Leakage-Resilient Circuits against NC^1, Revisited

Yuyu Wang*

University of Electronic Science and Technology of China, Chengdu, China
wangyuyu@uestc.edu.cn

Abstract

In this study, we revisit leakage-resilient circuits (LRCs) against NC^1-leakage and propose new constructions that minimize the reliance on leak-free hardware. Specifically, we first present a stateless LRC scheme that is resilient to NC^1-leakage, and then extend it to a leakage-tolerant circuit with auxiliary input (AI-LTC). By integrating this with a 2-adaptive leakage-resilient encoding scheme, we achieve a stateful LRC scheme that uses a secure hardware component. In comparison to the state-of-the-art constructions against NC^1-leakage by Miles and Viola (STOC 2013), both the encoder during the leak-free phase in our stateless LRC and the secure hardware component in our stateful LRC are typically much smaller, as their sizes are independent of the original circuit size. Additionally, we provide a non-black-box instantiation of stateful LRC, resulting in a smaller compiled circuit. The security of all our constructions is based on the very mild worst-case assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$, which is strictly weaker than the assumption $\mathsf{NC}^1 \subsetneq \mathsf{L}$ used by Miles and Viola.

Furthermore, we propose a generic conversion from AI-LTCs to non-interactive zero-knowledge proofs with offline simulation (oNIZK) for all NP in the fine-grained setting. Our instantiation derived from it has small common reference strings, perfect soundness, zero-knowledge against adversaries in NC^1 under $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$, and minimal verification complexity.

Finally, we show that any fine-grained oNIZK cannot simultaneously achieve perfect soundness and verifiable common reference strings, thereby ruling out the possibility of constructing stateful LRCs without secure hardware by eliminating the trusted setup of our AI-LTC.

Keywords: Leakage-resilient circuit · NC^1 · Complexity assumption · Fine-grained cryptography · Zero-knowledge proof.

1 Introduction

1.1 Background

In the realm of cryptography, one of the primary objectives is to secure computations against adversaries who may have "non-black-box" access to cryptographic algorithms. Indeed, side-channel attacks, which exploit the physical characteristics of computational devices to gain partial information about the internal operations of cryptographic algorithms, have emerged as a significant threat. To address these vulnerabilities, a substantial body of research has focused on developing leakage-resilient cryptography along two main lines. One line concentrates on constructing specific leakage-resilient cryptographic primitives, such as encryption schemes and signatures (e.g., [1, 7]). The other, which is the focus of this work, aims to develop *leakage-resilient circuits (LRCs)*. These circuits compile general algorithms, represented as circuits, into new ones that maintain security even when partial information is leaked during execution, while still preserving the functionality of the original circuits.

The pioneering work by Ishai, Sahai, and Wagner [14] introduced two variants of LRC: the stateless LRC and the stateful LRC. *Stateless* LRC is designed for one-shot computations. In this model, it is assumed that a pair of leak-free encoder and decoder are available. The computation algorithm, derived from the original circuit, takes the encoded inputs (generated by the encoder) and maps them to encoded outputs, which can later be decoded. During execution, all wire assignments in the computation

*Supported by the National Natural Science Foundation of China under Grant Number 62472073.

are exposed to some limited leakage. Stateless LRCs are applicable in scenarios such as computations performed by payment terminals and access control readers, hardware implementation of zero-knowledge proofs, and zero-knowledge probabilistically checkable proofs [13, 16, 6]. *Stateful* LRC is a more advanced variant that applies to systems maintaining a persistent secret state. A stateful LRC is initialized with an encoded secret state. In each subsequent invocation, it computes an external output based on the encoded state and an external input provided by the user, while simultaneously refreshing the encoded state. This process does not require additional leak-free encoders or decoders. The security of stateful LRCs ensures that the secret state remains protected, even in the presence of continuous leakage during each invocation. Stateful LRCs have a wide range of applications. For instance, they can be used to protect the secret key stored in the internal memory of a symmetric key primitive (e.g., AES), against potential side-channel attacks, especially when the key is used repeatedly [14].

Existing research on stateless and stateful LRCs primarily considers wire-probing leakage, only-computation leakage, and computationally bounded leakage. The wire-probing leakage model [14] allows the adversary to observe only specific wires in the circuit, capturing certain types of noisy leakage [9]. In the only-computation leakage model [17], each LRC is partitioned into topologically ordered parts, with the leakage function operating separately on each part. The computationally bounded leakage model [11], which is the *main focus* of this work, permits computationally bounded leakage functions that apply jointly to all wires in the circuit. Both the only-computation leakage and computationally bounded leakage fall under the category of *global leakage*. They are considered richer than wire-probing leakage but are incomparable to each other.

Typically, works in the computationally bounded leakage model address leakage functions computable in AC^0 (i.e., constant-depth polynomial-sized circuits consisting of AND, OR, and NOT gates of unbounded fan-in) and related constant-depth complexity classes (e.g., [11, 21, 6, 5, 15]). Techniques treating AC^0-leakage generally exploit the fact that the parity of strings cannot be computed by AC^0 circuits, allowing the wires to be encoded via an XOR-based secret sharing. However, when information is leaked through computations that take non-constant time (e.g., logarithmic time), systems relying on these constructions become vulnerable.

To address this issue, the elegant work by Miles and Viola [19] proposed candidates of LRCs against leakage functions in NC^1 (i.e., logarithmic-depth polynomial-sized circuits consisting of AND, OR, and NOT gates of fan-in 2), which is quantitatively richer and more complex than AC^0. They introduced a stateless construction and demonstrated how it could be converted into a stateful one using leak-free hardware components. Their constructions were later proved secure under the complexity assumption $\mathsf{NC}^1 \subsetneq \mathsf{L}$ by Miles [18]. Up until now, these remain the only known candidates of LRCs that address NC^1-leakage. However, their constructions come with certain limitations. Specifically, in their stateless construction, the size of the encoder used in the leak-free phase is linear in the size of the original circuit, meaning that the user must work at least as hard as computing the original circuit, rather than simply feeding the LRC with its input [6]. Additionally, their stateful construction requires a large number of leak-free components, which grows with the size of the original circuit.

In light of the above, we pose the following question:

Is it possible to reduce the requirement for leak-free hardware in LRCs against NC^1*-leakage?*

1.2 Our Contributions

In this work, we revisit LRCs against NC^1-leakage and propose new constructions using leak-free hardware with a compact total size, as detailed below.

First, we construct a stateless LRC scheme for any polynomial-sized function, without relying on additional secure hardware components other than the encoder and decoder. Next, we show how to convert our stateless LRC into a new variant of leakage-tolerant circuit (LTC) [15] that we call LTC with auxiliary input (AI-LTC). Recall that, as originally defined in [15], each LTC admits a distribution over simulators that can effectively fool the leakage using only the input/output pair, without relying on leak-free decoding algorithms. Compared to this original definition, our definition of AI-LTC differs from LTC in two main respects aside from the leakage class considered: first, it requires a trusted setup; second, each simulator must be entirely in the leakage class.

By combining an AI-LTC with our proposed 2-adaptive NC^1-leakage-resilient encoding scheme in a "black-box manner", we obtain a stateful LRC scheme against NC^1-leakage that incorporates a secure

hardware component, represented as an opaque gate. Notably, we adopt the definition of stateful LRC from [6], which slightly differs from that in [15], and our requirement that the simulator of the AI-LTC lies in the leakage class mentioned above is essential for achieving a construction satisfying this definition.

Both the size of the (leak-free) encoder in our stateless LRC and the total size of the opaque gate in our stateful LRC are *independent* of the size of the original circuit. Instead, these sizes are determined only by the length of its input, output, or state. In contrast, the state-of-the-art constructions by [19] require these sizes to grow linearly with the size of the original circuit. Similar to previous works such as [11, 19], the opaque gate of our stateful construction, which is invoked only once per invocation, has a fixed standardized functionality applicable to all circuits, and carries neither external inputs nor internal state (i.e., it is independent of the computation). Moreover, if the number of invocations is known in advance, the opaque gate can be replaced by a precomputed list generated with the initial encoded state.

Additionally, we present an alternative instantiation of the stateful LRC, compiled directly from our AI-LTC in a "non-black-box" way. While the size of its opaque gate is slightly larger, the size of the compiled circuit becomes smaller.

Our assumption. The security of all our constructions is proven under the mild worst-case assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$, which is *weaker* than the assumption $\mathsf{NC}^1 \subsetneq \mathsf{L}$ in [19]. Specifically, since $\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NL}/\mathsf{poly}$, and $\mathsf{NL}/\mathsf{poly} \subseteq \oplus\mathsf{L}/\mathsf{poly}$ [12], we have $\mathsf{L} \subseteq \oplus\mathsf{L}/\mathsf{poly}$ (the reverse inclusion is widely believed not to hold). Namely, our assumption is strictly implied by theirs. Moreover, according to Barrington [3], $\oplus\mathsf{L}/\mathsf{poly}$ is the class of languages with polynomial-sized branching programs and NC^1 languages have polynomial-sized branching programs of constant width. Namely, this assumption holds if there exists one language having only branching programs of super-constant width. It would be quite surprising if this does not hold.

Comparison. In Tables 1 and 2, we compare the sizes of the encoder and decoder (for the stateless LRCs), the opaque gate and setup algorithm (for the stateful LRCs), and the compiled circuit between our constructions and those proposed in [19]. In these tables, λ denotes the statistical security parameter, l_{in}, l_{out}, and l_{st} denote the lengths of the (external) input, the (external) output, and the state of the original circuit respectively, and $|\mathsf{F}_\lambda|$ denotes the size of the original circuit measured by the number of gates.

Scheme	Encoder Size	Decoder Size	Circuit Size	Assumption
MV13 [19]	$O(\lvert\mathsf{F}_\lambda\rvert \cdot \lambda^3)$	$O(l_{\mathsf{out}} \cdot \lambda^2)$	$O(\lvert\mathsf{F}_\lambda\rvert \cdot \lambda^2)$	$\mathsf{NC}^1 \subsetneq \mathsf{L}$
Ours (Section 3)	$O(l_{\mathsf{in}} \cdot \lambda^2)$	$O(l_{\mathsf{out}} \cdot \lambda + \lambda^2)$	$O(\lvert\mathsf{F}_\lambda\rvert \cdot \lambda^3)$	$\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$

Table 1: Comparison of existing stateless LRCs against NC^1-leakage.

Scheme	Opaque Gate Size	Setup Size	Circuit Size	Assumption
MV13 [19]	$O(\lvert\mathsf{F}_\lambda\rvert \cdot \lambda^3)$	$O(l_{\mathsf{st}} \cdot \lambda^2)$	$O(\lvert\mathsf{F}_\lambda\rvert \cdot \lambda^3)$	$\mathsf{NC}^1 \subsetneq \mathsf{L}$
Ours (Section 5.1)	$O(l_{\mathsf{st}} \cdot \lambda^2 + l_{\mathsf{out}} \cdot \lambda)$	$O(l_{\mathsf{st}} \cdot \lambda^2)$	$O(l_{\mathsf{st}} \cdot \lambda^6 + \lvert\mathsf{F}_\lambda\rvert \cdot \lambda^3)$	$\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$
Ours (Section 5.2)	$O(l_{\mathsf{st}} \cdot \lambda^3 + l_{\mathsf{out}} \cdot \lambda)$	$O(l_{\mathsf{st}} \cdot \lambda^3)$	$O(l_{\mathsf{st}} \cdot \lambda^4 + \lvert\mathsf{F}_\lambda\rvert \cdot \lambda^3)$	$\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$

Table 2: Comparison of existing stateful LRCs against NC^1-leakage.

Practical relevance of NC^1-leakage. As in [19], the leakage class we consider consists of all NC^1-circuits with logarithmic output length. Note that even a single bit of NC^1-leakage can compromise existing AC^0-LRCs (e.g., [11, 21, 6]), since NC^1 encompasses natural leakage functions such as Parity and $\mathsf{Majority}$ that go beyond AC^0. With logarithmically many leaked bits, an adversary may additionally learn the indices of wires carrying secret values or identify secret users. Also, beyond capturing all attacks modeled by AC^0-leakage (e.g., selective failure attacks), NC^1-leakage also captures side-channel attacks in the Hamming distance framework, where the adversary observes the Hamming distance between secret values via leakage. Moreover, NC^1 corresponds to the class of languages recognized by polynomial-size alternating Turing machines operating in logarithmic space with bounded alternations [22], and thus captures polynomial-time leakage attacks with constrained memory.

By-product: fine-grained proof. As a by-product, we provide a generic transformation from AI-LTCs and leakage-resilient encodings to *non-interactive zero-knowledge proof systems with offline simulation (oNIZK) in the fine-grained setting*, as defined by Ball et al. [2]. We note that zero-knowledge with offline simulation requires the existence of a distribution of computationally bounded simulators. It is sufficient for indistinguishability-based applications and implies both *witness hiding* and *weak zero-knowledge*, as shown in [2].

Recall that in the fine-grained setting, adversaries are required to have bounded resources, and honest users are limited to no more resources than the adversaries. A key motivation of fine-grained cryptography is to base the security on very mild assumptions, particularly those not implied by one-way functions. To support all NP statements, the definition in [2] allows the prover to run in polynomial time. Otherwise, it cannot even decide the truth of the statement along with the witness.

By instantiating the underlying AI-LTC and encoding using our constructions against NC^1, we immediately obtain an NC^1-fine-grained oNIZK, where common reference string (CRS) generator, verifier, and simulator all run within NC^1. Notably, our verifier executes a circuit with minimal complexity. Specifically, it checks input consistency, gate consistency, and that the final output bit is 1 in parallel, using only an AC^0 circuit with depth 3. Our construction is perfectly sound, and its zero-knowledge (with offline simulation) holds against any adversary in NC^1 under the assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$.

We note that previous NC^1-fine-grained (o)NIZKs for all NP in [2, 25] can also be converted into ones with verifier having minimal complexity, by setting the assignment to all wires during verification as the proof itself. However, the construction in [2] suffers from a long uniform random string (URS) that grows linearly with the size of the statement's circuit, and the one in [25] increases the proof size by a constant factor compared to ours. More importantly, our construction offers a new generic approach for constructing fine-grained oNIZK and potentially derives oNIZKs in other fine-grained settings by instantiating the underlying AI-LTC against different leakage classes.

Negative Results. The primary difference between our AI-LTC and the original definition in [15] is that we require a trusted setup to output auxiliary inputs for the compiled circuit. If these auxiliary inputs could be sampled using only public coins, we would eliminate the need for leak-free hardware components in our stateful LRC. Unfortunately, we present a negative result showing that this approach may *not* be feasible. Specifically, we show that any fine-grained oNIZK with perfect soundness for a language admitting a fine-grained subset membership problem cannot have a verifiable CRS (e.g., a URS). Combined with our AI-LTC-to-oNIZK transformation, this rules out public-coin setup for the corresponding relation-checking function family.

Note that our negative result on AI-LTC does not conflict with the positive results on LTCs in [15], since our definition requires the simulator to be executable within the leakage class, whereas the original definition in [15] allows for inefficient simulators that take leakage on the input/output pair. Indeed, they have shown that the inefficient simulator of their LTC against parity leakage is inherent. Thus, one potential way to circumvent this impossibility result is to modify our AI-LTC to one without efficient simulation. However, this is challenging, as it requires switching the distribution of the matrix in the public parameters used throughout the entire circuit in the hybrids of our proof. Moreover, even if such a modification is possible, we may need to adopt the relaxed definition of stateful LRC adopted in [15], where the real leakage in each invocation is simulated by a possibly inefficient simulator, given the output of a function in the leakage class. Another potential approach is to make the auxiliary input size independent of the output length, enabling the use of AI-LTC to generate the auxiliary input for the next epoch and eliminate the opaque gate in the stateful LRC. However, this is not necessarily simpler, as the auxiliary input mainly serves to refresh the outputs. We leave the exploration of how to bypass this negative result and achieve stateful LRC against NC^1-leakage without secure hardware as an open problem. [1]

In Figure 1, we provide an overview of all our contributions.

1.3 Technical Overview

Stateless LRC. We begin by constructing a stateless LRC. The leakage function we consider consists of all NC^1 circuits with logarithmic output length. Our approach is based on the indistinguishability

[1] This open problem was subsequently resolved in [24], which constructs stateful LRCs against NC^1-leakage without leak-free hardware.

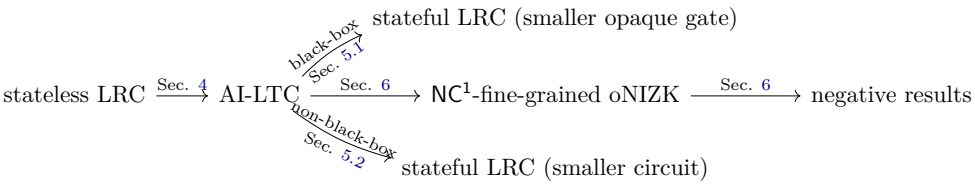


Figure 1: Overview of our contributions. All LRCs are secure against NC^1-leakage.

of two specific distributions $\mathcal{ZS}_\lambda$ and $\mathcal{OS}_\lambda$, which output $\lambda \times \lambda$ matrices with rank $\lambda - 1$ and full rank respectively. This indistinguishability is implied by $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$.

Our strawman strategy is to implement a stateless LRC for any polynomial-sized circuit F_λ using the fully homomorphic encryption (FHE) scheme proposed in [25] in the fine-grained setting. This scheme supports homomorphic evaluation of all circuits with constant multiplicative depth and logarithmic depth, and is secure against all NC^1 circuits under the assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$. In the encoding phase, we set a matrix $\mathbf{M}$, whose transpose is sampled from $\mathcal{ZS}_\lambda$, as the public key of the FHE scheme. For each bit x_i of an input $\mathbf{x}$, we compute its FHE ciphertext $\mathbf{C}_i = \mathbf{M}\mathbf{R}_i + x_i \otimes \mathbf{I}_\lambda$, where $\mathbf{R}_i$ is randomly sampled from $\{0,1\}^{\lambda\times\lambda}$ and $\mathbf{I}_\lambda$ is the identity matrix. In the computation phase, we replace each addition (respectively, multiplication) gate of F_λ by a "gate gadget" that performs additively (respectively, multiplicatively) homomorphic evaluation on its input wire bundles assigned with FHE ciphertexts. We refer to the wires carrying ciphertexts of wire values in F_λ as the "external wires". In the decoding phase, we compute the secret key of $\mathbf{M}$, which is the non-zero vector in the kernel of $\mathbf{M}^\top$, to decrypt the final output ciphertexts.

Correctness follows directly from that of the underlying FHE scheme. However, the resilience to leakage on external wires does not follow from the security of the FHE scheme. This is because F_λ could be any polynomial-size function, whereas the security of the FHE scheme holds only for homomorphic evaluation of a restricted class of circuits, as aforementioned, and breaks down when the evaluation exceeds NC^1.

To address this issue, we adopt the framework in [11] to introduce internal wires for each gate gadget, showing that: (i) the external wire distributions can be simulated in NC^1 given the wire assignments of F_λ on input $\mathbf{x}$, and (ii) each gate gadget is "reconstructible", i.e., its internal wire distributions can be simulated in NC^1 given the external wires and the internal randomness used to generate them. In essence, the internal wires refresh the output ciphertext of each gate gadget by adding a ciphertext of 0, and we exploit both the additive homomorphism of the FHE and the linearity of its internal randomness to construct simulators for proving these two properties, while maintaining correctness. Unlike in [11, 19], our simulator can also simulate the internal random coins used to generate ciphertexts of 0, and therefore does not have to generate the ciphertexts during the leak-free phase, ensuring that the size of our encoder is compact and independent of the original circuit. For details on the simulators, we refer the reader to Section 3.

With the simulators in place, the wire values of gate gadgets can be simulated in parallel in NC^1, allowing us to switch the distribution of $\mathbf{M}$ from $\mathcal{ZS}_\lambda$ to $\mathcal{OS}_\lambda$ without detection. Further, when $\mathbf{M}$ becomes full rank, the ciphertexts generated in the encoding phase become uniformly distributed, ensuring that no information about $\mathbf{x}$ is leaked. Here, a subtle issue arises: the indistinguishability between $\mathcal{ZS}_\lambda$ and $\mathcal{OS}_\lambda$ only guarantees resilience to 1-bit NC^1-leakage. To overcome this, we show that the statistical distance between the NC^1-leakage on a logarithmic number of bits of matrices from $\mathcal{ZS}_\lambda$ and $\mathcal{OS}_\lambda$ is also negligibly small, thereby achieving stronger resilience to multiple bits of leakage.

Extension to AI-LTC. We now provide an overview of how we compile our stateless LRCs into AI-LTCs. We require that the original circuit F_λ be deterministic. Namely, for any randomized function F_λ, its internal randomness is treated as part of the input and is required for simulation.

As mentioned earlier, the security of our stateless LRC is proven by exploiting the indistinguishability of $\mathcal{ZS}_\lambda$ and $\mathcal{OS}_\lambda$ (against NC^1 circuits) to switch $\mathbf{M}$ to a full-rank matrix. Unfortunately, this approach becomes infeasible in the context of AI-LTC, since the decoding phase is no longer leak-free. When the secret key of the underlying FHE, i.e., the non-zero vector in the kernel of $\mathbf{M}^\top$, is exposed, the indistinguishability is compromised. To overcome this issue, we propose a new decoding algorithm for the FHE ciphertexts of the output gate gadgets based on a fast Gaussian elimination process. Thanks to the special structure of $\mathbf{M}$, we show that the internal wires of the decoding algorithm can be simulated by an

NC^1-simulator, given access to $\mathbf{M}$, the plaintexts, and the internal randomness of the ciphertexts. This enables us to switch $\mathbf{M}$ to a full-rank matrix in the security proof. Here, we note that the simulator works only when the internal randomness of the ciphertexts is kept hidden in the real world. To ensure this, we refresh the ciphertexts using the auxiliary input, whose size is determined only by the output length of the original circuit, generated in the setup. We refer the reader to Section 4 for details on the decoding algorithm and the security analysis, which forms part of our main technical contribution in this section.

After making $\mathbf{M}$ full rank, the input ciphertexts become uniformly random and independent of the input $\mathbf{x}$. As a result, the entire computation phase becomes independent of $\mathbf{x}$, except for the decoding phase, which can be simulated using $\mathsf{F}_\lambda(\mathbf{x})$ and the internal randomness used in the encryption. Furthermore, we can simulate the internal wires for generating the input ciphertexts using $\mathbf{x}$ and the inverse of $\mathbf{M}$. This allows us to achieve a simulator that successfully fools the leakage, using $\mathbf{x}$ and $\mathsf{F}_\lambda(\mathbf{x})$. Note that when F_λ is randomized, its internal randomness is necessary for simulation as aforementioned, since we are not aware of any way to generate an FHE ciphertext of a random bit with only public coins.

Conversion from AI-LTC to Stateful LRC. To construct stateful LRCs from AI-LTCs, we adopt the generic framework outlined in [15, 11], utilizing encodings. We begin by constructing an encoding scheme based on the fine-grained encryption against NC^1-adversaries as presented in [8], and prove its security against 2-adaptive NC^1-leakage. Using this encoding scheme, we encode the initial state of the original circuit during the setup phase of the stateful LRC. In each invocation, the AI-LTC invokes its own setup algorithm, treated as an opaque gate with the size being independent of the original circuit size here, to generate the auxiliary input. Then it recovers the state by decoding, executes the computation phase, and encodes the new state for the subsequent invocation. The leakage-tolerance ensures that the adversary's view can be simulated based only on the external input/output pairs and the encoded states, and the 2-adaptive leakage-resilience of the encoding scheme guarantees that the leakage on each encoded state, which appears in at most two consecutive invocations, provides little advantage to the adversary.

It is important to note that internal randomness is essential for encoding in each invocation, whereas our AI-LTC supports only deterministic circuits. A direct solution would be to invoke the opaque gate to sample encodings of randomness as an additional input. However, this would cause the total size of the opaque gate to be squared. To address this, we show that the underlying encoding scheme possesses a desirable composability property, which allows us to extend the encoding to include an arbitrary-length random string in its message using only public coins. This enables the AI-LTC to obtain randomness for encoding the updated state "for free".

We further note that we adopt the definition of stateful LRC from [6], which slightly differs from [15]. In [6], it is required that the output of the leakage function on the simulated values be indistinguishable from real leakage in each invocation. Therefore, we must ensure that the simulator of the underlying AI-LTC operates within the leakage class as aforementioned so that we can switch encodings of states to dummy ones when proving security. In contrast, in [15], real leakage in each invocation is simulated by a possibly inefficient simulator, given the output of a function in the leakage class. Hence, their underlying AI-LTC simulator need not lie entirely in the leakage class.

Instantiation with smaller circuit size. Additionally, we propose an alternative and potentially more technical approach to convert our instantiation of AI-LTC into a stateful LRC in a non-black-box manner. It reduces the circuit size compared to the instantiation derived from the generic approach mentioned above, albeit at the cost of a larger opaque gate. To reduce the circuit size, instead of using the "decoding-encoding" operation, we directly encrypt the state during each invocation using the underlying FHE scheme employed in our AI-LTC. The ciphertext is then "embedded" into the circuit without requiring an additional layer of leakage-resilient encoding. For state updates, we utilize our Gaussian elimination-based decoding algorithm to re-encrypt the ciphertext via the bootstrapping technique of traditional FHE schemes. In this re-encryption process, only a portion of the FHE ciphertext of the current state is used, while the ciphertext of the new state is generated in its complete form. This allows us to leverage the technique from our previous instantiation to sample the internal randomness of the new ciphertext using only public coins, thus keeping both the resulting circuit and the opaque gate compact.

