Secure Computation against NC^1 Leakage without Secure Hardware

Yuyu Wang*

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

Abstract

In this work, we construct (stateful) leakage-resilient circuits (LRCs) secure against bounded-output-length leakage functions computable by NC^1 circuits under the mild worst-case assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$, without relying on any leak-free hardware components, thereby resolving the open problem left by Bogdanov, Ishai, and Srinivasan (CRYPTO 2019; Journal of Cryptology, 2021) and Wang (CRYPTO 2025).

Concretely, we first construct a leakage-tolerant circuit with succinct setup (sAI-LTC) secure against 2-adaptive NC^1 leakage, and then generically combine it with a 2-adaptive leakage-resilient composable encoding scheme to obtain the desired LRC.

We further give a direct non-black-box instantiation that optimizes the compiled circuit size at the cost of a slightly larger setup, matching the circuit size of Wang's construction that relies on leak-free hardware while using a more compact setup.

Finally, we show that our sAI-LTC generically implies a fine-grained multi-theorem non-interactive proof system for all NP, with compact common reference strings, perfect soundness, and multi-theorem zero-knowledge with offline simulation against NC^1 adversaries.

Keywords: Leakage-resilient circuit · NC^1 · Complexity assumption · Non-interactive proof system.

1 Introduction

1.1 Background

In cryptography, a central objective is to protect computations against adversaries who can observe more than just an algorithm's input/output behavior. In practice, cryptographic implementations often leak partial information through physical side channels, such as power consumption, timing, and electromagnetic radiation, and adversaries may exploit this leakage to recover secret data. Attacks based on such physical observations are commonly referred to as side-channel attacks and pose a fundamental threat to the security of real-world cryptographic systems.

To address this challenge, the study of leakage-resilient cryptography has developed along two main directions. One line of work focuses on constructing specific cryptographic primitives (e.g., encryption and signatures) that remain secure in the presence of leakage on secret keys or internal randomness (e.g., [1, 5]). The other line, which is the focus of this work, studies *leakage-resilient circuits (LRCs)*, aiming to protect general computations against leakage on their intermediate steps and internal states.

The notion of LRC was introduced by Ishai, Sahai, and Wagner [13]. In their definition, a *stateful* LRC maintains an encoded secret state generated in the initial setup and, in each invocation, takes an external input, returns an external output, and refreshes the encoding. All intermediate wire values, including those involved in state updates, are subject to leakage in every invocation. Security requires that the original secret state remains hidden despite continual leakage across an arbitrary polynomial number of executions. We note that [13] also introduced *stateless* LRCs, which capture one-shot computations and can be viewed as a special, strictly weaker case of stateful LRCs. In particular, they do not protect

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

persistent secrets across repeated executions and are typically instantiated using leak-free encoders and decoders. Unless stated otherwise, all LRCs considered in this work are stateful.

Early works focused on LRCs against *wire-probing leakage* [13], in which the adversary observes a limited number of wires, capturing certain forms of noisy leakage [7]. Subsequent works introduced richer forms of *global leakage*, including the *only-computation leakage* model [15], where leakage is applied separately to different phases of the computation, the *bounded-communication leakage* model [11], where leakage is applied to arbitrarily partitioned wires with bounded length and may be interleaved across parts, and the *computationally bounded leakage* model [9], which allows computationally bounded leakage functions that apply jointly to all wires in the circuit.

Most existing constructions in the computationally bounded leakage model handle leakage functions (with bounded output length) computable in AC^0 (i.e., constant-depth, polynomial-sized circuits consisting of unbounded fan-in AND, OR, and NOT gates) [9, 20, 4, 3, 14]. These constructions crucially exploit the limited expressive power of AC^0, in particular its inability to compute parity, which enables simple XOR-based secret-sharing schemes for encoding wire values. However, once the leakage is allowed to be computed by circuits of non-constant depth, these techniques immediately break down. Indeed, as noted by Bogdanov, Ishai, and Srinivasan [4] (whose preliminary version appeared at CRYPTO 2019), who proposed LRCs that are unconditionally secure against AC^0 leakage functions with logarithmic output length without using leak-free hardware (i.e., hardware whose internals never leak), extending their techniques to handle richer leakage classes such as NC^1 (i.e., logarithmic-depth, polynomial-sized circuits over fan-in two AND, OR, and NOT gates), which is strictly more expressive than AC^0, under standard complexity-theoretic assumptions and still without leak-free hardware, remains an *open* challenge.

On the other hand, Miles and Viola [17] proposed an LRC candidate secure against NC^1 leakage functions (with logarithmic output length). Miles [16] subsequently proved its security under the mild worst-case complexity assumption $\mathsf{NC}^1 \subsetneq \mathsf{L}$. A major drawback of this construction, however, is its reliance, in each invocation, on additional leak-free hardware components whose number is proportional to the size of the original circuit.

More recently, Wang [22] reduced this overhead by constructing LRCs whose leak-free hardware size depends only on the internal-state and external-output lengths, rather than on the size of the original circuit, while achieving the same security under an even weaker assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$. To this end, Wang introduced a *leakage-tolerant circuit with auxiliary input (AI-LTC)* against NC^1 leakage, which relies on a leak-free setup used to generate auxiliary inputs and admits a distribution over efficient simulators that can fool the leakage given only input/output pairs, and proposed a generic transformation from AI-LTC to LRC. In the resulting LRC, a leak-free hardware component runs the setup in each invocation, and its size depends on the internal-state and external-output lengths. A natural question is whether the setup can be a public-coin algorithm, thereby eliminating the need for secure hardware. Unfortunately, Wang proved that this is impossible, under $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$, [1] and left the question of whether it is possible to bypass this negative result and achieve LRCs against NC^1 leakage without leak-free hardware *open*.

1.2 Our Contributions

In this work, we give the first full affirmative answer to the open problem explicitly posed in [4, 22] as discussed above, by constructing LRCs that are secure against NC^1 leakage functions with logarithmic output length (i.e., the same leakage class considered in [17, 22]), while requiring no leak-free hardware.

We begin by introducing a stronger variant of AI-LTC, which we call 2-*adaptive leakage-tolerant circuit with succinct setup (sAI-LTC)*, together with an instantiation secure against NC^1 leakage. Informally, a 2-adaptive sAI-LTC strengthens the security notion of AI-LTC by tolerating adversaries that may adaptively choose two leakage functions, one taking as input the auxiliary input and the other taking as input the wire values of the computation. More importantly, sAI-LTCs satisfy a succinctness property: the size of the setup depends only on the security parameter and is independent of parameters of the original circuit, including its input/output lengths.

[1] The original definition of leakage-tolerant circuits in [14] does not require auxiliary inputs, as it allows simulators to be inefficient and to take leakage on the input/output pair. They showed that such LTCs suffice for a relaxed notion of LRCs, where the real leakage in each invocation is simulated by a possibly inefficient simulator given the output of a function in the leakage class.

We then present a generic construction of LRC from 2-adaptive leakage-resilient composable encoding (LRCE) [22] and 2-adaptive sAI-LTC, yielding an instantiation secure against NC^1 leakage without relying on any leak-free hardware. The resulting construction has an initial setup of size $O(l_{\mathsf{st}} \cdot \lambda + \lambda^2)$, and a compiled circuit (i.e., the computation algorithm) of size $O(|\mathsf{F}_\lambda| \cdot \lambda^3 + l_{\mathsf{st}} \cdot \lambda^6 + \lambda^7)$, where λ, $|\mathsf{F}_\lambda|$, and l_{st} denote the security parameter, the size of the original circuit measured by the number of gates, and the length of the internal state, respectively.

We also give a non-black-box, direct construction of LRC with the same security. Although its setup algorithm has a larger size of $O(l_{\mathsf{st}} \cdot \lambda^2)$, the resulting circuit is smaller, with size $O(|\mathsf{F}_\lambda| \cdot \lambda^3 + l_{\mathsf{st}} \cdot \lambda^4)$.

The security of all our constructions is proven under the mild worst-case assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$. As noted in [6, 22], this assumption holds if there exists a language that has only branching programs of super-constant width, and it would be quite surprising if this were not the case.

Implications for NC^1 leakage. As already emphasized in [22], the NC^1 leakage class is strictly more expressive than AC^0 and captures natural leakage functions such as PARITY and MAJORITY. In fact, even a single bit of NC^1 leakage suffices to break existing LRCs against AC^0 leakage, while allowing logarithmically many leaked bits further enables adversaries to extract structural information, such as the locations of wires carrying secrets and the identities of secret users. NC^1 leakage also captures side-channel attacks in the Hamming-distance model, as well as certain polynomial-time leakage attacks that use logarithmic space.

Comparison. In Table 1, we compare our constructions with prior works in terms of the sizes of the leak-free hardware (modeled as opaque gates), the setup algorithm, and the compiled circuit. All constructions in the table are secure against NC^1 leakage functions with logarithmic output length.

Scheme	Opaque gate size	Setup size	Compiled circuit size	Assumption
MV13 [17]	$O(\lvert\mathsf{F}_\lambda\rvert \cdot \lambda^3)$	$O(l_{\mathsf{st}} \cdot \lambda^2)$	$O(\lvert\mathsf{F}_\lambda\rvert \cdot \lambda^3)$	$\mathsf{NC}^1 \subsetneq \mathsf{L}$
W25 [22]	$O(l_{\mathsf{st}} \cdot \lambda^2 + l_{\mathsf{out}} \cdot \lambda)$	$O(l_{\mathsf{st}} \cdot \lambda^2)$	$O(\lvert\mathsf{F}_\lambda\rvert \cdot \lambda^3 + l_{\mathsf{st}} \cdot \lambda^6)$	$\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$
W25 [22]	$O(l_{\mathsf{st}} \cdot \lambda^3 + l_{\mathsf{out}} \cdot \lambda)$	$O(l_{\mathsf{st}} \cdot \lambda^3)$	$O(\lvert\mathsf{F}_\lambda\rvert \cdot \lambda^3 + l_{\mathsf{st}} \cdot \lambda^4)$	$\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$
Ours (Section 4)	0	$O(l_{\mathsf{st}} \cdot \lambda + \lambda^2)$	$O(\lvert\mathsf{F}_\lambda\rvert \cdot \lambda^3 + l_{\mathsf{st}} \cdot \lambda^6 + \lambda^7)$	$\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$
Ours (Section 5)	0	$O(l_{\mathsf{st}} \cdot \lambda^2)$	$O(\lvert\mathsf{F}_\lambda\rvert \cdot \lambda^3 + l_{\mathsf{st}} \cdot \lambda^4)$	$\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$

Table 1: Comparison of existing LRCs against NC^1 leakage. λ denotes the security parameter. l_{in}, l_{out}, and l_{st} denote the lengths of the external input, external output, and internal state of the original circuit, respectively. $|\mathsf{F}_\lambda|$ denotes the size of the original circuit measured by the number of gates.

One can see that our first instantiation achieves the smallest setup size amongst all listed constructions, at the cost of larger compiled circuit size for large security parameters. Our second instantiation matches the compiled circuit size of the second construction in [22], while using a smaller setup. At the same time, it achieves the same setup size as the first construction in [22], while yielding a smaller compiled circuit (for large security parameters). None of our constructions rely on leak-free hardware, whereas the constructions of [17] and [22] exploit leak-free hardware components whose sizes are proportional to the total size of the original circuit and to the internal-state and external-output lengths, respectively.

Fine-grained multi-theorem proof system. While [22] proposed a generic transformation from AI-LTC to fine-grained *single-theorem* non-interactive zero-knowledge proof system with offline simulation (oNIZK), where zero-knowledge with offline simulation implies witness hiding and weak zero-knowledge [2], we show that replacing the underlying AI-LTC with our sAI-LTC yields a fine-grained *multi-theorem* oNIZK. As in [22], in our instantiation the CRS generator, verifier, and simulator all run in NC^1, and perfect soundness is satisfied. Under the same assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$, our construction further achieves multi-theorem zero-knowledge with offline simulation against NC^1 adversaries, allowing a single *compact* CRS to be safely reused for generating any polynomial number of proofs without compromising zero-knowledge. In contrast, in prior fine-grained oNIZKs under the assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$ [2, 22], the CRS sizes grow with the number of proofs and an a priori bound on that number is required.

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

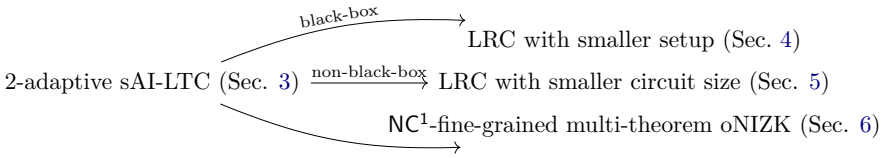

Figure 1: Overview of our contributions. All LRCs are secure against NC^1 leakage.

1.3 Technical Overview

A central challenge in constructing LRCs is that the simulator, without access to the underlying plaintext state, must continuously fool leakage over an unbounded polynomial number of invocations. Existing constructions secure against NC^1 leakage [17, 22] address this by relying on leak-free hardware to provide fresh auxiliary information for simulation in each invocation. Without secure hardware, however, the initial setup becomes the only leak-free point of control, after which all computations are exposed to leakage. Thus, the simulator must rely only on setup-generated information to support simulation throughout all invocations.

Our work takes the construction of [22] as a starting point, which realizes LRCs using leak-free hardware via a generic transformation from AI-LTCs. We then show how to eliminate the reliance on leak-free hardware.

The AI-LTC-to-LRC transformation of [22]. At a high level, the construction of [22] has two steps: it first constructs an AI-LTC for a single invocation, and then combines this AI-LTC with an LRCE to obtain an LRC.

Recall that for an AI-LTC, there exists a distribution over simulators that can fool the leakage given only input/output pairs. Its instantiation against NC^1 leakage proposed in [22] builds on the fine-grained fully homomorphic encryption (FHE) scheme of Wang and Pan (WP) [24]. The security of this FHE scheme relies, under the assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$, on the indistinguishability of two distributions $\mathsf{ZeroSamp}_\lambda$ and $\mathsf{OneSamp}_\lambda$ over $\lambda \times \lambda$ matrices of rank $\lambda - 1$ and full rank, respectively, against NC^1 circuits. Moreover, [22] shows that $\mathsf{ZeroSamp}_\lambda$ and $\mathsf{OneSamp}_\lambda$ fool all NC^1 leakage functions with logarithmic output length.

For any polynomial-sized stateless function F_λ, the AI-LTC encrypts the input $\mathbf{x}$ bit by bit under a public key $\mathbf{M}^\top \xleftarrow{\$} \mathsf{ZeroSamp}_\lambda$, homomorphically evaluates F_λ, and then decodes the output ciphertexts using Gaussian elimination. The security of the underlying WP-FHE does not directly suffice for homomorphic evaluation beyond NC^1 or for the decoding procedure under leakage. To reduce the security of the resulting AI-LTC to the indistinguishability between $\mathsf{ZeroSamp}_\lambda$ and $\mathsf{OneSamp}_\lambda$, [22] introduces *internal wires*. These wires carry ciphertexts of 0's together with their randomness, and are used to re-randomize the output of each gate gadget. This gives two simulation properties. First, the *external wires*, namely the wires carrying ciphertexts, can be simulated in NC^1 given $\mathbf{M}$ and the wire values of the original function. Second, each gate gadget before decoding is *reconstructible*: given $\mathbf{M}$, the relevant wire values of the original function, and the randomness of the ciphertexts involved in the gadget, its internal wire distribution can be simulated in NC^1. These properties allow the proof to switch $\mathbf{M}$ from $\mathsf{ZeroSamp}_\lambda$ to $\mathsf{OneSamp}_\lambda$. Under $\mathsf{OneSamp}_\lambda$, $\mathbf{M}$ is full rank. Hence, random ciphertext wires are uniformly distributed and hide the underlying wire values. Moreover, using $\mathbf{M}^{-1}$, the input and output ciphertext wires can be simulated in NC^1 from the given input and output values. Thus, apart from the refresh wires needed before decoding, the wire distribution can be simulated from the input/output pair alone. These remaining refresh wires are provided by the leak-free setup as auxiliary input.

The resulting AI-LTCs are then compiled into LRCs using standard encoding-based transformations [9, 14]. In the setup phase, the initial state is encoded using a 2-adaptive LRCE, instantiated via the fine-grained encryption scheme of Degwekar, Vaikuntanathan, and Vasudevan (DVV) [6]. In each invocation, the LRC runs the AI-LTC to decode the current state, execute the computation, and re-encode the updated state. Security follows from the security of the underlying AI-LTC, allowing the adversary's view to be simulated given only the encodings of states and the input/output pairs, together with the 2-adaptive security of the LRCE, ensuring that the leakage on encodings reveals little useful information. The setup algorithm of the underlying AI-LTC is inherited as the leak-free hardware.

A strawman: bounded LRC without secure hardware. In order to eliminate the use of leak-free

hardware, a natural idea is to generate the auxiliary inputs using only public coins. However, Wang [22] showed that this is impossible under the assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$. To circumvent this impossibility, a tempting alternative is to let the underlying AI-LTC itself generate the auxiliary input for each subsequent invocation, rather than relying on secure hardware, with the hope that the security of the AI-LTC keeps the generation procedure hidden from the adversary.

At first glance, this approach faces two conceptual issues. First, the AI-LTC of [22] supports only deterministic circuits, while the setup algorithm that generates the auxiliary input is inherently randomized. Fortunately, [22] shows that the underlying LRCE scheme can extend the encoding of the current state to include DVV ciphertexts of an arbitrary number of random bits using only public coins, enabling the AI-LTC to recover the randomness required without exposing it to leakage. Second, each auxiliary input generated in an invocation necessarily appears in the next, and is thus exposed to two adaptively chosen leakage functions. We handle this by reducing the security of the AI-LTC to the indistinguishability between $\mathsf{ZeroSamp}_\lambda$ and $\mathsf{OneSamp}_\lambda$ against 2-adaptive leakage functions instead.

Despite resolving these two issues, the main remaining obstacle is circularity: the auxiliary input now becomes part of the output itself, while in the AI-LTC, the size of the auxiliary input depends on, and in fact exceeds, the output size. As a result, incorporating the auxiliary input into the output increases the output length, which in turn increases the required size of the auxiliary input, and so on. A naive workaround would be to ensure that the auxiliary input used in the current invocation is longer than what will be required in the next. However, this would not only make the initial auxiliary input prohibitively large, but, more importantly, would impose an *a priori* bound on the total number of invocations.

Eliminating the a priori bound. To overcome the above problem, we extend the AI-LTC to an sAI-LTC, in which the size of the setup is independent of the original function, and thus remove the circular dependency. [2]

To achieve this, we take a closer look at the decoding phase in [22], since this is exactly where the auxiliary input is used. After the homomorphic evaluation, the AI-LTC obtains, for each output bit, a WP-FHE ciphertext under the public key $\mathbf{M}$, represented as a matrix in $\{0,1\}^{\lambda\times\lambda}$. The decoder does not use the entire ciphertext matrix to recover the output bit, but only its rightmost column, which is essentially a DVV ciphertext.

The decoding procedure necessarily exposes a randomly chosen internal randomness vector consistent with each DVV ciphertext being decoded. For a fixed DVV ciphertext, there are two possible consistent randomness vectors, and their difference is a non-zero vector in the kernel of $\mathbf{M}$. Since the homomorphic evaluation phase may leak information about randomness correlated with one of these vectors, exposing the other during decoding would allow the leakage from the two phases to be combined to recover this kernel vector, immediately compromising security. To prevent this, [22] refreshes the output ciphertexts using ciphertexts of 0's generated as auxiliary input in the leak-free setup, exploiting the additive homomorphism of the DVV encryption. This decorrelates the randomness exposed during decoding from that exposed during homomorphic evaluation. The cost is that the auxiliary input grows with the output length.

Since generating such a long auxiliary input is infeasible in our case, we instead seek a short object from which all the fresh ciphertexts needed for refresh can be derived. This leads to a new variant of the *ciphertext bank* machinery. Ciphertext banks were first introduced in [10] in the context of the only-computation leakage model, and have since been used extensively in AC^0-bounded leakage settings (e.g., [20, 4]). At a high level, a ciphertext bank consists of a small collection of ciphertexts from which one can derive an arbitrary polynomial number of ciphertexts of the same plaintext. Instantiating this idea in our setting, however, is far from straightforward, since both the leakage model and the structure of the ciphertexts and compiled circuits differ significantly from those in prior work.

