Fully Homomorphic NIZK and NIWI Proofs

Prabhanjan Ananth[*] MIT
Apoorvaa Deshpande[†] Brown University
Yael Tauman Kalai[‡] MSR and MIT
Anna Lysyanskaya[§] Brown University

June 20, 2019

Abstract

In this work, we define and construct *fully homomorphic* non-interactive zero knowledge (FH-NIZK) and non-interactive witness-indistinguishable (FH-NIWI) proof systems.

We focus on the NP complete language L, where, for a boolean circuit C and a bit b, the pair $(C, b) \in L$ if there exists an input $\mathbf{w}$ such that $C(\mathbf{w}) = b$. For this language, we call a non-interactive proof system *fully homomorphic* if, given instances $(C_i, b_i) \in L$ along with their proofs Π_i, for $i \in \{1, \ldots, k\}$, and given any circuit $D : \{0,1\}^k \to \{0,1\}$, one can efficiently compute a proof Π for $(C^*, b) \in L$, where $C^*(\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(k)}) = D(C_1(\mathbf{w}^{(1)}), \ldots, C_k(\mathbf{w}^{(k)}))$ and $D(b_1, \ldots, b_k) = b$. The key security property is *unlinkability*: the resulting proof Π is indistinguishable from a fresh proof of the same statement.

Our first result, under the Decision Linear Assumption (DLIN), is an FH-NIZK proof system for L in the common random string model. Our more surprising second result (under a new decisional assumption on groups with bilinear maps) is an FH-NIWI proof system that requires no setup.

[*]Email: `prabhanjan@csail.mit.edu`.
[†]Email: `acdeshpa@cs.brown.edu`. Part of this work was done at Microsoft Research, New England.
[‡]Email: `yael@microsoft.com`.
[§]Email: `anna@cs.brown.edu`.

Contents

1 Introduction

Homomorphism is a desirable feature that enhances the capabilities of many cryptographic systems. Most notably, the concept of fully homomorphic encryption [RAD78, Gen09, BV14] has revolutionized the area of cryptography. Other primitives such as homomorphic signatures [BF11, GVW15] and homomorphic secret sharing [BGI$^+$18] have also found useful cryptographic applications [KW18, BCG$^+$17]. In this work, we study homomorphism in the context of non-interactive proof systems. Our goal is to design homomorphic proof systems with secrecy guarantees; specifically, we focus on the most common secrecy guarantees studied in the literature, namely zero-knowledge [BDMP91] and witness indistinguishability [BOV05, DN00].

Our Work: Fully-Homomorphic NIZK and NIWI Proofs. We introduce the notion of fully-homomorphic non-interactive zero-knowledge (FH-NIZK) and witness-indistinguishable (FH-NIWI) proof systems. In the simplest setting, this proof system allows for combining proofs for the instances A and B into a proof for the instance $A \wedge B$. In the more general setting, this proof system allows for combining proofs for multiple instances $A_1, \ldots, A_n$ using a function f into a single proof for $f(A_1, \ldots, A_n)$.

A naïve attempt to combine proofs for the instances $(A_1, \ldots, A_n)$ using a function f is to simply output the concatenation of the individual proofs on each of the instances $A_1, \ldots, A_n$ together with the function f. However, this combined proof does not resemble an honestly generated proof for the instance $f(A_1, \ldots, A_n)$. Our goal is to combine proofs in a way that is indistinguishable from an honestly generated proof for the instance $f(A_1, \ldots, A_n)$. We call this property *unlinkability.*

There are several reasons why unlinkability is an interesting feature: Firstly, it is often desirable to hide the fact that a proof was obtained by combining multiple proofs. Unlinkability also preserves the privacy of the underlying proof; namely, it ensures that homomorphic evaluation of multiple NIZK (resp., NIWI) proofs still results in a NIZK (resp., NIWI) proof. Moreover, it guarantees that the homomorphic evaluation can be multi-hop, meaning that the proofs can be evaluated upon multiple times. We describe the homomorphic evaluation procedure and unlinkability property below.

