Bootstrapping Homomorphic Encryption via Functional Encryption*

Nir Bitansky† Tomer Solomon‡

September 14, 2023

Abstract

Homomorphic encryption is a central object in modern cryptography, with far-reaching applications. Constructions supporting homomorphic evaluation of arbitrary Boolean circuits have been known for over a decade, based on standard lattice assumptions. However, these constructions are *leveled*, meaning that they only support circuits up to some a-priori bounded depth. These leveled constructions can be *bootstrapped* into *fully* homomorphic ones, but this requires additional *circular security assumptions*, which are construction-dependent, and where reductions to standard lattice assumptions are no longer known. Alternative constructions are known based on indistinguishability obfuscation, which has been recently constructed under standard assumptions. However, this alternative requires subexponential hardness of the underlying primitives.

We prove a new bootstrapping theorem relying on *functional encryption*, which is known based on standard *polynomial* hardness assumptions. As a result we obtain the first fully homomorphic encryption scheme that avoids both circular security assumptions and super-polynomial hardness assumptions. The construction is secure against uniform adversaries, and can be made non-uniformly secure, under a widely-believed worst-case complexity assumption (essentially that non-uniformity does not allow arbitrary polynomial speedup).

At the heart of the construction is a new proof technique based on cryptographic puzzles. Unlike most cryptographic reductions, our security reduction does not fully treat the adversary as a black box, but rather makes explicit use of its running time (or circuit size).

1 Introduction

Starting from the breakthrough work of Gentry [Gen09], homomorphic encryption has changed the face of cryptography, bringing about the era of encrypted computation, with an abundance of new applications and new cryptographic tools. In the context of homomorphic encryption for general Boolean circuits, an important distinction is between schemes that are *fully homomorphic* and schemes that are only *leveled homomorphic*. Leveled homomorphic schemes have an a-priori bound d on the depth of circuits for which homomorphic evaluation is supported. In particular, the size of parameters in these schemes may scale with d. In contrast, fully homomorphic schemes allow to evaluate circuits of arbitrary polynomial depth with no a-priori bound.

By now there are various, relatively simple, constructions of leveled homomorphic encryption schemes based on standard lattice assumptions (c.f. [BV11, BGV12, GSW13]). However, constructing fully homomorphic encryption from similar assumptions remains a long standing open problem. Nonetheless, Gentry [Gen09] showed that by making an additional *circular security assumption* the above schemes (and in fact any leveled scheme that can evaluate its own decryption circuit) can be made fully homomorphic. However, in this case we no longer have a reduction to similar lattice assumptions. Furthermore, the corresponding

*This work was supported in part by the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (grant agreement No. 101042417, acronym SPP) and by ISF grants 484/18, 2137/19.

†Tel Aviv University, email nirbitan@tau.ac.il. Member of the Check Point Institute of Information Security.

‡Tel Aviv University, email tomer.solomon1@gmail.com.

assumption is construction-dependent, in the sense that it assumes circular security of any specific encryption scheme considered, rather than based on a clean problem that can be studied in isolation.

An alternative construction of fully homomorphic encryption was shown by Canetti et al. [CLTV15] based on indistinguishability obfuscation. Combining this with the recent breakthrough of Jain, Lin, and Sahai [JLS21, JLS22] who constructed indistinguishability obfuscation from standard assumptions, leads to fully homomorphic encryption from standard assumptions. The resulting construction, however, suffers from a caveat — it requires that the underlying assumptions are all sub-exponentially secure.

The work of Agrikola et al. [ACH20] showed that fully homomorphic encryption can be obtained from polynomially-secure indistinguishability obfuscation and *extremely lossy functions* (ELFs) [Zha16]. However, the above mentioned constructions of indistinguishability obfuscation are based on subexponential hardness assumptions, whereas ELFs are only known based on exponential hardness assumptions. Other constructions of fully homomorphic encryption are based on non-standard forms of obfuscation or functional encryption, which are not known to be reducible to standard assumptions [ABF$^+$13].

Overall, a construction of fully homomorphic encryption based on standard polynomial assumptions has yet to be achieved.

1.1 Our Result

In this work we prove a new bootstrapping theorem based on *functional encryption.*

Theorem 1.1 (Informal)**.** *Assuming (polynomially-secure) semi-compact public-key functional encryption, any leveled homomorphic encryption scheme that can evaluate (slightly more than) its own decryption circuit, can be turned into a fully-homomorphic scheme. The scheme is secure against uniform efficient adversaries. Assuming also that for any $c \in \mathbb{N}$, $\boldsymbol{P} \not\subseteq \boldsymbol{ioSIZE}(n^c)$, it is secure against efficient non-uniform adversaries.*

The required functional encryption schemes were recently constructed by Jain, Lin, and Sahai from standard polynomial assumptions (LPN, Bilinear SXDH, and shallow PRGs) [JLS21, JLS22] and thus combining with known leveled homomorphic encryption schemes (e.g. [BV11] from LWE [Reg05]), we obtain the first fully homomorphic encryption scheme that does not rely on circular security nor does it require super-polynomial hardness assumptions.

The assumption that for all $c \in \mathbb{N}$, $\mathbf{P} \not\subset \mathbf{ioSIZE}(n^c)$, required in the non-uniform setting, essentially means that circuits of a fixed polynomial size n^c, cannot decide all of $\mathbf{P}$ in the worst case. (Formally, there is a language $\mathcal{L}_c$ in $\mathbf{P}$, which they fail to decide in the worst-case for all large enough instances.) This can be viewed as a natural generalization of *the time hierarchy theorem,* which holds unconditionally. In particular, this assumption follows from widely believed worst-case assumptions such as *the non-uniform exponential time hypothesis.*

Polynomial Security through Cryptographic Puzzles. Our main conceptual contribution is a new proof technique that leverages cryptographic puzzles for the sake of polynomial security. An interesting aspect of the technique is that unlike most cryptographic security reductions, our security reduction does not fully treat the adversary as a black box, but rather makes explicit use of its running time (or circuit size).

1.2 Technical Overview

In this section we provide an overview of the main new technical ideas behind our results.

We first briefly recall the syntax and requirements from leveled and fully homomorphic encryption schemes. A homomorphic encryption scheme consists of an encryption algorithm Enc, a decryption algorithm Dec, and a homomorphic evaluation algorithm Eval. These algorithms have associated public encryption and evaluation keys $\mathsf{ek}, \mathsf{evk}$ and a secret decryption key dk, which are sampled jointly from some generation algorithm Gen. In a leveled scheme, this generation algorithm depends on the maximal depth d of supported circuits, and the size of keys $\mathsf{ek}, \mathsf{evk}, \mathsf{dk}$ may scale polynomially with d. In contrast, in fully homomorphic

schemes the keys are of fixed size, and yet can support the homomorphic evaluation of circuits of arbitrary polynomial depth.

Gentry's Bootstrapping. Our starting point is Gentry's bootstrapping technique [Gen09]. Here the basic idea is that a leveled scheme that supports slightly more than the depth d^* of its own decryption circuit can be transformed into a leveled scheme for computations of arbitrary depth. Specifically, to support circuits of depth $d \gg d^*$, we generate $d+1$ sets of keys $(\mathsf{ek}_0, \mathsf{evk}_0, \mathsf{dk}_0), \ldots, (\mathsf{ek}_d, \mathsf{evk}_d, \mathsf{dk}_d)$, and produce a new set of keys:

$$\begin{aligned} \mathsf{ek} &= \mathsf{ek}_0 \\ \mathsf{evk} &= \begin{pmatrix} \mathsf{Enc}_{\mathsf{ek}_1}(\mathsf{dk}_0) & \mathsf{Enc}_{\mathsf{ek}_2}(\mathsf{dk}_1) & \ldots & \mathsf{Enc}_{\mathsf{ek}_d}(\mathsf{dk}_{d-1}) \\ \mathsf{evk}_1 & \mathsf{evk}_2 & \ldots & \mathsf{evk}_d \end{pmatrix} \\ \mathsf{dk} &= \mathsf{dk}_d \end{aligned}$$

Then, starting from an encryption of the message $\mathsf{Enc}_{\mathsf{ek}_0}(m)$, a circuit of arbitrary depth d can be homomorphically evaluated layer by layer according to the following invariant: given encryptions $\mathsf{Enc}_{\mathsf{ek}_i}(v_i)$ of the circuit wire values v_i in layer i, we can obtain encryptions $\mathsf{Enc}_{\mathsf{ek}_{i+1}}(v_{i+1})$ of the wire values v_{i+1} in the next layer. This is done starting from $\mathsf{Enc}_{\mathsf{ek}_{i+1}}(\mathsf{dk}_i)$, which is part of the evaluation key evk, and homomorphically evaluating the circuit that given as input the decryption key dk_i, decrypts $\mathsf{Enc}_{\mathsf{ek}_i}(v_i)$, and computes v_{i+1} from v_i. The security of the scheme reduces to that of the underlying encryption by a standard hybrid argument, where starting from $\mathsf{Enc}_{\mathsf{ek}_d}(\mathsf{dk}_{d-1})$, we switch each encryption $\mathsf{Enc}_{\mathsf{ek}_i}(\mathsf{dk}_{i-1})$ with an encryption $\mathsf{Enc}_{\mathsf{ek}_i}(0)$ of garbage (say, the all-zero string), which is possible as we have already eliminated dk_i from the adversary's view.

The above is still a leveled scheme as the size of keys grows with the maximal depth d that can be evaluated. To make the scheme fully homomorphic, Gentry samples a single set of keys $(\mathsf{ek}, \mathsf{evk}, \mathsf{dk})$ and collapses the chain of encryptions $\mathsf{Enc}_{\mathsf{ek}_1}(\mathsf{dk}_0) \ldots \mathsf{Enc}_{\mathsf{ek}_d}(\mathsf{dk}_{d-1})$ into one encryption $\mathsf{Enc}_{\mathsf{ek}}(\mathsf{dk})$ where the decryption key is encrypted under its corresponding encryption key. While the size of keys is now fixed regardless of the depth of evaluated circuit, we can no longer invoke the above reduction to the security of the underlying encryption scheme. Proving security requires that the underlying scheme is circularly secure, namely that an encryption of the scheme's own decryption key does not compromise its security.