We instantiate the ciphertext bank as a matrix in $\{0,1\}^{\lambda\times\lambda}$ consisting of λ DVV ciphertexts of 0's. From this bank, we generate an arbitrary number of ciphertexts of 0's by taking public random linear combinations of the ciphertexts in the bank, exploiting the additive homomorphism of the DVV encryption. Crucially, we show that leakage from decoding the refreshed output ciphertexts can be reduced to leakage on their refreshed randomness. Although this randomness may still be correlated with that used during homomorphic evaluation, the ciphertext-bank argument absorbs the effect of the

[2] We require the setup size itself to be independent of the original function, rather than merely requiring this for the auxiliary input. Otherwise, the function would need to run the setup, encode its randomness, and then be compiled using a setup whose size depends on the function, leading to another circular dependency.

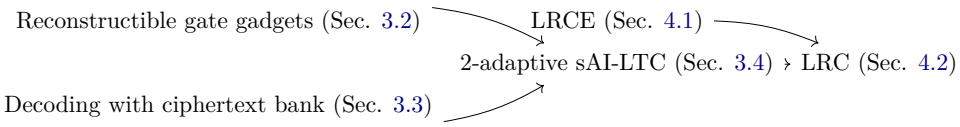

Figure 2: Overview of the generic route from sAI-LTC to LRC without secure hardware.

alternative consistent randomness that real decoding may expose, preventing recovery of the non-zero kernel vector.

Our core theorem (see Theorem 3.4 in Section 3.3) formalizes this reduction: after refreshing, leakage from the real decoding wires can be simulated from the randomness of the refreshed ciphertexts. This randomness consists of the randomness of the unrefreshed output ciphertexts together with random linear combinations of the bank randomness. The key point in the proof is to remove the possible difference between this refreshed randomness and the randomness actually exposed by real decoding: as discussed above, they may differ by a kernel component. Using a Raz-type extractor argument [18, 12], we show that this component can be absorbed by re-randomizing the hidden randomness underlying the ciphertext bank, without changing the distribution seen by the adversary.

Figure 2 summarizes our sAI-LTC-based generic LRC construction.

LRC with optimized circuit size. The above construction already removes secure hardware. We next explain how to further reduce the compiled circuit size.

As shown in Table 1, the compiled circuit size of the instantiation obtained from the generic framework above is $O(l_{\mathsf{st}} \cdot \lambda^6 + \lambda^7 + |\mathsf{F}_\lambda| \cdot \lambda^3)$, where the terms $O(l_{\mathsf{st}} \cdot \lambda^6)$ and $O(\lambda^7)$ are particularly undesirable. Specifically, the size of the decoding procedure for recovering randomness used to generate the encoding of the updated state is $O(l_{\mathsf{st}} \cdot \lambda^3)$, while that for recovering the randomness used to generate the ciphertext bank is $O(\lambda^4)$. Moreover, compiling each gate of these decoding procedures incurs an additional λ^3 multiplicative overhead, leading to the blow-up. Essentially, this inefficiency is due to the fact that both the state and the internal randomness used for encoding are themselves encoded twice.

To reduce the circuit size, we adopt the non-black-box approach of [22] and keep the encoded state directly in the WP-FHE form, rather than adding an outer LRCE layer. The encoded state maintained between invocations consists of the current public key $\mathbf{M}$, the current ciphertext bank, and WP-FHE ciphertexts of the plaintext state under $\mathbf{M}$. In each invocation, given the external input, the LRC homomorphically computes ciphertexts encoding both the external output and the next encoded state, namely the next public key, the next ciphertext bank, and WP-FHE ciphertexts of the updated plaintext state under the next public key. These ciphertexts are refreshed using 0-ciphertexts derived from the current ciphertext bank, and then decoded by Gaussian elimination with respect to $\mathbf{M}$.

Security follows by repeating the sAI-LTC hybrid argument over the invocations. In each invocation, gate-gadget reconstructibility handles the pre-decoding wires, while our core theorem (Theorem 3.4) handles leakage from the refreshed decoding procedures using the refreshed randomness. This puts the leakage in a form analogous to that in the sAI-LTC proof and allows the $\mathsf{ZeroSamp}_\lambda$-to-$\mathsf{OneSamp}_\lambda$ switch. Under $\mathsf{OneSamp}_\lambda$, $\mathbf{M}$ is full rank, so the ciphertext wires are uniform and hide the underlying plaintext state and computation. The next encoded state is again distributed in the required form, allowing the induction to proceed.

The subtle point is how to obtain ciphertexts of the next public key and the next ciphertext bank via homomorphic evaluation. A direct approach would first generate WP-FHE ciphertexts of the randomness underlying the next bank and then homomorphically derive ciphertexts of the next public key and ciphertext bank. However, this approach faces two obstacles. First, WP-FHE ciphertexts of the required form cannot be generated using only public coins. Second, even if this were possible, the resulting construction would still fail to eliminate the $O(\lambda^7)$ overhead.

We avoid this by exploiting the alternative representation of $\mathsf{ZeroSamp}_\lambda$. Once the public randomness is fixed, the next public key and the entire next ciphertext bank can be generated from a single short vector in $\{0,1\}^{\lambda-1}$ using only additive operations. Hence, instead of producing WP-FHE ciphertexts for all entries of these objects, we generate public-coin DVV ciphertexts of this short vector. These ciphertexts are shorter and can be generated using only public coins. By the additive homomorphic property of DVV, they yield ciphertexts of the next public key and ciphertext bank. As a result, we obtain a significantly smaller compiled circuit by avoiding the $O(\lambda^7)$ overhead in generating the next

public key and ciphertext bank.

Fine-grained multi-theorem oNIZK from sAI-LTC. In [22], it is shown that an AI-LTC for arbitrary polynomial-sized functions generically implies a fine-grained oNIZK for all NP. In this transformation, the setup algorithm of the AI-LTC serves as the CRS generator. Given a statement/witness pair, the prover encodes the witness using an LRCE and runs the AI-LTC to recover the witness and check validity. The proof consists of the tableau corresponding to the computation phase of the AI-LTC. Perfect soundness follows from the correctness of the AI-LTC, and zero-knowledge with offline simulation follows from the fact that an AI-LTC simulator (sampled offline) can fool any adversary in the leakage class using a dummy encoding.

The resulting oNIZK is inherently single-theorem: the CRS is tied to individual proofs and cannot be reused across multiple statements, since the simulator of the underlying AI-LTC generates the auxiliary input jointly with its tableau. Our sAI-LTC removes this restriction, as the simulated auxiliary input is generated *independently* of the simulation for the computation phase (see Section 3.4). Moreover, it satisfies a natural property called parallel-composability, which ensures the existence of simulators that can fool the leakage on the tableaus in parallel executions of the computation phases. As a result, the CRS can be generated once and reused for an arbitrary polynomial number of statements, yielding a fine-grained multi-theorem oNIZK with compact CRSs.

2 Preliminaries

Notations. All arithmetic computations in this work are over $GF(2)$.

For a distribution $\mathcal{D}$, by $x \xleftarrow{\$} \mathcal{D}$ we denote sampling x according to $\mathcal{D}$. If $\mathcal{D}$ is a set, then x is sampled uniformly at random from $\mathcal{D}$. For an algorithm $\mathcal{A}$, by $y \xleftarrow{\$} \mathcal{A}(x)$ (respectively, $y = \mathcal{A}(x)$) we denote running $\mathcal{A}$ probabilistically (respectively, deterministically) on input x and obtaining output y. $y \xleftarrow{\$} \mathcal{A}(x)$ is equivalent to $y = \mathcal{A}(x; r)$ for $r \xleftarrow{\$} \mathcal{R}$, where $\mathcal{R}$ is the randomness space of $\mathcal{A}$. 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 input and output wires) of $\mathcal{A}(x; r)$ for $r \xleftarrow{\$} \mathcal{R}$. Moreover, $(\mathcal{W}, y) \Leftarrow \mathcal{A}(x)$ is equivalent to $y = \mathcal{A}(x; r)$ and $\mathcal{W} = \mathsf{W}[\mathcal{A}(x; r)]$ for $r \xleftarrow{\$} \mathcal{R}$. By $|\mathcal{A}|$ we denote the size of $\mathcal{A}$, measured by the number of gates. By $\mathcal{A}^{\|q}$ we denote q independent parallel executions of $\mathcal{A}$.

The *statistical distance* between random variables x and y over a common support $\mathcal{D}$ is defined as $\Delta(x, y) = \frac{1}{2} \sum_{v \in \mathcal{D}} \big|\Pr[x = v] - \Pr[y = v]\big|$. By $\mathsf{negl}(\cdot)$ we denote an unspecified negligible function.

By $\mathbf{x} \in \{0,1\}^n$ we denote a column vector of length n, and by x_i its ith entry. By $|\mathbf{x}|$ we denote the length of $\mathbf{x}$. For a matrix $\mathbf{A} \in \{0,1\}^{n \times t}$, by $\mathsf{Span}(\mathbf{A})$ and $\mathsf{Ker}(\mathbf{A})$ we denote its span $\{\mathbf{y} \,|\, \exists \mathbf{x} \text{ s.t. } \mathbf{y} = \mathbf{A}\mathbf{x}\}$ and kernel $\{\mathbf{x} \,|\, \mathbf{A}\mathbf{x} = \mathbf{0}\}$ respectively. By $\overline{\mathbf{A}}$ and $\underline{\mathbf{A}}$ we denote the upper $(n-1) \times t$ submatrix and the last row of $\mathbf{A}$ respectively. By $\mathbf{I}_n$ we denote the 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 others being 0, and by $\mathbf{0}$ and $\mathbf{1}$ we denote the all-zero and all-one vector or matrix (of unspecified sizes) respectively. By $[n]$ and $[n\!:\!m]$ we denote the sets $\{1, \cdots, n\}$ and $\{n, \cdots, m\}$ respectively. For $c \in \{0,1\}$, $\mathbf{A} \in \{0,1\}^{n_1 \times m_1}$, and $\mathbf{B} \in \{0,1\}^{n_2 \times m_2}$, the Kronecker product $\otimes$ is defined by

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

2.1 Function Families

In this section, we recall several function families.

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 the restricted-output variants of NC^1 are closed under composition in this sense.

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 ($\oplus\mathsf{L}/\mathsf{poly}$). $\oplus\mathsf{L}/\mathsf{poly}$ 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 $\mathcal{M}_\lambda(x)$.

Definition 2.4 ($\mathsf{AC}^0[2]$). 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 $\mathsf{AC}^0[2] \subsetneq \mathsf{NC}^1$ [19, 21], and multiplication of a constant number of polynomial-sized matrices (capturing addition of a polynomial number of matrices) can be performed in $\mathsf{AC}^0[2]$. The definition of AC^0 is identical to that of $\mathsf{AC}^0[2]$, except that *no* unbounded fan-in PARITY gates are allowed.

2.2 Leakage-Resilient Circuit

In this section, we recall the definitions of stateful function and LRC.

Definition 2.5 (Stateful function [13]). 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 2.6 (Leakage-resilient circuit [4]). For a stateful function family $\{\mathsf{F}_\lambda : \{0,1\}^{l_{\mathsf{st}}} \times \{0,1\}^{l_{\mathsf{in}}} \to \{0,1\}^{l_{\mathsf{st}}} \times \{0,1\}^{l_{\mathsf{out}}}\}$, an $\{\mathcal{L}_\lambda\}$-*leakage-resilient circuit (LRC)* consists of polynomial-sized function families $\{\mathsf{Setup}_\lambda, \mathsf{Comp}_\lambda\}$ such that

- Setup_λ on input an initial state $\mathbf{s}_0 \in \{0,1\}^{l_{\mathsf{st}}}$ of F_λ returns an encoding st_0,
- Comp_λ on input an encoded state st and an external input $\mathbf{x} \in \{0,1\}^{l_{\mathsf{in}}}$ returns 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 polynomial $q = q(\lambda)$, all initial state $\mathbf{s}_0 \in \{0,1\}^{l_{\mathsf{st}}}$ of F_λ, and all $\mathsf{st}_0 \in \mathsf{Setup}_\lambda(\mathbf{s}_0)$, the same outputs must be obtained by q stateful invocations of $\mathsf{F}_\lambda[\mathbf{s}_0]$ and $\mathsf{Comp}_\lambda[\mathsf{st}_0]$ on any sequence of external inputs $\mathbf{x}_1, \cdots, \mathbf{x}_q \in \{0,1\}^{l_{\mathsf{in}}}$.

Leakage-resilience is satisfied if there exists a polynomial-sized stateful simulator $\{\mathsf{Sim}_\lambda\}$ such that for any $\lambda \in \mathbb{N}$, any polynomial $q = q(\lambda)$, any (possibly unbounded) adversary $\mathcal{A}$, and any $\mathbf{s}_0 \in \{0,1\}^{l_{\mathsf{st}}}$, we have

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

where the experiments $\mathsf{Real}_{\mathcal{A}}$ and $\mathsf{Ideal}_{\mathcal{A}}$ are defined as in *Figure* 3.

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

Figure 3: Definitions of Real and Ideal.

$\underline{\mathsf{M}_\lambda(\mathbf{r} \in \{0,1\}^{\lambda(\lambda-1)/2})}$:
Parse $\mathbf{r} = (r_{i,j})_{1 \le i < j \le \lambda}$

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

$\underline{\mathsf{ZeroSamp}_\lambda}$:
$\mathbf{r} \xleftarrow{\$} \{0,1\}^{\lambda(\lambda-1)/2}$, $\widetilde{\mathbf{r}} \xleftarrow{\$} \{0,1\}^{\lambda-1}$
Return $\mathsf{M}_\lambda(\mathbf{r})||\mathsf{M}_\lambda(\mathbf{r})\widetilde{\mathbf{r}}$

$\underline{\mathsf{OneSamp}_\lambda}$:
$\mathbf{r} \xleftarrow{\$} \{0,1\}^{\lambda(\lambda-1)/2}$, $\widetilde{\mathbf{r}} \xleftarrow{\$} \{0,1\}^{\lambda-1}$
Return $\mathsf{M}_\lambda(\mathbf{r})||(\mathsf{M}_\lambda(\mathbf{r})\widetilde{\mathbf{r}} + \mathbf{e}_1)$

Figure 4: Definitions of M_λ, $\mathsf{ZeroSamp}_\lambda$, and $\mathsf{OneSamp}_\lambda$. Recall that $\mathbf{e}_1$ denotes the column vector with the first element equal to 1 and the others equal to 0's.

2.3 Basic Tools

We now recall several function families $\{\mathsf{M}_\lambda, \mathsf{ZeroSamp}_\lambda, \mathsf{OneSamp}_\lambda\}$ defined in [6, 24] as in Figure 4.

Note that in [6], $\mathsf{ZeroSamp}_\lambda$ and $\mathsf{OneSamp}_\lambda$ are originally defined as

$$(\mathbf{e}_1||\mathsf{M}_\lambda(\mathbf{r}))\begin{pmatrix} \mathbf{0} & 0 \\ \mathbf{I}_{\lambda-1} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{I}_{\lambda-1} & \widetilde{\mathbf{r}} \\ \mathbf{0} & 1 \end{pmatrix} \text{ and } (\mathbf{e}_1||\mathsf{M}_\lambda(\mathbf{r}))\begin{pmatrix} \mathbf{0} & 1 \\ \mathbf{I}_{\lambda-1} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{I}_{\lambda-1} & \widetilde{\mathbf{r}} \\ \mathbf{0} & 1 \end{pmatrix},$$

where $\mathbf{r} \xleftarrow{\$} \{0,1\}^{\lambda(\lambda-1)/2}$ and $\widetilde{\mathbf{r}} \xleftarrow{\$} \{0,1\}^{\lambda-1}$, respectively. These definitions are equivalent to ours up to a straightforward calculation, as also mentioned in the proof of Lemma 6 of [24]. Both $\mathsf{M}_\lambda(\mathbf{r})$ for any $\mathbf{r} \in \{0,1\}^{\lambda(\lambda-1)/2}$ and any $\mathbf{M} \in \mathsf{ZeroSamp}_\lambda$ have rank $\lambda - 1$, whereas $\mathsf{M}_\lambda(\mathbf{r}) \,\|\, \mathbf{e}_1$ and any $\mathbf{M} \in \mathsf{OneSamp}_\lambda$ are of full rank [6, 8, 24].

We further note that $\mathbf{X} = \mathbf{MR}$ for $\mathbf{M}^\top = \begin{pmatrix} \mathsf{M}_\lambda(\mathbf{r})^\top \\ \widetilde{\mathbf{r}}^\top \mathsf{M}_\lambda(\mathbf{r})^\top \end{pmatrix}^\top \xleftarrow{\$} \mathsf{ZeroSamp}_\lambda$ 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 $\mathsf{ZeroSamp}_\lambda$.

Lemma 2.7 (Lemma 3 in [8]). *For all $\lambda \in \mathbb{N}$ and all $\mathbf{M}^\top \in \mathsf{ZeroSamp}_\lambda$, there exists a unique non-zero vector $\mathbf{M}^\perp$ in $\mathsf{Ker}(\mathbf{M}^\top)$ such that $\mathbf{M}^\perp \in \{0,1\}^{\lambda-1} \times \{1\}$.*

By Lemma 2.7, for any $\mathbf{M}^\top \in \mathsf{ZeroSamp}_\lambda$ we have ${\mathbf{M}^\perp}^\top \mathbf{e}_\lambda \neq 0$, implying that $\mathbf{e}_\lambda \notin \mathsf{Span}(\mathbf{M})$. Hence, $\mathbf{M}||\mathbf{e}_\lambda$ must be of full rank.

Lemma 2.8 (Lemma 4 in [8]). *For all $\lambda \in \mathbb{N}$ and all $\mathbf{M}^\top \in \mathsf{ZeroSamp}_\lambda$, there exists a unique non-zero vector $\mathbf{k}$ in $\mathsf{Ker}(\mathbf{M})$, and we have $\mathbf{k} \in \{1\} \times \{0,1\}^{\lambda-1}$.*

Indistinguishability against NC^1. The following lemma shows that $\mathsf{ZeroSamp}_\lambda$ and $\mathsf{OneSamp}_\lambda$ are indistinguishable against 2-adaptive leakage functions in NC^1 with logarithmic output length. [3]

Lemma 2.9 (Lemma A.1 in [23]). *If $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$ holds, 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:*

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

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

Raz-type extractor against NC^1. Changing the parameter from λ to $\lambda + 1$, for $\mathbf{M}^\top \xleftarrow{\$} \mathsf{ZeroSamp}_{\lambda+1}$ and $\mathbf{M}'^\top \xleftarrow{\$} \mathsf{OneSamp}_{\lambda+1}$, we have that $\overline{\mathbf{M}}$ and $\mathbf{M}'$ are of full rank and $\underline{\mathbf{M}}$ is the linear combination of the rows in $\overline{\mathbf{M}}$, as discussed above. As a result, for any polynomial $l = l(\lambda)$ in λ, the following two distributions are identical.

$$(\mathbf{MS}, \mathbf{M}'\mathbf{S}') : \mathbf{M}^\top \xleftarrow{\$} \mathsf{ZeroSamp}_{\lambda+1}, \mathbf{M}'^\top \xleftarrow{\$} \mathsf{OneSamp}_{\lambda+1}, \mathbf{S}, \mathbf{S}' \xleftarrow{\$} \{0,1\}^{(\lambda+1)\times l},$$

[3] Here, we note that if we adopt the infinitely-often version of $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$, holding only for an infinitely large number of values of λ, then our schemes given later satisfy *infinitely-often security*.

$$\left(\begin{pmatrix}\mathbf{R}\\ \widetilde{\mathbf{r}}^\top\mathbf{R}\end{pmatrix},\begin{pmatrix}\mathbf{R}'\\ \mathbf{r}^\top\end{pmatrix}\right):\mathbf{R},\mathbf{R}'\xleftarrow{\$}\{0,1\}^{\lambda\times l},\widetilde{\mathbf{r}}\xleftarrow{\$}\{0,1\}^{\lambda},\mathbf{r}\xleftarrow{\$}\{0,1\}^{l}.$$

Then the following lemma immediately follows from Lemma 2.9.