We define the notion of a fully-homomorphic proof system for the NP-complete language $L_{\mathcal{U}}$ which consists of instances (C, b), where C is a boolean circuit with single-bit output and b is a bit, such that there exists a witness $\mathbf{w}$ (a vector of bits) for which $C(\mathbf{w}) = b$. A non-interactive proof system for proving membership in this language consists of the algorithms Prove and Verify. A fully homomorphic proof system additionally has the algorithm Eval defined as follows:

Homomorphic Evaluation (Eval): On input k instances $\{z_i = (C_i, b_i)\}_{i\in[k]}$ accompanied with proofs $\{\Pi_i\}_{i\in[k]}$ for the statements $\{z_i \in L_{\mathcal{U}}\}_{i\in[k]}$, and a circuit $D : \{0,1\}^k \to \{0,1\}$, Eval outputs a proof Π^* for the statement $z^* = (C^*, D(b_1, \ldots, b_k)) \in L_{\mathcal{U}}$, where C^* is defined to be the circuit that on input $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$ outputs $D(C_1(\mathbf{w}_1), \ldots, C_k(\mathbf{w}_k))$.

We define *unlinkability* as follows: A proof Π^* output by Eval on input $\{z_i \in L_{\mathcal{U}}\}_{i\in[k]}$ accompanied with proofs $\{\Pi_i\}_{i\in[k]}$, where Π_i is output by Prove on input z_i and a valid witness $\mathbf{w}_i$, should be indistinguishable from the output of Prove on input the instance $(C^*, D(b_1, \ldots, b_k))$ and witness $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$. As mentioned above, unlinkability guarantees that the evaluation property preserves zero-knowledge (ZK) or witness-indistinguishability (WI) of an evaluated proof, depending on whether the fresh proof is ZK or WI respectively. We refer the reader to Figure 1 for an illustrative description of unlinkability, and refer the reader to Section 4 for our definition of fully homomorphic proofs.

Our Results. We construct both a NIZK and a NIWI fully homomorphic proof system.

Theorem 1 (Informal)**.** *Assuming Decisional Linear Assumption (DLIN), there exists a fully-homomorphic non-interactive zero-knowledge proof system in the common random string model.*

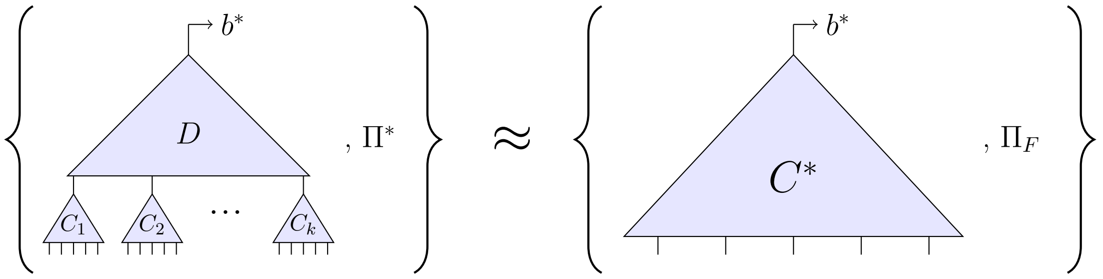

Figure 1: Unlinkability property of Fully Homomorphic Proofs: Let Π^* be the output of Eval on input $\{(C_i, b_i) \in L_{\mathcal{U}}\}_{i\in[k]}$ accompanied with proofs $\{\Pi_i\}_{i\in[k]}$, where Π_i is output by Prove on input (C_i, b_i) and a valid witness $\mathbf{w}_i$. Let C^* be the circuit that on input $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$, outputs $D(C_1(\mathbf{w}_1), \ldots, C_k(\mathbf{w}_k))$ and let Π_F be an honestly generated proof for the instance $(C^*, b^*) \in L_{\mathcal{U}}$. We require that Π^* is computationally indistinguishable from Π_F.

For constructing FH NIWI proofs, we rely on a new decisional assumption on groups with bilinear maps called *DLIN with leakage*, defined in Figure 2 (below).