Transformation to fine-grained oNIZK. In our generic construction of fine-grained oNIZK for all NP based on AI-LTC, the setup algorithm of the AI-LTC acts as the CRS generator. The prover encodes the witness using a leakage-resilient encoding scheme, and runs the AI-LTC for a function that recovers the witness via decoding and verifies its validity for the statement. The proof consists of the assignment to

all wires in the AI-LTC. The verifier accepts the proof if and only if the assignments are consistent with the gate operations and the final output bit is 1, which can be checked in AC^0.

The correctness and perfect soundness of the oNIZK follow directly from the correctness of the underlying AI-LTC and encoding scheme. Here, note that an adversary may produce illegal encodings for breaking soundness, while this is not an issue, since verification only checks whether the witness exists, regardless of how it was generated. Zero-knowledge with offline simulation is guaranteed since any adversary in the leakage class must be fooled by the sampled AI-LTC simulator using the encoded witness and the statement. Then the encoding of the witness can be replaced by an encoding of 0's without being detectable by the adversary. By instantiating the AI-LTC with our construction, we obtain a fine-grained oNIZK for all NP statements, with the CRS generator, verifier, and simulators running in NC^1.

Limitations. To achieve the negative result, we show that for a fine-grained oNIZK with perfect soundness and verifiable CRSs, a simulated CRS/proof pair of any valid statement in the fine-grained membership problem, such as decisions on $\mathcal{ZS}_\lambda$ and $\mathcal{OS}_\lambda$, should pass the verification of both the proof and the CRS. However, replacing a valid statement with an invalid one, which should remain undetected by the simulator, breaks the perfect soundness of the system.

Note that while the high-level idea is somewhat similar to the negative result on subversion soundness in [4], there are subtle differences we must address. The key distinction is that we do not assume the adversary can generate the CRS. Instead, we only require that the CRS is verifiable, while assuming perfect soundness. This ensures that our negative result captures CRS generated with public coins, such as the URS, which is not covered in [4]. Also, we only require that the proof system satisfy a weak version of zero-knowledge, i.e., zero-knowledge with offline simulation, which strengthens the negative result. Additionally, we must ensure our result holds in the fine-grained setting, requiring a more careful treatment of the complexity in breaking the subset membership problem.

Combining this negative result with the conversion from AI-LTCs to oNIZKs, we immediately obtain our negative result on AI-LTCs.

2 Preliminaries

Notations. We note that all arithmetic computations are over $GF(2)$ in this work. Namely, all arithmetic computations are performed with a modulus of 2.

By $x \xleftarrow{\$} \mathcal{D}$ we denote the process of sampling x according to the distribution $\mathcal{D}$. If $\mathcal{D}$ is a set, then x is sampled uniformly at random from $\mathcal{D}$. By $y \xleftarrow{\$} \mathcal{A}(x)$ (respectively, $y = \mathcal{A}(x)$) we denote the process of running a probabilistic (respectively, deterministic) circuit $\mathcal{A}$ on input x and obtaining the output y. Let $\mathcal{R}$ be the randomness space of $\mathcal{A}$. Then $y \xleftarrow{\$} \mathcal{A}(x)$ is equivalent to $y = A(x; r)$ for $r \xleftarrow{\$} \mathcal{R}$. In this work, we model a public-coin algorithm by including its internal random coins in the output. By $\mathsf{W}[\mathcal{A}(x)]$ we denote the assignment to all wires (including the input and output wires) of $\mathcal{A}(x; r)$ for $r \xleftarrow{\$} \mathcal{R}$. By $(\mathcal{W}, y) \Leftarrow \mathcal{A}(x)$ we mean $y = \mathcal{A}(x; r)$ and $\mathcal{W} = \mathsf{W}[\mathcal{A}(x; r)]$ for $r \xleftarrow{\$} \mathcal{R}$. By $|\mathcal{A}|$ we denote the size, measured by the number of gates, of $\mathcal{A}$. The *statistical distance* of two random variables x, y over the common support $\mathcal{D}$ is defined as

$$\Delta(x, y) = \frac{1}{2} \sum_{v \in \mathcal{D}} |\Pr[x = v] - \Pr[y = v]| = \max_{\mathcal{S} \subseteq \mathcal{D}} |\Pr[x \in \mathcal{S}] - \Pr[y \in \mathcal{S}]|.$$

By negl we denote an unspecified negligible function.

By $\mathbf{x} \in \{0,1\}^n$ we denote a column vector with size n, and by x_i we denote the ith element of $\mathbf{x}$. For a matrix $\mathbf{A} \in \{0,1\}^{n \times t}$ with rank $t' \leq n$, we denote the sets $\{\mathbf{y} \mid \exists \mathbf{x} \text{ s.t. } \mathbf{y} = \mathbf{A}\mathbf{x}\}$ and $\{\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{0}\}$ by $\mathsf{Span}(\mathbf{A})$ (i.e., the span of $\mathbf{A}$) and $\mathsf{Ker}(\mathbf{A})$ (i.e., the kernel of $\mathbf{A}$) respectively. By $\mathbf{A}^\perp \in \{0,1\}^{n \times (n-t')}$ we denote a non-zero matrix of rank $n - t'$ in $\mathsf{Ker}(\mathbf{A}^\top)$. It follows that for any $\mathbf{y} \notin \mathsf{Span}(\mathbf{A})$, we have $\mathbf{y}^\top \mathbf{A}^\perp \neq \mathbf{0}$. By $\overline{\mathbf{A}}$ (respectively, $\underline{\mathbf{A}}$) we denote the upper $(n-1) \times t$ matrix (respectively, lower $1 \times t$ vector) of $\mathbf{A}$. By $\mathbf{I}_n$ we denote an identity matrix in $\{0,1\}^{n \times n}$. By $\mathbf{e}_n$ we denote the column vector in $\{0,1\}^n$ with the nth element being 1 and the other elements being 0. By $\mathbf{0}$ we denote a zero vector or matrix. By $[n]$ and $[n : m]$ we denote the sets $\{1, \cdots, n\}$ and $\{n, \cdots, m\}$ respectively.

For any matrices $\mathbf{A} = (a_{i,j})_{i \in [n_1], j \in [m_1]} \in \{0,1\}^{n_1 \times m_1}$ and $\mathbf{B} \in \{0,1\}^{n_2 \times m_2}$ and any bit $c \in \{0,1\}$, the Kronecker product $\otimes$ is defined as

$$c \otimes \mathbf{A} = (a_{i,j} \cdot c)_{i \in [n_1], j \in [m_1]} \text{ and } \mathbf{A} \otimes \mathbf{B} = (b_{i,j} \otimes \mathbf{A})_{i \in [n_2], j \in [m_2]} \in \{0,1\}^{n_1 n_2 \times m_1 m_2}.$$

2.1 Function Families

In this section, we recall the definitions of function family, NC^1 circuit, $\mathsf{AC}^0[2]$ circuit, and $\oplus\mathsf{L}/\mathsf{poly}$ circuit. Note that $\mathsf{AC}^0[2] \subsetneq \mathsf{NC}^1$ [20, 23].

Definition 2.1 (Function family). A *function family* is a family of (possibly randomized) functions $\mathcal{F} = \{\mathsf{F}_\lambda\}_{\lambda\in\mathbb{N}}$, where for each λ, F_λ has a domain $\mathcal{D}_\lambda$ and a range $\mathcal{R}_\lambda$. For brevity, we refer to it as $\{\mathsf{F}_\lambda\}$.

By $\{\widetilde{\mathcal{C}_\lambda}\}$ we denote the class of function families that is equivalent to $\{\mathcal{C}_\lambda\}$, but without any specific restriction on the output length. A class of function families $\{\mathcal{C}_\lambda\}$ is said to be *closed under composition* if for all $\lambda \in \mathbb{N}$ and all $\mathsf{F}_\lambda \in \mathcal{C}_\lambda, \mathsf{F}'_\lambda \in \widetilde{\mathcal{C}_\lambda}$, we have $\mathsf{F}_\lambda \circ \mathsf{F}'_\lambda \in \mathcal{C}_\lambda$. Note that NC^1 (with restricted output length), as defined below, is closed under composition.

Definition 2.2 (NC^1). The class of (non-uniform) NC^1 function families is the set of all function families $\mathcal{F} = \{\mathsf{F}_\lambda\}$ for which there is a polynomial $p(\cdot)$ and constant c such that for each λ, F_λ can be computed by a (randomized) circuit of size $p(\lambda)$, depth $c\log(\lambda)$, and fan-in 2 using AND, OR, and NOT gates.

Definition 2.3 ($\mathsf{AC}^0[2]$). The class of (non-uniform) $\mathsf{AC}^0[2]$ function families is the set of all function families $\mathcal{F} = \{\mathsf{F}_\lambda\}$ for which there is a polynomial $p(\cdot)$ and constant c such that for each λ, F_λ can be computed by a (randomized) circuit of size $p(\lambda)$, depth c, and unbounded fan-in using AND, OR, NOT, and PARITY gates.

Note that multiplication of a constant number of polynomial-sized matrices, which also includes addition of a polynomial number of matrices (or bits), can be performed in $\mathsf{AC}^0[2]$, since it can be executed in constant-depth with PARITY gates in parallel.

The definition of AC^0 is identical to that of $\mathsf{AC}^0[2]$, except that it does not use unbounded fan-in PARITY gates.

Definition 2.4 ($\oplus\mathsf{L}/\mathsf{poly}$). $\oplus\mathsf{L}/\mathsf{poly}$ is the set of all boolean function families $\mathcal{F} = \{\mathsf{F}_\lambda\}$ for which there is a constant c such that for each λ, there is a non-deterministic Turing machine $\mathcal{M}_\lambda$ such that for each input x with length λ, $\mathcal{M}_\lambda(x)$ uses at most $c\log(\lambda)$ space, and $\mathsf{F}_\lambda(x)$ is equal to the parity of the number of accepting paths of $\mathcal{M}_\lambda(x)$.

2.2 Sampling Procedure

We now recall the definitions of four sampling procedures $\{\mathcal{LS}_\lambda, \mathcal{RS}_\lambda, \mathcal{ZS}_\lambda, \mathcal{OS}_\lambda\}$ in Figure 2 [8]. Note that the output of $\mathcal{ZS}_\lambda$ (respectively, $\mathcal{OS}_\lambda$) is always a matrix of rank $\lambda - 1$ (respectively, of full rank) in $\{0,1\}^{\lambda\times\lambda}$.

$\underline{\mathcal{LS}_\lambda}$:
For $i, j \in [\lambda]$ s.t. $i < j$: $r_{i,j} \xleftarrow{\$} \{0,1\}$

$$\text{Return } \begin{pmatrix} 1 & r_{1,2} & \cdots & r_{1,\lambda-1} & r_{1,\lambda} \\ 0 & 1 & \cdots & r_{2,\lambda-1} & r_{2,\lambda} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & r_{\lambda-1,\lambda} \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$$

$\underline{\mathcal{RS}_\lambda}$:
For $i \in [\lambda - 1]$: $r_i \xleftarrow{\$} \{0,1\}$

$$\text{Return } \begin{pmatrix} 1 & 0 & \cdots & 0 & r_1 \\ 0 & 1 & \cdots & 0 & r_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & r_{\lambda-1} \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$$

$\underline{\mathcal{ZS}_\lambda}$:
$\mathbf{R}_0 \xleftarrow{\$} \mathcal{LS}_\lambda$
$\mathbf{R}_1 \xleftarrow{\$} \mathcal{RS}_\lambda$
Return $\mathbf{R}_0\mathbf{M}_0^\lambda\mathbf{R}_1$

$\underline{\mathcal{OS}_\lambda}$:
$\mathbf{R}_0 \xleftarrow{\$} \mathcal{LS}_\lambda$
$\mathbf{R}_1 \xleftarrow{\$} \mathcal{RS}_\lambda$
Return $\mathbf{R}_0\mathbf{M}_1^\lambda\mathbf{R}_1$

Figure 2: Definitions of $\mathcal{LS}_\lambda$, $\mathcal{RS}_\lambda$, $\mathcal{ZS}_\lambda$, and $\mathcal{OS}_\lambda$. By $\mathbf{M}_0^\lambda$ and $\mathbf{M}_1^\lambda$ we denote the $\lambda \times \lambda$ matrices $\mathbf{M}_0^\lambda = \begin{pmatrix} \mathbf{0} & 0 \\ \mathbf{I}_{\lambda-1} & \mathbf{0} \end{pmatrix}$ and $\mathbf{M}_1^\lambda = \begin{pmatrix} \mathbf{0} & 1 \\ \mathbf{I}_{\lambda-1} & \mathbf{0} \end{pmatrix}$ respectively.

Below we recall two lemmas on $\{\mathcal{ZS}_\lambda, \mathcal{OS}_\lambda\}$ given in [8].

Lemma 2.5 (Lemma 4.3 in [8]). *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$ *holds, then for any* $\{\ell_\lambda : \{0,1\}^{\lambda\times\lambda} \to \{0,1\}\} \in \mathsf{NC}^1$ *and all* $\lambda \in \mathbb{N}$*, we have* $\Delta\big(\ell_\lambda(\mathbf{M}_0), \ell_\lambda(\mathbf{M}_1)\big) \le \mathsf{negl}(\lambda)$*, where* $\mathbf{M}_0^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$ *and* $\mathbf{M}_1^\top \xleftarrow{\$} \mathcal{OS}_\lambda$.

Here, we follow previous works (e.g., [10, 25]) to adopt the stronger notion of $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$ mentioned in the second paragraph of Remark 3.1 in [8] for simplicity. If we only assume the infinitely-often version of $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$, holding only for an infinitely large number of values of λ, then our schemes given later satisfy *infinitely-often security*.

Lemma 2.6 (Lemma 3 in [10]). *For all $\lambda \in \mathbb{N}$ and all $\mathbf{M}^\top \in \mathcal{ZS}_\lambda$, it holds that $\mathsf{Ker}(\mathbf{M}^\top) = \{\mathbf{0}, \mathbf{M}^\perp\}$ where $\mathbf{M}^\perp \in \{0,1\}^{\lambda-1} \times \{1\}$.*

By Lemma 2.6, one can see that $\mathbf{e}_\lambda \notin \mathsf{Span}(\mathbf{M})$ for any $\mathbf{M}^\top \in \mathcal{ZS}_\lambda$, since ${\mathbf{M}^\perp}^\top \mathbf{e}_\lambda = 1$. Hence, $\mathbf{M}||\mathbf{e}_\lambda$ must be of rank λ.

Alternative representation of $\mathcal{ZS}_\lambda$. Let $\mathbf{r} = (r_{i,j})_{1 \le i < j \le \lambda} \in \{0,1\}^{\lambda(\lambda-1)/2}$. Following [25], we define a function family $\{\mathsf{F}_\lambda\}$ as

$$\mathsf{F}_\lambda(\mathbf{r}) = \begin{pmatrix} r_{1,2} & r_{1,3} & \cdots & r_{1,\lambda-1} & r_{1,\lambda} \\ 1 & r_{2,3} & \cdots & r_{2,\lambda-1} & r_{2,\lambda} \\ 0 & 1 & \cdots & r_{3,\lambda-1} & r_{3,\lambda} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & r_{\lambda-1,\lambda} \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

As shown in [25], the distribution of $\mathsf{F}_\lambda(\mathbf{r})||\mathsf{F}_\lambda(\mathbf{r})\widetilde{\mathbf{r}}$ where $\mathbf{r} \xleftarrow{\$} \{0,1\}^{\lambda(\lambda-1)/2}$ and $\widetilde{\mathbf{r}} \xleftarrow{\$} \{0,1\}^{\lambda-1}$ is exactly the same as the output distribution of $\mathcal{ZS}_\lambda$.

We further note that $\mathbf{X} = \mathbf{MR}$ for $\mathbf{M}^\top = (\mathsf{F}_\lambda(\mathbf{r})||\mathsf{F}_\lambda(\mathbf{r})\widetilde{\mathbf{r}})^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$ and $\mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ can be computed by sampling $\overline{\mathbf{X}} \xleftarrow{\$} \{0,1\}^{(\lambda-1)\times\lambda}$ and computing $\underline{\mathbf{X}} = \widetilde{\mathbf{r}}^\top \overline{\mathbf{X}}$, with the overhead (i.e., the number of gates) being $O(\lambda^2)$ rather than $O(\lambda^3)$, given the intermediate value $\widetilde{\mathbf{r}}$ used by $\mathcal{ZS}_\lambda$.

2.3 Leakage-Resilient Circuits

In this section, we recall the definitions of stateless LRC, AI-LTC, stateful function, and stateful LRC.

Definition 2.7 (Stateless leakage-resilient circuit). For a (stateless) function family $\{\mathsf{F}_\lambda : \{0,1\}^{l_{\mathsf{in}}} \to \{0,1\}^{l_{\mathsf{out}}}\}$, [2] a *stateless $\{\mathcal{L}_\lambda\}$-leakage-resilient circuit (LRC)* consists of polynomial-sized function families $\{\mathsf{SLEnc}_\lambda, \mathsf{SLComp}_\lambda, \mathsf{SLDec}_\lambda\}$ such that

- SLEnc_λ on input $\mathbf{x} \in \{0,1\}^{l_{\mathsf{in}}}$ returns an encoded input $\widehat{\mathbf{x}}$,
- SLComp_λ on input $\widehat{\mathbf{x}}$ returns an encoded output $\widehat{\mathbf{y}}$,
- SLDec_λ on input $\widehat{\mathbf{y}}$ returns an output $\mathbf{y} \in \{0,1\}^{l_{\mathsf{out}}}$.

Correctness is satisfied if for all $\lambda \in \mathbb{N}$ and all $\mathbf{x} \in \{0,1\}^{l_{\mathsf{in}}}$, we have

$$\mathsf{F}_\lambda(\mathbf{x}) = \mathsf{SLDec}_\lambda(\mathsf{SLComp}_\lambda(\mathsf{SLEnc}_\lambda(\mathbf{x}))).$$

Leakage-resilience is satisfied if for all $\lambda \in \mathbb{N}$, all $\mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^{l_{\mathsf{in}}}$, and all $\ell_\lambda \in \mathcal{L}_\lambda$, we have

$$\Delta\Big(\ell_\lambda(\mathsf{W}[\mathsf{SLComp}_\lambda(\mathsf{ct}_0)]), \ell_\lambda(\mathsf{W}[\mathsf{SLComp}_\lambda(\mathsf{ct}_1)])\Big) \le \mathsf{negl}(\lambda)$$

where $\mathsf{ct}_0 \xleftarrow{\$} \mathsf{SLEnc}_\lambda(\mathbf{x}_0)$ and $\mathsf{ct}_1 \xleftarrow{\$} \mathsf{SLEnc}_\lambda(\mathbf{x}_1)$.

Definition 2.8 (Leakage-tolerant circuit with auxiliary input). For a function family $\{\mathsf{F}_\lambda : \{0,1\}^{l_{\mathsf{in}}} \to \{0,1\}^{l_{\mathsf{out}}}\}$, an *$\{\mathcal{L}_\lambda\}$-leakage-tolerant circuit with auxiliary input (AI-LTC)* consists of polynomial-sized function families $\{\mathsf{LTSetup}_\lambda, \mathsf{LTComp}_\lambda\}$ such that

- $\mathsf{LTSetup}_\lambda$ outputs an auxiliary input aux,
- LTComp_λ on input an auxiliary input aux and an input $\mathbf{x} \in \{0,1\}^{l_{\mathsf{in}}}$ returns an output $\mathbf{y} \in \{0,1\}^{l_{\mathsf{out}}}$.

Correctness is satisfied if for all $\lambda \in \mathbb{N}$, all $\mathbf{x} \in \{0,1\}^{l_{\mathsf{in}}}$, and all $\mathsf{aux} \in \mathsf{LTSetup}_\lambda$, we have $\mathsf{LTComp}_\lambda(\mathsf{aux}, \mathbf{x}) = \mathsf{F}_\lambda(\mathbf{x})$.

Leakage-tolerance is satisfied if there exists a family of polynomial-time distributions $\{\mathcal{D}_\lambda\}$ over circuits in $\{\widetilde{\mathcal{L}}_\lambda\}$ (see Section 2.1 for the notation) such that for all $\lambda \in \mathbb{N}$, all $\mathbf{x} \in \{0,1\}^{l_{\mathsf{in}}}$, and all $\ell_\lambda \in \mathcal{L}_\lambda$, we have

$$\Delta\Big(\ell_\lambda(\mathsf{W}[\mathsf{LTComp}_\lambda(\mathsf{aux}, \mathbf{x})]), \ell_\lambda\big(\mathsf{LTSim}_\lambda(\mathbf{x}, \mathsf{F}_\lambda(\mathbf{x}))\big)\Big) \le \mathsf{negl}(\lambda)$$

where $\mathsf{aux} \xleftarrow{\$} \mathsf{LTSetup}_\lambda$ and $\mathsf{LTSim}_\lambda \xleftarrow{\$} \mathcal{D}_\lambda$.

[2] Here, l_{in} and l_{out} are polynomials in λ. Similar statements apply to other definitions in this section.

Definition 2.9 (Stateful function [14]). A *stateful function* F is represented by a circuit with the following extensions:

- F may contain *memory cells*, which are stateful gates with fan-in 1. On each invocation of the circuit, these memory cells output the previous input to the gate, while storing the current input for the next invocation.
- F may contain cycles, where each cycle traverses at least one memory cell.

Let $\mathbf{s}$ be the state for the memory cells. By $(\mathbf{s}', \mathbf{y}) = \mathsf{F}[\mathbf{s}](\mathbf{x})$ we mean that F, with memory cells filled with $\mathbf{s}$, takes the external input $\mathbf{x}$, returns $\mathbf{y}$ as the external output, and updates the memory cells to the new state $\mathbf{s}'$.

Definition 2.10 (Stateful leakage-resilient circuit [6]). For a stateful function family $\{\mathsf{F}_\lambda : \{0,1\}^{l_{\mathsf{st}}} \times \{0,1\}^{l_{\mathsf{in}}} \to \{0,1\}^{l_{\mathsf{st}}} \times \{0,1\}^{l_{\mathsf{out}}}\}$, a *stateful* $\{\mathcal{L}_\lambda\}$*-leakage-resilient circuit (LRC)* consists of polynomial-sized function families $\{\mathsf{SFSetup}_\lambda, \mathsf{SFComp}_\lambda\}$ such that

- $\mathsf{SFSetup}_\lambda$ on input an initial state $\mathbf{s}_0 \in \{0,1\}^{l_{\mathsf{st}}}$ of F_λ returns an encoded initial state st_0,
- SFComp_λ on input an encoded state st and an external input $\mathbf{x} \in \{0,1\}^{l_{\mathsf{in}}}$ outputs an updated encoded state st' and an external output $\mathbf{y} \in \{0,1\}^{l_{\mathsf{out}}}$.

Correctness is satisfied if for all $\lambda \in \mathbb{N}$, all polynomials $q = q(\lambda)$, all initial states $\mathbf{s}_0 \in \{0,1\}^{l_{\mathsf{st}}}$ of F_λ, and all $\mathsf{st}_0 \in \mathsf{SFSetup}_\lambda(\mathbf{s}_0)$, the same outputs must be obtained by q stateful invocations of $\mathsf{F}_\lambda[\mathbf{s}_0]$ and $\mathsf{SFComp}_\lambda[\mathsf{st}_0]$ on input any sequence $\mathbf{x}_1, \cdots, \mathbf{x}_q \in \{0,1\}^{l_{\mathsf{in}}}$.

Leakage-resilience is satisfied if there exists a polynomial-sized simulator $\{\mathsf{Sim}_\lambda\}$ such that for all $\lambda \in \mathbb{N}$, all polynomials $q = q(\lambda)$, all (possibly unbounded) adversaries $\mathcal{A}$, and all initial states $\mathbf{s}_0$, we have

$$|\Pr[\mathsf{Real}_{\mathcal{A}}] - \Pr[\mathsf{Ideal}_{\mathcal{A}}]| \le \mathsf{negl}(\lambda),$$

where the experiments $\mathsf{Real}_{\mathcal{A}}$ and $\mathsf{Ideal}_{\mathcal{A}}$ are defined as in *Figure* 3. Here Sim_λ may be stateful: it may maintain private state across the q invocations, initialized before the first invocation and updated whenever it is invoked in the ideal experiment.

$\underline{\mathsf{Real}_{\mathcal{A}}}$:	$\underline{\mathsf{Ideal}_{\mathcal{A}}}$:
$\mathsf{st}_0 \xleftarrow{\$} \mathsf{SFSetup}_\lambda(\mathbf{s}_0)$	$\mathbf{y}_0 = \mathbf{z}_0 = \bot$
$\mathbf{y}_0 = \mathbf{z}_0 = \bot$	For $i = 1, \cdots, q$
For $i = 1, \cdots, q$	• $(\mathbf{x}_i, \ell^i_\lambda \in \mathcal{L}_\lambda) \xleftarrow{\$} \mathcal{A}(\mathbf{y}_{i-1}, \mathbf{z}_{i-1})$
• $(\mathbf{x}_i, \ell^i_\lambda \in \mathcal{L}_\lambda) \xleftarrow{\$} \mathcal{A}(\mathbf{y}_{i-1}, \mathbf{z}_{i-1})$	• $(\mathbf{s}_i, \mathbf{y}_i) \Leftarrow \mathsf{F}_\lambda[\mathbf{s}_{i-1}](\mathbf{x}_i)$
• $(\mathcal{W}_i, (\mathsf{st}_i, \mathbf{y}_i)) \Leftarrow \mathsf{SFComp}_\lambda[\mathsf{st}_{i-1}](\mathbf{x}_i)$	• $\mathbf{z}_i = \ell^i_\lambda(\mathsf{Sim}_\lambda(\mathbf{x}_i, \mathbf{y}_i))$
• $\mathbf{z}_i = \ell^i_\lambda(\mathcal{W}_i)$	Output whatever $\mathcal{A}$ outputs
Output whatever $\mathcal{A}$ outputs	

Figure 3: Definitions of Real and Ideal.