Lemma 2.10 *If* $\mathsf{NC}^1\subsetneq\oplus\mathsf{L}/\mathsf{poly}$ *holds, then for all* $\{\ell_\lambda:\{0,1\}^{(\lambda+1)\times l}\to\{0,1\}^{O(\log\lambda)}\}\in\mathsf{NC}^1$*, all* $\lambda\in\mathbb{N}$*, and all polynomial* $l=l(\lambda)$*, we have* $\Delta\big(\ell_\lambda(\mathbf{R},\widetilde{\mathbf{r}}^\top\mathbf{R}),\ell_\lambda(\mathbf{R},\mathbf{r}^\top)\big)\le\mathsf{negl}(\lambda)$*, where* $\mathbf{R}\xleftarrow{\$}\{0,1\}^{\lambda\times l}$*,* $\widetilde{\mathbf{r}}\xleftarrow{\$}\{0,1\}^{\lambda}$*, and* $\mathbf{r}\xleftarrow{\$}\{0,1\}^{l}$*.*

Remark. Indeed, since NC^1 is contained in L (i.e., the complexity class of decision problems that can be solved by a deterministic Turing machine using a logarithmic amount of writable memory space), Lemma 2.10 is closely related to Raz's space lower bound for learning parities [18, 12], which shows that the statistical distance bound in Lemma 2.10 holds for ℓ_λ being any distinguisher with storage limited to $c\cdot\lambda^2$ for some constant c. In this work, however, the exact statement needed below is straightforwardly derived from Lemma 2.9.

3 2-Adaptive sAI-LTC

In this section, we first define the notion of a 2-adaptive sAI-LTC. We then recall the gate gadgets together with their reconstruction procedures, and present the decoding procedure that uses the ciphertext bank. Finally, using these ingredients, we construct a 2-adaptive sAI-LTC secure against NC^1 under $\mathsf{NC}^1\subsetneq\oplus\mathsf{L}/\mathsf{poly}$.

3.1 Definition

A 2-adaptive sAI-LTC can be treated as an extension of the AI-LTC defined in [22], which converts a circuit into a leakage-tolerant one by additionally using an auxiliary input generated in a leak-free setup. Similar to AI-LTC, in an sAI-LTC, all wire assignments of the computation can be simulated using only the input/output pair, with restricted complexity. The main difference is that an sAI-LTC additionally tolerates 2-adaptive leakage, and its setup size must be independent of the original function.

Definition 3.1 (2-adaptive leakage-tolerant circuit with succinct setup). For a function family $\{\mathsf{F}_\lambda:\{0,1\}^{l_{\mathsf{in}}}\to\{0,1\}^{l_{\mathsf{out}}}\}$, a *2-adaptive* $\{\mathcal{L}_\lambda\}$*-leakage-tolerant circuit with succinct setup (sAI-LTC)* consists of polynomial-sized function families $\{\mathsf{LTSetup}_\lambda,\mathsf{LTComp}_\lambda\}$ such that

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

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 $\mathsf{LTComp}_\lambda(\mathsf{aux},\mathbf{x})=\mathsf{F}_\lambda(\mathbf{x})$.

2-adaptive leakage-tolerance is satisfied if there exists a family of polynomial-time samplable distributions $\{\mathsf{DSim}_\lambda\}$ supported on $\{0,1\}^{l_{\mathsf{aux}}}\times\widetilde{\mathcal{L}}_\lambda$ (see Section 2.1 for the notation) such that for all $\lambda\in\mathbb{N}$, all $\ell_\lambda\in\mathcal{L}_\lambda$, and all functions $\mathcal{A}$ with range $\{0,1\}^{l_{\mathsf{in}}}\times\mathcal{L}_\lambda$, the following two distributions are statistically close with negligible distance:

$$\big(\ell_\lambda(\mathsf{aux}),\ell'_\lambda(\mathsf{W}[\mathsf{LTComp}_\lambda(\mathsf{aux},\mathbf{x})])\big):\mathsf{aux}\xleftarrow{\$}\mathsf{LTSetup}_\lambda,(\mathbf{x},\ell'_\lambda)=\mathcal{A}(\ell_\lambda(\mathsf{aux})),$$
$$\big(\ell_\lambda(\mathsf{aux}),\ell'_\lambda(\mathsf{LTSim}_\lambda(\mathbf{x},\mathsf{F}_\lambda(\mathbf{x})))\big):(\mathsf{aux},\mathsf{LTSim}_\lambda)\xleftarrow{\$}\mathsf{DSim}_\lambda,(\mathbf{x},\ell'_\lambda)=\mathcal{A}(\ell_\lambda(\mathsf{aux})).$$

Succinctness is satisfied if $|\mathsf{LTSetup}_\lambda|$ depends only on λ, and is independent of $|\mathsf{F}_\lambda|$, l_{in}, and l_{out}.

(Non-adaptive) sAI-LTC is defined in the same way, except that the first leakage function is omitted, equivalently the output of ℓ_λ is restricted to $\perp$.

3.2 Gate Gadgets with Reconstruction Procedures

In Figure 5, we recall the addition and multiplication gate gadgets $\{\mathsf{Add}_\lambda,\mathsf{Mult}_\lambda\}\in\mathsf{AC}^0[2]$ that are reconstructible using simulators $\{\mathsf{SimAdd}_\lambda,\mathsf{SimMult}_\lambda\}\in\mathsf{AC}^0[2]$, as defined in [22].

Briefly, in Figure 5, the addition gate gadget Add_λ and the multiplication gate gadget Mult_λ respectively perform additive and multiplicative homomorphic evaluations under the WP-FHE scheme of [24]. $\mathbf{W}_{d_1}$

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

Figure 5: Definition of $\{\mathsf{Add}_\lambda, \mathsf{Mult}_\lambda, \mathsf{SimAdd}_\lambda, \mathsf{SimMult}_\lambda\}$. All matrices are in $\{0,1\}^{\lambda\times\lambda}$.

and $\mathbf{W}_{d_2}$ denote the input ciphertexts, $\mathbf{W}_{d_3}$ denotes the output ciphertext, and $\mathbf{Z}_{d_1}, \mathbf{Z}_{d_2}, \mathbf{Z}_{d_3}$ denote the internal randomness of these ciphertexts. $\mathbf{MR}$ denotes a ciphertext of 0 used for refreshing. These gate gadgets also perform refreshing and can be simulated in NC^1 using SimAdd_λ and $\mathsf{SimMult}_\lambda$, given the internal randomness and plaintexts of the ciphertexts. This ensures that, for homomorphic evaluation over an arbitrary polynomial-sized circuit, the pre-decoding wires can be simulated in NC^1 given $\mathbf{M}$, the corresponding wire values of the original computation, and the randomness of the simulated ciphertexts, which will be crucial in the subsequent security proof for our sAI-LTC.

Lemma 3.2 (Lemma 3.2 in [23]). *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*

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

is identical to that of

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

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

3.3 Decoding Procedure with Ciphertext Bank

We now recall the decoding algorithm $\{\mathsf{Dec}_\lambda\}$ and its simulator $\{\mathsf{SimDec}_\lambda\}$ from [22], as defined in Figure 6, and later introduce a new simulation method based on a ciphertext bank mechanism.

At a high level, given a vector $\mathbf{y}$ that can be viewed as a DVV ciphertext under a public key $\mathbf{M}^\top \in \mathsf{ZeroSamp}_\lambda$, Dec_λ determines whether $\mathbf{y} \in \mathsf{Span}(\mathbf{M})$, which suffices to recover the corresponding plaintext bit. The simulator SimDec_λ is designed to fool leakage on the wires of Dec_λ. While the complexity of Dec_λ is beyond NC^1, SimDec_λ runs in $\mathsf{AC}^0[2]$. As in [22], for notational simplicity we write $\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{Y})$ to denote applying Dec_λ to each vector $\mathbf{y}_{i,j}$ in $\mathbf{Y}$, and use SimDec_λ analogously (see the caption of Figure 6).

The following lemma implies the correctness of the decoding algorithm.

Lemma 3.3 (Lemma 4.1 in [23]). *For all* $\lambda \in \mathbb{N}$, *all* $\mathbf{M}^\top \in \mathsf{ZeroSamp}_\lambda$, *all* $\mathbf{z} \in \{0,1\}^\lambda$, *and all* $b \in \{0,1\}$, *we have* $b = \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{Mz} + \mathbf{e}_\lambda \cdot b)$.

Here, $\mathbf{Mz} + \mathbf{e}_\lambda \cdot b$ is in the form of a DVV ciphertext with the plaintext b and internal randomness $\mathbf{z}$, and Lemma 3.3 guarantees that Dec_λ correctly recovers b.

Simulation via ciphertext bank. In [23], SimDec_λ is shown to fool leakage when given randomness that is uniform conditioned on the DVV ciphertext being decoded. Our core theorem extends this to polynomially many fixed ciphertexts after they are refreshed using random linear combinations from a fixed-size ciphertext bank: leakage from decoding can be simulated from the refreshed ciphertext randomness,

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

Parse $\overline{\mathbf{M}} = \mathsf{M}_\lambda(\mathbf{r})^\top$ 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 $x_j = y_{j-1} - x_1 r_{1,j} - x_2 r_{2,j} - \cdots - x_{j-1} r_{j-1,j}$

return $b = \underline{\mathbf{y}} - \underline{\mathbf{M}}\mathbf{x}$ where $\mathbf{x} = (x_i)_{i\in[\lambda]}$

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

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

Set $\mathbf{y} = (y_i)_{i\in[\lambda]} = \mathbf{M}\mathbf{x} + \mathbf{e}_\lambda \cdot b$

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 6: Definitions of $\{\mathsf{Dec}_\lambda\}$ and $\{\mathsf{SimDec}_\lambda\}$. The function family $\{\mathsf{M}_\lambda\}$ is defined in Section 2.3. Throughout the rest of the paper, we slightly abuse notation by writing $\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{Y})$ to denote $\big(\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{y}_{i,j})\big)_{i\in[m],\, j\in[n]}$, and similarly writing $\mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{X}, \mathbf{B})$ to denote $\big(\mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{x}_{i,j}, b_{i,j})\big)_{i\in[m],\, j\in[n]}$, where $\mathbf{Y} = (\mathbf{y}_{i,j})_{i\in[m],\, j\in[n]}$, $\mathbf{X} = (\mathbf{x}_{i,j})_{i\in[m],\, j\in[n]}$, and $\mathbf{B} = (b_{i,j})_{i\in[m],\, j\in[n]}$. (Recall that W returns all the intermediate wire values of a computation and $\mathbf{e}_\lambda$ denotes the column vector with the last element equal to 1 and the others equal to 0's. See the "Notation" paragraph in Section 2).

namely the original ciphertext randomness plus the corresponding random linear combinations of the bank randomness.

Let $\mathcal{L}_\lambda$ be the class of all function families in NC^1 with output length $O(\log \lambda)$. Our core theorem is as follows.

Theorem 3.4 (Core Theorem). *If $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$ holds, then for all $\lambda \in \mathbb{N}$, all polynomial $l = l(\lambda)$, all $\ell_\lambda \in \mathcal{L}_\lambda$, all $\mathbf{M}^\top \in \mathsf{ZeroSamp}_\lambda$, all $\mathbf{Z} \in \{0,1\}^{\lambda\times l}$, and all $\mathbf{y} \in \{0,1\}^l$, the following two distributions are statistically close with negligible distance:*

$$\Big(\ell_\lambda\big(\mathsf{W}[\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{Y} + \mathbf{U}\mathbf{R})], \mathbf{R}\big), \mathbf{U}\Big)$$

$$\text{and } \Big(\ell_\lambda\big(\mathsf{W}[\mathbf{Y} + \mathbf{U}\mathbf{R}], \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{Z} + \mathbf{V}\mathbf{R}, \mathbf{y}^\top), \mathbf{R}\big), \mathbf{U}\Big)$$

where $\mathbf{V} \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$, $\mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times l}$, $\mathbf{Y} = \mathbf{M}\mathbf{Z} + \mathbf{e}_\lambda \cdot \mathbf{y}^\top$, and $\mathbf{U} = \mathbf{M}\mathbf{V}$.

To interpret the theorem, note that $\mathbf{U} = \mathbf{M}\mathbf{V}$ is the ciphertext bank: its columns are DVV ciphertexts of 0 with randomness given by the columns of $\mathbf{V}$. Multiplying by the public random matrix $\mathbf{R}$ gives refresh ciphertexts $\mathbf{U}\mathbf{R} = \mathbf{M}\mathbf{V}\mathbf{R}$, and hence $\mathbf{Y} + \mathbf{U}\mathbf{R} = \mathbf{M}(\mathbf{Z} + \mathbf{V}\mathbf{R}) + \mathbf{e}_\lambda \cdot \mathbf{y}^\top$. Thus $\mathbf{Z} + \mathbf{V}\mathbf{R}$ is the refreshed randomness. The theorem states that, after this refresh, leakage from real decoding can be simulated from this refreshed randomness.

Proof sketch. To prove the theorem, we first show that the decoding wires can be simulated in NC^1 using $\mathbf{Z} + \mathbf{V}\mathbf{R} + \mathbf{k} \cdot \mathbf{r}^\top$, where $\mathbf{k}$ is the unique non-zero vector in $\ker(\mathbf{M})$ and $\mathbf{r}$ is uniformly random. This has exactly the distribution of the randomness exposed in the real decoding phase, conditioned on $\mathbf{Y} + \mathbf{U}\mathbf{R}$. We then use the Raz-type extractor statement in Lemma 2.10 to replace $\mathbf{r}^\top$ by $\mathbf{b}^\top\mathbf{R}$, where $\mathbf{b}$ is a short uniformly random vector. Finally, the extra term $\mathbf{k} \cdot \mathbf{b}^\top\mathbf{R}$ is absorbed by re-randomizing $\mathbf{V}$: replacing $\mathbf{V}$ by $\mathbf{V} + \mathbf{k} \cdot \mathbf{b}^\top$ preserves both the distribution of $\mathbf{V}$ and the product $\mathbf{M}\mathbf{V}$. Thus, leakage on the decoding wires is reduced to leakage on $\mathbf{Z} + \mathbf{V}\mathbf{R}$.

Proof. Let $\lambda \in \mathbb{N}$ be the security parameter and $\mathbf{M}^\top$ be any matrix in $\mathsf{ZeroSamp}_\lambda$. There exists a unique non-zero vector $\mathbf{k}$ in $\mathsf{Ker}(\mathbf{M})$, according to Lemma 2.8. We then prove the following lemma, which we will use later.

Lemma 3.5 *For all* $\mathbf{z} \in \{0,1\}^\lambda$ *and all* $b \in \{0,1\}$*, we have*

$$\Delta(\mathsf{W}[\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{y})], \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{z} + \mathbf{k} \cdot r, b)) = 0,$$

where $r \xleftarrow{\$} \{0,1\}$ *and* $\mathbf{y} = \mathbf{M}\mathbf{z} + \mathbf{e}_\lambda \cdot b$*.*

Proof of Lemma 3.5. Since the rank of $\mathbf{M}$ is $\lambda - 1$ and $\mathbf{k}$ is the unique vector in $\mathsf{Ker}(\mathbf{M})\backslash\{0\}$, for fixed $\mathbf{y}$, $\mathbf{M}$, and b, there exist two solutions $\mathbf{w} = \mathbf{z}$ and $\mathbf{w} = \mathbf{z} + \mathbf{k}$ to the equation $\mathbf{y} = \mathbf{M}\mathbf{w} + \mathbf{e}_\lambda \cdot b$. Namely, $\mathbf{w} = \mathbf{z} + \mathbf{k} \cdot r$ for $r \xleftarrow{\$} \{0,1\}$ is uniformly distributed conditioned on $\mathbf{y} = \mathbf{M}\mathbf{w} + \mathbf{e}_\lambda \cdot b$.

On the other hand, the vector $\mathbf{x}$ generated in the intermediate steps of Dec_λ (see Figure 6) is uniformly distributed when conditioned on $\overline{\mathbf{y}} = \overline{\mathbf{M}}\mathbf{x}$, and 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.

As a result, $\mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{z} + \mathbf{k} \cdot r, b)$ for $r \xleftarrow{\$} \{0,1\}$ perfectly simulates the distribution of $\mathsf{W}[\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{y})]$, completing the proof of Lemma 3.5. □

Next, for any $\ell_\lambda \in \mathcal{L}_\lambda$, any polynomial $l = l(\lambda)$, any $\mathbf{Z} \in \{0,1\}^{\lambda \times l}$, any $\mathbf{y} \in \{0,1\}^l$, and $\mathbf{Y} = \mathbf{M}\mathbf{Z} + \mathbf{e}_\lambda \cdot \mathbf{y}^\top$, we proceed with the proof via a sequence of intermediate games. In each game, $\big(\ell_\lambda(\mathcal{W}, \mathbf{R}), \mathbf{U}\big)$ is generated as in Figure 7.

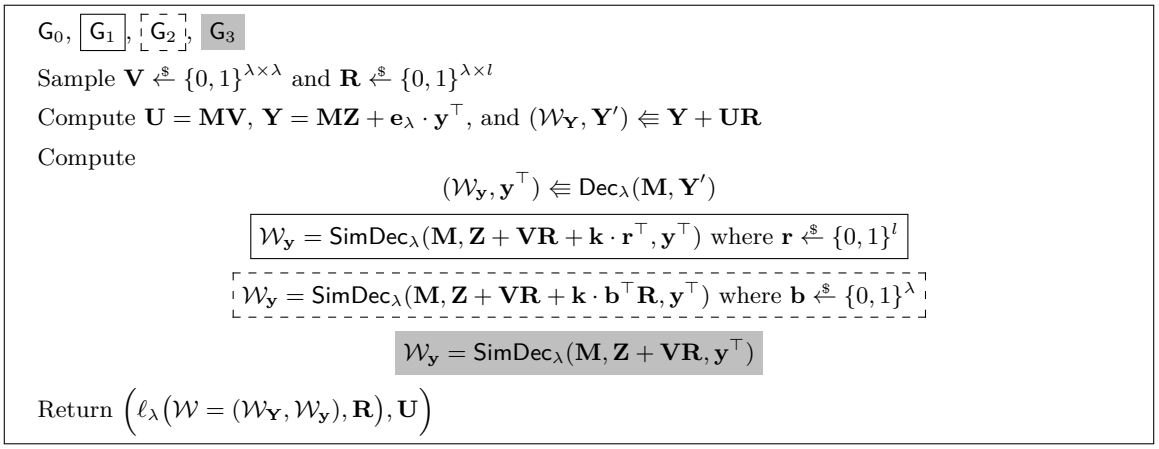

Figure 7: Definitions of $\mathcal{W}$ in $\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3$.

Game G_0. In G_0, $\mathbf{U}$ is generated as $\mathbf{U} = \mathbf{M}\mathbf{V}$ for $\mathbf{V} \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$, and ℓ_λ takes as input $\mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times l}$ and $\mathcal{W} = (\mathcal{W}_\mathbf{Y}, \mathcal{W}_\mathbf{y})$ generated as

$$(\mathcal{W}_\mathbf{Y}, \mathbf{Y}') \Leftarrow \mathbf{Y} + \mathbf{U}\mathbf{R} \text{ and } (\mathcal{W}_\mathbf{y}, \mathbf{y}^\top) \Leftarrow \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{Y}').$$

Game G_1. In G_1, $\mathcal{W}_\mathbf{y}$ is generated as $\mathcal{W}_\mathbf{y} = \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{Z} + \mathbf{V}\mathbf{R} + \mathbf{k} \cdot \mathbf{r}^\top, \mathbf{y}^\top)$, where $\mathbf{r} \xleftarrow{\$} \{0,1\}^l$ instead.

