{\mathsf{CT}_{\mathsf{ind},\mathsf{SP}}\}_{\mathsf{ind}\in[k]\setminus\{1\}}\right)\right) = (\mathsf{CT}_{\mathsf{sym},1}, \mathsf{CT}_{\mathsf{st},1})$.

2. Invoking kFE.Dec on the ciphertext $\mathsf{CT}_{1,\mathsf{SYM},j}$ encoding the $j$th symbol of $\mathbf{w}_1$ along with the special ciphertexts $\mathsf{CT}_{\mathsf{ind},\mathsf{SP}}$ encoding $|\mathbf{w}_{\mathsf{ind}}|$ for $\mathsf{ind} \neq 1$ gives $(\mathsf{CT}_{\mathsf{sym},j}, \perp)$. By correctness of kFE decryption, we have: $\mathsf{kFE.Dec}\left(\mathsf{SK}_{\mathsf{Agg}}, \left(\mathsf{CT}_{1,\mathsf{SYM},j}, \{\mathsf{CT}_{\mathsf{ind},\mathsf{SP}}\}_{\mathsf{ind}\in[k]\setminus\{1\}}\right)\right) = (\mathsf{CT}_{\mathsf{sym},j}, \perp)$.

3. Finally, $\forall\ \mathsf{ind} \in [k] \setminus \{1\}$, invoking kFE.Dec on the ciphertext $\mathsf{CT}_{\mathsf{ind},\mathsf{SYM},j}$ encoding the $j$th symbol of $\mathbf{w}_{\mathsf{ind}}$ along with the special ciphertexts $\mathsf{CT}_{\mathsf{ind}',\mathsf{SP}}$ encoding $|\mathbf{w}_{\mathsf{ind}'}|$ for $\mathsf{ind} \neq \mathsf{ind}'$ computes the new global position of the symbol in the aggregated string and outputs $\left(\mathsf{CT}_{\mathsf{sym},\widetilde{L_i}+j}, \perp\right)$. By correctness of kFE decryption, we have: $\mathsf{kFE.Dec}\left(\mathsf{SK}_{\mathsf{Agg}}, \left(\mathsf{CT}_{\mathsf{ind},\mathsf{SYM},j}, \{\mathsf{CT}_{\mathsf{ind}',\mathsf{SP}}\}_{\mathsf{ind}'\in[k]\setminus\{\mathsf{ind}\}}\right)\right) = \left(\mathsf{CT}_{\mathsf{sym},\widetilde{L_i}+j}, \perp\right)$, where $\widetilde{L_i} = \sum_{m=1}^{\mathsf{ind}-1} \ell_m$.

Note that $\mathsf{F.Eval}(\mathsf{K}_0, (\mathsf{pos}\|\mathsf{rand}\|\mathsf{gsalt}))$ is the randomness used to compute each of these ciphertext components, where pos refers to the global position specific to a symbol in the aggregate input string.

Correctness of TM Execution. The 1FE scheme supports the functionality $\mathsf{Next} := \mathsf{Next}_{\mathsf{1FE.PK},\mathsf{rand},M,\perp,\perp}$. Let the newly generated and organized sequence of ciphertexts based on time steps be as follows: $\left((\mathsf{CT}_{\mathsf{sym},1}, \mathsf{CT}_{\mathsf{st},1}), \{\mathsf{CT}_{\mathsf{sym},i}\}_{i\in[2,L_k]}\right)$ with $L_k = \sum_{i=1}^{k} \ell_i$. Let $\mathbf{w} = (w_1, w_2, \ldots, w_{\ell_1}, w_{\ell_1+1},\ w_{\ell_1+2}, \ldots, w_{\ell_1+\ell_2}, \ldots, w_{L_k})$ be the aggregated input string and define $\tau = \mathsf{runtime}(M, \mathbf{w})$. For any time step $t \in [\tau - 2]$, we have

1. Let $t \in [\tau - 2] \setminus [\ell]$. If the current work tape cell was accessed[13], at some time step $\tilde{t} < t$, then $\mathsf{CT}_{\mathsf{sym},t}$ encoding $(\mathsf{SYM}, \mathsf{key\text{-}id}, \mathsf{K}_{t+1}, t, \ell, \mathsf{gsalt}, \sigma_t, \mathsf{Trap})$ was constructed at time step $\tilde{t}$. Note that $\sigma_t$ may be the blank symbol $\beta$. When $t \in [\ell]$, $\mathsf{CT}_{\mathsf{sym},t}$ is constructed at time step $t$ via the Agg circuit.

2. The ciphertext component $\mathsf{CT}_{\mathsf{st},t}$ encoding $(\mathsf{ST}, \mathsf{q}_t, \mathsf{gsalt})$ at time step $t$ was constructed at time step $t-1$ for $t > 1$ and at time step 1, when $t = 1$.

[13]We assume that every time a cell is accessed, it is written to, by writing the same symbol again if no change is made.

3. The randomness $r_t = \mathsf{F.Eval}(\mathsf{K}_{\tilde{t}+1}, (\mathsf{t}\|\mathsf{rand}\|\mathsf{gsalt})) = \mathsf{F.Eval}(\mathsf{K}_{\mathsf{t}}, (\mathsf{t}\|\mathsf{rand}\|\mathsf{gsalt}))$ binds $\mathsf{CT}_{\mathsf{sym},t}$ and $\mathsf{CT}_{\mathsf{st},t}$ and both the encoded messages also share the same global salt.

Thus, at any given time step $t \in [\tau - 2]$, we have a complete ciphertext of 1FE which may be fed again with $\mathsf{SK}_{\mathsf{Next}}$ to 1FE.Dec in order to proceed with the computation. Thus, the execution of 1FE.Dec at the $(\tau - 2)^{\text{th}}$ time step provides the complete pair $(\mathsf{CT}_{\mathsf{sym},\tau-1}, \mathsf{CT}_{\mathsf{st},\tau-1})$. By the correctness of 1FE scheme again, at time step $t = \tau - 1$, invoking $\mathsf{1FE.Dec}(\mathsf{SK}_{\mathsf{Next}}, (\mathsf{CT}_{\mathsf{sym},\tau-1}, \mathsf{CT}_{\mathsf{st},\tau-1}))$ outputs either "Accept" or "Reject" by simulating the execution of $M$ for the final time step $\tau$ inside the function Next, thus correctly outputting $M(\mathbf{w})$.

### 4.3 Proof of Security for multi-input TMFE

Security of the above construction follows the same blueprint as the proof in Section 3 except that instead of single input functionality ReRand, we now use a $k$-input functionality Agg to aggregate and rerandomize the inputs. We emphasize that the outputs produced by the Agg functionality are exactly the same as the outputs produced by ReRand functionality in Section 3: namely a sequence of 1FE ciphertexts encoding the symbol and global position, computed using randomness derived from a cPRF. Hence, the chief new ingredient in the security proof is the security of Agg functionality, which is derived from the security of the kFE scheme.

Formally, we argue that:

Theorem 4.1. Assume that the $k$ input FE for circuits kFE satisfies standard indistinguishability (Definition 2.5), and the single input FE for circuits 1FE satisfies distributional indistinguishability (Definition 2.3). Assume that the cPRF is secure according to definition 2.3. Then, the above construction of $k$ input kTMFE satisfies standard indistinguishability (Definition 2.8).

The proof follows the outline of the single input case, except that now we must additionally keep track of multiple execution threads corresponding to various combinations of ciphertexts across multiple users, i.e. various "global salt" values. In more detail, if each of $k$ users makes $Q$ ciphertext requests, then we have $Q^k$ total possible combinations of ciphertexts, each yielding a different execution thread per key. Note that each of the $Q^k$ combinations is identified with a unique "global salt". We will assume w.l.o.g that there is a lexicographic ordering on all the global salt values; this can be easily ensured by associating a counter value with each random salt. We do not explicitly include this for notational brevity.

In the single input case, we replaced the execution chain of a machine over an input string from $b = 0$ to $b = 1$, step by step, and enumerated over all keys. Now, we again replace an execution chain step by step as in the single input case, but additionally enumerate over all $Q^k$ combinations for each key, as well as over all keys as before. The number of hybrids grows multiplicatively by $Q^k$. Since the proof structure follows mostly like the single input case, we provide below only the conceptual description of the main ideas and the hybrids' sequence in the proof. Details are provided in Appendix C.

## 5 Indistinguishability Obfuscation for Turing Machines

In this section we construct indistinguishability obfuscation for Turing machines with bounded length input, i.e. the input length $n = n(\lambda)$ is any fixed polynomial in the security parameter.

Our construction is a straightforward adaptation of the miFE to iO compiler for circuits [GGG+14] to Turing machines. To support inputs of length $n$, we need an $(n+1)$-ary miFE

for Turing machines denoted as $(\mathsf{n}+1)$-$\mathsf{TMFE}$; we instantiate this with our construction from Section 4.

In more detail, the obfuscation of $M$ comprises the secret key $\mathsf{SK}_{\mathsf{U}}$ for the Universal Turing machine and $(2n+1)$ ciphertexts under the $(n+1)$-$\mathsf{TMFE}$ scheme, where the first $2n$ ciphertexts $\{\mathsf{CT}_i^b\}_{i\in[n],b\in\{0,1\}}$ encode bits 0 and 1 respectively for each of $n$ positions while the last ciphertext $\mathsf{CT}_M$ encodes machine $\langle M\rangle$. To evaluate $\mathsf{iO}(M)$ on an input $\mathbf{x} = (x_1, \ldots, x_n) \in \Sigma_\lambda^n$, the evaluator runs $(\mathsf{n}+1)$-$\mathsf{TMFE.Dec}\left(\mathsf{SK}_{\mathsf{U}}, (\{\mathsf{CT}_i^{x_i}\}_{i\in[n]}, \mathsf{CT}_M)\right)$ to get $M(\mathbf{x})$. To argue security we only need the $(\mathsf{n}+1)$-$\mathsf{TMFE}$ scheme to be selectively secure against two ciphertext queries per slot and a single key query, as in the case of circuits.

## 5.1 Construction

Let $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ denote an ensemble of Turing machines with alphabet $\Sigma_\lambda = \{0,1\}$. Let $\mathsf{Encode} = \{\mathsf{Encode}_\lambda : \mathcal{M}_\lambda \to \Sigma_{\mathsf{enc}}^*\}_{\lambda\in\mathbb{N}}$ be an ensemble of encoding schemes for $\mathcal{M}$ on alphabet $\Sigma_{\mathsf{enc}}$ such that for any $M \in \mathcal{M}_\lambda, \mathsf{Encode}_\lambda\ (M) = \langle M\rangle$. Further, let $\mathcal{U} = \{\mathsf{U}_\lambda\}_{\lambda\in\mathbb{N}}$ denote the set of Universal Turing machines parameterized by the security parameter with alphabet $\Sigma_{\mathcal{U}} = \Sigma_{\mathsf{enc}} \cup \Sigma_\lambda$ such that for all $\lambda \in \mathbb{N}$, for any $M \in \mathcal{M}_\lambda$ and any $\mathbf{x} = (x_1, \ldots, x_n) \in \Sigma_\lambda^n$, $\mathsf{U}_\lambda(\mathbf{x}, \langle M\rangle)$ takes $\mathbf{x}$ and an encoding $\langle M\rangle$ of $M$, simulates $M$ on $\mathbf{x}$ and outputs $M(\mathbf{x})$.

Let $(\mathsf{n}+1)$-$\mathsf{TMFE}$ denote the $(n+1)$-ary multi-input functional encryption scheme for Turing machines with alphabet $\Sigma_{\mathcal{U}}$. We construct an ensemble of indistinguishability obfuscators $\mathsf{iO} = \{\mathsf{iO}_\lambda\}_{\lambda\in\mathbb{N}}$ with $\mathsf{iO}_\lambda = (\mathsf{iO.Obf}, \mathsf{iO.Eval})$ for $\mathcal{M}_\lambda$ with inputs $\mathbf{x} \in \Sigma_\lambda^n$ as follows.

$\mathsf{iO.Obf}(1^\lambda, 1^n, M)$: On input the security parameter $\lambda$, a bound $n \in \mathbb{N}$ and a Turing machine $M \in \mathcal{M}_\lambda$, do the following:

1. Compute the encoding of $M$ as $\mathsf{Encode}_\lambda\ (M) = \langle M\rangle$.
2. Compute a master secret key $\mathsf{MSK} \leftarrow (\mathsf{n}+1)$-$\mathsf{TMFE.Setup}\ (1^\lambda, 1^{n+1})$.
3. Compute the secret key for machine $\mathsf{U}_\lambda$ as $\mathsf{SK}_{\mathsf{U}} \leftarrow (\mathsf{n}+1)$-$\mathsf{TMFE.KeyGen}(\mathsf{MSK}, \mathsf{U}_\lambda)$.
4. For $i \in [n]$, compute the encryptions $\mathsf{CT}_i^b = (\mathsf{n}+1)$-$\mathsf{TMFE.Enc}(\mathsf{MSK}, (b, i)), b \in \Sigma_\lambda$.
5. Compute the encoding of $M$ as $\mathsf{CT}_{n+1} = (\mathsf{n}+1)$-$\mathsf{TMFE.Enc}(\mathsf{MSK}, (\langle M\rangle, n+1))$.
6. Output the obfuscated machine as $\widetilde{M} = \left(\mathsf{SK}_{\mathsf{U}}, \left(\{\mathsf{CT}_i^b\}_{i\in\{1,\ldots,n\},b\in\Sigma_\lambda}, \mathsf{CT}_{n+1}\right)\right)$.

$\mathsf{iO.Eval}(\widetilde{M}, \mathbf{x})$: On input the obfuscated machine $\widetilde{M}$ and an input $\mathbf{x} \in \Sigma_\lambda^n$, do the following:

1. Parse $\widetilde{M} = \left(\mathsf{SK}_{\mathsf{U}}, \left(\{\mathsf{CT}_i^b\}_{i\in\{1,\ldots,n\},b\in\Sigma_\lambda}, \mathsf{CT}_{n+1}\right)\right)$ and $\mathbf{x} = (x_1, \ldots, x_n)$.
2. Compute and output $(\mathsf{n}+1)$-$\mathsf{TMFE.Dec}\left(\mathsf{SK}_{\mathsf{U}}, (\mathsf{CT}_1^{x_1}, \ldots, \mathsf{CT}_n^{x_n}, \mathsf{CT}_{n+1})\right)$.

Correctness and Efficiency. Correctness is directly followed by the correctness of $(\mathsf{n}+1)$-$\mathsf{TMFE}$ scheme. Since the $(\mathsf{n}+1)$-$\mathsf{TMFE}$ we use is compact, the obfuscation size obtained by the above scheme is $\mathsf{poly}(\lambda, |\mathsf{U}|, |M|, n)$.

## 5.2 Proof of Security

We show that the construction is secure. Formally:

Theorem 5.1. Assume that $(\mathsf{n}+1)$-$\mathsf{TMFE}$ is a 1-key, 2-ciphertext selectively secure $(n+1)$-ary multi-input functional encryption scheme for Turing machines which satisfies standard indistinguishability (Section 2.2.2). Then the construction in Section 5.1 is a secure indistinguishability obfuscator for the Turing machines (Section 2.2.3) with bounded input length $n$.

Proof. Consider two Turing machines $M_0, M_1 \in \mathcal{M}_\lambda$ such that $\forall \mathbf{x} \in \Sigma^n_\lambda, M_0(\mathbf{x}) = M_1(\mathbf{x})$. We now show that if there exists a $\mathsf{PPT}$ adversary $\mathcal{A}$ that distinguishes between $\widetilde{M_0} = \mathsf{iO}(M_0)$ and $\widetilde{M_1} = \mathsf{iO}(M_1)$ with non-negligible advantage, then there exists another $\mathsf{PPT}$ adversary $\mathcal{B}$ which breaks the $(\mathsf{n}+1)$-$\mathsf{TMFE}$ scheme with the same advantage. We construct $\mathcal{B}$ as follows.

$\mathcal{B}$ runs $\mathcal{A}$ to get two functionally equivalent machines $M_0, M_1 \in \mathcal{M}_\lambda$. It does the following:

1. $\mathcal{B}$ prepares a pair of sequences $(\mathbf{x}^0, \mathbf{x}^1)$, each containing two challenge message vectors for the $(\mathsf{n}+1)$-$\mathsf{TMFE}$ challenger $\mathcal{C}$ such that for all $b \in \{0,1\}, \mathbf{x}^b = \left\{(x^b_{1,1}, \ldots, x^b_{n+1,1}), (x^b_{1,2}, \ldots, x^b_{n+1,2})\right\}$.
   - For all $i \in [n]$, $\mathcal{B}$ sets $x^0_{i,1} = x^1_{i,1} = 0$ and $x^0_{i,2} = x^1_{i,2} = 1$
   - For $i = n+1$, $\mathcal{B}$ sets $x^b_{n+1,1} = x^b_{n+1,2} = \langle M_b \rangle$, where $\mathsf{Encode}_\lambda(\mathsf{M_b}) = \langle \mathsf{M_b} \rangle$.

   $\mathcal{B}$ sends the pair $(\mathbf{x}^0, \mathbf{x}^1)$ to $\mathcal{C}$ and receives $(\mathsf{CT}_{1,j}, \ldots, \mathsf{CT}_{n+1,j})_{j\in[2]}$.
2. $\mathcal{B}$ requests $\mathcal{C}$ for a secret key corresponding to machine $\mathsf{U}_\lambda$ and receives $\mathsf{SK_U}$.
3. $\mathcal{B}$ sends $\widetilde{M} = (\mathsf{SK_U}, (\{\mathsf{CT}_{1,j}, \ldots, \mathsf{CT}_{n,j}\}_{j\in[2]}, \mathsf{CT}_{n+1,1}))$ as the challenge obfuscation to $\mathcal{A}$ and outputs a bit $b'$ returned by $\mathcal{A}$.

This completes the description of the reduction $\mathcal{B}$. We first observe that for any $\mathbf{x} = (x_1, \ldots, x_n) \in \Sigma^n_\lambda$, since $M_0$ and $M_1$ are functionally equivalent Turing machines, we have that:

$$\mathsf{U}_\lambda(\mathbf{x}, \langle M_0 \rangle) = M_0(\mathbf{x}) = M_1(\mathbf{x}) = \mathsf{U}_\lambda(\mathbf{x}, \langle M_1 \rangle)$$

Further, $\mathcal{A}$ being a valid $\mathsf{iO}$ adversary, we have $\mathsf{runtime}(M_0, \mathbf{x}) = \mathsf{runtime}(M_1, \mathbf{x})$. Thus $\mathcal{B}$ is a valid $(\mathsf{n}+1)$-$\mathsf{TMFE}$ adversary. Hence, if the $(\mathsf{n}+1)$-$\mathsf{TMFE}$ challenger had chosen challenge bit 0, then the obfuscation $\widetilde{M}$ is of $M_0$, else of $M_1$. Thus the advantage of $\mathcal{A}$ in distinguishing the two cases translates exactly to the advantage of $\mathcal{B}$ against the $(\mathsf{n}+1)$-$\mathsf{TMFE}$ scheme. □

Acknowledgement. We thank Vinod Vaikuntanathan for suggesting the generic transformation from FE to decomposable FE.

## References

[AB08] Sanjeev Arora and Boaz Barak. Complexity theory: A modern approach. Online draft at http://www. cs. princeton. edu/theory/complexity, 2008.

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

[ACC+16] Prabhanjan Ananth, Yu-Chi Chen, Kai-Min Chung, Huijia Lin, and Wei-Kai Lin. Delegating ram computations with adaptive soundness and privacy. In Theory of Cryptography Conference, pages 3–30. Springer, 2016.

[AIK11] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011, pages 120–129, 2011.

[AIK14] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. SIAM J. Comput., 43(2):905–929, 2014.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In CRYPTO, pages 308–326. Springer, 2015.

[AJS17] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation for turing machines: Constant overhead and amortization. In Crypto. Springer, 2017.

[AS16] Prabhanjan Ananth and Amit Sahai. Functional encryption for turing machines. In Theory of Cryptography Conference (TCC), pages 125–153. Springer, 2016.

[AS17a] Shweta Agrawal and Ishaan Preet Singh. Reusable garbled deterministic finite automata from lwe. In ICALP, 2017.

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

[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.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Public Key Cryptography, pages 501–519, 2014.

[BGJS15] Saikrishna Badrinarayanan, Divya Gupta, Abhishek Jain, and Amit Sahai. Multi-input functional encryption for unbounded arity functions. In Advances in Cryptology–ASIACRYPT 2015, pages 27–51. Springer, 2015.

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In STOC, 2015.

[BKS16] Zvika Brakerski, Ilan Komargodski, and Gil Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In Eurocrypt, 2016.

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In Theory of Cryptography Conference, pages 391–418. Springer, 2016.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on, pages 1480–1498. IEEE, 2015.

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

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

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Asiacrypt. Springer, 2013.

[CCC+15] 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.

[CCHR15] Ran Canetti, Yilei Chen, Justin Holmgren, and Mariana Raykova. Succinct adaptive garbled ram. Cryptology ePrint Archive, Report 2015/1074, 2015. https://eprint.iacr.org/2015/1074.

[CH16] Ran Canetti and Justin Holmgren. Fully succinct garbled ram. In Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, pages 169–178. ACM, 2016.

[CHJV15] 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.

[CHN+16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In Proceedings of the forty-eighth annual ACM symposium on Theory of Computing, pages 1115–1127. ACM, 2016.

[CIJ+13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In CRYPTO, 2013.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Theory of Cryptography - 12th Theory of Cryptography Conference, TCC, 2015.

[CMR17] Brent Carmer, Alex J Malozemoff, and Mariana Raykova. 5gen-c: multi-input functional encryption and program obfuscation for arithmetic circuits. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 747–764. ACM, 2017.

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In EUROCRYPT, pages 578–602, 2014.

[GGH+13] 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/.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. Journal of the ACM, 33(4):792–807, 1986.

[GHRW14] Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Outsourcing private RAM computation. In 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS, 2014.

[GKP+13a] 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.

[GKP+13b] 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.

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In CRYPTO, pages 579–604. Springer, 2016.

[GPSZ16] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. Technical report, Cryptology ePrint Archive, Report 2016/102, 2016. http://eprint. iacr. org/2016/102, 2016.

[GS16] Sanjam Garg and Akshayaram Srinivasan. Single-key to multi-key functional encryption with polynomial loss. In Theory of Cryptography Conference, pages 419–442. Springer, 2016.

[Imp] Russell Impagliazzo. Notes on turing machines. http://cseweb.ucsd.edu/classes/sp11/cse201A-a/ln412.pdf.

[JSW17] Zahra Jafargholi, Alessandra Scafuro, and Daniel Wichs. Adaptively indistinguishable garbled circuits. In Theory of Cryptography Conference, pages 40–71. Springer, 2017.

[KLW15] 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.

[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.

[KNT18a] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Obfustopia built on secret-key functional encryption. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 603–648. Springer, 2018.

[KNT18b] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Simple and generic constructions of succinct functional encryption. In IACR International Workshop on Public Key Cryptography, pages 187–217. Springer, 2018.

[KNTY19] Fuyuki Kitagawa, Ryo Nishimaki, Keisuke Tanaka, and Takashi Yamakawa. Adaptively secure and succinct functional encryption: improving security and efficiency, simultaneously. In Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III 39, pages 521–551. Springer, 2019.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, CCS '13, 2013.

[KS17] Ilan Komargodski and Gil Segev. From minicrypt to obfustopia via private-key functional encryption. In EUROCRYPT, 2017.

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

[LM16] Baiyu Li and Daniele Micciancio. Compactness vs collusion resistance in functional encryption. In Theory of Cryptography Conference, pages 443–468. Springer, 2016.

[LO13] Steve Lu and Rafail Ostrovsky. How to garble ram programs? In Advances in Cryptology – EUROCRYPT, 2013.

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Output-compressing randomized encodings and applications. In TCC-A, 2016.

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

[LZ17] Qipeng Liu and Mark Zhandry. Decomposable obfuscation: A framework for building applications of obfuscation from polynomial hardness. In Theory of Cryptography Conference, pages 138–169. Springer, 2017.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. IACR Cryptology ePrint Archive, 2010:556, 2010.

[PF79] Nicholas Pippenger and Michael J Fischer. Relations among complexity measures. Journal of the ACM (JACM), 26(2):361–381, 1979.

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

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In STOC, 2014. http://eprint.iacr.org/2013/454.pdf.

# APPENDICES

## A Definitions: Turing Machines

We recall the definition of a Turing machine (TM). A TM $M$ is represented by the tuple $(Q, \Gamma, \beta, \Sigma, \delta, q_{st}, F)$ where $Q$ is a finite set of states, $\Gamma$ is a finite alphabet, $\beta \in \Gamma$ is the blank symbol, $\Sigma \subseteq \Gamma \setminus \{\beta\}$ is the set of input symbols, $\mathsf{q_{st}}$ is the start state, $F = \{q_{acc}, q_{rej}\}$ where $q_{acc} \in Q$ is the accept state, $q_{rej} \in Q$ is the reject state and $\delta : Q \setminus F \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function (stored as a table). Upon input $\mathbf{w} = (w_1, \ldots, w_k) \in \Sigma^k$ for some arbitrary polynomial $k$, the machine $M$ accepts the input if and only if given a tape initialised with the input $\mathbf{w}$ and the head at $w_1$, following the TM's transition function leads to $q_{acc}$. We say $M(\mathbf{w}) = 1$ iff $M$ accepts $\mathbf{w}$ and 0 otherwise. We also denote the runtime of a TM $M$ (i.e. number of steps the head takes) on an input $\mathbf{w}$ by $\mathsf{runtime}(M, \mathbf{w})$.