Compressing the Keys via Obfuscation. Aiming to avoid the circular security assumption, Canetti et al. [CLTV15] consider an alternative approach toward compressing Gentry's bootstrapping. (To be more precise, the exact details in [CLTV15] are somewhat different, but the core is essentially similar). The basic observation is that, using a pseudo-random function (PRF) family $\{f_k\}_k$, the process of generating the long evaluation key evk can be described by a succinct circuit EVK that has a hardwired PRF key k. Specifically, for $i \geq 1$, the randomness r_i for generating the i-th set of keys, and encryption randomness r'_i for the i-th encryption, are both derived using PRF $f_k(i)$. The succinct circuit EVK given input i, derives the key generation randomness (r_{i-1}, r_i) and encryption randomness r'_i, computes the corresponding keys, and outputs $(\mathsf{evk}_i, \mathsf{Enc}_{\mathsf{ek}_i}(\mathsf{dk}_{i-1}; r'_i))$. We then replace the previously large evaluation key evk with an obfuscated version $\widetilde{\mathsf{EVK}}$ of the succinct circuit EVK. To allow for homomorphic evaluation of circuits of arbitrary polynomial depth, the space of input indices $\{1, \ldots, \overline{d}\}$ is taken to be of super-polynomial size $\lambda^{\omega(1)}$ in the security parameter λ.

Using similar techniques to those in [BGL+15, CLTV15] this construction can be proven secure if the obfuscation is instantiated using indistinguishability obfuscation (IO) and the PRF is *puncturable* [BW13, KPTZ13, BGI14] (the concept of puncturing is discussed in more detail later on). However, the proof requires that the PRF, IO, and underlying leveled homomorphic encryption are all super-polynomially secure. Roughly speaking, the cause for this loss is that the security reduction extends the previously described hybrid argument, and in particular suffers a loss in the indistinguishability gap through $d = \lambda^{\omega(1)}$ hybrids.

Overall, this construction has two sources of super-polynomial loss:

1. The super-polynomial number of hybrids described above.

2. Known IO reductions incur a loss proportional to the size of the input space of obfuscated circuits, which in this case is again super-polynomial (and in fact conjectured to be inherent [GLSW15]).

Our solution essentially addresses these two sources of super-polynomial loss separately. We next explain the main ideas involved.

Polynomial Number of Hybrids through Cryptographic Puzzles. The essential problem is that the two requirements of full homomorphism on one hand, and a polynomial reduction on the other hand, collide. To support computations of arbitrary polynomial depth, we cannot restrict the length d of the key chain to any specific polynomial. At the same time if the length d of the key chain is super-polynomial, then the number of hybrids and loss in the security proof is also super-polynomial. Roughly speaking, to avoid the super-polynomial loss, we aim that an adversary of polynomial running time (or circuit size) $t = \mathrm{poly}(\lambda)$ will only be able to access a corresponding prefix of the key chain

$$\begin{array}{cccccccc} \mathsf{Enc}_{\mathsf{ek}_1}(\mathsf{dk}_0) & \mathsf{Enc}_{\mathsf{ek}_2}(\mathsf{dk}_1) & \dots & \mathsf{Enc}_{\mathsf{ek}_t}(\mathsf{dk}_{t-1}) & \mathsf{Enc}_{\mathsf{ek}_{t+1}}(\mathsf{dk}_t) & \dots & \mathsf{Enc}_{\mathsf{ek}_d}(\mathsf{dk}_{d-1}) \\ \mathsf{evk}_1 & \mathsf{evk}_2 & \dots & \mathsf{evk}_t & \mathsf{evk}_{t+1} & \dots & \mathsf{evk}_d \end{array} .$$

Had we managed that, then the number of hybrids will become proportional to the adversary's polynomial running time, as required.

To achieve this guarantee, we use an appropriate notion of *cryptographic puzzles*. Generally speaking, cryptographic puzzles are encryption schemes where decryption does not require a decryption key, but rather the investment of some pre-specified amount of computational resources. Relevant to our context are *relaxed time-lock puzzles* (RTLPs) as defined in [BGJ+16], where this resource is simply *time*. Specifically, an RTLP allows anyone to encrypt a message m relative to a difficulty parameter T so that the resulting ciphertext (aka puzzle) Z can be decrypted in time $\approx T$, whereas adversaries running in time $\ll T$ do not learn anything about the underlying message. Encryption on the other hand, only requires time $\mathrm{polylog}(T)$. (Both encryption and decryption also have some fixed polynomial dependence on the security parameter λ, which we suppress here for simplicity.)

Such RTLPs relax the notion of (non-relaxed) TLPs by Rivest, Shamir, and Wagner [RSW96], which requires that decryption is hard even for parallel adversaries that have sequential time $\ll T$, although possibly a number of processors larger than T. Following [BGJ+16], RTLPs against *uniform* adversaries can be constructed based on one-way functions and succinct randomized encodings [BGL+15, CHJV15, KLW15], which in turn can be constructed from indistinguishability obfuscation *with logarithmic input space* [AL18, GS18, KNTY19] (and accordingly also from standard polynomial assumptions [JLS21, JLS22]). We obtain RTLPs against non-uniform adversaries, under the worst-case complexity assumption stated in Theorem 1.1 (in addition to succinct randomized encodings). The security guarantee achieved by the construction that for any polynomial $t = \mathrm{poly}(\lambda)$, there exists a polynomial $T = t^{O(1)}$ such that adversaries running in time t cannot break the semantic security of puzzles of difficulty T. The construction itself is similar in spirit to the construction of TLPs from succinct randomized encodings and non-parallelizing languages in [BGJ+16] and can be found in Section 2.1.

Augmenting the Construction with RTLPs. Equipped with RTLPs we augment the previous construction as follows. Rather than considering the circuit EVK that explicitly outputs the key chain, we augment the circuit EVK so that it outputs increasingly difficult puzzles encrypting the key chain. That is, on input i, EVK will not output $\mathsf{Enc}_{\mathsf{ek}_i}(\mathsf{dk}_{i-1})$ in the clear, but rather output a RTLP Z_i encrypting $\mathsf{Enc}_{\mathsf{ek}_i}(\mathsf{dk}_{i-1})$. The required randomness for computing the puzzle Z_i, will again be derived by applying another PRF $f_{k'}$ to i. In terms of functionality, we can still homomorphically evaluate circuits of arbitrary polynomial depth d, by simply solving the puzzles. This only incurs an additive overhead of $\approx d^2$ (we explain later how this overhead can be reduced). At the same time, we are now able to obtain a polynomial reduction, as we effectively prevent the adversary from accessing the key chain, except for a polynomially bounded prefix.

In a bit more detail, given an adversary of polynomial running time $t = \mathrm{poly}(\lambda)$. The hybrid strategy will start from index T, such that puzzles of difficulty cannot be solved by t-size adversaries. Intuitively, we can now replace the puzzle Z_T encrypting the key chain link $\mathsf{Enc}_{\mathsf{ek}_T}(\mathsf{dk}_{T-1})$, with a puzzle that encrypts garbage. Now, having eliminated the secret key dk_{T-1} from the adversary's view, we can proceed with the

hybrid argument to the next link and so on. Overall, we will have a polynomial number $T = t^{O(1)}$ of hybrids. The exact details behind this hybrid argument are overall similar to [CLTV15], they are based on IO, used to obfuscate the augmented circuit EVK, and puncturable PRFs.

Unlike typical cryptographic reductions that treat the adversary as a complete black-box, regardless of its efficiency, our reduction depends on the specific running time of the adversary, and has a diagonalization flavour.

Avoiding the Super-polynomial Loss of General Purpose IO. Having removed the first mentioned source of super-polynomial loss (the super-polynomial number of hybrids), we now turn to deal with the second source, which is the inherent loss in using general-purpose IO. Toward this, we aim to leverage the fact that for circuits with a polynomial-size input domain, existing IO constructions do admit a polynomial reduction to standard (polynomial) assumptions. However, as explained above, we cannot directly rely on this fact since the circuit EVK (generating the key chain) has a super-polynomial size input space $[d] = \{1, \ldots, d = \lambda^{\omega(1)}\}$, so that it will be able to support homomorphic evaluation of circuits of any polynomial depth. Nevertheless, we show a simple trick to overcome this difficulty.

The basic idea is that rather than having a single key generation circuit EVK over the domain $[d]$, we split it to ℓ circuits $\mathsf{EVK}_1, \ldots, \mathsf{EVK}_\ell$ where $d = 2^\ell$ and EVK_i is defined over $(2^{i-1}, 2^i]$. Each of these circuits EVK_i has its own PRF key k_i used to generate the chain links in the corresponding interval as well as the last decryption key $\mathsf{dk}_{2^{i-1}}$ of the previous interval, allowing EVK_i to "continue the chain" from where EVK_{i-1} left off. Now we can resort to the same hybrid strategy as before only that we only ever invoke IO security for the obfuscated circuits $\widetilde{\mathsf{EVK}}_1, \ldots, \widetilde{\mathsf{EVK}}_{\log T}$, which are all defined over domains of size at most $T = t^{O(1)}$, determined by the adversary's polynomial size t.[1]

On the Overhead of Homomorphic Evaluation. As mentioned above, homomorphically evaluating a circuit of depth d in our scheme incurs an additive overhead of $\approx d^2$. Indeed, as the complexity of puzzles grows with time, the overall work needed is proportional to $\sum_{i \in [d]} i$. This overhead can be reduced to $\approx d$ by embedding cryptographic puzzles only in a sparse set of positions rather than in every position as described before. For instance, we can embed a puzzle in every position that is a power of two. This way the overall work is $\sum_{i \in \log d} 2^i = O(d)$. This strategy still enables a polynomial security reduction.

While the amortized overhead in the above adaptation is linear, most of the computational overhead is incurred toward the end of the homomorphic evaluation. However, by another simple tweak to the scheme, we can ensure that the worst-case overhead in each individual step is fixed and independent of the total depth of the computation. The idea is to equally divide the process of solving the puzzle corresponding to position 2^i among all 2^{i-1} steps in the interval $[2^{i-1}, 2^i)$, so that at every step we incur a fixed overhead. This can be easily done by obtaining the corresponding puzzle already at step 2^{i-1} and unrolling the puzzle computation through the next 2^{i-1} steps.

Even after these optimizations a caveat of the solution is that the space complexity of homomorphic evaluation may now grow with the space complexity of solving puzzles, which could be as large as the difficulty T. This is in contrast to schemes based on circular security where the required space will be the same as required for the evaluated computation. To circumvent this additional overhead, we may use a leveled scheme which requires space proportional to space used by the evaluated computation, along stronger puzzles, which can be solved using fixed space, independent of the difficulty parameter. We note that existing lattice-based constructions (for example [BV11, BGV12, GSW13]) of leveled scheme satisfy this space-preserving condition. We also note such puzzles follow for example from the sequential hardness of iterated squaring, previously used in the context of verifiable delay functions [EFKP20]. Alternatively, such puzzles may be constructed using the same recipe of [BGJ+16] assuming the complexity assumption that there exists $d \in \mathbb{N}$ such that for all $c \in \mathbb{N}$, $\mathbf{P} \cap \mathbf{DSPACE}(n^d) \not\subset \mathbf{ioSIZE}(n^c)$ and assuming a space-preserving succinct randomized encoding scheme. While there do exist such constructions based on (super-polynomial domain) IO (c.f. [CH16]), a construction from functional encryption (and in particular, polynomial assumptions) is not known. We conjecture that it may be constructed using existing techniques [AL18, GS18, KNTY19] and

[1] We note that in the ITCS 2023 proceedings version of this paper, a significantly more complicated solution, based on decomposalbe IO [LZ21], was presented.

leave it for future work.