Lemma 3.6 *The distribution of* $\big(\ell_\lambda(\mathcal{W}, \mathbf{R}), \mathbf{U}\big)$ *in* G_0 *is identical to that in* G_1*.*

Proof. In G_0, we have

$$\mathbf{Y}' = \mathbf{Y} + \mathbf{U}\mathbf{R} = \mathbf{M}\mathbf{Z} + \mathbf{e}_\lambda \cdot \mathbf{y}^\top + \mathbf{M}\mathbf{V}\mathbf{R} = \mathbf{M}(\mathbf{Z} + \mathbf{V}\mathbf{R}) + \mathbf{e}_\lambda \cdot \mathbf{y}^\top.$$

Then for $\mathbf{r} \xleftarrow{\$} \{0,1\}^l$, the distribution of $\mathsf{W}[\mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{Y}')]$ is identical to that of $\mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{Z} + \mathbf{V}\mathbf{R} + \mathbf{k} \cdot \mathbf{r}^\top, \mathbf{y}^\top)$, according to Lemma 3.5, completing the proof of Lemma 3.6. □

Game G_2. G_2 is the same as G_1 except that $\mathcal{W}_\mathbf{y}$ is generated as

$$\mathcal{W}_\mathbf{y} = \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{Z} + \mathbf{V}\mathbf{R} + \mathbf{k} \cdot \mathbf{b}^\top\mathbf{R}, \mathbf{y}^\top).$$

Lemma 3.7 *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$ *holds, then the distribution of* $\big(\ell_\lambda(\mathcal{W}, \mathbf{R}), \mathbf{U}\big)$ *in* G_1 *is statistically close to that in* G_2 *with negligible distance.*

Proof. Fixing all the seeds except for $\mathbf{r}$, $\mathbf{b}$, and $\mathbf{R}$ in G_1 and G_2, such that the statistical distance of the two distributions is maximized, one can see that no information about $(\mathbf{r}^\top, \mathbf{R})$ or $(\mathbf{b}^\top\mathbf{R}, \mathbf{R})$ is revealed beyond the NC^1 leakage, since the multiplication of two vectors or matrices, the addition of matrices, and SimDec_λ are in $\mathsf{AC}^0[2]$, and $\{\ell_\lambda\} \in \mathsf{NC}^1$. Then Lemma 3.7 follows immediately from Lemma 2.10. □

Game G_3. G_3 is the same as G_2 except that $\mathcal{W}_{\mathbf{y}}$ is generated as $\mathcal{W}_{\mathbf{y}} = \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{Z} + \mathbf{VR}, \mathbf{y}^\top)$ instead of $\mathcal{W}_{\mathbf{y}} = \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{Z} + \mathbf{VR} + \mathbf{k} \cdot \mathbf{b}^\top\mathbf{R}, \mathbf{y}^\top)$.

Lemma 3.8 *The distribution of $\big(\ell_\lambda(\mathcal{W}, \mathbf{R}), \mathbf{U}\big)$ in G_2 is identical to that in G_3.*

Proof. Since $\mathbf{Mk} = \mathbf{0}$, we have

$$\big(\mathbf{MV}, \mathbf{VR} + \mathbf{k} \cdot \mathbf{b}^\top\mathbf{R}\big) = \big(\mathbf{M}(\mathbf{V} + \mathbf{k} \cdot \mathbf{b}^\top), (\mathbf{V} + \mathbf{k} \cdot \mathbf{b}^\top)\mathbf{R}\big)$$

Moreover, for $\mathbf{V} \xleftarrow{\$} \{0,1\}^{\lambda \times \lambda}$, the following two distributions are identical:

$$\big(\mathbf{M}(\mathbf{V} + \mathbf{k} \cdot \mathbf{b}^\top), (\mathbf{V} + \mathbf{k} \cdot \mathbf{b}^\top)\mathbf{R}\big) \text{ and } (\mathbf{MV}, \mathbf{VR}).$$

Therefore, for $\mathbf{V} \xleftarrow{\$} \{0,1\}^{\lambda \times \lambda}$, the following distributions are identical as well:

$$\big(\mathbf{MV}, \mathbf{VR} + \mathbf{k} \cdot \mathbf{b}^\top\mathbf{R}\big) \text{ and } (\mathbf{MV}, \mathbf{VR}).$$

Hence, the distributions of $\big(\ell_\lambda(\mathcal{W}, \mathbf{R}), \mathbf{U}\big)$ in G_2 and G_3 are identical, completing the proof of Lemma 3.8. □

Putting all the above together, we have that the distributions of $\big(\ell_\lambda(\mathcal{W}, \mathbf{R}), \mathbf{U}\big)$ in G_0 and G_3 are statistically close with negligible distance under the assumption $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$, completing the proof of Theorem 3.4. □

3.4 Construction of 2-Adaptive sAI-LTC

We now show how to construct a 2-adaptive sAI-LTC using the gate gadgets and the decoding procedure with ciphertext banks.

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 family. We give our construction of 2-adaptive $\{\mathcal{L}_\lambda\}$-sAI-LTC for $\{\mathsf{F}_\lambda : \{0,1\}^{l_{\mathsf{in}}} \to \{0,1\}^{l_{\mathsf{out}}}\}$, denoted by $\mathsf{sAI\text{-}LTC}$, in Figure 8.

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

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathsf{ZeroSamp}_\lambda$ and $\mathbf{V} \xleftarrow{\$} \{0,1\}^{\lambda \times \lambda}$, and compute $\mathbf{U} = \mathbf{MV}$

Output $\mathsf{aux} = (\mathbf{M}, \mathbf{U})$

$\underline{\mathsf{LTComp}_\lambda(\mathsf{aux} = (\mathbf{M}, \mathbf{U}), \mathbf{x})}$:

The ith input gate x_i in F_λ is implemented as $\mathbf{MZ}_i + x_i \otimes \mathbf{I}_\lambda$ 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

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

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

Set $\mathbf{Y} = (\mathbf{y}_i)_{i \in [l_{\mathsf{out}}]}$, where $\mathbf{y}_i$ is the rightmost column of the matrix representing the wire bundle for the ith final output for each $i \in [l_{\mathsf{out}}]$

Compute $\mathbf{y}^\top = \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{Y} + \mathbf{UR})$ where $\mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda \times l_{\mathsf{out}}}$

Return $\mathbf{y}$

Figure 8: Definition of $\mathsf{sAI\text{-}LTC} = \{\mathsf{LTSetup}_\lambda, \mathsf{LTComp}_\lambda\}$. $\{\mathsf{Add}_\lambda, \mathsf{Mult}_\lambda, \mathsf{Dec}_\lambda\}$ are the function families defined as in Figures 5 and 6.

Theorem 3.9 *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$ *holds, then* $\mathsf{sAI\text{-}LTC}$ *is an* $\{\mathcal{L}_\lambda\}$*-sAI-LTC for* $\{\mathsf{F}_\lambda\}$.

Proof sketch. By Theorem 3.4, after refresh, the decoding wires can be simulated in NC^1 from $\mathbf{M}$, the output value, and the refreshed randomness of the output ciphertexts. Before decoding, each wire value w_i of the original computation is represented by a WP-FHE wire bundle $\mathbf{MZ}_i + w_i \otimes \mathbf{I}_\lambda$. By Lemma 3.2, the internal wires of the addition and multiplication gadgets are reconstructible in NC^1 from the corresponding wire bundles and randomness. Thus, at this intermediate stage, leakage on all wires can be simulated by an NC^1 procedure depending on $\mathbf{M}$ and the original computation wires, without revealing additional information about the kernel vector $\mathbf{k}$. This allows us to switch $\mathbf{M}$ from $\mathsf{ZeroSamp}_\lambda$ to $\mathsf{OneSamp}_\lambda$. Under $\mathsf{OneSamp}_\lambda$, $\mathbf{M}$ is full rank, and thus a random bundle $\mathbf{MZ}_i + w_i \otimes \mathbf{I}_\lambda$ is uniformly distributed and hides w_i. Moreover, using $\mathbf{M}^{-1}$, the simulator can sample input and output wire bundles and compute internal randomness making them consistent with the given input and output values. Hence, the entire wire distribution of the sAI-LTC can be simulated from the input/output pair alone.

Proof. **Succinctness.** Succinctness follows since sampling a matrix from $\mathsf{ZeroSamp}_\lambda$ together with λ vectors in $\mathsf{Span}(\mathbf{M})$ can be performed with $O(\lambda^2)$ overhead, by reusing the intermediate values generated during the sampling of $\mathsf{ZeroSamp}_\lambda$, as discussed in Section 2.3.

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

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

into ones of the form

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

Therefore, for each $i \in [l_{\mathsf{out}}]$, we must have

$$\mathbf{Y}_i = \mathbf{MZ}'_i + y_i \otimes \mathbf{I}_\lambda \text{ for some } \mathbf{Z}'_i \in \{0,1\}^{\lambda \times \lambda},$$

where y_i is the ith bit of the correct output $\mathbf{y}$. Moreover, we have

$$\mathbf{y}_i + \mathbf{Ur}_i = \mathbf{Mz}_i + \mathbf{e}_\lambda \cdot y_i \text{ for some } \mathbf{z}_i \in \{0,1\}^\lambda,$$

since $\mathbf{Ur}_i \in \mathsf{Span}(\mathbf{M})$. As a result, we must have

$$\mathbf{y}^\top = \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{Y} + \mathbf{UR})$$

when $\mathbf{M}^\top \in \mathsf{ZeroSamp}_\lambda$, according to Lemma 3.3, completing the proof of correctness.

2-adaptive leakage-tolerance. We give the description of the distribution family $\{\mathsf{DSim}_\lambda\}$ in Figure 9. Note that for every $(\mathsf{aux}, \mathsf{LTSim}_\lambda)$ in the support of DSim_λ, LTSim_λ runs in $\mathsf{AC}^0[2]$, since it involves only sampling random bits and performing multiplications of a constant number of matrices in parallel.

Next, for any security parameter $\lambda \in \mathbb{N}$, any $\ell_\lambda \in \mathcal{L}_\lambda$, and any function $\mathcal{A}$, we define a sequence of hybrid games $\mathsf{G}_0, \ldots, \mathsf{G}_6$. Here, G_0 and G_6 correspond to the real and simulated games respectively, as defined in Definition 3.1. For $\mathsf{G}_1, \ldots, \mathsf{G}_5$, the auxiliary input and the assignments to all wires are generated as in Figure 10.

Game G_0**.** This is the real game, where ℓ_λ takes as input aux for $\mathsf{aux} \xleftarrow{\$} \mathsf{LTSetup}_\lambda$ and outputs some bit string $\mathbf{b}$, $\mathcal{A}$ on input $\mathbf{b}$ outputs some $\ell'_\lambda \in \mathcal{L}_\lambda$ and some $\mathbf{x} \in \{0,1\}^{l_{\mathsf{in}}}$ where $\mathbf{y} = \mathsf{F}_\lambda(\mathbf{x})$, and ℓ'_λ on input $\mathcal{W} \xleftarrow{\$} \mathsf{W}[\mathsf{LTComp}_\lambda(\mathsf{aux}, \mathbf{x})]$ outputs some leaked bit string $\mathbf{b}'$.

Game G_1**.** G_1 differs from G_0 in that to generate $\mathcal{W}$, the challenger first generates the ciphertext $\mathbf{W}_i$ for each wire in F_λ (in parallel), and then simulates the wire assignments of each addition and multiplication gate gadget using $\{\mathsf{SimAdd}_\lambda, \mathsf{SimMult}_\lambda\}$.

Lemma 3.10 *The distributions of* $(\ell_\lambda(\mathsf{aux}), \ell'_\lambda(\mathcal{W}))$ *in* G_0 *and* G_1 *are identical.*

Lemma 3.10 follows immediately from Lemma 3.2.

Game G_2**.** In G_2, $\mathcal{W}_\mathbf{y}$ is now generated as $\mathcal{W}_\mathbf{y} = \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{Z} + \mathbf{VR}, \mathbf{y}^\top)$.

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

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathsf{OneSamp}_\lambda$ and $\mathbf{V} \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ and compute $\mathbf{U} = \mathbf{MV}$

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})

Sample $\mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times l_{\mathsf{out}}}$ and include $\mathbf{R}$ in $\mathcal{W}$

Return $(\mathsf{aux} = (\mathbf{M}, \mathbf{U}), \mathsf{LTSim}_\lambda)$, where LTSim_λ is defined below with $(\mathcal{W}, \mathbf{V}, \mathbf{M}^{-1})$ hardwired into it

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

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

Set $\mathbf{Y} = (\mathbf{y}_i)_{i\in[l_{\mathsf{out}}]}$, where $\mathbf{y}_i$ is the rightmost column of the matrix representing the wire bundles for the ith final output for each $i \in [l_{\mathsf{out}}]$

Include $(\mathsf{W}[\mathbf{Y} + \mathbf{UR}], \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{M}^{-1}(\mathbf{Y} - \mathbf{e}_\lambda \cdot \mathbf{y}^\top) + \mathbf{VR}, \mathbf{y}^\top))$ in $\mathcal{W}$

Return $\mathcal{W}$

Figure 9: Definition of $\{\mathsf{DSim}_\lambda\}$. $\{\mathsf{SimDec}_\lambda\}$ is the function family defined as in Figure 6.

Lemma 3.11 *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$ *holds, then the distribution of* $(\ell_\lambda(\mathsf{aux}), \ell'_\lambda(\mathcal{W}))$ *in* G_1 *is statistically close to that in* G_2 *with negligible distance.*

Proof. Fix all the seeds except for $\mathbf{R}$, $\mathbf{V}$, and the internal randomness of Dec_λ in G_1 and G_2, in a way that the statistical distance between the leakage in the two games is maximized. One can see that the output of ℓ'_λ reveals no information about $\mathcal{W}_{\mathbf{y}}$ and $\mathbf{R}$ beyond their NC^1 leakage, since LTSim_λ runs in $\mathsf{AC}^0[2]$, and ℓ'_λ runs in NC^1. Moreover, beyond the leakage produced by ℓ'_λ, no additional information about $\mathbf{V}$ is revealed except for $\mathbf{U} = \mathbf{MV}$. Then Lemma 3.11 follows immediately from Theorem 3.4. □

Game G_3**.** G_3 is the same as G_2 except that $\mathbf{M}^\top$ is sampled as $\mathbf{M}^\top \xleftarrow{\$} \mathsf{OneSamp}_\lambda$ instead of $\mathbf{M}^\top \xleftarrow{\$} \mathsf{ZeroSamp}_\lambda$.

Lemma 3.12 *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$ *holds, then the distribution of* $(\ell_\lambda(\mathsf{aux}), \ell'_\lambda(\mathcal{W}))$ *in* G_2 *is statistically close to that in* G_3 *with negligible distance.*

Proof. In G_2 and G_3, $\ell_\lambda(\mathsf{aux})$ and $\ell'_\lambda(\mathcal{W})$ (with $\mathsf{W}[\mathsf{F}_\lambda(\mathbf{x})]$ hardwired in ℓ'_λ) are computable in NC^1 given $\mathbf{M}$, since operations for all addition and multiplication gate gadgets are performed in parallel in $\mathsf{AC}^0[2]$, and we have $\{\mathsf{SimDec}_\lambda\} \in \mathsf{AC}^0[2]$. Therefore, the leakage in each game can be treated as 2-adaptive NC^1 leakage on $\mathbf{M}$. Then Lemma 3.12 follows immediately from Lemma 2.9. □

Game G_4**.** G_4 is the same as G_3 except that $\mathbf{Y} = (\mathbf{y}_i)_{i\in[l_{\mathsf{out}}]}$ and $\mathbf{Z} = (\mathbf{z}_i)_{i\in[l_{\mathsf{out}}]}$ (i.e., the rightmost columns of $(\mathbf{Y}_i)_{i\in[l_{\mathsf{out}}]}$ and $(\mathbf{Z}_{\mathsf{out}_i})_{i\in[l_{\mathsf{out}}]}$ respectively) are respectively generated as $\mathbf{Y} \xleftarrow{\$} \{0,1\}^{\lambda\times l_{\mathsf{out}}}$ and $\mathbf{Z} = \mathbf{M}^{-1}(\mathbf{Y} - \mathbf{e}_\lambda \cdot \mathbf{y}^\top)$.

Lemma 3.13 *The distributions of* $(\ell_\lambda(\mathsf{aux}), \ell'_\lambda(\mathcal{W}))$ *in* G_3 *and* G_4 *are identical.*

Proof. In G_3, we have $\mathbf{y}_i = \mathbf{Mz}_i + \mathbf{e}_\lambda \cdot y_i$ for all $i \in [l_{\mathsf{out}}]$, i.e., $\mathbf{Y} = \mathbf{MZ} + \mathbf{e}_\lambda \cdot \mathbf{y}^\top$. Then Lemma 3.13 follows immediately from the fact that $\mathbf{M}^\top \in \mathsf{OneSamp}_\lambda$ is of full rank and the following distributions are the same:

$$(\mathbf{Z}, \mathbf{MZ} + \mathbf{e}_\lambda \cdot \mathbf{y}^\top) \text{ and } (\mathbf{M}^{-1}(\mathbf{Y} - \mathbf{e}_\lambda \cdot \mathbf{y}^\top), \mathbf{Y})$$

where $\mathbf{Z} \xleftarrow{\$} \{0,1\}^{\lambda\times l_{\mathsf{out}}}$ and $\mathbf{Y} \xleftarrow{\$} \{0,1\}^{\lambda\times l_{\mathsf{out}}}$. □

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

Setup :

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathsf{ZeroSamp}_\lambda$ $\boxed{\mathbf{M}^\top \xleftarrow{\$} \mathsf{OneSamp}_\lambda}$

Sample $\mathbf{V} \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$, set $\mathbf{U} = \mathbf{MV}$, and return $\mathsf{aux} = (\mathbf{M}, \mathbf{U})$

Computation :

Set $\mathcal{W}$ initially as $\{\mathbf{M}, \mathbf{U}\}$

For the ith input wire x_i in $\mathsf{F}_\lambda(\mathbf{x})$, run

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

$$\mathbf{X}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}, \mathbf{Z}_i = \mathbf{M}^{-1}(\mathbf{X}_i - x_i \otimes \mathbf{I}_\lambda)$$

and include $\mathsf{W}[\mathbf{MZ}_i + x_i \otimes \mathbf{I}_\lambda]$ in $\mathcal{W}$

For each remaining non-input and non-output wire assigned the value w_i in $\mathsf{F}_\lambda(\mathbf{x})$, run

$$\mathbf{Z}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}, \mathbf{W}_i = \mathbf{MZ}_i + w_i \otimes \mathbf{I}_\lambda$$

For the ith output wire y_i in $\mathsf{F}_\lambda(\mathbf{x})$, run

$$\mathbf{Z}_{\mathsf{out}_i} \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}, \mathbf{Y}_i = \mathbf{MZ}_{\mathsf{out}_i} + y_i \otimes \mathbf{I}_\lambda \quad \mathbf{Y} \xleftarrow{\$} \{0,1\}^{\lambda\times l_{\mathsf{out}}}, \mathbf{Z} = \mathbf{M}^{-1}(\mathbf{Y} - \mathbf{e}_\lambda \cdot \mathbf{y}^\top)$$

Let $\mathbf{y}_i$ (respectively, $\mathbf{z}_i$) be the rightmost column of $\mathbf{Y}_i$ (respectively, $\mathbf{Z}_{\mathsf{out}_i}$)

Let $\mathbf{Y} = (\mathbf{y}_i)_{i\in[l_{\mathsf{out}}]}$ and $\mathbf{Z} = (\mathbf{z}_i)_{i\in[l_{\mathsf{out}}]}$

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

Sample $\mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda\times l_{\mathsf{out}}}$, compute

$$(\mathcal{W}_\mathbf{Y}, \mathbf{Y}') \Leftarrow \mathbf{Y} + \mathbf{UR}, (\mathcal{W}_\mathbf{y}, \mathbf{y}^\top) \Leftarrow \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{Y}') \quad \boxed{\mathcal{W}_\mathbf{y} = \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{Z} + \mathbf{VR}, \mathbf{y}^\top)}$$

and include $(\mathcal{W}_\mathbf{Y}, \mathcal{W}_\mathbf{y})$ in $\mathcal{W}$

Return $\mathcal{W}$

Figure 10: Generation of wire assignments in $\mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3, \mathsf{G}_4, \mathsf{G}_5$. $\{\mathsf{SimAdd}_\lambda, \mathsf{SimMult}_\lambda\}$ are the function families defined as in Figure 5. Operations for input gate gadgets, addition and multiplication gate gadgets, and output gate gadgets are performed in parallel within each respective category.

Game G_5. In G_5, for all $i \in [l_{\mathsf{in}}]$, $\mathbf{X}_i$ is sampled 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 3.14 *The distributions of $(\ell_\lambda(\mathsf{aux}), \ell'_\lambda(\mathcal{W}))$ in G_4 and G_5 are identical.*

Proof. Lemma 3.14 follows immediately from the fact that $\mathbf{M}^\top \in \mathsf{OneSamp}_\lambda$ 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 $\mathbf{X}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$. □

Game G_6. In G_6, aux and $\mathcal{W}$ are generated as

$$(\mathsf{aux}, \mathsf{LTSim}_\lambda) \xleftarrow{\$} \mathsf{DSim}_\lambda \text{ and } \mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda(\mathbf{x}, \mathbf{y}).$$

Namely, G_6 differs from G_5 in that each addition (respectively, multiplication) gate in F_λ is implemented using Add_λ (respectively, Mult_λ) now.