Oblivious Turing Machines

Our construction makes use of oblivious Turing machines.

Definition A.1 (Oblivious Turing Machine [Imp]). An Oblivious Turing Machine (OTM) is a Turing Machine for which there exists a function $t$ such that, at every timestep $i$ the machine head is at cell $t(i)$ regardless 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 [PF79]. Here, we describe a simple transformation that incurs a quadratic blowup in running time.

Given a TM $M$, a simple OTM construction adds an additional marker for the head location. Now, to simulate step $i$ in the TM, the OTM, scans from cell 1 to cell $i$, ensuring that it reads the current head location. Now, it moves back from cell $i$ to 1, writing the correct symbol for the next step, while also updating the state. Once back at cell 1, simulation of step $i$ is complete, and the OTM moves to a state simulate $q_{i+1}$ and if $q_{i+1}$ is not an accepting or rejecting state, it moves to simulating step $i+1$. Since in step $i$, we would need to scan at most $i$ cells (as that is the farthermost the head could have moved), a $O(t)$ computation, now takes $O(t^2)$. Also, if we are willing to reveal the runtime of the given input on the Turing Machine, then we can stop simulating after the last timestep $t$. A more efficient transformation due to Pippenger-Fischer[PF79] reduces the time required to $O(t \log t)$.

We note that a slightly different definition of OTMs [AB08] requires that the head movements are the same for all inputs of the same size, which would imply that the OTM runs in worst case time. However, if we are willing to reveal the running time of a machine on a given input, then the OTM can be made to halt once the input has been decided. In particular, if $\mathsf{runtime}(M_1, \mathbf{w}_1) = \mathsf{runtime}(M_2, \mathbf{w}_2)$, then the head movements of the OTM corresponding to $M_1$ and the OTM corresponding to $M_2$ are exactly the same.

In our construction, the OTM will be provided the input length of the message as an explicit input, and can use this to compute the head movements at any given time step.

# B Missing Details in Proof of Theorem 3.1

Proof. In the following we argue that consecutive hybrids as defined in Section 3.3 are indistinguishable.

Claim B.1. If $\mathsf{1FE_1}$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(0)$ and $\mathcal{H}(1,1)$ are indistinguishable.

Proof. Given a $\mathsf{PPT}$ adversary $\mathcal{A}$ that distinguishes $\mathcal{H}(0)$ and $\mathcal{H}(1,1)$, we construct another $\mathsf{PPT}$ adversary $\mathcal{B}$ who breaks the security of the $\mathsf{1FE_1}$ scheme as follows.

1. $\mathcal{B}$ receives $\mathsf{1FE_1.PK}$ from the $\mathsf{1FE_1}$ challenger and returns this to $\mathcal{A}$. Additionally, it samples by itself $(\mathsf{1FE_2.PK}, \mathsf{1FE_2.MSK}) \leftarrow \mathsf{1FE_2.Setup}(1^\lambda), \mathsf{salt} \leftarrow \{0,1\}^\lambda$ and two random strings $\mathsf{ct}_1, \mathsf{ct}_2 \leftarrow \mathcal{C}^{\mathsf{SKE}}$, where $\mathcal{C}^{\mathsf{SKE}}$ denotes the ciphertext space of the $\mathsf{SKE}$ scheme.

2. When $\mathcal{A}$ outputs a pair of challenge distributions $(\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ with support $\Sigma^\ell$ for any arbitrary $\ell = \mathsf{poly}(\lambda)$ and a function query $M$ obeying the admissibility criteria that $\forall b \in \{0,1\}, \mathbf{w}_b \leftarrow \mathcal{D}_b^\ell$, $\mathsf{runtime}(M, \mathbf{w}_0) = \mathsf{runtime}(M, \mathbf{w}_1)$ and $M(\mathbf{w}_0) \overset{c}{\approx} M(\mathbf{w}_1)$, $\mathcal{B}$ does the following.

   (a) To simulate the challenge ciphertext, it first samples a pair of challenge messages $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow (\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ such that $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$. It also samples a root cPRF key $\mathsf{K}_0 \leftarrow \mathsf{F.Setup}(1^\lambda)$.

   (b) $\mathcal{B}$ executes the oblivious TM $M$ on $\mathbf{w}_0$ to learn the (symbol, state) pairs $(\sigma^0_{T-1}, \mathsf{q}^0_{T-1})$ and $(\sigma^0_{T-2}, \mathsf{q}^0_{T-2})$ at time steps $T-1$ and $T-2$ respectively. It also records the time steps $(T', T-2)$ and $(T'', T-3)$ when the individual components of these two (symbol,

state) pairs are generated and then prepares a new pair of challenge distributions $(\widehat{\mathcal{D}}_0^\ell, \widehat{\mathcal{D}}_1^\ell)$ for the $\mathsf{1FE_1}$ challenger as follows.

i. For $b = 0$, $\widehat{\mathcal{D}}_0^\ell = \{\mathbf{x}_0 = (x_{0,1}, \ldots, x_{0,\ell})\}$, where $\forall i \in [\ell]\ x_{0,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^0)$, with $\mathsf{Trap}^0.\mathsf{mode\text{-}real} = 1$ and all other fields set to $\perp$.

ii. For $b = 1$, $\widehat{\mathcal{D}}_1^\ell = \{\mathbf{x}_1 = (x_{1,1}, \ldots, x_{1,\ell})\}$, where $\forall i \in [\ell]\ x_{1,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^1)$, with the modified fields in $\mathsf{Trap}^1$ as shown in Figure 11.

| | | | | | |
|---|---|---|---|---|---|
| $\mathsf{mode\text{-}real} : \perp$ | $\mathsf{key\text{-}id} : \mathsf{salt}$ | $\mathsf{val}_0 : w_{0,i}$ | $\mathsf{val}_1 : w_{1,i}$ | $\mathsf{SKE.K} : \perp$ | $\perp$ |
| $\mathsf{mode\text{-}trap}_1 : 1$ | $\mathsf{Target\ TS}_1 : T-1$ | $\mathsf{Sym\ TS}_1 : T'$ | $\mathsf{Sym\ val}_1 : \sigma^0_{T-1}$ | $\mathsf{ST\ TS}_1 : T-2$ | $\mathsf{ST\ val}_1 : \mathsf{q}^0_{T-1}$ |
| $\mathsf{mode\text{-}trap}_2 : 1$ | $\mathsf{Target\ TS}_2 : T-2$ | $\mathsf{Sym\ TS}_2 : T''$ | $\mathsf{Sym\ val}_2 : \sigma^0_{T-2}$ | $\mathsf{ST\ TS}_2 : T-3$ | $\mathsf{ST\ val}_2 : \mathsf{q}^0_{T-2}$ |
| $\mathsf{mode\text{-}trap}_3 : \perp$ | $\mathsf{Target\ TS} : \perp$ | $\mathsf{Sym\ TS} : \perp$ | $\perp$ | $\mathsf{ST\ TS} : \perp$ | $\perp$ |

Figure 11: $\mathsf{Trap}^1$ configuration in $\mathcal{H}(1,1)$

(c) It sends the distribution pair to the $\mathsf{1FE_1}$ challenger and relays the response back to $\mathcal{A}$.

(d) To simulate a function key for $M$, $\mathcal{B}$ first requests for a function key to the $\mathsf{1FE_1}$ challenger for the function $\mathsf{ReRand}_{\mathsf{1FE_2.PK},\mathsf{salt},\mathsf{q_{st}},\perp,\perp}$ and receives $\mathsf{SK}_{\mathsf{ReRand}}$. $\mathcal{B}$ computes by itself $\mathsf{SK}_{\mathsf{Next}} \leftarrow \mathsf{1FE_2.KeyGen}(\mathsf{1FE_2.MSK}, \mathsf{Next}_{\mathsf{1FE_2.PK},\mathsf{salt},M,\mathsf{ct}_1,\mathsf{ct}_2})$ and returns a function key for $M$ as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a guess, $\mathcal{B}$ does the same.

Observe that for all time steps $t \notin \{T', T-2, T'', T-3\}$, the decryption outputs are exactly the same ciphertexts in both the $\mathcal{H}(0)$ and $\mathcal{H}(1,1)$, since these ciphertexts are computed according to the real world functionality of $\mathsf{Next}$. At a time step $t \in \{T', T-2, T'', T-3\}$ in $\mathcal{H}(1,1)$, the decryption mimics the real world decryption of $\mathcal{H}(0)$ due to the execution paths in the $\mathsf{Next}$ function conditioned on $\mathsf{Trap}^1.\mathsf{mode\text{-}trap}_1 = \mathsf{Trap}^1.\mathsf{mode\text{-}trap}_2 = 1$. Therefore, $\mathcal{B}$ is an admissible adversary against the $\mathsf{1FE_1}$ challenger since the outputs for the two challenge message sets are exactly the same. If $b = 0$, $\mathcal{A}$ sees the distribution of $\mathcal{H}(0)$, while if $b = 1$, $\mathcal{A}$ sees the distribution of $\mathcal{H}(1,1)$. Thus the advantage of $\mathcal{A}$ translates to the advantage of $\mathcal{B}$. □

Claim B.2. If $\mathsf{SKE}$ is a secure symmetric-key encryption scheme, then hybrids $\mathcal{H}(1,1)$ and $\mathcal{H}(1,2)$ are indistinguishable.

Proof. Given a $\mathsf{PPT}$ adversary $\mathcal{A}$ that distinguishes $\mathcal{H}(1,1)$ and $\mathcal{H}(1,2)$, we construct another $\mathsf{PPT}$ adversary $\mathcal{B}$ who breaks the security of the $\mathsf{SKE}$ scheme as follows.

1. $\mathcal{B}$ samples $(\mathsf{1FE_1.PK}, \mathsf{1FE_1.MSK}) \leftarrow \mathsf{1FE_1.Setup}(1^\lambda)$, $(\mathsf{1FE_2.PK}, \mathsf{1FE_2.MSK}) \leftarrow \mathsf{1FE_2.Setup}(1^\lambda)$ and $\mathsf{salt} \leftarrow \{0,1\}^\lambda$. It sends $\mathsf{PK} = \mathsf{1FE_1.PK}$ to $\mathcal{A}$.

2. When $\mathcal{A}$ outputs a pair of challenge distributions $(\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ with support $\Sigma^\ell$ for any arbitrary $\ell = \mathsf{poly}(\lambda)$ and a function query $M$ obeying the admissibility criteria that $\forall b \in \{0,1\}, \mathbf{w}_b \leftarrow \mathcal{D}_b^\ell$, $\mathsf{runtime}(M, \mathbf{w}_0) = \mathsf{runtime}(M, \mathbf{w}_1)$ and $M(\mathbf{w}_0) \overset{c}{\approx} M(\mathbf{w}_1)$, $\mathcal{B}$ does the following.

(a) To simulate the challenge ciphertext, it first samples a pair of challenge messages $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow (\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ such that $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$. It also samples a root cPRF key $\mathsf{K}_0 \leftarrow \mathsf{F.Setup}(1^\lambda)$.

(b) $\mathcal{B}$ executes the oblivious TM $M$ on $\mathbf{w}_0$ to learn the (symbol, state) pairs $(\sigma^0_{T-1}, \mathsf{q}^0_{T-1})$ and $(\sigma^0_{T-2}, \mathsf{q}^0_{T-2})$ at time steps $T-1$ and $T-2$ respectively. It also records the time steps $(T', T-2)$ and $(T'', T-3)$ when the individual components of these two (symbol, state) pairs are generated. It then simulates the encryption oracle by computing $\mathsf{CT}_i = \mathsf{1FE_1.Enc}(\mathsf{1FE_1.PK}, x_{1,i})$, where $\forall i \in [\ell], x_{1,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^1)$ and $\mathsf{Trap}^1$ is as per Figure 11. It returns the ciphertext $\mathsf{CT} = \{\mathsf{CT}_i\}_{i\in[\ell]}$ to $\mathcal{A}$.

(c) To simulate a function key for $M$, $\mathcal{B}$ does the following.

i. It first computes $\mathsf{SK}_{\mathsf{ReRand}} \leftarrow \mathsf{1FE_1.KeyGen}(\mathsf{1FE_1.MSK}, \mathsf{ReRand}_{\mathsf{1FE_2.PK},\mathsf{salt},\mathsf{q_{st}},\perp,\perp})$.

ii. It then computes $\mathsf{1FE_2}$ encodings of $(\sigma^0_{T-1}, \mathsf{q}^0_{T-1})$ as follows.

- Compute a delegated cPRF key $\mathsf{K}_T = \mathsf{F.KeyDel}(\mathsf{K}_0, f_T)$ and generate the encryption randomness for time step $T-1$ as $r_{T-1} = \mathsf{F.Eval}(\mathsf{K}_0, (T-1\|\mathsf{salt}))$.
- Compute the $\mathsf{1FE_2}$ symbol ciphertext to be given as output at time step $T'$ for the future time step $T-1$ as $\mathsf{CT}^0_{\mathsf{sym},T-1} = \mathsf{1FE_2.Enc}(\mathsf{1FE_2.PK_1}, \mathbf{z}^0_1; r_{T-1})$, where $\mathbf{z}^0_1 = (\mathsf{SYM}, \mathsf{salt}, \mathsf{K}_T, T-1, \ell, \sigma^0_{T-1}, \mathsf{Trap}^1)$ and $\mathsf{Trap}^1$ is as per Figure 11.
- Compute the $\mathsf{1FE_2}$ state ciphertext to be given as output at time step $T-2$ for the future time step $T-1$ as $\mathsf{CT}^0_{\mathsf{st},T-1} = \mathsf{1FE_2.Enc}(\mathsf{1FE_2.PK_2}, \mathbf{z}^0_2; r_{T-1})$, where $\mathbf{z}^0_2 = (\mathsf{ST}, \mathsf{q}^0_{T-1})$.

iii. It sends the $\mathsf{1FE_2}$ ciphertexts $\mathsf{CT}^0_{\mathsf{sym},T-1}, \mathsf{CT}^0_{\mathsf{st},T-1}$ to the challenger of the $\mathsf{SKE}$ scheme and gets back $\mathsf{ct}_1, \mathsf{ct}_2$.

iv. $\mathcal{B}$ then computes $\mathsf{SK}_{\mathsf{Next}} \leftarrow \mathsf{1FE_2.KeyGen}(\mathsf{1FE_2.MSK}, \mathsf{Next}_{\mathsf{1FE_2.PK},\mathsf{salt},M,\mathsf{ct}_1,\mathsf{ct}_2})$ and returns a function key for $M$ as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a guess, $\mathcal{B}$ does the same.

Note that the only difference between the two hybrids is that the $\mathsf{SKE}$ encryptions programmed in the function key is random in $\mathcal{H}(1,1)$ and are valid $\mathsf{SKE}$ encryptions of $(\mathsf{CT}^0_{\mathsf{sym},T-1}, \mathsf{CT}^0_{\mathsf{st},T-1})$ encoding the (symbol, state) pair for time step $T-1$ in $\mathcal{H}(1,2)$. Hence the advantage of an adversary who distinguishes between the two hybrids translates to an advantage of an adversary against the $\mathsf{SKE}$ scheme. □

Claim B.3. If $\mathsf{1FE_1}$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(1,2)$ and $\mathcal{H}(1,3)$ are indistinguishable.

Proof. Given a $\mathsf{PPT}$ adversary $\mathcal{A}$ that distinguishes $\mathcal{H}(1,2)$ and $(1,3)$, we construct another $\mathsf{PPT}$ adversary $\mathcal{B}$ who breaks the security of the $\mathsf{1FE_1}$ scheme as follows.

1. $\mathcal{B}$ receives $\mathsf{1FE_1.PK}$ from the $\mathsf{1FE_1}$ challenger and returns this to $\mathcal{A}$. Additionally, it samples by itself $(\mathsf{1FE_2.PK}, \mathsf{1FE_2.MSK}) \leftarrow \mathsf{1FE_2.Setup}(1^\lambda), \mathsf{salt} \leftarrow \{0,1\}^\lambda$ and a key $\mathsf{K} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$.

2. When $\mathcal{A}$ outputs a pair of challenge distributions $(\mathcal{D}^\ell_0, \mathcal{D}^\ell_1)$ with support $\Sigma^\ell$ for any arbitrary $\ell = \mathsf{poly}(\lambda)$ and a function query $M$ obeying the admissibility criteria that $\forall b \in \{0,1\}, \mathbf{w}_b \leftarrow \mathcal{D}^\ell_b$, $\mathsf{runtime}(M, \mathbf{w}_0) = \mathsf{runtime}(M, \mathbf{w}_1)$ and $M(\mathbf{w}_0) \overset{c}{\approx} M(\mathbf{w}_1)$, $\mathcal{B}$ does the following.

(a) To simulate the challenge ciphertext, it first samples a pair of challenge messages $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow (\mathcal{D}^\ell_0, \mathcal{D}^\ell_1)$ such that $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$. It also samples a root cPRF key $\mathsf{K}_0 \leftarrow \mathsf{F.Setup}(1^\lambda)$.

(b) $\mathcal{B}$ executes the oblivious TM $M$ on $\mathbf{w}_0$ to learn the (symbol, state) pairs $(\sigma^0_{T-1}, \mathsf{q}^0_{T-1})$ and $(\sigma^0_{T-2}, \mathsf{q}^0_{T-2})$ at time steps $T-1$ and $T-2$ respectively. It also records the time steps $(T', T-2)$ and $(T'', T-3)$ when the individual components of these two (symbol, state) pairs are generated and then prepares a new pair of challenge distributions $(\widehat{\mathcal{D}}^\ell_0, \widehat{\mathcal{D}}^\ell_1)$ for the $\mathsf{1FE_1}$ challenger as follows.

i. For $b=0$, $\widehat{\mathcal{D}}^\ell_0 = \{\mathbf{x}_0 = (x_{0,1}, \ldots, x_{0,\ell})\}$, where $\forall i \in [\ell]\ x_{0,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^0)$ with the fields of $\mathsf{Trap}^0$ being same as that of in $\mathsf{Trap}^1$ in $\mathcal{H}(1,2)$ as per Figure 11.

ii. For $b=1$, $\widehat{\mathcal{D}}^\ell_1 = \{\mathbf{x}_1 = (x_{1,1}, \ldots, x_{1,\ell})\}$, where $\forall i \in [\ell]\ x_{1,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^1)$, with the modified fields in $\mathsf{Trap}^1$ as shown in Figure 12.

| | | | | | |
|---|---|---|---|---|---|
| mode-real : $\perp$ | key-id : salt | $\mathsf{val}_0 : w_{0,i}$ | $\mathsf{val}_1 : w_{1,i}$ | SKE.K : K | $\perp$ |
| $\mathsf{mode\text{-}trap}_1 : \perp$ | Target $\mathsf{TS}_1 : \perp$ | Sym $\mathsf{TS}_1 : \perp$ | Sym $\mathsf{val}_1 : \perp$ | ST $\mathsf{TS}_1 : \perp$ | ST $\mathsf{val}_1 : \perp$ |
| $\mathsf{mode\text{-}trap}_2 : 1$ | Target $\mathsf{TS}_2 : T-2$ | Sym $\mathsf{TS}_2 : T''$ | Sym $\mathsf{val}_2 : \sigma^0_{T-2}$ | ST $\mathsf{TS}_2 : T-3$ | ST $\mathsf{val}_2 : \mathsf{q}^0_{T-2}$ |
| $\mathsf{mode\text{-}trap}_3 : 1$ | Target TS : $T-1$ | Sym TS : $T'$ | $\perp$ | ST TS : $T-2$ | $\perp$ |

Figure 12: $\mathsf{Trap}^1$ configuration in $\mathcal{H}(1,3)$

(c) It sends the distribution pair to the $\mathsf{1FE_1}$ challenger and relays the response back to $\mathcal{A}$.

(d) To simulate a function key for $M$, $\mathcal{B}$ does the following.

i. It requests for a function key for $\mathsf{ReRand}_{\mathsf{1FE_2.PK},\mathsf{salt},\mathsf{q_{st}},\perp,\perp}$ to the $\mathsf{1FE_1}$ challenger and receives $\mathsf{SK}_{\mathsf{ReRand}}$.

ii. It then computes $\mathsf{1FE_2}$ encodings of $(\sigma^0_{T-1}, \mathsf{q}^0_{T-1})$ as follows.

- Compute a delegated cPRF key $\mathsf{K}_T = \mathsf{F.KeyDel}(\mathsf{K}_0, f_T)$ and generate the encryption randomness for time step $T-1$ as $r_{T-1} = \mathsf{F.Eval}(\mathsf{K}_0, (T-1 \| \mathsf{salt}))$.
- Compute the $\mathsf{1FE_2}$ symbol ciphertext to be given as output at time step $T'$ for the future time step $T-1$ as $\mathsf{CT}^0_{\mathsf{sym},T-1} = \mathsf{1FE_2.Enc}(\mathsf{1FE_2.PK}_1, \mathbf{z}^0_1; r_{T-1})$, where $\mathbf{z}^0_1 = (\mathsf{SYM}, \mathsf{salt}, \mathsf{K}_T, T-1, \ell, \sigma^0_{T-1}, \mathsf{Trap}^1)$ and $\mathsf{Trap}^1$ is as per Figure 12 now.
- Compute the $\mathsf{1FE_2}$ state ciphertext to be given as output at time step $T-2$ for the future time step $T-1$ as $\mathsf{CT}^0_{\mathsf{st},T-1} = \mathsf{1FE_2.Enc}(\mathsf{1FE_2.PK}_2, \mathbf{z}^0_2; r_{T-1})$, where $\mathbf{z}^0_2 = (\mathsf{ST}, \mathsf{q}^0_{T-1})$.

iii. Once it has generated the two $\mathsf{1FE_2}$ ciphertexts $\mathsf{CT}^0_{\mathsf{sym},T-1}$ and $\mathsf{CT}^0_{\mathsf{st},T-1}$, it computes two SKE ciphertexts $\mathsf{ct}_1 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}^0_{\mathsf{sym},T-1})$ and $\mathsf{ct}_2 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}^0_{\mathsf{st},T-1})$.

iv. Finally, it computes $\mathsf{SK}_{\mathsf{Next}} \leftarrow \mathsf{1FE_2.KeyGen}(\mathsf{1FE_2.MSK}, \mathsf{Next}_{\mathsf{1FE_2.PK},\mathsf{salt},M,\mathsf{ct}_1,\mathsf{ct}_2})$ and returns a function key for $M$ as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a guess, $\mathcal{B}$ does the same.

Observe that for all time steps $t \notin \{T', T-2\}$, the decryption outputs are exactly the same ciphertexts in both $\mathcal{H}(1,2)$ and $\mathcal{H}(1,3)$. At time step $t \in \{T', T-2\}$ in $\mathcal{H}(1,2)$, $\mathsf{Trap}^0.\mathsf{mode\text{-}trap}_1 = 1$ (in Figure 11) dictates the decryption to output two decomposed components of a single $\mathsf{1FE_2}$ ciphertext, one component encoding $\mathsf{Trap}^0.\mathsf{Sym\ val}_1 = \sigma^0_{T-1}$ at time step $T'$ and the other encoding $\mathsf{Trap}^0.\mathsf{ST\ val}_1 = \mathsf{q}^0_{T-1}$ at time step $T-2$. Alternatively in $\mathcal{H}(1,3)$, $\mathsf{Trap}^1.\mathsf{mode\text{-}trap}_3 = 1$ (in Figure 12) dictates the decryption to firstly use $\mathsf{Trap}^1.\mathsf{SKE.K} = \mathsf{K}$ to

decrypt the hardwired ciphertext $\mathsf{ct}_1$ and output $\mathsf{CT}^0_{\mathsf{sym},T-1}$ at time step $T'$ (respectively, $\mathsf{ct}_2$ and output $\mathsf{CT}^0_{\mathsf{st},T-1}$ at time step $T-2$). In both the hybrids, these symbol and state ciphertext pieces are computed for target time step $T-1$. Thus $\mathcal{B}$ is an admissible $\mathsf{1FE}_1$ adversary. If $b=0$, $\mathcal{A}$ sees the distribution of $\mathcal{H}(1,2)$, while if $b=1$, $\mathcal{A}$ sees the distribution of $\mathcal{H}(1,3)$. Hence the advantage of $\mathcal{A}$ translates to the advantage of $\mathcal{B}$. □

Claim B.4. If $\mathsf{1FE}_1$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(1,3)$ and $\mathcal{H}(1,4)$ are indistinguishable.

Proof. Given a $\mathsf{PPT}$ adversary $\mathcal{A}$ that distinguishes $\mathcal{H}(1,3)$ and $\mathcal{H}(1,4)$, we construct another $\mathsf{PPT}$ adversary $\mathcal{B}$ who breaks the security of the $\mathsf{1FE}_1$ scheme as follows.