Let f, h, g be three random generators in a group $\mathbb{G}$. The assumption states that $\mathcal{D}_0(1^\lambda) \approx_c \mathcal{D}_1(1^\lambda)$, where:

– $\mathcal{D}_0(1^\lambda)$: Choose $R, S, t \leftarrow \mathbb{Z}_p^*$ and output (f, h, g) along with the following matrix:

$$\begin{bmatrix} f^R & h^S & g^{R+S} \\ f^{R^2} & h^{RS-t} & g^{R(R+S+1)-t} \\ f^{RS+t} & h^{S^2} & g^{S(R+S+1)+t} \end{bmatrix}$$

– $\mathcal{D}_1(1^\lambda)$: Choose $R, S, t \leftarrow \mathbb{Z}_p^*$ and output (f, h, g) along with the following matrix:

$$\begin{bmatrix} f^R & h^S & g^{R+S-1} \\ f^{R^2} & h^{RS-t} & g^{R(R+S-1)-t} \\ f^{RS+t} & h^{S^2} & g^{S(R+S-1)+t} \end{bmatrix}$$

Figure 2: Description of the DLIN with leakage, with respect to a group $\mathbb{G}$ of prime order p with a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We refer to this as DLIN with leakage assumption since the first row in both the distributions are indistinguishable assuming DLIN, and the second and third rows can be viewed as leakage.

For a more detailed description of the assumption, we refer the reader to Section 5.3. We prove that our assumption holds in the bilinear generic group model in Appendix A.1.

Theorem 2 (Informal)**.** *Assuming DLIN with Leakage, there exists a fully-homomorphic non-interactive witness-indistinguishable proof system in the plain model (i.e. without setup).*

Related Works. Most relevant to our work is the work on malleable proof systems [CKLM12, CKLM13b], who studied unary transformations, i.e., when Eval receives a *single* instance-proof pair and outputs a mauled instance along with the corresponding proof. The work of [CKLM12] studied malleable proof systems for specific relations, and [CKLM13b] studied malleability for general relations albeit under knowledge assumptions. Moreover, these works consider NIZK proof systems and thus require trusted setup. We note that [CKLM12] satisfies a stronger proof of knowledge property (called controlled-malleable simulation-sound extractability) that we don't achieve in this work.

The notion of malleability, although seemingly limited due to its unary nature, has found many applications, such as verifiable shuffles [CKLM12], delegatable anonymous credentials [BCC$^+$09a, CKLM13a] and leakage-resilient proof systems [AGP14]. Re-randomizability [BCC$^+$09a], a special case of malleability, has also been studied in the literature. Following [CKLM12, CKLM13b], [ACJ17] construct *privately* malleable NIZK proof systems, and the works of [AN11, AGM18] study homomorphic proof systems for specific relations.

It is important to stress that all the prior works, even in the case of unary transformations studied in the context of malleable proofs [CKLM12, CKLM13b], assume trusted setup. Thus, in the context of WI proof systems, our results are especially surprising since it allows for combining proofs that were created completely independently, with no shared setup.

We now describe some applications of fully-homomorphic proofs.

Private Incremental Proofs. Incremental proofs, introduced by Valiant [Val08], allow for merging many computationally sound proofs [Mic00] into one proof which is as short and easily verifiable as the original proofs. Incremental proofs have been applied in several contexts such as proof-carrying data [BCCT13] and cryptographic image authentication mechanisms [NT16]. It is useful in two types of settings: one where the computation dynamically evolves over a period of time, hence a proof of correctness of the entire computation cannot be computed all at once, and the other where different entities wish to compute a proof for the correctness of computation in a distributed setting.

The focus of prior works on incremental proofs was on succinctness whereas the focus of our work is on privacy. While our work does not achieve succinctness, as we will see later achieving privacy alone turns out to be quite challenging (especially, in the context of fully-homomorphic NIWIs). We hope that our tools can be combined with succinct incremental proofs to yield incremental proofs that enjoy both succinctness and privacy guarantees.

Commit-and-Compute Paradigm. Another application of fully-homomorphic proofs is the commit-and-compute paradigm. At a high level, the commit-and-compute paradigm allows a prover to commit to its sensitive data, and later on, prove statements about the committed data. Proofs from different provers can then be combined to infer arbitrary statements about the committed data. We give two examples that illustrate the applicability of this paradigm: (i) Regulation of cryptocurrency activities and, (ii) Verifiable data analysis.