Lemma 3.15 *The distributions of $(\ell_\lambda(\mathsf{aux}), \ell'_\lambda(\mathcal{W}))$ in G_5 and G_6 are identical.*

Lemma 3.15 follows immediately from Lemma 3.2.
Putting all the above together, Theorem 3.9 immediately follows. □

Overheads. In our sAI-LTC, the sizes of $\mathsf{LTSetup}_\lambda$ and LTComp_λ are $O(\lambda^2)$ and $O(|\mathsf{F}_\lambda| \cdot \lambda^3)$ respectively. In contrast, the sizes of the setup algorithm and the computation algorithm of the AI-LTC in [22] are $O(\lambda^2 + l_{\mathsf{out}} \cdot \lambda)$ and $O(|\mathsf{F}_\lambda| \cdot \lambda^3)$ respectively, where the former depends on the output length l_{out} and is asymptotically larger than ours for large l_{out}.

4 Generic Construction of LRC against NC^1 without Leak-Free Hardware

In this section, we propose a generic conversion from 2-adaptive sAI-LTC to LRC without any leak-free hardware, using a 2-adaptive LRCE.

4.1 2-Adaptive LRCE

Below we recall the definition and an instantiation of 2-adaptive LRCE.

Definition 4.1 (2-adaptive leakage-resilient composable encoding [22]). A *2-adaptive $\{\mathcal{L}_\lambda, l_r\}$-leakage-resilient composable encoding (LRCE) 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 encoding ct,
- D_λ on input ct deterministically 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 $\lambda \in \mathbb{N}$, all $\ell_\lambda \in \mathcal{L}_\lambda$, all $\mathbf{m}_0, \mathbf{m}_1 \in \{0,1\}^{l_{\mathsf{m}}}$, and all functions $\mathcal{A}$ with range $\mathcal{L}_\lambda$, the following two distributions are statistically close with negligible distance:

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

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

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

$\{\mathcal{L}_\lambda\}$-*Leakage-resilient encoding* is defined in the same way as above, except that the adversary $\mathcal{A}$ only outputs $\perp$ and composability is not required.

Instantiation. Let $\mathcal{L}_\lambda$ be the class of all function families in NC^1 with output length $O(\log \lambda)$. It was shown in [22] that, assuming $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$, for each polynomial $l_r = l_r(\lambda)$ with $l_\mathsf{r} \leq l_\mathsf{m}$, there exists a 2-adaptive $\{\mathcal{L}_\lambda, l_r\}$-LRCE with message length $l_m = l_m(\lambda)$. We refer the reader to Appendix A for the instantiation.

4.2 Generic Construction of LRC

We now present our generic construction of LRC from sAI-LTC and LRCE.

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, $\{\mathcal{L}_\lambda\}$ be a class of function families that is closed under composition, $\mathsf{LRCE} = \{\mathsf{E}_\lambda, \mathsf{D}_\lambda\}$ be an $\{\mathcal{L}_\lambda, l_\mathsf{r}\}$-LRCE scheme with message space $\{0,1\}^{l_\mathsf{st}+l_\mathsf{r}}$, where $\{\mathsf{E}_\lambda\}$ can be decomposed into $\{\mathsf{E}^1_\lambda, \mathsf{E}^2_\lambda\}$ and $\{\mathsf{Dpub}_\lambda\}$ be the associated public-coin distribution family. Let $\mathsf{sAI\text{-}LTC} = \{\mathsf{LTSetup}_\lambda, \mathsf{LTComp}_\lambda\}$ be an $\{\mathcal{L}_\lambda\}$-sAI-LTC for $\{\mathsf{F}^*_\lambda\}$, where $\{\mathsf{F}^*_\lambda\}$ is a function family defined as in Figure 11. [4] Let the sizes of the internal randomness used by E^1_λ and $\mathsf{LTSetup}_\lambda$ be l_e and l_a respectively, where $l_\mathsf{e} + l_\mathsf{a} = l_\mathsf{r}$. We propose an $\{\mathcal{L}_\lambda\}$-LRC for $\{\mathsf{F}_\lambda\}$, denoted by LRC, in Figure 12.

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

Compute $(\mathbf{s}, \mathbf{r}, \mathbf{r}_\mathsf{aux}) = \mathsf{D}_\lambda(\mathsf{ct}||\mathsf{ct}_\mathsf{rand})$, where $|\mathbf{s}| = l_\mathsf{st}$, $|\mathbf{r}| = l_\mathsf{e}$, and $|\mathbf{r}_\mathsf{aux}| = l_\mathsf{a}$

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

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

Figure 11: Definition of $\{\mathsf{F}^*_\lambda\}$.

$\underline{\mathsf{Setup}_\lambda(\mathbf{s}) :}$

Run $\mathsf{ct} \xleftarrow{\$} \mathsf{E}^1_\lambda(\mathbf{s})$ and sample $\mathsf{aux} \xleftarrow{\$} \mathsf{LTSetup}_\lambda$

Return $\mathsf{st} = (\mathsf{ct}, \mathsf{aux})$

$\underline{\mathsf{Comp}_\lambda(\mathsf{st} = (\mathsf{ct}, \mathsf{aux}), \mathbf{x}):}$

Sample $\mathsf{ct}_\mathsf{rand} \xleftarrow{\$} \mathsf{Dpub}_\lambda$ and compute $\big((\mathsf{ct}', \mathsf{aux}'), \mathbf{y}\big) \xleftarrow{\$} \mathsf{LTComp}_\lambda(\mathsf{aux}, (\mathsf{ct}, \mathsf{ct}_\mathsf{rand}, \mathbf{x}))$

Return $\big(\mathsf{st}' = (\mathsf{ct}', \mathsf{aux}'), \mathbf{y}\big)$

Figure 12: Definition of $\mathsf{LRC} = \{\mathsf{Setup}_\lambda, \mathsf{Comp}_\lambda\}$.

In the setup phase, the initial state is encoded using the underlying LRCE, and only the first-round auxiliary input is generated. In each invocation, the compiled circuit uses the sAI-LTC to decode the current encoded state, evaluate the prescribed computation, and produce both the updated state encoding and the auxiliary input for the next invocation. Thus, unlike [22], later auxiliary inputs are generated by the previous invocation itself, without leak-free hardware. This is possible because the sAI-LTC setup is succinct, and the randomness needed for the next state encoding and auxiliary input is supplied through the public-coin extension of the LRCE.

Theorem 4.2 LRC *is an* $\{\mathcal{L}_\lambda\}$*-LRC for* $\{\mathsf{F}_\lambda\}$.

Proof sketch. The proof proceeds inductively over the invocations. Initially, setup generates an initial LRCE encoding of the plaintext state together with the first-round auxiliary input. In each subsequent invocation, the public-coin extension of the LRCE supplies an encoding of the fresh random coins needed

[4] Here, F^*_λ runs $\mathsf{LTSetup}_\lambda$ while F^*_λ is compiled using $\{\mathsf{LTSetup}_\lambda, \mathsf{LTComp}_\lambda\}$. This is possible due to the succinctness of the sAI-LTC.

to generate the next encoded state and the next auxiliary input. For each invocation, the 2-adaptive leakage-tolerance of the sAI-LTC allows us to replace the real auxiliary input and wire assignment by a simulated auxiliary input and the simulated wire assignment for the corresponding compiled input/output pair. This also hides the computation that generates the auxiliary input for the next invocation, since that generation is part of the current sAI-LTC computation. The leakage-resilience of the LRCE then allows us to switch the LRCE encoding of the updated state and the encoded randomness extension to dummy encodings. Hence the next encoded state has the simulated form required for the induction to continue. Iterating this argument over all invocations gives the simulator in the ideal experiment. We refer the reader to Appendix B for the detailed proof.

Instantiation. By instantiating the underlying LRCE as in Figure 15, we obtain

$$\begin{aligned}|\mathsf{F}^*_\lambda| &= \underbrace{O\big((l_{\mathsf{st}} + l_{\mathsf{st}} \cdot \lambda + \lambda^2)\lambda^2\big)}_{|\mathsf{D}_\lambda|} + |\mathsf{F}_\lambda| + \underbrace{O(l_{\mathsf{st}} \cdot \lambda + \lambda^2)}_{|\mathsf{E}^1_\lambda|} + \underbrace{O(\lambda^2)}_{|\mathsf{LTSetup}_\lambda|} \\ &= O(l_{\mathsf{st}} \cdot \lambda^3 + \lambda^4) + |\mathsf{F}_\lambda|.\end{aligned}$$

By further instantiating the underlying sAI-LTC as in Figure 8, we have $|\mathsf{Setup}_\lambda| = O(l_{\mathsf{st}} \cdot \lambda + \lambda^2)$ and

$$|\mathsf{Comp}_\lambda| = |\mathsf{F}^*_\lambda| \cdot \lambda^3 = O\big((l_{\mathsf{st}} \cdot \lambda^3 + \lambda^4) + |\mathsf{F}_\lambda|\big) \cdot \lambda^3 = O(l_{\mathsf{st}} \cdot \lambda^6 + \lambda^7 + |\mathsf{F}_\lambda| \cdot \lambda^3),$$

where the multiplicative overhead of λ^3 for each gate in F^*_λ arises from matrix multiplication over $\{0,1\}^{\lambda \times \lambda}$. Notably, our construction does not rely on any leak-free opaque gates.

In contrast, in the LRCs of [22], the setup algorithm and the compiled circuit have sizes $O(l_{\mathsf{st}} \cdot \lambda^2)$ and $O(l_{\mathsf{st}} \cdot \lambda^6 + |\mathsf{F}_\lambda| \cdot \lambda^3)$, or $O(l_{\mathsf{st}} \cdot \lambda^3)$ and $O(l_{\mathsf{st}} \cdot \lambda^4 + |\mathsf{F}_\lambda| \cdot \lambda^3)$, respectively. Their constructions additionally rely on opaque gates of sizes depending on the internal-state and external-output lengths. In the LRC of [17], the setup algorithm and the compiled circuit have sizes $O(l_{\mathsf{st}} \cdot \lambda^2)$ and $O(|\mathsf{F}_\lambda| \cdot \lambda^3)$ respectively, and their construction exploits $O(|\mathsf{F}_\lambda| \cdot \lambda^3)$ opaque gates, further depending on the size of the original circuit.

5 LRC with Optimized Circuit Size

In this section, we propose a non-black-box construction of LRCs against NC^1 without leak-free hardware that achieves a smaller circuit size compared to our instantiation in the previous section, at the cost of a larger setup.

Roughly, our construction can be viewed as an extension of the non-black-box re-encryption approach of [22]. In contrast to their construction, we further incorporate the decoding technique based on ciphertext banks from Section 3, which allows us to eliminate the need for any leak-free hardware. Moreover, we design a more refined algorithm for generating the encoded state in each invocation, avoiding any additional blow-up in the complexity of state updates.

Let $\{\mathcal{L}_\lambda\}$ denote 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. Our construction of $\{\mathcal{L}_\lambda\}$-LRC for $\{\mathsf{F}_\lambda\}$, denoted by LRC^*, is given in Figure 13.

Theorem 5.1 *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$ *holds, then* LRC^* *is an* $\{\mathcal{L}_\lambda\}$*-LRC for* $\{\mathsf{F}_\lambda\}$.

Proof sketch. Similar to the proof of our generic construction, we proceed by induction over the invocations. Assume that at the beginning of the i-th invocation, the current encoded state has the required fresh form: it consists of the current public key $\mathbf{M}$, the current ciphertext bank, and WP-FHE ciphertexts of the current plaintext state. We view this invocation as a WP-FHE homomorphic computation followed by ciphertext-bank-based decoding, including the decodings for the external output and the next encoded state. By gadget reconstructibility and Theorem 3.4, the leakage from this invocation can be simulated from $\mathbf{M}$, the plaintext wire values, and the corresponding ciphertext randomness, including the randomness of the simulated pre-decoding wire bundles and the refreshed randomness used for decoding. This enables the $\mathsf{ZeroSamp}_\lambda$-to-$\mathsf{OneSamp}_\lambda$ switch. Under $\mathsf{OneSamp}_\lambda$, $\mathbf{M}$ is full rank, so the WP-FHE wire bundles hide the plaintext wire values. Hence the computation producing the next encoded state is hidden, and the decoded next encoded state again has the required fresh form. Repeating this over all invocations gives the ideal simulation. We refer the reader to Appendix C for the detailed proof.

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

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathsf{ZeroSamp}_\lambda$ and $\mathbf{V} \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ and compute $\mathbf{U} = \mathbf{MV}$

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

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

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

Computation:

For each $i \in [l_{\mathsf{st}}]$, represent the ith state gate of F_λ by the wire bundle $\mathbf{S}_i$

Each input gate x_i of F_λ is implemented by sampling $\mathbf{Z}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ and setting

$$\mathbf{X}_i = \mathbf{MZ}_i + x_i \otimes \mathbf{I}_\lambda$$

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

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

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

Let $\mathbf{Y} = (\mathbf{y}_i)_{i\in[l_{\mathsf{out}}]}$ be the sequence of rightmost columns of the matrices representing the output wires in F_λ, sample $\mathbf{R}_y \xleftarrow{\$} \{0,1\}^{\lambda\times l_{\mathsf{out}}}$ and compute

$$\mathbf{y}^\top = \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{Y} + \mathbf{UR}_y)$$

State update:

Sample $\mathbf{r} \xleftarrow{\$} \{0,1\}^{\lambda(\lambda-1)/2}$, set $\overline{\mathbf{M}'} = \mathsf{M}_\lambda(\mathbf{r})^\top$, and sample $\overline{\mathbf{U}'} \xleftarrow{\$} \{0,1\}^{(\lambda-1)\times\lambda}$

Sample $\mathbf{C} \xleftarrow{\$} \{0,1\}^{\lambda\times(\lambda-1)}$ and $\mathbf{R}_{\mathsf{st}} \xleftarrow{\$} \{0,1\}^{\lambda\times 2\lambda}$, and compute

$$(\underline{\mathbf{M}'} \| \underline{\mathbf{U}'}) = \mathsf{Dec}_\lambda\big(\mathbf{M},\ \mathbf{C}(\overline{\mathbf{M}'} \| \overline{\mathbf{U}'}) + \mathbf{UR}_{\mathsf{st}}\big)$$

Define $\mathbf{M}' = \begin{pmatrix}\overline{\mathbf{M}'}\\ \underline{\mathbf{M}'}\end{pmatrix}$ and $\mathbf{U}' = \begin{pmatrix}\overline{\mathbf{U}'}\\ \underline{\mathbf{U}'}\end{pmatrix}$, where $\underline{\mathbf{M}'}, \underline{\mathbf{U}'} \in \{0,1\}^{1\times\lambda}$.

Let $(\mathbf{s}_i)_{i\in[l_{\mathsf{st}}]}$ be the rightmost columns of the matrices representing the updated state

For each $i \in [l_{\mathsf{st}}]$, sample $\mathbf{P}_i, \mathbf{R}_{\mathbf{S}_i} \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$ and compute

$$\mathbf{S}_i'^\top = \mathsf{Dec}_\lambda\big(\mathbf{M},\ \mathbf{P}_i\mathbf{M}'^\top + (\mathbf{U}\otimes\mathbf{I}_\lambda)\mathbf{R}_{\mathbf{S}_i} + \mathbf{s}_i \otimes \mathbf{I}_\lambda\big).$$

Return $\Big(\mathsf{st}' = \big(\mathbf{M}', \mathbf{U}', (\mathbf{S}_i')_{i\in[l_{\mathsf{st}}]}\big), \mathbf{y}\Big)$.

Figure 13: Definition of $\mathsf{LRC}^* = \{\mathsf{Setup}^*_\lambda, \mathsf{Comp}^*_\lambda\}$

Overheads. In this instantiation, the setup algorithm and the compiled circuit in each invocation have sizes $O(l_{\mathsf{st}} \cdot \lambda^2)$ and $O(l_{\mathsf{st}} \cdot \lambda^4 + |\mathsf{F}_\lambda| \cdot \lambda^3)$, respectively, where the overhead $O(l_{\mathsf{st}} \cdot \lambda^4)$ stems from the decoding procedure for recovering $\mathbf{S}'_i$ for all $i \in [l_{\mathsf{st}}]$, during the state update, and $O(|\mathsf{F}_\lambda| \cdot \lambda^3)$ stems from running Add_λ and Mult_λ for each gate. All the other lower-order overheads are absorbed by the asymptotic notation.

As shown in Table 1, compared with our instantiation in Section 4, the compiled circuit size is smaller (than $O(l_{\mathsf{st}} \cdot \lambda^6 + \lambda^7 + |\mathsf{F}_\lambda| \cdot \lambda^3)$) for larger λ, while the setup complexity is larger (than $O(l_{\mathsf{st}} \cdot \lambda + \lambda^2)$). At the same time, it matches the compiled circuit size of the second construction in [22], while using a strictly smaller setup (than $O(l_{\mathsf{st}} \cdot \lambda^3)$).

6 Fine-Grained Multi-Theorem oNIZK

In this section, we define fine-grained multi-theorem oNIZK and present a generic construction from sAI-LTC with parallel-composability and leakage-resilient encoding, yielding an instantiation against NC^1 adversaries with compact CRSs and optimal verification complexity.

Definition 6.1 (Multi-theorem non-interactive zero-knowledge proof with offline simulation). A *multi-theorem $\{\mathcal{C}_\lambda\}$-non-interactive zero-knowledge proof with offline simulation (oNIZK)* for a 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 common reference string 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 $\mathsf{Ver}_\lambda(\mathsf{crs}, \mathbf{x}, \pi) = 1$.

Multi-theorem zero-knowledge with offline simulation is satisfied if for all polynomial $q = q(\lambda)$ in λ, there exists a polynomial-time samplable distribution family $\{\mathcal{D}_\lambda^{\mathsf{NIZK}}\}$ whose samples are of the form $(\mathsf{crs}', \mathsf{NIZKSim}_\lambda)$, where $\mathsf{NIZKSim}_\lambda \in \widetilde{\mathcal{C}}_\lambda$, such that for all $\lambda \in \mathbb{N}$, all $(\mathbf{x}_i, \mathbf{w}_i)_{i\in[q]}$ with $\mathsf{R}_\lambda(\mathbf{x}_i, \mathbf{w}_i) = 1$ for all $i \in [q]$, and any distinguisher $\mathcal{A}_\lambda \in \mathcal{C}_\lambda$ with 1-bit output, we have:

$$\Delta\Big(\mathcal{A}_\lambda\big(\mathsf{crs}, (\mathsf{Prove}_\lambda(\mathsf{crs}, \mathbf{x}_i, \mathbf{w}_i))_{i\in[q]}\big), \mathcal{A}_\lambda\Big(\mathsf{crs}', \mathsf{NIZKSim}_\lambda((\mathbf{x}_i)_{i\in[q]})\Big)\Big) \le \mathsf{negl}(\lambda)$$

where $\mathsf{crs} \xleftarrow{\$} \mathsf{Gen}_\lambda$ and $(\mathsf{crs}', \mathsf{NIZKSim}_\lambda) \xleftarrow{\$} \mathcal{D}_\lambda^{\mathsf{NIZK}}$.

Perfect soundness is satisfied if for all $\lambda \in \mathbb{N}$, all $\mathsf{crs} \in \mathsf{Gen}_\lambda$, all $\mathbf{x}$ such that there is no $\mathbf{w}$ such that $\mathsf{R}_\lambda(\mathbf{x}, \mathbf{w}) = 1$, and all π, we have $\mathsf{Ver}_\lambda(\mathsf{crs}, \mathbf{x}, \pi) = 0$.

Besides perfect soundness, the above definition differs from the original definition of fine-grained oNIZK in [2] in that simulated CRSs are generated independently of simulated statements, and zero-knowledge holds for proofs of multiple statements under the same CRS.

Below we define parallel-composability for sAI-LTC.

Definition 6.2 (Parallel-composable sAI-LTC). An $\{\mathcal{L}_\lambda\}$-sAI-LTC $\{\mathsf{LTSetup}_\lambda, \mathsf{LTComp}_\lambda\}$ for a function family $\{\mathsf{F}_\lambda\}$ is said to be parallel-composable if for any polynomial $q = q(\lambda)$, $\{\mathsf{LTSetup}_\lambda, \mathsf{LTComp}_\lambda^{\|q}\}$ is an $\{\mathcal{L}_\lambda\}$-sAI-LTC for $\{\mathsf{F}_\lambda^{\|q}\}$. [5]

It is straightforward to see that our sAI-LTC in Figure 8 is parallel-composable.