1. $\mathcal{B}$ receives $\mathsf{1FE}_1.\mathsf{PK}$ from the $\mathsf{1FE}_1$ challenger and returns this to $\mathcal{A}$. Additionally, it samples by itself $(\mathsf{1FE}_2.\mathsf{PK}, \mathsf{1FE}_2.\mathsf{MSK}) \leftarrow \mathsf{1FE}_2.\mathsf{Setup}(1^\lambda), \mathsf{salt} \leftarrow \{0,1\}^\lambda$ and a key $\mathsf{K} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$.
2. When $\mathcal{A}$ outputs a pair of challenge distributions $(\mathcal{D}^\ell_0, \mathcal{D}^\ell_1)$ with support $\Sigma^\ell$ for any arbitrary $\ell = \mathsf{poly}(\lambda)$ and a function query $M$ obeying the admissibility criteria that $\forall b \in \{0,1\}, \mathbf{w}_b \leftarrow \mathcal{D}^\ell_b$, $\mathsf{runtime}(M, \mathbf{w}_0) = \mathsf{runtime}(M, \mathbf{w}_1)$ and $M(\mathbf{w}_0) \overset{c}{\approx} M(\mathbf{w}_1)$, $\mathcal{B}$ does the following.
   (a) To simulate the challenge ciphertext, it first samples a pair of challenge messages $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow (\mathcal{D}^\ell_0, \mathcal{D}^\ell_1)$ such that $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$. It also samples a root cPRF key $\mathsf{K}_0 \leftarrow \mathsf{F.Setup}(1^\lambda)$.
   (b) $\mathcal{B}$ executes the oblivious TM $M$ on $\mathbf{w}_0$ to learn the (symbol, state) pairs $(\sigma^0_{T-1}, \mathsf{q}^0_{T-1})$ and $(\sigma^0_{T-2}, \mathsf{q}^0_{T-2})$ at time steps $T-1$ and $T-2$ respectively. It also records the time steps $(T', T-2)$ and $(T'', T-3)$ when the individual components of these two (symbol, state) pairs are generated. It then computes a root key punctured at point $(T-1\|\mathsf{salt})$ as $\mathsf{K}^{T-1}_0 = \mathsf{F.Constrain}(\mathsf{K}_0, (T-1\|\mathsf{salt}))$ and then prepares a new pair of challenge distributions $(\widehat{\mathcal{D}}^\ell_0, \widehat{\mathcal{D}}^\ell_1)$ for the $\mathsf{1FE}_1$ challenger as follows.
      i. For $b=0$, $\widehat{\mathcal{D}}^\ell_0 = \{\mathbf{x}_0 = (x_{0,1}, \ldots, x_{0,\ell})\}$, where $\forall i \in [\ell]$ $x_{0,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^1)$ with the fields of $\mathsf{Trap}^1$ being same as that of in $\mathsf{Trap}^1$ in $\mathcal{H}(1,3)$ as per Figure 12.
      ii. For $b=1$, $\widehat{\mathcal{D}}^\ell_1 = \{\mathbf{x}_1 = (x_{1,1}, \ldots, x_{1,\ell})\}$, where $\forall i \in [\ell]$ $x_{1,i} = (\mathsf{K}^{T-1}_0, i, \ell, w_{0,i}, \mathsf{Trap}^1)$, with $\mathsf{Trap}^1$ as per Figure 12.
   (c) It sends the distribution pair to the $\mathsf{1FE}_1$ challenger and relays the response back to $\mathcal{A}$.
   (d) To simulate a function key for $M$, $\mathcal{B}$ does the following.
      i. It requests for a function key for $\mathsf{ReRand}_{\mathsf{1FE}_2.\mathsf{PK},\mathsf{salt},\mathsf{q}_{\mathsf{st}},\perp,\perp}$ to the $\mathsf{1FE}_1$ challenger and receives $\mathsf{SK}_{\mathsf{ReRand}}$.
      ii. It then computes $\mathsf{1FE}_2$ encodings of $(\sigma^0_{T-1}, \mathsf{q}^0_{T-1})$, as follows.
         - Compute a punctured, delegated key $\mathsf{K}^{T-1}_T = \mathsf{F.KeyDel}(\mathsf{K}^{T-1}_0, f_T)$ and generate the encryption randomness for time step $T-1$ as $r_{T-1} = \mathsf{F.Eval}(\mathsf{K}_0, (T-1\|\mathsf{salt}))$.
         - Compute the $\mathsf{1FE}_2$ symbol ciphertext to be given as output at time step $T'$ for the future time step $T-1$ as $\mathsf{CT}^0_{\mathsf{sym},T-1} = \mathsf{1FE}_2.\mathsf{Enc}(\mathsf{1FE}_2.\mathsf{PK}_1, \mathbf{z}^0_1; r_{T-1})$, where $\mathbf{z}^0_1 = (\mathsf{SYM}, \mathsf{salt}, \mathsf{K}^{T-1}_T, T-1, \ell, \sigma^0_{T-1}, \mathsf{Trap}^1)$ and $\mathsf{Trap}^1$ is as per Figure 12.

- Compute the $\mathsf{1FE}_2$ state ciphertext to be given as output at time step $T-2$ for future time step $T-1$ as $\mathsf{CT}^0_{\mathsf{st},T-1} = \mathsf{1FE}_2.\mathsf{Enc}(\mathsf{1FE}_2.\mathsf{PK}_2, \mathbf{z}^0_2; r_{T-1})$, where $\mathbf{z}^0_2 = (\mathsf{ST}, \mathsf{q}^0_{T-1})$.

iii. Once it has generated the two $\mathsf{1FE}_2$ ciphertexts $\mathsf{CT}^0_{\mathsf{sym},T-1}$ and $\mathsf{CT}^0_{\mathsf{st},T-1}$, it computes two $\mathsf{SKE}$ ciphertexts $\mathsf{ct}_1 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}^0_{\mathsf{sym},T-1})$ and $\mathsf{ct}_2 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}^0_{\mathsf{st},T-1})$.

iv. Finally, it computes $\mathsf{SK}_{\mathsf{Next}} \leftarrow \mathsf{1FE}_2.\mathsf{KeyGen}(\mathsf{1FE}_2.\mathsf{MSK}, \mathsf{Next}_{\mathsf{1FE}_2.\mathsf{PK},\mathsf{salt},M,\mathsf{ct}_1,\mathsf{ct}_2})$ and returns a function key for $M$ as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a guess, $\mathcal{B}$ does the same.

Note that the only difference in $\mathcal{H}(1,3)$ and $\mathcal{H}(1,4)$ is the replacement of the root cPRF key $\mathsf{K}_0$ with a punctured root key $\mathsf{K}_0^{T-1}$ at point $(T-1\|\mathsf{salt})$ in time step $T-1$ in the $\mathsf{1FE}_1$ ciphertext. Moreover, in both the hybrids, the field $\mathsf{Trap}^1.\mathsf{mode\text{-}trap}_3 = 1$ dictates the output at time step $t \in \{T', T-2\}$ to be a ciphertext component for time step $T-1$ as argued in Claim B.3. Thus, the cPRF key is only required to compute randomness at points $\neq (T-1\|\mathsf{salt})$ for which the punctured root key suffices. Further, it evaluates to the same value as the normal key on all such points in both the hybrids. As a consequence, the decryption values are exactly the same for all the time steps proving the admissibility of $\mathcal{B}$. Thus if $b=0$, $\mathcal{A}$ sees the distribution of $\mathcal{H}(1,3)$, while if $b=1$, $\mathcal{A}$ sees the distribution of $\mathcal{H}(1,4)$. Hence the advantage of $\mathcal{A}$ translates to the advantage of $\mathcal{B}$. □

Claim B.5. If $\mathsf{F}$ is a secure punctured, delegatable cPRF scheme, then hybrids $\mathcal{H}(1,4)$ and $\mathcal{H}(1,5)$ are indistinguishable.

Proof. Given a $\mathsf{PPT}$ adversary $\mathcal{A}$ that distinguishes $\mathcal{H}(1,4)$ and $\mathcal{H}(1,5)$, we construct another $\mathsf{PPT}$ adversary $\mathcal{B}$ who breaks the security of the punctured, delegatable cPRF scheme $\mathsf{F}$ as follows.

1. $\mathcal{B}$ samples $(\mathsf{1FE}_1.\mathsf{PK}, \mathsf{1FE}_1.\mathsf{MSK}) \leftarrow \mathsf{1FE}_1.\mathsf{Setup}(1^\lambda)$, $(\mathsf{1FE}_2.\mathsf{PK}, \mathsf{1FE}_2.\mathsf{MSK}) \leftarrow \mathsf{1FE}_2.\mathsf{Setup}(1^\lambda)$, $\mathsf{salt} \leftarrow \{0,1\}^\lambda$ and $\mathsf{K} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$. It sends $\mathsf{PK} = \mathsf{1FE}_1.\mathsf{PK}$ to $\mathcal{A}$.

2. When $\mathcal{A}$ outputs a pair of challenge distributions $(\mathcal{D}^\ell_0, \mathcal{D}^\ell_1)$ with support $\Sigma^\ell$ for any arbitrary $\ell = \mathsf{poly}(\lambda)$ and a function query $M$ obeying the admissibility criteria that $\forall b \in \{0,1\}, \mathbf{w}_b \leftarrow \mathcal{D}^\ell_b$, $\mathsf{runtime}(M, \mathbf{w}_0) = \mathsf{runtime}(M, \mathbf{w}_1)$ and $M(\mathbf{w}_0) \overset{c}{\approx} M(\mathbf{w}_1)$, $\mathcal{B}$ does the following.

   (a) To simulate the challenge ciphertext, it first samples a pair of challenge messages $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow (\mathcal{D}^\ell_0, \mathcal{D}^\ell_1)$ such that $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$. It receives $\mathsf{K}_0^{T-1}$ on querying for a punctured key at the point $(T-1\|\mathsf{salt})$ to the cPRF challenger for $\mathsf{F}$.

   (b) $\mathcal{B}$ executes the oblivious TM $M$ on $\mathbf{w}_0$ to learn the (symbol, state) pairs $(\sigma^0_{T-1}, \mathsf{q}^0_{T-1})$ and $(\sigma^0_{T-2}, \mathsf{q}^0_{T-2})$ at time steps $T-1$ and $T-2$ respectively. It also records the time steps $(T', T-2)$ and $(T'', T-3)$ when the individual components of these two (symbol, state) pairs are generated. It then simulates the encryption oracle by computing $\mathsf{CT}_i = \mathsf{1FE}_1.\mathsf{Enc}(\mathsf{1FE}_1.\mathsf{PK}, x_{1,i})$, where $\forall i \in [\ell], x_{1,i} = (\mathsf{K}_0^{T-1}, i, \ell, w_{0,i}, \mathsf{Trap}^1)$ and $\mathsf{Trap}^1$ is as per Figure 12. It returns the ciphertext $\mathsf{CT} = \{\mathsf{CT}_i\}_{i\in[\ell]}$ to $\mathcal{A}$.

   (c) To simulate a function key for $M$, $\mathcal{B}$ does the following.

      i. It first computes $\mathsf{SK}_{\mathsf{ReRand}} \leftarrow \mathsf{1FE}_1.\mathsf{KeyGen}(\mathsf{1FE}_1.\mathsf{MSK}, \mathsf{ReRand}_{\mathsf{1FE}_2.\mathsf{PK},\mathsf{salt},\mathsf{q}_{\mathsf{st}},\perp,\perp})$.

ii. It then computes $\mathsf{1FE}_2$ encodings of $(\sigma^0_{T-1}, \mathsf{q}^0_{T-1})$, as follows.

- Compute a delegated key from the punctured root key as $\mathsf{K}^{T-1}_T = \mathsf{F.KeyDel}(\mathsf{K}^{T-1}_0, f_T)$.
- Query the cPRF challenger at point $(T-1\|\mathsf{salt})$ to receive an encryption randomness $R_E$ for time step $T-1$.
- Compute the $\mathsf{1FE}_2$ symbol ciphertext to be given as output at time step $T'$ for the future time step $T-1$ as $\mathsf{CT}^0_{\mathsf{sym},T-1} = \mathsf{1FE}_2.\mathsf{Enc}(\mathsf{1FE}_2.\mathsf{PK}_1, \mathbf{z}^0_1; R_E)$, where $\mathbf{z}^0_1 = (\mathsf{SYM}, \mathsf{salt}, \mathsf{K}^{T-1}_T, T-1, \ell, \sigma^0_{T-1}, \mathsf{Trap}^1)$ and $\mathsf{Trap}^1$ is as per Figure 12.
- Compute the $\mathsf{1FE}_2$ state ciphertext to be given as output at time step $T-2$ for the future time step $T-1$ as $\mathsf{CT}^0_{\mathsf{st},T-1} = \mathsf{1FE}_2.\mathsf{Enc}(\mathsf{1FE}_2.\mathsf{PK}_2, \mathbf{z}^0_2; R_E)$, where $\mathbf{z}^0_2 = (\mathsf{ST}, \mathsf{q}^0_{T-1})$.

iii. Once it has generated the two $\mathsf{1FE}_2$ ciphertexts $\mathsf{CT}^0_{\mathsf{sym},T-1}$ and $\mathsf{CT}^0_{\mathsf{st},T-1}$, it computes two $\mathsf{SKE}$ ciphertexts $\mathsf{ct}_1 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}^0_{\mathsf{sym},T-1})$ and $\mathsf{ct}_2 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}^0_{\mathsf{st},T-1})$.

iv. Finally, it computes $\mathsf{SK}_{\mathsf{Next}} \leftarrow \mathsf{1FE}_2.\mathsf{KeyGen}(\mathsf{1FE}_2.\mathsf{MSK}, \mathsf{Next}_{\mathsf{1FE}_2.\mathsf{PK},\mathsf{salt},M,\mathsf{ct}_1,\mathsf{ct}_2})$ and returns a function key for $M$ as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a guess, $\mathcal{B}$ does the same.

Note that when $R_E$ is computed using $\mathsf{K}_0$ as a pseudorandom value, $\mathcal{A}$'s view is identical to that of $\mathcal{H}(1,4)$, and when $R_E$ is sampled uniformly at random, $\mathcal{A}$'s view is identical to that of $\mathcal{H}(1,5)$. Hence the advantage of $\mathcal{A}$ in distinguishing $\mathcal{H}(1,4)$ and $\mathcal{H}(1,5)$ translates to the advantage of $\mathcal{B}$ in breaking the security of the punctured, delegatable cPRF $\mathsf{F}$. □

Claim B.6. If $\mathsf{1FE}_2$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(1,5)$ and $\mathcal{H}(1,6)$ are indistinguishable.

Proof. Given a $\mathsf{PPT}$ adversary $\mathcal{A}$ that distinguishes $\mathcal{H}(1,5)$ and $\mathcal{H}(1,6)$, we construct another $\mathsf{PPT}$ adversary $\mathcal{B}$ who breaks the security of the $\mathsf{1FE}_2$ scheme as follows.

1. $\mathcal{B}$ samples $(\mathsf{1FE}_1.\mathsf{PK}, \mathsf{1FE}_1.\mathsf{MSK}) \leftarrow \mathsf{1FE}_1.\mathsf{Setup}(1^\lambda), \mathsf{salt} \leftarrow \{0,1\}^\lambda$ and $\mathsf{K} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$ and gets $\mathsf{1FE}_2.\mathsf{PK}$ from the $\mathsf{1FE}_2$ challenger. It sends $\mathsf{PK} = \mathsf{1FE}_1.\mathsf{PK}$ to $\mathcal{A}$.

2. When $\mathcal{A}$ outputs a pair of challenge distributions $(\mathcal{D}^\ell_0, \mathcal{D}^\ell_1)$ with support $\Sigma^\ell$ for any arbitrary $\ell = \mathsf{poly}(\lambda)$ and a function query $M$ obeying the admissibility criteria that $\forall b \in \{0,1\}, \mathbf{w}_b \leftarrow \mathcal{D}^\ell_b$, $\mathsf{runtime}(M, \mathbf{w}_0) = \mathsf{runtime}(M, \mathbf{w}_1)$ and $M(\mathbf{w}_0) \overset{c}{\approx} M(\mathbf{w}_1)$, $\mathcal{B}$ does the following.

   (a) To simulate the challenge ciphertext, it first samples a pair of challenge messages $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow (\mathcal{D}^\ell_0, \mathcal{D}^\ell_1)$ such that $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$. It also samples a root cPRF key $\mathsf{K}_0 \leftarrow \mathsf{F.Setup}(1^\lambda)$.

   (b) $\mathcal{B}$ executes the oblivious TM $M$ on both $\mathbf{w}_0$ and $\mathbf{w}_1$ to learn the two (symbol, state) pairs $(\sigma^0_{T-1}, \mathsf{q}^0_{T-1})$ and $(\sigma^1_{T-1}, \mathsf{q}^1_{T-1})$ respectively at time step $T-1$. Additionally, $\mathcal{B}$ also learns the (symbol, state) pair $(\sigma^0_{T-2}, \mathsf{q}^0_{T-2})$ that is generated at time step $T-2$ when $M$ is executed on $\mathbf{w}_0$. Further, it records the time steps $(T', T-2)$ and $(T'', T-3)$ when the individual components of these (symbol, state) pairs for $\mathbf{w}_0$ and $\mathbf{w}_1$ are generated and then computes a root key punctured at point $(T-1\|\mathsf{salt})$ as $\mathsf{K}^{T-1}_0 = \mathsf{F.Constrain}(\mathsf{K}_0, (T-1\|\mathsf{salt}))$. It then simulates the encryption oracle by computing $\mathsf{CT}_i = \mathsf{1FE}_1.\mathsf{Enc}(\mathsf{1FE}_1.\mathsf{PK}, x_{1,i})$, where $\forall i \in [\ell], x_{1,i} = (\mathsf{K}^{T-1}_0, i, \ell, w_{0,i}, \mathsf{Trap}^1)$ and $\mathsf{Trap}^1$ is as per Figure 12. It returns the ciphertext $\mathsf{CT} = \{\mathsf{CT}_i\}_{i\in[\ell]}$ to $\mathcal{A}$.

(c) To simulate a function key for $M$, $\mathcal{B}$ does the following.

i. It first computes $\mathsf{SK}_{\mathsf{ReRand}} \leftarrow \mathsf{1FE}_1.\mathsf{KeyGen}(\mathsf{1FE}_1.\mathsf{MSK}, \mathsf{ReRand}_{\mathsf{1FE}_2.\mathsf{PK},\mathsf{salt},\mathsf{q}_{\mathsf{st}},\perp,\perp})$.

ii. In order to construct a function key for $\mathsf{Next}$, $\mathcal{B}$ needs to hardwire two $\mathsf{SKE}$ ciphertexts which it computes with the help of $\mathsf{1FE}_2$ challenger as follows.

- Delegate the punctured root key to compute $\mathsf{K}_T^{T-1} = \mathsf{F.KeyDel}(\mathsf{K}_0^{T-1}, f_T)$.
- Create a $\mathsf{1FE}_2$ challenge message pair as $((\mathbf{z}_1^0, \mathbf{z}_2^0), (\mathbf{z}_1^1, \mathbf{z}_2^1))$ such that $\forall b \in \{0,1\}$, $\mathbf{z}_1^b = (\mathsf{SYM}, \mathsf{salt}, \mathsf{K}_T^{T-1}, T-1, \ell, \sigma_{T-1}^b, \mathsf{Trap}^1)$ and $\mathbf{z}_2^b = (\mathsf{ST}, \mathsf{q}_{T-1}^b)$.
- It sends the challenge message pair $((\mathbf{z}_1^0, \mathbf{z}_2^0), (\mathbf{z}_1^1, \mathbf{z}_2^1))$ to the $\mathsf{1FE}_2$ challenger and gets back $(\mathsf{CT}_{\mathsf{sym},T-1}, \mathsf{CT}_{\mathsf{st},T-1})$.
- It then computes the two $\mathsf{SKE}$ ciphertexts $\mathsf{ct}_1 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}_{\mathsf{sym},T-1})$ and $\mathsf{ct}_2 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}_{\mathsf{st},T-1})$.

iii. $\mathcal{B}$ receives $\mathsf{SK}_{\mathsf{Next}}$ for requesting a function key for $\mathsf{Next}_{\mathsf{1FE}_2.\mathsf{PK},\mathsf{salt},\mathsf{M},\mathsf{ct}_1,\mathsf{ct}_2}$ to the $\mathsf{1FE}_2$ challenger and returns a function key for $M$ as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a guess, $\mathcal{B}$ does the same.

Note that the function key queried by $\mathcal{B}$ to the $\mathsf{1FE}_2$ challenger is for a function $\mathsf{Next}$ that outputs $\mathsf{1FE}_2$ ciphertexts that are indistinguishable by the security of $\mathsf{1FE}_2$ itself. Therefore, $\mathcal{B}$ is an admissible $\mathsf{1FE}_2$ adversary. Further, when the ciphertext for time step $T-1$ is computed as a $\mathsf{1FE}_2$ encryption of a (symbol, state) pair corresponding to bit $b = 0$, $\mathcal{A}$'s view is identical to that of $\mathcal{H}(1,5)$, and when the ciphertext for time step $T-1$ is computed as a $\mathsf{1FE}_2$ encryption of a (symbol, state) pair corresponding to bit $b = 1$, $\mathcal{A}$'s view is identical to that of $\mathcal{H}(1,6)$. Thus, the advantage of $\mathcal{A}$ in distinguishing $\mathcal{H}(1,5)$ and $\mathcal{H}(1,6)$ translates to the advantage of $\mathcal{B}$ in breaking the $\mathsf{1FE}_2$ scheme. □

Claim B.7. If $\mathsf{F}$ is a secure punctured, delegatable cPRF scheme, then hybrids $\mathcal{H}(1,6)$ and $\mathcal{H}(1,7)$ are indistinguishable.

Proof. The proof is almost identical to Claim B.5 where the reduction plays as an adversary against the cPRF challenger and simulates the $\mathsf{TMFE}$ adversary $\mathcal{A}$. The only major exceptions now are that $\mathcal{B}$ runs $M$ on both the challenge messages $\mathbf{w}_0$ and $\mathbf{w}_1$ to know the (symbol, state) pairs for both of them at the required time steps for constructing the data structure $\mathsf{Trap}$ and that the $\mathsf{1FE}_2$ ciphertext for the symbol and state corresponds to bit $b = 1$. Hence, we omit the details. □

Claim B.8. If $\mathsf{1FE}_1$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(1,7)$ and $\mathcal{H}(1,8)$ are indistinguishable.

Proof. The proof is almost identical to Claim B.4 where the reduction plays as an adversary against the $\mathsf{1FE}_1$ challenger and simulates the $\mathsf{TMFE}$ adversary $\mathcal{A}$. The only major exceptions now are that $\mathcal{B}$ runs $M$ on both the challenge messages $\mathbf{w}_0$ and $\mathbf{w}_1$ to know the (symbol, state) pairs for both of them at the required time steps for constructing the data structure $\mathsf{Trap}$ and that the $\mathsf{1FE}_2$ ciphertext for the symbol and state corresponds to bit $b = 1$. Hence, we omit the details. □

Claim B.9. If $\mathsf{1FE}_1$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(1,8)$ and $\mathcal{H}(2,1)$ are indistinguishable.

Proof. Given a PPT adversary $\mathcal{A}$ that distinguishes $\mathcal{H}(1,8)$ and $\mathcal{H}(2,1)$, we construct another PPT adversary $\mathcal{B}$ who breaks the security of the $\mathsf{1FE_1}$ scheme as follows.

1. $\mathcal{B}$ receives $\mathsf{1FE_1.PK}$ from the $\mathsf{1FE_1}$ challenger and returns this to $\mathcal{A}$. Additionally, it samples by itself $(\mathsf{1FE_2.PK}, \mathsf{1FE_2.MSK}) \leftarrow \mathsf{1FE_2.Setup}(1^\lambda), \mathsf{salt} \leftarrow \{0,1\}^\lambda$ and two random strings $\mathsf{ct}_1, \mathsf{ct}_2 \leftarrow \mathcal{C}^{\mathsf{SKE}}$, where $\mathcal{C}^{\mathsf{SKE}}$ denotes the ciphertext space of the SKE scheme.

2. When $\mathcal{A}$ outputs a pair of challenge distributions $(\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ with support $\Sigma^\ell$ for any arbitrary $\ell = \mathsf{poly}(\lambda)$ and a function query $M$ obeying the admissibility criteria that $\forall b \in \{0,1\}, \mathbf{w}_b \leftarrow \mathcal{D}_b^\ell$, $\mathsf{runtime}(M, \mathbf{w}_0) = \mathsf{runtime}(M, \mathbf{w}_1)$ and $M(\mathbf{w}_0) \overset{c}{\approx} M(\mathbf{w}_1)$, $\mathcal{B}$ does the following.

   (a) To simulate the challenge ciphertext, it first samples a pair of challenge messages $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow (\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ such that $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$. It also samples a root cPRF key $\mathsf{K}_0 \leftarrow \mathsf{F.Setup}(1^\lambda)$.