2 Preliminaries

Strings And Binary Representation:

- Given a string $x \in \{0,1\}^*$, let $|x|$ be its length.

Efficient Computation:

- We denote by PPT probabilistic polynomial-time Turing machines.
- For a PPT algorithm M, we denote by $M(x;r)$ the output of M on input x and random coins r, and by $M(x)$ the random variable, given by sampling the coins r uniformly at random.
- A polynomial-size circuit family $\mathcal{C}$ is a sequence of circuits $\mathcal{C} = \{ C_\lambda \}_{\lambda \in \mathbb{N}}$, such that each circuit C_λ is of polynomial size $\lambda^{O(1)}$ and has $\lambda^{O(1)}$ input and output bits. We also consider probabilistic circuits that may toss random coins.

Indistinguishability:

- A function $f : \mathbb{N} \to [0,1]$ is negligible if $f(\lambda) = \lambda^{-\omega(1)}$ and is noticeable if $f(\lambda) = \lambda^{-O(1)}$.
- Two ensembles of random variables $\mathcal{X} = \{X_i\}_{\lambda \in \mathbb{N}, i \in I_\lambda}$, $\mathcal{Y} = \{Y_i\}_{\lambda \in \mathbb{N}, i \in I_\lambda}$ over the same set of indices $I = \cup_{\lambda \in \mathbb{N}} I_\lambda$ are said to be *computationally indistinguishable* (respectively, *statistically indistinguishable*), denoted by $\mathcal{X} \approx_c \mathcal{Y}$ (respectively, $\mathcal{X} \approx_s \mathcal{Y}$), if for every polynomial-size (respectively, unbounded) distinguisher $\mathcal{A} = \{ \mathcal{A}_\lambda \}_{\lambda \in \mathbb{N}}$ there exists a negligible function negl such that for all $\lambda \in \mathbb{N}, i \in I_\lambda$,

$$|\Pr(\mathcal{A}(X_i) = 1) - \Pr(\mathcal{A}(Y_i) = 1)| \leq \text{negl}(\lambda) \ .$$

2.1 Relaxed Time-Lock Puzzles

A puzzle instance is associated with a pair of parameters: a security parameter λ determining the cryptographic security of the puzzle, and a difficulty parameter t determining how computationally difficult is to solve the puzzle.

Definition 2.1 (Puzzle)**.** *A puzzle scheme* Puzzle *consists of two algorithms* $\mathsf{Puzzle} = (\mathsf{Gen}, \mathsf{Solve})$ *satisfying the following:*

- **Syntax***:*
  - $Z \leftarrow \mathsf{Gen}(t, s)$ *is a probabilistic algorithm that takes as input a difficulty parameter t and a solution $s \in \{0,1\}^\lambda$, where λ is a security parameter, and outputs a puzzle Z.*
  - $\mathsf{Solve}(Z)$ *is a deterministic algorithm that takes as input a puzzle Z and outputs a solution s.*
- **Completeness***: For every security parameter $\lambda \in \mathbb{N}$, difficulty parameter $t \in \mathbb{N}$, solution $s \in \{0,1\}^\lambda$ and a puzzle $Z \in \{0,1\}^*$ in the support of* $\mathsf{Gen}(t, s)$*,* $\mathsf{Solve}(Z)$ *outputs s.*
- **Efficiency***:*
  - $Z \leftarrow \mathsf{Gen}(t, s)$ *can be computed in time* $\text{poly}(\log(t), \lambda)$*.*
  - $\mathsf{Solve}(Z)$ *can be computed in time* $t \cdot \text{poly}(\lambda)$*.*

Definition 2.2 (Relaxed Time-Lock Puzzle)**.** *A puzzle scheme* Puzzle *is said to be Relaxed Time-Lock Puzzle (RTLP) if for every polynomial $q(\cdot)$ there exists a polynomial $t(\cdot)$ such that for any large enough $\lambda \in \mathbb{N}$ and $s_0, s_1 \in \{0,1\}^\lambda$ and adversary $\mathcal{A}$ of size at most q, it holds that*

$$\Pr\left[b \leftarrow \mathcal{A}(Z) \;\middle|\; \begin{array}{c} b \leftarrow \{0,1\} \\ Z \leftarrow \mathsf{Gen}(t(\lambda), s_b) \end{array}\right] \leq \frac{1}{2} + \frac{1}{q(\lambda)} \ .$$

Where the probability is also over the random coins tossed by $\mathcal{A}$.

Comparison to previous notions of TLPs. The classical notion of TLPs [RSW96] focuses on the depth of the adversary rather than its total size, which is allowed to be polynomially larger than the difficulty parameter. Bitansky et al. [BGJ+16] also consider a relaxed version of TLPs, which focuses on size rather than depth as does our definition. Their definition of RTLP is stronger. Specifically, they require a fixed polynomial gap between q and t. That is, there exists a constant $\varepsilon < 1$ such that the definition holds for any two polynomials q, t such that $q \leq t^\varepsilon$. In contrast, in our definition, there is no such fixed gap.

The (non-relaxed) TLP constructed in [BGJ+16] relies on SRE in conjunction with a complexity assumption about non-parallelizable languages. We observe that RTLP can be obtained by a similar construction and proof as in [BGJ+16] with the following weaker complexity assumption, which essentially says that non-uniformity doesn't provide arbitrary polynomial speed-up for uniform polynomial computations.

Assumption 2.1. *For any polynomial $q(\cdot)$ there exists a polynomial $Q(\cdot)$ and a language $\mathcal{L} \in \boldsymbol{DTIME}(Q)$ such that any family $\mathcal{C} = \{\mathcal{C}_\lambda\}$ of size-$q(\lambda)$ circuits fails to decide $\mathcal{L}_\lambda = \mathcal{L} \cap \{0,1\}^\lambda$ for all large enough λ.*[2]

Theorem 2.3 (RTLP under SRE and Assumption 2.1)**.** *Assuming SRE and Assumption 2.1 there exists Relaxed Time-Lock Puzzle.*

We find Assumption 2.1 rather mild. It can be seen as a generalization of the (unconditional) time-hierarchy theorem for uniform computation. It is in the same spirit of Nisan-Impagliazzo-Wigderson style assumptions used to derandomize BPP [NW94, IW97]. We also note that this assumption follows easily for instance from the non-uniform **ETH** assumption [BPSS23]. For completeness we provide the construction of RTLP and its security proof in Appendix A.

Next, since SRE can be constructed from semi-compact FE [AL18, GS18, KNTY19], we have the following corollary:

Corollary 2.4 (RTLP under FE and Assumption 2.1)**.** *Assuming semi-compact FE and Assumption 2.1, there exists Relaxed Time-Lock Puzzle.*

2.1.1 Uniform Relaxed Time-Lock Puzzle

RTLP can be further relaxed by considering only unifrom PPT adversaries. Note in this case the challenge is also sampled uniformly by the adversary.

Definition 2.5 (Uniform Relaxed Time-Lock Puzzle)**.** *A puzzle scheme* Puzzle *is said to be Uniform Relaxed Time-Lock Puzzle (denoted URTLP) if for every polynomial $q(\cdot)$ there exists a polynomial $t(\cdot)$ such that for any large enough $\lambda \in \mathbb{N}$ and uniform PPT adversary $\mathcal{A} = (M, D)$ where the running time of both M and D is at most q, it holds that:*

$$\Pr\left[b \leftarrow D(a, Z) \;\middle|\; \begin{array}{c} a, s_0, s_1 \leftarrow M(1^\lambda), \text{where } s_0, s_1 \in \{0,1\}^\lambda \text{ and } a \in \{0,1\}^* \\ b \leftarrow \{0,1\} \\ Z \leftarrow \mathsf{Gen}(t(\lambda), s_b) \end{array}\right] \leq \frac{1}{2} + \frac{1}{q(\lambda)} \ .$$

Where the probability is also over the random coins tossed by $\mathcal{A}$.

In the uniform setting Assumption 2.1 is not required, and the following is theorem B.5 from [BGJ+16].

[2] This is a more explicit statement of the assumption that for any $c \in \mathbb{N}$, $\mathbf{P} \not\subseteq \mathbf{ioSIZE}(n^c)$, which is stated in the introduction.

Theorem 2.6 (URTLP exists under SRE)**.** *Assuming SRE and OWF, there exists Uniform Relaxed Time-Lock Puzzle.*

And again, since SRE can be constructed from semi-compact FE [AL18, GS18, KNTY19] we have the following corollary:

Corollary 2.7 (Uniform Relaxed Time-Lock Puzzle under FE)**.** *Assuming semi-compact FE, there exists Uniform Relaxed Time-Lock Puzzle.*

2.2 Indistinguishability Obfuscation

We now define Indistinguishability Obfuscation ($i\mathcal{O}$) for circuits. This notion guarantees that the obfuscations of two circuits are computationally indistinguishable, as long as the circuits are of the same size and are functionally equivalent. We restrict the definition to circuits with a polynomial size input space.