Let $\{\mathcal{L}_\lambda\}$ be a class of function families that is closed under composition and contains AC^0. Let $\{\mathsf{R}_\lambda\}$ be any relation family. Let $\{\mathsf{E}_\lambda, \mathsf{D}_\lambda\}$ be an $\{\mathcal{L}_\lambda\}$-leakage-resilient encoding scheme. For any polynomial $q = q(\lambda)$, let $\{\mathsf{LTSetup}_\lambda \in \widetilde{\mathcal{L}}_\lambda, \mathsf{LTComp}_\lambda\}$ be a parallel-composable $\{\mathcal{L}_\lambda\}$-sAI-LTC for $\{\mathsf{F}_\lambda(\cdot,\cdot) = \mathsf{R}_\lambda\big(\cdot, \mathsf{D}_\lambda(\cdot)\big)\}$. We give the multi-theorem oNIZK, denoted by NIZK, for $\{\mathsf{R}_\lambda\}$ as in Figure 14.

Theorem 6.3 NIZK *is a multi-theorem $\{\widetilde{\mathcal{L}}_\lambda\}$-oNIZK with a depth-3 AC^0 verifier.*

The proof of Theorem 6.3 is similar to the proof of the oNIZK in [22, 23], except that, in the simulation, the simulated CRS is sampled independently of the simulated statements/proofs, rather than being

[5] Recall that the superscript $\|q$ denotes q independent parallel executions of the algorithm (see Section 2).

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

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

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

Run $\mathsf{ct} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{w})$ and $(\mathcal{W}, b) \Leftarrow \mathsf{LTComp}_\lambda(\mathsf{aux}, (\mathbf{x}, \mathsf{ct}))$, and return $\pi = \mathcal{W}$

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

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

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

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

Return 1 iff all checks pass

Figure 14: Definition of $\mathsf{NIZK} = \{\mathsf{Gen}_\lambda, \mathsf{Prove}_\lambda, \mathsf{Ver}_\lambda\}$.

generated by the simulator together with the proof. We refer the reader to Appendix D for the detailed proof.

By instantiating the above construction with our sAI-LTC and the LRCE given in Figures 8 and 15 respectively, we obtain the following corollary.

Corollary 6.4 *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$ *holds, then there exists a multi-theorem* NC^1*-oNIZK for all* NP *with CRS size* $2\lambda^2$ *and a depth-*3 AC^0 *verifier.*

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). https://doi.org/10.1007/978-3-642-00457-5_28 (Cited on page 1.)

[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). https://doi.org/10.1007/978-3-030-56877-1_24 (Cited on page 3, 22.)

[3] 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). https://doi.org/10.4230/LIPIcs.ITCS.2022.26 (Cited on page 2.)

[4] Bogdanov, A., Ishai, Y., Srinivasan, A.: Unconditionally secure computation against low-complexity leakage. Journal of Cryptology **34**(4), 38 (Oct 2021). https://doi.org/10.1007/s00145-021-09402-2 (Cited on page 2, 5, 8.)

[5] 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). https://doi.org/10.1109/FOCS.2010.55 (Cited on page 1.)

[6] 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). https://doi.org/10.1007/978-3-662-53015-3_19 (Cited on page 3, 4, 9.)

[7] Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. Journal of Cryptology **32**(1), 151–177 (Jan 2019). https://doi.org/10.1007/s00145-018-9284-1 (Cited on page 2.)

[8] Egashira, S., Wang, Y., Tanaka, K.: Fine-grained cryptography revisited. Journal of Cryptology **34**(3), 23 (Jul 2021). https://doi.org/10.1007/s00145-021-09390-3 (Cited on page 9.)

[9] 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, 4.)

[10] Goldwasser, S., Rothblum, G.N.: How to compute in the presence of leakage. In: 53rd FOCS. pp. 31–40. IEEE Computer Society Press (Oct 2012). https://doi.org/10.1109/FOCS.2012.34 (Cited on page 5.)

[11] Goyal, V., Ishai, Y., Maji, H.K., Sahai, A., Sherstov, A.A.: Bounded-communication leakage resilience via parity-resilient circuits. In: Dinur, I. (ed.) 57th FOCS. pp. 1–10. IEEE Computer Society Press (Oct 2016). https://doi.org/10.1109/FOCS.2016.10 (Cited on page 2.)

[12] Guan, J., Zhandry, M.: Simple schemes in the bounded storage model. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 500–524. Springer, Cham (May 2019). https://doi.org/10.1007/978-3-030-17659-4_17 (Cited on page 6, 10.)

[13] 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). https://doi.org/10.1007/978-3-540-45146-4_27 (Cited on page 1, 2, 8.)

[14] 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). https://doi.org/10.1007/978-3-031-58737-5_8 (Cited on page 2, 4.)

[15] 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). https://doi.org/10.1007/978-3-540-24638-1_16 (Cited on page 2.)

[16] Miles, E.: Iterated group products and leakage resilience against NC1. In: Naor, M. (ed.) ITCS 2014. pp. 261–268. ACM (Jan 2014). https://doi.org/10.1145/2554797.2554822 (Cited on page 2.)

[17] 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). https://doi.org/10.1145/2488608.2488640 (Cited on page 2, 3, 4, 20.)

[18] Raz, R.: A time-space lower bound for a large class of learning problems. In: Umans, C. (ed.) 58th FOCS. pp. 732–742. IEEE Computer Society Press (Oct 2017). https://doi.org/10.1109/FOCS.2017.73 (Cited on page 6, 10.)

[19] 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.)

[20] 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). https://doi.org/10.1007/978-3-642-32009-5_32 (Cited on page 2, 5.)

[21] 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). https://doi.org/10.1145/28395.28404 (Cited on page 8.)

[22] Wang, Y.: Leakage-resilient circuits against $\mathsf{NC1}$, revisited. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part V. LNCS, vol. 16004, pp. 38–69. Springer, Cham (Aug 2025). https://doi.org/10.1007/978-3-032-01901-1_2 (Cited on page 2, 3, 4, 5, 6, 7, 10, 11, 18, 19, 20, 22, 25.)

[23] Wang, Y.: Leakage-resilient circuits against NC1, revisited. Cryptology ePrint Archive, Report 2025/1463 (2025), https://eprint.iacr.org/2025/1463 (Cited on page 9, 11, 22, 25.)

[24] 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). https://doi.org/10.1007/978-3-031-07085-3_11 (Cited on page 4, 9, 10.)

Appendix

A Instantiation of 2-Adaptive LRCE

In this section, we recall the construction of 2-adaptive LRCE in [22].

Let $\{\mathcal{L}_\lambda\}$ be the class of all function families in NC^1 with output length $O(\log \lambda)$. We recall an instantiation of 2-adaptive LRCE in Figure 15.

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

Run $\mathbf{M}^\top \xleftarrow{\$} \mathsf{ZeroSamp}_\lambda$, $\mathbf{R} \xleftarrow{\$} \{0,1\}^{\lambda \times l_\mathsf{m}}$, and $\mathbf{C} = \mathbf{MR} + \mathbf{e}_\lambda \mathbf{m}^\top$

Return $\mathsf{ct} = (\mathbf{M}, \mathbf{C})$

$\underline{\mathsf{D}_\lambda(\mathsf{ct} = (\mathbf{M}, \mathbf{C})) :}$

Return $\mathbf{m} = \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{C}; \mathbf{0})$

$\underline{\mathsf{E}^1_\lambda(\mathbf{m} \in \{0,1\}^{l_\mathsf{m} - l_\mathsf{r}}) :}$

Run $\mathbf{M}^\top \xleftarrow{\$} \mathsf{ZeroSamp}_\lambda$, $\mathbf{R}_1 \xleftarrow{\$} \{0,1\}^{\lambda \times (l_\mathsf{m} - l_\mathsf{r})}$, and $\mathbf{C}_1 = \mathbf{MR}_1 + \mathbf{e}_\lambda \mathbf{m}^\top$

Return $\mathsf{ct} = (\mathbf{M}, \mathbf{C}_1)$

$\underline{\mathsf{E}^2_\lambda(\mathsf{ct} = (\mathbf{M}, \mathbf{C}), \mathbf{r} \in \{0,1\}^{l_\mathsf{r}}) :}$

Run $\mathbf{R}_2 \xleftarrow{\$} \{0,1\}^{\lambda \times l_\mathsf{r}}$ and $\mathbf{C}_2 = \mathbf{MR}_2 + \mathbf{e}_\lambda \mathbf{r}^\top$, and return $\mathsf{ct}_\mathsf{rand} = \mathbf{C}_2$

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

Return $\mathsf{c} \xleftarrow{\$} \{0,1\}^{\lambda \times l_\mathsf{r}}$

Figure 15: Definition of $\mathsf{LRCE} = \{\mathsf{E}_\lambda, \mathsf{D}_\lambda\}$ with the associated decomposed function families $\{\mathsf{E}^1_\lambda, \mathsf{E}^2_\lambda\}$ and public-coin distribution family $\{\mathsf{Dpub}_\lambda\}$. See Figure 6 for the definition of $\{\mathsf{Dec}_\lambda\}$. $\{\mathsf{D}_\lambda\}$ is deterministic since the internal random coins of Dec_λ are fixed as $\mathbf{0}$.

Theorem A.1 (Theorem 5.3 in [23]). *If* $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$ *holds, then for any polynomial* $l_r = l_r(\lambda)$ *and* $l_m = l_m(\lambda)$ *such that* $l_\mathsf{r} \leq l_\mathsf{m}$, LRCE *is a 2-adaptive* $\{\mathcal{L}_\lambda, l_r\}$*-LRCE with message length* l_m*.*

B Proof of Theorem 4.2

In this section, we give the proof of Theorem 4.2.

Proof. Correctness follows from that of $\mathsf{sAI\text{-}LTC}$ and LRCE.

Let $\mathcal{A}$ be any adversary, q be the total number of invocations, and let $\{\mathsf{DSim}_\lambda\}$ be the distribution family guaranteed by the security of $\mathsf{sAI\text{-}LTC}$, whose samples are of the form $(\mathsf{aux}, \mathsf{LTSim}_\lambda)$. We give the description of the simulator $\{\mathsf{Sim}_\lambda\}$ in Figure 16, and prove the leakage-resilience of 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 17.

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 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 $\mathsf{ct}_\mathsf{rand} \xleftarrow{\$} \mathsf{E}^2_\lambda(\mathsf{ct}, \mathbf{r} || \mathbf{r}_\mathsf{aux})$, where $\mathbf{r} \xleftarrow{\$} \{0,1\}^{l_\mathsf{e}}$ and $\mathbf{r}_\mathsf{aux} \xleftarrow{\$} \{0,1\}^{l_\mathsf{a}}$, instead of $\mathsf{ct}_\mathsf{rand} \xleftarrow{\$} \mathsf{Dpub}_\lambda$, in each invocation.

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

Proof. Lemma B.1 follows immediately from the composability of LRCE. □

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

Compute $\mathsf{ct} \xleftarrow{\$} \mathsf{E}^1_\lambda(\mathbf{0})$ and sample $(\mathsf{aux}, \mathsf{LTSim}_\lambda) \xleftarrow{\$} \mathsf{DSim}_\lambda$

Set the initial encoded state as $\mathsf{st} = (\mathsf{ct}, \mathsf{aux})$

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 $(\mathsf{aux}', \mathsf{LTSim}'_\lambda) \xleftarrow{\$} \mathsf{DSim}_\lambda$

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

Update the encoded state to $\mathsf{st}' = (\mathsf{ct}', \mathsf{aux}')$ and return $\mathcal{W}$

Figure 16: 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 in each invocation, respectively. By $(\mathbf{x}, \mathbf{y})$ we denote the external input and output of F_λ in the current invocation.

G_0, $\boxed{\mathsf{G}_1}$, $\boxed{\mathsf{G}_{\mathsf{c}.0}}$, $\mathsf{G}_{\mathsf{c}.1}$
Setup:

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

Run $\mathsf{aux} \xleftarrow{\$} \mathsf{LTSetup}_\lambda$ $(\mathsf{aux}, \mathsf{LTSim}_\lambda) \xleftarrow{\$} \mathsf{DSim}_\lambda$

Set the initial encoded state as $\mathsf{st} = (\mathsf{ct}, \mathsf{aux})$

Computation:

Let $\mathsf{st} = (\mathsf{ct}, \mathsf{aux})$ be the current encoded state

Run $\mathsf{ct}_{\mathsf{rand}} \xleftarrow{\$} \mathsf{Dpub}_\lambda$ $\boxed{\mathsf{ct}_{\mathsf{rand}} \xleftarrow{\$} \mathsf{E}^2_\lambda(\mathsf{ct}, \mathbf{r}||\mathbf{r}_{\mathsf{aux}}) \text{ where } \mathbf{r} \xleftarrow{\$} \{0,1\}^{l_{\mathsf{e}}} \text{ and } \mathbf{r}_{\mathsf{aux}} \xleftarrow{\$} \{0,1\}^{l_{\mathsf{a}}}}$

$\mathsf{ct}_{\mathsf{rand}} \xleftarrow{\$} \mathsf{E}^2_\lambda(\mathsf{ct}, \mathbf{0})$

Run $\big(\mathcal{W}, ((\mathsf{ct}', \mathsf{aux}'), \mathbf{y})\big) \Leftarrow \mathsf{LTComp}_\lambda(\mathsf{aux}, (\mathsf{ct}, \mathsf{ct}_{\mathsf{rand}}, \mathbf{x}))$

Run $(\mathbf{s}', \mathbf{y}) = \mathsf{F}_\lambda[\mathbf{s}](\mathbf{x})$, $\mathsf{ct}' = \mathsf{E}^1_\lambda(\mathbf{s}'; \mathbf{r})$, and $\mathsf{aux}' = \mathsf{LTSetup}_\lambda(\mathbf{r}_{\mathsf{aux}})$
Run $\mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathsf{ct}, \mathsf{ct}_{\mathsf{rand}}, \mathbf{x}), (\mathsf{ct}', \mathsf{aux}', \mathbf{y}))$

$(\mathsf{aux}', \mathsf{LTSim}'_\lambda) \xleftarrow{\$} \mathsf{DSim}_\lambda$ $\mathsf{ct}' \xleftarrow{\$} \mathsf{E}^1_\lambda(\mathbf{0})$

Update the encoded state as $\mathsf{st}' = (\mathsf{ct}', \mathsf{aux}')$ and return $\mathcal{W}$

Figure 17: Generation of wire assignments in the intermediate games.

Game $\mathsf{G}_{\mathsf{c}.0}$ and $\mathsf{G}_{\mathsf{c}.1}$. $\mathsf{G}_{0.1}$ is the same as G_1.

For all $\mathsf{c} \in [q+1]$, $\mathsf{G}_{\mathsf{c}.0}$ differs from $\mathsf{G}_{\mathsf{c}-1.1}$ in that aux and $\mathcal{W}$ in the cth invocation are generated as

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

where $\mathbf{x}$ is sent by $\mathcal{A}$, $(\mathbf{s}', \mathbf{y}) = \mathsf{F}_\lambda[\mathbf{s}](\mathbf{x})$, $\mathsf{ct}' = \mathsf{E}^1_\lambda(\mathbf{s}'; \mathbf{r})$, and $\mathsf{aux}' = \mathsf{LTSetup}_\lambda(\mathbf{r}_{\mathsf{aux}})$. $\mathsf{G}_{\mathsf{c}.1}$ differs from $\mathsf{G}_{\mathsf{c}.0}$ in that ct and $\mathsf{ct}_{\mathsf{rand}}$ in the cth invocation are generated as $\mathsf{ct}||\mathsf{ct}_{\mathsf{rand}} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{0})$ (i.e., $\mathsf{ct} \xleftarrow{\$} \mathsf{E}^1_\lambda(\mathbf{0})$ and $\mathsf{ct}_{\mathsf{rand}} \xleftarrow{\$} \mathsf{E}^2_\lambda(\mathsf{ct}, \mathbf{0})$).

Note that there is no $(q+1)$th invocation, and thus the simulation of $\mathcal{W}$ and $\mathsf{ct}_{\mathsf{rand}}$ in $\mathsf{G}_{q+1.0}$ and $\mathsf{G}_{q+1.1}$ can be omitted.

Lemma B.2 *The views of $\mathcal{A}$ in $\mathsf{G}_{\mathsf{c}-1.1}$ and $\mathsf{G}_{\mathsf{c}.0}$ are statistically close with negligible distance for all $\mathsf{c} \in [q+1]$.*

Proof. We first observe that the auxiliary input aux in the cth invocation is generated during the setup phase when $\mathsf{c} = 1$, and is generated as aux' in the $(\mathsf{c}-1)$th invocation when $\mathsf{c} > 1$. Also, aux and $\mathcal{W}$ in

the cth invocation are generated independently of the wire assignments in the first $\mathsf{c}-2$ invocations, as well as those in the invocations after the cth.

Next, consider the hybrids $\mathsf{G}_{\mathsf{c}-1.1}$ and $\mathsf{G}_{\mathsf{c}.0}$. Fix all seeds except for the internal randomness used to generate aux, LTSim_λ, and $\mathcal{W}$ in the cth invocation, in a way that maximizes the adversary's distinguishing advantage conditioned on this fixing. Under this conditioning, the leakage (with fixed seeds) reveals no additional information about aux and $\mathcal{W}$ beyond the 2-adaptive leakage in $\mathcal{L}_\lambda$.

Therefore, by the 2-adaptive $\{\mathcal{L}_\lambda\}$-leakage-tolerance of $\mathsf{sAI\text{-}LTC}$, the view of $\mathcal{A}$ in $\mathsf{G}_{\mathsf{c}-1.1}$ is statistically close, with negligible distance, to its view in $\mathsf{G}_{\mathsf{c}.0}$, where in the cth invocation aux and $\mathcal{W}$ are generated using DSim_λ. This completes the proof of Lemma B.2. □

Lemma B.3 *The views of $\mathcal{A}$ in $\mathsf{G}_{\mathsf{c}.0}$ and $\mathsf{G}_{\mathsf{c}.1}$ are statistically close with negligible distance for all $\mathsf{c} \in [q+1]$.*

Proof. We first observe that when $\mathsf{c}=1$, ct is generated during the setup phase, and when $\mathsf{c}>1$, it is generated as ct' in the $(\mathsf{c}-1)$th invocation. Also, $\mathsf{ct}||\mathsf{ct}_{\mathsf{rand}}$ in the cth invocation are generated independently of the wire assignments in the first $(\mathsf{c}-2)$ invocations, as well as those in the invocations after the cth.

Next, consider the hybrids $\mathsf{G}_{\mathsf{c}.0}$ and $\mathsf{G}_{\mathsf{c}.1}$. Fix all seeds except for those used (by E^1_λ and E^2_λ) to generate ct and $\mathsf{ct}_{\mathsf{rand}}$ in the cth invocation, in a way that maximizes the adversary's distinguishing advantage conditioned on this fixing. Under this conditioning, the leakage (with fixed seeds) reveals no additional information about $\mathsf{ct}||\mathsf{ct}_{\mathsf{rand}}$ beyond the 2-adaptive leakage in $\mathcal{L}_\lambda$.

Therefore, by the 2-adaptive $\{\mathcal{L}_\lambda\}$-leakage-resilience of LRCE, the view of $\mathcal{A}$ in $\mathsf{G}_{\mathsf{c}.0}$ is statistically close, with negligible distance, to its view in $\mathsf{G}_{\mathsf{c}.1}$, where $\mathsf{ct}||\mathsf{ct}_{\mathsf{rand}}$ in the cth invocation is generated as a dummy encoding. This completes the proof of Lemma B.3. □

One can see that the view of $\mathcal{A}$ in $\mathsf{G}_{q+1.1}$ is identical to its view in the ideal game $\mathsf{Ideal}_{\mathcal{A}}$ (see Figure 3), where in each invocation, $\mathcal{A}$ receives the current external output together with the leakage on the wire assignments of the LRC generated by the simulator Sim_λ, and then outputs a new pair of external input and leakage function for the next invocation.

Putting all the above together, Theorem 4.2 immediately follows. □

C Proof of Theorem 5.1

In this section, we give the proof of Theorem 5.1.

Proof. **Correctness.** We first prove the correctness of the computation phase.