   (b) $\mathcal{B}$ executes the oblivious TM $M$ on both $\mathbf{w}_0$ and $\mathbf{w}_1$ to learn the two (symbol, state) pairs $(\sigma_{T-2}^0, \mathsf{q}_{T-2}^0)$ and $(\sigma_{T-1}^1, \mathsf{q}_{T-1}^1)$ at time steps $T-2$ and $T-1$ respectively. Further, it records the time steps $(T'', T-3)$ and $(T', T-2)$ when the individual components of these (symbol, state) pairs for $\mathbf{w}_0$ and $\mathbf{w}_1$ respectively are generated and then prepares a new pair of challenge distributions $(\widehat{\mathcal{D}}_0^\ell, \widehat{\mathcal{D}}_1^\ell)$ for the $\mathsf{1FE_1}$ challenger as follows.

      i. For $b=0$, $\widehat{\mathcal{D}}_0^\ell = \{\mathbf{x}_0 = (x_{0,1}, \ldots, x_{0,\ell})\}$, where $\forall i \in [\ell]$ $x_{0,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^0)$ with $\mathsf{Trap}^0$ being same as $\mathsf{Trap}^1$ from Figure 12.

      ii. For $b=1$, $\widehat{\mathcal{D}}_1^\ell = \{\mathbf{x}_1 = (x_{1,1}, \ldots, x_{1,\ell})\}$, where $\forall i \in [\ell]$ $x_{1,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^1)$, with the new fields in $\mathsf{Trap}^1$ as shown in Figure 13.

| | | | | | |
|---|---|---|---|---|---|
| mode-real : $\perp$ | key-id : salt | $\mathsf{val}_0 : w_{0,i}$ | $\mathsf{val}_1 : w_{1,i}$ | SKE.K : $\perp$ | $\perp$ |
| $\mathsf{mode\text{-}trap}_1 : 1$ | Target $\mathsf{TS}_1 : T-1$ | Sym $\mathsf{TS}_1 : T'$ | Sym $\mathsf{val}_1 : \sigma_{T-1}^1$ | ST $\mathsf{TS}_1 : T-2$ | ST $\mathsf{val}_1 : \mathsf{q}_{T-1}^1$ |
| $\mathsf{mode\text{-}trap}_2 : 1$ | Target $\mathsf{TS}_2 : T-2$ | Sym $\mathsf{TS}_2 : T''$ | Sym $\mathsf{val}_2 : \sigma_{T-2}^0$ | ST $\mathsf{TS}_2 : T-3$ | ST $\mathsf{val}_2 : \mathsf{q}_{T-2}^0$ |
| $\mathsf{mode\text{-}trap}_3 : \perp$ | Target TS : $\perp$ | Sym TS : $\perp$ | $\perp$ | ST TS : $\perp$ | $\perp$ |

Figure 13: $\mathsf{Trap}^1$ configuration in $\mathcal{H}(2,1)$

   (c) It sends the distribution pair to the $\mathsf{1FE_1}$ challenger and relays the response back to $\mathcal{A}$.

   (d) To simulate a function key for $M$, $\mathcal{B}$ first requests for a function key to the $\mathsf{1FE_1}$ challenger for the function $\mathsf{ReRand}_{\mathsf{1FE_2.PK},\mathsf{salt},\mathsf{q_{st}},\perp,\perp}$ and receives $\mathsf{SK}_{\mathsf{ReRand}}$. $\mathcal{B}$ computes by itself $\mathsf{SK}_{\mathsf{Next}} \leftarrow \mathsf{1FE_2.KeyGen}(\mathsf{1FE_2.MSK}, \mathsf{Next}_{\mathsf{1FE_2.PK},\mathsf{salt},M,\mathsf{ct}_1,\mathsf{ct}_2})$ and returns a function key for $M$ as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a guess, $\mathcal{B}$ does the same.

Note that for all time steps $t \notin \{T', T-2\}$, the decryption outputs are exactly the same ciphertexts in both the $\mathcal{H}(1,8)$ and $\mathcal{H}(2,1)$. At a time step $t \in \{T', T-2\}$ in $\mathcal{H}(2,1)$, the decryption mimics the decryption of $\mathcal{H}(1,8)$ dictated by $(\mathsf{Trap}^1.\mathsf{mode\text{-}trap}_1 = 1 \wedge \mathsf{Trap}^1.\mathsf{mode\text{-}trap}_3 = \perp)$. More specifically, in $\mathcal{H}(1,8)$ the symbol and state ciphertexts corresponding to time step $T-1$ is first computed by decrypting the SKE ciphertext components hardwired in Next and outputting them at time steps $T'$ and $T-2$ respectively. Alternatively, in $\mathcal{H}(2,1)$, $(\mathsf{Trap}^1.\mathsf{mode\text{-}trap}_1 = 1 \wedge \mathsf{Trap}^1.\mathsf{mode\text{-}trap}_3 = \perp)$ dictates that these ciphertext

components are computed with the same randomness at exactly the same time steps $T'$ and $T-2$ respectively. Thus $\mathcal{B}$ is an admissible adversary against the $\mathsf{1FE}_1$ challenger since the outputs for the two challenge message sets are exactly the same. If $b=0$, $\mathcal{A}$ sees the distribution of $\mathcal{H}(1,8)$, while if $b=1$, $\mathcal{A}$ sees the distribution of $\mathcal{H}(2,1)$. Thus the advantage of $\mathcal{A}$ translates to the advantage of $\mathcal{B}$. □

Claim B.10. If $\mathsf{SKE}$ is a secure symmetric-key encryption scheme, then hybrids $\mathcal{H}(2,1)$ and $\mathcal{H}(2,2)$ are indistinguishable.

Proof. Given a $\mathsf{PPT}$ adversary $\mathcal{A}$ that distinguishes $\mathcal{H}(2,1)$ and $\mathcal{H}(2,2)$, we construct another $\mathsf{PPT}$ adversary $\mathcal{B}$ who breaks the security of the $\mathsf{SKE}$ scheme as follows.

1. $\mathcal{B}$ samples $(\mathsf{1FE}_1.\mathsf{PK}, \mathsf{1FE}_1.\mathsf{MSK}) \leftarrow \mathsf{1FE}_1.\mathsf{Setup}(1^\lambda)$, $(\mathsf{1FE}_2.\mathsf{PK}, \mathsf{1FE}_2.\mathsf{MSK}) \leftarrow \mathsf{1FE}_2.\mathsf{Setup}(1^\lambda)$ and $\mathsf{salt} \leftarrow \{0,1\}^\lambda$. It sends $\mathsf{PK} = \mathsf{1FE}_1.\mathsf{PK}$ to $\mathcal{A}$.

2. When $\mathcal{A}$ outputs a pair of challenge distributions $(\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ with support $\Sigma^\ell$ for any arbitrary $\ell = \mathsf{poly}(\lambda)$ and a function query $M$ obeying the admissibility criteria that $\forall b \in \{0,1\}, \mathbf{w}_b \leftarrow \mathcal{D}_b^\ell$, $\mathsf{runtime}(M, \mathbf{w}_0) = \mathsf{runtime}(M, \mathbf{w}_1)$ and $M(\mathbf{w}_0) \overset{c}{\approx} M(\mathbf{w}_1)$, $\mathcal{B}$ does the following.

   (a) To simulate the challenge ciphertext, it first samples a pair of challenge messages $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow (\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ such that $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$. It also samples a root cPRF key $\mathsf{K}_0 \leftarrow \mathsf{F.Setup}(1^\lambda)$.

   (b) $\mathcal{B}$ executes the oblivious TM $M$ on both $\mathbf{w}_0$ and $\mathbf{w}_1$ to learn the (symbol, state) pairs $(\sigma_{T-2}^0, \mathsf{q}_{T-2}^0)$ and $(\sigma_{T-1}^1, \mathsf{q}_{T-1}^1)$ at time steps $T-2$ and $T-1$ respectively. It also records the time steps $(T'', T-3)$ and $(T', T-2)$ when the individual components of these two (symbol, state) pairs are generated. It then simulates the encryption oracle by computing $\mathsf{CT}_i = \mathsf{1FE}_1.\mathsf{Enc}(\mathsf{1FE}_1.\mathsf{PK}, x_{1,i})$, where $\forall i \in [\ell], x_{1,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^1)$ and $\mathsf{Trap}^1$ is as per Figure 13. It returns the ciphertext $\mathsf{CT} = \{\mathsf{CT}_i\}_{i\in[\ell]}$ to $\mathcal{A}$.

   (c) To simulate a function key for $M$, $\mathcal{B}$ does the following.

      i. It first computes $\mathsf{SK}_{\mathsf{ReRand}} \leftarrow \mathsf{1FE}_1.\mathsf{KeyGen}(\mathsf{1FE}_1.\mathsf{MSK}, \mathsf{ReRand}_{\mathsf{1FE}_2.\mathsf{PK},\mathsf{salt},\mathsf{q}_{\mathsf{st}},\perp,\perp})$.

      ii. It then computes $\mathsf{1FE}_2$ encodings of $(\sigma_{T-2}^0, \mathsf{q}_{T-2}^0)$ as follows.

         - Compute a delegated cPRF key $\mathsf{K}_{T-1} = \mathsf{F.KeyDel}(\mathsf{K}_0, f_{T-1})$ and generate the encryption randomness for time step $T-2$ as $r_{T-2} = \mathsf{F.Eval}(\mathsf{K}_0, (T-2 \| \mathsf{salt}))$.
         - Compute the $\mathsf{1FE}_2$ symbol ciphertext to be given as output at time step $T''$ for the future time step $T-2$ as $\mathsf{CT}_{\mathsf{sym},T-2}^0 = \mathsf{1FE}_2.\mathsf{Enc}(\mathsf{1FE}_2.\mathsf{PK}_1, \mathbf{z}_1^0; r_{T-2})$, where $\mathbf{z}_1^0 = (\mathsf{SYM}, \mathsf{salt}, \mathsf{K}_{T-1}, T-2, \ell, \sigma_{T-2}^0, \mathsf{Trap}^1)$ and $\mathsf{Trap}^1$ is as per Figure 13.
         - Compute the $\mathsf{1FE}_2$ state ciphertext to be given as output at time step $T-3$ for the future time step $T-2$ as $\mathsf{CT}_{\mathsf{st},T-2}^0 = \mathsf{1FE}_2.\mathsf{Enc}(\mathsf{1FE}_2.\mathsf{PK}_2, \mathbf{z}_2^0; r_{T-2})$, where $\mathbf{z}_2^0 = (\mathsf{ST}, \mathsf{q}_{T-2}^0)$.

      iii. It sends the $\mathsf{1FE}_2$ ciphertexts $\mathsf{CT}_{\mathsf{sym},T-2}^0, \mathsf{CT}_{\mathsf{st},T-2}^0$ to the challenger of the $\mathsf{SKE}$ scheme and gets back $\mathsf{ct}_1, \mathsf{ct}_2$.

      iv. $\mathcal{B}$ then computes $\mathsf{SK}_{\mathsf{Next}} \leftarrow \mathsf{1FE}_2.\mathsf{KeyGen}(\mathsf{1FE}_2.\mathsf{MSK}, \mathsf{Next}_{\mathsf{1FE}_2.\mathsf{PK},\mathsf{salt},M,\mathsf{ct}_1,\mathsf{ct}_2})$ and returns a function key for $M$ as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a guess, $\mathcal{B}$ does the same.

Note that the only difference between the two hybrids is that the $\mathsf{SKE}$ ciphertexts hard-wired in the function key are random strings in $\mathcal{H}(2,1)$ and are valid $\mathsf{SKE}$ encryptions of $(\mathsf{CT}^0_{\mathsf{sym},T-2}, \mathsf{CT}^0_{\mathsf{st},T-2})$ encoding the (symbol, state) pair for time step $T-2$ in $\mathcal{H}(2,2)$. Hence the advantage of an adversary who distinguishes between the two hybrids translates to an advantage of an adversary against the $\mathsf{SKE}$ scheme. □

Claim B.11. If $\mathsf{1FE}_1$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(2,2)$ and $\mathcal{H}(2,3)$ are indistinguishable.

Proof. Given a $\mathsf{PPT}$ adversary $\mathcal{A}$ that distinguishes $\mathcal{H}(2,2)$ and $\mathcal{H}(2,3)$, we construct another $\mathsf{PPT}$ adversary $\mathcal{B}$ who breaks the security of the $\mathsf{1FE}_1$ scheme as follows.

1. $\mathcal{B}$ receives $\mathsf{1FE}_1.\mathsf{PK}$ from the $\mathsf{1FE}_1$ challenger and returns this to $\mathcal{A}$. Additionally, it samples by itself $(\mathsf{1FE}_2.\mathsf{PK}, \mathsf{1FE}_2.\mathsf{MSK}) \leftarrow \mathsf{1FE}_2.\mathsf{Setup}(1^\lambda), \mathsf{salt} \leftarrow \{0,1\}^\lambda, \mathsf{K} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$.
2. When $\mathcal{A}$ outputs a pair of challenge distributions $(\mathcal{D}^\ell_0, \mathcal{D}^\ell_1)$ with support $\Sigma^\ell$ for any arbitrary $\ell = \mathsf{poly}(\lambda)$ and a function query $M$ obeying the admissibility criteria that $\forall b \in \{0,1\}, \mathbf{w}_b \leftarrow \mathcal{D}^\ell_b$, $\mathsf{runtime}(M, \mathbf{w}_0) = \mathsf{runtime}(M, \mathbf{w}_1)$ and $M(\mathbf{w}_0) \overset{c}{\approx} M(\mathbf{w}_1)$, $\mathcal{B}$ does the following.
   (a) To simulate the challenge ciphertext, it first samples a pair of challenge messages $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow (\mathcal{D}^\ell_0, \mathcal{D}^\ell_1)$ such that $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$. It also samples a root cPRF key $\mathsf{K}_0 \leftarrow \mathsf{F.Setup}(1^\lambda)$.
   (b) $\mathcal{B}$ executes the oblivious TM $M$ on both $\mathbf{w}_0$ and $\mathbf{w}_1$ to learn the two (symbol, state) pairs $(\sigma^0_{T-2}, \mathsf{q}^0_{T-2})$ and $(\sigma^1_{T-1}, \mathsf{q}^1_{T-1})$ at time steps $T-2$ and $T-1$ respectively. Further, it records the time steps $(T'', T-3)$ and $(T', T-2)$ when the individual components of these (symbol, state) pairs for $\mathbf{w}_0$ and $\mathbf{w}_1$ respectively are generated and then prepares a new pair of challenge distributions $(\widehat{\mathcal{D}}^\ell_0, \widehat{\mathcal{D}}^\ell_1)$ for the $\mathsf{1FE}_1$ challenger as follows.
      i. For $b=0$, $\widehat{\mathcal{D}}^\ell_0 = \{\mathbf{x}_0 = (x_{0,1}, \ldots, x_{0,\ell})\}$, where $\forall i \in [\ell]$ $x_{0,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^0)$ with $\mathsf{Trap}^0$ being same as $\mathsf{Trap}^1$ from Figure 13.
      ii. For $b=1$, $\widehat{\mathcal{D}}^\ell_1 = \{\mathbf{x}_1 = (x_{1,1}, \ldots, x_{1,\ell})\}$, where $\forall i \in [\ell]$ $x_{1,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^1)$, with the new fields in $\mathsf{Trap}^1$ as shown in Figure 14.

| | | | | | |
|---|---|---|---|---|---|
| mode-real : $\perp$ | key-id : salt | $\mathsf{val}_0 : w_{0,i}$ | $\mathsf{val}_1 : w_{1,i}$ | SKE.K : K | $\perp$ |
| $\mathsf{mode\text{-}trap}_1 : 1$ | Target $\mathsf{TS}_1 : T-1$ | Sym $\mathsf{TS}_1 : T'$ | Sym $\mathsf{val}_1 : \sigma^1_{T-1}$ | ST $\mathsf{TS}_1 : T-2$ | ST $\mathsf{val}_1 : \mathsf{q}^1_{T-1}$ |
| $\mathsf{mode\text{-}trap}_2 : \perp$ | Target $\mathsf{TS}_2 : \perp$ | Sym $\mathsf{TS}_2 : \perp$ | Sym $\mathsf{val}_2 : \perp$ | ST $\mathsf{TS}_2 : \perp$ | ST $\mathsf{val}_2 : \perp$ |
| $\mathsf{mode\text{-}trap}_3 : 1$ | Target TS : $T-2$ | Sym TS : $T''$ | $\perp$ | ST TS : $T-3$ | $\perp$ |

Figure 14: $\mathsf{Trap}^1$ configuration in $\mathcal{H}(2,3)$

   (c) It sends the distribution pair to the $\mathsf{1FE}_1$ challenger and relays the response back to $\mathcal{A}$.
   (d) To simulate a function key for $M$, $\mathcal{B}$ does the following.
      i. It requests for a function key for $\mathsf{ReRand}_{\mathsf{1FE}_2.\mathsf{PK},\mathsf{salt},\mathsf{q}_{\mathsf{st}},\perp,\perp}$ to the $\mathsf{1FE}_1$ challenger and receives $\mathsf{SK}_{\mathsf{ReRand}}$.

ii. It then computes $\mathsf{1FE_2}$ encodings of $(\sigma^0_{T-2}, \mathsf{q}^0_{T-2})$, as follows.

- Compute a delegated cPRF key $\mathsf{K}_{T-1} = \mathsf{F.KeyDel}(\mathsf{K}_0, f_{T-1})$ and generate the encryption randomness for time step $T-2$ as $r_{T-2} = \mathsf{F.Eval}(\mathsf{K}_0, (T-2 \| \mathsf{salt}))$.
- Compute the $\mathsf{1FE_2}$ symbol ciphertext to be given as output at time step $T''$ for the future time step $T-2$ as $\mathsf{CT}^0_{\mathsf{sym},T-2} = \mathsf{1FE_2.Enc}(\mathsf{1FE_2.PK_1}, \mathbf{z}^0_1; r_{T-2})$, where $\mathbf{z}^0_1 = (\mathsf{SYM}, \mathsf{salt}, \mathsf{K}_{T-1}, T-2, \ell, \sigma^0_{T-2}, \mathsf{Trap}^1)$ and $\mathsf{Trap}^1$ is as per Figure 14.
- Compute the $\mathsf{1FE_2}$ state ciphertext to be given as output at time step $T-3$ for future time step $T-2$ as $\mathsf{CT}^0_{\mathsf{st},T-2} = \mathsf{1FE_2.Enc}(\mathsf{1FE_2.PK_2}, \mathbf{z}^0_2; r_{T-2})$, where $\mathbf{z}^0_2 = (\mathsf{ST}, \mathsf{q}^0_{T-2})$.

iii. Once it has generated the two $\mathsf{1FE_2}$ ciphertexts $\mathsf{CT}^0_{\mathsf{sym},T-2}$ and $\mathsf{CT}^0_{\mathsf{st},T-2}$, it computes two $\mathsf{SKE}$ ciphertexts $\mathsf{ct_1} = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}^0_{\mathsf{sym},T-2})$ and $\mathsf{ct_2} = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}^0_{\mathsf{st},T-2})$.

iv. Finally, it computes $\mathsf{SK}_{\mathsf{Next}} \leftarrow \mathsf{1FE_2.KeyGen}(\mathsf{1FE_2.MSK}, \mathsf{Next}_{\mathsf{1FE_2.PK},\mathsf{salt},M,\mathsf{ct_1},\mathsf{ct_2}})$ and returns a function key for $M$ as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a guess, $\mathcal{B}$ does the same.

Observe that for all time steps $t \notin \{T'', T-3\}$, the decryption outputs are exactly the same ciphertexts in both the $\mathcal{H}(2,2)$ and $\mathcal{H}(2,3)$. At time step $t \in \{T'', T-3\}$ in $\mathcal{H}(2,2)$, $(\mathsf{Trap}^0.\mathsf{mode\text{-}trap_2} = 1 \wedge \mathsf{Trap}^0.\mathsf{mode\text{-}trap_3} = \perp)$ (in Figure 13) dictates the decryption to output two decomposed components of a single $\mathsf{1FE_2}$ ciphertext, one component encoding $\mathsf{Trap}^0.\mathsf{Sym}\ \mathsf{val}_2 = \sigma^0_{T-2}$ at time step $T''$ and the other encoding $\mathsf{Trap}^0.\mathsf{ST}\ \mathsf{val}_2 = \mathsf{q}^0_{T-2}$ at time step $T-3$. Alternatively in $\mathcal{H}(2,3)$, $(\mathsf{Trap}^0.\mathsf{mode\text{-}trap_2} = \perp \wedge \mathsf{Trap}^1.\mathsf{mode\text{-}trap_3} = 1)$ (in Figure 14) dictates the decryption to firstly use $\mathsf{Trap}^1.\mathsf{SKE.K} = \mathsf{K}$ to decrypt the hardwired ciphertext $\mathsf{ct_1}$ and output $\mathsf{CT}^0_{\mathsf{sym},T-2}$ at time step $T''$ (respectively, $\mathsf{ct_2}$ and output $\mathsf{CT}^0_{\mathsf{st},T-2}$ at time step $T-3$). In both the hybrids these symbol and state ciphertext pieces are computed for target time step $T-2$. Thus $\mathcal{B}$ is an admissible $\mathsf{1FE_1}$ adversary. If $b=0$, $\mathcal{A}$ sees the distribution of $\mathcal{H}(2,2)$, while if $b=1$, $\mathcal{A}$ sees the distribution of $\mathcal{H}(2,3)$. Hence the advantage of $\mathcal{A}$ translates to the advantage of $\mathcal{B}$. □

Claim B.12. If $\mathsf{1FE_1}$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(2,3)$ and $\mathcal{H}(2,4)$ are indistinguishable.

Proof. Given a $\mathsf{PPT}$ adversary $\mathcal{A}$ that distinguishes $\mathcal{H}(2,3)$ and $\mathcal{H}(2,4)$, we construct another $\mathsf{PPT}$ adversary $\mathcal{B}$ who breaks the security of the $\mathsf{1FE_1}$ scheme as follows.

1. $\mathcal{B}$ receives $\mathsf{1FE_1.PK}$ from the $\mathsf{1FE_1}$ challenger and returns this to $\mathcal{A}$. Additionally, it samples by itself $(\mathsf{1FE_2.PK}, \mathsf{1FE_2.MSK}) \leftarrow \mathsf{1FE_2.Setup}(1^\lambda), \mathsf{salt} \leftarrow \{0,1\}^\lambda$ and a key $\mathsf{K} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$.

2. When $\mathcal{A}$ outputs a pair of challenge distributions $(\mathcal{D}^\ell_0, \mathcal{D}^\ell_1)$ with support $\Sigma^\ell$ for any arbitrary $\ell = \mathsf{poly}(\lambda)$ and a function query $M$ obeying the admissibility criteria that $\forall b \in \{0,1\}, \mathbf{w}_b \leftarrow \mathcal{D}^\ell_b$, $\mathsf{runtime}(M, \mathbf{w}_0) = \mathsf{runtime}(M, \mathbf{w}_1)$ and $M(\mathbf{w}_0) \overset{c}{\approx} M(\mathbf{w}_1)$, $\mathcal{B}$ does the following.

   (a) To simulate the challenge ciphertext, it first samples a pair of challenge messages $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow (\mathcal{D}^\ell_0, \mathcal{D}^\ell_1)$ such that $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$. It also samples a root cPRF key $\mathsf{K}_0 \leftarrow \mathsf{F.Setup}(1^\lambda)$.

(b) $\mathcal{B}$ executes the oblivious TM $M$ on both $\mathbf{w}_0$ and $\mathbf{w}_1$ to learn the two (symbol, state) pairs $(\sigma^0_{T-2}, \mathsf{q}^0_{T-2})$ and $(\sigma^1_{T-1}, \mathsf{q}^1_{T-1})$ at time steps $T-2$ and $T-1$ respectively. Further, it records the time steps $(T'', T-3)$ and $(T', T-2)$ when the individual components of these (symbol, state) pairs for $\mathbf{w}_0$ and $\mathbf{w}_1$ respectively are generated. It then computes a root key punctured at point $(T-2\|\mathsf{salt})$ as $\mathsf{K}_0^{T-2} = \mathsf{F.Constrain}(\mathsf{K}_0, (T-2\|\mathsf{salt}))$ and prepares a new pair of challenge distributions $(\widehat{\mathcal{D}}^\ell_0, \widehat{\mathcal{D}}^\ell_1)$ for the $\mathsf{1FE_1}$ challenger as follows.

i. For $b=0$, $\widehat{\mathcal{D}}^\ell_0 = \{\mathbf{x}_0 = (x_{0,1}, \ldots, x_{0,\ell})\}$, where $\forall i \in [\ell]$ $x_{0,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^1)$ with $\mathsf{Trap}^1$ as per Figure 14.

ii. For $b=1$, $\widehat{\mathcal{D}}^\ell_1 = \{\mathbf{x}_1 = (x_{1,1}, \ldots, x_{1,\ell})\}$, where $\forall i \in [\ell]$ $x_{1,i} = (\mathsf{K}_0^{T-2}, i, \ell, w_{0,i}, \mathsf{Trap}^1)$, with $\mathsf{Trap}^1$ as per Figure 14.

(c) It sends the distribution pair to the $\mathsf{1FE_1}$ challenger and relays the response back to $\mathcal{A}$.

(d) To simulate a function key for $M$, $\mathcal{B}$ does the following.

i. It requests for a function key for $\mathsf{ReRand}_{\mathsf{1FE_2.PK},\mathsf{salt},\mathsf{q_{st}},\perp,\perp}$ to the $\mathsf{1FE_1}$ challenger and receives $\mathsf{SK}_{\mathsf{ReRand}}$.

ii. It then computes $\mathsf{1FE_2}$ encodings of $(\sigma^0_{T-2}, \mathsf{q}^0_{T-2})$, as follows.