3 Stateless Leakage-Resilient Circuit against NC^1

In this section, we propose a stateless LRC based on $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$.

Before presenting our construction, we extend Lemma 2.5 to show the indistinguishability of matrices sampled from $\mathcal{ZS}_\lambda$ and $\mathcal{OS}_\lambda$ in the presence of circuits in NC^1 with logarithmic output length.

Lemma 3.1 *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$*, then for any* $\{\ell_\lambda : \{0,1\}^{\lambda\times\lambda} \to \{0,1\}^{O(\log\lambda)}\} \in \mathsf{NC}^1$ *and all* $\lambda \in \mathbb{N}$*, we have* $\Delta\big(\ell_\lambda(\mathbf{M}_0), \ell_\lambda(\mathbf{M}_1)\big) \le \mathsf{negl}(\lambda)$*, where* $\mathbf{M}_0^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$ *and* $\mathbf{M}_1^\top \xleftarrow{\$} \mathcal{OS}_\lambda$*.*

Proof. Assume that there exists a constant c such that for sufficiently large λ, there exists a circuit ℓ_λ of depth $O(\log\lambda)$ and output length $c\log\lambda$ such that $\Delta\big(\ell_\lambda(\mathbf{M}_0), \ell_\lambda(\mathbf{M}_1)\big) \ge \lambda^{-k}$ for some constant k. Let $\mathcal{S} \subseteq \{0,1\}^{c\log\lambda}$ be the set that maximizes $|\Pr[\ell_\lambda(\mathbf{M}_0) \in \mathcal{S}] - \Pr[\ell_\lambda(\mathbf{M}_1) \in \mathcal{S}]|$. We construct a function family $\{\ell'_\lambda\} \in \mathsf{NC}^1$ such that for each λ, ℓ'_λ on input any matrix $\mathbf{M} \in \{0,1\}^{\lambda\times\lambda}$ outputs 1 iff $\ell_\lambda(\mathbf{M}) \in \mathcal{S}$. Since the size of $\mathcal{S}$ is polynomial in λ, checking $\mathbf{x} \in \mathcal{S}$ for any $\mathbf{x} \in \{0,1\}^{c\log(\lambda)}$ can be done in NC^1, i.e., we have $\{\ell'_\lambda\} \in \mathsf{NC}^1$. Moreover, we have

$$\begin{aligned}&|\Pr[\ell'_\lambda(\mathbf{M}_0) = 1] - \Pr[\ell'_\lambda(\mathbf{M}_1) = 1]| \\ &\qquad = \max_{\mathcal{S}\subseteq\{0,1\}^{c\log\lambda}} |\Pr[\ell_\lambda(\mathbf{M}_0) \in \mathcal{S}] - \Pr[\ell_\lambda(\mathbf{M}_1) \in \mathcal{S}]| \ge \lambda^{-k},\end{aligned}$$

which contradicts Lemma 2.5, completing the proof of Lemma 3.1. □

Reconstructible gate gadgets. In Figure 4, we define gate gadgets $\{\mathsf{Add}_\lambda, \mathsf{Mult}_\lambda\}$ and show that they are reconstructible using $\{\mathsf{SimAdd}_\lambda, \mathsf{SimMult}_\lambda\}$. One can see that all these function families are in $\mathsf{AC}^0[2]$, since they involve only sampling random bits and multiplications and additions of a constant number of matrices in parallel.

$\underline{\mathsf{Add}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2})}$:	$\underline{\mathsf{SimAdd}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}, \mathbf{Z}_{d_1}, \mathbf{Z}_{d_2}, \mathbf{Z}_{d_3})}$
Sample $\mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$	Compute $\mathbf{R} = \mathbf{Z}_{d_3} - (\mathbf{Z}_{d_1} + \mathbf{Z}_{d_2})$
Return $\mathbf{W}_{d_3} = \mathbf{W}_{d_1} + \mathbf{W}_{d_2} + \mathbf{MR}$	Return $\mathsf{W}[\mathsf{Add}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}; \mathbf{R})]$
$\underline{\mathsf{Mult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2})}$:	$\underline{\mathsf{SimMult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}, \mathbf{Z}_{d_1}, \mathbf{Z}_{d_2}, \mathbf{Z}_{d_3}, w_{d_1})}$:
Sample $\mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$	Compute $\mathbf{R} = \mathbf{Z}_{d_3} - \mathbf{Z}_{d_1}\mathbf{W}_{d_2} - w_{d_1} \otimes \mathbf{Z}_{d_2}$
Return $\mathbf{W}_{d_3} = \mathbf{W}_{d_1}\mathbf{W}_{d_2} + \mathbf{MR}$	Return $\mathsf{W}[\mathsf{Mult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}; \mathbf{R})]$

Figure 4: Definition of $\{\mathsf{Add}_\lambda, \mathsf{Mult}_\lambda, \mathsf{SimAdd}_\lambda, \mathsf{SimMult}_\lambda\}$. All matrices are in $\{0,1\}^{\lambda\times\lambda}$.

Lemma 3.2 *Let* $\mathbf{W}_{d_1} = \mathbf{MZ}_{d_1} + w_{d_1} \otimes \mathbf{I}_\lambda$ *and* $\mathbf{W}_{d_2} = \mathbf{MZ}_{d_2} + w_{d_2} \otimes \mathbf{I}_\lambda$ *for some* $\mathbf{M} \in \{0,1\}^{\lambda\times\lambda}$ *and some* $\mathbf{Z}_{d_1}, \mathbf{Z}_{d_2} \in \{0,1\}^{\lambda\times\lambda}$. *The distribution of*

$$\begin{aligned}&\big(\mathsf{W}[\mathsf{Add}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}; \mathbf{R})], \mathsf{Add}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}; \mathbf{R})\big)\\ &\Big(\textit{respectively, } \big(\mathsf{W}[\mathsf{Mult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}; \mathbf{R})], \mathsf{Mult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}; \mathbf{R})\big)\Big)\end{aligned}$$

is identical to that of

$$\begin{aligned}&\big(\mathsf{SimAdd}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}, \mathbf{Z}_{d_1}, \mathbf{Z}_{d_2}, \mathbf{Z}_{d_3}), \mathbf{MZ}_{d_3} + (w_{d_1} + w_{d_2}) \otimes \mathbf{I}_\lambda\big)\\ &\Big(\textit{respectively, } \big(\mathsf{SimMult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}, \mathbf{Z}_{d_1}, \mathbf{Z}_{d_2}, \mathbf{Z}_{d_3}, w_{d_1}), \mathbf{MZ}_{d_3} + (w_{d_1} \cdot w_{d_2}) \otimes \mathbf{I}_\lambda\big)\Big)\end{aligned}$$

where $\mathbf{R}, \mathbf{Z}_{d_3} \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$.

Proof. Since we have

$$\begin{aligned}\mathsf{Add}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}; \mathbf{R}) &= \mathbf{W}_{d_1} + \mathbf{W}_{d_2} + \mathbf{MR}\\ &= \mathbf{M}(\mathbf{Z}_{d_1} + \mathbf{Z}_{d_2} + \mathbf{R}) + (w_{d_1} + w_{d_2}) \otimes \mathbf{I}_\lambda,\\ \mathsf{Mult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}; \mathbf{R}) &= \mathbf{W}_{d_1}\mathbf{W}_{d_2} + \mathbf{MR} = (\mathbf{MZ}_{d_1} + w_{d_1} \otimes \mathbf{I}_\lambda)\mathbf{W}_{d_2} + \mathbf{MR}\\ &= \mathbf{MZ}_{\mathbf{d_1}}\mathbf{W}_{d_2} + w_{d_1} \otimes (\mathbf{MZ}_{d_2} + w_{d_2} \otimes \mathbf{I}_\lambda) + \mathbf{MR}\\ &= \mathbf{M}(\mathbf{Z}_{d_1}\mathbf{W}_{d_2} + w_{d_1} \otimes \mathbf{Z}_{d_2} + \mathbf{R}) + w_{d_1}w_{d_2} \otimes \mathbf{I}_\lambda,\end{aligned}$$

the distributions of

$$(\mathsf{Add}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}; \mathbf{R}), \mathbf{R}) \text{ and } (\mathbf{MZ}_{d_3} + (w_{d_1} + w_{d_2}) \otimes \mathbf{I}_\lambda, \mathbf{Z}_{d_3} - \mathbf{Z}_{d_1} - \mathbf{Z}_{d_2})$$

are identical, and similarly, the distributions of

$$(\mathsf{Mult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}; \mathbf{R}), \mathbf{R}) \text{ and } (\mathbf{MZ}_{d_3} + w_{d_1}w_{d_2} \otimes \mathbf{I}_\lambda, \mathbf{Z}_{d_3} - \mathbf{Z}_{d_1}\mathbf{W}_{d_2} - w_{d_1} \otimes \mathbf{Z}_{d_2})$$

are identical as well. Then Lemma 3.2 immediately follows. □

Construction of stateless LRC. Let $\{\mathcal{L}_\lambda\}$ be the class of all function families in NC^1 with output length $O(\log\lambda)$ and $\{\mathsf{F}_\lambda : \{0,1\}^{l_{\mathsf{in}}} \to \{0,1\}^{l_{\mathsf{out}}}\}$ be any polynomial-sized function. Without loss of generality, we assume that $\{\mathsf{F}_\lambda\}$ consists only of addition and multiplication gates, and the same argument applies to the other sections. We give a stateless $\{\mathcal{L}_\lambda\}$-LRC SL-LRC in Figure 5.

$\underline{\mathsf{SLEnc}_\lambda(\mathbf{x} \in \{0,1\}^{l_{\mathsf{in}}})}$:

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$

Sample $\mathbf{Z}_i \xleftarrow{\$} \{0,1\}^{\lambda \times \lambda}$ and compute $\mathbf{X}_i = \mathbf{M}\mathbf{Z}_i + x_i \otimes \mathbf{I}_\lambda$ for each $i \in [l_{\mathsf{in}}]$

Return $\widehat{\mathbf{x}} = (\mathbf{M}, (\mathbf{X}_i)_{i \in [l_{\mathsf{in}}]})$

$\underline{\mathsf{SLComp}_\lambda(\widehat{\mathbf{x}} = (\mathbf{M}, (\mathbf{X}_i)_{i \in [l_{\mathsf{in}}]}))}$:

Let the ith input gate x_i in F_λ be represented by $\mathbf{X}_i$ for $i \in [l_{\mathsf{in}}]$

Each gate $w_{d_1} + w_{d_2}$ (respectively, $w_{d_1} w_{d_2}$) in F_λ is implemented as $\mathsf{Add}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2})$ (respectively, $\mathsf{Mult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2})$) where $(\mathbf{W}_{d_1}, \mathbf{W}_{d_2})$ are the wire bundles representing (w_{d_1}, w_{d_2})

Let $(\mathbf{Y}_i)_{i \in [l_{\mathsf{out}}]}$ be the matrices representing the wire bundles for the final output

Return $\widehat{\mathbf{y}} = (\mathbf{M}, (\mathbf{Y}_i)_{i \in [l_{\mathsf{out}}]})$

$\underline{\mathsf{SLDec}_\lambda(\widehat{\mathbf{y}} = (\mathbf{M}, (\mathbf{Y}_i)_{i \in [l_{\mathsf{out}}]}))}$:

Compute $\mathbf{M}^\perp$ such that $\mathbf{M}^\perp \neq \mathbf{0}$ and $\mathbf{M}^\perp \in \mathsf{Ker}(\mathbf{M}^\top)$

For $i \in [l_{\mathsf{out}}]$, let $\mathbf{y}_i$ be the λth column vector in $\mathbf{Y}_i$, compute $y_i = {\mathbf{M}^\perp}^\top \mathbf{y}_i$

Return $\mathbf{y} = (y_i)_{i \in [l_{\mathsf{out}}]}$

Figure 5: Definition of $\mathsf{SL\text{-}LRC} = \{\mathsf{SLEnc}_\lambda, \mathsf{SLComp}_\lambda, \mathsf{SLDec}_\lambda\}$. $\{\mathsf{Add}_\lambda, \mathsf{Mult}_\lambda\}$ are function families defined as in Figure 4.

Theorem 3.3 *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$ *holds, then* $\mathsf{SL\text{-}LRC}$ *in Figure 5 is a stateless* $\{\mathcal{L}_\lambda\}$*-LRC for* $\{\mathsf{F}_\lambda\}$.

Proof. **Correctness.** Let $\mathbf{W}_{d_1} = \mathbf{M}\mathbf{Z}_{d_1} + w_{d_1} \otimes \mathbf{I}_\lambda$ and $\mathbf{W}_{d_2} = \mathbf{M}\mathbf{Z}_{d_2} + w_{d_2} \otimes \mathbf{I}_\lambda$ for some $\mathbf{Z}_{d_1}, \mathbf{Z}_{d_2} \in \{0,1\}^{\lambda \times \lambda}$ and some $w_{d_1}, w_{d_2} \in \{0,1\}$. We have

$$\mathsf{Add}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}; \mathbf{R}) = \mathbf{M}(\mathbf{Z}_{d_1} + \mathbf{Z}_{d_2} + \mathbf{R}) + (w_{d_1} + w_{d_2}) \otimes \mathbf{I}_\lambda,$$
$$\mathsf{Mult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}; \mathbf{R}) = \mathbf{M}(\mathbf{Z}_{d_1}\mathbf{W}_{d_2} + w_{d_1} \otimes \mathbf{Z}_{d_2} + \mathbf{R}) + w_{d_1} w_{d_2} \otimes \mathbf{I}_\lambda,$$

as shown in the proof of Lemma 3.2. Namely, Add_λ and Mult_λ correctly output encodings for $w_{d_1} + w_{d_2}$ and $w_{d_1} w_{d_2}$ respectively.

Therefore, when the input wire bundles are $(\mathbf{M}, (\mathbf{X}_i = \mathbf{M}\mathbf{Z}'_i + x_i \otimes \mathbf{I}_\lambda)_{i \in [l_{\mathsf{in}}]})$, for each $i \in [l_{\mathsf{out}}]$, we must have $\mathbf{Y}_i = \mathbf{M}\mathbf{Z}'_i + y_i \otimes \mathbf{I}_\lambda$, where y_i is the ith bit of $\mathsf{F}_\lambda(\mathbf{x})$, for some $\mathbf{Z}'_i \in \{0,1\}^{\lambda \times \lambda}$, i.e., $\mathbf{y}_i = \mathbf{M}\mathbf{z}_i + \mathbf{e}_\lambda \cdot y_i$ for some $\mathbf{z}_i \in \{0,1\}^\lambda$. By Lemma 2.6, the λth bit in $\mathbf{M}^\perp$ is 1, i.e., ${\mathbf{M}^\perp}^\top \mathbf{e}_\lambda = 1$. Thus we have $y_i = {\mathbf{M}^\perp}^\top \mathbf{y}_i$, completing the proof of correctness.

Leakage-resilience. Let λ be the security parameter. For any $\mathbf{x}_0, \mathbf{x}_1 \in \{0,1\}^{l_{\mathsf{in}}}$, and any $\ell_\lambda \in \mathcal{L}_\lambda$, we define a sequence of intermediate games.

Game G_0**.** This is the real game, where ℓ_λ takes as input $\mathcal{W}$ generated as $\widehat{\mathbf{x}} \xleftarrow{\$} \mathsf{SLEnc}_\lambda(\mathbf{x}_0)$ and $\mathcal{W} \xleftarrow{\$} \mathsf{W}[\mathsf{SLComp}_\lambda(\widehat{\mathbf{x}})]$, and outputs some bit string $\mathbf{b}$.

Game G_1**.** Let $\mathbf{w}^0$ be the assignment to all wires in F_λ on input $\mathbf{x}_0$. G_1 is the same as G_0 except that $\mathcal{W}$ is computed as $\mathcal{W} \xleftarrow{\$} \mathsf{SLSim}_{\mathsf{F}_\lambda, \mathbf{w}^0}(\mathbf{M})$, where $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$, and $\mathsf{SLSim}_{\mathsf{F}_\lambda, \mathbf{w}^0}$ is a (randomized) circuit with $\mathbf{w}^0$ hardwired into it, as defined in Figure 6.

Lemma 3.4 *The distribution of* $\ell_\lambda(\mathcal{W})$ *in* G_0 *is identical to that in* G_1.

Lemma 3.4 follows immediately from Lemma 3.2.

Game G_2**.** G_2 differs from G_1 only in the way that $\mathbf{M}^\top$ is sampled, i.e., $\mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda$ instead of $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$.

Lemma 3.5 *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$ *holds, then the statistical distance between the distributions of* $\ell_\lambda(\mathcal{W})$ *in* G_1 *and* G_2 *is negligible.*

$\underline{\mathsf{SLSim}_{\mathsf{F}_\lambda,\mathbf{w}^0}(\mathbf{M})}$:

$\mathcal{W}$ is initially set as $\{\mathbf{M}\}$

For each wire indexed by i in F_λ assigned with $w_i^0 \in \{0,1\}$, compute a wire bundle $\mathbf{W}_i = \mathbf{M}\mathbf{Z}_i + w_i^0 \otimes \mathbf{I}_\lambda$ where $\mathbf{Z}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$

For each gate $w_{d_3}^0 = w_{d_1}^0 + w_{d_2}^0$, include $\mathsf{SimAdd}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}, \mathbf{Z}_{d_1}, \mathbf{Z}_{d_2}, \mathbf{Z}_{d_3})$ in $\mathcal{W}$

For each gate $w_{d_3}^0 = w_{d_1}^0 w_{d_2}^0$, include $\mathsf{SimMult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}, \mathbf{Z}_{d_1}, \mathbf{Z}_{d_2}, \mathbf{Z}_{d_3}, w_{d_1}^0)$ in $\mathcal{W}$

Return $\mathcal{W}$

Figure 6: Definition of $\mathsf{SLSim}_{\mathsf{F}_\lambda,\mathbf{w}^0}$. $\{\mathsf{SimAdd}_\lambda, \mathsf{SimMult}_\lambda\}$ are the function families defined as in Figure 4. Operations for all input gate gadgets are performed in parallel, followed by parallel execution of operations for addition and multiplication gate gadgets.

Proof. First we note that $\mathsf{SLSim}_{\mathsf{F}_\lambda,\mathbf{w}^0}$ runs in $\mathsf{AC}^0[2]$, since $\mathbf{W}_i$ for all i are computable in parallel in $\mathsf{AC}^0[2]$, followed by running SimAdd_λ and $\mathsf{SimMult}_\lambda$, which are also in $\mathsf{AC}^0[2]$, for all gate gadgets in parallel.

We now construct a function ℓ'_λ, which on input $\mathbf{M}$ with $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$ or $\mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda$, runs $\mathcal{W} \xleftarrow{\$} \mathsf{SLSim}_{\mathsf{F}_\lambda,\mathbf{w}^0}(\mathbf{M})$ and outputs $\ell_\lambda(\mathcal{W})$. Since $\mathsf{SLSim}_{\mathsf{F}_\lambda,\mathbf{w}^0}$ runs in $\mathsf{AC}^0[2]$ and $\{\ell_\lambda\} \in \mathsf{NC}^1$, we have $\{\ell'_\lambda\} \in \mathsf{NC}^1$.

When $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$ (respectively, $\mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda$), the distribution of $\ell_\lambda(\mathcal{W}) = \ell'_\lambda(\mathbf{M})$ is identical to that in G_1 (respectively, G_2). Since the statistical distance between the distributions of $\ell'_\lambda(\mathbf{M})$ in G_1 and G_2 is negligible by Lemma 3.1, the statistical distance between the distributions of $\ell_\lambda(\mathcal{W})$ in G_1 and G_2 is negligible as well, completing the proof of Lemma 3.5. □

Game G_3. In G_3, ℓ_λ takes as input $\mathcal{W}$ where $\widehat{\mathbf{x}} \xleftarrow{\$} \mathsf{SLEnc}'_\lambda(\mathbf{x}_0)$, and $(\mathcal{W}, \widehat{\mathbf{y}}) \Leftarrow \mathsf{SLComp}_\lambda(\widehat{\mathbf{x}})$, and outputs some leaked bit string $\mathbf{b}$. Here, $\{\mathsf{SLEnc}'_\lambda\}$, as defined in Figure 7, is the same as $\{\mathsf{SLEnc}_\lambda\}$ except that $\mathbf{M}^\top$ is sampled from $\mathcal{OS}_\lambda$. One can see that G_3 is the same as G_2 except that the wire values of addition and multiplication gate gadgets are generated using Add_λ and Mult_λ now.

$\underline{\mathsf{SLEnc}'_\lambda(\mathbf{x} \in \{0,1\}^{l_{\mathsf{in}}})}$:

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda$

Sample $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ and compute $\mathbf{X}_i = \mathbf{M}\mathbf{R}_i + x_i \otimes \mathbf{I}_\lambda$ for all $i \in [l_{\mathsf{in}}]$

Return $\widehat{\mathbf{x}} = (\mathbf{M}, (\mathbf{X}_i)_{i\in[l_{\mathsf{in}}]})$

Figure 7: Definition of $\{\mathsf{SLEnc}'_\lambda\}$.

Lemma 3.6 *The distribution of $\ell_\lambda(\mathcal{W})$ in G_3 is identical to its distribution in G_2.*

Similar to Lemma 3.4, Lemma 3.6 follows immediately from Lemma 3.2.

Game G_4. In G_4, $\widehat{\mathbf{x}}$ is generated as $\widehat{\mathbf{x}} \xleftarrow{\$} \mathsf{SLEnc}'_\lambda(\mathbf{x}_1)$ instead of $\widehat{\mathbf{x}} \xleftarrow{\$} \mathsf{SLEnc}'_\lambda(\mathbf{x}_0)$.

Lemma 3.7 *The distribution of $\ell_\lambda(\mathcal{W})$ in G_4 is identical to its distribution in G_3.*

Proof. To prove Lemma 3.7, it suffices to prove that the distributions of $\widehat{\mathbf{x}}_0 \xleftarrow{\$} \mathsf{SLEnc}'_\lambda(\mathbf{x}_0)$ and $\widehat{\mathbf{x}}_1 \xleftarrow{\$} \mathsf{SLEnc}'_\lambda(\mathbf{x}_1)$ are identical. This follows immediately from the fact that $\mathbf{M}$ is full rank when $\mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda$, and thus $\mathbf{X}_i = \mathbf{M}\mathbf{R}_i + x_i \otimes \mathbf{I}_\lambda$ for any $x_i \in \{0,1\}$ and $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ is uniformly distributed in $\{0,1\}^{\lambda\times\lambda}$. □

Let H_0 be another real game, where $\mathcal{W}$ is generated as $\mathcal{W} \xleftarrow{\$} \mathsf{W}[\mathsf{SLComp}_\lambda(\widehat{\mathbf{x}})]$ for $\widehat{\mathbf{x}} \xleftarrow{\$} \mathsf{SLEnc}_\lambda(\mathbf{x}_1)$. By doing all the previous steps in the reverse order, we can prove that the statistical distance between the distributions of $\ell_\lambda(\mathcal{W})$ in G_4 and H_0 is negligible.

Putting all the above together, Theorem 3.3 immediately follows. □

Overheads. In our construction, the encoder size is $O(l_{\mathsf{in}} \cdot \lambda^2)$, as we can compute $\mathbf{X}_i = \mathbf{M}\mathbf{Z}_i$ for each i using $O(\lambda^2)$ gates, given the intermediate values used in $\mathcal{ZS}_\lambda$, as mentioned in Section 2.2. The decoder size is $O(l_{\mathsf{out}} \cdot \lambda + \lambda^2)$, with $O(\lambda^2)$ accounting for the computation of the non-zero vector in the kernel, and $O(l_{\mathsf{out}} \cdot \lambda)$ for recovering $\mathbf{y}$. The total size of the circuit is $O(|\mathsf{F}_\lambda| \cdot \lambda^3)$, with the overhead coming from matrix multiplication in $\{0,1\}^{\lambda \times \lambda}$ for each gate gadget.

In comparison, the stateless LRC treating NC^1-leakage in [19] relies on the alternating group A_λ, where the overhead for the multiplication between each pair of elements is $O(\lambda)$. [3] Their construction involves running all the $O(|\mathsf{F}_\lambda| \cdot \lambda)$ small secure hardware components (denoted by D_{id} in [19]) during the encoding phase, where the size of each component is λ^2. Namely, their encoder size is $O(|\mathsf{F}_\lambda| \cdot \lambda^3)$. Their decoder size is $O(l_{\mathsf{out}} \cdot \lambda^2)$, with overhead from λ multiplications in A_λ for each output wire. Their circuit size is $O(|\mathsf{F}_\lambda| \cdot \lambda^2)$, with overhead from the multiplication of λ elements in A_λ for each gate gadget.

4 Leakage-Tolerant Circuit with Auxiliary Input

In this section, we present a decoding algorithm with efficient simulation and exploit it to propose AI-LTCs against NC^1 under the assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$.

Simulatable decoding. In Figure 8, we define a decoder $\{\mathsf{Dec}_\lambda\}$ and a simulator $\{\mathsf{SimDec}_\lambda\}$. One can see that $\{\mathsf{SimDec}_\lambda\} \in \mathsf{AC}^0[2]$ since $\mathcal{W}_i$ for all $i \in [\lambda]$ are computed in parallel in $\mathsf{AC}^0[2]$.