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

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

into wire bundles of the form

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

Therefore, in each invocation, letting y_i denote the ith output bit, s'_i the ith updated state bit, and $\mathbf{Y}_i$ and $\mathbf{T}_i$ the matrices representing y_i and s'_i respectively, we have, for each $i \in [l_{\mathsf{out}}]$,

$$\mathbf{Y}_i = \mathbf{M}\mathbf{Z}_i + y_i \otimes \mathbf{I}_\lambda \text{ for some } \mathbf{Z}_i \in \{0,1\}^{\lambda\times\lambda},$$

and for each $i \in [l_{\mathsf{st}}]$,

$$\mathbf{T}_i = \mathbf{M}\mathbf{R}_i + s'_i \otimes \mathbf{I}_\lambda \text{ for some } \mathbf{R}_i \in \{0,1\}^{\lambda\times\lambda}.$$

Namely, we have

$$\mathbf{Y} + \mathbf{U}\mathbf{R}_y = \mathbf{M}(\mathbf{Z} + \mathbf{V}\mathbf{R}_y) + \mathbf{e}_\lambda \cdot \mathbf{y}^\top \text{ for some } \mathbf{Z} \in \{0,1\}^{\lambda\times l_{\mathsf{out}}}$$

and

$$\mathbf{s}_i = \mathbf{M}\mathbf{r}_i + \mathbf{e}_\lambda \cdot s'_i \text{ for some } \mathbf{r}_i \in \{0,1\}^\lambda.$$

As a result, by Lemma 3.3, the decoder outputs the correct external output $\mathbf{y}$ and a correct encoded state is generated in the computation phase, in each invocation.

Next we show the correctness of the state update.

Since the rank of $\mathbf{M}\|\mathbf{e}_\lambda$ is λ, in the state update, we must have

$$\mathbf{C} = \mathbf{M}\mathbf{H} + \mathbf{e}_\lambda \cdot \widetilde{\mathbf{r}}^\top \text{ for some } \mathbf{H} \in \{0,1\}^{\lambda\times(\lambda-1)} \text{ and } \widetilde{\mathbf{r}} \in \{0,1\}^{\lambda-1}.$$

Therefore, we have

$$\begin{aligned}\mathbf{C}' &= \mathbf{C}(\overline{\mathbf{M}'}\|\overline{\mathbf{U}'}) + \mathbf{U}\mathbf{R}_{\mathsf{st}} \\ &= (\mathbf{M}\mathbf{H} + \mathbf{e}_\lambda \cdot \widetilde{\mathbf{r}}^\top)(\overline{\mathbf{M}'}\|\overline{\mathbf{U}'}) + \mathbf{M}\mathbf{V}\mathbf{R}_{\mathsf{st}} \\ &= \mathbf{M}\big(\mathbf{H}(\overline{\mathbf{M}'}\|\overline{\mathbf{U}'}) + \mathbf{V}\mathbf{R}_{\mathsf{st}}\big) + \mathbf{e}_\lambda \cdot \widetilde{\mathbf{r}}^\top(\overline{\mathbf{M}'}\|\overline{\mathbf{U}'}).\end{aligned}$$

By Lemma 3.3, it follows that

$$(\underline{\mathbf{M}'}\|\underline{\mathbf{U}'}) = \mathsf{Dec}_\lambda\big(\mathbf{M}, \mathbf{C}(\overline{\mathbf{M}'}\|\overline{\mathbf{U}'}) + \mathbf{U}\mathbf{R}_{\mathsf{st}}\big) = \widetilde{\mathbf{r}}^\top(\overline{\mathbf{M}'}\|\overline{\mathbf{U}'}).$$

Hence, we have $\mathbf{M}'^\top \in \mathsf{ZeroSamp}_\lambda$ and $\mathbf{U}' \in \mathsf{Span}(\mathbf{M}')$, i.e., $\mathbf{M}'$ and $\mathbf{U}'$ in st' are generated correctly.

We now turn to the correctness of $(\mathbf{S}_i)_{i\in[l_{\mathsf{st}}]}$. Since the rank of $(\mathbf{M}\|\mathbf{e}_\lambda)$ is λ, the rank of $(\mathbf{M}\otimes\mathbf{I}_\lambda)\|(\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)$ is λ^2. Hence, for each $i \in [l_{\mathsf{st}}]$, we must have

$$\mathbf{P}_i = (\mathbf{M}\otimes\mathbf{I}_\lambda)\mathbf{H}_i + (\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)\mathbf{Q}_i'^\top \text{ for some } \mathbf{H}_i \in \{0,1\}^{\lambda^2\times\lambda} \text{ and } \mathbf{Q}_i' \in \{0,1\}^{\lambda\times\lambda}.$$

Moreover, since $\mathbf{s}_i = \mathbf{M}\mathbf{r}_i + \mathbf{e}_\lambda \cdot s_i'$ for some $\mathbf{r}_i \in \{0,1\}^\lambda$ as shown before, and $\mathbf{U} = \mathbf{M}\mathbf{V}$ for some $\mathbf{V} \in \{0,1\}^{\lambda\times\lambda}$, we have

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

By Lemma 3.3, it follows that

$$\mathbf{S}_i' = \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{P}_i')^\top = \mathbf{M}'\mathbf{Q}_i' + s_i'\otimes\mathbf{I}_\lambda.$$

Hence, each $\mathbf{S}_i'$ in the refreshed state st' is a correct encoding of s_i', completing the proof of correctness.

Leakage-resilience. Let λ be the security parameter, $\mathcal{A}$ be any adversary, and q be the total number of invocations. We describe the simulator Sim_λ^* in Figure 18 and prove the leakage-resilience of LRC^* by introducing a sequence of intermediate games $\mathsf{G}_0, \ldots, \mathsf{G}_6$. Here, G_0 and G_6 correspond to the real and ideal games, respectively, as defined in Figure 3. For each intermediate game $\mathsf{G}_1, \ldots, \mathsf{G}_5$, the assignments to all wires in every invocation are generated as in Figures 19 and 20.

Game G_0. G_0 is the real game $\mathsf{Real}_\mathcal{A}$ (see Figure 3). In each invocation, $\mathcal{A}$ receives the current external output together with leakage on the honestly generated wire assignments of the LRC, and then outputs a new pair of external input and leakage function for the next invocation.

Game G_1. G_1 differs from G_0 in that, in each invocation, the challenger extends the external input $\mathbf{x}$ to a vector $\mathbf{w}$ that contains the assignments to all wires in the computation $\mathsf{F}_\lambda[\mathbf{s}](\mathbf{x})$, and then simulates the wire assignments for each gate gadget in parallel using $\{\mathsf{SimAdd}_\lambda, \mathsf{SimMult}_\lambda\}$.

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

Lemma C.1 follows immediately from Lemma 3.2.

Game G_2. Game G_2 differs from G_1 in the state update procedure of each invocation. Specifically, instead of sampling $\mathbf{C} \xleftarrow{\$} \{0,1\}^{\lambda\times(\lambda-1)}$ uniformly at random, the challenger generates

$$\mathbf{C} = \mathbf{M}\mathbf{H} + \mathbf{e}_\lambda \cdot \widetilde{\mathbf{r}}^\top \text{ where } \mathbf{H} \xleftarrow{\$} \{0,1\}^{\lambda\times(\lambda-1)} \text{ and } \widetilde{\mathbf{r}} \xleftarrow{\$} \{0,1\}^{\lambda-1}.$$

Moreover, for each $i \in [l_{\mathsf{st}}]$, instead of sampling $\mathbf{P}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$ uniformly at random, the challenger generates

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

Setup:

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathsf{OneSamp}_\lambda$ and $\mathbf{V} \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$, and compute $\mathbf{U} = \mathbf{MV}$

For $i \in [l_{\mathsf{st}}]$, sample $\mathbf{S}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$

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

Computation:

Compute $\mathbf{V} = \mathbf{M}^{-1}\mathbf{U}$

For $i \in [l_{\mathsf{st}}]$, represent the ith state wire of F_λ by $\mathbf{S}_i$ in the current state

For the ith input gate x_i in F_λ, run

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

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

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

and include $\mathcal{W}_i$ in $\mathcal{W}$

Let $\mathbf{Y} = (\mathbf{y}_i)_{i\in[l_{\mathsf{out}}]}$ be the sequence of the rightmost columns of the matrices representing the output wires of F_λ. Sample $\mathbf{R}_\mathbf{y} \xleftarrow{\$} \{0,1\}^{\lambda\times l_{\mathsf{out}}}$, compute

$$\mathcal{W}_\mathbf{Y} = \mathsf{W}[\mathbf{Y} + \mathbf{UR}_\mathbf{y}] \text{ and } \mathcal{W}_\mathbf{y} = \mathsf{SimDec}_\lambda\left(\mathbf{M},\ \mathbf{M}^{-1}(\mathbf{Y} - \mathbf{e}_\lambda \cdot \mathbf{y}^\top) + \mathbf{VR}_\mathbf{y},\ \mathbf{y}^\top\right),$$

and include $(\mathcal{W}_\mathbf{Y}, \mathcal{W}_\mathbf{y})$ in $\mathcal{W}$

State update:

Sample $\mathbf{M'}^\top \xleftarrow{\$} \mathsf{OneSamp}_\lambda$ and $\mathbf{V}' \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$, and compute $\mathbf{U}' = \mathbf{M}'\mathbf{V}'$

Sample $\mathbf{C} \xleftarrow{\$} \{0,1\}^{\lambda\times(\lambda-1)}$ and $\mathbf{R}_{\mathsf{st}} \xleftarrow{\$} \{0,1\}^{\lambda\times 2\lambda}$, compute

$$(\mathcal{W}_\mathbf{C}, \mathbf{C}') \Leftarrow \mathbf{C}(\overline{\mathbf{M}'}||\overline{\mathbf{U}'}) + \mathbf{UR}_{\mathsf{st}},$$

$$\mathcal{W}_{\mathsf{st}} = \mathsf{SimDec}_\lambda\left(\mathbf{M},\ \mathbf{M}^{-1}(\mathbf{C}' - \mathbf{e}_\lambda \cdot (\underline{\mathbf{M}'}||\underline{\mathbf{U}'})),\ \underline{\mathbf{M}'}||\underline{\mathbf{U}'}\right),$$

and include $(\mathcal{W}_\mathbf{C}, \mathcal{W}_{\mathsf{st}})$ in $\mathcal{W}$

Letting $(\mathbf{s}_i)_{i\in[l_{\mathsf{st}}]}$ be the rightmost columns of the matrices representing the updated state in F_λ, for each $i \in [l_{\mathsf{st}}]$, sample $\mathbf{P}_i, \mathbf{R}_{\mathbf{S}_i} \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$ and $\mathbf{S}'_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$, compute

$$(\mathcal{W}_{\mathbf{P}_i}, \mathbf{P}'_i) \Leftarrow \mathbf{P}_i \mathbf{M'}^\top + (\mathbf{U} \otimes \mathbf{I}_\lambda)\mathbf{R}_{\mathbf{S}_i} + \mathbf{s}_i \otimes \mathbf{I}_\lambda,$$

$$\mathcal{W}_{\mathbf{S}'_i} = \mathsf{SimDec}_\lambda\left(\mathbf{M},\ (\mathbf{M} \otimes \mathbf{I}_\lambda)^{-1}\left(\mathbf{P}'_i - (\mathbf{e}_\lambda \otimes \mathbf{I}_\lambda)\mathbf{S}_i'^\top\right),\ \mathbf{S}_i'^\top\right)$$

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

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

Figure 18: Definition of $\{\mathsf{Sim}^*_\lambda\}$. $\{\mathsf{SimDec}_\lambda\}$ is the function family defined as in Figure 6.

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

Setup:

Sample $\mathbf{M}^\top \xleftarrow{\$} \mathsf{ZeroSamp}_\lambda$ $\mathbf{M}^\top \xleftarrow{\$} \mathsf{OneSamp}_\lambda$ and $\mathbf{V} \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$

Compute $\mathbf{U} = \mathbf{MV}$

For $i \in [l_{\mathsf{st}}]$, run $\mathbf{Q}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ and $\mathbf{S}_i = \mathbf{MQ}_i + s_i \otimes \mathbf{I}_\lambda$ $\mathbf{S}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$, $\mathbf{Q}_i = \mathbf{M}^{-1}(\mathbf{S}_i - s_i \otimes \mathbf{I}_\lambda)$

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

Computation:

For each $i \in [l_{\mathsf{st}}]$, represent the ith state wire of F_λ by $\mathbf{S}_i$ in the current state, and set its associated randomness as $\mathbf{Z}_i = \mathbf{Q}_i$.

For each remaining non-state wire in $\mathsf{F}_\lambda(\mathbf{x})$ assigned the value w_i, sample $\mathbf{Z}_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$ and compute

$$\mathbf{W}_i = \mathbf{MZ}_i + w_i \otimes \mathbf{I}_\lambda$$

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

For each $i \in [l_{\mathsf{out}}]$, suppose that the matrix representing the ith output wire is of the form $\mathbf{Y}_i = \mathbf{MZ}_{\mathsf{out}_i} + y_i \otimes \mathbf{I}_\lambda$ and let $\mathbf{y}_i$ (respectively, $\mathbf{z}_i$) denote the rightmost column of $\mathbf{Y}_i$ (respectively, $\mathbf{Z}_{\mathsf{out}_i}$). Set

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

Sample $\mathbf{R}_\mathbf{y} \xleftarrow{\$} \{0,1\}^{\lambda\times l_{\mathsf{out}}}$ and compute

$$(\mathcal{W}_\mathbf{Y}, \mathbf{Y}') \Leftarrow \mathbf{Y} + \mathbf{UR}_\mathbf{y}, (\mathcal{W}_\mathbf{y}, \mathbf{y}^\top) \Leftarrow \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{Y}')$$

$\mathcal{W}_\mathbf{y} = \mathsf{SimDec}_\lambda(\mathbf{M}, \mathbf{Z} + \mathbf{VR}_\mathbf{y}, \mathbf{y}^\top)$, $\mathbf{Z} = \mathbf{M}^{-1}(\mathbf{Y} - \mathbf{e}_\lambda \cdot \mathbf{y}^\top)$

Include $(\mathcal{W}_\mathbf{Y}, \mathcal{W}_\mathbf{y})$ in $\mathcal{W}$

Generate $(\mathcal{W}_\mathbf{C}, \mathcal{W}_{\mathsf{st}})$ and $(\mathcal{W}_{\mathbf{P}_i}, \mathcal{W}_{\mathbf{S}'_i})_{i\in[l_{\mathsf{st}}]}$ for the state update as specified in Figure 20 and include them in $\mathcal{W}$

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

Figure 19: Generation of wire assignments in the intermediate games $\mathsf{G}_1, \cdots, \mathsf{G}_5$. $\{\mathsf{SimAdd}_\lambda, \mathsf{SimMult}_\lambda\}$ are function families defined as in Figure 5.

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

State update:

Run

$$\mathbf{r} \xleftarrow{\$} \{0,1\}^{\lambda(\lambda-1)/2}, \overline{\mathbf{M}'} = \mathsf{M}_\lambda(\mathbf{r})^\top, \overline{\mathbf{U}'} \xleftarrow{\$} \{0,1\}^{(\lambda-1)\times\lambda}$$

$$\mathbf{M}'^\top \xleftarrow{\$} \mathsf{OneSamp}_\lambda,\ \mathbf{V}' \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda},\ \mathbf{U}' = \mathbf{M}'\mathbf{V}'$$

Run

$$\mathbf{C} \xleftarrow{\$} \{0,1\}^{\lambda\times(\lambda-1)}, \mathbf{R}_{\mathsf{st}} \xleftarrow{\$} \{0,1\}^{\lambda\times 2\lambda}$$

$$\boxed{\mathbf{H} \xleftarrow{\$} \{0,1\}^{\lambda\times(\lambda-1)},\ \widetilde{\mathbf{r}} \xleftarrow{\$} \{0,1\}^{\lambda-1},\ \mathbf{C} = \mathbf{M}\mathbf{H} + \mathbf{e}_\lambda \cdot \widetilde{\mathbf{r}}^\top}$$

$$\mathbf{C} \xleftarrow{\$} \{0,1\}^{\lambda\times(\lambda-1)}$$

Run

$$(\mathcal{W}_{\mathbf{C}}, \mathbf{C}') \Leftarrow \mathbf{C}(\overline{\mathbf{M}'}||\overline{\mathbf{U}'}) + \mathbf{U}\mathbf{R}_{\mathsf{st}}, \left(\mathcal{W}_{\mathsf{st}}, (\underline{\mathbf{M}'}||\underline{\mathbf{U}'})\right) \Leftarrow \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{C}')$$

$$\boxed{\mathcal{W}_{\mathsf{st}} = \mathsf{SimDec}_\lambda\left(\mathbf{M}, \mathbf{H}(\overline{\mathbf{M}'}||\overline{\mathbf{U}'}) + \mathbf{V}\mathbf{R}_{\mathsf{st}}, \widetilde{\mathbf{r}}^\top(\overline{\mathbf{M}'}||\overline{\mathbf{U}'})\right)}$$

$$\mathcal{W}_{\mathsf{st}} = \mathsf{SimDec}_\lambda\left(\mathbf{M}, \mathbf{M}^{-1}(\mathbf{C}' - \mathbf{e}_\lambda \cdot (\underline{\mathbf{M}'}||\underline{\mathbf{U}'})), \underline{\mathbf{M}'}||\underline{\mathbf{U}'}\right)$$

Let $(\mathbf{s}_i)_{i\in[l_{\mathsf{st}}]}$ denote the rightmost columns of the matrices representing the updated state in F_λ and let $\mathbf{r}_i$ denote the rightmost column of the associated randomness matrix $\mathbf{Z}_i$

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

$$\mathbf{P}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}, \mathbf{R}_{\mathbf{S}_i} \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$$

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

$$\mathbf{P}_i \xleftarrow{\$} \{0,1\}^{\lambda^2\times\lambda}$$

and

$$(\mathcal{W}_{\mathbf{P}_i}, \mathbf{P}'_i) \Leftarrow \mathbf{P}_i\mathbf{M}'^\top + (\mathbf{U}\otimes\mathbf{I}_\lambda)\mathbf{R}_{\mathbf{S}_i} + \mathbf{s}_i\otimes\mathbf{I}_\lambda, (\mathcal{W}_{\mathbf{S}'_i}, \mathbf{S}'^\top_i) \Leftarrow \mathsf{Dec}_\lambda(\mathbf{M}, \mathbf{P}'_i)$$

$$\boxed{\mathcal{W}_{\mathbf{S}'_i} = \mathsf{SimDec}_\lambda\left(\mathbf{M}, \mathbf{H}_i\mathbf{M}'^\top + (\mathbf{V}\otimes\mathbf{I}_\lambda)\mathbf{R}_{\mathbf{S}_i} + \mathbf{r}_i\otimes\mathbf{I}_\lambda, \mathbf{S}'^\top_i\right) \text{ for } \mathbf{S}'_i = \mathbf{M}'\mathbf{Q}'_i + s'_i\otimes\mathbf{I}_\lambda}$$

$$\mathcal{W}_{\mathbf{S}'_i} = \mathsf{SimDec}_\lambda\left(\mathbf{M}, (\mathbf{M}\otimes\mathbf{I}_\lambda)^{-1}(\mathbf{P}'_i - (\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)\mathbf{S}'^\top_i), \mathbf{S}'^\top_i\right) \quad \mathbf{S}'_i \xleftarrow{\$} \{0,1\}^{\lambda\times\lambda}$$

Figure 20: Generation of wire assignments in the state updates of the intermediate games.

Lemma C.2 *The views of $\mathcal{A}$ in G_1 and G_2 are identical.*

Proof. Since $\mathbf{M}||\mathbf{e}_\lambda$ is of rank λ for $\mathbf{M}^\top \in \mathsf{ZeroSamp}_\lambda$, $(\mathbf{M}\otimes\mathbf{I}_\lambda||\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda) = (\mathbf{M}||\mathbf{e}_\lambda)\otimes\mathbf{I}_\lambda$ must be of rank λ^2. As a result, $\mathbf{C}$ and each $\mathbf{P}_i$ generated in G_2 are uniformly distributed in $\{0,1\}^{\lambda\times(\lambda-1)}$ and $\{0,1\}^{\lambda^2\times\lambda}$ respectively, completing the proof of Lemma C.2. □

Game G_3. G_3 is the same as G_2 except that $\mathcal{W}_{\mathbf{y}}$, $\mathcal{W}_{\mathsf{st}}$, and $(\mathcal{W}_{\mathbf{S}'_i})_{i\in[l_{\mathsf{st}}]}$ are simulated by SimDec_λ as in Figure 19, instead of being generated by Dec_λ.