- Compute a punctured, delegated key $\mathsf{K}^{T-2}_{T-1} = \mathsf{F.KeyDel}(\mathsf{K}_0^{T-2}, f_{T-1})$ and generate the encryption randomness for time step $T-2$ as $r_{T-2} = \mathsf{F.Eval}(\mathsf{K}_0, (T-2\|\mathsf{salt}))$.
- Compute the $\mathsf{1FE_2}$ symbol ciphertext to be given as output at time step $T''$ for the future time step $T-2$ as $\mathsf{CT}^0_{\mathsf{sym},T-2} = \mathsf{1FE_2.Enc}(\mathsf{1FE_2.PK}_1, \mathbf{z}^0_1; r_{T-2})$, where $\mathbf{z}^0_1 = (\mathsf{SYM}, \mathsf{salt}, \mathsf{K}^{T-2}_{T-1}, T-2, \ell, \sigma^0_{T-2}, \mathsf{Trap}^1)$ and $\mathsf{Trap}^1$ is as per Figure 14.
- Compute the $\mathsf{1FE_2}$ state ciphertext to be given as output at time step $T-3$ for future time step $T-2$ as $\mathsf{CT}^0_{\mathsf{st},T-2} = \mathsf{1FE_2.Enc}(\mathsf{1FE_2.PK}_2, \mathbf{z}^0_2; r_{T-2})$, where $\mathbf{z}^0_2 = (\mathsf{ST}, \mathsf{q}^0_{T-2})$.

iii. Once it has generated the two $\mathsf{1FE_2}$ ciphertexts $\mathsf{CT}^0_{\mathsf{sym},T-2}$ and $\mathsf{CT}^0_{\mathsf{st},T-2}$, it computes two $\mathsf{SKE}$ ciphertexts $\mathsf{ct}_1 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}^0_{\mathsf{sym},T-2})$ and $\mathsf{ct}_2 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}^0_{\mathsf{st},T-2})$.

iv. Finally, it computes $\mathsf{SK}_{\mathsf{Next}} \leftarrow \mathsf{1FE_2.KeyGen}(\mathsf{1FE_2.MSK}, \mathsf{Next}_{\mathsf{1FE_2.PK},\mathsf{salt},M,\mathsf{ct}_1,\mathsf{ct}_2})$ and returns a function key for $M$ as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a guess, $\mathcal{B}$ does the same.

Note that the only difference in $\mathcal{H}(2,3)$ and $\mathcal{H}(2,4)$ is the replacement of the root cPRF key $\mathsf{K}_0$ with a punctured root key $\mathsf{K}_0^{T-2}$ at point $(T-2\|\mathsf{salt})$ in time step $T-2$ in the $\mathsf{1FE_1}$ ciphertext. Moreover, in both the hybrids, the field $\mathsf{Trap}^1.\mathsf{mode\text{-}trap}_3 = 1$ dictates the output at time step $t \in \{T'', T-3\}$ to be a ciphertext component for time step $T-2$ as argued in Claim B.11. Thus, the cPRF key is only required to compute randomness at points $\neq (T-2\|\mathsf{salt})$ for which the punctured root key suffices. Further, it evaluates to the same value as the normal key on all such points in both the hybrids. As a consequence, the decryption values are exactly the same for all the time steps proving the admissibility of $\mathcal{B}$. Thus if $b=0$, $\mathcal{A}$ sees the distribution of $\mathcal{H}(2,3)$, while if $b=1$, $\mathcal{A}$ sees the distribution of $\mathcal{H}(2,4)$. Hence the advantage of $\mathcal{A}$ translates to the advantage of $\mathcal{B}$. □

Claim B.13. If $\mathsf{F}$ is a secure punctured, delegatable cPRF scheme, then hybrids $\mathcal{H}(2,4)$ and $\mathcal{H}(2,5)$ are indistinguishable.

Proof. The proof is almost identical to Claim B.5 where the reduction plays as an adversary against the cPRF challenger and simulates the $\mathsf{TMFE}$ adversary $\mathcal{A}$ with the following major exceptions.

1. $\mathcal{B}$ runs $M$ on both the sampled messages $\mathbf{w}_0$ and $\mathbf{w}_1$ to know the (symbol, state) pairs at the time steps $T-2$ and $T-1$ respectively for constructing the data structure $\mathsf{Trap}$ as in $\mathcal{H}(2,4)$. The challenge ciphertext encodes $\mathsf{K}_0^{T-2}$, i.e., a root key punctured at point $(T-2\|\mathsf{salt})$.
2. The cPRF challenger is queried at the point $(T-2\|\mathsf{salt})$ to receive an encryption randomness for time step $T-2$. This is used in computing the $\mathsf{1FE_2}$ ciphertext encoding the (symbol, state) pair generated at time steps $(T'', T-3)$ for time step $T-2$ when $M$ is run on $\mathbf{w}_0$.

The other details follow as before and hence we omit them. □

Claim B.14. If $\mathsf{1FE_2}$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(2,5)$ and $\mathcal{H}(2,6)$ are indistinguishable.

Proof. Given a $\mathsf{PPT}$ adversary $\mathcal{A}$ that distinguishes $\mathcal{H}(2,5)$ and $\mathcal{H}(2,6)$, we construct another $\mathsf{PPT}$ adversary $\mathcal{B}$ who breaks the security of the $\mathsf{1FE_2}$ scheme as follows.

1. $\mathcal{B}$ samples $(\mathsf{1FE_1.PK}, \mathsf{1FE_1.MSK}) \leftarrow \mathsf{1FE_1.Setup}(1^\lambda), \mathsf{salt} \leftarrow \{0,1\}^\lambda$ and $\mathsf{K} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$ and gets $\mathsf{1FE_2.PK}$ from the $\mathsf{1FE_2}$ challenger. It sends $\mathsf{PK} = \mathsf{1FE_1.PK}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs a pair of challenge distributions $(\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ with support $\Sigma^\ell$ for any arbitrary $\ell = \mathsf{poly}(\lambda)$ and a function query $M$ obeying the admissibility criteria that $\forall b \in \{0,1\}, \mathbf{w}_b \leftarrow \mathcal{D}_b^\ell$, $\mathsf{runtime}(M, \mathbf{w}_0) = \mathsf{runtime}(M, \mathbf{w}_1)$ and $M(\mathbf{w}_0) \stackrel{c}{\approx} M(\mathbf{w}_1)$, $\mathcal{B}$ does the following.
   (a) To simulate the challenge ciphertext, it first samples a pair of challenge messages $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow (\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ such that $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$. It also samples a root cPRF key $\mathsf{K}_0 \leftarrow \mathsf{F.Setup}(1^\lambda)$.
   (b) $\mathcal{B}$ executes the oblivious TM $M$ on both $\mathbf{w}_0$ and $\mathbf{w}_1$ to learn the two (symbol, state) pairs $(\sigma_{T-2}^0, \mathsf{q}_{T-2}^0)$ and $(\sigma_{T-2}^1, \mathsf{q}_{T-2}^1)$ respectively at time step $T-2$. Additionally, $\mathcal{B}$ also learns the (symbol, state) pair $(\sigma_{T-1}^1, \mathsf{q}_{T-1}^1)$ that is generated at time step $T-1$ when $M$ is executed on $\mathbf{w}_1$. Further, it records the time steps $(T'', T-3)$ and $(T', T-2)$ when the individual components of these (symbol, state) pairs for $\mathbf{w}_0$ and $\mathbf{w}_1$ are generated and then computes a root key punctured at point $(T-2\|\mathsf{salt})$ as $\mathsf{K}_0^{T-2} = \mathsf{F.Constrain}(\mathsf{K}_0, (T-2\|\mathsf{salt}))$. It then simulates the encryption oracle by computing $\mathsf{CT}_i = \mathsf{1FE_1.Enc}(\mathsf{1FE_1.PK}, x_{1,i})$, where $\forall i \in [\ell], x_{1,i} = (\mathsf{K}_0^{T-2}, i, \ell, w_{0,i}, \mathsf{Trap}^1)$ and $\mathsf{Trap}^1$ is as per Figure 14. It returns the ciphertext $\mathsf{CT} = \{\mathsf{CT}_i\}_{i\in[\ell]}$ to $\mathcal{A}$.
   (c) To simulate a function key for $M$, $\mathcal{B}$ does the following.
      i. It first computes $\mathsf{SK}_{\mathsf{ReRand}} \leftarrow \mathsf{1FE_1.KeyGen}(\mathsf{1FE_1.MSK}, \mathsf{ReRand}_{\mathsf{1FE_2.PK},\mathsf{salt},\mathsf{q_{st}},\perp,\perp})$.
      ii. In order to construct a function key for $\mathsf{Next}$, $\mathcal{B}$ needs to hardwire two $\mathsf{SKE}$ ciphertexts which it computes with the help of $\mathsf{1FE_2}$ challenger as follows.
         - Delegate the punctured root key to compute $\mathsf{K}_{T-1}^{T-2} = \mathsf{F.KeyDel}(\mathsf{K}_0^{T-2}, f_{T-1})$.

- Create a $\mathsf{1FE}_2$ challenge message pair as $((\mathbf{z}_1^0, \mathbf{z}_2^0), (\mathbf{z}_1^1, \mathbf{z}_2^1))$ such that $\forall b \in \{0,1\}$, $\mathbf{z}_1^b = (\mathsf{SYM}, \mathsf{salt}, \mathsf{K}_{T-1}^{T-2}, T-2, \ell, \sigma_{T-2}^b, \mathsf{Trap}^1)$ and $\mathbf{z}_2^b = (\mathsf{ST}, \mathsf{q}_{T-2}^b)$, where $\mathsf{Trap}^1$ is as per Figure 14.
- It sends the challenge message pair $((\mathbf{z}_1^0, \mathbf{z}_2^0), (\mathbf{z}_1^1, \mathbf{z}_2^1))$ to the $\mathsf{1FE}_2$ challenger and gets back $(\mathsf{CT}_{\mathsf{sym},T-2}, \mathsf{CT}_{\mathsf{st},T-2})$.
- It then computes the two $\mathsf{SKE}$ ciphertexts $\mathsf{ct}_1 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}_{\mathsf{sym},T-2})$ and $\mathsf{ct}_2 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}_{\mathsf{st},T-2})$.

iii. $\mathcal{B}$ receives $\mathsf{SK}_{\mathsf{Next}}$ for requesting a function key for $\mathsf{Next}_{\mathsf{1FE}_2.\mathsf{PK},\mathsf{salt},\mathsf{M},\mathsf{ct}_1,\mathsf{ct}_2}$ to the $\mathsf{1FE}_2$ challenger and returns a function key for $M$ as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a guess, $\mathcal{B}$ does the same.

Note that the function key queried by $\mathcal{B}$ to the $\mathsf{1FE}_2$ challenger is for a function $\mathsf{Next}$ that outputs $\mathsf{1FE}_2$ ciphertexts that are indistinguishable by the security of $\mathsf{1FE}_2$ itself. Therefore, $\mathcal{B}$ is an admissible $\mathsf{1FE}_2$ adversary. Further, when the ciphertext for time step $T-2$ is computed as a $\mathsf{1FE}_2$ encryption of a (symbol, state) pair corresponding to bit $b = 0$, $\mathcal{A}$'s view is identical to that of $\mathcal{H}(2,5)$, and when the ciphertext for time step $T-2$ is computed as a $\mathsf{1FE}_2$ encryption of a (symbol, state) pair corresponding to bit $b = 1$, $\mathcal{A}$'s view is identical to that of $\mathcal{H}(2,6)$. Thus, the advantage of $\mathcal{A}$ in distinguishing $\mathcal{H}(2,5)$ and $\mathcal{H}(2,6)$ translates to the advantage of $\mathcal{B}$ in breaking the $\mathsf{1FE}_2$ scheme. □

Claim B.15. If $\mathsf{F}$ is a secure punctured, delegatable cPRF scheme, then hybrids $\mathcal{H}(2,6)$ and $\mathcal{H}(2,7)$ are indistinguishable.

Proof. The proof is similar to Claim B.7 and hence we omit the details. □

Claim B.16. If $\mathsf{1FE}_1$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(2,7)$ and $\mathcal{H}(2,8)$ are indistinguishable.

Proof. The proof is similar to Claim B.8 and hence we omit the details. □

Claim B.17. If $\mathsf{1FE}_1$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(2,8)$ and $\mathcal{H}(3,1)$ are indistinguishable.

Proof. Given a $\mathsf{PPT}$ adversary $\mathcal{A}$ that distinguishes $\mathcal{H}(2,8)$ and $\mathcal{H}(3,1)$, we construct another $\mathsf{PPT}$ adversary $\mathcal{B}$ who breaks the security of the $\mathsf{1FE}_1$ scheme as follows.

1. $\mathcal{B}$ receives $\mathsf{1FE}_1.\mathsf{PK}$ from the $\mathsf{1FE}_1$ challenger and returns this to $\mathcal{A}$. Additionally, it samples by itself $(\mathsf{1FE}_2.\mathsf{PK}, \mathsf{1FE}_2.\mathsf{MSK}) \leftarrow \mathsf{1FE}_2.\mathsf{Setup}(1^\lambda)$, $\mathsf{salt} \leftarrow \{0,1\}^\lambda$ and two random strings $\mathsf{ct}_1, \mathsf{ct}_2 \leftarrow \mathcal{C}^{\mathsf{SKE}}$, where $\mathcal{C}^{\mathsf{SKE}}$ denotes the ciphertext space of the $\mathsf{SKE}$ scheme.

2. When $\mathcal{A}$ outputs a pair of challenge distributions $(\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ with support $\Sigma^\ell$ for any arbitrary $\ell = \mathsf{poly}(\lambda)$ and a function query $M$ obeying the admissibility criteria that $\forall b \in \{0,1\}, \mathbf{w}_b \leftarrow \mathcal{D}_b^\ell$, $\mathsf{runtime}(M, \mathbf{w}_0) = \mathsf{runtime}(M, \mathbf{w}_1)$ and $M(\mathbf{w}_0) \overset{c}{\approx} M(\mathbf{w}_1)$, $\mathcal{B}$ does the following.

   (a) To simulate the challenge ciphertext, it first samples a pair of challenge messages $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow (\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ such that $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$. It also samples a root cPRF key $\mathsf{K}_0 \leftarrow \mathsf{F.Setup}(1^\lambda)$.

(b) $\mathcal{B}$ executes the oblivious TM $M$ on both $\mathbf{w}_0$ and $\mathbf{w}_1$ to learn the two (symbol, state) pairs $(\sigma^0_{T-3}, \mathsf{q}^0_{T-3})$ and $(\sigma^1_{T-2}, \mathsf{q}^1_{T-2})$ at time steps $T-3$ and $T-2$ respectively. Further, it records the time steps $(T''', T-4)$ and $(T'', T-3)$ when the individual components of these (symbol, state) pairs for $\mathbf{w}_0$ and $\mathbf{w}_1$ respectively are generated and then prepares a new pair of challenge distributions $(\widehat{\mathcal{D}}^\ell_0, \widehat{\mathcal{D}}^\ell_1)$ for the $\mathsf{1FE_1}$ challenger as follows.

i. For $b=0$, $\widehat{\mathcal{D}}^\ell_0 = \{\mathbf{x}_0 = (x_{0,1}, \ldots, x_{0,\ell})\}$, where $\forall i \in [\ell]$ $x_{0,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^0)$ with $\mathsf{Trap}^0$ being same as $\mathsf{Trap}^1$ from Figure 14.

ii. For $b=1$, $\widehat{\mathcal{D}}^\ell_1 = \{\mathbf{x}_1 = (x_{1,1}, \ldots, x_{1,\ell})\}$, where $\forall i \in [\ell]$ $x_{1,i} = (\mathsf{K}_0, i, \ell, w_{0,i}, \mathsf{Trap}^1)$, with the new fields in $\mathsf{Trap}^1$ as shown in Figure 15.

| mode-real : $\perp$ | key-id : salt | $\mathsf{val}_0 : w_{0,i}$ | $\mathsf{val}_1 : w_{1,i}$ | SKE.K : $\perp$ | $\perp$ |
|---|---|---|---|---|---|
| mode-trap$_1$ : 1 | Target TS$_1$ : $T-2$ | Sym TS$_1$ : $T''$ | Sym val$_1$ : $\sigma^1_{T-2}$ | ST TS$_1$ : $T-3$ | ST val$_1$ : $\mathsf{q}^1_{T-2}$ |
| mode-trap$_2$ : 1 | Target TS$_2$ : $T-3$ | Sym TS$_2$ : $T'''$ | Sym val$_2$ : $\sigma^0_{T-3}$ | ST TS$_2$ : $T-4$ | ST val$_2$ : $\mathsf{q}^0_{T-3}$ |
| mode-trap$_3$ : $\perp$ | Target TS : $\perp$ | Sym TS : $\perp$ | $\perp$ | ST TS : $\perp$ | $\perp$ |

Figure 15: $\mathsf{Trap}^1$ configuration in $\mathcal{H}(3,1)$

(c) It sends the distribution pair to the $\mathsf{1FE_1}$ challenger and relays the response back to $\mathcal{A}$.

(d) To simulate a function key for $M$, $\mathcal{B}$ first requests for a function key to the $\mathsf{1FE_1}$ challenger for the function $\mathsf{ReRand}_{\mathsf{1FE_2.PK},\mathsf{salt},\mathsf{q_{st}},\perp,\perp}$ and receives $\mathsf{SK}_{\mathsf{ReRand}}$. $\mathcal{B}$ computes by itself $\mathsf{SK}_{\mathsf{Next}} \leftarrow \mathsf{1FE_2.KeyGen}(\mathsf{1FE_2.MSK}, \mathsf{Next}_{\mathsf{1FE_2.PK},\mathsf{salt},M,\mathsf{ct}_1,\mathsf{ct}_2})$ and returns a function key for $M$ as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ to $\mathcal{A}$.

Note that the (symbol, state) pair for time step $T-2$ has already been switched to correspond to $b=1$ from the prior hybrid. Thus, maintaining the trapdoor information for time step $T-1$ is now redundant and follows by normal decryption from time step $T-2$. The (symbol,state) pair for time step $T-3$ now corresponds to bit $b=0$ and therefore the decryption chain inconsistency arises at time step $T-2$ now. Hence, intuitively we "slide" the trapdoor by replacing a new trapdoor data structure in $\mathcal{H}(3,1)$ in a way that still maintains functional equivalence with $\mathcal{H}(2,8)$ at all the time steps but contains hardwired information about the time steps $T-3$ and $T-2$ now. We show the admissibility of the reduction $\mathcal{B}$ as follows.

Observe that for all time steps $t \notin \{T'', T-3, T''', T-4\}$, the decryption outputs are exactly the same sequence of ciphertexts in both the hybrids which are output by the normal decryption. At a time step $t \in \{T'', T-3, T''', T-4\}$ in $\mathcal{H}(3,1)$, the decryption is dictated by $\mathsf{Trap}^1.\mathsf{mode\text{-}trap}_1 = \mathsf{Trap}^1.\mathsf{mode\text{-}trap}_2 = 1$. In particular, the ciphertext components for time step $T-2$ corresponding to $b=1$ is output at time steps $T''$ and $T-3$ and is triggered by $\mathsf{Trap}^0.\mathsf{mode\text{-}trap}_3 = 1$ in $\mathcal{H}(2,8)$ and $\mathsf{Trap}^1.\mathsf{mode\text{-}trap}_1 = 1$ in $\mathcal{H}(3,1)$. On the other hand, the ciphertext components for time step $T-3$ corresponding to $b=0$ is output at time steps $T'''$ and $T-4$ and is triggered by the normal decryption in $\mathcal{H}(2,8)$ and by $\mathsf{Trap}^0.\mathsf{mode\text{-}trap}_2 = 1$ in $\mathcal{H}(3,1)$. The ciphertext components for time step $T-1$ corresponding to $b=1$ is output at time steps $T'$ and $T-2$ and is triggered by $\mathsf{Trap}^0.\mathsf{mode\text{-}trap}_1 = 1$ in $\mathcal{H}(2,8)$ and by the normal decryption (as a consequence of already having the outputs at time step $T-2$ switched to $b=1$) in $\mathcal{H}(3,1)$. Further, note that all these ciphertext components are exactly the same for both the hybrids.

Therefore, $\mathcal{B}$ is an admissible adversary against the $\mathsf{1FE_1}$ challenger since the outputs for the two challenge message sets are exactly the same. Hence $\mathcal{A}$ sees the distribution of $\mathcal{H}(2,8)$,

if $b = 0$, and that of $\mathcal{H}(3,1)$, if $b = 1$. Thus the advantage of $\mathcal{A}$ translates to the advantage of $\mathcal{B}$. □

Note that $\mathcal{H}(3,i)$ is analogous to $\mathcal{H}(2,i), \forall i \in [8]$. Now consider any pair of challenge message vectors $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$ of arbitrary length $\ell$ with any TM $M$ taking $T$ time steps to halt on either inputs. In general, we have that $\mathcal{H}(t,i)$ is analogous to $\mathcal{H}(t-1,i)$, for all $t \in [3, T-(\ell+1)], i \in [8]$. Observe further that for any given $k \in [3, T-(\ell+1)]$, we have the following computational indistinguishability chain via the intermediate hybrids.

$$\mathcal{H}(k-1,1) \overset{c}{\approx} \mathcal{H}(k-1,2) \overset{c}{\approx} \cdots \overset{c}{\approx} \mathcal{H}(k-1,8) \overset{c}{\approx} \mathcal{H}(k,1) \overset{c}{\approx} \mathcal{H}(k,2) \overset{c}{\approx} \cdots \overset{c}{\approx} \mathcal{H}(k,8)$$

We can easily extend this computational indistinguishability chain further to have the following.

$$\mathcal{H}(0) \overset{c}{\approx} \mathcal{H}(1,1) \overset{c}{\approx} \mathcal{H}(1,8) \overset{c}{\approx} \mathcal{H}(2,1) \overset{c}{\approx} \mathcal{H}(2,8) \overset{c}{\approx} \cdots \overset{c}{\approx} \mathcal{H}(T-(\ell+1),1) \overset{c}{\approx} \mathcal{H}(T-(\ell+1),8)$$

Note that in $\mathcal{H}(T-(\ell+1),8)$, the (symbol, state) pair corresponding to the output at time step $\ell+1$ has already been switched to $b = 1$. Proceeding one step backward in the execution chain we reach time step $\ell$ where the $\mathsf{1FE_2}$ ciphertext components are computed partially by each of $\mathsf{SK_{ReRand}}$ and $\mathsf{SK_{Next}}$. More specifically, at any time step $j \in [2,\ell]$ the $\mathsf{1FE_2}$ ciphertext component encoding the "symbol" $w_j$ is output by $\mathsf{ReRand}$. Accordingly, the $\mathsf{1FE_2}$ ciphertext component encoding the "state" $q_j$ for the same time step $j$ is output by $\mathsf{Next}$ only when it gets $(w_{j-1}, q_{j-1})$ as input, i.e., the symbol and state at time step $j-1$, each of which is encrypted with the exact same randomness. Hence, to proceed with the security proof at any time step $j \in [2,\ell]$, while switching from $b = 0$ to $b = 1$ the reduction $\mathcal{B}$ simulating $\mathsf{1FE_1}$ itself will now hardwire the $\mathsf{SKE}$ ciphertext encoding $\mathsf{1FE_2.CT}(w_{b,j})$ into $\mathsf{ReRand}$ and the $\mathsf{SKE}$ ciphertext encoding $\mathsf{1FE_2.CT}(\mathsf{q}_j^b)$ into $\mathsf{Next}$ after receiving them from the $\mathsf{1FE_2}$ challenger. At time step $j = 1$, $\mathcal{B}$ hardwires the $\mathsf{SKE}$ ciphertext encoding both $\mathsf{1FE_2.CT}(w_{b,1})$ and $\mathsf{1FE_2.CT}(\mathsf{q_{st}})$ into $\mathsf{ReRand}$ function only. This is since $\mathsf{ReRand}$ outputs the (symbol, state) ciphertext pair at the first time step as per functionality. Indistinguishability between these hybrids is as before.

Similar to the transition from $\mathcal{H}(2,8)$ to $\mathcal{H}(3,1)$, at time step $j = \ell+1$ we slide the trapdoor to switch the ciphertext in slot 1 for time step $\ell+1$ (corresponding to $b = 1$) and slot 2 for time step $\ell$ (corresponding to $b = 0$). We also set $\mathsf{Trap}^1.\mathsf{mode\text{-}trap}_3 = \perp, \mathsf{Trap}^1.\mathsf{mode\text{-}trap}_1 = \mathsf{Trap}^1.\mathsf{mode\text{-}trap}_2 = 1$. The decryption values being exactly the same in $\mathcal{H}(T-(\ell+1),8)$ and $\mathcal{H}(T-\ell,1)$, security follows from $\mathsf{1FE_1}$.

However, once we reach time step $\ell$ at $\mathcal{H}(T-\ell,8)$ when the bit $b$ (for time step $\ell$) has already been switched from 0 to 1 and we are about to slide the trapdoor to go to the next hybrid, we must add an additional hybrid $\mathcal{H}(T-j,9)$ for all $j \in [1,\ell]$, as discussed in Section 3.3, namely:

$\mathcal{H}(T-j,9)$ : In this hybrid, we modify the $\mathsf{1FE_1}$ challenge ciphertext in position $j$ as follows: the encoded message is changed corresponding to $b = 1$ and flag $\mathsf{mode\text{-}real} = 1$. The other flags $\mathsf{mode\text{-}trap}_1 = \mathsf{mode\text{-}trap}_2 = \mathsf{mode\text{-}trap}_3 = \perp$.

Consider $j = \ell$. Note that all ciphertexts previous to time step $\ell$ remain unchanged, and output their corresponding symbol ciphertexts correctly. The $\mathsf{Next}$ circuit outputs the state ciphertext for time step $\ell$ corresponding to bit $b = 1$. The only difference between this hybrid and the previous one is that here we use the real mode to output the symbol ciphertext for $b = 1$ whereas previously we used the trapdoor mode to output the same symbol CT. Hence, decryption values in both hybrids are exactly the same, and indistinguishability follows from security of $\mathsf{1FE_1}$.