We show that Dec_λ can determine whether a vector $\mathbf{y} = \mathbf{M}\mathbf{v} + \mathbf{e}_\lambda \cdot b$ for $\mathbf{v} \xleftarrow{\$} \{0,1\}^\lambda$ lies inside or outside $\mathsf{Span}(\mathbf{M})$ for each $\mathbf{M}^\top \in \mathcal{ZS}_\lambda$, corresponding to whether $b = 0$. Moreover, we show that the wire values of Dec_λ can be simulated by SimDec_λ given $\mathbf{v}$. Specifically, we have $\{\mathbf{x} | \mathbf{y} = \mathbf{M}\mathbf{x} + \mathbf{e}_\lambda \cdot b\} = \{\mathbf{x} | \overline{\mathbf{y}} = \overline{\mathbf{M}}\mathbf{x}\}$, since $\underline{\mathbf{M}}^\top \in \mathsf{Span}(\overline{\mathbf{M}}^\top)$. Our decoding algorithm proceeds by first solving $\mathbf{x}$ such that $\mathbf{y} = \overline{\mathbf{M}}\mathbf{x}$ and then computes $b = \underline{\mathbf{y}} - \underline{\mathbf{M}}\mathbf{x}$. Note that there are 2 possible solutions for $\mathbf{x}$, and the solution is determined by randomly selecting x_1. When x_1 happens to be v_1, $\mathbf{x}$ must be $\mathbf{v}$, otherwise, $\mathbf{x} = \mathbf{v} + \mathbf{M}^{\top\perp}$. While computing $\mathbf{x}$ lies beyond the computational power of NC^1, we can simulate the decoding process within NC^1 using SimDec_λ and the knowledge of $\mathbf{v}$, since, like $\mathbf{x}$, $\mathbf{v}$ is also uniformly distributed given that $\overline{\mathbf{y}} = \overline{\mathbf{M}}\mathbf{v}$.

$\underline{\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{y})}$:

Parse $\overline{\mathbf{M}} = \mathsf{F}_\lambda(\mathbf{r})^\top$ where $\mathbf{r} = (r_{1,2}, \cdots, r_{1,\lambda}, r_{2,3}, \cdots, r_{2,\lambda}, \cdots, r_{\lambda-1,\lambda})$

Randomly choose $x_1 \xleftarrow{\$} \{0,1\}$

For $j \in [2:\lambda]$, compute $x_j = y_{j-1} - x_1 r_{1,j} - x_2 r_{2,j} - \cdots - x_{j-1} r_{j-1,j}$

Set $\mathbf{x} = (x_i)_{i \in [\lambda]}$

return $b = \underline{\mathbf{y}} - \underline{\mathbf{M}}\mathbf{x}$

$\underline{\mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{v}, b)}$:

Parse $\overline{\mathbf{M}} = \mathsf{F}_\lambda(\mathbf{r})^\top$ where $\mathbf{r} = (r_{1,2}, \cdots, r_{1,\lambda}, r_{2,3}, \cdots, r_{2,\lambda}, \cdots, r_{\lambda-1,\lambda})$

Set $\mathbf{x} = \mathbf{v}$ and $\mathbf{y} = \mathbf{M}\mathbf{v} + \mathbf{e}_\lambda \cdot b$

For $i \in [2:\lambda]$, compute $\mathcal{W}_{i-1} = \mathsf{W}[y_{i-1} - x_1 r_{1,i} - x_2 r_{2,i} - \cdots - x_{i-1} r_{i-1,i}]$

Compute $\mathcal{W}_\lambda = \mathsf{W}[\underline{\mathbf{y}} - \underline{\mathbf{M}}\mathbf{x}]$

Return $\{\mathcal{W}_i\}_{i \in [\lambda]}$

Figure 8: Definitions of $\{\mathsf{Dec}_\lambda\}$ and $\{\mathsf{SimDec}_\lambda\}$. $\{\mathsf{F}_\lambda\}$ is the function family defined in Section 2.2. In the rest of the paper, we adopt an abuse of notation for $\{\mathsf{Dec}_\lambda\}$ and $\{\mathsf{SimDec}_\lambda\}$, i.e., we use Dec_λ to denote $(\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{y}_{i,j}))_{i \in [m], j \in [n]}$ as $\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{Y})$ and use SimDec_λ to denote $(\mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{x}_{i,j}, b_{i,j}))_{i \in [m], j \in [n]}$ as $\mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{X}, \mathbf{B})$, where $(\mathbf{Y}, \mathbf{X}, \mathbf{B}) = (\mathbf{y}_{i,j}, \mathbf{x}_{i,j}, b_{i,j})_{i \in [m], j \in [n]}$.

Lemma 4.1 *For all $\lambda \in \mathbb{N}$, we have $\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{y}) = b$ for all $\mathbf{M}^\top \in \mathcal{ZS}_\lambda$, all $\mathbf{v} \in \{0,1\}^\lambda$, all $b \in \{0,1\}$, and $\mathbf{y} = \mathbf{M}\mathbf{v} + \mathbf{e}_\lambda \cdot b$.*

[3] In [19], the stateless LRC is called the LRC in the single-query setting.

Proof. For any fixed $b \in \{0,1\}$ and $\mathbf{M}^\top \in \mathcal{ZS}_\lambda$, $\mathbf{x}$ generated in the intermediate steps of Dec_λ is uniformly distributed conditioned on $\overline{\mathbf{y}} = \overline{\mathbf{M}}\mathbf{x}$. Since $\underline{\mathbf{M}} = \widetilde{\mathbf{r}}^\top \overline{\mathbf{M}}$ for some $\widetilde{\mathbf{r}} \in \{0,1\}^{\lambda-1}$, we have

$$\underline{\mathbf{y}} - \underline{\mathbf{M}}\mathbf{x} = \underline{\mathbf{y}} - \widetilde{\mathbf{r}}^\top \overline{\mathbf{M}}\mathbf{x} = \underline{\mathbf{y}} - \widetilde{\mathbf{r}}^\top \overline{\mathbf{y}} = \underline{\mathbf{M}}\mathbf{v} + b - \widetilde{\mathbf{r}}^\top \overline{\mathbf{M}}\mathbf{v} = \underline{\mathbf{M}}\mathbf{v} + b - \underline{\mathbf{M}}\mathbf{v} = b,$$

completing the proof of Lemma 4.1. □

Lemma 4.2 *For all $\lambda \in \mathbb{N}$ and $b \in \{0,1\}$, we have*

$$\Delta(\mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{v}, b), \mathsf{W}[\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{y})]) = 0,$$

where $\mathbf{M}^\top \in \mathcal{ZS}_\lambda$, $\mathbf{v} \xleftarrow{\$} \{0,1\}^\lambda$, and $\mathbf{y} = \mathbf{M}\mathbf{v} + \mathbf{e}_\lambda \cdot b$.

Proof. For any fixed $b \in \{0,1\}$ and $\mathbf{M}^\top \in \mathcal{ZS}_\lambda$, $\mathbf{x}$ generated in the intermediate steps of Dec_λ is uniformly distributed conditioned on $\overline{\mathbf{y}} = \overline{\mathbf{M}}\mathbf{x}$. Moreover, since $\underline{\mathbf{M}} = \widetilde{\mathbf{r}}^\top \overline{\mathbf{M}}$ for some $\widetilde{\mathbf{r}} \in \{0,1\}^{\lambda-1}$, $\mathbf{x}$ is uniformly distributed conditioned on $\mathbf{y} = \mathbf{M}\mathbf{x} + \mathbf{e}_\lambda \cdot b$ as well. Hence, the distributions of $(\mathbf{M}, \mathbf{v}, b)$ and $(\mathbf{M}, \mathbf{x}, b)$ are identical. Therefore, $\mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{v}, b)$ perfectly simulates $\mathsf{W}[\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{y})]$, completing the proof of Lemma 4.2. □

Construction of AI-LTC. Let $\{\mathcal{L}_\lambda\}$ be the class of all function families in NC^1 with output length $O(\log \lambda)$ and $\{\mathsf{F}_\lambda : \{0,1\}^{l_{\mathsf{in}}} \to \{0,1\}^{l_{\mathsf{out}}}\}$ be any polynomial-sized function. We give an $\{\mathcal{L}_\lambda\}$-AI-LTC $\mathsf{AI\text{-}LTC}$ in Figure 9.

$\underline{\mathsf{LTSetup}_\lambda}$:

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$

Sample $\mathbf{v}_i \xleftarrow{\$} \{0,1\}^\lambda$ and compute $\mathbf{u}_i = \mathbf{M}\mathbf{v}_i$ for $i \in [l_{\mathsf{out}}]$

Output $\mathsf{aux} = \left(\mathbf{M}, (\mathbf{u}_i)_{i \in [l_{\mathsf{out}}]}\right)$

$\underline{\mathsf{LTComp}_\lambda(\mathsf{aux} = \left(\mathbf{M}, (\mathbf{u}_i)_{i \in [l_{\mathsf{out}}]}\right), \mathbf{x})}$:

The ith input gate x_i in F_λ is implemented as $\mathbf{M}\mathbf{Z}_i + x_i \otimes \mathbf{I}_\lambda$ where $\mathbf{Z}_i \xleftarrow{\$} \{0,1\}^{\lambda \times \lambda}$

Each gate $w_{d_1} + w_{d_2}$ (respectively, $w_{d_1} w_{d_2}$) in F_λ is implemented as $\mathsf{Add}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2})$ (respectively, $\mathsf{Mult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2})$) where $(\mathbf{W}_{d_1}, \mathbf{W}_{d_2})$ are the wire bundles representing (w_{d_1}, w_{d_2})

Let $(\mathbf{Y}_i)_{i \in [l_{\mathsf{out}}]}$ be the matrices representing the wire bundles for the final output

For $i \in [l_{\mathsf{out}}]$, compute $\mathbf{y}_i = \mathbf{y}'_i + \mathbf{u}_i$ and $y_i = \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{y}_i)$ where $\mathbf{y}'_i$ is the rightmost column of $\mathbf{Y}_i$

Return $\mathbf{y} = (y_i)_{i \in [l_{\mathsf{out}}]}$

Figure 9: Definition of $\mathsf{AI\text{-}LTC} = \{\mathsf{LTSetup}_\lambda, \mathsf{LTComp}_\lambda\}$. $\{\mathsf{Add}_\lambda, \mathsf{Mult}_\lambda, \mathsf{Dec}_\lambda\}$ are the function families defined as in Figures 4 and 8.

Theorem 4.3 *If $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$ holds, then $\mathsf{AI\text{-}LTC}$ in Figure 9 is an $\{\mathcal{L}_\lambda\}$-AI-LTC for $\{\mathsf{F}_\lambda\}$.*

Proof. **Correctness.** As shown in the proof of Lemma 3.2, Add_λ (respectively, Mult_λ) can correctly convert wire bundles of the form

$$\mathbf{W}_{d_1} = \mathbf{M}\mathbf{Z}_{d_1} + w_{d_1} \otimes \mathbf{I}_\lambda \text{ and } \mathbf{W}_{d_2} = \mathbf{M}\mathbf{Z}_{d_2} + w_{d_2} \otimes \mathbf{I}_\lambda$$

into ones of the form

$$\mathbf{W}_{d_3} = \mathbf{M}\mathbf{Z}_{d_3} + (w_{d_1} + w_{d_2}) \otimes \mathbf{I}_\lambda \text{ (respectively, } \mathbf{W}_{d_3} = \mathbf{M}\mathbf{Z}_{d_3} + w_{d_1} w_{d_2} \otimes \mathbf{I}_\lambda).$$

Therefore, for each $i \in [l_{\mathsf{out}}]$, we must have $\mathbf{Y}_i = \mathbf{M}\mathbf{Z}'_i + y_i \otimes \mathbf{I}_\lambda$ for some $\mathbf{Z}'_i \in \{0,1\}^{\lambda \times \lambda}$, where y_i is the ith bit of $\mathsf{F}_\lambda(\mathbf{x})$. Namely, we have $\mathbf{y}_i = \mathbf{M}\mathbf{z}_i + \mathbf{e}_\lambda \cdot y_i$ for some $\mathbf{z}_i \in \{0,1\}^\lambda$, since $\mathbf{u}_i \in \mathsf{Span}(\mathbf{M})$. As a result, for $i \in [l_{\mathsf{out}}]$ we must have $y_i = \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{y}_i)$ when $\mathbf{M}^\top \in \mathcal{ZS}_\lambda$, according to Lemma 4.1, completing the proof of correctness.

$\underline{\mathcal{D}_\lambda :}$

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda$ and initially set $\mathcal{W}$ as $\{\mathbf{M}\}$

For $i \in [l_{\mathsf{in}}]$, represent the ith input gate by $\mathbf{X}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$

For each gate $w_{d_1} + w_{d_2}$ (respectively, $w_{d_1} w_{d_2}$) in F_λ, compute

$$(\mathcal{W}_i, \mathbf{W}_{d_3}) \Leftarrow \mathsf{Add}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}) \text{ (respectively, } (\mathcal{W}_i, \mathbf{W}_{d_3}) \Leftarrow \mathsf{Mult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}))$$

and include $\mathcal{W}_i$ in $\mathcal{W}$, where $(\mathbf{W}_{d_1}, \mathbf{W}_{d_2})$ are wire bundles representing (w_{d_1}, w_{d_2})

Return LTSim_λ with $\mathcal{W}$ and $\mathbf{M}^{-1}$ hardwired into it, defined as follows

$\underline{\mathsf{LTSim}_\lambda(\mathbf{x}, \mathbf{y})}$**:**

For the ith input gate x_i, include $\mathsf{W}[\mathbf{MZ}_i + x_i \otimes \mathbf{I}_\lambda]$ in $\mathcal{W}$ where $\mathbf{Z}_i = \mathbf{M}^{-1}(\mathbf{X}_i - x_i \otimes \mathbf{I}_\lambda)$

For $i \in [l_{\mathsf{out}}]$, let $\mathbf{Y}_i$ be the wire bundle representing the ith output in F_λ and $\mathbf{y}'_i$ be the rightmost column of $\mathbf{Y}_i$, run

$$\mathbf{z}_i \xleftarrow{\$} \{0,1\}^\lambda, \mathbf{y}_i = \mathbf{Mz}_i + \mathbf{e}_\lambda \cdot y_i, \mathbf{u}_i = \mathbf{y}_i - \mathbf{y}'_i$$

and include $(\mathbf{u}_i, \mathsf{W}[\mathbf{y}'_i + \mathbf{u}_i], \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{z}_i, y_i))$ in $\mathcal{W}$

Return $\mathcal{W}$

Figure 10: Definition of $\{\mathcal{D}_\lambda\}$. $\{\mathsf{SimDec}_\lambda\}$ is the function family defined as in Figure 8.

Leakage-tolerance. We give the description of the family of distributions $\{\mathcal{D}_\lambda\}$ over polynomial-sized simulators in Figure 10.

We note that each $\mathsf{LTSim}_\lambda \in \mathcal{D}_\lambda$ runs in $\mathsf{AC}^0[2]$ and thus in NC^1, since it involves only sampling random bits and multiplications and additions of a constant number of matrices in parallel. Next, for any security parameter $\lambda \in \mathbb{N}$, any $\mathbf{x} \in \{0,1\}^{l_{\mathsf{in}}}$ such that $\mathbf{y} = \mathsf{F}_\lambda(\mathbf{x})$, and any $\ell_\lambda \in \mathcal{L}_\lambda$, we define a sequence of intermediate games.

Game G_0. This is the real game where ℓ_λ takes as input $\mathsf{W}[\mathsf{LTComp}_\lambda(\mathsf{aux}, \mathbf{x})]$ for $\mathsf{aux} \xleftarrow{\$} \mathsf{LTSetup}_\lambda$, and outputs some leaked bit string $\mathbf{b}$.

Game G_1. Let $\mathbf{w}$ be the string containing the assignment to all wires in F_λ on input $\mathbf{x}$. In G_1, $\mathcal{W}$ is computed as $\mathcal{W} \xleftarrow{\$} \mathsf{Sim}_{\mathsf{F}_\lambda,\mathbf{w}}(\mathbf{M})$, where $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$ and $\mathsf{Sim}_{\mathsf{F}_\lambda,\mathbf{w}}$ is a (randomized) circuit with $\mathbf{w}$ hardwired into it, as defined in Figure 11.

Lemma 4.4 *The distributions of $\ell_\lambda(\mathcal{W})$ in G_0 and G_1 are identical.*

The proof of Lemma 4.4 follows immediately from Lemma 3.2.

Game G_2. G_2 is the same as G_1 except that in $\mathsf{Sim}_{\mathsf{F}_\lambda,\mathbf{w}}$, $\mathbf{u}_i$ is generated as $\mathbf{u}_i = \mathbf{y}_i - \mathbf{y}'_i$ where $\mathbf{y}_i = \mathbf{Mz}_i + \mathbf{e}_\lambda \cdot y_i$ for $\mathbf{z}_i \xleftarrow{\$} \{0,1\}^\lambda$.

Lemma 4.5 *The distributions of $\ell_\lambda(\mathcal{W})$ in G_1 and G_2 are identical.*

Proof. In G_1, for each output gate y_i, we have

$$\mathbf{y}'_i = \mathbf{Mz}'_i + \mathbf{e}_\lambda \cdot y_i \text{ and } \mathbf{y}_i = \mathbf{y}'_i + \mathbf{u}_i = \mathbf{M}(\mathbf{z}'_i + \mathbf{v}_i) + \mathbf{e}_\lambda \cdot y_i,$$

where $\mathbf{z}'_i \in \{0,1\}^\lambda$ and $\mathbf{u}_i = \mathbf{Mv}_i$ for $\mathbf{v}_i \xleftarrow{\$} \{0,1\}^\lambda$. Hence, the distributions of

$$(\mathbf{y}_i, \mathbf{u}_i) \text{ and } (\mathbf{Mz}_i + \mathbf{e}_\lambda \cdot y_i, \mathbf{Mz}_i + \mathbf{e}_\lambda \cdot y_i - \mathbf{y}'_i)$$

where $\mathbf{z}_i \xleftarrow{\$} \{0,1\}^\lambda$ are identical. Then Lemma 4.5 immediately follows. □

Game G_3. G_3 is the same as G_2 except that in $\mathsf{Sim}_{\mathsf{F}_\lambda,\mathbf{w}}$, $\mathcal{W}_{y_i}$ is generated as $\mathcal{W}_{y_i} = \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{z}_i, y_i)$ instead of $\mathcal{W}_{y_i} \leftarrow \mathsf{W}[\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{y}_i)]$ for all $i \in [l_{\mathsf{out}}]$.

Lemma 4.6 *The distributions of $\ell_\lambda(\mathcal{W})$ in G_2 and G_3 are identical.*

G_1, $\boxed{\mathsf{G}_2}$, $\boxed{\mathsf{G}_3}$, G_4

$\mathsf{Sim}_{\mathsf{F}_\lambda,\mathbf{w}}(\mathbf{M}^\top \in \mathcal{ZS}_\lambda \quad \mathbf{M}^\top \in \mathcal{OS}_\lambda)$:

Sample $\mathbf{v}_i \xleftarrow{\$} \{0,1\}^\lambda$ and compute $\mathbf{u}_i = \mathbf{M}\mathbf{v}_i$ for $i \in [l_{\mathsf{out}}]$

Set $\mathcal{W}$ initially as $\{\mathbf{M}, (\mathbf{u}_i)_{i\in[l_{\mathsf{out}}]}\}$

For the ith input gate x_i in F_λ, run $(\mathcal{W}_{\mathbf{x}_i}, \mathbf{X}_i) \Leftarrow (\mathbf{M}\mathbf{Z}_i + x_i \otimes \mathbf{I}_\lambda)$ where $\mathbf{Z}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ and include $\mathcal{W}_{\mathbf{x}_i}$ in $\mathcal{W}$

For each wire indexed by i in F_λ assigned with w_i, compute $\mathbf{W}_i = \mathbf{M}\mathbf{Z}_i + w_i \otimes \mathbf{I}_\lambda$ where $\mathbf{Z}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$

For each gate $w_{d_3} = w_{d_1} + w_{d_2}$, include $\mathsf{SimAdd}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}, \mathbf{Z}_{d_1}, \mathbf{Z}_{d_2}, \mathbf{Z}_{d_3})$ in $\mathcal{W}$

For each gate $w_{d_3} = w_{d_1} w_{d_2}$, include $\mathsf{SimMult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2}, \mathbf{Z}_{d_1}, \mathbf{Z}_{d_2}, \mathbf{Z}_{d_3}, w_{d_1})$ in $\mathcal{W}$

For $i \in [l_{\mathsf{out}}]$, let $\mathbf{Y}_i$ be the wire bundle representing the ith output in F_λ and $\mathbf{y}'_i$ be the rightmost column of $\mathbf{Y}_i$, run

$$\boxed{\mathbf{u}_i = \mathbf{y}_i - \mathbf{y}'_i \text{ where } \mathbf{y}_i = \mathbf{M}\mathbf{z}_i + \mathbf{e}_\lambda \cdot y_i \text{ for } \mathbf{z}_i \xleftarrow{\$} \{0,1\}^\lambda}$$

$$(\mathcal{W}_{\mathbf{y}'_i}, \mathbf{y}_i) \Leftarrow \mathbf{y}'_i + \mathbf{u}_i, (\mathcal{W}_{y_i}, y_i) \Leftarrow \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{y}_i) \; \boxed{\mathcal{W}_{y_i} = \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{z}_i, y_i)}$$

and include $(\mathcal{W}_{\mathbf{y}'_i}, \mathcal{W}_{y_i})$ in $\mathcal{W}$

Return $\mathcal{W}$

Figure 11: Definitions of $\mathsf{Sim}_{\mathsf{F}_\lambda,\mathbf{w}}$ in $\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3, \mathsf{G}_4$. $\{\mathsf{SimAdd}_\lambda, \mathsf{SimMult}_\lambda\}$ are the function families defined as in Figure 4. Operations for input gate gadgets, addition and multiplication gate gadgets, and output gate gadgets are performed in parallel within each respective category.

Lemma 4.6 follows immediately from Lemma 4.2.

Game G_4. G_4 is the same as G_3 except that $\mathbf{M}$ is sampled as $\mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda$ instead of $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$.

Lemma 4.7 *If $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$ holds, then the distributions of $\ell_\lambda(\mathcal{W})$ in G_3 and G_4 are statistically close with negligible distance.*

Proof. We first note that $\mathsf{Sim}_{\mathsf{F}_\lambda,\mathbf{w}}$ runs in $\mathsf{AC}^0[2]$, since operations for all input gate gadgets, addition and multiplication gate gadgets, and output gate gadgets are performed in $\mathsf{AC}^0[2]$ within each respective category, and we have $\{\mathsf{SimDec}_\lambda\} \in \mathsf{AC}^0[2]$. Hence, we have $\{\ell_\lambda \circ \mathsf{Sim}_{\mathsf{F}_\lambda,\mathbf{w}}\} \in \mathsf{NC}^1$. Then Lemma 4.7 follows immediately from Lemma 3.1. □

Game G_5. G_5 is the same as G_4 except that assignments to wires of all addition and multiplication gate gadgets are generated in the same way as in G_0. Namely, for each gate $w_{d_1} + w_{d_2}$ (respectively, $w_{d_1} w_{d_2}$) in F_λ, we compute $(\mathcal{W}_i, \mathbf{W}_{d_3}) \Leftarrow \mathsf{Add}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2})$ (respectively, $(\mathcal{W}_i, \mathbf{W}_{d_3}) \Leftarrow \mathsf{Mult}_\lambda(\mathbf{M}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2})$).

Lemma 4.8 *The distributions of $\ell_\lambda(\mathcal{W})$ in G_4 and G_5 are identical.*

Similar to Lemma 4.4, Lemma 4.8 follows immediately from Lemma 3.2.

Game G_6. In G_6, $\mathcal{W}$ is generated as $\mathsf{LTSim}_\lambda(\mathbf{x}, \mathbf{y})$ where $\mathsf{LTSim}_\lambda \xleftarrow{\$} \mathcal{D}_\lambda$. One can see that $\ell_\lambda(\mathcal{W})$ in G_6 is identical to that in G_5 except that $\mathbf{X}_i$ is generated as $\mathbf{X}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ and $\mathbf{Z}_i$ is generated as $\mathbf{Z}_i = \mathbf{M}^{-1}(\mathbf{X}_i - x_i \otimes \mathbf{I}_\lambda)$.

Lemma 4.9 *The distributions of $\ell_\lambda(\mathcal{W})$ in G_5 and G_6 are identical.*

Proof. Lemma 4.9 follows immediately from the fact that $\mathbf{M}^\top \in \mathcal{OS}_\lambda$ is of full rank and the following distributions are the same:

$$(\mathbf{Z}_i, \mathbf{M}\mathbf{Z}_i + x_i \otimes \mathbf{I}_\lambda) \text{ and } (\mathbf{M}^{-1}(\mathbf{X}_i - x_i \otimes \mathbf{I}_\lambda), \mathbf{X}_i)$$

where $\mathbf{Z}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ and $\mathbf{X}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ □

Putting all the above together, Theorem 4.3 immediately follows. □

5 Stateful Leakage-Resilient Circuit against NC^1

In this section, we propose a generic construction of stateful LRC from AI-LTC, which leads to an instantiation of stateful LRC against NC^1-leakage using one compact hardware component in each invocation. Additionally, we give another instantiation with a smaller circuit size constructed in a "non-black-box" manner.

5.1 Generic Construction

2-adaptive leakage-resilient encoding. Below we define a 2-adaptive leakage-resilient encoding scheme and a property we call composability. We then propose an instantiation against NC^1-leakage under the assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$.