Definition 2.8 (Polynomial Domain Secure Indistinguishability Obfuscation ($i\mathcal{O}$) For Circuits)**.** *A uniform PPT algorithm $i\mathcal{O}$ is called polynomial domain secure indistinguishability obfuscator for polynomial-sized circuits if the following holds:*

- **Correctness.** *For every $\lambda \in \mathbb{N}$, circuit C, and circuit C' in the support of $i\mathcal{O}(1^\lambda, C)$, the circuits C and C' are functionally equivalent.*
- **Polynomial Domain Security.** *For every constant $c \in \mathbb{N}$ and every two circuit ensembles $\{C_{0,\lambda}\}_{\lambda\in\mathbb{N}}$ and $\{C_{1,\lambda}\}_{\lambda\in\mathbb{N}}$ of polynomial-sized circuits with input length $c \cdot \log(\lambda)$ where for every $\lambda \in \mathbb{N}$, $C_{1,\lambda}$ and $C_{1,\lambda}$ are of the same size and are functionally equivalent, it holds that:*

$$\{i\mathcal{O}(1^\lambda, C_{0,\lambda})\}_{\lambda\in\mathbb{N}} \approx_c \{i\mathcal{O}(1^\lambda, C_{1,\lambda})\}_{\lambda\in\mathbb{N}} \ .$$

Theorem 2.9 ($i\mathcal{O}$ Exists assuming FE [AJ15, BV18, LZ21])**.** *Assuming polynomial secure FE, there exists polynomial domain secure indistinguishability obfusactor $i\mathcal{O}$.*

2.3 Puncturable pseudorandom function

Definition 2.10 (PPRF)**.** *Let $n(\cdot), m(\cdot)$ be some polynomials. A puncturable pseudorandom function (PPRF) consists of efficient algorithms $\mathsf{PPRF} = (\mathsf{Punc}, \mathsf{Eval})$ where:*

- **Syntax:**
  - $\mathsf{Punc}(s, x)$ *is a deterministic algorithm that punctures a seed s at a point $x \in \{0,1\}^{n(\lambda)}$ and outputs $s\{x\}$. We call $s\{x\}$ the punctured seed at the point x.*
  - $\mathsf{Eval}(s, x)$ *is a deterministic algorithm that takes as input a seed s and query input $x \in \{0,1\}^{n(\lambda)}$ and outputs $y \in \{0,1\}^{m(n)}$.*
- **Correctness:** *For every $\lambda \in \mathbb{N}$ and two distinct inputs $x, x' \in \{0,1\}^{n(\lambda)}$ it holds that*

$$\mathsf{Eval}(s, x') = \mathsf{Eval}(s\{x\}, x') \ ,$$

  where $s \leftarrow \{0,1\}^\lambda$ and $s\{x\} \leftarrow \mathsf{Punc}(s, x)$.
- **Pseudorandomness at the punctured point:**

$$\{s\{x\}, \mathsf{Eval}(s, x)\}_{\lambda\in\mathbb{N}, x\in\{0,1\}^{n(\lambda)}} \approx_c \{s\{x\}, r\}_{\lambda\in\mathbb{N}, x\in\{0,1\}^{n(\lambda)}} \ ,$$

  where $s \leftarrow \{0,1\}^\lambda$ and $s\{x\} \leftarrow \mathsf{Punc}(s, x)$, and $r \leftarrow \{0,1\}^{m(\lambda)}$ is sampled in uniform.

2.4 Homomorphic Encryption

Definition 2.11 (Encryption Scheme)**.** *An encryption scheme consists of three PPT algorithms (*Gen*,* Enc*,* Dec*) satisfying:*

- **Syntax:**
  - $\mathsf{Gen}(1^\lambda)$ *is a probabilistic algorithm that takes as input* 1^λ *and outputs the keys* $\mathsf{dk}, \mathsf{ek}, \mathsf{evk} \in \{0,1\}^*$.
  - $\mathsf{Enc}_{\mathsf{ek}}(m)$ *is a probabilistic algorithm that takes as input a message* $m \in \{0,1\}^*$ *and a key* $\mathsf{ek} \in \{0,1\}^*$ *and outputs a ciphertext* $ct \in \{0,1\}^*$.
  - $\mathsf{Dec}_{\mathsf{dk}}(ct)$ *is a determinisitc algorithm that takes as input a ciphertext* $ct \in \{0,1\}^*$ *and a key* $\mathsf{dk} \in \{0,1\}^*$ *and outputs a message* $m \in \{0,1\}^*$.
- **Correctness:** *For any* $\lambda \in \mathbb{N}$ *and* $(\mathsf{dk}, \mathsf{ek}, \mathsf{evk}) \leftarrow \mathsf{Gen}(1^\lambda)$*, message* $m \in \{0,1\}^*$*,*

$$\mathsf{Dec}_{\mathsf{dk}}(\mathsf{Enc}_{\mathsf{ek}}(m))) = m$$

- **Semantic Security.** *For any polynomial-size distinguisher* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ *there exists a negligible function* negl *such that for all* $\lambda \in \mathbb{N}$*,*

$$|\Pr[\mathcal{A}(\mathsf{ek}, \mathsf{evk}, \mathsf{Enc}_{\mathsf{ek}}(0)) = 1] - \Pr[\mathcal{A}(\mathsf{ek}, \mathsf{evk}, \mathsf{Enc}_{\mathsf{ek}}(1)) = 1]| \leq \mathrm{negl}(\lambda) \ ,$$

where $(\mathsf{dk}, \mathsf{ek}, \mathsf{evk}) \leftarrow \mathsf{Gen}(1^\lambda)$.

Definition 2.12 (Fully Homomorphic Encryption)**.** *A scheme* FHE *is said to be fully homomorphic encryption scheme if it is an encryption scheme, with an additional PPT algorithm* Eval*, with the following additional properties:*

1. **Full homomorphism:** *For any polynomial* $\ell(\lambda)$*, large enough* $\lambda \in \mathbb{N}$*, circuit* C *of size at most* $\ell(\lambda)$*, and message* m*,*
$$\Pr_{(\mathsf{dk},\mathsf{ek},\mathsf{evk})\leftarrow\mathsf{Gen}(1^\lambda)}[\mathsf{Dec}_{\mathsf{dk}}(\mathsf{Eval}_{\mathsf{ek}}(C, \mathsf{Enc}_{\mathsf{ek}}(m))) = C(m)] = 1 \ .$$
2. **Compactness** *There exists a fixed polynomial* poly*, such that for any* $\lambda \in \mathbb{N}$ *and* $(\mathsf{dk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda)$*, message* $m \in \{0,1\}^*$*, and circuit* C *with input size* $|m|$
$$|\mathsf{Eval}_{\mathsf{evk}}(C, \mathsf{Enc}_{\mathsf{ek}}(m))| \leq \mathrm{poly}(|C(m)|, \lambda) \ .$$

2.4.1 Chain Homomorphic Encryption

We now turn to define a chain homomorphic encryption scheme. The definition captures the idea of iteratively evaluating the circuit level by level using a chain of encrypted keys, where $\mathsf{Enc}_{\mathsf{ek}_i}(\mathsf{dk}_{i-1})$ is used to evaluate the i-th level of the circuit. This is a specific form of leveled homomorphic schemes, which is satisfied by common constructions. The definition abstracts away the low level details of Gentry's construction [Gen09]. We first define an ℓ-key-chain, which is an array of length ℓ where the i-th element is $\mathsf{Enc}_{\mathsf{ek}_i}(\mathsf{dk}_{i-1})$.

Definition 2.13 (ℓ-key-chain)**.** *Let* CHE *be an encyrption scheme, let* $\ell \in \mathbb{N}$*, and let* $(\mathsf{ek}_0, \mathsf{dk}_0), ..., (\mathsf{ek}_\ell, \mathsf{dk}_\ell)$ *be* $\ell + 1$ *tuples of keys, where for each* i*,* $(\mathsf{ek}_i, \mathsf{dk}_i)$ *are in the support of* Gen*. An* ℓ*-key-chain is the following array of ciphertexts*

$$L_\ell = [\mathsf{Enc}_{\mathsf{ek}_1}(\mathsf{dk}_0), ..., \mathsf{Enc}_{\mathsf{ek}_\ell}(\mathsf{dk}_{\ell-1})] \ .$$

We say a chain begins with dk_0 *to indicate the first element in the array* L_ℓ *is* $\mathsf{Enc}_{\mathsf{ek}_1}(\mathsf{dk}_0)$*. Similarly, we say the chain ends with* ek_ℓ *to indicate the last element in the array is* $\mathsf{Enc}_{\mathsf{ek}_\ell}(\mathsf{dk}_{\ell-1})$.

Definition 2.14 (Chain Homomorphic Encryption)**.** *An encryption scheme* CHE *is said to be chain homomorphic if there exists a deterministic polynomial algorithm* CEval *for which the following holds:*

- **Chain homomorphism:** *For any* $\ell, \lambda \in \mathbb{N}$ *and* $(\mathsf{dk}_0, \mathsf{ek}_0) \leftarrow \mathsf{CHE.Gen}(1^\lambda)$*, message* $m \in \{0,1\}^*$*, and circuit* C *with input size* $|m|$ *and depth at most* ℓ*, and a* ℓ*-key-chain* L_ℓ *beginning with* dk_0 *and ending with* ek_ℓ

$$\mathsf{Dec}_{\mathsf{dk}_\ell}(\mathsf{CEval}(L_\ell, C, \mathsf{Enc}_{\mathsf{ek}_0}(m))) = C(m) \ .$$

- **Compactness** *There exists a fixed polynomial* poly*, such that for any* $\ell, \lambda \in \mathbb{N}$ *and* $(\mathsf{dk}_0, \mathsf{ek}_0) \leftarrow \mathsf{Gen}(1^\lambda)$*, message* $m \in \{0,1\}^*$*, and circuit* C *with input size* $|m|$ *and depth at most* ℓ*, and a* ℓ*-key-chain* L_ℓ *beginning with* dk_0*,*

$$|\mathsf{CEval}(L_\ell, C, \mathsf{Enc}_{\mathsf{ek}_0}(m))| \leq \mathrm{poly}(|C(m)|, \lambda) \ .$$

3 Constructing Fully Homomorphic Encryption

3.1 FHE Construction Details

In this section we build fully homomorphic encryption scheme secure against polynomial adversaries.

Chain of encrypted keys. Our FHE encryption scheme has in the background an exponentially long ($\approx 2^\lambda$-long) chain of encrypted keys. The i-th node in the chain is the $(i-1)$-th decryption key encrypted under the i-th encryption key, i.e. $\mathsf{CHE.Enc}_{\mathsf{ek}_i}(\mathsf{dk}_{i-1})$.

Partition the chain to intervals. We partition the range $[0, 2^\lambda - 1]$ into λ intervals. For $1 \leq j \leq \lambda - 1$, the j-th interval is of length 2^j and consists of the nodes in the range $[2^j, 2^{j+1} - 1]$ in the chain. For $j = 0$ we add 0 to the first interval, and set the interval to be $[0, 1]$. For $0 \leq i \leq 2^\lambda - 1$ we denote by $J(i)$ the index j of the corresponding interval where i resides.

Compressing the intervals. The first interval $j = 0$ is just $\mathsf{CHE.Enc}_{\mathsf{ek}_1}(\mathsf{dk}_0)$. For $j \geq 1$, the scheme compresses the intervals using the circuits $\{\mathsf{EVK}_j\}$ and puzzles $\{Z_j\}$ in the following manner:

- The first node of the j-th interval, corresponding to the index 2^j in the chain, is provided by the puzzle $Z_j = \mathsf{Puzzle.Gen}(i, \mathsf{CHE.Enc}_{\mathsf{ek}_i}(\mathsf{dk}_{i-1}))$.
- The rest of the nodes are compressed by EVK_j. The circuit EVK_j admits inputs of length j, where a string in $\{0,1\}^j$ is identified as an integer in the range $[2^j, 2^{j+1} - 1]$ in the natural way. On input $i \in [2^j + 1, 2^{j+1} - 1]$ the circuit EVK_j outputs $\mathsf{CHE.Enc}_{\mathsf{ek}_i}(\mathsf{dk}_{i-1})$ (for $i = 2^j$ the circuit outputs $\perp$).