As before from $\mathcal{H}(1,8)$ to $\mathcal{H}(2,5)$ and $\mathcal{H}(2,6)$ to $\mathcal{H}(3,1)$, we get two similar sequence of hybrids from $\mathcal{H}(T-(\ell+1),8)$ to $\mathcal{H}(T-\ell,5)$ and from $\mathcal{H}(T-\ell,6)$ to $\mathcal{H}(T-\ell,8)$. Additionally we now go from $\mathcal{H}(T-\ell,8)$ to $\mathcal{H}(T-(\ell-1),1)$ via the intermediate extra hybrid $\mathcal{H}(T-\ell,9)$ as follows.

$$\mathcal{H}(T-(\ell+1),8) \overset{\mathsf{1FE_1}}{\overset{c}{\approx}} \mathcal{H}(T-\ell,1) \overset{\mathsf{SKE}}{\overset{c}{\approx}} \mathcal{H}(T-\ell,2) \overset{\mathsf{1FE_1}}{\overset{c}{\approx}} \mathcal{H}(T-\ell,3) \overset{\mathsf{1FE_1}}{\overset{c}{\approx}} \mathcal{H}(T-\ell,4) \overset{\mathsf{cPRF}}{\overset{c}{\approx}} \mathcal{H}(T-\ell,5)$$

$$\mathcal{H}(T-\ell,6) \overset{\mathsf{cPRF}}{\overset{c}{\approx}} \mathcal{H}(T-\ell,7) \overset{\mathsf{1FE_1}}{\overset{c}{\approx}} \mathcal{H}(T-\ell,8) \quad \text{and} \quad \underbrace{\mathcal{H}(T-\ell,8) \overset{\mathsf{1FE_1}}{\overset{c}{\approx}} \mathcal{H}(T-\ell,9)} \overset{\mathsf{1FE_1}}{\overset{c}{\approx}} \mathcal{H}(T-(\ell-1),1)$$

In the following claims, we show a formal reduction between $\mathcal{H}(T-\ell,5)$ and $\mathcal{H}(T-\ell,6)$ and sketch a high level proof of computational indistinguishability for that of between $\mathcal{H}(T-\ell,8)$ and $\mathcal{H}(T-\ell,9)$ thereby connecting the above computational indistinguishability chains into one when the symbol and state at time step $\ell$ gets switched from $b=0$ to $b=1$ finally.

Claim B.18. If $\mathsf{1FE_2}$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(T-\ell,5)$ and $\mathcal{H}(T-\ell,6)$ are indistinguishable.

Proof. Given a $\mathsf{PPT}$ adversary $\mathcal{A}$ that distinguishes $\mathcal{H}(T-\ell,5)$ and $\mathcal{H}(T-\ell,6)$, we construct another $\mathsf{PPT}$ adversary $\mathcal{B}$ who breaks the security of the $\mathsf{1FE_2}$ scheme as follows.

1. $\mathcal{B}$ samples $(\mathsf{1FE_1.PK}, \mathsf{1FE_1.MSK}) \leftarrow \mathsf{1FE_1.Setup}(1^\lambda)$, $\mathsf{salt} \leftarrow \{0,1\}^\lambda$ and $\mathsf{K} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$ and gets $\mathsf{1FE_2.PK}$ from the $\mathsf{1FE_2}$ challenger. It sends $\mathsf{PK} = \mathsf{1FE_1.PK}$ to $\mathcal{A}$.

2. When $\mathcal{A}$ outputs a pair of challenge distributions $(\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ with support $\Sigma^\ell$ for any arbitrary $\ell = \mathsf{poly}(\lambda)$ and a function query $M$ which obeys the admissibility criteria, $\mathcal{B}$ does the following.

   (a) To simulate the challenge ciphertext, it first samples a pair of challenge messages $(\mathbf{w}_0, \mathbf{w}_1) \leftarrow (\mathcal{D}_0^\ell, \mathcal{D}_1^\ell)$ such that $\{\mathbf{w}_b\}_{b\in\{0,1\}} = \{(w_{b,1}, \ldots, w_{b,\ell})\}_{b\in\{0,1\}}$. It also samples a root cPRF key $\mathsf{K}_0 \leftarrow \mathsf{F.Setup}(1^\lambda)$.

   (b) $\mathcal{B}$ learns the two (symbol, state) pairs at time step $\ell$ when the oblivious TM $M$ is run on both $\mathbf{w}_0$ and $\mathbf{w}_1$. Denote these pairs as $(\sigma_\ell^0, \mathsf{q}_\ell^0)$ and $(\sigma_\ell^1, \mathsf{q}_\ell^1)$, where $\sigma_\ell^b = w_{b,\ell}$ now. $\mathcal{B}$ also learns the (symbol, state) pair at time step $\ell+1$ when $M$ is run on $\mathbf{w}_1$ and denote this pair as $(\sigma_{\ell+1}^1, \mathsf{q}_{\ell+1}^1)$. Further, it also records the time steps $(\ell, \ell-1)$ and $(\ell', \ell), \ell' \leq \ell$ when the individual components of the (symbol, state) pairs $(\sigma_\ell^b, \mathsf{q}_\ell^b), \forall b \in \{0,1\}$ and $(\sigma_{\ell+1}^1, \mathsf{q}_{\ell+1}^1)$ respectively are generated and then computes a root key punctured at point $(\ell \| \mathsf{salt})$ as $\mathsf{K}_0^\ell = \mathsf{F.Constrain}(\mathsf{K}_0, (\ell \| \mathsf{salt}))$. It then simulates the encryption oracle by computing $\mathsf{CT}_i = \mathsf{1FE_1.Enc}(\mathsf{1FE_1.PK}, x_{1,i})$, where $\forall i \in [\ell], x_{1,i} = (\mathsf{K}_0^\ell, i, \ell, w_{0,i}, \mathsf{Trap}^1)$ and $\mathsf{Trap}^1$ is as per $\mathcal{H}(T-\ell,3)$ shown in Figure 16. It returns the ciphertext $\mathsf{CT} = \{\mathsf{CT}_i\}_{i\in[\ell]}$ to $\mathcal{A}$.

| | | | | | |
|---|---|---|---|---|---|
| mode-real : $\perp$ | key-id : salt | $\mathsf{val}_0 : w_{0,i}$ | $\mathsf{val}_1 : w_{1,i}$ | SKE.K : K | $\perp$ |
| $\mathsf{mode\text{-}trap}_1 : 1$ | Target $\mathsf{TS}_1 : \ell+1$ | Sym $\mathsf{TS}_1 : \ell'$ | Sym $\mathsf{val}_1 : \sigma_{\ell+1}^1$ | ST $\mathsf{TS}_1 : \ell$ | ST $\mathsf{val}_1 : \mathsf{q}_{\ell+1}^1$ |
| $\mathsf{mode\text{-}trap}_2 : \perp$ | Target $\mathsf{TS}_2 : \perp$ | Sym $\mathsf{TS}_2 : \perp$ | Sym $\mathsf{val}_2 : \perp$ | ST $\mathsf{TS}_2 : \perp$ | ST $\mathsf{val}_2 : \perp$ |
| $\mathsf{mode\text{-}trap}_3 : 1$ | Target TS : $\ell$ | Sym TS : $\ell$ | $\perp$ | ST TS : $\ell-1$ | $\perp$ |

Figure 16: $\mathsf{Trap}^1$ configuration in $\mathcal{H}(T-\ell,3)$

   (c) To simulate a function key for $M$, $\mathcal{B}$ does the following.

      i. In order to construct a function key for ReRand and Next, $\mathcal{B}$ now needs to hardwire an SKE ciphertext in each of the functions which it computes with the help of $\mathsf{1FE_2}$ challenger as follows.

- Delegate the punctured root key to compute $\mathsf{K}^{\ell}_{\ell+1} = \mathsf{F.KeyDel}(\mathsf{K}^{\ell}_0, f_{\ell+1})$.
- Create a $\mathsf{1FE_2}$ challenge message pair as $((\mathbf{z}^0_1, \mathbf{z}^0_2), (\mathbf{z}^1_1, \mathbf{z}^1_2))$ such that $\forall b \in \{0,1\}$, $\mathbf{z}^b_1 = (\mathsf{SYM}, \mathsf{salt}, \mathsf{K}^{\ell}_{\ell+1}, \ell, \ell, \sigma^b_{\ell}, \mathsf{Trap}^1)$ and $\mathbf{z}^b_2 = (\mathsf{ST}, \mathsf{q}^b_{\ell})$.
- It sends the challenge message pair $((\mathbf{z}^0_1, \mathbf{z}^0_2), (\mathbf{z}^1_1, \mathbf{z}^1_2))$ to the $\mathsf{1FE_2}$ challenger and gets back $(\mathsf{CT}_{\mathsf{sym},\ell}, \mathsf{CT}_{\mathsf{st},\ell})$.
- It then computes the two $\mathsf{SKE}$ ciphertexts $\mathsf{ct}_1 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}_{\mathsf{sym},\ell})$ and $\mathsf{ct}_2 = \mathsf{SKE.Enc}(\mathsf{K}, \mathsf{CT}_{\mathsf{st},\ell})$.

ii. $\mathcal{B}$ now computes by itself $\mathsf{SK}_{\mathsf{ReRand}} \leftarrow \mathsf{1FE_1.KeyGen}(\mathsf{1FE_1.MSK}, \mathsf{ReRand}_{\mathsf{1FE_2.PK},\mathsf{salt},\mathsf{q}_{\mathsf{st}},\mathsf{ct}_1,\perp})$.

iii. $\mathcal{B}$ receives $\mathsf{SK}_{\mathsf{Next}}$ for requesting a function key for $\mathsf{Next}_{\mathsf{1FE_2.PK},\mathsf{salt},\mathsf{M},\perp,\mathsf{ct}_2}$ to the $\mathsf{1FE_2}$ challenger and returns a function key for $M$ as $\mathsf{SK}_M = (\mathsf{SK}_{\mathsf{ReRand}}, \mathsf{SK}_{\mathsf{Next}})$ to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a guess, $\mathcal{B}$ does the same.

Note that the function key queried by $\mathcal{B}$ to the $\mathsf{1FE_2}$ challenger is for a function $\mathsf{Next}$ that outputs $\mathsf{1FE_2}$ ciphertexts that are indistinguishable by the security of $\mathsf{1FE_2}$ itself. Therefore, $\mathcal{B}$ is an admissible $\mathsf{1FE_2}$ adversary. Further, when the ciphertext for time step $\ell$ is computed as a $\mathsf{1FE_2}$ encryption of a (symbol, state) pair corresponding to bit $b = 0$, $\mathcal{A}$'s view is identical to that of $\mathcal{H}(T-\ell, 5)$, and when the ciphertext for time step $\ell$ is computed as a $\mathsf{1FE_2}$ encryption of a (symbol, state) pair corresponding to bit $b = 1$, $\mathcal{A}$'s view is identical to that of $\mathcal{H}(T-\ell, 6)$. Hence the advantage of $\mathcal{A}$ in distinguishing $\mathcal{H}(T-\ell, 5)$ and $\mathcal{H}(T-\ell, 6)$ translates to the advantage of $\mathcal{B}$ in breaking the $\mathsf{1FE_2}$ scheme. □

Claim B.19. If $\mathsf{1FE_1}$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(T-\ell, 8)$ and $\mathcal{H}(T-\ell, 9)$ are indistinguishable.

Proof. We describe the proof at a high level and omit the details. Note that all ciphertexts previous to time step $\ell$ remain unchanged, and output their corresponding symbol ciphertexts correctly. The $\mathsf{Next}$ circuit outputs the state ciphertext for time step $\ell$ corresponding to bit $b = 1$. The only difference between this hybrid and the previous one is that here we use the real mode to output the symbol ciphertext for $b = 1$ whereas previously we used the trapdoor mode to output the same symbol ciphertext. Hence, decryption values in both hybrids are exactly the same. When the $\mathsf{1FE_1}$ ciphertext for time step $\ell$ is computed corresponding to $b = 0$, $\mathcal{A}$'s view is identical to that of $\mathcal{H}(T-\ell, 8)$, and when the $\mathsf{1FE_1}$ ciphertext for time step $\ell$ is computed corresponding to $b = 1$, $\mathcal{A}$'s view is identical to that of $\mathcal{H}(T-\ell, 9)$. Hence the advantage of $\mathcal{A}$ in distinguishing $\mathcal{H}(T-\ell, 8)$ and $\mathcal{H}(T-\ell, 9)$ translates to the advantage of $\mathcal{B}$ in breaking the $\mathsf{1FE_1}$ scheme. □

Denoting $\tau = (T-j)$ for any $j \in [\ell]$, we get a sequence of hybrids shown below, where we define $\mathcal{H}(T, 1) \triangleq \mathcal{H}(T)$ and have the final Claim B.20 which completes the proof of Theorem 3.1.

$$\mathcal{H}(\tau, 8) \overset{\mathsf{1FE_1}}{\underset{c}{\approx}} \mathcal{H}(\tau, 9) \overset{\mathsf{1FE_1}}{\underset{c}{\approx}} \mathcal{H}(\tau+1, 1) \overset{\mathsf{SKE}}{\underset{c}{\approx}} \cdots \overset{\mathsf{cPRF}}{\underset{c}{\approx}} \mathcal{H}(\tau+1, 5) \overset{\mathsf{1FE_2}}{\underset{c}{\approx}} \mathcal{H}(\tau+1, 6) \overset{\mathsf{cPRF}}{\underset{c}{\approx}} \cdots \overset{\mathsf{1FE_1}}{\underset{c}{\approx}}$$
$$\mathcal{H}(\tau+1, 8) \overset{\mathsf{1FE_1}}{\underset{c}{\approx}} \mathcal{H}(\tau+1, 9) \overset{\mathsf{1FE_1}}{\underset{c}{\approx}} \mathcal{H}(\tau+2, 1)$$

Claim B.20. If $\mathsf{1FE_1}$ is a secure $\mathsf{CktFE}$ scheme, then hybrids $\mathcal{H}(T-1, 9)$ and $\mathcal{H}(T)$ are indistinguishable.

Proof. Note that $\mathsf{Trap}^1.\mathsf{mode\text{-}real} = 1$ for ciphertexts in both worlds. The only difference between both these hybrids is that in the former $\mathsf{Trap}$ contains other information whereas in the latter all other fields disabled with $\perp$. However, since $\mathsf{Trap}^1.\mathsf{mode\text{-}real} = 1$, these fields anyway play no role in decryption, so the decryption values stay the same.

Selective Security. The above proof shows security as per the weak selective definition, in which the adversary submits the challenge messages and keys at the same time. This can be easily strengthened to selective security in which the key requests can be made after seeing the challenge ciphertext. Since the full selective game requires an additional trapdoor structure, we did not show it here for ease of exposition, as the current proof is already quite complex. Note that currently, the proof is restricted to weak selective because in order to program the symbol and state messages for some time step in the $\mathsf{Trap}$ data structure, the machine which produces these symbol, state pairs must be specified. This dependency may be easily overcome by instead having an additional trapdoor data structure in the key, which contains the above information. Thus, the challenge ciphertext can be programmed without knowledge of the keys, and selective security can be achieved. We defer details to the full version of the paper. □

□

# C Missing Details in Proof of Theorem 4.1

The modified trapdoor data structure is shown in Figure 17. There is an additional field that records the global salt value.

| mode-real | key-id | global-salt | $\mathsf{val}_0$ | $\mathsf{val}_1$ | SKE.K |
|---|---|---|---|---|---|
| $\mathsf{mode\text{-}trap}_1$ | Target $\mathsf{TS}_1$ | Sym $\mathsf{TS}_1$ | Sym $\mathsf{val}_1$ | ST $\mathsf{TS}_1$ | ST $\mathsf{val}_1$ |
| $\mathsf{mode\text{-}trap}_2$ | Target $\mathsf{TS}_2$ | Sym $\mathsf{TS}_2$ | Sym $\mathsf{val}_2$ | ST $\mathsf{TS}_2$ | ST $\mathsf{val}_2$ |
| $\mathsf{mode\text{-}trap}_3$ | Target TS | Sym TS | $\perp$ | ST TS | $\perp$ |

Figure 17: Data Structure $\mathsf{Trap}$ used for Proof

The Hybrids. We consider the case where the adversary makes a single key query but makes $Q$ ciphertext queries in each co-ordinate. We assume a lexicographic ordering over the $Q^k$ global salt values, and denote by $\mathsf{gsalt}_j$ the $j^{th}$ member of this sequence.

$\mathcal{H}(0)$: This is the real world, when $\mathsf{mode\text{-}real} = 1$ and $\mathsf{mode\text{-}trap}_1 = \mathsf{mode\text{-}trap}_2 = \mathsf{mode\text{-}trap}_3 = \perp$ for all ciphertexts.

For $j \in [Q^k]$, do:

$\mathcal{H}(j, 1, 1)$: In this world, all ciphertexts (constructed by the encryptor as well as function keys) have $\mathsf{mode\text{-}real} = \perp$, $\mathsf{mode\text{-}trap}_1 = 1$, $\mathsf{mode\text{-}trap}_2 = 1$, $\mathsf{mode\text{-}trap}_3 = \perp$. We program the last link in the decryption chain corresponding to $\mathsf{gsalt}_j$ for switching bit $b$ by setting:

$$\mathsf{Target}\ \mathsf{TS}_1 = T - 1, \mathsf{Target}\ \mathsf{TS}_2 = T - 2$$

The fields $\mathsf{Sym}\ \mathsf{TS}_1$ and $\mathsf{ST}\ \mathsf{TS}_1$ contain the time steps when the symbol and state ciphertext pieces are generated for time step $T - 1$, and the fields $\mathsf{Sym}\ \mathsf{val}_1$ and $\mathsf{ST}\ \mathsf{val}_1$ contain the symbol and state values which must be encrypted by the function key in the above time steps when $\mathsf{mode\text{-}trap}_1$ is set.

Indistinguishability follows from security of $\mathsf{kFE}$, since the decryption values in both hybrids are exactly the same.

$\mathcal{H}(j, 1, 2)$: Hardwire the $\mathsf{Next}$ key with an $\mathsf{SKE}$ encryption of symbol and state ciphertexts output at step $T-1$ corresponding to execution thread $\mathsf{gsalt}_j$ for $b = 0$. Use the same ciphertexts would be generated in the previous hybrid.

Indistinguishability follows from security of $\mathsf{SKE}$, since the only difference is the value of the message encrypted using $\mathsf{SKE}$ which is embedded in the key.

$\mathcal{H}(j, 1, 3)$: Set $\mathsf{mode\text{-}trap}_1 = \bot$, $\mathsf{mode\text{-}trap}_2 = 1$, $\mathsf{mode\text{-}trap}_3 = 1$ and $\mathsf{Target\ TS} = T - 1$. In this hybrid the hardwired value in the key is used to be output as step $T-1$ ciphertext corresponding to execution thread $\mathsf{gsalt}_j$.

Indistinguishability follows from security of $\mathsf{kFE}$, since the decryption values in both hybrids are exactly the same.

$\mathcal{H}(j, 1, 4)$: Change normal root key $\mathsf{K}_0$ to punctured root key $\mathsf{K}_0^{T-1}$ which punctures all delegated keys at point $(T-1 \| \mathsf{key\text{-}id} \| \mathsf{gsalt}_j)$.

Indistinguishability follows from security of $\mathsf{kFE}$.

$\mathcal{H}(j, 1, 5)$: Switch the randomness in the $\mathsf{1FE}$ ciphertexts which are hardwired in the key to true randomness.

Indistinguishability follows from security of punctured $\mathsf{cPRF}$ for the aforementioned function family, since the remainder of the distribution only uses the punctured key.

$\mathcal{H}(j, 1, 6)$: Switch the value encoded in the $\mathsf{1FE}$ ciphertexts which are hardwired in the key to correspond to $b = 1$.

Indistinguishability follows from security of $\mathsf{1FE}$.

$\mathcal{H}(j, 1, 7)$: Switch randomness back to PRF randomness in the ciphertext hardwired in key, using the punctured key for all but the hardwired ciphertext.

Indistinguishability follows from security of $\mathsf{cPRF}$ as discussed above.

$\mathcal{H}(j, 1, 8)$: Switch the punctured root key to the normal root key.

Indistinguishability follows from security of $\mathsf{kFE}$ as discussed above.

$\mathcal{H}(j, 2, 1)$: Switch ciphertext in slot 1 for target $T-1$ to be for $b = 1$. Slot 2 remains $b = 0$. Set $\mathsf{mode\text{-}trap}_3 = \bot$ and $\mathsf{mode\text{-}trap}_1 = \mathsf{mode\text{-}trap}_2 = 1$.

Indistinguishability follows from security of $\mathsf{kFE}$, since the decryption values in both hybrids are exactly the same.

$\mathcal{H}(j, 2, 2)$: Hardwire key with $\mathsf{SKE}$ encryption of $\mathsf{1FE}$ ciphertext for time step $T-2$ and bit $b = 0$ (same as hybrid $(1, 2)$ but for $T-2$).

Indistinguishability follows from security of $\mathsf{SKE}$ as above.

$\mathcal{H}(j, 2, 3)$: Set $\mathsf{mode\text{-}trap}_1 = 1$ with target $T-1$, $\mathsf{mode\text{-}trap}_2 = \bot$, and $\mathsf{mode\text{-}trap}_3 = 1$ with target $T-2$.

Indistinguishability follows from security of $\mathsf{kFE}$, since the decryption values in both hybrids are exactly the same.

$\mathcal{H}(j, 2, 4)$: Switch normal root key to punctured key at position $T - 2$.

Indistinguishability follows from security of $\mathsf{kFE}$ as discussed above.

$\mathcal{H}(j, 2, 5)$: Switch randomness to true in the ciphertext hardwired in key.

Indistinguishability follows from security of $\mathsf{cPRF}$ as discussed above.

$\mathcal{H}(j, 2, 6)$: Switch hardwired $\mathsf{1FE}$ ciphertext for step $T - 2$ to correspond to bit $b = 1$.

Indistinguishability follows from security of $\mathsf{1FE}$.

$\mathcal{H}(j, 2, 7)$: Switch randomness back to use the $\mathsf{PRF}$ in the ciphertext hardwired in key.

Indistinguishability follows from security of $\mathsf{cPRF}$ as discussed above.

$\mathcal{H}(j, 2, 8)$: Switch punctured root key to normal root key.

Indistinguishability follows from security of $\mathsf{kFE}$ as discussed above.

$\mathcal{H}(j, 3, 1)$: Intuitively, we slide the trapdoor left by one step, i.e. change target time-steps to $T - 2$ and $T - 3$ in the ciphertext. Now slot 1 for $T - 2$ corresponds to $b = 1$ and slot 2 for $T - 3$ to $b = 0$. Set $\mathsf{mode\text{-}real} = \mathsf{mode\text{-}trap}_3 = \bot$ and $\mathsf{mode\text{-}trap}_1 = \mathsf{mode\text{-}trap}_2 = 1$.

Indistinguishability follows from security of $\mathsf{kFE}$, since the decryption values in both hybrids are exactly the same. Note that now slot $T-1$ is redundant, since $T-2$ ciphertext is already switched to $b = 1$.

Hybrid $\mathcal{H}(j, 3, i)$ will be analogous to $\mathcal{H}(j, 2, i)$ for $i \in [8]$.

As we proceed left in the execution chain one step at a time, we reach step $\ell$ where $\ell = |\mathbf{w}|$, i.e. time steps for which $\mathsf{kFE}$ ciphertexts are provided by the encryptor. At this point we will hardwire the $\mathsf{Agg}$ key instead for the symbol ciphertexts and the $\mathsf{Next}$ key for the state ciphertexts with the exception at time step 1 when we will hardwire both the symbol ciphertext and the start state ciphertext in $\mathsf{Agg}$ key itself.

After going through all the global salt values and all the key values, we replace the challenge ciphertext to have $\mathsf{mode\text{-}real} = 1$ and message corresponding to $b = 1$, one step at a time. This is analogous to the case of single input $\mathsf{TMFE}$, except that we must additionally track global salt values.

$\mathcal{H}(T)$: In this hybrid all ciphertexts have $\mathsf{mode\text{-}real} = 1$, all other trapdoor information is set to $\bot$ and $b = 1$ is used. This is the real world with $b = 1$.

Indistinguishability from $\mathcal{H}(j + 1, 1, 1)$ follows from security of $\mathsf{kFE}$ since the decryption values in both hybrids are exactly the same.

# D Constrained PRF for our Function Family

Our proof makes use of a set of delegatable constrained pseudorandom functions ($\mathsf{cPRF}$). We require $T$ delegatable $\mathsf{cPRF}$s, denoted by $\mathsf{F}_i$ for $i \in [T]$, where each $\mathsf{cPRF}$ in turn supports $T$ delegations. The sequence of delegated keys for $\mathsf{F}_i$ are denoted by $\{K_{i,t}\}_{i,t\in[T]}$, corresponding to functions $f_{i,t}$, such that the satisfying set of $f_{i,t+1}$ is strictly contained within the satisfying set of $f_{i,t}$, for all $i, t \in [T]$.

In more detail, for any polynomial $\mathsf{poly}(\lambda)$, define $f_{i,t} : \{0,1\}^{\lambda+\mathsf{poly}(\lambda)} \to \{0,1\}$ as follows.

$$f_{i,t}(x\|z) = 1 \quad if \;\; x \geq t \;\wedge (x\|z) \neq i$$
$$= 0 \quad otherwise$$

Thus, the root key (and hence all delegated keys) of $\mathsf{F}_i$ are punctured at the point $i$.