Definition 5.1 (2-adaptive leakage-resilient encoding). An *encoding scheme* with message length l_{m} consists of polynomial-sized function families $\{\mathsf{E}_\lambda, \mathsf{D}_\lambda\}$ such that

- E_λ on input a message $\mathbf{m} \in \{0,1\}^{l_{\mathsf{m}}}$ outputs an encoded message $\widehat{\mathbf{m}}$,
- D_λ on input $\widehat{\mathbf{m}}$ outputs a message $\mathbf{m} \in \{0,1\}^{l_{\mathsf{m}}}$.

Correctness is satisfied if for any $\lambda \in \mathbb{N}$ and any $\mathbf{m} \in \{0,1\}^{l_{\mathsf{m}}}$, we have $\mathsf{D}_\lambda(\mathsf{E}_\lambda(\mathbf{m})) = \mathbf{m}$.

2-adaptive $\{\mathcal{L}_\lambda\}$-leakage-resilience is satisfied if for all $\lambda \in \mathbb{N}$, any $\ell_\lambda \in \mathcal{L}_\lambda$, any $\mathbf{m}_0, \mathbf{m}_1 \in \{0,1\}^{l_{\mathsf{m}}}$, and any function $\mathcal{A}$ with its range being $\mathcal{L}_\lambda$, the following two distributions are statistically close with negligible distance:

$$\big(\ell_\lambda(\widehat{\mathbf{m}}), \ell'_\lambda(\widehat{\mathbf{m}})\big) : \widehat{\mathbf{m}} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{m}_0), \ell'_\lambda = \mathcal{A}(\ell_\lambda(\widehat{\mathbf{m}})),$$

$$\big(\ell_\lambda(\widehat{\mathbf{m}}), \ell'_\lambda(\widehat{\mathbf{m}})\big) : \widehat{\mathbf{m}} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{m}_1), \ell'_\lambda = \mathcal{A}(\ell_\lambda(\widehat{\mathbf{m}})).$$

$\{\mathcal{L}_\lambda\}$-leakage-resilience is satisfied if the output of $\mathcal{A}$ is restricted to $\perp$.

Definition 5.2 (Composability). An *encoding scheme* $\mathsf{Encode} = \{\mathsf{E}_\lambda, \mathsf{D}_\lambda\}$ with message length $l_{\mathsf{m}} = l_{\mathsf{m}}{}' + l_{\mathsf{r}}$ (see Definition 5.1) satisfies l_{r}-composability if

1. for all $\lambda \in \mathbb{N}$, E_λ can be decomposed into E^1_λ and E^2_λ, such that for all $\mathbf{m} \in \{0,1\}^{l_{\mathsf{m}}{}'}$ and $\mathbf{r} \in \{0,1\}^{l_{\mathsf{r}}}$, we have $\mathsf{E}_\lambda(\mathbf{m}, \mathbf{r}) = \mathsf{E}^1_\lambda(\mathbf{m}) || \mathsf{E}^2_\lambda(\mathsf{E}^1_\lambda(\mathbf{m}), \mathbf{r})$,
2. there exists a public-coin distribution family $\{\mathcal{R}_{\lambda,l_{\mathsf{r}}}\}$ such that for all $\lambda \in \mathbb{N}$, all $\mathbf{m} \in \{0,1\}^{l_{\mathsf{m}}{}'}$, and all $\mathsf{ct} \in \mathsf{E}^1_\lambda(\mathbf{m})$, the following distributions are identical:

$$\mathsf{ct}_{\mathsf{rand}} \xleftarrow{\$} \mathcal{R}_{\lambda,l_{\mathsf{r}}} \text{ and } \mathsf{ct}_{\mathsf{rand}} \xleftarrow{\$} \mathsf{E}^2_\lambda(\mathsf{ct}, \mathbf{r}) \text{ where } \mathbf{r} \xleftarrow{\$} \{0,1\}^{l_{\mathsf{r}}}.$$

Our encoding scheme. Let $\{\mathcal{L}_\lambda\}$ be the class of all function families in NC^1 with output length $O(\log \lambda)$. We give a 2-adaptive $\{\mathcal{L}_\lambda\}$-leakage-resilient encoding scheme Encode with any polynomial message length $l_{\mathsf{m}} = l_{\mathsf{m}}(\lambda)$ in Figure 12.

$\underline{\mathsf{E}_\lambda(\mathbf{m} \in \{0,1\}^{l_{\mathsf{m}}}) :}$	$\underline{\mathsf{D}_\lambda(\widehat{\mathbf{m}} = (\mathbf{M}, \mathbf{S})) :}$
Sample $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$ and $\mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda \times l_{\mathsf{m}}}$	Compute $\mathbf{m} = \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{S}; \mathbf{0})$
Compute $\mathbf{S} = \mathbf{MR} + \mathbf{e}_\lambda \mathbf{m}^\top$	Return $\mathbf{m}$
Return $\widehat{\mathbf{m}} = (\mathbf{M}, \mathbf{S})$	

Figure 12: Definition of $\mathsf{Encode} = \{\mathsf{E}_\lambda, \mathsf{D}_\lambda\}$. See Figure 8 for the definition of $\{\mathsf{Dec}_\lambda\}$. Here Dec_λ is deterministic, as its internal random coins are always fixed as 0's.

Theorem 5.3 *If $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$ holds, then Encode is a 2-adaptive $\{\mathcal{L}_\lambda\}$-leakage-resilient encoding scheme and satisfies l_{r}-composability for any $l_{\mathsf{r}} \le l_{\mathsf{m}}$.*

To prove Theorem 5.3, we present a lemma showing the 2-adaptive indistinguishability of matrices from $\mathcal{ZS}_\lambda$ and $\mathcal{OS}_\lambda$, and then proceed with the proof by switching the distribution of $\mathbf{M}^\top$ between $\mathcal{ZS}_\lambda$ and $\mathcal{OS}_\lambda$ in the presence of 2-adaptive leakage on it. For the detailed proof, we refer the reader to Appendix A.

Construction of stateful LRC. Let $\{\mathcal{L}_\lambda\}$ be a class of function families that is closed under composition. Let $\{\mathsf{F}_\lambda : \{0,1\}^{l_{\mathsf{st}}} \times \{0,1\}^{l_{\mathsf{in}}} \to \{0,1\}^{l_{\mathsf{st}}} \times \{0,1\}^{l_{\mathsf{out}}}\}$ be any polynomial-sized stateful function family. Let $\mathsf{Encode} = \{\mathsf{E}_\lambda, \mathsf{D}_\lambda\}$ be a 2-adaptive $\{\mathcal{L}_\lambda\}$-leakage-resilient encoding scheme satisfying l_{r}-composability. Assume that $\{\mathsf{E}_\lambda\}$ can be decomposed into $\{\mathsf{E}^1_\lambda, \mathsf{E}^2_\lambda\}$ with the message space and internal randomness space of E^1_λ being $\{0,1\}^{l_{\mathsf{st}}}$ and $\{0,1\}^{l_{\mathsf{r}}}$ respectively, and the associated public-coin distribution family being $\{\mathcal{R}_{\lambda,l_{\mathsf{r}}}\}$. We define the function family $\{\mathsf{F}^*_\lambda\}$ as in Figure 13. Let $\mathsf{AI\text{-}LTC} = \{\mathsf{LTSetup}_\lambda, \mathsf{LTComp}_\lambda\}$ be an $\{\mathcal{L}_\lambda\}$-AI-LTC for $\{\mathsf{F}^*_\lambda\}$. We propose a stateful $\{\mathcal{L}_\lambda\}$-LRC $\mathsf{SF\text{-}LRC}$ for $\{\mathsf{F}_\lambda\}$ in Figure 14.

$\underline{\mathsf{F}^*_\lambda(\mathsf{ct}, \mathsf{ct}_{\mathsf{rand}}, \mathbf{x}) :}$

Compute $(\mathbf{s}, \mathbf{r}) = \mathsf{D}_\lambda(\mathsf{ct}||\mathsf{ct}_{\mathsf{rand}})$, $(\mathbf{s}', \mathbf{y}) = \mathsf{F}_\lambda[\mathbf{s}](\mathbf{x})$, and $\mathsf{ct}' = \mathsf{E}^1_\lambda(\mathbf{s}'; \mathbf{r})$

Return $(\mathsf{ct}', \mathbf{y})$

Figure 13: Definition of $\{\mathsf{F}^*_\lambda\}$.

$\underline{\mathsf{SFSetup}_\lambda(\mathbf{s}) :}$	$\underline{\mathsf{SFComp}_\lambda(\mathsf{st} = \mathsf{ct}, \mathbf{x}):}$
Run $\mathsf{ct} \xleftarrow{\$} \mathsf{E}^1_\lambda(\mathbf{s})$	Sample $\mathsf{ct}_{\mathsf{rand}} \xleftarrow{\$} \mathcal{R}_{\lambda,l_{\mathsf{r}}}$ and $\mathsf{aux} \xleftarrow{\$} \mathcal{O}_{\mathsf{aux}}$
Return $\mathsf{st} = \mathsf{ct}$	Run $(\mathsf{ct}', \mathbf{y}) \xleftarrow{\$} \mathsf{LTComp}_\lambda(\mathsf{aux}, (\mathsf{ct}, \mathsf{ct}_{\mathsf{rand}}, \mathbf{x}))$
	Return $(\mathsf{st}' = \mathsf{ct}', \mathbf{y})$

Figure 14: Definition of $\mathsf{SF\text{-}LRC} = \{\mathsf{SFSetup}_\lambda, \mathsf{SFComp}_\lambda\}$. $\mathcal{O}_{\mathsf{aux}}$ is an opaque gate that runs $\mathsf{LTSetup}_\lambda$ and returns its output.

Theorem 5.4 $\mathsf{SF\text{-}LRC}$ *is a stateful* $\{\mathcal{L}_\lambda\}$*-LRC for* $\{\mathsf{F}_\lambda\}$.

Roughly, as discussed in the technical overview (see Section 1.3), we can treat $\mathsf{ct}_{\mathsf{rand}}$ sampled with public coins as an encoding of random strings. Then the leakage-resilience of $\mathsf{SF\text{-}LRC}$ follows from the fact that the computation phase of $\mathsf{AI\text{-}LTC}$ reveals no information on the state beyond its encoding, and we can replace the encoding by an encoding of 0, which effectively fools the 2-adaptive leakage. We refer the reader to Appendix B for the detailed proof.

Overheads of our instantiation. By instantiating the underlying leakage-resilient encoding scheme as in Figure 12 and the underlying AI-LTC as in Figure 9, the size of the setup algorithm and that of the opaque gate in each invocation are $O(l_{\mathsf{st}} \cdot \lambda^2)$ and $O(l_{\mathsf{st}} \cdot \lambda^2 + l_{\mathsf{out}} \cdot \lambda)$ respectively, and the size of the stateful LRC is

$$\underbrace{O(l_{\mathsf{st}} \cdot \lambda^3 + |\mathsf{F}_\lambda|)}_{\text{size of } \mathsf{F}^*_\lambda} \cdot \lambda^3 + O(l_{\mathsf{st}} \cdot \lambda^2 + l_{\mathsf{out}} \cdot \lambda) = O(l_{\mathsf{st}} \cdot \lambda^6 + |\mathsf{F}_\lambda| \cdot \lambda^3),$$

where the size overhead of λ^3 for each gate in F^*_λ comes from the multiplication of matrices in $\{0,1\}^{\lambda \times \lambda}$.

In comparison, in the stateful LRC in [19], the opaque gate size is $O(|\mathsf{F}_\lambda| \cdot \lambda^3)$, with overhead from running λ small secure hardware components (denoted by D_{id} in [19]) of size $O(\lambda^2)$ for each gate gadget. The setup and decoder sizes are $O(l_{\mathsf{st}} \cdot \lambda^2)$ and $O(l_{\mathsf{out}} \cdot \lambda^2)$ respectively, with overhead from the multiplication of λ elements in A_λ for each state or output wire. The circuit size is $O(|\mathsf{F}_\lambda| \cdot \lambda^3)$, coming from the fact that the opaque gate has this size. [4]

5.2 Instantiation with Smaller Circuit Size

Let $\{\mathcal{L}_\lambda\}$ be the class of all function families in NC^1 with output length $O(\log \lambda)$. Let $\{\mathsf{F}_\lambda : \{0,1\}^{l_{\mathsf{st}}} \times \{0,1\}^{l_{\mathsf{in}}} \to \{0,1\}^{l_{\mathsf{st}}} \times \{0,1\}^{l_{\mathsf{out}}}\}$ be any polynomial-sized stateful function family. We propose a stateful $\{\mathcal{L}_\lambda\}$-LRC $\mathsf{SF\text{-}LRC}^*$ in Figure 15, which has smaller circuit size, compared to our instantiation of the generic construction proposed in the last section.

[4] Excluding the opaque gate, the circuit size in [19] is $O(|\mathsf{F}_\lambda| \cdot \lambda^2)$.

$\underline{\mathsf{SFSetup}^*_\lambda(\mathbf{s}) :}$

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$

Sample $\mathbf{V}_i \xleftarrow{\$} \{0,1\}^{\lambda^2 \times \lambda}$ and compute $\mathbf{U}_i = (\mathbf{M} \otimes \mathbf{I}_\lambda)\mathbf{V}_i$ for $i \in [l_{\mathsf{st}}]$

Sample $\mathbf{Q}_i \xleftarrow{\$} \{0,1\}^{\lambda \times \lambda}$ and compute $\mathbf{S}_i = \mathbf{M}\mathbf{Q}_i + s_i \otimes \mathbf{I}_\lambda$ for $i \in [l_{\mathsf{st}}]$

Output $\mathsf{st} = \left(\mathbf{M}, (\mathbf{S}_i, \mathbf{U}_i)_{i\in[l_{\mathsf{st}}]}\right)$

$\underline{\mathsf{SFComp}^*_\lambda(\mathsf{st} = \left(\mathbf{M}, (\mathbf{S}_i, \mathbf{U}_i)_{i\in[l_{\mathsf{st}}]}\right), \mathbf{x}):}$

State Refresh:

Sample $(\mathbf{M}', (\mathbf{u}'_i)_{i\in[l_{\mathsf{out}}]}, (\mathbf{U}'_i)_{i\in[l_{\mathsf{st}}]}) \xleftarrow{\$} \mathcal{O}^*_{\mathsf{aux}}$

For $i \in [l_{\mathsf{st}}]$, run $\mathbf{P}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}, \mathbf{D}_i = \mathbf{P}_i\mathbf{M}'^\top + \mathbf{U}_i + \mathbf{s}_i \otimes \mathbf{I}_\lambda, \mathbf{T}_i^\top = \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{D}_i)$ where $\mathbf{s}_i$ is the rightmost column of $\mathbf{S}_i$

Computation:

The ith wire of the state in F_λ is represented by $\mathbf{T}_i$

The ith input gate x_i in F_λ is implemented as

$$\mathbf{X}_i = \mathbf{M}'\mathbf{Z}_i + x_i \otimes \mathbf{I}_\lambda \text{ for } \mathbf{Z}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$$

Each gate $w_{d_1} + w_{d_2}$ (respectively, $w_{d_1}w_{d_2}$) in F_λ is implemented as

$$\mathsf{Add}_\lambda(\mathbf{M}', \mathbf{W}_{d_1}, \mathbf{W}_{d_2}) \text{ (respectively, } \mathsf{Mult}_\lambda(\mathbf{M}', \mathbf{W}_{d_1}, \mathbf{W}_{d_2}))$$

where $(\mathbf{W}_{d_1}, \mathbf{W}_{d_2})$ are the wire bundles representing (w_{d_1}, w_{d_2})

Let $(\mathbf{S}'_i)_{i\in[l_{\mathsf{st}}]}$ and $(\mathbf{Y}_i)_{i\in[l_{\mathsf{out}}]}$ be the matrices representing the updated state and final output in F_λ respectively

For $i \in [l_{\mathsf{out}}]$, letting $\mathbf{y}'_i$ be the rightmost column of $\mathbf{Y}_i$, compute

$$\mathbf{y}_i = \mathbf{y}'_i + \mathbf{u}'_i \text{ and } y_i = \mathsf{Dec}_\lambda(\mathbf{M}', \mathbf{y}_i)$$

Return $\left(\mathsf{st}' = \left(\mathbf{M}', (\mathbf{S}'_i, \mathbf{U}'_i)_{i\in[l_{\mathsf{st}}]}\right), \mathbf{y} = (y_i)_{i\in[l_{\mathsf{out}}]}\right)$

Figure 15: Definition of $\mathsf{SF\text{-}LRC}^* = \{\mathsf{SFSetup}^*_\lambda, \mathsf{SFComp}^*_\lambda\}$. $\mathcal{O}^*_{\mathsf{aux}}$ is the opaque gate defined in Figure 16.

$\underline{\mathcal{O}^*_{\mathsf{aux}} :}$

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$

Sample $\mathbf{v}_i \xleftarrow{\$} \{0,1\}^\lambda$ and compute $\mathbf{u}_i = \mathbf{M}\mathbf{v}_i$ for $i \in [l_{\mathsf{out}}]$

Sample $\mathbf{V}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$ and compute $\mathbf{U}_i = (\mathbf{M} \otimes \mathbf{I}_\lambda)\mathbf{V}_i$ for $i \in [l_{\mathsf{st}}]$

Output $\left(\mathbf{M}, (\mathbf{u}_i)_{i\in[l_{\mathsf{out}}]}, (\mathbf{U}_i)_{i\in[l_{\mathsf{st}}]}\right)$

Figure 16: Definition of the opaque gate $\mathcal{O}^*_{\mathsf{aux}}$.

Theorem 5.5 *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$*, then* $\mathsf{SF\text{-}LRC}^*$ *is a stateful* $\{\mathcal{L}_\lambda\}$*-LRC for* $\{\mathsf{F}_\lambda\}$*.*

Roughly, our construction of $\mathsf{SF\text{-}LRC}^*$ can be seen as a "non-black-box" extension of our AI-LTC given in Section 4, with the addition of a state-updating procedure using the re-encryption technique as discussed in Section 1.3. Throughout the proof, we aim to build a simulator that runs in NC^1 for each invocation, and then switch the distribution of $\mathbf{M}^\top$ to $\mathcal{OS}_\lambda$ in each encoded state. We refer the reader to Appendix C for the full proof.

Overheads. In this instantiation, the sizes of the setup algorithm and the opaque gate in each invocation are $O(l_{\mathsf{st}} \cdot \lambda^3)$ and $O(l_{\mathsf{st}} \cdot \lambda^3 + l_{\mathsf{out}} \cdot \lambda)$ respectively, where the latter is larger than $O(l_{\mathsf{st}} \cdot \lambda^2 + l_{\mathsf{out}} \cdot \lambda)$ in our construction in Section 5.1. However, the compiled circuit size is $O(l_{\mathsf{st}} \cdot \lambda^4 + |\mathsf{F}_\lambda| \cdot \lambda^3)$, which is smaller

than $O(l_{\mathsf{st}} \cdot \lambda^6 + |\mathsf{F}_\lambda| \cdot \lambda^3)$ in our construction in Section 5.1.

6 Fine-grained oNIZK from AI-LTC and Limitations

In this section, we present a generic transformation from AI-LTCs to fine-grained oNIZKs, by using a leakage-resilient encoding scheme.

6.1 Fine-Grained oNIZK

We now recall the definition of oNIZK (with perfect soundness) in the fine-grained setting, and then give its generic construction. Below, for a language family $\{\mathcal{V}_\lambda\}$ with the associated relation $\{\mathsf{R}_\lambda\}$, by $\mathbf{x} \in \mathcal{V}_\lambda$ we mean that there exists $\mathbf{w}$ such that $\mathsf{R}_\lambda(\mathbf{x}, \mathbf{w}) = 1$.

Definition 6.1 (Non-interactive zero-knowledge proof with offline simulation). A $\{\mathcal{C}_\lambda\}$-*non-interactive zero-knowledge proof with offline simulation (oNIZK)* for a language family $\{\mathcal{V}_\lambda\}$ with the associated relation $\{\mathsf{R}_\lambda\}$ consists of polynomial-sized function families $\{(\mathsf{Gen}_\lambda \in \mathcal{C}_\lambda, \mathsf{Prove}_\lambda, \mathsf{Ver}_\lambda \in \mathcal{C}_\lambda)\}$ such that

- Gen_λ returns a CRS crs.
- $\mathsf{Prove}_\lambda(\mathsf{crs}, \mathbf{x}, \mathbf{w})$ returns a proof π.
- $\mathsf{Ver}_\lambda(\mathsf{crs}, \mathbf{x}, \pi)$ deterministically returns 1 (accept) or 0 (reject).

Completeness is satisfied if for all $\lambda \in \mathbb{N}$, all $(\mathbf{x}, \mathbf{w})$ such that $\mathsf{R}_\lambda(\mathbf{x}, \mathbf{w}) = 1$, all $\mathsf{crs} \in \mathsf{Gen}_\lambda$, and all $\pi \in \mathsf{Prove}_\lambda(\mathsf{crs}, \mathbf{x}, \mathbf{w})$, we have $\mathsf{Ver}_\lambda(\mathsf{crs}, \mathbf{x}, \pi) = 1$.

Zero-knowledge with offline simulation is satisfied if there exists a polynomial-time distribution family $\{\mathcal{D}_\lambda^{\mathsf{NIZK}}\}$ over circuits in $\{\mathcal{C}_\lambda\}$ such that for all $\lambda \in \mathbb{N}$, any $(\mathbf{x}, \mathbf{w})$ such that $\mathsf{R}_\lambda(\mathbf{x}, \mathbf{w}) = 1$, and any adversary $\mathcal{A}_\lambda \in \mathcal{C}_\lambda$, we have

$$\begin{aligned}&|\Pr[1 \xleftarrow{\$} \mathcal{A}_\lambda(\mathsf{crs}, \pi) | \mathsf{crs} \xleftarrow{\$} \mathsf{Gen}_\lambda, \pi \xleftarrow{\$} \mathsf{Prove}_\lambda(\mathsf{crs}, \mathbf{x}, \mathbf{w})] \\ &- \Pr[1 \xleftarrow{\$} \mathcal{A}_\lambda(\mathsf{crs}, \pi) | \mathsf{NIZKSim}_\lambda \xleftarrow{\$} \mathcal{D}_\lambda^{\mathsf{NIZK}}, (\mathsf{crs}, \pi) \xleftarrow{\$} \mathsf{NIZKSim}_\lambda(\mathbf{x})]| \leq \mathsf{negl}(\lambda).\end{aligned}$$

Perfect soundness is satisfied if for all $\lambda \in \mathbb{N}$, all $\mathsf{crs} \in \mathsf{Gen}_\lambda$, all $\mathbf{x} \notin \mathcal{V}_\lambda$, and all π, we have $\mathsf{Ver}_\lambda(\mathsf{crs}, \mathbf{x}, \pi) = 0$.

As shown in [2], zero-knowledge with offline simulation implies both *witness hiding* and *weak zero-knowledge*.

Our construction. Let $\{\mathcal{L}_\lambda\}$ be a class of function families that is closed under composition and contains all circuits in AC^0. Let $\{\mathcal{C}_\lambda\}$ be $\{\widetilde{\mathcal{L}}_\lambda\}$ (see Section 2.1 for the notation). Let $\{\mathcal{V}_\lambda\}$ be a language family. Let $\{\mathsf{R}_\lambda\}$ be the associated relation for $\{\mathcal{V}_\lambda\}$. Let $\mathsf{Encode} = \{\mathsf{E}_\lambda, \mathsf{D}_\lambda\}$ be an $\{\mathcal{L}_\lambda\}$-leakage-resilient encoding scheme. Let $\mathsf{AI\text{-}LTC} = \{\mathsf{LTSetup}_\lambda \in \mathcal{C}_\lambda, \mathsf{LTComp}_\lambda\}$ [5] be an $\{\mathcal{L}_\lambda\}$-AI-LTC for $\{\mathsf{F}_\lambda\}$ such that $\mathsf{F}_\lambda(\mathbf{x}, \mathsf{ct}) = \mathsf{R}_\lambda\big(\mathbf{x}, \mathsf{D}_\lambda(\mathsf{ct})\big)$ for all λ, all $\mathbf{x}$, and all ct. We construct an $\{\mathcal{C}_\lambda\}$-oNIZK NIZK, as defined in Figure 17, for a language family with the associated relation $\{\mathsf{R}_\lambda\}$.

Theorem 6.2 NIZK *is a* $\{\mathcal{C}_\lambda\}$*-oNIZK for* $\{\mathcal{V}_\lambda\}$ *with a (depth-3)* AC^0*-verifier.*

At a high level, the correctness and soundness of NIZK follow from the correctness of the underlying AI-LTC and encoding scheme. Zero-knowledge with offline simulation is guaranteed, as we can fool the adversary by using AI-LTC simulators and dummy encodings. We refer the reader to Appendix D for the detailed proof.

Instantiation. By instantiating the underlying AI-LTC and encoding scheme in NIZK by the ones in Figures 9 and 12 respectively, we immediately obtain an NC^1-oNIZK. Namely, we have the following corollary.

Corollary 6.3 *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$ *holds, then there exists an* NC^1*-oNIZK with a (depth-3)* AC^0*-verifier for all* NP*.*

[5] We require that $\mathsf{LTSetup}_\lambda \in \mathcal{C}_\lambda$ to match the definition of fine-grained oNIZK. This requirement is mild and satisfied by our AI-LTC in Section 4 when $\{\mathcal{C}_\lambda\}$ is NC^1. It is unnecessary for proving our negative result.