Regulation of Cryptocurrency Activities. New regulation laws regarding cryptocurrency activities are being formulated and some of them require that financial transactions involving cryptocurrencies be reported [oSBS15]. The regulators can then infer different conclusions about the state of the digital economy; for instance, they can conclude the debt of different entities, and publish the findings for the public. The involved entities may have motives to lie about their finances and this would in turn lead the regulators to arrive at false conclusions. We can address this problem using fully homomorphic NIZK or NIWI proofs: Each entity can now commit to their financial transactions (to maintain privacy) and submit a proof that the financial transactions reported to the regulators are valid. The regulators can collect the data and the proofs from all the entities, publish its conclusions along with a proof that is obtained by homomorphically computing on the individual proofs.

Verifiable Data Analysis. Consulting firms often collect data from different research groups, perform analysis on the joint dataset and then share the analyzed results with different organizations. For instance, there are firms that collect medical data from different research groups and share the analysis on the medical data to pharmaceutical companies. This raises concerns about trusting the research groups and the consulting firms to not lie about their conclusions. We can tackle this concern by using fully homomorphic NIZK or NIWI proofs. The research groups can publish their (committed) data along with a proof that it was collected from valid sources, without revealing the identity of the sources. The consulting firms can then perform analysis on the joint data sets and homomorphically compute a proof that the analysis was performed correctly. Moreover, the homomorphically computed proof will also hide the identities of the research groups involved in sharing the data to the firms.

Commit-and-compute paradigm is formalized by defining the NP language L_{COM}, a modification of $L_{\mathcal{U}}$ so that the instance includes a vector of commitments along with (C, b). The language L_{COM} is as follows:

$$L_{\mathsf{COM}} = \big\{(C, (\mathsf{com}_1, \ldots, \mathsf{com}_n), b) \;\mid\; \exists \{w_i, r_i\} \text{ such that } C(w_1, \ldots, w_n) = b \wedge \{\mathsf{com}_i = \mathsf{Commit}(w_i, r_i)\}\big\}$$

The evaluation is defined similarly to that of homomorphic Eval for $L_{\mathcal{U}}$. We show how to instantiate the commit-and-compute paradigm using fully-homomorphic proofs in Section 8.

Roadmap for Rest of the Sections. In Section 2, we give an overview of our techniques. In Section 3, we describe some notation and definitions. In Section 4, we present our definition of fully homomorphic NIZK and NIWI proof systems. In Section 5, we define and instantiate the building blocks for our constructions, and describe our DLIN with Leakage assumption (in Section 5.3). In Section 6, we construct fully homomorphic NIZK proofs for NP from DLIN. In Section 7, we construct fully homomorphic NIWI proofs from the DLIN with Leakage assumption. Finally in Section 8, we define and instantiate the Commit-and-Compute paradigm.

2 Technical Overview

Let us start with some intuition. Suppose we want to generate a proof for the satisfiability of $C_1 \wedge C_2$ for some circuits C_1, C_2. Given a proof Π_1 for the satisfiability of C_1 and a proof Π_2 for the satisfiability of C_2, clearly $\Pi = (\Pi_1, \Pi_2)$ is a proof for the satisfiability of $C_1 \wedge C_2$. However, such a proof does not satisfy unlinkability. Moreover, the structure of the proof $\Pi = (\Pi_1, \Pi_2)$ may be different from that of a fresh proof computed for the satisfiability of $C_1 \wedge C_2$.

To achieve homomorphism and unlinkability, a natural candidate is a proof system that works gate-by-gate as follows: Commit to all the wire values of the circuit and prove that each gate is consistent with the committed values. Such a proof structure is a good candidate because structurally, a proof of the composed instance $C_1 \wedge C_2$ will be similar to a fresh proof.

Indeed the beautiful work of Groth, Ostrovsky and Sahai [GOS06a] (henceforth referred to as GOS) has this proof structure and it is the starting point for our FH NIZK construction as well as our FH NIWI construction. GOS constructed NIZK and NIWI proofs under the decisional linear (DLIN) assumption. First in Section 2.1, we describe our FH NIZK construction which builds on the GOS NIZK. Then in Section 2.2, we describe our FH NIWI construction which contains the bulk of the technical difficulty in this work.