Randomness. The randomness for generating the nodes in the chain is derived using a PPRF. The j-th interval has a corresponding PPRF seed S_j. The seed is used for deriving randomness for the keys and the encryption.

Ciphertexts. A ciphertext in this scheme is a ciphertext of the underlying leveled homomorphic scheme CHE, paired with a key index, which is the index of the decryption key from the chain that should be used for decryption. Formally, an encryption in the scheme is a tuple $\mathsf{ct} = (c, i)$, where i represents the key index in the chain, and the decryption of $\mathsf{ct} = (c, i)$ is $\mathsf{CHE.Dec}_{\mathsf{dk}_i}(c)$. So a fresh encryption is encrypted under ek_0, i.e. of the form $(\mathsf{ct}, 0)$, and the output of a homomorphic evaluation of a circuit of depth d_1 is decryptable under dk_{d_1}, i.e. of the form (c, d_1).

Keys.

- ek: The encryption key of the scheme is ek_0.
- evk: The evaluation key is $(\mathsf{CHE.Enc}_{\mathsf{ek}_1}(\mathsf{dk}_0), \{(i\mathcal{O}(\mathsf{EVK}_j), Z_j)\})$ (the circuits EVK_j are defined formally in Fig. 1 and are padded to some upper bound.)
- dk: The decryption key is the set of seeds $\{S_j\}$ from which the randomness for generating the keys is derived using PPRF.

Function evaluation. A function f is given to $\mathsf{FHE.Eval}$ as a circuit compatible with CEval. $\mathsf{FHE.Eval}$ evaluates the function by first producing long enough key-chain L, and then evaluating $\mathsf{CEval}(L, C, ct)$. Finally, the output of the $\mathsf{FHE.Eval}$ algorithm contains the output of the last evaluated layer, plus the index of the current position in the chain of keys, which is the key under which the output is decryptable.

3.2 FHE Formal Description

In this section we formally describe the construction. The construction uses the following primitives:

- Puncturable pseudorandom function PPRF.
- Chain homomorphic encryption scheme CHE.
- Polynomial domain secure indistinguishability obfuscation for circuits iO.
- Puzzle scheme Puzzle.

For simplicity of representation we assume w.l.o.g. the following about CHE schemes:

- $\mathsf{CHE.Gen}$ and $\mathsf{CHE.Enc}$ use λ bits of randomness and that $\mathsf{CHE.CEval}$ is deterministic.
- We assume that a decryption key dk is of size d for some function $d = \lambda^{\varepsilon}$, and that an encryption $\mathsf{CHE.Enc}_{\mathsf{ek}}(\mathsf{dk})$ of a decryption key dk is of size λ.

We also assume w.l.o.g. the output of the PPRF is of length 2λ, where the first λ bits of $\mathsf{PPRF.Eval}(S_{j(i)}, i)$ are used as randomness for generating $\mathsf{ek}_i, \mathsf{dk}_i$, and the second λ bits are used as randomness for the encryption procedure itself, i.e. in order to encrypt dk_{i-1} under ek_i.

The scheme is defined formally by the following algorithms:

Algorithm $\mathsf{FHE.Gen}$

Input: A security parameter $\lambda \in \mathbb{N}$.

1. Sample PPRF seeds $S_0 ..., S_{\lambda-1} \leftarrow \{0,1\}^{\lambda}$.
2. Generate decryption keys for the indices $\{\, 2^k - 1 \mid 1 \leq k \leq \lambda - 1 \,\}$ and encryption keys for the indexes $\{\, 2^k \mid 1 \leq k \leq \lambda - 1 \,\}$. Each such key ek_i / dk_i is generated using pseudorandom coins derived from $S_{j(i)}$.
3. For $j = 0, ...\lambda - 1$, encrypt dk_{2^j-1} under ek_{2^j} by
$$ct_j \leftarrow \mathsf{CHE.Enc}_{\mathsf{ek}_{2^j}}(\mathsf{dk}_{2^j-1}) \ .$$
4. For $j = 1, ...\lambda - 1$ generate puzzle with difficulty 2^j by
$$Z_j \leftarrow \mathsf{Puzzle.Gen}(2^j, ct_j) \ .$$
5. For $j = 1, ...\lambda - 1$ generate obfuscated key-chain circuits by
$$\mathsf{evk}_j \leftarrow i\mathcal{O}(1^{\lambda}, \mathsf{EVK}_j) \ ,$$
where the seed S_j is hardwired in EVK_j.
6. Set $\mathsf{evk}_0 = ct_0$.
7. Set $\mathsf{evk} = \mathsf{evk}_0, (Z_1, \mathsf{evk}_1), ..., (Z_{\lambda-1}, \mathsf{evk}_{\lambda-1})$.
8. Set $\mathsf{dk} = S_0, ..., S_{\lambda-1}$.
9. Set $\mathsf{ek} = \mathsf{ek}_0$.
10. Output $\mathsf{dk}, \mathsf{ek}, \mathsf{evk}$.

Algorithm FHE.Enc

Input: An encryption key $\mathsf{ek} = \mathsf{ek}_0$ and a message $m \in \{0,1\}^*$

1. Encrypt $ct \leftarrow \mathsf{CHE.Enc}_{\mathsf{ek}_0}(m)$.
2. Output $(ct, 0)$.

Algorithm FHE.Dec

Input: A decryption key $\mathsf{dk} = S_0, ..., S_{\lambda-1}$ and a ciphertext $ct = (ct', \ell)$

1. Generate dk_ℓ using the pseudorandom coins $\mathsf{PPRF.Eval}(S_{j(\ell)}, \ell)$ by
$$\mathsf{dk}_\ell \leftarrow \mathsf{CHE.Gen}(1^\lambda) \ ,$$
2. Output the decryption $\mathsf{CHE.Dec}_{\mathsf{dk}_\ell}(ct')$.

Algorithm FHE.Eval

Input:

- A circuit C of depth ℓ, compatible with CEval.
- A ciphertext $ct = (ct',\ 0)$.
- Evaluation key $\mathsf{evk} = ct_0, (Z_1, \mathsf{evk}_1), ..., (Z_{\lambda-1}, \mathsf{evk}_{\lambda-1})$.

Evaluation:

1. Generate ℓ-key-chain L_ℓ by computing $\mathsf{evk}_{j(i)}(i)$ for $i = 1, ..., \ell$ and solving the corresponding puzzles Z_j along the way, i.e. for indexes $\ell \geq i = 2^k$ for $k \geq 1$.
2. Evaluate $\mathsf{CEval}(L_\ell, C, ct')$ and output the result.

3.3 Security

In this section we state and prove the security of the scheme.

Theorem 3.1 (FHE scheme is secure)**.** *If $i\mathcal{O}$ is polynomial domain secure indistinguihsability obfuscator,* PPRF *is a puncutrable pseudorandom function,* CHE *is a semantically secure leveled homomorphic encryption scheme, and* Puzzle *is URTLP, then the* FHE *scheme is semantically secure against uniform PPT adversaries. Furthermore, if* Puzzle *is also RTLP, then the scheme is semantically secure against non-uniform polynomial adversaries as well.*

Proof. We prove security against non-uniform adversaries assuming the puzzle is sound against non-uniform adversaries. The uniform case is analogous. Let $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ be a poly-size adversary. We go through a series of hybrids. Fix $b \in \{0,1\}$, and let p_i^b the probability of $\mathcal{A}$ to output 1 in hybrid i when it gets an encryption of b, i.e.

$$p_i^b = \Pr\left[\mathcal{A}(\mathsf{ek}, \mathsf{evk}, \mathsf{Enc}_{\mathsf{ek}}(b)) = 1\right] \ .$$

Let $\delta(\cdot) = \frac{1}{\mathrm{poly}(\cdot)}$ be some inverse polynomial function. We will show that for large enough λ it holds that

$$\left|p_0^0 - p_0^1\right| \leq \delta \ ,$$

and achieve semantic security since δ was arbitrary inverse polynomial function. Throughout the proof we use the following common notations:

$$\begin{aligned} ct_i &\leftarrow \mathsf{CHE.Enc}_{\mathsf{ek}_i}(\mathsf{dk}_{i-1}) \\ ct'_i &\leftarrow \mathsf{CHE.Enc}_{\mathsf{ek}_i}(0^d) \\ Z_j &\leftarrow \mathsf{Puzzle.Gen}(2^j, ct_{2^j}) \\ Z'_j &\leftarrow \mathsf{Puzzle.Gen}(2^j, ct'_{2^j}) \ . \end{aligned}$$

We go through the following hybrids:

Circuit EVK_j, for $j \geq 1$

Hardwired values:

- Seed S_j.

Input: An index $i \in [2^j + 1, 2^{j+1} - 1]$ (described as a j bits string).

1. Generate pseudorandom coins by

$$(r_g^{i-1}, r_e^{i-1}) = \mathsf{PPRF.Eval}(S_j, i-1) \ ,$$
$$(r_g^{i}, r_e^{i}) = \mathsf{PPRF.Eval}(S_j, i) \ .$$

2. Generate $i-1$-th decryption key using pseudorandom coins r_g^{i-1} by

$$(\mathsf{dk}_{i-1}, \mathsf{ek}_{i-1}) = \mathsf{CHE.Gen}(1^\lambda; r_g^{i-1}) \ .$$

3. Generate i-th encryption key using pseudorandom coins r_g^i by

$$(\mathsf{dk}_{i}, \mathsf{ek}_{i}) = \mathsf{CHE.Gen}(1^\lambda; r_g^{i}) \ .$$

4. Encrypt dk_{i-1} under ek_i using pseudorandom coins r_e^i by

$$ct_i = \mathsf{CHE.Enc}_{\mathsf{ek}_i}(\mathsf{dk}_{i-1}; r_e^i) \ .$$

5. Output ct_i.

Figure 1: Circuit for generating the j-interval of the key chain.