Lemma C.3 *If $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$ holds, then the views of $\mathcal{A}$ in G_2 and G_3 are statistically close with negligible distance.*

Proof. As shown in the proof of correctness, we have

$$\mathbf{Y} + \mathbf{U}\mathbf{R}_{\mathbf{y}} = \mathbf{M}(\mathbf{Z} + \mathbf{V}\mathbf{R}_{\mathbf{y}}) + \mathbf{e}_\lambda \cdot \mathbf{y}^\top$$

and

$$\mathbf{C}' = \mathbf{M}\big(\mathbf{H}(\overline{\mathbf{M}'}||\overline{\mathbf{U}'}) + \mathbf{V}\mathbf{R}_{\mathsf{st}}\big) + \mathbf{e}_\lambda \cdot \widetilde{\mathbf{r}}^\top(\overline{\mathbf{M}'}||\overline{\mathbf{U}'}),$$

and for each $i \in [l_{\mathsf{st}}]$,

$$\mathbf{P}'_i = (\mathbf{M}\otimes\mathbf{I}_\lambda)\big(\mathbf{H}_i\mathbf{M}'^\top + (\mathbf{V}\otimes\mathbf{I}_\lambda)\mathbf{R}_{\mathbf{S}_i} + \mathbf{r}_i\otimes\mathbf{I}_\lambda\big) + (\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)\big(\mathbf{Q}'^\top_i\mathbf{M}'^\top + s'_i\otimes\mathbf{I}_\lambda\big).$$

Then fix all the seeds except for $\mathbf{R}_{\mathbf{y}}$, $\mathbf{R}_{\mathsf{st}}$, $(\mathbf{R}_{\mathbf{S}_i})_{i\in[l_{\mathsf{st}}]}$, and $\mathbf{V}$, and the internal randomness of Dec_λ in G_2 and G_3, in a way that the statistical distance between the leakage in the two games is maximized. One can see that the output of ℓ'_λ reveals no information about $\mathcal{W}_{\mathbf{y}}$, $\mathcal{W}_{\mathsf{st}}$, and $(\mathcal{W}_{\mathbf{S}'_i})_{i\in[l_{\mathsf{st}}]}$ and the randomness beyond their NC^1 leakage. Moreover, beyond the leakage produced by ℓ'_λ, no additional information about $\mathbf{V}$ is revealed except for $\mathbf{U} = \mathbf{M}\mathbf{V}$. Therefore, Lemma C.3 follows immediately from Theorem 3.4. □

Game G_4. In G_4, in each invocation, $\mathbf{M}^\top$ is sampled from $\mathsf{OneSamp}_\lambda$, $\mathbf{C}$ and $(\mathbf{P}_i)_{i\in[l_{\mathsf{st}}]}$ are uniformly sampled, and $\mathcal{W}_{\mathbf{y}}$, $\mathcal{W}_{\mathsf{st}}$, and $(\mathcal{W}_{\mathbf{S}'_i})_{i\in[l_{\mathsf{st}}]}$ are generated without using $\mathbf{Z}$, $\mathbf{H}$, and $(\mathbf{H}_i)_{i\in[l_{\mathsf{st}}]}$, as in Figure 20.

Lemma C.4 *If $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$ holds, then the views of $\mathcal{A}$ in G_3 and G_4 are statistically close with negligible distance.*

Proof. We first define a sequence of intermediate games $\mathsf{G}_{3.0}, \cdots, \mathsf{G}_{3.q}$, where $\mathsf{G}_{3.0}$ (respectively, $\mathsf{G}_{3.q}$) is the same as G_3 (respectively, G_4). For each $\mathsf{c} \in [q]$, $\mathsf{G}_{3.\mathsf{c}}$ differs from $\mathsf{G}_{3.\mathsf{c}-1}$ in that the matrix $\mathbf{M}^\top$ in the cth invocation (i.e., $\mathbf{M}'^\top$ in the $(\mathsf{c}-1)$th invocation, or $\mathbf{M}^\top$ in the initialization when $\mathsf{c}=1$) is sampled from $\mathsf{OneSamp}_\lambda$ instead of $\mathsf{ZeroSamp}_\lambda$, and the values $\mathbf{C}$, $\mathcal{W}_{\mathbf{y}}$, $\mathcal{W}_{\mathsf{st}}$, and $(\mathcal{W}_{\mathbf{S}'_i})_{i\in[l_{\mathsf{st}}]}$ in the cth invocation are simulated using $\mathbf{M}^{-1}$ as in G_4 of Figure 20.

For an arbitrary $\mathsf{c} \in [q]$, we fix all randomness in $\mathsf{G}_{3.\mathsf{c}-1}$ except for the internal randomness used by $\mathsf{ZeroSamp}_\lambda$ to generate $\mathbf{M}^\top$ in the cth invocation, in a way that maximizes the adversary's distinguishing advantage conditioned on this fixing. Under this conditioning, no information on $\mathbf{M}$ is revealed except for the 2-adaptive NC^1 leakage on it, since all gate gadgets can be simulated in parallel in $\mathsf{AC}^0[2]$ and $\{\mathsf{SimDec}_\lambda\} \in \mathsf{AC}^0[2]$. Moreover, $\mathbf{M}$ is independent of all wire assignments in the first $\mathsf{c}-2$ invocations (if $\mathsf{c} > 2$) and in all invocations after the cth one.

Therefore, by Lemma 2.9, if $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L/poly}$ holds, the view in the hybrid where $\mathbf{M}^\top$ is sampled from $\mathsf{OneSamp}_\lambda$ in the cth invocation is statistically close to the view in $\mathsf{G}_{3.\mathsf{c}-1}$ with negligible distance.

Furthermore, when $\mathbf{M}^\top \in \mathsf{OneSamp}_\lambda$ (i.e., $\mathbf{M}$ has full rank), the matrices $\mathbf{C}$ and $(\mathbf{P}_i)_{i\in[l_{\mathsf{st}}]}$ are uniformly distributed. In this case, we may sample $\mathbf{M}'^\top$ directly from $\mathsf{ZeroSamp}_\lambda$, since $\mathbf{C}$ no longer reveals any information about $\widetilde{\mathbf{r}}$.

Finally, as shown in the proof of Lemma C.3, we have

$$\mathbf{Y} = \mathbf{M}\mathbf{Z} + \mathbf{e}_\lambda \cdot \mathbf{y}^\top$$

and

$$\mathbf{C}' = \mathbf{M}\big(\mathbf{H}(\overline{\mathbf{M}'}||\overline{\mathbf{U}'}) + \mathbf{V}\mathbf{R}_{\mathsf{st}}\big) + \mathbf{e}_\lambda \cdot (\underline{\mathbf{M}'}||\underline{\mathbf{U}'}),$$

and for each $i \in [l_{\mathsf{st}}]$,

$$\mathbf{P}'_i = (\mathbf{M}\otimes\mathbf{I}_\lambda)\big(\mathbf{H}_i\mathbf{M}'^\top + (\mathbf{V}\otimes\mathbf{I}_\lambda)\mathbf{R}_{\mathbf{S}_i} + \mathbf{r}_i\otimes\mathbf{I}_\lambda\big) + (\mathbf{e}_\lambda\otimes\mathbf{I}_\lambda)\mathbf{S}'^\top_i.$$

Hence, we have

$$\mathbf{Z} = \mathbf{M}^{-1}(\mathbf{Y} - \mathbf{e}_\lambda \cdot \mathbf{y}^\top),$$
$$\mathbf{H}(\overline{\mathbf{M}'}||\overline{\mathbf{U}'}) + \mathbf{V}\mathbf{R}_{\mathsf{st}} = \mathbf{M}^{-1}\big(\mathbf{C}' - \mathbf{e}_\lambda \cdot (\underline{\mathbf{M}'}||\underline{\mathbf{U}'})\big),$$

and

$$\mathbf{H}_i\mathbf{M}'^\top + (\mathbf{V} \otimes \mathbf{I}_\lambda)\mathbf{R}_{\mathbf{S}_i} + \mathbf{r}_i \otimes \mathbf{I}_\lambda = (\mathbf{M} \otimes \mathbf{I}_\lambda)^{-1}\big(\mathbf{P}'_i - (\mathbf{e}_\lambda \otimes \mathbf{I}_\lambda)\mathbf{S}'^\top_i\big).$$

Consequently, the view of $\mathcal{A}$ in $\mathsf{G}_{3.\mathsf{c}}$ is statistically close to its view in $\mathsf{G}_{3.\mathsf{c}-1}$ with negligible distance under $\mathsf{NC}^1 \subsetneq \oplus\mathsf{L}/\mathsf{poly}$. Finally, applying Lemma 2.9 once more switches $\mathbf{M}'^\top$ generated in the qth invocation from $\mathsf{ZeroSamp}_\lambda$ to $\mathsf{OneSamp}_\lambda$, yielding G_4. Applying a hybrid argument completes the proof of Lemma C.4. □

Game G_5. G_5 differs from G_4 in that $\mathbf{S}_i$ are sampled uniformly for all $i \in [l_{\mathsf{st}}]$.

Lemma C.5 *The views of $\mathcal{A}$ in G_4 and G_5 are identical.*

Proof. In each invocation of G_4, $\mathbf{M}^\top$ is sampled from $\mathsf{OneSamp}_\lambda$ and therefore has full rank. Moreover, the values revealed to $\mathcal{A}$ do not leak any information about $\mathbf{Q}'_i$ for any i, except through $\mathbf{S}'_i$. As a result, replacing $\mathbf{S}_i$ with uniformly random matrices for all i does not change the view of $\mathcal{A}$, completing the proof of Lemma C.5. □

Game G_6. G_6 is the ideal game $\mathsf{Ideal}_{\mathcal{A}}$ (see Figure 3), where in each invocation, the assignments to all wires of the LRC are simulated by Sim^*_λ (see Figure 18). Namely, each addition (respectively, multiplication) gate in F_λ is implemented using Add_λ (respectively, Mult_λ) now.

Lemma C.6 *The views of $\mathcal{A}$ in G_5 and G_6 are identical.*

Lemma C.6 follows immediately from Lemma 3.2.
Putting all the above together, Theorem 5.1 immediately follows. □

D Proof of Theorem 6.3

In this section, we give the proof of Theorem 6.3.

At a high level, correctness and soundness of NIZK follow from the correctness of the underlying sAI-LTC and encoding scheme. For zero-knowledge with offline simulation, we use the sAI-LTC simulator together with dummy encodings to generate a simulated proof that fools the adversary.

Proof. **Complexity.** We first note that $(\mathsf{Gen}_\lambda, \mathsf{Ver}_\lambda) \in \widetilde{\mathcal{L}}_\lambda$, since $\mathsf{LTSetup}_\lambda \in \widetilde{\mathcal{L}}_\lambda$ and all verification checks can be performed in parallel by an AC^0 circuit with depth 3.

Correctness. By the correctness of $\mathsf{sAI\text{-}LTC}$ and that of the encoding scheme, 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 $(\mathbf{x}, \mathbf{w})$ with $\mathsf{R}_\lambda(\mathbf{x}, \mathbf{w}) = 1$, all $\mathsf{ct} \in \mathsf{E}_\lambda(\mathbf{w})$, and all $\mathsf{aux} \in \mathsf{LTSetup}_\lambda$, completing the proof of correctness.

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

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

by the correctness of $\mathsf{sAI\text{-}LTC}$. Hence, there must exist a valid witness $\mathbf{w} = \mathsf{D}_\lambda(\mathsf{ct})$, completing the proof of soundness.

Multi-theorem zero-knowledge with offline simulation. Let $\lambda \in \mathbb{N}$ be the security parameter and $q = q(\lambda)$ be any polynomial. Let $\{\mathsf{DSim}_\lambda\}$ be the distribution family guaranteed by the parallel-composable sAI-LTC $\{\mathsf{LTSetup}_\lambda, \mathsf{LTComp}^{\|q}_\lambda\}$. The distribution family $\{\mathcal{D}^{\mathsf{NIZK}}_\lambda\}$ for simulated CRSs and simulators is given in Figure 21.

We first note that each $\mathsf{NIZKSim}_\lambda$ sampled from $\mathcal{D}^{\mathsf{NIZK}}_\lambda$ runs in $\widetilde{\mathcal{L}}_\lambda$, since each LTSim_λ sampled from DSim_λ runs in $\widetilde{\mathcal{L}}_\lambda$. Let $(\mathbf{x}_i, \mathbf{w}_i)$ be any sequence of statement/witness pairs such that $\mathsf{R}_\lambda(\mathbf{x}_i, \mathbf{w}_i) = 1$ for

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

Sample $(\mathsf{aux}, \mathsf{LTSim}_\lambda) \xleftarrow{\$} \mathsf{DSim}_\lambda$ and compute $\mathsf{ct}_i \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{0})$ for all $i \in [q]$

Return $(\mathsf{aux}, \mathsf{NIZKSim}_\lambda)$, where $\mathsf{NIZKSim}_\lambda$ is defined as below with LTSim_λ and $(\mathsf{ct}_i)_{i\in[q]}$ hardwired into it

$\underline{\mathsf{NIZKSim}_\lambda((\mathbf{x}_i)_{i=1}^q):}$

Compute $\mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda\big(((\mathbf{x}_i, \mathsf{ct}_i)_{i\in[q]}, \mathbf{1})\big)$

Return $\pi = \mathcal{W}$

Figure 21: Definition of $\{\mathcal{D}_\lambda^{\mathsf{NIZK}}\}$.

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

Sample $\mathsf{aux} \xleftarrow{\$} \mathsf{LTSetup}_\lambda$

Compute $\mathsf{ct}_i \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{w}_i)$ for all $i \in [q]$ $\boxed{\mathsf{ct}_i \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{0}) \text{ for all } i \le \mathsf{c}}$,

Compute $\mathcal{W} \xleftarrow{\$} \mathsf{W}[\mathsf{LTComp}_\lambda^{\|q}((\mathsf{aux}, \mathbf{x}_i, \mathsf{ct}_i)_{i\in[q]})]$

$\boxed{\text{Sample } (\mathsf{aux}, \mathsf{LTSim}_\lambda) \xleftarrow{\$} \mathsf{DSim}_\lambda \text{ and compute } \mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathbf{x}_i, \mathsf{ct}_i)_{i\in[q]}, \mathbf{1})}$

Return $\mathsf{crs} = \mathsf{aux}$ and $\pi = \mathcal{W}$

Figure 22: Generation of (crs, π) in the intermediate games.

all $i \in [q]$, and let $\mathcal{A}_\lambda$ be any distinguisher in $\mathcal{L}_\lambda$. We define a sequence of hybrid games. In each game, $\mathcal{A}_\lambda$ receives a CRS crs and a proof π generated as in Figure 22.

Game G_0. In G_0, $\mathcal{A}_\lambda$ is given $\mathsf{crs} = \mathsf{aux}$ and $\pi = \mathcal{W} = (\mathcal{W}_i)_{i\in[q]}$, where $\mathsf{aux} \xleftarrow{\$} \mathsf{LTSetup}_\lambda$, and

$$\mathsf{ct}_i \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{w}_i) \text{ and } \mathcal{W}_i \xleftarrow{\$} \mathsf{W}[\mathsf{LTComp}_\lambda(\mathsf{aux}, \mathbf{x}_i, \mathsf{ct}_i)] \text{ for all } i \in [q]$$

$$\text{(equivalently, } \mathcal{W} \xleftarrow{\$} \mathsf{W}[\mathsf{LTComp}_\lambda^{\|q}((\mathsf{aux}, \mathbf{x}_i, \mathsf{ct}_i)_{i\in[q]})]).$$

It then returns a bit β.

Game G_1. G_1 is identical to G_0, except that aux and $\mathcal{W}$ are generated as

$$(\mathsf{aux}, \mathsf{LTSim}_\lambda) \xleftarrow{\$} \mathsf{DSim}_\lambda \text{ and } \mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathbf{x}_i, \mathsf{ct}_i)_{i\in[q]}, \mathbf{1}).$$

Lemma D.1 *The distribution of $\mathcal{A}_\lambda(\mathsf{crs}, \pi)$ in G_0 is statistically close to that in G_1 with negligible distance.*

Proof. In G_0, we first fix the randomness of E_λ, such that the adversary's distinguishing advantage is maximized conditioned on this fixing. We then define a leakage function ℓ_λ that, on input aux and $\mathcal{W}$, returns $\mathcal{A}_\lambda(\mathsf{aux}, \mathcal{W})$.

Since $\mathcal{A}_\lambda \in \mathcal{L}_\lambda$, we have $\ell_\lambda \in \mathcal{L}_\lambda$. Moreover, when

$$\mathsf{aux} \xleftarrow{\$} \mathsf{LTSetup}_\lambda \text{ and } \mathcal{W} \xleftarrow{\$} \mathsf{W}[\mathsf{LTComp}_\lambda^{\|q}((\mathsf{aux}, \mathbf{x}_i, \mathsf{ct}_i)_{i\in[q]})]$$

$$\text{(respectively,} (\mathsf{aux}, \mathsf{LTSim}_\lambda) \xleftarrow{\$} \mathsf{DSim}_\lambda \text{ and } \mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathbf{x}_i, \mathsf{ct}_i)_{i\in[q]}, \mathbf{1})),$$

the view of $\mathcal{A}_\lambda$ is exactly the same as its view in G_0 (respectively, G_1).

As a result, by the $\{\mathcal{L}_\lambda\}$-leakage-tolerance of $\mathsf{sAI\text{-}LTC}$, the output distribution of $\mathcal{A}_\lambda$ in G_1 is statistically close to that in G_0 with negligible distance, completing the proof of Lemma D.1. [6] □

[6] Note that the view of $\mathcal{A}_\lambda$ additionally contains aux, while this does not affect the analysis since aux is already included in $\mathcal{W}$.

Game $\mathsf{G}_{2.\mathsf{c}}$. $\mathsf{G}_{2.0}$ is identical to G_1. For all $\mathsf{c} \in [q]$, $\mathsf{G}_{2.\mathsf{c}}$ is identical to $\mathsf{G}_{2.\mathsf{c}-1}$ except that ct_c is generated as $\mathsf{ct}_\mathsf{c} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{0})$. One can see that $\mathsf{G}_{2.q}$ is exactly the game in which $\mathcal{A}_\lambda$ is given simulated crs and π.

Lemma D.2 *The distribution of $\mathcal{A}_\lambda(\mathsf{crs}, \pi)$ in $\mathsf{G}_{2.\mathsf{c}-1}$ is statistically close to that in $\mathsf{G}_{2.\mathsf{c}}$ with negligible distance.*

Proof. In $\mathsf{G}_{2.\mathsf{c}-1}$ and $\mathsf{G}_{2.\mathsf{c}}$, we first fix all the randomness except that used to generate ct_c, such that the adversary's distinguishing advantage is maximized conditioned on this fixing. We then define a leakage function ℓ'_λ that on input ct_c runs $\mathcal{W} \xleftarrow{\$} \mathsf{LTSim}_\lambda((\mathbf{x}_i, \mathsf{ct}_i)_{i\in[q]}, \mathbf{1})$, where $(\mathsf{ct}_i)_{i\neq\mathsf{c}}$ are fixed, and returns $\mathcal{A}_\lambda(\mathsf{aux}, \mathcal{W})$.

Since $\mathsf{LTSim}_\lambda \in \widetilde{\mathcal{L}}_\lambda$ and $\mathcal{A}_\lambda \in \mathcal{L}_\lambda$, we have $\ell'_\lambda \in \mathcal{L}_\lambda$. Moreover, when $\mathsf{ct}_\mathsf{c} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{w}_\mathsf{c})$ (respectively, $\mathsf{ct}_\mathsf{c} \xleftarrow{\$} \mathsf{E}_\lambda(\mathbf{0})$), the view of $\mathcal{A}_\lambda$ is exactly the same as its view in $\mathsf{G}_{2.\mathsf{c}-1}$ (respectively, $\mathsf{G}_{2.\mathsf{c}}$).

As a result, by the $\{\mathcal{L}_\lambda\}$-leakage-resilience of the encoding scheme, the output distribution of $\mathcal{A}_\lambda$ in $\mathsf{G}_{2.\mathsf{c}}$ is statistically close to that in $\mathsf{G}_{2.\mathsf{c}-1}$ with negligible distance, completing the proof of Lemma D.2. □

Putting all the above together, Theorem 6.3 immediately follows. □