2.1 Overview: Fully Homomorphic NIZK

Recall that an $L_{\mathcal{U}}$ instance is of the form (C, out) where $C : \{0,1\}^t \rightarrow \{0,1\}$ and $\mathsf{out} \in \{0,1\}$. Let $\mathbf{w} = (w_1, \ldots, w_t)$ be a witness such that $C(\mathbf{w}) = \mathsf{out}$. Let us first recall the GOS NIZK proof for $L_{\mathcal{U}}$.

GOS NIZK. The GOS NIZK proof system is associated with a commitment scheme with public parameters (as we elaborate on later). The CRS consists of the parameters pp for the commitment scheme. The prover on input (C, out) along with witness $\mathbf{w}$ does the following:

1. Let $w_1, \ldots, w_n$ be the values induced by witness $\mathbf{w} = (w_1, \ldots, w_t)$ on all the wires of the circuit C. Commit to all the wire values with respect to pp, except the output wire. For every $i \in [n-1]$, denote by $\mathbf{c}_i$ the commitment to wire value w_i. Denote by $\mathbf{c}_n = w_n$.
2. For each $i \in [n]$, prove that the commitment $\mathbf{c}_i$ is a commitment to a boolean value. We refer to such proofs by *Bit Proofs.*
3. For each gate in C, prove that the commitments to the input and the output wires of the gate are consistent with the gate functionality. We refer to such proofs by *Gate Proofs.*

In their construction, GOS use a commitment scheme which has two indistinguishable modes of public parameters: perfectly binding and perfectly hiding. Loosely speaking, the perfectly binding mode is used to argue perfect soundness, and the perfectly hiding mode is used to argue zero-knowledge. In addition, they require the commitment scheme to be additively homomorphic and the additive homomorphism is used in the Gate Proofs.

GOS constructed NIWI proof systems for Bit Proofs and Gate Proofs, and proved that this is sufficient for their NIZK construction. Both Bit and Gate Proofs are computed using the openings of the commitments as the witness. Our FH NIZK construction follows a similar template (our NIZK construction is identical to the GOS NIZK) but in order to achieve unlinkability, we need additional properties from the commitment scheme as well as from the Bit Proofs and Gate Proofs, as we explain below.

Homomorphic Evaluation. Homomorphic evaluation works as follows: On input k instances $\{z_i = (C_i, b_i)\}_{i\in[k]}$ along with proofs $\{\Pi_i\}_{i\in[k]}$ where each Π_i is a proof that $z_i \in L_{\mathcal{U}}$, and a circuit D, we want to output a proof that $(C^*, b^*) \in L_{\mathcal{U}}$ where C^* is the composed circuit and $b^* = D(b_1, \ldots, b_k)$. First, compute a fresh proof for the circuit D with witness $(b_1, \ldots, b_k)$. Note that the fresh proof for (D, b^*) together with the proofs $\{\Pi_i\}_{i\in[k]}$, forms a verifying proof with respect to (C^*, b^*). This follows from the fact that in each proof Π_i, the output wire b_i is given in the clear. However this combined proof is distinguishable from a fresh proof (given the individual proofs $\{\Pi_i\}_{i\in[k]}$). Thus, to achieve unlinkability, we randomize this entire proof.

Randomizing the NIZK Proof. A proof system is said to be randomizable [BCC+09b] if given a proof Π for an instance x, it is possible to randomize the proof Π to obtain a proof Π' for x, such that Π' is indistinguishable from a fresh proof for x. Randomizability of a proof system is sufficient for achieving unlinkability in our construction, as explained above.

At a high level, we randomize the proof Π as follows: Randomize all the commitments in the proof, and then "update" the existing proofs to be with respect to the randomized commitments. Thus, given the original Bit Proofs and Gate Proofs, we need to be able to "maul" them to be with respect to the new randomized commitments in such a way that the updated proofs are distributed as fresh Bit Proofs and Gate Proofs. We refer to such proofs as *malleable proofs.*

Ingredients for our FH NIZK. In summary, for constructing FH NIZK, we use a commitment scheme from GOS, which is also randomizable (we describe the corresponding scheme $(\mathsf{C.Setup}, \mathsf{C.Commit}, \mathsf{C.Rand})$ in Definition 9, Section 5.1). We also need malleable proof systems for Bit proofs and for Gate proofs (we describe the corresponding proof systems $(\mathsf{Bit.Prove}, \mathsf{Bit.Verify}, \mathsf{Bit.Maul})$ and $(\mathsf{N.Prove}, \mathsf{N.Verify}, \mathsf{N.Maul})$ in Section 6.2).