- $\mathcal{H}_0$: This is the original security game, when $\mathcal{A}$ gets an encryption of b.
- $\mathcal{H}_1$: This is like $\mathcal{H}_0$, except that for some interval j^*, the puzzle Z_{j^*}, wrapping an encryption of $\mathsf{dk}_{2^{j^*}}$, is switched with the puzzle Z'_{j^*}, wrapping an encryption of 0^d instead. It follows by puzzle security that there exists an index $j^* = O(\log(\lambda))$, independent of b, such that $|p_0^b - p_1^b| \leq \frac{\delta(\lambda)}{3}$. This is formally proven in Proposition 3.2.
- $\mathcal{H}_2^j$ for $j = j^* - 1, ..., 1$: Each such hybrid is like the previous one, except that the j-th chain interval generator circuit EVK_j (Fig. 1) is switched with the circuit EVK'_j (Fig. 2), generating a dummy interval containing encryptions of 0^d. Also, the puzzle Z_j, wrapping an encryption of dk_{2^j}, is switched with the puzzle Z'_j, wrapping encryption of 0^d. Indistinguishability between $\mathcal{H}_2^{j^*-1}$ and $\mathcal{H}_1$ and between two consecutive hybrids $\mathcal{H}_2^j$ and $\mathcal{H}_2^{j-1}$ is proved in Proposition 3.3.
- $\mathcal{H}_3$: This is like $\mathcal{H}_2^1$, but evk_0 is an encryption of 0^d under ek_1, instead of an encryption of dk_0. Moreover, the encryption key $\mathsf{ek} = \mathsf{ek}_0$ is sampled using independent randomness. Indistinguishability is proved in Proposition 3.4.
- $\mathcal{H}_4$: This is like $\mathcal{H}_3$, but we switch to an encryption of 0 instead of encryption of b. In particular, p_4^b is in fact independent of b. Indistinguishability follows directly from the semantic security of the scheme CHE.

Now, when we switched between $\mathcal{H}_0$ and $\mathcal{H}_1$, the probability of $\mathcal{A}$ to output 1 changed by at most $\frac{\delta}{3}$. In the rest of the hybrids the probability changed by at most $\mu(\lambda)$ for some negligible function $\mu(\cdot)$, since there are $O(2^{j^*}) = \mathrm{poly}(\lambda, \frac{1}{\delta(\lambda)}) = \mathrm{poly}(\lambda)$ many such hybrids and between two consecutive hybrids it changed only negligibly. Finally, by triangle inequality, it follows that

$$\left|p_0^0 - p_0^1\right| \leq \left|p_0^0 - p_4^0\right| + \left|p_0^1 - p_4^0\right| \leq \frac{\delta(\lambda)}{3} + \frac{\delta(\lambda)}{3} + 2\mu(\lambda) < \delta(\lambda) \ ,$$

Circuit EVK'_j, for $j \geq 1$

Hardwired values:

- Seed S_j.

Input: An index $i \in [2^j + 1, 2^{j+1} - 1]$ (described as a j bits string).

1. Generate pseudorandom coins by
$$(r_g^i, r_e^i) = \mathsf{PPRF.Eval}(S_j, i) \ .$$
2. Generate i-th encryption key using pseudorandom coins r_g^i by
$$(\mathsf{dk}_i, \mathsf{ek}_i) = \mathsf{CHE.Gen}(1^\lambda; r_g^i) \ .$$
3. Encrypt 0^ℓ under ek_i using pseudorandom coins r_e^i by
$$ct_i = \mathsf{CHE.Enc}_{\mathsf{ek}_i}(0^d; r_e^i) \ .$$
4. Output ct_i.

Figure 2: Circuit for generating a garbage j-interval

for large enough λ. □

Proposition 3.2 (Switching Puzzles)**.** *Let $\delta = \frac{1}{\mathrm{poly}(\lambda)}$, and let $\mathcal{A}$ be an adversary of size* $\mathrm{poly}(\lambda)$*. Then there exists an interval $j^* = O(\log(\lambda))$ such that with respect to it*

$$\forall b \in \{0,1\} \ \ |p_0^b - p_1^b| \leq \frac{\delta(\lambda)}{3} \ .$$

Proof. Fix $b \in \{0,1\}$, and let j^* be an interval number to be decided later, let $i^* = 2^{j^*}$, and let

$$ct_{i^*} \leftarrow \mathsf{CHE.Enc}_{\mathsf{ek}_{i^*}}(\mathsf{dk}_{i^*-1})$$
$$ct'_{i^*} \leftarrow \mathsf{CHE.Enc}_{\mathsf{ek}_{i^*}}(0^d) \ .$$

Consider the following non-uniform adversary $\mathcal{B}$ to the puzzle security game. It samples $s_0 = ct_{i^*}$ and $s_1 = ct'_{i^*}$ and sends them to its challenger, and receives a puzzle $\widehat{Z}_{j^*}$ of difficulty i^*. It then uses the puzzle to sample an evaluation key of the scheme and runs $\mathcal{A}$. $\mathcal{B}$ outputs as $\mathcal{A}$ does. Observe that if the puzzle wraps ct_{i^*} then $\mathcal{A}$ runs in the settings of $\mathcal{H}_0$, and if the puzzle wraps ct'_{i^*} then $\mathcal{A}$ runs in the settings of $\mathcal{H}_1$. Therefore it follows that $\mathcal{B}$'s success chances are:

$$\Pr\left[\alpha \leftarrow \mathcal{B}(Z) \;\middle|\; \begin{array}{c} \alpha \leftarrow \{0,1\} \\ Z \leftarrow \mathsf{Puzzle.Gen}(i^*, s_\alpha) \end{array}\right] = \frac{1}{2} + \frac{1}{2}(p_1^b - p_0^b) \ .$$

Consider now a bound on the size of $\mathcal{B}$ and denote it as $q_{\mathcal{B}}(\cdot)$. Let $q_{\mathcal{A}}(\cdot)$ be a polynomial bound on the size of $\mathcal{A}$. Note the size of the $\mathcal{B}$ is a fixed polynomial overhead of the size of $\mathcal{A}$, and in particular this overhead does not depend on j^*. We may assume w.l.o.g. that $q_{\mathcal{B}} \geq \frac{6}{\delta}$. Let $t(\cdot)$ the polynomial guaranteed from the puzzle security with respect to the polynomial $q_{\mathcal{B}}(\cdot)$, and set $j^* = \lceil \log(t) \rceil$. So we can apply the puzzle security, and obtain that for this i^*, $\mathcal{B}$'s success chance is bounded by $\frac{1}{2} + \frac{1}{q_{\mathcal{B}}} \leq \frac{1}{2} + \frac{\delta}{6}$ for large enough λ. Since this argument is symmetrical, we get that $\left|p_1^b - p_0^b\right| \leq \frac{\delta}{3}$. Moreover, by construction $i^* = \mathrm{poly}(\lambda, \frac{1}{\delta(\lambda)}) = \mathrm{poly}(\lambda)$, so $j^* = O(\log(\lambda))$. Finally, note the same argument holds for both $b = 0, 1$.

Remark 3.1. Note that the above reduction is actually uniform, and just needs the index i^*, which only depends on the running time or size of $\mathcal{A}$, its success chance and the overhead of the reduction. In the

non-uniform settings $\mathcal{B}$ can get it as a non-uniform advice. In the uniform case, note that $i^* = 2^{O((\log(\lambda))} = \text{poly}(\lambda)$ for some polynomial poly. The function $i^*(\cdot)$ can be part of the code of the reduction. Therefore, the same proof works in case $\mathcal{A}$ is uniform and the puzzle is URTLP. We also note that if there is a fixed polynomial $\text{poly}(\cdot)$ such that $i^* = \text{poly}(q_{\mathcal{B}})$, one can compute such appropriate i^* given only a bound on $\mathcal{A}$'s running time and success chance, resulting in a single uniform reduction $\mathcal{B}$ to all PPT adversaries. We note the Uniform Relaxed Time-Lock Puzzle from [BGJ+16] has such fixed polynomial gap.

□

Proposition 3.3 (Moving Along The Chain)**.** *Let $j^* = O(\log(\lambda))$ be an interval number. Then hybrids $\mathcal{H}_1$ and $\mathcal{H}_2^{j^*-1}$ are indistinguishable, and for any $1 \leq j \leq j^* - 2$ hybrids $\mathcal{H}_2^{j+1}$ and $\mathcal{H}_2^j$ are indistinguishable.*

Proof. We prove indistinguishability between $\mathcal{H}_2^{j+1}$ and $\mathcal{H}_2^j$, indistinguishability between $\mathcal{H}_1$ and $\mathcal{H}_2^{j^*-1}$ is proved analogously, see Remark 3.2. Fix $1 \leq j \leq j^* - 2$, and consider the following hybrids:

- $\mathcal{T}_{2^{j+1}}$: This is $\mathcal{H}_2^{j+1}$, i.e. we the use j-th chain interval generator circuit EVK_j (Fig. 1) and the puzzle Z_j, which wraps an encryption dk_{2^j}.
- $\mathcal{T}_k$ for $k = 2^{j+1} - 1, ..., 2^j + 1$: In this hybrid we switch to use the hybrid circuit EVK_j^k (Fig. 3), which outputs encryptions of 0^d on inputs $i \geq k$ and encryption of dk_{i-1} on inputs $i < k$.
- $\mathcal{T}_{2^j}$: The circuit hybrid $\mathsf{EVK}_j^{2^j+1}$ is replaced with the dummy interval generator circuit EVK'_j (Fig. 2), generating a dummy interval containing encryptions of 0^d. Also, the puzzle Z_j, wrapping an encryption of dk_{2^j}, is switched with the puzzle Z'_j, wrapping encryption of 0^d.