Overview. We provide a construction for a cPRF $\mathsf{F}$ which supports puncturing and delegation as required; the $T$ cPRFs $\mathsf{F}_i$ for $i \in [T]$ may each be constructed similarly. To begin, note that we require the root key of $\mathsf{F}$ to be punctured at a point $i^*$ (say). The cPRF construction for punctured PRF [BW13, KPTZ13, BGI14] (which is in turn inherited from the standard PRG based GGM [GGM86]) immediately satisfies this constraint, so we are left with the question of delegation.

Recall that we are required to delegate $T$ times, where $T$ is the (polynomial) runtime of the Turing machine on the encrypted input (please see Section 3), and the $j^{th}$ delegated key must support evaluation of points $\{(k\|z) : z \in \{0,1\}^\lambda\}$ for $k \geq j$, except when $(k\|z) = i^*$. This may be viewed as the $j^{th}$ key being punctured on points $[1, j-1] \cup i^*$. We show that the GGM based construction for puncturing a single point can be extended to puncturing an interval (plus an extra point). Intuitively, puncturing an interval corresponds to puncturing at most $\lambda$ internal nodes in the GGM tree. In more detail, we show that regardless of the value of $j$, it suffices to puncture at most $\lambda$ points in the GGM tree to achieve puncturing of the entire interval $[1, j-1]$.

Construction. Formally, the cPRF $\mathsf{F}$ is defined as follows. Our constrain algorithm takes as input the set of points on which to puncture the PRF, as opposed to the satisfying set. We compute the GGM tree as in Figure 18 and number the leaves from 1 to $2^{(\lambda+\mathsf{poly}(\lambda))}$.

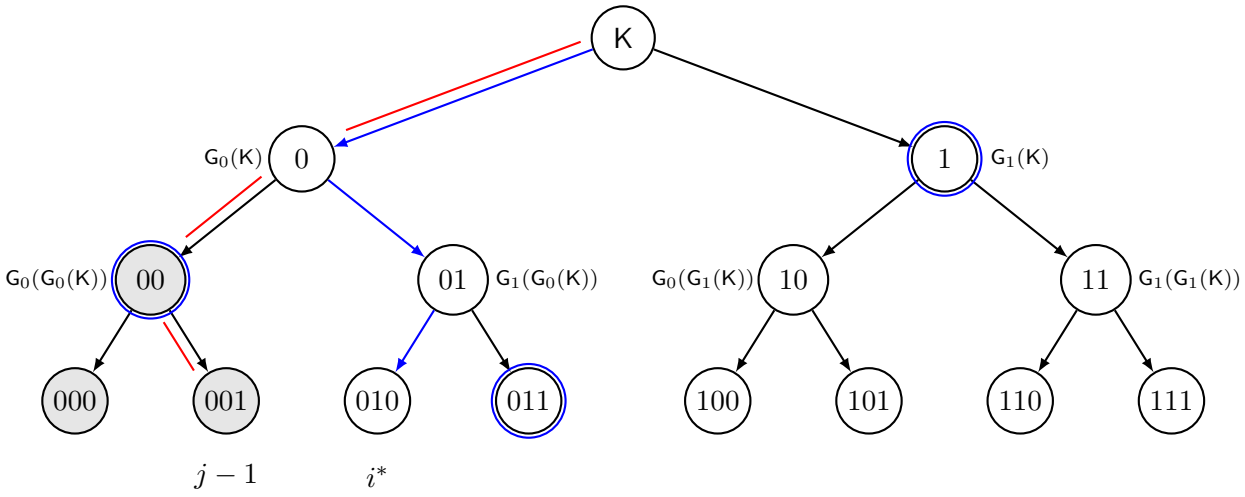


Figure 18: To puncture $i^* = 010$ draw path from root to $i^*$ and reveal nodes that are siblings along the path. To puncture interval $[1, 2] \cup \{i^*\} = \{000, 001\} \cup \{010\}$, compute the set $\mathsf{Grey} = \{000, 001, 00\}$ and the punctured set $\mathcal{P} = \{00, 010\}$. Further compute the initial revealed set $\mathcal{R}_0 = \{(1, 01), (1, 00, 011)\}$ and replace 00 and 01 by 011 to get the final revealed set $\mathcal{R}_f = \{1, 011\}$.

$\mathsf{Setup}(1^\lambda)$: Sample a standard length doubling PRG $G$ with seed $\mathbf{s}_0$. Output $\mathsf{pk} = G$ and $\mathsf{K}_0 = \mathbf{s}_0$. As usual, we will denote by $G_0$ the first half of the PRG output and by $G_1$ the second half.

$\mathsf{Constrain}(\mathsf{K}_0, [1, j-1] \cup i^*)$: Upon input the root key $\mathsf{K}_0$ and the set of points to be punctured, do the following:

1. Compute puncturing set $\mathcal{P}$: Initialize $\mathcal{P}$ to contain the point $i^*$. Compute the path from the root node to node corresponding to point $j-1$. For any right edge $(a, b)$ along the path, mark the left child of $a$ grey. Mark the final node $j-1$ grey. At

this point we have a set of grey nodes which must be punctured. Minimize this set by checking whether both children of a node are grey, in which case, also mark the parent grey. Finally, add the grey nodes which do not have grey parents to a set $\mathcal{P}$.

2. Computing revealed set $\mathcal{R}$: For every node in the set $\mathcal{P}$, compute the punctured key (as in GGM) as follows. For every node $k \in \mathcal{P}$, compute the path from the root to $k$, and add the siblings of all nodes along the path to the set $\mathcal{R}$ [14]. Trim this set so as to remove conflicts caused by overlapping paths as follows: if any punctured node $b$ in $\mathcal{P}$ is a descendent of some node $a$ in $\mathcal{R}$, remove $a$ from $\mathcal{R}$, compute the path from $a$ to $b$ and add all the siblings of the nodes on this path to $\mathcal{R}$. Repeat until there are no more changes to $\mathcal{R}$.
3. Output $\mathsf{K}_j = \mathcal{R}$.

$\mathsf{KeyDel}(\mathsf{K}_j, f_{j+1})$: Given the punctured key for set $[1, j]$, compute the punctured key for $[1, j+1]$ as follows. Note that it suffices to delegate from $j$ to $j+1$ to imply delegation from $j$ to any $j'$ for $j' > j$.

1. Consider the case when $j$ is a left child and $j+1$ is a right child of the same parent. In this case, the set $\mathsf{K}_j$ contains the node corresponding to $j+1$. Delete this node and return the resultant set as $\mathsf{K}_{j+1}$.
2. Consider the case when $j$ is a right child and $j+1$ is a left child of the neighbouring parent. In this case $\mathsf{K}_j$ contains the parent of node $j+1$. Use the parent to evaluate the value corresponding to node $j+2$, remove the parent and add the value corresponding to node $j+2$.

$\mathsf{Eval}(\mathsf{K}_j, y)$: Evaluate the GGM tree on input $y$ as $G_{y_1} \circ \ldots \circ G_{y_n}(\mathbf{s}_0)$ and output it. Note that $\mathsf{K}_j$ contains enough information to compute the path from root to $y$ as long as $y$ is supported by $\mathsf{K}_j$.

Correctness. We argue correctness of the $\mathsf{Constrain}$ algorithm first. To begin, we claim that to puncture the interval $[1, j-1]$, it suffices to compute the path from the root node to $j-1$, and puncture the left siblings of any right edges along the path, i.e. if $(a, b)$ is a right edge along the path, we puncture the left child of $a$. Since a descendent of the left child of $a$ must necessarily have value $< j-1$, it is necessary to puncture these nodes. Moreover it is sufficient, along with $j-1$ to puncture these nodes, because i) any node of value $< j-1$ must have an ancestor, say $a_{j-1}$ which lies along the path $P$ from root to $j-1$ ii) If $(a_{j-1}, b_{j-1}) \in P$ for some $b_{j-1}$, then $(a_{j-1}, b_{j-1})$ is a right edge. Since $\mathsf{Constrain}$ algorithm populates $\mathcal{P}$ with this set of points and then minimizes this set, we have that $\mathcal{P}$ represents the punctured points in the tree. Next, we argue that the nodes returned via the set $\mathcal{R}$ is correct: to see this, note that $\mathcal{R}$ is initially populated with all the constrained keys for each punctured point in $\mathcal{P}$, and this set is trimmed to remove conflicts caused by overlapping paths. Thus, the resultant nodes returned in the set $\mathcal{R}$ capture the intersection of points whose evaluation is admitted by each punctured key.

Finally, note that the $\mathsf{Constrain}$ algorithm runs in polynomial time: this is because we may use binary search to compute the path from the root to any node in the graph, and all operations deal with listing the siblings along these paths which take $O(\mathsf{poly}(\lambda))$. Moreover, we note that there is at most one punctured point at every level for any interval $[1, j-1]$, which implies that the total runtime of $\mathsf{Constrain}$ is $O((\mathsf{poly}(\lambda))^2)$.

[14] Note that this is exactly the constrained key provided for a single punctured point in the GGM based construction.

Correctness of Eval is immediate, since evaluation is exactly the same as GGM evaluation. Correctness of KeyDel is also straightforward, since we only delegate one step at a time, hence it suffices to simply puncture one additional node corresponding to a point $j+1$, which is either the right child of the same parent as $j$, or the left child of the neighbouring parent. Puncturing a single node is immediate in either of these cases, as described in KeyDel above.

Security. We argue that given a punctured key, an adversary cannot distinguish a pseudorandom value from a random value on any input $y$ that is not supported by the punctured key. Since by construction of the constrained key, the adversary does not possess any node along the path from the root to the node corresponding to $y$, we have that the node corresponding to $y$ is pseudorandom by the standard hybrid argument for GGM security.

# E Constructing DI Secure Functional Encryption

Let 1FE be a single input functional encryption scheme which satisfies standard indistinguishability based security. We will construct a single input functional encryption scheme DiFE satisfying distributional indistinguishability as shown below. Our proof follows the strategy of embedding a hidden thread in the functionality which is only active during simulation [CIJ+13, ABSV15]. We note that the scheme presented below is public key, but directly lends itself to a private key version by instead relying on private key 1FE.

$\mathsf{DiFE.Setup}(1^\lambda, 1^n)$: Upon input the security parameter and length of input message, do the following:

1. Invoke $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{1FE.Setup}(1^\lambda, 1^{n+\lambda+1})$ and output $(\mathsf{PK}, \mathsf{MSK})$.

$\mathsf{DiFE.Enc}(\mathsf{PK}, \mathbf{x})$: Upon input the public key PK and a vector $\mathbf{x} \in \mathcal{X}^n$, do the following:

1. Output $\mathsf{CT}_\mathbf{x} = \mathsf{1FE.Enc}\big(\mathsf{PK}, (\mathbf{x}, \mathbf{0}, 0)\big)$.

$\mathsf{DiFE.KeyGen}(\mathsf{MSK}, f)$: Upon input the master secret key MSK and a circuit $f$, do the following:

1. Choose CT randomly from the space of Sym ciphertexts.
2. Output $\mathsf{SK}_f = \mathsf{1FE.KeyGen}(\mathsf{MSK}, f')$ where $f'$ is as defined in Figure 19.

| Functionality $f'_{f,\mathsf{CT}}(\mathbf{x}, \mathsf{Sym.K}, \mathsf{mode})$ |
|---|
| If $\mathsf{mode} = 0$, output $y = f(\mathbf{x})$ else output $y = \mathsf{Sym.Dec}(\mathsf{K}, \mathsf{CT})$. |

Figure 19: Functionality $f'_{f,\mathsf{CT}}$

$\mathsf{DiFE.Dec}(\mathsf{PK}, \mathsf{CT}_\mathbf{x}, \mathsf{SK}_f)$: Upon input the public key PK, a ciphertext $\mathsf{CT}_\mathbf{x}$ and a function key $\mathsf{SK}_f$, compute $\mathsf{1FE.Dec}(\mathsf{PK}, \mathsf{CT}_\mathbf{x}, \mathsf{SK}_f)$ and output it.

Correctness. We have by correctness of 1FE that decryption recovers $f(\mathbf{x})$ as desired.

Proof of Security.

Next, we argue that the DiFE scheme constructed above is secure.

Theorem E.1. Assume that 1FE is an FE scheme that satisfies standard indistinguishability based security and that Sym is a secure symmetric key encryption scheme. Then, the DiFE scheme constructed above satisfies distributional indistinguishability based security.

Proof. The proof proceeds via a sequence of hybrids where the first hybrid corresponds to an encryption of vector $\mathbf{x}_0$ chosen from distribution $D_0$ and the last hybrid corresponds to an encryption of vector $\mathbf{x}_1$ chosen from distribution $D_1$.

Hybrid 0: This is the real world with $\mathbf{x}_0 \leftarrow D_0$.

Hybrid 1: In this world, we hardwire the output of the function $y = f(\mathbf{x}_0)$, where $\mathbf{x}_0 \leftarrow D_0$ into the function key using symmetric key encryption. That is, let $\mathsf{CT} = \mathsf{Sym.Enc}(\mathsf{Sym.K}, \mathsf{y})$.

Hybrid 2: In this world, change the message in the ciphertext, i.e. message encoded is $(\perp, \mathsf{Sym.K}, \mathsf{mode} = 1)$.

Hybrid 3: In this world, we change the value of $y$ to $y = f(\mathbf{x}_1)$.

Hybrid 4: In this world, we change the message encrypted to $(\mathbf{x}_1, \mathbf{0}, \mathsf{mode} = 0)$ where $\mathbf{x}_1 \leftarrow D_1$.

Hybrid 5: In this world, we change the value of CT hardwired in the key back to random.

Next, we argue that consecutive hybrids are indistinguishable.

Lemma E.2. Hybrids 0 and 1 are indistinguishable assuming the security of Sym.

Proof. The only thing that changes between Hybrid 0 and 1 is the choice of CT, so that in the former it is chosen randomly and in the latter case it is an honest encryption of the scheme Sym. Given an adversary $\mathcal{A}$ who distinguishes between Hybrid 0 and Hybrid 1, we construct an adversary $\mathcal{B}$ who breaks the semantic security of Sym.

$\mathcal{B}$ generates the public key honestly and returns it to $\mathcal{A}$. When $\mathcal{A}$ outputs two challenge distributions $D_0, D_1$, $\mathcal{B}$ samples $\mathbf{x}_0 \leftarrow D_0$. It honestly computes ciphertexts for $(\mathbf{x}_0, \mathbf{0}, \mathsf{mode} = 0)$ and returns these to $\mathcal{A}$. When $\mathcal{A}$ requests a function key $f$, $\mathcal{B}$ computes the value $y = f(\mathbf{x}_0)$, and sends $y$ to the Sym challenger. The Sym challenger responds with CT which is either an honest encryption of $y$ or an element chosen randomly from the ciphertext space. $\mathcal{B}$ uses CT in constructing the function key and returns this to $\mathcal{A}$. Now, if CT is random, $\mathcal{A}$ sees the view of Hybrid 0 and if it is an encryption of $y$, it sees the view of Hybrid 1. □

Lemma E.3. Hybrids 1 and 2 are indistinguishable assuming the security of 1FE.

Proof. The only difference between Hybrids 1 and 2 is that in the former the encrypted message is $(\mathbf{x}_0, \mathbf{0}, \mathsf{mode} = 0)$ and in the latter it is $(\perp, \mathsf{Sym.K}, \mathsf{mode} = 1)$. Assume there is an adversary $\mathcal{A}$ who distinguishes between Hybrid 1 and Hybrid 2, we construct an adversary $\mathcal{B}$ who can break the security of 1FE.

$\mathcal{B}$ does the following:

1. It obtains the public key from the 1FE challenger and returns this to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs two distribution pairs $(D_0, D_1)$, it samples $\mathbf{x}_0 \leftarrow D_0$, Sym.K and returns challenges $(\mathbf{x}_0, \mathbf{0}, \mathsf{mode} = 0)$ and $(\perp, \mathsf{Sym.K}, \mathsf{mode} = 1)$ to the 1FE challenger. It obtains an encryption of one of them chosen at random and returns this to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs a function $f$, $\mathcal{B}$ constructs the function $f'$ as described in Figure 19 and sends this to the $\mathsf{1FE}$ challenger. Here, $\mathsf{CT}$ is computed as $\mathsf{Sym.Enc}(\mathsf{Sym.K}, \mathsf{y})$ where $y = f(\mathbf{x}_0)$. It returns the obtained key to $\mathcal{A}$.

4. When $\mathcal{A}$ outputs a guess bit, it outputs the same.

When the $\mathsf{1FE}$ challenger returns an encryption of $(\mathbf{x}_0, \mathbf{0}, \mathsf{mode} = 0)$, $\mathcal{A}$ sees the view of Hybrid 1, and when it returns an encryption of $(\perp, \mathsf{Sym.K}, \mathsf{mode} = 1)$, it sees the view of Hybrid 2. Note that in either case the decrypted value is the same. Thus, the advantage of $\mathcal{A}$ translates to the advantage of $\mathcal{B}$. □

Lemma E.4. Hybrids 2 and 3 are indistinguishable since $f(\mathbf{x}_0) \approx f(\mathbf{x}_1)$.

Proof. The only thing that differs in these 2 hybrids is the value of $y$. Given an adversary $\mathcal{A}$ who distinguishes between hybrids 2 and 3, we construct an adversary $\mathcal{B}$ who distinguishes between $f(\mathbf{x}_0)$ and $f(\mathbf{x}_1)$. $\mathcal{B}$ does the following:

1. It samples the public key honestly and gives it to $\mathcal{A}$.

2. When $\mathcal{A}$ outputs challenge distributions $D_0$ and $D_1$, it computes the ciphertext for $(\perp, \mathsf{Sym.K}, \mathsf{mode} = 1)$ honestly and returns this.

3. When $\mathcal{A}$ outputs a key request for function $f$, $\mathcal{B}$ outputs $(D_0, D_1, f)$ to the distribution challenger. $\mathcal{B}$ receives $y_0 = f(\mathbf{x}_0)$ or $y_1 = f(\mathbf{x}_1)$, where $\mathbf{x}_b \leftarrow D_b$ for $b \in \{0, 1\}$. It uses this to construct the circuit $f'$. It then computes the key for $f'$ honestly and returns this to $\mathcal{A}$.

4. When $\mathcal{A}$ outputs a guess, $\mathcal{B}$ outputs the same.

If $\mathcal{B}$ receives $y_0$, $\mathcal{A}$ sees the distribution of Hybrid 2, else it sees the distribution of Hybrid 3. The advantage of $\mathcal{A}$ therefore translates to an advantage of $\mathcal{B}$. □

Lemma E.5. Hybrids 3 and 4 are indistinguishable assuming the security of $\mathsf{1FE}$.

Proof. The only difference between Hybrids 3 and 4 is that in the former, the message encoded in the ciphertext is $(\perp, \mathsf{Sym.K}, \mathsf{mode} = 1)$ and in the latter the message encrypted is $(\mathbf{x}_1, \mathbf{0}, \mathsf{mode} = 0)$. Note that in both cases, we have the same output of decryption hence the two ciphertexts are indistinguishable by security of $\mathsf{1FE}$. □

Lemma E.6. Hybrids 4 and 5 are indistinguishable assuming the security of $\mathsf{Sym}$.

Proof. The proof is similar to Lemma E.2. □

□

# F Constructing Decomposable Functional Encryption for Circuits

Given any single-input circuit FE scheme $\mathsf{1FE}$ satisfying standard indistinguishability based security, a projective garbled circuit scheme $\mathsf{GC} = (\mathsf{GCirc}, \mathsf{GInp}, \mathsf{GEval})$ with indistinguishability based security [JSW17] supporting a circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ with $n$-bit inputs, a simple $\mathsf{PRF}$ $\mathsf{F} = (\mathsf{F.Setup}, \mathsf{F.Eval})$ and a symmetric encryption scheme $\mathsf{SYM}$, we can construct a single-input decomposable FE scheme $\mathsf{DFE}$ supporting the circuit class $\mathcal{C}$. We note that projective garbled

circuit schemes satisfying indistinguishability based security are implied from one-way functions [JSW17]. The intuition behind the construction is as follows.

Intuition: The public key and master secret key for DFE would be the same as that of 1FE. Given an $n$-bit message $\mathbf{x} = (x_1, \ldots, x_n)$, the DFE encryption algorithm samples a PRF key $\mathsf{K}$ and generates $n$ 1FE ciphertexts encoding $(\mathsf{K}, i, x_i)$. DFE key generation takes the master secret key and a circuit $C$ as input and generates a secret key for a circuit $\widehat{C}_{C,\mathsf{salt}}$. The circuit $\widehat{C}_{C,\mathsf{salt}}$ takes a 1FE message $(\mathsf{K}, i, x_i)$ as input and generates a garbled circuit $\widetilde{C}$ corresponding to $C$ and a garbled input label for the $i^{\text{th}}$ bit $x_i$ using randomness $\mathsf{PRF}(\mathsf{K}, \mathsf{salt})$. This relies on the projective property of GC [JSW17], i.e., each bit of the garbled input $\widetilde{\mathbf{x}}$ only depends on one bit of the actual input $\mathbf{x}$. For decryption, DFE runs the 1FE decryption on all the $n$ 1FE ciphertexts to obtain the garbled circuit $\widetilde{C}$ and the garbled input $\widetilde{\mathbf{x}}$ and then evaluates the garbled circuit to get the output $C(\mathbf{x})$.

For proving security, we additionally need to rely on a symmetric key scheme following standard techniques employing trapdoor modes from [CIJ+13, ABSV15]. The details follow as shown below.

$\mathsf{DFE.Setup}(1^\lambda, 1^n)$: On input the security parameter $\lambda$ and input message size $n$, do the following:

1. Generate $(\mathsf{1FE.PK}, \mathsf{1FE.MSK}) \leftarrow \mathsf{1FE.Setup}(1^\lambda, 1^{2\lambda+\log n+2})$.
2. Output $(\mathsf{PK}, \mathsf{MSK}) = (\mathsf{1FE.PK}, \mathsf{1FE.MSK})$.

$\mathsf{DFE.Enc}(\mathsf{PK}, \mathbf{x})$: On input the public key $\mathsf{PK}$ and a message $\mathbf{x} = (x_1, \ldots, x_n)$ of length $n = |\mathbf{x}|$, do the following:

1. Sample a PRF key $\mathsf{K} \leftarrow \mathsf{F.Setup}(1^\lambda)$ and set a flag $\mathsf{mode} = 0$.
2. Compute $\mathsf{CT}_{x_i} = \mathsf{1FE.Enc}(\mathsf{PK}, (\mathsf{K}, \mathbf{0}, i, x_i, \mathsf{mode})), \forall i \in [n]$ and output $\mathsf{CT}_{\mathbf{x}} = \{\mathsf{CT}_{x_i}\}_{i\in[n]}$.

$\mathsf{DFE.KeyGen}(\mathsf{MSK}, C)$: On input the master secret key $\mathsf{MSK}$ and a circuit $C \in \mathcal{C}_\lambda$, do the following:

1. Sample a random $\mathsf{salt} \leftarrow \{0,1\}^\lambda$, $\mathsf{CT}_i \leftarrow \{0,1\}^{\ell(\lambda)}, \forall i \in [0, n]$.
2. Output $\mathsf{SK}_{\widehat{C}} = \mathsf{1FE.KeyGen}(\mathsf{MSK}, \widehat{C}_{C,\mathsf{salt},\{\mathsf{CT}_i\}_{i\in[n]},\mathsf{CT}_0})$, where $\widehat{C}_{C,\mathsf{salt},\{\mathsf{SYM.CT}_i\}_{i\in[n]},\mathsf{SYM.CT}_{\widetilde{C}}}$ is a circuit described in Figure 20.

$\mathsf{DFE.Dec}(\mathsf{SK}_{\widehat{C}}, \mathsf{CT}_{\mathbf{x}})$: On input a function key $\mathsf{SK}_{\widehat{C}}$ and a decomposed ciphertext $\mathsf{CT}_{\mathbf{x}} = \{\mathsf{CT}_{x_i}\}_{i\in[n]}$, do the following:

1. For $i = 1$, invoke $\mathsf{1FE.Dec}(\mathsf{SK}_{\widehat{C}}, \mathsf{CT}_{x_1})$ to obtain a pair $(\ell_{1,x_1}, \widetilde{C})$.
2. For all $i \in [2, n]$, invoke $\mathsf{1FE.Dec}(\mathsf{SK}_{\widehat{C}}, \mathsf{CT}_{x_i})$ to obtain $(\ell_{i,x_i}, \perp)$.
3. Note that $\widetilde{\mathbf{x}} = \{\ell_{i,x_i}\}_{i\in[n]}$ represents the labels corresponding to the garbled input underlying $\mathsf{CT}_{\mathbf{x}}$ generated as outputs of $\widehat{C}$, while $\widetilde{C}$ represents the garbled circuit for $C$.
4. Run $\mathsf{GEval}(\widetilde{C}, \widetilde{\mathbf{x}})$ to get $\mathbf{y}$.

Correctness. We have by correctness of 1FE.Dec that it outputs the garbled input $\widetilde{\mathbf{x}}$ and the garbled circuit $\widetilde{C}$ correctly. The correctness of GEval implies that decryption recovers $C(\mathbf{x})$ as desired.

Functionality $\widehat{C}_{C,\mathsf{salt},\{\mathsf{SYM.CT}_i\}_{i\in[n]},\mathsf{SYM.CT}_{\widetilde{C}}}(\mathsf{K},\mathsf{SYM.K},i,x_i,\mathsf{mode})$

(a) Initialize the vector $\mathsf{out} = (c_1, c_2)$, where $c_j = \perp, \forall j \in [2]$.