$\underline{\mathsf{Gen}_\lambda}$:

Sample $\mathsf{aux} \xleftarrow{\$} \mathsf{LTSetup}_\lambda$ and return $\mathsf{crs} = \mathsf{aux}$

$\underline{\mathsf{Prove}_\lambda(\mathsf{crs}, \mathbf{x}, \mathbf{w})}$:

Compute $\mathsf{ct} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{w})$, $(\mathcal{W}, b) \Leftarrow \mathsf{LTComp}_\lambda(\mathsf{crs}, (\mathbf{x}, \mathsf{ct}))$, and return $\mathcal{W}$

$\underline{\mathsf{Ver}_\lambda(\mathsf{crs} = \mathsf{aux}, \mathbf{x}, \pi = \mathcal{W})}$:

Check if the input wire assignment in $\mathcal{W}$ is consistent with aux and $\mathbf{x}$

Check if the wire assignments in $\mathcal{W}$ are consistent with all gates in LTComp_λ

Check that the output wire of LTComp_λ in $\mathcal{W}$ is assigned 1

Return 1 iff all checks pass

Figure 17: Definition of $\mathsf{NIZK} = \{\mathsf{Gen}_\lambda, \mathsf{Prove}_\lambda, \mathsf{Ver}_\lambda\}$.

6.2 Limitations on Fine-Grained oNIZK and AI-LTC

We now define verifiable CRS generation and (fine-grained) subset membership problem, and then show that any oNIZK (with perfect soundness) does not have verifiable CRS, assuming the existence of a subset membership problem for its language family.

Definition 6.4 (Verifiable CRS generation). A $\{\mathcal{C}_\lambda\}$-oNIZK with the CRS generator $\{\mathsf{Gen}_\lambda\}$ has *verifiable CRS generation* if there exists a function family $\{\mathsf{Check}_\lambda \in \mathcal{C}_\lambda\}$ such that Check_λ on input a CRS crs outputs 1 (accept) if $\mathsf{crs} \in \mathsf{Gen}_\lambda$ and 0 (reject) otherwise.

Definition 6.5 (Subset membership problem). A $\{\mathcal{C}_\lambda\}$-*subset membership problem* for a language family $\{\mathcal{V}_\lambda\}$ with the associated polynomial-time relation $\{\mathsf{R}_\lambda\}$ consists of two distribution families $\{\mathcal{YS}_\lambda\}$ and $\{\mathcal{NS}_\lambda\}$, sampleable by circuits in $\mathcal{C}_\lambda$, such that for all $\lambda \in \mathbb{N}$ and all $\mathcal{A}_\lambda \in \mathcal{C}_\lambda$, we have

- $\mathsf{R}_\lambda(\mathbf{x}, \mathbf{w}) = 1$ for all $(\mathbf{x}, \mathbf{w}) \in \mathcal{YS}_\lambda$ and $\mathbf{x} \notin \mathcal{V}_\lambda$ for all $\mathbf{x} \in \mathcal{NS}_\lambda$,
- $\Delta\big(\mathcal{A}_\lambda(\mathbf{x}_0), \mathcal{A}_\lambda(\mathbf{x}_1)\big) \leq \mathsf{negl}(\lambda)$ for $(\mathbf{x}_0, \mathbf{w}) \xleftarrow{\$} \mathcal{YS}_\lambda$ and $\mathbf{x}_1 \xleftarrow{\$} \mathcal{NS}_\lambda$.

Theorem 6.6 *Let $\{\mathcal{C}_\lambda\}$ be a class of function families closed under composition. If there exists a $\{\mathcal{C}_\lambda\}$-subset membership problem $\{\mathcal{YS}_\lambda, \mathcal{NS}_\lambda\}$ for $\{\mathcal{V}_\lambda\}$, then any $\{\mathcal{C}_\lambda\}$-oNIZK for $\{\mathcal{V}_\lambda\}$ does not have verifiable CRS generation.* [6]

As mentioned in the technical overview (see Section 1.3), we prove the negative result by building a simulator breaking the perfect soundness, using the verification for CRSs. We refer the reader to Appendix E for the detailed proof.

Negative result on AI-LTCs. One can easily see that the CRS of NIZK in Figure 17 is exactly the auxiliary input of the underlying AI-LTC, and it can have verifiable CRS generation if the setup of $\mathsf{AI\text{-}LTC}$ is a public-coin algorithm. We immediately have the following negative result on AI-LTCs for the relation-checking function family used in the transformation, combining Theorem 6.6 with Theorem 6.2.

Corollary 6.7 *Let $\{\mathcal{L}_\lambda\}$ be a class of function families that is closed under composition and contains all circuits in AC^0. Let $\mathsf{Encode} = \{\mathsf{E}_\lambda, \mathsf{D}_\lambda\}$ be an $\{\mathcal{L}_\lambda\}$-leakage-resilient encoding scheme with arbitrary polynomial message length. Assume the existence of an $\{\mathcal{L}_\lambda\}$-subset membership problem $\{\mathcal{YS}_\lambda, \mathcal{NS}_\lambda\}$ for a language family $\{\mathcal{V}_\lambda\}$ in NP with associated relation $\{\mathsf{R}_\lambda\}$. Let $\{\mathsf{F}_\lambda\}$ be the function family defined by $\mathsf{F}_\lambda(\mathbf{x}, \mathsf{ct}) = \mathsf{R}_\lambda(\mathbf{x}, \mathsf{D}_\lambda(\mathsf{ct}))$. Then there is no $\{\mathcal{L}_\lambda\}$-AI-LTC with public-coin setup for $\{\mathsf{F}_\lambda\}$.*

References

[1] Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Berlin, Heidelberg (Mar 2009) (Cited on page 1.)

[6] Indeed, the negative result holds even if the CRS generator is not in $\mathcal{C}_\lambda$.

[2] Ball, M., Dachman-Soled, D., Kulkarni, M.: New techniques for zero-knowledge: Leveraging inefficient provers to reduce assumptions, interaction, and trust. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part III. LNCS, vol. 12172, pp. 674–703. Springer, Cham (Aug 2020) (Cited on page 4, 21.)

[3] Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In: 18th ACM STOC. pp. 1–5. ACM Press (May 1986) (Cited on page 3.)

[4] Bellare, M., Fuchsbauer, G., Scafuro, A.: NIZKs with an untrusted CRS: Security in the face of parameter subversion. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 777–804. Springer, Berlin, Heidelberg (Dec 2016) (Cited on page 7.)

[5] Bogdanov, A., Dinesh, K., Filmus, Y., Ishai, Y., Kaplan, A., Srinivasan, A.: Bounded indistinguishability for simple sources. In: Braverman, M. (ed.) ITCS 2022. vol. 215, pp. 26:1–26:18. LIPIcs (Jan / Feb 2022) (Cited on page 2.)

[6] Bogdanov, A., Ishai, Y., Srinivasan, A.: Unconditionally secure computation against low-complexity leakage. Journal of Cryptology 34(4), 38 (Oct 2021) (Cited on page 2, 3, 6, 10.)

[7] Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: 51st FOCS. pp. 501–510. IEEE Computer Society Press (Oct 2010) (Cited on page 1.)

[8] Degwekar, A., Vaikuntanathan, V., Vasudevan, P.N.: Fine-grained cryptography. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 533–562. Springer, Berlin, Heidelberg (Aug 2016) (Cited on page 6, 8, 9.)

[9] Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. Journal of Cryptology 32(1), 151–177 (Jan 2019) (Cited on page 2.)

[10] Egashira, S., Wang, Y., Tanaka, K.: Fine-grained cryptography revisited. Journal of Cryptology 34(3), 23 (Jul 2021) (Cited on page 9.)

[11] Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting circuits from computationally bounded and noisy leakage. SIAM J. Comput. 43(5), 1564–1614 (2014) (Cited on page 2, 3, 5, 6.)

[12] Gál, A., Wigderson, A.: Boolean complexity classes vs. their arithmetic analogs. Random Struct. Algorithms 9(1-2), 99–111 (1996) (Cited on page 3.)

[13] Genkin, D., Ishai, Y., Weiss, M.: How to construct a leakage-resilient (stateless) trusted party. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 209–244. Springer, Cham (Nov 2017) (Cited on page 2.)

[14] Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Berlin, Heidelberg (Aug 2003) (Cited on page 1, 2, 10.)

[15] Ishai, Y., Song, Y.: Leakage-tolerant circuits. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part IV. LNCS, vol. 14654, pp. 196–225. Springer, Cham (May 2024) (Cited on page 2, 3, 4, 6.)

[16] Ishai, Y., Weiss, M., Yang, G.: Making the best of a leaky situation: Zero-knowledge PCPs from leakage-resilient circuits. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part II. LNCS, vol. 9563, pp. 3–32. Springer, Berlin, Heidelberg (Jan 2016) (Cited on page 2.)

[17] Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Berlin, Heidelberg (Feb 2004) (Cited on page 2.)

[18] Miles, E.: Iterated group products and leakage resilience against NC1. In: Naor, M. (ed.) ITCS 2014. pp. 261–268. ACM (Jan 2014) (Cited on page 2.)

[19] Miles, E., Viola, E.: Shielding circuits with groups. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 251–260. ACM Press (Jun 2013) (Cited on page 2, 3, 5, 14, 19.)

[20] Razborov, A.A.: Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. Mathematical notes of the Academy of Sciences of the USSR 41(4) (Apr 1987) (Cited on page 8.)

[21] Rothblum, G.N.: How to compute under AC^0 leakage without secure hardware. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 552–569. Springer, Berlin, Heidelberg (Aug 2012) (Cited on page 2, 3.)

[22] Ruzzo, W.L.: On uniform circuit complexity. J. Comput. Syst. Sci. 22(3), 365–383 (1981) (Cited on page 3.)

[23] Smolensky, R.: Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In: Aho, A. (ed.) 19th ACM STOC. pp. 77–82. ACM Press (May 1987) (Cited on page 8.)

[24] Wang, Y.: Secure computation against NC1 leakage without secure hardware. Cryptology ePrint Archive, Report 2026/1217 (2026), https://eprint.iacr.org/2026/1217 (Cited on page 4.)

[25] Wang, Y., Pan, J.: Non-interactive zero-knowledge proofs with fine-grained security. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 305–335. Springer, Cham (May / Jun 2022) (Cited on page 4, 5, 9.)

A Proof of Theorem 5.3

In this section, we give the proof of Theorem 5.3.

Proof. Correctness follows immediately from Lemma 4.1. Below we prove the 2-adaptive leakage-resilience and composability of Encode.

2-adaptive $\{\mathcal{L}_\lambda\}$-leakage-resilience. We now prove the following lemma, and then proceed with the proof of leakage-resilience based on it.

Lemma A.1 *If $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$, then for any $\lambda \in \mathbb{N}$, any constant $c \in \mathbb{N}$, any $\ell_\lambda : \{0,1\}^{\lambda\times\lambda} \to \{0,1\}^{c\log\lambda} \in \mathcal{L}_\lambda$, and any function $\mathcal{A} : \{0,1\}^{c\log\lambda} \to \mathcal{L}_\lambda$, the following two distributions are statistically close with negligible distance:*

$$\big(\ell_\lambda(\mathbf{M}), \ell'_\lambda(\mathbf{M})\big) : \mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda, \ell'_\lambda = \mathcal{A}(\ell_\lambda(\mathbf{M})),$$

$$\big(\ell_\lambda(\mathbf{M}), \ell'_\lambda(\mathbf{M})\big) : \mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda, \ell'_\lambda = \mathcal{A}(\ell_\lambda(\mathbf{M})).$$

Proof. Let $\ell'^{(1)}_\lambda, \cdots, \ell'^{(\lambda^c)}_\lambda \in \mathcal{L}_\lambda$ be all circuits that $\mathcal{A}(\cdot)$ outputs on input $1, \cdots, \lambda^c$ respectively. We construct a function $\ell''_\lambda : \{0,1\}^{\lambda\times\lambda} \to \{0,1\}^{2c\log\lambda}$ as follows.

ℓ''_λ on input $\mathbf{M}$, where $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$ or $\mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda$, runs $\ell_\lambda(\mathbf{M})$. Then it selects $\ell'^{(\ell_\lambda(\mathbf{M}))}_\lambda \in \{\ell'^{(i)}_\lambda\}_{i\in[\lambda^c]}$ (identifying $\ell_\lambda(\mathbf{M})$ with the natural number in $[\lambda^c]$ it represents), where the selection procedure can be performed with polynomial size and constant depth in parallel. Finally it outputs $(\ell_\lambda(\mathbf{M}), \ell'^{(\ell_\lambda(\mathbf{M}))}_\lambda(\mathbf{M}))$.

One can see that ℓ''_λ is computable in NC^1, and thus we have

$$\Delta\big(\ell''_\lambda(\mathbf{M}), \ell''_\lambda(\mathbf{M}')\big) \le \mathsf{negl}(\lambda)$$

by Lemma 3.1. As a result, the advantage of $(\ell_\lambda, \mathcal{A})$ in distinguishing the two distributions in Lemma A.1 is also negligible, completing the proof. □

We now proceed with the proof using Lemma A.1.

Let $\mathbf{m}_0$ and $\mathbf{m}_1$ be any bit strings in $\{0,1\}^{l_\mathsf{m}}$. Then for all sufficiently large $\lambda \in \mathbb{N}$, any $\ell_\lambda \in \mathcal{L}_\lambda$, and any $\mathcal{A}$ with its range being $\mathcal{L}_\lambda$, the following distributions are statistically close with negligible distance:

$$\big(\ell_\lambda(\mathbf{M},\mathbf{C}), \ell'_\lambda(\mathbf{M},\mathbf{C})\big) : \mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda, \mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times l_\mathsf{m}}, \mathbf{C} = \mathbf{MR} + \mathbf{e}_\lambda\mathbf{m}_0^\top, \ell'_\lambda = \mathcal{A}(\ell_\lambda(\mathbf{M},\mathbf{C})),$$

$$\big(\ell_\lambda(\mathbf{M},\mathbf{C}), \ell'_\lambda(\mathbf{M},\mathbf{C})\big) : \mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda, \mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times l_\mathsf{m}}, \mathbf{C} = \mathbf{MR} + \mathbf{e}_\lambda\mathbf{m}_0^\top, \ell'_\lambda = \mathcal{A}(\ell_\lambda(\mathbf{M},\mathbf{C})).$$

This follows from Lemma A.1 and the fact that computing $\mathbf{MR} + \mathbf{e}_\lambda\mathbf{m}_0^\top$ for either $\mathbf{M}^\top \in \mathcal{ZS}_\lambda$ or $\mathbf{M}^\top \in \mathcal{OS}_\lambda$ can be performed in NC^1.

Similarly, for all sufficiently large $\lambda \in \mathbb{N}$, the following distributions are also statistically close with a negligible distance:

$$\big(\ell_\lambda(\mathbf{M},\mathbf{C}), \ell'_\lambda(\mathbf{M},\mathbf{C})\big) : \mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda, \mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times l_\mathsf{m}}, \mathbf{C} = \mathbf{MR} + \mathbf{e}_\lambda\mathbf{m}_1^\top, \ell'_\lambda = \mathcal{A}(\ell_\lambda(\mathbf{M},\mathbf{C})),$$

$$\big(\ell_\lambda(\mathbf{M},\mathbf{C}), \ell'_\lambda(\mathbf{M},\mathbf{C})\big) : \mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda, \mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times l_\mathsf{m}}, \mathbf{C} = \mathbf{MR} + \mathbf{e}_\lambda\mathbf{m}_1^\top, \ell'_\lambda = \mathcal{A}(\ell_\lambda(\mathbf{M},\mathbf{C})).$$

Moreover, for $\mathbf{M}^\top \in \mathcal{OS}_\lambda$ (i.e., when $\mathbf{M}$ is of full rank) and $\mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times l_\mathsf{m}}$, the following distributions are identical:

$$\mathbf{MR} + \mathbf{e}_\lambda\mathbf{m}_0^\top \text{ and } \mathbf{MR} + \mathbf{e}_\lambda\mathbf{m}_1^\top,$$

since both are uniformly distributed in $\{0,1\}^{\lambda\times l_\mathsf{m}}$. Then combining all the above together, the following distributions are statistically close with a negligible distance:

$$\big(\ell_\lambda(\mathbf{M},\mathbf{C}), \ell'_\lambda(\mathbf{M},\mathbf{C})\big) : \mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda, \mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times l_\mathsf{m}}, \mathbf{C} = \mathbf{MR} + \mathbf{e}_\lambda\mathbf{m}_0^\top, \ell'_\lambda = \mathcal{A}(\ell_\lambda(\mathbf{M},\mathbf{C})),$$

$$\big(\ell_\lambda(\mathbf{M},\mathbf{C}), \ell'_\lambda(\mathbf{M},\mathbf{C})\big) : \mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda, \mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times l_\mathsf{m}}, \mathbf{C} = \mathbf{MR} + \mathbf{e}_\lambda\mathbf{m}_1^\top, \ell'_\lambda = \mathcal{A}(\ell_\lambda(\mathbf{M},\mathbf{C})).$$

i.e., the 2-adaptive leakage-resilience of Encode holds under $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$.

l_r-composability. We decompose $\{\mathsf{E}_\lambda\}$ into $\{\mathsf{E}^1_\lambda\}$ and $\{\mathsf{E}^2_\lambda\}$ and define associated public-coin distribution $\mathcal{R}_{\lambda,l_\mathsf{r}}$, as in Figure 18.

$\underline{\mathsf{E}^1_\lambda(\mathbf{m} \in \{0,1\}^{l_\mathsf{m}'}):}$	$\underline{\mathsf{E}^2_\lambda(\mathsf{ct} = (\mathbf{M}, \mathbf{C}_1), \mathbf{r} \in \{0,1\}^{l_\mathsf{r}}):}$	$\underline{\mathcal{R}_{\lambda,l_\mathsf{r}}:}$
Sample $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$ and $\mathbf{R}_1 \xleftarrow{\$} \{0,1\}^{\lambda \times l_\mathsf{m}'}$	Sample $\mathbf{R}_2 \xleftarrow{\$} \{0,1\}^{\lambda \times l_\mathsf{r}}$	Sample $\mathbf{C}_2 \xleftarrow{\$} \{0,1\}^{\lambda \times l_\mathsf{r}}$
Compute $\mathbf{C}_1 = \mathbf{M}\mathbf{R}_1 + \mathbf{e}_\lambda \mathbf{m}^\top$	Compute $\mathbf{C}_2 = \mathbf{M}\mathbf{R}_2 + \mathbf{e}_\lambda \mathbf{r}^\top$	
Return $\mathsf{ct} = (\mathbf{M}, \mathbf{C}_1)$	Return $\mathsf{ct}_\mathsf{rand} = \mathbf{C}_2$	

Figure 18: Definition of $\{\mathsf{E}^1_\lambda, \mathsf{E}^2_\lambda\}$. By $l_\mathsf{m}{}'$ we denote $l_\mathsf{m} - l_\mathsf{r}$.

It is straightforward that for any $\mathbf{m} \in \{0,1\}^{l_\mathsf{m}'}$ and $\mathbf{r} \in \{0,1\}^{l_\mathsf{r}}$, we have $\mathsf{E}_\lambda(\mathbf{m}, \mathbf{r}) = \mathsf{E}^1_\lambda(\mathbf{m}) || \mathsf{E}^2_\lambda(\mathsf{ct}, \mathbf{r})$. Moreover, since $(\mathbf{M}||\mathbf{e}_\lambda)$ is of rank λ, the distribution of

$$\mathbf{M}\mathbf{R}_2 + \mathbf{e}_\lambda \mathbf{r}^\top = (\mathbf{M}||\mathbf{e}_\lambda) \begin{pmatrix} \mathbf{R}_2 \\ \mathbf{r}^\top \end{pmatrix} \quad \text{where } \mathbf{R}_2 \xleftarrow{\$} \{0,1\}^{\lambda \times l_\mathsf{r}}, \mathbf{r} \xleftarrow{\$} \{0,1\}^{l_\mathsf{r}}$$

is uniform in $\{0,1\}^{\lambda \times l_\mathsf{r}}$, and thus identical to the public-coin distribution $\mathcal{R}_{\lambda,l_\mathsf{r}}$, completing the proof of composability.

Putting all the above together, Theorem 5.3 immediately follows. □

B Proof of Theorem 5.4

In this section, we give the proof of Theorem 5.4.

Proof. Correctness follows immediately from the correctness of AI-LTC and the correctness and composability of Encode.

Let $\mathcal{A}$ be any adversary, q be the total number of invocations, $\{\mathcal{D}_\lambda\}$ be the distribution family over the simulators of AI-LTC. We give the description of the simulator $\{\mathsf{Sim}_\lambda\}$ in Figure 19, and prove the leakage-resilience of SF-LRC via a sequence of intermediate games, where in each game, the assignment $\mathcal{W}$ to all wires in each invocation is generated as in Figure 20.

$\underline{\mathsf{Sim}_\lambda(\mathbf{x}, \mathbf{y}):}$
Setup:

Compute $\mathsf{ct} \xleftarrow{\$} \mathsf{E}^1_\lambda(\mathbf{0})$

Set the initial encoded state as $\mathsf{st} = \mathsf{ct}$

Computation:

Run $\mathsf{ct}_\mathsf{rand} \xleftarrow{\$} \mathsf{E}^2_\lambda(\mathsf{ct}, \mathbf{0})$, $\mathsf{ct}' \xleftarrow{\$} \mathsf{E}^1_\lambda(\mathbf{0})$, and $\mathsf{LTSim}_\lambda \xleftarrow{\$} \mathcal{D}_\lambda$

Update the encoded state as $\mathsf{st}' = \mathsf{ct}'$ and return $\mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathsf{ct}, \mathsf{ct}_\mathsf{rand}, \mathbf{x}), (\mathsf{ct}', \mathbf{y}))$

Figure 19: Definition of $\{\mathsf{Sim}_\lambda\}$. By **Setup** and **Computation**, we refer to the simulations for the output of the setup and the wire assignments during the computation phase, respectively.

Game G_0. G_0 is the real game $\mathsf{Real}_\mathcal{A}$ (see Figure 3), where in each invocation, $\mathcal{A}$ obtains the current external output and leakage on the honestly generated assignment to wires of the LRC, and returns a new pair of external input and leakage function for the next invocation.

Game G_1. G_1 differs from G_0 in that $\mathsf{ct}_\mathsf{rand}$ is generated as $\mathsf{ct}_\mathsf{rand} \xleftarrow{\$} \mathsf{E}^2_\lambda(\mathsf{ct}, \mathbf{r})$ where $\mathbf{r} \xleftarrow{\$} \{0,1\}^{l_\mathsf{r}}$, instead of $\mathsf{ct}_\mathsf{rand} \xleftarrow{\$} \mathcal{R}_{\lambda,l_\mathsf{r}}$, in each invocation.

Lemma B.1 *The views of $\mathcal{A}$ in G_0 and G_1 are identical.*

Lemma B.1 follows immediately from (the second condition of) the composability of Encode (see Definition 5.2).

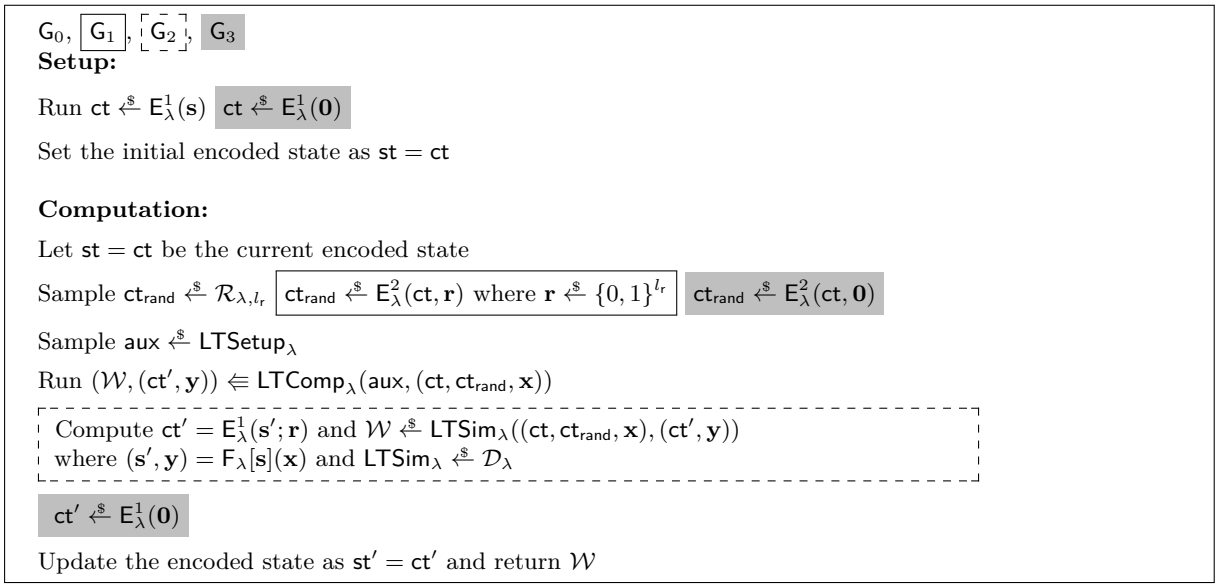


Figure 20: Generation of assignments to all wires in the intermediate games.

Game G_2. G_2 differs from G_1 in that $\mathcal{W}$ is now generated as

$$\mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathsf{ct}, \mathsf{ct}_{\mathsf{rand}}, \mathbf{x}), (\mathsf{ct}', \mathbf{y})),$$

where $(\mathbf{s}', \mathbf{y}) = \mathsf{F}_\lambda[\mathbf{s}](\mathbf{x})$, $\mathsf{ct}' = \mathsf{E}^1_\lambda(\mathbf{s}'; \mathbf{r})$, and $\mathsf{LTSim}_\lambda \xleftarrow{\$} \mathcal{D}_\lambda$ in each invocation.