We start with proving indistinguishability between $\mathcal{T}_{k+1}$ and $\mathcal{T}_k$, for $k = 2^{j+1} - 2, ..., 2^j + 1$. Indistinguishability between $\mathcal{T}_{2^{j+1}}$ and $\mathcal{T}_{2^{j+1}-1}$ is proved analogously, see Remark 3.2. Indistinguishability between $\mathcal{T}_{2^j+1}$ and $\mathcal{T}_{2^j}$ is also proved along the same lines and is handled later. Fix $2^j + 1 \leq k \leq 2^{j+1} - 2$, and consider the following hybrids:

- $\mathcal{T}_{k+1}^0$: This is $\mathcal{T}_{k+1}$, i.e. the hybrid circuit EVK_j^{k+1} is used.
- $\mathcal{T}_k^1$: In this hybrid we replace the seed S_j hardwired in the circuit EVK_j^{k+1} with a punctured seed $S_j[k]$, and use hardwired randomness $(r_g^k, r_e^k) = \mathsf{PPRF.Eval}(S_j, k)$ for generating ek_k and the encryption ct_k, respectively. Let C_1 be the circuit after the above changes. Note that the circuits C_1 and EVK_j^{k+1} are functionally equivalent, and that their input space is of size $\text{poly}(\lambda)$ since they admit input of size $j \leq j^* = O(\log(\lambda))$. Thus the hybrids are indistinguishable by polynomial domain security of $i\mathcal{O}$.
- $\mathcal{T}_{k+1}^2$: In this hybrid the pseudorandom coins $(r_g^k, r_e^k) = \mathsf{PPRF.Eval}(S_j, k)$ are replaced with independent uniform coins $(\widehat{r}_g^k, \widehat{r}_e^k) \leftarrow \{0,1\}^{2\lambda}$. Indistinguishability is due to pseudorandomness at the punctured point of PPRF. Let C_2 be the circuit after the above changes.
- $\mathcal{T}_{k+1}^3$: In this hybrid on the input k a hardwired ciphertext $ct_k = \mathsf{CHE.Enc}_{\mathsf{ek}_k}(\mathsf{dk}_{k-1}; \widehat{r}_e^k)$ is used instead of computed. Let C_3 be the circuit after the above changes. Note that C_2 and C_3 are functionally equivalent, thus the hybrids are indistinguishable by polynomial domain security of $i\mathcal{O}$.
- $\mathcal{T}_i^4$: In this hybrid the ciphertext ct_k is replaced with the ciphertext $ct'_k = \mathsf{CHE.Enc}_{\mathsf{ek}_k}(0^d; \widehat{r}_e^k)$. Let C_4 be the circuit after the above changes. The hybrids are indistinguishable by the semantic security of CHE.
- $\mathcal{T}_{k+1}^5$: In this hybrid we switch back to use pseudorandom coins $(r_g^k, r_e^k) = \mathsf{PPRF.Eval}(S_j, k)$ for generating ek_k and the encryption ct'_k, respectively. Indistinguushability is due to pseudorandomness at the punctured point of PPRF. Let C_5 be the circuit after the above changes.

- $\mathcal{T}^6_{k+1}$: This is $\mathcal{T}_k$, i.e. the hybrid circuit EVK^k_j is used, which outputs an encryption of 0^d also on input k. Note that the circuits EVK^k_j and C_5 are functionally equivalent, thus the hybrids are indistinguishable by polynomial domain security of $i\mathcal{O}$.

Finally, proving that $\mathcal{T}_{2^j+1}$ and $\mathcal{T}_{2^j}$ are indistinguishable can be done along the same lines. In particular, the circuits EVK'_j and $\mathsf{EVK}^{2^j+1}_j$ are functionally equivalent, thus their obfuscations are indistinguishable. Replacing the puzzle is done by puncturing the seed S_j in position 2^j, and then switching from the ciphertext ct_{2^j} to ct'_{2^j}, resulting in switching from Z_j to Z'_j, and then returning to the original seed S_j.

Remark 3.2. Proving that hybrids $\mathcal{T}_{2^{j+1}}$ and $\mathcal{T}_{2^{j+1}-1}$ are indistinguishable is done by the exact same argument. In particular, the puzzle Z'_{j+1} wraps an encryption of 0^d, and thus does not leak information about S_j. This is also the case when proving indistinguishability between $\mathcal{H}_1$ and $\mathcal{H}^{j^*-1}_2$.

□

Proposition 3.4 (Removing dk_0)**.** *Hybrids $\mathcal{H}^1_2$ and $\mathcal{H}_3$ are indistinguishable.*

Proof. We go through the following hybrids:

- $\mathcal{T}_1$: This is $\mathcal{H}^1_2$.
- $\mathcal{T}_2$: In this hybrid the encryption key $\mathsf{ek} = \mathsf{ek}_1$ and the encryption $\mathsf{CHE.Enc}_{\mathsf{ek}_1}(\mathsf{dk}_0)$ are generated using uniform independent random coins instead of using pseudorandom coins. Indistinguushability is due to pseudorandomness at the punctured point of PPRF.
- $\mathcal{T}_3$: In this hybrid we switch to encryption $\mathsf{CHE.Enc}_{\mathsf{ek}_1}(0^d)$ instead of $\mathsf{CHE.Enc}_{\mathsf{ek}_1}(\mathsf{dk}_0)$. Indistinguishability is due to the semantic security of the CHE scheme.

□

References

[ABF+13] Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S. Dov Gordon, Stefano Tessaro, and David A. Wilson. On the relationship between functional encryption, obfuscation, and fully homomorphic encryption. In Martijn Stam, editor, *Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, volume 8308 of *Lecture Notes in Computer Science*, pages 65–84. Springer, 2013.

[ACH20] Thomas Agrikola, Geoffroy Couteau, and Dennis Hofheinz. The usefulness of sparsifiable inputs: How to avoid subexponential io. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part I*, volume 12110 of *Lecture Notes in Computer Science*, pages 187–219. Springer, 2020.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2015.

[AL18] Prabhanjan Ananth and Alex Lombardi. Succinct garbling schemes from functional encryption through a local simulation paradigm. In Amos Beimel and Stefan Dziembowski, editors, *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 455–472. Springer, 2018.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519. Springer, 2014.

[BGJ+16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 345–356. ACM, 2016.

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 439–448. ACM, 2015.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 309–325. ACM, 2012.

[BPSS23] Nir Bitansky, Omer Paneth, Dana Shamir, and Tomer Solomon. Non-interactive universal arguments. Cryptology ePrint Archive, Paper 2023/458, 2023. `https://eprint.iacr.org/2023/458`.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106. IEEE Computer Society, 2011.

[BV18] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *J. ACM*, 65(6):39:1–39:37, 2018.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300. Springer, 2013.

[CH16] Ran Canetti and Justin Holmgren. Fully succinct garbled RAM. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 169–178. ACM, 2016.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 429–437. ACM, 2015.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 468–497. Springer, 2015.

[EFKP20] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Continuous verifiable delay functions. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT*

2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III, volume 12107 of *Lecture Notes in Computer Science*, pages 125–154. Springer, 2020.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.

[GLSW15] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 151–170. IEEE Computer Society, 2015.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001.

[GS18] Sanjam Garg and Akshayaram Srinivasan. A simple construction of io for turing machines. In Amos Beimel and Stefan Dziembowski, editors, *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 425–454. Springer, 2018.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229. ACM, 1997.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 60–73. ACM, 2021.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb {F}_p$, dlin, and prgs in nc^0. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 670–699. Springer, 2022.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 419–428. ACM, 2015.

[KNTY19] Fuyuki Kitagawa, Ryo Nishimaki, Keisuke Tanaka, and Takashi Yamakawa. Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 521–551. Springer, 2019.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 669–684. ACM, 2013.

[LZ21] Qipeng Liu and Mark Zhandry. Decomposable obfuscation: A framework for building applications of obfuscation from polynomial hardness. *J. Cryptol.*, 34(3):35, 2021.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[RSW96] Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, MIT, 1996.

[Zha16] Mark Zhandry. The magic of elfs. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 479–508. Springer, 2016.

A Constructing Relaxed Time-Lock Puzzles

In this section we prove Theorem 2.3. We follow the blueprint of [BGJ+16] to construct Relaxed Time-Lock Puzzle based on SRE and Assumption 2.1, which says that uniform polynomial computations cannot be computed non uniformly in a fixed polynomial size. In other words, non-uniformity doesn't provide arbitrary polynomial speed up for uniform polynomial computations.

Definition A.1 (Succinct Randomized Encoding)**.** *A succinct randomized encoding scheme* SRE *consists of two algorithms* $\mathsf{SRE} = (\mathsf{Encode}, \mathsf{Decode})$ *satisfying the following properties:*

- **Syntax:**
  - $\widehat{M}(x) \leftarrow \mathsf{Encode}(M, x, t, 1^\lambda)$ *is a probabilistic algorithm that takes as input a machine* M*, input* x*, time bound* t*, and a security parameter* 1^λ*. The algorithms outputs a randomized encoding* $\widehat{M}(x)$*.*
  - $y \leftarrow \mathsf{Decode}(\widehat{M}(x))$ *is a deterministic algorithm that takes as input a randomized encoding* $\widehat{M}(x)$ *and computes an output* $y \in \{0,1\}^\lambda$*.*
- **Functionality:** *for every input* x *and machine* M *such that on input* x*,* M *halts in* t *steps and produces a* λ*-bit string, it holds that* $y = M(x)$ *with overwhelming probability over the coins of* Encode*.*
- **Security:** *There exists a PPT simulator* Sim *satisfying: for any poly-size distinguisher* $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ *and polynomials* $m(\cdot), n(\cdot), t(\cdot)$ *there exists a negligible function* $\mu(\cdot)$*, such that for any* λ*, machine* $M \in \{0,1\}^{m(\lambda)}$ *and input* $x \in \{0,1\}^{n(\lambda)}$

$$\left| \Pr\left[\mathcal{D}(\widehat{M}(x)) = 1 \;\middle|\; \widehat{M}(x) \leftarrow \mathsf{Encode}(M, x, t(\lambda), 1^\lambda)\right] - \right.$$
$$\left. \Pr\left[\mathcal{D}(\widehat{S}_y) = 1 \;\middle|\; \widehat{S}_y \leftarrow \mathsf{Sim}(y, 1^{m(\lambda)}, 1^{n(\lambda)}, t(\lambda), 1^\lambda)\right] \right| \leq \mu(\lambda) \ ,$$

  where y *is the output of* $M(x)$ *after* $t(\lambda)$ *steps.*
- **Efficiency:** *For any machine* M *that on input* x *produces a* λ*-bit output in* t *steps:*

– $\mathsf{Encode}(M, x, t, 1^\lambda)$ *can be computed in time* $\mathrm{poly}\log(t) \cdot \mathrm{poly}(|M|, |x|, \lambda)$.

– $\mathsf{Decode}(\widehat{M}(x))$ *can be computed in time* $t \cdot \mathrm{poly}(|M|, |x|, \lambda)$.

RTLP Construction. Let SRE be a succinct randomized encoding scheme. For $s \in \{0,1\}^\lambda$ and $t \leq 2^\lambda$, let M_s^t be a machine that on input $x \in \{0,1\}^\lambda$, outputs the string s after t steps (we assume here that $t \geq \lambda + \omega(1)$). Further, assume that M_s^t is described using 3λ bits (which is possible for large enough λ). Then, the relaxed time-lock puzzle $\mathsf{Puzzle} = (\mathsf{Gen}, \mathsf{Solve})$ is defined as follows:

$\mathsf{Gen}(t, s)$**:** Samples $\widehat{M}_s^t(0^\lambda) \leftarrow \mathsf{SRE.Encode}(M_s^t, 0^\lambda, t, 1^\lambda)$.

$\mathsf{Solve}(Z)$: Outputs $\mathsf{SRE.Decode}(Z)$.