(b) If $\mathsf{mode} = 1$, do the following:

i. Let $\mathsf{out}.c_1 = \mathsf{SYM.Dec}(\mathsf{SYM.K}, \mathsf{SYM.CT}_i)$.

ii. If $i = 1$, let $\mathsf{out}.c_2 = \mathsf{SYM.Dec}(\mathsf{SYM.K}, \mathsf{SYM.CT}_{\widetilde{C}})$.

(c) If $\mathsf{mode} = 0$, do the following:

i. Compute randomness $r = \mathsf{F.Eval}(\mathsf{K}, \mathsf{salt})$.

ii. Use randomness $r$ to generate the garbled circuit for $C$ as $(\widetilde{C}, \mathsf{sk}) = \mathsf{GCirc}(1^\lambda, C; r)$ as well as to generate the label corresponding to the $i^{\text{th}}$ input wire as $\ell_{i,x_i} = \mathsf{GInp}(\mathsf{sk}, (x_i, i); r)$.

iii. Let $\mathsf{out}.c_1 = \ell_{i,x_i}$. If $i = 1$, let $\mathsf{out}.c_2 = \widetilde{C}$.

(d) Output : $\mathsf{out}$.

Figure 20: Functionality $\widehat{C}_{C,\mathsf{salt},\{\mathsf{SYM.CT}_i\}_{i\in[n]},\mathsf{SYM.CT}_{\widetilde{C}}}$

Proof of Security.

Next, we argue that the DFE scheme constructed above is secure.

Theorem F.1. Assume that 1FE is an FE scheme satisfying standard indistinguishability based security, GC is a projective garbling scheme for circuits satisfying indistinguishability based security, Sym is a secure symmetric key encryption scheme and F is a secure PRF. Then, the DFE scheme constructed above is a single-input, decomposable FE scheme satisfying standard indistinguishability based security.

Proof. The proof proceeds via a sequence of hybrids where the first hybrid corresponds to an encryption of message $\mathbf{x}_0 \in \{0,1\}^n$ and the last hybrid corresponds to an encryption of message $\mathbf{x}_1 \in \{0,1\}^n$.

Hybrid 1: This is the real world with message $\mathbf{x}_0 = (x_1^0, \ldots, x_n^0) \in \{0,1\}^n$.

Hybrid 2: In this world, we hardwire $\widehat{C}$ with its output, namely the garbled circuit $(\widetilde{C}, \mathsf{sk})$ and input labels $\{\ell_{i,x_i^0}\}_{i\in[n]}$, using symmetric key encryption.

Hybrid 3: In this world, we change the message in each of the $n$ 1FE ciphertexts from $(\mathsf{K}, \mathbf{0}, i, x_i^0, 0)$ to $(\perp, \mathsf{Sym.K}, i, \perp, 1)$, i.e., the message encoded in $\mathsf{CT}_{x_i}$ is $(\perp, \mathsf{Sym.K}, i, \perp, \mathsf{mode} = 1), \forall i \in [n]$.

Hybrid 4: In this world, we use true randomness to generate the garbled circuit and garbled inputs instead of using randomness generated by PRF. Everything else remains the same as that of the previous hybrid. Note that the garbled input labels $\{\ell_{i,x_i^0}\}_{i\in[n]}$ encoded by $\{\mathsf{SYM.CT}_i\}_{i\in[n]}$ correspond to the input message bits of $\mathbf{x}_0 = (x_1^0, \ldots, x_n^0)$ from the previous hybrids.

Hybrid 5: In this world, we change the garbled input labels to $\{\ell_{i,x_i^1}\}_{i\in[n]}$ corresponding to the input message bits of $\mathbf{x}_1 = (x_1^1, \ldots, x_n^1)$ encoded by $\{\mathsf{SYM.CT}_i\}_{i\in[n]}$ and hardwired in the key for $\widehat{C}$.

Hybrid 6: In this world, we change the true randomness back to randomness generated by the $\mathsf{PRF}$ for computing the garbled circuit and garbled inputs. Everything else remains the same as that of the previous hybrid. Note that the garbled input labels $\{\ell_{i,x_i^1}\}_{i\in[n]}$ encoded by $\{\mathsf{SYM.CT}_i\}_{i\in[n]}$ now correspond to the input message bits of $\mathbf{x}_1 = (x_1^1, \ldots, x_n^1)$ from the previous hybrid.

Hybrid 7: In this world, we change the message in each of the $n$ $\mathsf{1FE}$ ciphertexts from $(\bot, \mathsf{Sym.K}, i, \bot, 1)$ to $(\mathsf{K}, \mathbf{0}, i, x_i^1, 0)$, i.e., the message encoded in $\mathsf{CT}_{x_i}$ now is $(\mathsf{K}, \mathbf{0}, i, x_i^1, 0), \forall i \in [n]$.

Hybrid 8: In this world, we change the hardwired values in $\widehat{C}$ corresponding to the $\{\mathsf{SYM.CT}_i\}_{i\in[n]}$ and $\mathsf{SYM.CT}_{\widetilde{C}}$ slots back to random strings from the ciphertext space of $\mathsf{SYM}$. Note that this corresponds to the real world with message $\mathbf{x}_1 = (x_1^1, \ldots, x_n^1) \in \{0,1\}^n$.

Next, we argue that consecutive hybrids are indistinguishable.

Lemma F.2. Hybrids 1 and 2 are indistinguishable assuming the security of $\mathsf{SYM}$.

Proof. The only thing that changes between Hybrid 1 and 2 are the choices of $\{\mathsf{SYM.CT}_i\}_{i\in[0,n]}$, so that in the former it is chosen randomly and in the latter case it is an honest encryption of the scheme $\mathsf{SYM}$. Given an adversary $\mathcal{A}$ which distinguishes between Hybrid 1 and Hybrid 2, we construct an adversary $\mathcal{B}$ which breaks the semantic security of $\mathsf{SYM}$. $\mathcal{B}$ does the following:

1. $\mathcal{B}$ generates $(\mathsf{PK}, \mathsf{MSK}) = (\mathsf{1FE.PK}, \mathsf{1FE.MSK}) \leftarrow \mathsf{1FE.Setup}(1^\lambda, 1^{2\lambda+\log n+2})$ honestly and returns $\mathsf{PK}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs a pair of challenge messages $(\mathbf{x}_0, \mathbf{x}_1)$, $\mathcal{B}$ samples a $\mathsf{PRF}$ key $\mathsf{K} \leftarrow \mathsf{F.Setup}(1^\lambda)$ and honestly computes $\mathsf{CT}_{x_i} = \mathsf{1FE.Enc}(\mathsf{PK}, (\mathsf{K}, \mathbf{0}, i, x_i^0, \mathsf{mode} = 0)), \forall i \in [n]$ and returns $\mathsf{CT}_{\mathbf{x}} = \{\mathsf{CT}_{x_i}\}_{i\in[n]}$ to $\mathcal{A}$.
3. When $\mathcal{A}$ requests a function key for $C$, $\mathcal{B}$ samples $\mathsf{salt} \leftarrow \{0,1\}^\lambda$ and computes $r = \mathsf{F.Eval}(\mathsf{K}, \mathsf{salt})$. It then generates the garbled circuit $(\widetilde{C}, \mathsf{sk}) = \mathsf{GCirc}(1^\lambda, C; r)$ and the input labels $\{\ell_{i,x_i^0} = \mathsf{GInp}(\mathsf{sk}, (i, x_i); r)\}_{i\in[n]}$ honestly. $\mathcal{B}$ then sends $\big(\{\ell_{i,x_i^0}\}_{i\in[n]}, \widetilde{C}\big)$ to the $\mathsf{SYM}$ challenger. The $\mathsf{SYM}$ challenger responds with $\big(\{\mathsf{SYM.CT}_i\}_{i\in[n]}, \mathsf{SYM.CT}_{\widetilde{C}}\big)$ upon which $\mathcal{B}$ constructs the circuit $\widehat{C}_{C,\mathsf{salt},\{\mathsf{SYM.CT}_i\}_{i\in[n]},\mathsf{SYM.CT}_{\widetilde{C}}}$ and further generates a secret key $\mathsf{SK}_{\widehat{C}} = \mathsf{1FE.KeyGen}\big(\mathsf{MSK}, \widehat{C}_{C,\mathsf{salt},\{\mathsf{SYM.CT}_i\}_{i\in[n]},\mathsf{SYM.CT}_{\widetilde{C}}}\big)$ honestly. $\mathcal{B}$ sends $\mathsf{SK}_{\widehat{C}}$ to $\mathcal{A}$.
4. When $\mathcal{A}$ outputs a guess bit, it outputs the same.

Now, $\mathcal{A}$ sees the view of Hybrid 1 if $\big(\{\mathsf{SYM.CT}_i\}_{i\in[n]}, \mathsf{SYM.CT}_{\widetilde{C}}\big)$ is random and $\mathcal{A}$ sees the view of Hybrid 2 if $\big(\{\mathsf{SYM.CT}_i\}_{i\in[n]}, \mathsf{SYM.CT}_{\widetilde{C}}\big)$ is an encryption of $\big(\{\ell_{i,x_i^0}\}_{i\in[n]}, \widetilde{C}\big)$. □

Lemma F.3. Hybrids 2 and 3 are indistinguishable assuming the security of $\mathsf{1FE}$.

Proof. The only difference between Hybrids 2 and 3 is that in the former the encrypted messages are $\{(\mathsf{K}, \mathbf{0}, i, x_i^0, \mathsf{mode} = 0)\}_{i\in[n]}$ and in the latter as $\{(\bot, \mathsf{SYM.K}, i, \bot, \mathsf{mode} = 1)\}_{i\in[n]}$. Assuming

there is an adversary $\mathcal{A}$ which distinguishes between Hybrid 2 and Hybrid 3, we construct an adversary $\mathcal{B}$ which breaks the security of $\mathsf{1FE}$.

$\mathcal{B}$ does the following:

1. It obtains the public key $\mathsf{PK}$ from the $\mathsf{1FE}$ challenger and returns this to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs a pair of challenge messages $(\mathbf{x}_0, \mathbf{x}_1)$, it samples $\mathsf{K} \leftarrow \mathsf{F.Setup}(1^\lambda)$, a symmetric encryption key $\mathsf{SYM.K}$ and then returns $n$ $\mathsf{1FE}$ challenge message pairs $\{(\mathsf{K}, \mathbf{0}, i, x_i^0, \mathsf{mode} = 0)\ , (\perp, \mathsf{SYM.K}, i, \perp, \mathsf{mode} = 1)\}_{i\in[n]}$ w.l.o.g. to the $\mathsf{1FE}$ challenger. It obtains $\mathsf{CT}_{\mathbf{x}} = \{\mathsf{CT}_{x_i}\}_{i\in[n]}$ and returns this to $\mathcal{A}$.
3. When $\mathcal{A}$ outputs a function query for $C$, $\mathcal{B}$ constructs the function $\widehat{C}_{C,\mathsf{salt},\{\mathsf{SYM.CT}_i\}_{i\in[n]},\mathsf{SYM.CT}_{\widetilde{C}}}$ as described in Figure 20 and sends this to the $\mathsf{1FE}$ challenger. Here, $\mathsf{SYM.CT}_{\widetilde{C}}$ is computed as $\mathsf{SYM.Enc}(\mathsf{SYM.K}, \widetilde{C})$ where $(\widetilde{C}, \mathsf{sk}) = \mathsf{GCirc}(1^\lambda, C; \mathsf{F.Eval}(\mathsf{K}, \mathsf{salt}))$ while $\mathsf{SYM.CT}_i$ are computed as $\mathsf{SYM.Enc}(\mathsf{SYM.K}, \ell_{i,x_i^0})$ where $\ell_{i,x_i^0} = \mathsf{GInp}(\mathsf{sk}, (i, x_i^0); \mathsf{F.Eval}(\mathsf{K}, \mathsf{salt}))$, $\forall i \in [n]$. It returns the obtained key $\mathsf{SK}_{\widehat{C}}$ to $\mathcal{A}$.
4. When $\mathcal{A}$ outputs a guess bit, it outputs the same.

When the $\mathsf{1FE}$ challenger returns encryptions of $\{(\mathsf{K}, \mathbf{0}, i, x_i^0, \mathsf{mode} = 0)\}_{i\in[n]}$, $\mathcal{A}$ sees the view of Hybrid 1, and when it returns an encryption of $\{(\perp, \mathsf{SYM.K}, i, \perp, \mathsf{mode} = 1)\}_{i\in[n]}$, it sees the view of Hybrid 2. Note that in either case the decrypted value is the same and thus the reduction $\mathcal{B}$ is a valid $\mathsf{1FE}$ adversary. Thus, the advantage of $\mathcal{A}$ translates to the advantage of $\mathcal{B}$. □

Lemma F.4. Hybrids 3 and 4 are indistinguishable assuming the security of $\mathsf{PRF}$ $\mathsf{F}$.

Proof. The only difference in Hybrid 4 from Hybrid 3 is that instead of randomness generated by the $\mathsf{PRF}$, true randomness is used now to generate the garbled circuit and garbled input. Note that the $\mathsf{PRF}$ key is not explicitly needed in the Hybrid 3. Thus, assuming there is an adversary $\mathcal{A}$ which distinguishes between Hybrid 3 and Hybrid 4, we construct an adversary $\mathcal{B}$ which breaks the security of $\mathsf{PRF}$ $\mathsf{F}$. $\mathcal{B}$ does the following:

1. $\mathcal{B}$ generates $(\mathsf{PK}, \mathsf{MSK}) = (\mathsf{1FE.PK}, \mathsf{1FE.MSK}) \leftarrow \mathsf{1FE.Setup}(1^\lambda, 1^{2\lambda+\log n+2})$ honestly and returns $\mathsf{PK}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs a pair of challenge messages $(\mathbf{x}_0, \mathbf{x}_1)$, $\mathcal{B}$ samples $\mathsf{SYM.K}$ and simulates the challenge message as $\mathsf{CT}_{\mathbf{x}} = \{\mathsf{CT}_{x_i}\}_{i\in[n]}$ where $\mathsf{CT}_{x_i} = \mathsf{1FE.Enc}(\mathsf{PK}, (\perp, \mathsf{SYM.K}, i, \perp, 1))\}_{i\in[n]}$.
3. When $\mathcal{A}$ outputs a function query for $C$, $\mathcal{B}$ first queries the $\mathsf{PRF}$ challenger upon which it receives $r$. It then uses $r$ to compute the garbled circuit $(\widetilde{C}, \mathsf{sk}) = \mathsf{GCirc}(1^\lambda, C; r)$ as well as the garbled input labels $\ell_{i,x_i^0} = \mathsf{GInp}(\mathsf{sk}, (i, x_i^0); r) \forall i \in [n]$, honestly. $\mathcal{B}$ then samples $\mathsf{salt} \leftarrow \{0,1\}^\lambda$ and computes $\{\mathsf{SYM.CT}_i = \mathsf{SYM.Enc}(\mathsf{SYM.K}, \ell_{i,x_i^0})\}_{i\in[n]}$ and $\mathsf{SYM.CT}_{\widetilde{C}} = \mathsf{SYM.Enc}(\mathsf{SYM.K}, \widetilde{\mathsf{C}})$. It then computes $\mathsf{SK}_{\widehat{C}} = \mathsf{1FE.KeyGen}(\mathsf{MSK}, \widehat{C}_{C,\mathsf{salt},\{\mathsf{SYM.CT}_i\}_{i\in[n]},\mathsf{SYM.CT}_{\widetilde{C}}})$ for the function $\widehat{C}_{C,\mathsf{salt},\{\mathsf{SYM.CT}_i\}_{i\in[n]},\mathsf{SYM.CT}_{\widetilde{C}}}$ as described in Figure 20 and returns $\mathsf{SK}_{\widehat{C}}$ to $\mathcal{A}$.
4. When $\mathcal{A}$ outputs a guess bit, $\mathcal{B}$ outputs the same.

If $\mathcal{B}$ had received $r = \mathsf{F.Eval}(\mathsf{K}, \mathsf{salt})$ from the the $\mathsf{PRF}$ challenger, $\mathcal{A}$ sees the distribution of Hybrid 3, else it sees the distribution of Hybrid 4 if $r$ was sampled uniformly at random by the $\mathsf{PRF}$ challenger. The advantage of $\mathcal{A}$ therefore translates to an advantage of $\mathcal{B}$. □

Lemma F.5. Hybrids 4 and 5 are indistinguishable assuming the security of $\mathsf{GC}$.

Proof. The only difference between Hybrids 4 and 5 is that in the former, the messages encoded in $\{\mathsf{SYM.CT}_i\}_{i\in[n]}$ ciphertexts hardwired in $\widehat{C}$ were $\{\ell_{i,x_i^0}\}_{i\in[n]}$ while in the later, the encoded messages are $\{\ell_{i,x_i^1}\}_{i\in[n]}$. Note that in both cases, we have the same output of decryption since $C(\mathbf{x}_0) = C(\mathbf{x}_1)$ and hence the two hybrids are indistinguishable by indistinguishability based security of $\mathsf{GC}$. More formally, we show that if there is an adversary $\mathcal{A}$ which distinguishes between Hybrid 4 and Hybrid 5, we construct an adversary $\mathcal{B}$ which breaks the security of $\mathsf{GC}$. $\mathcal{B}$ does the following:

1. $\mathcal{B}$ generates $(\mathsf{PK}, \mathsf{MSK}) = (\mathsf{1FE.PK}, \mathsf{1FE.MSK}) \leftarrow \mathsf{1FE.Setup}(1^\lambda, 1^{2\lambda+\log n+2})$ honestly and returns $\mathsf{PK}$ to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs a pair of challenge messages $(\mathbf{x}_0, \mathbf{x}_1)$, $\mathcal{B}$ samples $\mathsf{SYM.K}$ and simulates the challenge message as $\mathsf{CT}_{\mathbf{x}} = \{\mathsf{CT}_{x_i}\}_{i\in[n]}$ where $\mathsf{CT}_{x_i} = \mathsf{1FE.Enc}(\mathsf{PK}, (\bot, \mathsf{SYM.K}, i, \bot, 1))\}_{i\in[n]}$.
3. When $\mathcal{A}$ outputs a function query for $C$, $\mathcal{B}$ first constructs and sends the challenge message pair $((C, \mathbf{x}_0), (C, \mathbf{x}_1))$ to the $\mathsf{GC}$ challenger. On receiving $(\widetilde{C}, \widetilde{\mathbf{x}} = \{\ell_{i,x_i}\}_{i\in[n]})$ from the $\mathsf{GC}$ challenger, $\mathcal{B}$ computes $\{\mathsf{SYM.CT}_i = \mathsf{SYM.Enc}(\mathsf{SYM.K}, \ell_{i,x_i})\}_{i\in[n]}$ and $\mathsf{SYM.CT}_{\widetilde{C}} = \mathsf{SYM.Enc}(\mathsf{SYM.K}, \widetilde{\mathsf{C}})$. It then samples $\mathsf{salt} \leftarrow \{0,1\}^\lambda$ and generates a function key for the function $\widehat{C}_{C,\mathsf{salt},\{\mathsf{SYM.CT}_i\}_{i\in[n]},\mathsf{SYM.CT}_{\widetilde{C}}}$ as $\mathsf{SK}_{\widehat{C}} = \mathsf{1FE.KeyGen}(\mathsf{MSK}, \widehat{C}_{C,\mathsf{salt},\{\mathsf{SYM.CT}_i\}_{i\in[n]},\mathsf{SYM.CT}_{\widetilde{C}}})$. $\mathcal{B}$ returns $\mathsf{SK}_{\widehat{C}}$ to $\mathcal{A}$.
4. When $\mathcal{A}$ outputs a guess bit, $\mathcal{B}$ outputs the same.

Note that since $\mathcal{A}$ is a valid $\mathsf{DFE}$ adversary satisfying $C(\mathbf{x}_0) = C(\mathbf{x}_1)$, this implies $\mathcal{B}$ is a valid $\mathsf{GC}$ adversary. Further, if the $\mathsf{GC}$ challenger had returned $(\widetilde{C}, \widetilde{\mathbf{x}} = \{\ell_{i,x_i^0}\}_{i\in[n]})$, then $\mathcal{A}$ sees the view of Hybrid 4 and if the $\mathsf{GC}$ challenger had returned $(\widetilde{C}, \widetilde{\mathbf{x}} = \{\ell_{i,x_i^1}\}_{i\in[n]})$, then $\mathcal{A}$ sees the view of Hybrid 5. The advantage of $\mathcal{A}$ therefore translates to an advantage of $\mathcal{B}$. □

Lemma F.6. Hybrids 5 and 6 are indistinguishable assuming the security of $\mathsf{PRF}$ $\mathsf{F}$.

Proof. The proof is similar to Lemma F.4. □

Lemma F.7. Hybrids 6 and 7 are indistinguishable assuming the security of $\mathsf{1FE}$.

Proof. The proof is similar to Lemma F.3. □

Lemma F.8. Hybrids 7 and 8 are indistinguishable assuming the security of $\mathsf{SYM}$.

Proof. The proof is similar to Lemma F.2. □

□

## F.1 Decomposable Functional Encryption for Circuits: Instantiations

We note that most functional encryption schemes in the literature are already decomposable, since a long input $\mathbf{x}$ is typically encoded bit by bit, using a separate public key component. Indeed, we do not know of any exceptions in the literature. For instance, recall the ciphertext of [ABSV15]:

$$\begin{aligned} \mathsf{CT}_0 &\leftarrow \mathsf{OneCT.Enc}(\mathsf{OneCT.SK}, \mathbf{x}) \text{ and} \\ \mathsf{CT}_1 &\leftarrow \mathsf{Sel.Enc}(\mathsf{Sel.MPK}, (\mathsf{OneCT.SK}, \mathsf{K}, 0^\lambda, 0)). \end{aligned}$$

Above, $\mathsf{CT}_1$ is a ciphertext component which is independent of the message (and depends only on randomness), hence it may be denoted as $\mathsf{CT}_{\mathsf{indpt}}$ in the notation above. Therefore, it remains to show that $\mathsf{CT}_0$ is decomposable. This depends on the particular $\mathsf{OneCT}$ scheme that is chosen, but for instance, it was shown in [AS17a] that the $\mathsf{OneCT}$ succinct FE scheme from LWE constructed by [GKP+13b] is decomposable. We refer the reader to [AS17a] for details.

We note that the recent constructions of FE from constant degree multilinear maps [Lin17, LT17] also satisfy decomposability, despite the fact that they precompute high degree monomials which are encoded. To see this, note that the encrypt algorithm in [Lin17, LT17] takes as input a message $\mathbf{x}$ and chooses a PRG seed $\mathbf{s}$ (say) represented as a matrix. The encryptor computes a long message $\mathbf{y}$ (say) that consists of monomials computed over $\mathbf{x}, \mathbf{s}$. While the computation of arbitrary monomials would violate decomposability, in the above constructions, the monomials are linear in bits of $\mathbf{x}$, and the high degree terms are all computed over the bits of the seed $\mathbf{s}$. Our construction requires that the bits corresponding to the symbol and state of a TM be encoded separately, and these would form the input $\mathbf{x}$ in the constructions of [Lin17, LT17]. Intuitively, the PRG seed is used to derive randomness meant for computing a randomized encoding and is chosen independently of the input message $\mathbf{x}$. Hence, the constructions of [Lin17, LT17] also satisfy decomposability required by our compilers.

Next, we sketch how the construction of [GGH+13] can be seen to satisfy decomposability, with minor modifications. The ciphertext for a single bit message $m$ in this scheme is $(e_1, e_2, \pi)$, where $e_1 = \mathsf{Enc}(\mathsf{PK}_1, m)$ and $e_2 = \mathsf{Enc}(\mathsf{PK}_2, m)$ and $\pi$ is a NIZK proof that $e_1$ and $e_2$ both encrypt the same bit. Note that here the two ciphertexts $e_1$ and $e_2$ are using distinct public key encryption schemes (i.e. these are not ciphertext components in decomposable FE). To argue decomposability, consider message $\mathbf{m} = (m_1, \ldots, m_n)$ as a vector of $n$ bits rather than a single bit. Then, we may compute the encryptions bit by bit, and also test equality bit by bit in the NIZK, tying together all bits of $m$ by common randomness, satisfying the given definition of decomposability.

In more detail, we may compute $e_1 = (e_{1,1}, \ldots, e_{1,n})$ and $e_2 = (e_{2,1}, \ldots, e_{2,n})$ as well as $e^* = \mathsf{Enc}(\mathsf{PK}_3, \mathsf{R})$ where:

- $\forall i \in [n]$, $e_{1,i}$ and $e_{2,i}$ encode message $(m_i, \mathsf{R})$ where $\mathsf{R}$ is shared across all $i$. Note that $\mathsf{R}$ here is part of the encoded message (the encryption randomness used to construct the ciphertexts $e_{1,i}$ and $e_{2,i}$ is different and not denoted here).
- Denote by $\pi_i$ the NIZK proof that $e_{1,i}$ and $e_{2,i}$ encode the same bit $m_i$ and that $e_{1,i}, e_{2,i}$ encode the same $\mathsf{R}$ as $e^*$.

Then, the $n$ ciphertext components of the decomposable FE are $(e_{1,1}, e_{2,1}, \pi_1), \ldots, (e_{1,n}, e_{2,n}, \pi_n)$ and the independent ciphertext component is $e^*$ ($\mathsf{CT}_{\mathsf{indpt}}$ from Definition 2.1.1). Note that if an attacker tried to replace any one piece in this set, the $\mathsf{R}$ would not match (except with negligible probability) and the NIZK proof would not validate.

The proof of security is similar to [GGH+13].