Lemma B.2 *The views of $\mathcal{A}$ in G_1 and G_2 are statistically close with negligible distance.*

Proof. We proceed with the proof by defining a sequence of intermediate games $\mathsf{G}_{1.0}, \cdots, \mathsf{G}_{1.q}$, where $\mathsf{G}_{1.0}$ and $\mathsf{G}_{1.q}$ are the same as G_1 and G_2 respectively and $\mathsf{G}_{1.\mathsf{c}}$ differs from $\mathsf{G}_{1.\mathsf{c}-1}$ in that $\mathcal{W}$ is sampled by LTSim_λ for $\mathsf{LTSim}_\lambda \xleftarrow{\$} \mathcal{D}_\lambda$ in the cth invocation. We now prove that the views of $\mathcal{A}$ in $\mathsf{G}_{1.\mathsf{c}-1}$ and $\mathsf{G}_{1.\mathsf{c}}$ are statistically close with negligible distance, for each $\mathsf{c} \in [q]$.

In $\mathsf{G}_{1.\mathsf{c}-1}$, we first fix all the seeds, except for the internal randomness used by $\mathsf{LTSetup}_\lambda$ and LTComp_λ in the cth invocation, such that the adversary's distinguishing advantage is maximized conditioned on this fixing. Here we can see that the randomness in the cth invocation is independent of the assignment to wires in the first $\mathsf{c}-1$ invocations and the invocations after the cth. Also, we can see that the leakage (with fixed seeds) in the cth invocation reveals no additional information on $\mathcal{W}$ except for leakage in $\mathcal{L}_\lambda$. Then due to the $\{\mathcal{L}_\lambda\}$-leakage-tolerance of $\mathsf{AI\text{-}LTC}$, the view of the adversary is statistically close to its view in $\mathsf{G}_{1.\mathsf{c}-1}$ with negligible distance, when we generate $\mathcal{W}$ as $\mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathsf{ct}, \mathsf{ct}_{\mathsf{rand}}, \mathbf{x}), (\mathsf{ct}', \mathbf{y}))$ for fixed $\Big((\mathsf{ct}, \mathsf{ct}_{\mathsf{rand}}, \mathbf{x}), (\mathsf{ct}', \mathbf{y})\Big)$ instead, completing the proof of Lemma B.2. □

Game G_3. G_3 differs from G_2 in that ct and $\mathsf{ct}_{\mathsf{rand}}$ are generated as $\mathsf{ct} \xleftarrow{\$} \mathsf{E}^1_\lambda(\mathbf{0})$ and $\mathsf{ct}_{\mathsf{rand}} \xleftarrow{\$} \mathsf{E}^2_\lambda(\mathsf{ct}, \mathbf{0})$ respectively in the setup and in each invocation.

Lemma B.3 *The views of $\mathcal{A}$ in G_2 and G_3 are statistically close with negligible distance.*

Proof. We proceed with the proof by defining a sequence of intermediate games $\mathsf{G}_{2.0}, \cdots, \mathsf{G}_{2.q}$, where $\mathsf{G}_{2.0}$ is the same as G_2. $\mathsf{G}_{2.\mathsf{c}}$ differs from $\mathsf{G}_{2.\mathsf{c}-1}$ in that the concatenation $\mathsf{ct}||\mathsf{ct}_{\mathsf{rand}}$ in the cth invocation is sampled as $\mathsf{ct}||\mathsf{ct}_{\mathsf{rand}} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{0})$. Here we note that when $\mathsf{c} = 1$, ct is generated in the setup, and when $\mathsf{c} > 1$, it is generated in the $(\mathsf{c}-1)$th invocation as ct'. We now prove that the views of $\mathcal{A}$ in $\mathsf{G}_{2.\mathsf{c}-1}$ and $\mathsf{G}_{2.\mathsf{c}}$ are statistically close with negligible distance, for each $\mathsf{c} \in [q]$.

In $\mathsf{G}_{2.\mathsf{c}-1}$, we first fix all the seeds, except for those used by E^1_λ to generate ct' in the $(\mathsf{c}-1)$th invocation (i.e., ct in the encoded state of the cth invocation) and those used by E^2_λ to generate $\mathsf{ct}_{\mathsf{rand}}$ in the cth invocation, such that the adversary's distinguishing advantage is maximized conditioned on this fixing. Here we can see that $\mathsf{ct}||\mathsf{ct}_{\mathsf{rand}}$ in the cth invocation is an encoding of (the fixed) $\mathbf{s}||\mathbf{r}$, which

is independent of the assignment to wires in the first $(\mathsf{c}-2)$ invocations and the invocations after the cth. Also, we can see that the leakage (with fixed seeds) in the $(\mathsf{c}-1)$th and cth invocations reveals no additional information on $\mathsf{ct}||\mathsf{ct}_{\mathsf{rand}}$ except for leakage in $\mathcal{L}_\lambda$. Due to the 2-adaptive $\{\mathcal{L}_\lambda\}$-leakage-resilience of Encode, for every $\mathsf{c} \in [q]$, the view of the adversary in $\mathsf{G}_{2.\mathsf{c}}$ is statistically close to its view in $\mathsf{G}_{2.\mathsf{c}-1}$ with negligible distance when the encoding in the cth invocation is replaced by an encoding of $\mathbf{0}$.

Finally, applying the 2-adaptive $\{\mathcal{L}_\lambda\}$-leakage-resilience of Encode once more replaces the final encoding generated in the qth invocation by an encoding of $\mathbf{0}$, yielding G_3. Applying a hybrid argument completes the proof of Lemma B.3. □

One can see that the view of $\mathcal{A}$ in G_3 is identical to its view in the ideal game $\mathsf{Ideal}_{\mathcal{A}}$ (see Figure 3), where in each invocation, $\mathcal{A}$ obtains the current external output and leakage on the assignment to wires of the LRC generated by the simulator Sim_λ, and returns a new pair of external input and leakage function for the next invocation, completing the proof of Theorem 5.4. □

C Proof of Theorem 5.5

In this section, we give the proof of Theorem 5.5.

Proof. **Correctness.** We first show the correctness of the refreshed state.

Since $\mathbf{M}||\mathbf{e}_\lambda$ is of rank λ, we have that $(\mathbf{M}\otimes\mathbf{I}_\lambda||\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)$ is of rank λ^2. Then in the state update, for $i \in [l_{\mathsf{st}}]$, we must have

$$\mathbf{P}_i = (\mathbf{M}\otimes\mathbf{I}_\lambda)\mathbf{H}_i + (\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)\mathbf{Q}_i'^{\top}$$

for some $\mathbf{H}_i \in \{0,1\}^{\lambda^2\times\lambda}$ and $\mathbf{Q}_i' \in \{0,1\}^{\lambda\times\lambda}$. Moreover, since $\mathbf{s}_i = \mathbf{M}\mathbf{r}_i + \mathbf{e}_\lambda\cdot s_i$ for some $\mathbf{r}_i \in \{0,1\}^\lambda$ and $\mathbf{U}_i = (\mathbf{M}\otimes\mathbf{I}_\lambda)\mathbf{V}_i$ for some $\mathbf{V}_i \in \{0,1\}^{\lambda^2\times\lambda}$, we have

$$\begin{aligned}\mathbf{D}_i =& \mathbf{P}_i\mathbf{M}'^{\top} + \mathbf{U}_i + \mathbf{s}_i\otimes\mathbf{I}_\lambda\\ =&(\mathbf{M}\otimes\mathbf{I}_\lambda)(\mathbf{H}_i\mathbf{M}'^{\top}+\mathbf{V}_i) + (\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)\mathbf{Q}_i'^{\top}\mathbf{M}'^{\top} + (\mathbf{M}\mathbf{r}_i+\mathbf{e}_\lambda\cdot s_i)\otimes\mathbf{I}_\lambda\\ =&(\mathbf{M}\otimes\mathbf{I}_\lambda)(\mathbf{H}_i\mathbf{M}'^{\top}+\mathbf{V}_i+\mathbf{r}_i\otimes\mathbf{I}_\lambda) + (\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)(\mathbf{Q}_i'^{\top}\mathbf{M}'^{\top}+s_i\otimes\mathbf{I}_\lambda).\end{aligned}$$

Therefore, we have

$$\mathbf{T}_i = \mathsf{Dec}_\lambda(\mathbf{M},\mathbf{D}_i)^{\top} = \mathbf{M}'\mathbf{Q}_i' + s_i\otimes\mathbf{I}_\lambda,$$

according to Lemma 4.1. Namely, the decoding result is a refreshed encoding of the current state with respect to the public parameter $\mathbf{M}'$.

We now proceed to show the correctness of the output in each invocation.

As shown in the proof of Lemma 3.2, Add_λ (respectively, Mult_λ) can correctly convert wire bundles of the form

$$\mathbf{W}_{d_1} = \mathbf{M}'\mathbf{Z}_{d_1} + w_{d_1}\otimes\mathbf{I}_\lambda \text{ and } \mathbf{W}_{d_2} = \mathbf{M}'\mathbf{Z}_{d_2} + w_{d_2}\otimes\mathbf{I}_\lambda$$

into ones of the form

$$\mathbf{W}_{d_3} = \mathbf{M}'\mathbf{Z}_{d_3} + (w_{d_1}+w_{d_2})\otimes\mathbf{I}_\lambda \text{ (respectively, } \mathbf{W}_{d_3} = \mathbf{M}'\mathbf{Z}_{d_3} + w_{d_1}w_{d_2}\otimes\mathbf{I}_\lambda).$$

Therefore, in each invocation, assuming that y_i is the ith bit of the external output and s_i' is the ith bit of the updated state, for each $i \in [l_{\mathsf{out}}]$, we have $\mathbf{Y}_i = \mathbf{M}'\mathbf{Z}_i + y_i\otimes\mathbf{I}_\lambda$ for some $\mathbf{Z}_i \in \{0,1\}^{\lambda\times\lambda}$, and for each $i \in [l_{\mathsf{st}}]$, we have $\mathbf{S}_i' = \mathbf{M}'\mathbf{R}_i + s_i'\otimes\mathbf{I}_\lambda$ for some $\mathbf{R}_i \in \{0,1\}^{\lambda\times\lambda}$. Namely, we have $\mathbf{y}_i = \mathbf{M}'\mathbf{z}_i + \mathbf{e}_\lambda\cdot y_i$ for some $\mathbf{z}_i \in \{0,1\}^\lambda$, since $\mathbf{u}_i' \in \mathsf{Span}(\mathbf{M}')$, and $\mathbf{s}_i' = \mathbf{M}'\mathbf{r}_i + \mathbf{e}_\lambda\cdot s_i'$ for some $\mathbf{r}_i \in \{0,1\}^\lambda$. Then for $i \in [l_{\mathsf{out}}]$, we must have $y_i = \mathsf{Dec}_\lambda(\mathbf{M}',\mathbf{y}_i)$ by Lemma 4.1, completing the proof of correctness.

Leakage-resilience. We first extend Lemma A.1 to treat 3-adaptive leakage, and then proceed with the proof based on it. [7]

[7] Indeed, Lemma A.1 can be extended to treat c-adaptive adversaries for any constant c.

Lemma C.1 *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$*, then for any* $\lambda \in \mathbb{N}$*, any constant* $c \in \mathbb{N}$*, any* $\ell_\lambda : \{0,1\}^{\lambda\times\lambda} \to \{0,1\}^{c\log\lambda} \in \mathcal{L}_\lambda$*, and any function* $\mathcal{A} : \{0,1\}^{c\log\lambda} \to \mathcal{L}_\lambda$*, the following two distributions are statistically close with negligible distance:*

$$\big(\ell_\lambda(\mathbf{M}), \ell'^1_\lambda(\mathbf{M}), \ell'^2_\lambda(\mathbf{M})\big) : \mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda, \ell'^1_\lambda = \mathcal{A}(\ell_\lambda(\mathbf{M})), \ell'^2_\lambda = \mathcal{A}(\ell'^1_\lambda(\mathbf{M})),$$

$$\big(\ell_\lambda(\mathbf{M}), \ell'^1_\lambda(\mathbf{M}), \ell'^2_\lambda(\mathbf{M})\big) : \mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda, \ell'^1_\lambda = \mathcal{A}(\ell_\lambda(\mathbf{M})), \ell'^2_\lambda = \mathcal{A}(\ell'^1_\lambda(\mathbf{M})).$$

Proof. Let $\ell'^{(1)}_\lambda, \cdots, \ell'^{(\lambda^c)}_\lambda \in \mathcal{L}_\lambda$ be all circuits that $\mathcal{A}(\cdot)$ outputs on input $1, \cdots, \lambda^c$ respectively. We construct a function $\ell''_\lambda : \{0,1\}^{\lambda\times\lambda} \to \{0,1\}^{3c\log\lambda}$ as follows.

ℓ''_λ on input $\mathbf{M}$, where $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$ or $\mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda$, runs $\alpha = \ell_\lambda(\mathbf{M})$. Then it selects $\ell'^{(\alpha)}_\lambda \in \{\ell'^{(i)}_\lambda\}_{i\in[\lambda^c]}$ (identifying $\ell_\lambda(\mathbf{M})$ with the natural number in $[\lambda^c]$ it represents), where the selection procedure can be performed with polynomial size and constant depth in parallel. It then lets $\beta = \ell'^{(\alpha)}_\lambda(\mathbf{M})$ and selects $\ell'^{(\beta)}_\lambda \in \{\ell'^{(i)}_\lambda\}_{i\in[\lambda^c]}$. Finally it outputs $(\ell_\lambda(\mathbf{M}), \ell'^{(\alpha)}_\lambda(\mathbf{M}), \ell'^{(\beta)}_\lambda(\mathbf{M}))$.

One can see that ℓ''_λ is computable in NC^1, and thus we have

$$\Delta\big(\ell''_\lambda(\mathbf{M}), \ell''_\lambda(\mathbf{M}')\big) \le \mathsf{negl}(\lambda),$$

according to Lemma 3.1. As a result, the advantage of $(\ell_\lambda, \mathcal{A})$ in distinguishing the two distributions in Lemma C.1 is also negligible, completing the proof. □

Let λ be the security parameter and $\mathcal{A}$ be any adversary and q be the total number of invocations. We give the description of the simulator Sim^*_λ in Figure 21, and prove the leakage-resilience by defining a sequence of intermediate games $\mathsf{G}_0, \cdots, \mathsf{G}_8$. In $\mathsf{G}_1, \cdots, \mathsf{G}_7$, the assignments to all wires in each invocation are generated as in Figure 22.

Game G_0**.** G_0 is the real game $\mathsf{Real}_\mathcal{A}$ (see Figure 3), where in each invocation, $\mathcal{A}$ obtains the current external output and leakage on the honestly generated assignment to wires of the stateful LRC, and returns a new pair of external input and leakage function for the next invocation.

Game G_1**.** G_1 differs from G_0 in that for each invocation, the challenger extends the external input $\mathbf{x}$ to $\mathbf{w}$ containing the assignment to all wires in $\mathsf{F}_\lambda[\mathbf{s}](\mathbf{x})$ and simulates the assignment for each gate gadget in parallel, in each invocation.

Lemma C.2 *The views of* $\mathcal{A}$ *in* G_0 *and* G_1 *are identical.*

Lemma C.2 follows immediately from Lemma 3.2.

Game G_2**.** G_2 is the same as G_1 except that $\mathbf{u}'_i$ is generated as $\mathbf{u}'_i = \mathbf{y}_i - \mathbf{y}'_i$ where $\mathbf{y}_i = \mathbf{M}'\mathbf{z}_i + \mathbf{e}_\lambda \cdot y_i$ for $\mathbf{z}_i \xleftarrow{\$} \{0,1\}^\lambda$.

Lemma C.3 *The views of* $\mathcal{A}$ *in* G_1 *and* G_2 *are identical.*

Proof. In G_1, for each output gate y_i, we have

$$\mathbf{y}'_i = \mathbf{M}'\mathbf{z}'_i + \mathbf{e}_\lambda \cdot y_i \text{ and } \mathbf{y}_i = \mathbf{y}'_i + \mathbf{u}'_i = \mathbf{M}'(\mathbf{z}'_i + \mathbf{v}'_i) + \mathbf{e}_\lambda \cdot y_i,$$

where $\mathbf{z}'_i \in \{0,1\}^\lambda$ and $\mathbf{u}'_i = \mathbf{M}'\mathbf{v}'_i$ for $\mathbf{v}'_i \xleftarrow{\$} \{0,1\}^\lambda$. Hence, the distributions of

$$(\mathbf{y}_i, \mathbf{u}'_i) \text{ and } (\mathbf{M}'\mathbf{z}_i + \mathbf{e}_\lambda \cdot y_i, \mathbf{y}_i - \mathbf{y}'_i)$$

where $\mathbf{z}_i \xleftarrow{\$} \{0,1\}^\lambda$ are identical, completing the proof of Lemma C.3. □

Game G_3**.** G_3 is the same as G_2 except that $\mathcal{W}_{y_i}$ is generated as $\mathcal{W}_{y_i} = \mathsf{SimDec}_\lambda(\mathbf{M}', \mathbf{z}_i, y_i)$ instead of $\mathcal{W}_{y_i} \leftarrow \mathsf{W}[\mathsf{Dec}_\lambda(\mathbf{M}', \mathbf{y}_i)]$ for all $i \in [l_{\mathsf{out}}]$.

Lemma C.4 *The views of* $\mathcal{A}$ *in* G_2 *and* G_3 *are identical.*

Lemma C.4 follows immediately from Lemma 4.2.

Game G_4**.** In G_4, in the state updating procedure of each invocation, $\mathbf{P}_i$ is generated as $\mathbf{P}_i = (\mathbf{M} \otimes \mathbf{I}_\lambda)\mathbf{H}_i + (\mathbf{e}_\lambda \otimes \mathbf{I}_\lambda) \cdot \mathbf{Q}'^\top_i$, where $\mathbf{H}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$ and $\mathbf{Q}'_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$, instead of $\mathbf{P}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$.

$\underline{\mathsf{Sim}^*_\lambda(\mathbf{x},\mathbf{y}):}$
Setup:

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda$ and $\mathbf{S}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ for $i \in [l_{\mathsf{st}}]$

State Refresh:

Sample $\mathbf{M'}^\top \xleftarrow{\$} \mathcal{OS}_\lambda$ and set $\mathcal{W} = \emptyset$

For $i \in [l_{\mathsf{st}}]$:

- sample $\mathbf{P}_i, \mathbf{R}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$ and $\mathbf{T}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$,
- compute $\mathbf{D}_i = (\mathbf{M}\otimes\mathbf{I}_\lambda)\mathbf{R}_i + (\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)\mathbf{T}_i^\top$ and $\mathbf{U}_i = \mathbf{D}_i - \mathbf{P}_i\mathbf{M'}^\top - \mathbf{s}_i\otimes\mathbf{I}_\lambda$, where $\mathbf{s}_i$ is the rightmost column of the matrix $\mathbf{S}_i$ in the current state
- include $\mathsf{W}[\mathbf{P}_i\mathbf{M'}^\top + \mathbf{U}_i + \mathbf{s}_i\otimes\mathbf{I}_\lambda]$ and $\mathsf{SimDec}_\lambda(\mathbf{M},\mathbf{R}_i,\mathbf{T}_i^\top)$ in $\mathcal{W}$

Computation:

Represent the ith state gate in F_λ by $\mathbf{T}_i$ for $i \in [l_{\mathsf{st}}]$

For the ith input gate x_i in F_λ, run

$$(\mathcal{W}_{\mathbf{x}_i}, \mathbf{X}_i) \Leftarrow (\mathbf{M'}\mathbf{Z}_i + x_i\otimes\mathbf{I}_\lambda) \text{ where } \mathbf{Z}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$$

and include $\mathcal{W}_{\mathbf{x}_i}$ in $\mathcal{W}$

For each gate $w_{d_1} + w_{d_2}$ (respectively, $w_{d_1}w_{d_2}$) in F_λ, compute

$$(\mathcal{W}_i, \mathbf{W}_{d_3}) \Leftarrow \mathsf{W}[\mathsf{Add}_\lambda(\mathbf{M'}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2})] \text{ (respectively, } (\mathcal{W}_i, \mathbf{W}_{d_3}) \Leftarrow \mathsf{W}[\mathsf{Mult}_\lambda(\mathbf{M'}, \mathbf{W}_{d_1}, \mathbf{W}_{d_2})])$$

and include $\mathcal{W}_i$ in $\mathcal{W}$, where $(\mathbf{W}_{d_1}, \mathbf{W}_{d_2})$ are the wire bundles representing (w_{d_1}, w_{d_2})

Let $(\mathbf{Y}_i)_{i\in[l_{\mathsf{out}}]}$ and $(\mathbf{S}'_i)_{i\in[l_{\mathsf{st}}]}$ be the matrices representing the final output and updated state in F_λ respectively and $(\mathbf{y}'_i)_{i\in[l_{\mathsf{out}}]}$ be the rightmost columns of $(\mathbf{Y}_i)_{i\in[l_{\mathsf{out}}]}$

For $i \in [l_{\mathsf{out}}]$, run

$$\mathbf{z}_i \xleftarrow{\$} \{0,1\}^\lambda, \mathbf{y}_i = \mathbf{M'}\mathbf{z}_i + \mathbf{e}_\lambda\cdot y_i, \mathbf{u}'_i = \mathbf{y}_i - \mathbf{y}'_i$$

and include $(\mathbf{u}'_i, \mathsf{W}[\mathbf{y}'_i + \mathbf{u}'_i], \mathsf{SimDec}_\lambda(\mathbf{M'}, \mathbf{z}_i, y_i))$ in $\mathcal{W}$

Execute the state refresh phase of the next invocation in advance to obtain and store $\mathbf{U}'_i$ (i.e., $\mathbf{U}_i$ of the next invocation)

Set the private state for the next invocation as $\mathsf{st}' = (\mathbf{M'}, (\mathbf{S}'_i, \mathbf{U}'_i)_{i\in[l_{\mathsf{st}}]})$, and return $\mathcal{W}$

Figure 21: Definition of $\{\mathsf{Sim}^*_\lambda\}$. $\{\mathsf{SimDec}_\lambda\}$ is the function family defined as in Figure 8.

Lemma C.5 *The views of $\mathcal{A}$ in G_3 and G_4 are identical.*

Proof. Since $\mathbf{M}||\mathbf{e}_\lambda$ is of rank λ for $\mathbf{M}^\top \in \mathcal{ZS}_\lambda$, $\mathbf{M}\otimes\mathbf{I}_\lambda||\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda = (\mathbf{M}||\mathbf{e}_\lambda)\otimes\mathbf{I}_\lambda$ must be of rank λ^2. As a result, each $\mathbf{P}_i$ generated in G_4 is uniformly distributed in $\{0,1\}^{\lambda^2\times\lambda}$, completing the proof of Lemma C.5. □

Game G_5. G_5 differs from G_4 in the way that $\mathbf{U}_i$, $\mathbf{D}_i$, and $\mathbf{T}_i$ are generated for all i.