Theorem A.2. *Under Assumption 2.1,* Puzzle *is RTLP.*

Proof. Note that completeness and efficiency follow readily from the functionality and efficiency properties of the SRE scheme. We therefore focus on proving security. Suppose toward contradiction this is not RTLP, so there exists a polynomial $q(\cdot)$ such that for every polynomial $t(\cdot)$ there exists q-size adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ and infinitely many $\lambda \in \mathbb{N}$ and $s_0, s_1 \in \{0,1\}^\lambda$ for which it holds that

$$\Pr\left[b \leftarrow \mathcal{A}_\lambda(Z) \;\middle|\; \begin{array}{c} b \leftarrow \{0,1\} \\ Z \leftarrow \mathsf{Puzzle.Gen}(t(\lambda), s_b) \end{array}\right] \geq \frac{1}{2} + \frac{1}{q(\lambda)} \ . \tag{1}$$

Let $Q(\cdot)$ be any polynomial and let $\mathcal{L} \in \mathbf{DTIME}(Q)$ be a language. Denote by $M_{s_0,s_1}^{\mathcal{L}}$ a machine that on input $x \in \{0,1\}^\lambda$ outputs s_1 if $x \in \mathcal{L}$ and s_0 if $x \notin \mathcal{L}$. For $Q > \lambda$ there exists such machine that runs in time $t = O(Q)$, since $s_0, s_1 \in \{0,1\}^\lambda$ and since $\mathcal{L} \in \mathbf{DTIME}(Q)$. Further, assume w.l.o.g. that $M_{s_0,s_1}^{\mathcal{L}}$ is described by 3λ bits (which is possible for large enough λ), and has the same description length as M_s^t. Let $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ be the corresponding q-size adversary for which Eq. (1) holds, i.e. for infinitely many $\lambda \in \mathbb{N}$ it holds that

$$\Pr\left[\mathcal{A}_\lambda(Z) = 1 \mid Z \leftarrow \mathsf{Puzzle.Gen}(t(\lambda), s_1)\right] - \Pr\left[\mathcal{A}_\lambda(Z) = 1 \mid Z \leftarrow \mathsf{Puzzle.Gen}(t(\lambda), s_0)\right] \geq \frac{2}{q(\lambda)} \ . \tag{2}$$

Consider now the following probabilistic non-uniform decider $\mathcal{B}' = \{\mathcal{B}'_\lambda\}_{\lambda \in \mathbb{N}}$ to $\mathcal{L}$. Given $x \in \{0,1\}^\lambda$, $\mathcal{B}'_\lambda$ acts as follows:

- Samples $Z \coloneqq \widehat{M}_{s_0,s_1}^{\mathcal{L}} \leftarrow \mathsf{SRE.Encode}(M_{s_0,s_1}^{\mathcal{L}}, x, t(\lambda), 1^\lambda)$.
- Obtains $b \leftarrow \mathcal{A}(Z)$.

Success Probability. We now show that $\mathcal{B}'$ distinguishes instances $x \in \mathcal{L}$ from instances $x \notin \mathcal{L}$ with noticeable advantage. For any $x \in \{0,1\}^\lambda$, let $b \in \{0,1\}$ indicate whether $x \in \mathcal{L}_\lambda$, and note that $s_b = M_{s_0,s_1}^{\mathcal{L}}(x) = M_{s_b}^t(0^\lambda)$. Therefore, by the security of the succinct randomized encoding scheme, there exists a PPT simulator Sim and a negligible function $\mu(\cdot)$ such that

$$\begin{aligned}
&\Pr\left[\mathcal{B}'_\lambda(x) = 1\right] = \\
&\Pr\left[\mathcal{A}_\lambda(\widehat{M}_{s_0,s_1}^{\mathcal{L}}) = 1 \;\middle|\; \widehat{M}_{s_0,s_1}^{\mathcal{L}} \leftarrow \mathsf{SRE.Encode}(M_{s_0,s_1}^{\mathcal{L}}, x, t\lambda), 1^\lambda)\right] = \\
&\Pr\left[\mathcal{A}_\lambda(\widehat{S}_{s_b}) = 1 \;\middle|\; \widehat{S}_{s_b} \leftarrow \mathsf{Sim}(s_b, 1^{3\lambda}, 1^\lambda, t(\lambda), 1^\lambda)\right] \pm \mu(\lambda)
\end{aligned}$$

Moreover,

$$\begin{aligned}
&\Pr\left[\mathcal{A}_\lambda(Z_b) = 1 \mid Z_b \leftarrow \mathsf{Puzzle.Gen}(t(\lambda), s_b)\right] = \\
&\Pr\left[\mathcal{A}_\lambda(\widehat{M}_{s_b}^t) = 1 \;\middle|\; \widehat{M}_{s_b}^t \leftarrow \mathsf{SRE.Encode}(M_{s_b}^t, 0^\lambda, t(\lambda), 1^\lambda)\right] = \\
&\Pr\left[\mathcal{A}_\lambda(\widehat{S}_{s_b}) = 1 \;\middle|\; \widehat{S}_{s_b} \leftarrow \mathsf{Sim}(s_b, 1^{3\lambda}, 1^\lambda, t(\lambda), 1^\lambda)\right] \pm \mu(\lambda)
\end{aligned}$$

It follows by our assumption towards contradiction (Eq. (1)) that for infinitely many large enough λ and any $x, \bar{x} \in \{0,1\}^\lambda$ where $x \in \mathcal{L}_\lambda$ and $\bar{x} \notin \mathcal{L}_\lambda$, it holds that

$$\Pr\left[\mathcal{B}'_\lambda(x) = 1\right] - \Pr\left[\mathcal{B}'_\lambda(\bar{x}) = 1\right] \geq \frac{2}{q(\lambda)} - 4\mu(\lambda) \geq \frac{1}{q(\lambda)} \ , \tag{3}$$

To obtain a deterministic decider $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda\in\mathbb{N}}$ for $\mathcal{L}$ from $\mathcal{B}'$, we rely on standard amplification showing that BPP/*poly* $\subseteq$ **P**/*poly*, ([Gol01], theorem 1.3.7). To that end we first construct a decider $\mathcal{B}'' = \{\mathcal{B}''_\lambda\}_{\lambda\in\mathbb{N}}$. Consider the threshold

$$\alpha := \frac{\min_{x\in\mathcal{L}_\lambda} \Pr\left[\mathcal{B}'_\lambda(x) = 1\right] + \max_{\bar{x}\notin\mathcal{L}_\lambda} \Pr\left[\mathcal{B}'_\lambda(\bar{x}) = 1\right]}{2} \ ,$$

The decider $\mathcal{B}''$ gets α as a non uniform advice, and runs $\mathcal{B}'$ for $C \cdot q^2(\lambda) \cdot \lambda$ times, for large enough constant C ($C = 10$ suffices). It then outputs 1 if $\mathcal{B}'$ outputs 1 at least α-fraction of the times, and otherwise outputs 0. Then, by standard Chernoff argument,

$$\Pr\left[\mathcal{B}''_\lambda(x) \neq \mathcal{L}(x)\right] < 2^{-\lambda} \ .$$

The final decider $\mathcal{B}$ is constructed by non-uniformly fixing the random coins of $\mathcal{B}''$.

Size. By the efficiency of the SRE scheme, the encoding takes time

$$\text{poly}(\log(t(\lambda)) \cdot \text{poly}(|M|, |x|, \lambda) < \text{poly}(\lambda) \cdot \text{poly}(|M|, |x|, \lambda) = p'_{\text{SRE}}(\lambda) = \text{poly}(\lambda) \ ,$$

since $t = O(Q)$, and Q is polynomial. So the size of $\mathcal{B}'_\lambda$ is at most $O(q(\lambda) + p'_{\text{SRE}}(\lambda)) = q_{\mathcal{B}'}(\lambda)$ for some polynomial $q_{\mathcal{B}'}(\cdot)$. Since we repeat $\mathcal{B}'_\lambda$ for $O(q^2(\lambda) \cdot \lambda)$ times, the final decider $\mathcal{B}_\lambda$ is of size $q_{\mathcal{B}}(\lambda) = O(q^2(\lambda) \cdot \lambda \cdot q_{\mathcal{B}'}(\lambda)) = \text{poly}(\lambda)$. Note the polynomial bound $q_{\mathcal{B}}(\cdot)$ does not depend on the polynomial $Q(\cdot)$.

Finally, we achieve a contradiction to Assumption 2.1, with respect to the polynomial $q_{\mathcal{B}}(\cdot)$. That is we get a size-$q_{\mathcal{B}}$ circuit family that for any polynomial Q, decides any language $\mathcal{L} \in \mathbf{DTIME}(Q)$ for infinitely many input length λ.

□

Circuit EVK_j^k, for $j \geq 1$

Hardwired values:

- Seed S_j.

Input: An index $i \in [2^j + 1, 2^{j+1} - 1]$ (described as a j bits string).

1. If $i \geq k$:
   - Generate pseudorandom coins by
   $$(r_g^i, r_e^i) = \mathsf{PPRF.Eval}(S_j, i) \ .$$
   - Generate i-th encryption key using pseudorandom coins r_g^i by
   $$(\mathsf{dk}_i, \mathsf{ek}_i) = \mathsf{CHE.Gen}(1^\lambda; r_g^i) \ .$$
   - Encrypt 0^ℓ under ek_i using pseudorandom coins $\mathsf{PPRF.Eval}(S_j, i)$ by
   $$ct_i = \mathsf{CHE.Enc}_{\mathsf{ek}_i}(0^d; r_e^i) \ .$$
   - Output ct_i.
2. Otherwise:
   - Generate pseudorandom coins by
   $$(r_g^{i-1}, r_e^{i-1}) = \mathsf{PPRF.Eval}(S_j, i-1) \ ,$$
   $$(r_g^i, r_e^i) = \mathsf{PPRF.Eval}(S_j, i) \ .$$
   - Generate $i-1$-th decryption key using pseudorandom coins r_g^{i-1} by
   $$(\mathsf{dk}_{i-1}, \mathsf{ek}_{i-1}) = \mathsf{CHE.Gen}(1^\lambda; r_g^{i-1}) \ .$$
   - Generate i-th encryption key using pseudorandom coins r_g^i by
   $$(\mathsf{dk}_i, \mathsf{ek}_i) = \mathsf{CHE.Gen}(1^\lambda; r_g^i) \ .$$
   - Encrypt dk_{i-1} under ek_i using pseudorandom coins r_e^i by
   $$ct_i = \mathsf{CHE.Enc}_{\mathsf{ek}_i}(\mathsf{dk}_{i-1}; r_e^i) \ .$$
   - Output ct_i.

Figure 3: Circuit for generating the j-interval of the key chain, punctured on inputs $i \geq k$.