Lemma C.6 *The views of $\mathcal{A}$ in G_4 and G_5 are identical.*

Proof. According to the proof of correctness, we have $\mathbf{s}_i = \mathbf{M}\mathbf{r}_i + \mathbf{e}_\lambda\cdot s_i$ for some $\mathbf{r}_i$ in G_4. Let $\mathbf{U}_i = (\mathbf{M}\otimes\mathbf{I}_\lambda)\mathbf{V}_i$ where $\mathbf{V}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$. We have

$$\begin{aligned}
\mathbf{D}_i =& \mathbf{P}_i\mathbf{M'}^\top + \mathbf{U}_i + \mathbf{s}_i\otimes\mathbf{I}_\lambda \\
=& (\mathbf{M}\otimes\mathbf{I}_\lambda)(\mathbf{H}_i\mathbf{M'}^\top + \mathbf{V}_i) + (\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)\mathbf{Q}_i'^\top\mathbf{M'}^\top + (\mathbf{M}\mathbf{r}_i + \mathbf{e}_\lambda\cdot s_i)\otimes\mathbf{I}_\lambda \\
=& (\mathbf{M}\otimes\mathbf{I}_\lambda)(\mathbf{H}_i\mathbf{M'}^\top + \mathbf{V}_i + \mathbf{r}_i\otimes\mathbf{I}_\lambda) + (\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)(\mathbf{Q}_i'^\top\mathbf{M'}^\top + s_i\otimes\mathbf{I}_\lambda).
\end{aligned}$$

G_1, $\boxed{\mathsf{G}_2}$, $\boxed{\mathsf{G}_3}$, G_4, G_5 $\boxed{\mathsf{G}_6}$ $\boxed{\mathsf{G}_7}$

Setup:

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$ $\boxed{\mathbf{M}^\top \xleftarrow{\$} \mathcal{OS}_\lambda}$

Sample $\mathbf{Q}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ and compute $\mathbf{S}_i = \mathbf{M}\mathbf{Q}_i + s_i \otimes \mathbf{I}_\lambda$ for $i \in [l_{\mathsf{st}}]$

Sample $\mathbf{V}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$ and compute $\mathbf{U}_i = (\mathbf{M}\otimes\mathbf{I}_\lambda)\mathbf{V}_i$ for $i \in [l_{\mathsf{st}}]$

State Refresh:

Sample $\mathbf{M'}^\top \xleftarrow{\$} \mathcal{ZS}_\lambda$ $\boxed{\mathbf{M'}^\top \xleftarrow{\$} \mathcal{OS}_\lambda}$ and set $\mathcal{W} = \emptyset$

Sample $\mathbf{V}'_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$ and compute $\mathbf{U}'_i = (\mathbf{M}'\otimes\mathbf{I}_\lambda)\mathbf{V}'_i$ for $i \in [l_{\mathsf{st}}]$

Sample $\mathbf{v}'_i \xleftarrow{\$} \{0,1\}^{\lambda}$ and compute $\mathbf{u}'_i = \mathbf{M}'\mathbf{v}'_i$ for $i \in [l_{\mathsf{out}}]$

For $i \in [l_{\mathsf{st}}]$:

sample $\mathbf{P}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$ $\mathbf{H}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$, $\mathbf{Q}'_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$, $\mathbf{P}_i = (\mathbf{M}\otimes\mathbf{I}_\lambda)\mathbf{H}_i + (\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)\cdot\mathbf{Q}'^\top_i$

Compute $(\mathbf{D}_i, \mathcal{W}_{\mathbf{D}_i}) \Leftarrow \mathbf{P}_i\mathbf{M'}^\top + \mathbf{U}_i + \mathbf{s}_i\otimes\mathbf{I}_\lambda$ and $(\mathcal{W}_{\mathbf{T}_i}, \mathbf{T}_i^\top) \Leftarrow \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{D}_i)$

$\mathbf{T}_i = \mathbf{M}'\mathbf{Q}'_i + s_i\otimes\mathbf{I}_\lambda$, $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$, $\mathbf{D}_i = (\mathbf{M}\otimes\mathbf{I}_\lambda)\mathbf{R}_i + (\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)\mathbf{T}_i^\top$, $\mathbf{U}_i = \mathbf{D}_i - \mathbf{P}_i\mathbf{M'}^\top - \mathbf{s}_i\otimes\mathbf{I}_\lambda$

$\boxed{\mathcal{W}_{\mathbf{T}_i} = \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{R}_i, \mathbf{T}_i^\top)}$

Include $\mathcal{W}_{\mathbf{D}_i}$ and $\mathcal{W}_{\mathbf{T}_i}$ in $\mathcal{W}$

Computation:

Represent the ith state gate in F_λ by $\mathbf{T}_i$ in the current state

For the ith input gate x_i in F_λ, run $(\mathcal{W}_{\mathbf{x}_i}, \mathbf{X}_i) \Leftarrow (\mathbf{M}'\mathbf{Z}_i + x_i\otimes\mathbf{I}_\lambda)$ for $\mathbf{Z}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ and include $\mathcal{W}_{\mathbf{x}_i}$ in $\mathcal{W}$

For the ith wire in F_λ assigned with w_i, set $\mathbf{W}_i = w_i\otimes\mathbf{I}_\lambda + \mathbf{M}'\mathbf{Z}_i$ for $\mathbf{Z}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$

For each gate $w_{d_3} = w_{d_1} + w_{d_2}$, include $\mathsf{SimAdd}_\lambda(\mathbf{M}', \mathbf{W}_{d_1}, \mathbf{W}_{d_2}, \mathbf{Z}_{d_1}, \mathbf{Z}_{d_2}, \mathbf{Z}_{d_3})$ in $\mathcal{W}$

For each gate $w_{d_3} = w_{d_1}w_{d_2}$, include $\mathsf{SimMult}_\lambda(\mathbf{M}', \mathbf{W}_{d_1}, \mathbf{W}_{d_2}, \mathbf{Z}_{d_1}, \mathbf{Z}_{d_2}, \mathbf{Z}_{d_3}, w_{d_1})$ in $\mathcal{W}$

For $i \in [l_{\mathsf{out}}]$, let $\mathbf{Y}_i$ be the wire bundle representing the ith output in F_λ and $\mathbf{y}'_i$ be the rightmost column of $\mathbf{Y}_i$, run

$$\boxed{\mathbf{u}'_i = \mathbf{y}_i - \mathbf{y}'_i \text{ where } \mathbf{y}_i = \mathbf{M}'\mathbf{z}_i + \mathbf{e}_\lambda\cdot y_i \text{ for } \mathbf{z}_i \xleftarrow{\$} \{0,1\}^\lambda}$$

$$(\mathcal{W}_{\mathbf{y}'_i}, \mathbf{y}_i) \Leftarrow \mathbf{y}'_i + \mathbf{u}'_i, (\mathcal{W}_{y_i}, y_i) \Leftarrow \mathsf{Dec}_\lambda(\mathbf{M}', \mathbf{y}_i) \; \boxed{\mathcal{W}_{y_i} = \mathsf{SimDec}_\lambda(\mathbf{M}', \mathbf{z}_i, y_i)}$$

and include $(\mathbf{u}'_i, \mathsf{W}[\mathbf{y}'_i + \mathbf{u}'_i], \mathsf{SimDec}_\lambda(\mathbf{M}', \mathbf{z}_i, y_i))$ in $\mathcal{W}$

Set the encoded state as $\mathsf{st}' = (\mathbf{M}', (\mathbf{S}'_i, \mathbf{U}'_i)_{i\in[l_{\mathsf{st}}]})$ and return $\mathcal{W}$

Figure 22: Generation of assignments to all wires in the intermediate games $\mathsf{G}_2, \cdots, \mathsf{G}_7$. $\{\mathsf{SimAdd}_\lambda, \mathsf{SimMult}_\lambda\}$ are function families defined as in Figure 4.

Since the distribution of $\mathbf{H}_i\mathbf{M'}^\top + \mathbf{V}_i + \mathbf{r}_i\otimes\mathbf{I}_\lambda$ is identical to that of $\mathbf{R}_i$ for $\mathbf{R}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$, the joint distributions of $(\mathbf{U}_i, \mathbf{D}_i, \mathbf{T}_i)$ in G_4 and G_5 are identical as well. □

Game G_6. G_6 differs from G_5 in that $\mathcal{W}_{\mathbf{T}_i}$ is generated as $\mathcal{W}_{\mathbf{T}_i} = \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{R}_i, \mathbf{T}_i^\top)$ instead of $\mathcal{W}_{\mathbf{T}_i} \xleftarrow{\$} \mathsf{W}[\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{D}_i)]$.

Lemma C.7 *The views of $\mathcal{A}$ in G_5 and G_6 are identical.*

Lemma C.7 follows immediately from Lemma 4.2.

Game G_7. G_7 differs from G_6 in that $\mathbf{M}^\top$ is sampled from $\mathcal{OS}_\lambda$ instead of $\mathcal{ZS}_\lambda$ in the setup and each invocation. To show the indistinguishability, we define a sequence of intermediate games $\mathsf{G}_{6.0}, \cdots, \mathsf{G}_{6.q}$, where $\mathsf{G}_{6.0}$ is the same as G_6 and $\mathsf{G}_{6.\mathsf{c}}$ differs from $\mathsf{G}_{6.\mathsf{c}-1}$ in that $\mathbf{M}^\top$ in the cth invocation is sampled from $\mathcal{OS}_\lambda$.

Lemma C.8 *If $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$ holds, then the views of $\mathcal{A}$ in G_6 and G_7 are statistically close with negligible distance.*

Proof. For each c, fixing all the seeds except for the internal randomness used by $\mathcal{ZS}_\lambda$ to generate $\mathbf{M}$ in the cth invocation (i.e., $\mathbf{M}'$ in the $(\mathsf{c}-1)$th invocation or $\mathbf{M}$ in the initialization when $\mathsf{c}=1$) in $\mathsf{G}_{6.\mathsf{c}-1}$, one can see that the assignment to wires in each invocation can be simulated in NC^1, since the assignments for all gate gadgets can be generated in $\mathsf{AC}^0[2]$ in parallel as in Figure 22 and $\{\mathsf{SimDec}_\lambda\} \in \mathsf{AC}^0[2]$. Also, the assignment to wires in the first $\mathsf{c}-3$ invocations (if $\mathsf{c}>2$) and the invocations after the cth one are independent of $\mathbf{M}$ in the cth invocation. [8] Then, for every $\mathsf{c} \in [q]$, by Lemma C.1, the view of $\mathcal{A}$ in $\mathsf{G}_{6.\mathsf{c}}$ is statistically close to its view in $\mathsf{G}_{6.\mathsf{c}-1}$ with negligible distance. Finally, applying Lemma C.1 once more, with the two remaining leakage functions fixed to constants, switches the final matrix $\mathbf{M}'^\top$ generated in the qth invocation to the distribution specified in G_7. Applying a hybrid argument completes the proof of Lemma C.8. □

Game G_8. In G_8, assignments to wires of all gates except the output gates in the computation procedure are generated as in G_0, and all $\mathbf{P}_i$ and $\mathbf{S}_i$ are generated as uniformly random matrices.

Lemma C.9 *The views of $\mathcal{A}$ in G_7 and G_8 are identical.*

Proof. In G_7, since $\mathbf{M} \otimes \mathbf{I}_\lambda$ is of full rank, each $\mathbf{P}_i = (\mathbf{M} \otimes \mathbf{I}_\lambda)\mathbf{H}_i + (\mathbf{e}_\lambda \otimes \mathbf{I}_\lambda) \cdot \mathbf{Q}_i'^\top$ is uniformly distributed in $\{0,1\}^{\lambda^2 \times \lambda}$ and reveals no information on $\mathbf{Q}_i'$. Hence, $\mathbf{T}_i = \mathbf{M}'\mathbf{Q}_i' + s_i \otimes \mathbf{I}_\lambda$ is uniformly distributed in $\{0,1\}^{\lambda \times \lambda}$ as well. Then Lemma C.9 follows immediately from Lemma 3.2. □

One can see that the view of $\mathcal{A}$ in G_8 is identical to its view in the ideal game $\mathsf{Ideal}_\mathcal{A}$ (see Figure 3), where in each invocation, $\mathcal{A}$ obtains the current external output and leakage on the assignment to wires of the LRC generated by the simulator Sim^*_λ, and returns a new pair of external input and leakage function for the next invocation. Then Theorem 5.5 immediately follows. □

D Proof of Theorem 6.2

In this section, we give the proof of Theorem 6.2.

Proof. **Complexity and correctness.** We first note that $(\mathsf{Gen}_\lambda, \mathsf{Ver}_\lambda) \in \mathcal{C}_\lambda$ since $\mathsf{LTSetup}_\lambda \in \mathcal{C}_\lambda$ and Ver_λ can perform all the checking procedures in parallel using an AC^0 circuit. Specifically, Ver_λ can check the consistency of the input and CRS assignments, the consistency of all gates, and whether the output wire is assigned 1, in parallel with a depth-2 circuit. It then uses an AND gate to output 1 if all the checks pass. Hence, the total depth of Ver_λ is 3.

Moreover, according to the correctness of $\mathsf{AI\text{-}LTC}$ and that of Encode, we have

$$\mathsf{LTComp}_\lambda(\mathsf{aux}, (\mathbf{x}, \mathsf{ct})) = \mathsf{F}_\lambda(\mathbf{x}, \mathsf{ct}) = \mathsf{R}_\lambda(\mathbf{x}, \mathsf{D}_\lambda(\mathsf{ct})) = \mathsf{R}_\lambda(\mathbf{x}, \mathbf{w}) = 1$$

for all $(\mathbf{x}, \mathbf{w})$ such that $\mathsf{R}_\lambda(\mathbf{x}, \mathbf{w}) = 1$, all $\mathsf{ct} \in \mathsf{E}_\lambda(\mathbf{w})$, and all $\mathsf{aux} \in \mathsf{LTSetup}_\lambda$. Then correctness immediately follows.

Soundness. For any $\mathbf{x}$ and $\mathcal{W}$ passing the verification, there must exist ct (included in $\mathcal{W}$) such that

$$\mathsf{R}_\lambda(\mathbf{x}, \mathsf{D}_\lambda(\mathsf{ct})) = \mathsf{F}_\lambda(\mathbf{x}, \mathsf{ct}) = \mathsf{LTComp}_\lambda(\mathsf{aux}, (\mathbf{x}, \mathsf{ct})) = 1,$$

according to the correctness of $\mathsf{AI\text{-}LTC}$. Hence, there exists a valid bit string $\mathbf{w} = \mathsf{D}_\lambda(\mathsf{ct})$ such that $\mathsf{R}_\lambda(\mathbf{x}, \mathbf{w}) = 1$, completing the proof of soundness.

Zero-knowledge with offline simulation. Let $\{\mathcal{D}_\lambda\}$ be the distribution family over the simulators of $\mathsf{AI\text{-}LTC}$. The description of the distribution family $\{\mathcal{D}^{\mathsf{NIZK}}_\lambda\}$ over simulators is given as in Figure 23.

[8] Note that $\mathbf{M}$ in the cth round is required for simulating the encoded state generated in the $(\mathsf{c}-2)$th round.

$\underline{\mathcal{D}_\lambda^{\mathsf{NIZK}}:}$

Compute $\mathsf{ct} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{0})$

Sample $\mathsf{LTSim}_\lambda \xleftarrow{\$} \mathcal{D}_\lambda$ and return $\mathsf{NIZKSim}_\lambda$ defined as below with LTSim_λ and ct hardwired into it

$\underline{\mathsf{NIZKSim}_\lambda(\mathbf{x})}$

Compute $\mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathbf{x}, \mathsf{ct}), 1)$

Set aux as the assignment to auxiliary input wires included in $\mathcal{W}$

Return $(\mathsf{crs} = \mathsf{aux}, \pi = \mathcal{W})$

Figure 23: Definition of $\{\mathcal{D}_\lambda^{\mathsf{NIZK}}\}$.

Let $\lambda \in \mathbb{N}$ be the security parameter. We note that all $\mathsf{NIZKSim}_\lambda \in \mathcal{D}_\lambda^{\mathsf{NIZK}}$ run in $\mathcal{C}_\lambda$, since we have $\mathsf{LTSim}_\lambda \in \mathcal{C}_\lambda$ for all $\mathsf{LTSim}_\lambda \in \mathcal{D}_\lambda$. Let $(\mathbf{x}, \mathbf{w})$ be a statement/witness pair such that $\mathsf{R}_\lambda(\mathbf{x}, \mathbf{w}) = 1$ and $\mathcal{A}_\lambda$ be any distinguisher in $\mathcal{C}_\lambda$. We define several intermediate games, where in each game, $\mathcal{A}_\lambda$ receives a CRS/proof pair (aux, π) generated by the challenger as described in Figure 24. Below by ϵ_i we denote the probability that $\mathcal{A}$ outputs 1 in G_i for all i.

G_0, $\boxed{\mathsf{G}_1}$, $\boxed{\mathsf{G}_2}$

Compute $\mathsf{ct} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{w})$ $\boxed{\mathsf{ct} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{0})}$,

Sample $\mathsf{aux} \xleftarrow{\$} \mathsf{LTSetup}_\lambda$ and compute $(\mathcal{W}, b) \Leftarrow \mathsf{LTComp}_\lambda(\mathsf{aux}, (\mathbf{x}, \mathsf{ct}))$

$\boxed{\begin{array}{l}\text{Sample } \mathsf{LTSim}_\lambda \xleftarrow{\$} \mathcal{D}_\lambda \text{ and compute } \mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathbf{x}, \mathsf{ct}), 1) \\ \text{Set } \mathsf{aux} \text{ as the assignment to auxiliary input wires included in } \mathcal{W}\end{array}}$

Return $(\mathsf{crs} = \mathsf{aux}, \pi = \mathcal{W})$

Figure 24: Generation of (crs, π) in the intermediate games.

Game G_0. In G_0, $\mathcal{A}_\lambda$ takes as input $(\mathsf{crs} = \mathsf{aux}, \pi = \mathcal{W})$, where $\mathsf{aux} \xleftarrow{\$} \mathsf{LTSetup}_\lambda$, $\mathsf{ct} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{w})$, and $\mathcal{W} \xleftarrow{\$} \mathsf{W}[\mathsf{LTComp}_\lambda(\mathsf{aux}, (\mathbf{x}, \mathsf{ct}))]$, and outputs some bit $\tilde{b}$.

Game G_1. G_1 differs from G_0 in the way $\mathcal{W}$ is generated. Specifically, $\mathcal{W}$ is generated as $\mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathbf{x}, \mathsf{ct}), 1)$ rather than $\mathcal{W} \xleftarrow{\$} \mathsf{W}[\mathsf{LTComp}_\lambda(\mathsf{aux}, (\mathbf{x}, \mathsf{ct}))]$.

Lemma D.1 $|\epsilon_1 - \epsilon_0| \leq \mathsf{negl}(\lambda)$.

Proof. In G_1, we first fix all the seeds used in E_λ, such that the adversary's distinguishing advantage is maximized conditioned on this fixing. Then we build a leakage function ℓ_λ with all the fixed seeds hardwired into it as follows.

On input $\mathcal{W}$ generated as $\mathcal{W} \xleftarrow{\$} \mathsf{W}[\mathsf{LTComp}_\lambda(\mathsf{aux}, (\mathbf{x}, \mathsf{ct}))]$ or $\mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathbf{x}, \mathsf{ct}), 1)$, where $\mathsf{aux} \xleftarrow{\$} \mathsf{LTSetup}_\lambda$ and $\mathsf{LTSim}_\lambda \xleftarrow{\$} \mathcal{D}_\lambda$, ℓ_λ returns whatever $\mathcal{A}_\lambda(\mathsf{aux}, \mathcal{W})$ outputs.

One can see that $\ell_\lambda \in \mathcal{L}_\lambda$, since $\mathcal{A}_\lambda$ runs in $\mathcal{C}_\lambda$ with its output length being 1. Moreover, when $\mathcal{W} \xleftarrow{\$} \mathsf{W}[\mathsf{LTComp}_\lambda(\mathsf{aux}, (\mathbf{x}, \mathsf{ct}))]$ (respectively, $\mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathbf{x}, \mathsf{ct}), 1)$), the view of $\mathcal{A}_\lambda$ is exactly the same as its view in G_0 (respectively, G_1). As a result, we have

$$\Delta\Big(\ell_\lambda(\mathsf{W}[\mathsf{LTComp}_\lambda(\mathsf{aux}, (\mathbf{x}, \mathsf{ct}))]), \ell_\lambda(\mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathbf{x}, \mathsf{ct}), 1))\Big) \geq |\epsilon_1 - \epsilon_0|,$$

where the left-hand side is negligible, due to the $\{\mathcal{L}_\lambda\}$-leakage-tolerance of $\mathsf{AI\text{-}LTC}$. Then Lemma D.1 immediately follows. □

Game G_2. G_2 differs from G_1 in that ct is generated as $\mathsf{ct} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{0})$.

Lemma D.2 $|\epsilon_2 - \epsilon_1| \leq \mathsf{negl}(\lambda)$.

Proof. In G_2, we first fix the randomness except for those used to generate ct, such that the adversary's distinguishing advantage is maximized conditioned on this fixing. We then build a leakage function ℓ'_λ as follows.

On input ct generated as $\mathsf{ct} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{w})$ or $\mathsf{ct} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{0})$, ℓ'_λ runs $\mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathbf{x}, \mathsf{ct}), 1)$, sets aux as the assignment to the auxiliary input wires included in $\mathcal{W}$, and returns whatever $\mathcal{A}_\lambda(\mathsf{aux}, \mathcal{W})$ outputs.

One can see that $\ell'_\lambda \in \mathcal{L}_\lambda$ since we have $\mathsf{LTSim}_\lambda \in \mathcal{C}_\lambda$ and $\mathcal{A}_\lambda \in \mathcal{C}_\lambda$. Moreover, when $\mathsf{ct} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{w})$ (respectively, $\mathsf{ct} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{0})$), the view of $\mathcal{A}_\lambda$ is exactly the same as its view in G_1 (respectively, G_2). Hence, we have $\Delta\Big(\ell'_\lambda(\mathsf{E}_\lambda(\mathbf{w})), \ell'_\lambda(\mathsf{E}_\lambda(\mathbf{0}))\Big) \geq |\epsilon_2 - \epsilon_1|$, where the left-hand side is negligible due to the $\{\mathcal{L}_\lambda\}$-leakage-resilience of Encode, completing the proof of Lemma D.2. □

Putting all the above together, Theorem 6.2 immediately follows. □

E Proof of Theorem 6.6

In this section, we give the proof of Theorem 6.6.

Proof. Let $\{\mathsf{Check}_\lambda \in \mathcal{C}_\lambda\}$ be the function family for CRS checking. We prove the following lemmas.

Lemma E.1 *For all $\lambda \in \mathbb{N}$, we have*

$$\Pr[\mathsf{Check}_\lambda(\mathsf{crs}) = 1, \mathsf{Ver}_\lambda(\mathsf{crs}, \mathbf{x}, \pi) = 1 | \\ (\mathbf{x}, \mathbf{w}) \xleftarrow{\$} \mathcal{YS}_\lambda, \mathsf{NIZKSim}_\lambda \xleftarrow{\$} \mathcal{D}_\lambda^{\mathsf{NIZK}}, (\mathsf{crs}, \pi) \xleftarrow{\$} \mathsf{NIZKSim}_\lambda(\mathbf{x})] \geq 1 - \mathsf{negl}(\lambda).$$

Proof. Let $(\mathbf{x}, \mathbf{w}) \xleftarrow{\$} \mathcal{YS}_\lambda$. We build an adversary $\{\mathcal{A}_\lambda\}$ against the zero-knowledge with offline simulation as follows.

$\mathcal{A}_\lambda$ on input (crs, π) where $(\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}_\lambda, \pi \xleftarrow{\$} \mathsf{Prove}_\lambda(\mathsf{crs}, \mathbf{x}, \mathbf{w}))$ or $(\mathsf{NIZKSim}_\lambda \xleftarrow{\$} \mathcal{D}_\lambda^{\mathsf{NIZK}}, (\mathsf{crs}, \pi) \xleftarrow{\$} \mathsf{NIZKSim}_\lambda(\mathbf{x}))$ returns 1 iff $\mathsf{Check}_\lambda(\mathsf{crs}) = \mathsf{Ver}_\lambda(\mathsf{crs}, \mathbf{x}, \pi) = 1$.

One can see that $\mathcal{A}_\lambda \in \mathcal{C}_\lambda$ since we have $\mathsf{Check}_\lambda \in \mathcal{C}_\lambda$ and $\mathsf{Ver}_\lambda \in \mathcal{C}_\lambda$. Moreover, when (crs, π) are honestly generated by Gen_λ and Prove_λ, $\mathcal{A}_\lambda$ must output 1. Then the probability that $\mathcal{A}_\lambda$ outputs 1 must be $1 - \mathsf{negl}(\lambda)$ when (crs, π) are generated by the simulator, due to the zero-knowledge with offline simulation. As a result, the probability that the simulated CRS/proof pair passes verification must be $1 - \mathsf{negl}(\lambda)$, completing the proof of Lemma E.1. □

Lemma E.2 *For all $\lambda \in \mathbb{N}$, we have*

$$\Pr[\mathsf{Check}_\lambda(\mathsf{crs}) = 1, \mathsf{Ver}_\lambda(\mathsf{crs}, \mathbf{x}, \pi) = 1 | \\ \mathbf{x} \xleftarrow{\$} \mathcal{NS}_\lambda, \mathsf{NIZKSim}_\lambda \xleftarrow{\$} \mathcal{D}_\lambda^{\mathsf{NIZK}}, (\mathsf{crs}, \pi) \xleftarrow{\$} \mathsf{NIZKSim}_\lambda(\mathbf{x})] \geq 1 - \mathsf{negl}(\lambda).$$

Proof. Assume that the probability in Lemma E.2 is $1 - \epsilon$. We build an adversary $\{\mathcal{A}_\lambda\}$ against the subset membership problem as follows.

$\mathcal{A}_\lambda$ on input $\mathbf{x}$, where $(\mathbf{x}, \mathbf{w}) \xleftarrow{\$} \mathcal{YS}_\lambda$ or $\mathbf{x} \xleftarrow{\$} \mathcal{NS}_\lambda$, runs $\mathsf{NIZKSim}_\lambda \xleftarrow{\$} \mathcal{D}_\lambda^{\mathsf{NIZK}}$ and $(\mathsf{crs}, \pi) \xleftarrow{\$} \mathsf{NIZKSim}_\lambda(\mathbf{x})$, and returns 1 iff $\mathsf{Check}_\lambda(\mathsf{crs}) = \mathsf{Ver}_\lambda(\mathsf{crs}, \mathbf{x}, \pi) = 1$.

When $\mathbf{x}$ is generated as $(\mathbf{x}, \mathbf{w}) \xleftarrow{\$} \mathcal{YS}_\lambda$ (respectively, $\mathbf{x} \xleftarrow{\$} \mathcal{NS}_\lambda$), the probability that $\mathcal{A}_\lambda$ outputs 1 is $1 - \mathsf{negl}(\lambda)$ by Lemma E.1 (respectively, $1 - \epsilon$ by assumption). Fixing all the internal randomness in $\mathcal{D}_\lambda^{\mathsf{NIZK}}$ maximizing the advantage of $\mathcal{A}_\lambda$, we have $\mathcal{A}_\lambda \in \mathcal{C}_\lambda$ and the advantage of $\mathcal{A}_\lambda$ is at least $|\mathsf{negl}(\lambda) - \epsilon|$. As a result, ϵ must be negligible, due to (the second property of) the subset membership problem, completing the proof of Lemma E.2. □

By Lemma E.2, there exists some $(\mathsf{crs}, \mathbf{x}, \pi)$ such that $\mathsf{crs} \in \mathsf{Gen}_\lambda$, $\mathbf{x} \notin \mathcal{V}_\lambda$, and $\mathsf{Ver}_\lambda(\mathsf{crs}, \mathbf{x}, \pi) = 1$, which contradicts the perfect soundness of NIZK, completing the proof of Theorem 6.6. □