As shown in GOS, both Bit Proofs and Gate Proofs can be reduced to *proofs of linearity* with respect to the NP language L_{Lin}. The language L_{Lin} is parameterized by three random group elements (f, h, g) in some underlying group $\mathbb{G}$ of prime order (which has a bilinear map), and whose instances consists of pairs (A, B), where $A = (f^{a_1}, h^{a_2}, g^{a_3})$ and $B = (f^{b_1}, h^{b_2}, g^{b_3})$, such that $a_1 + a_2 = a_3$ or $b_1 + b_2 = b_3$[1].

[1]If $a_1 + a_2 = a_3$ then A is said to be a linear tuple.

GOS constructed a NIWI proof for L_{Lin}. Recall that for our purposes, we need malleable proof systems for Bit Proofs and Gate Proofs, and as a result we need the underlying NIWI proof for L_{Lin} to be malleable with respect to randomization. Namely given a pair $(\mathbf{A}, \mathbf{B}) \in L_{\mathsf{Lin}}$ with a NIWI proof Π, it should be possible to maul the proof Π for $(\mathbf{A}, \mathbf{B})$ into a proof Π' for a randomization $(\mathbf{A}', \mathbf{B}')$ of $(\mathbf{A}, \mathbf{B})$. We show that the GOS proof for L_{Lin} has the desired malleability property, and we refer the reader to Section 5.2 for the description of the malleable proof system.

2.2 Overview: Fully Homomorphic NIWI

We now focus on our construction of a FH NIWI proof system for $L_{\mathcal{U}}$. As we will see, this is a significantly harder task compared to the FH NIZK, since NIWI is constructed in the plain model without a CRS.

The GOS NIWI Construction. We will first describe the GOS NIWI proof system. Recall that in the GOS NIZK construction, the CRS consists of the parameters pp of the commitment scheme. In a NIWI construction, there is no CRS. In the GOS NIWI, the prover chooses two parameters $(\mathsf{pp}^0, \mathsf{pp}^1)$ such that it is possible to publicly verify that one of them is binding. The NIWI proof for $(C, \mathsf{out}) \in L_{\mathcal{U}}$ is of the form $(\mathsf{pp}^0, \Pi^0, \mathsf{pp}^1, \Pi^1)$ where Π^b is the NIZK proof with respect to pp^b for each $b \in \{0, 1\}$.

Towards Homomorphic Evaluation and Unlinkability. It is not clear how to use the GOS NIWI construction to construct an FH NIWI. In particular, achieving unlinkability here is significantly harder. Intuitively, the difficulty stems from the fact that even though the GOS NIWI appears to be gate-by-gate, there is an over-arching pair of parameters associated with the entire proof, and this pair is different for different proofs.

In more detail, a fresh GOS NIWI proof as described above has two parameters $(\mathsf{pp}^0, \mathsf{pp}^1)$ associated with it. Thus, if we use an approach similar to the FH NIZK construction for composing proofs, namely if we prove that $(D(C_1, \ldots, C_k), b^*) \in L_{\mathcal{U}}$, given k instances $\{z_i = (C_i, b_i)\}_{i \in [k]}$ along with corresponding proofs $\{\Pi_i\}_{i \in [k]}$, where $b^* = D(b_1, \ldots, b_k)$, then the resulting composed proof will have $2k$ parameters associated with it. It is unclear how to randomize such a composed proof to look like a fresh proof which has only two parameters associated with it.

In order to achieve unlinkability in our construction, we diverge from the GOS construction. Rather than choosing a pair of parameters per proof, we choose a fresh pair of parameters $(\mathsf{pp}_j^0, \mathsf{pp}_j^1)$ for each gate of the circuit. As in the GOS construction, the honest prover chooses one of them to be binding and the other hiding such that one can publicly verify that indeed one of the parameters is binding. Recall that in the GOS NIWI construction, the prover committed to each wire value with respect to two parameters $(\mathsf{pp}^0, \mathsf{pp}^1)$. Now that we are choosing fresh parameters per gate, the question is which parameters do we use to commit to a wire value?

We associate four parameters $\mathsf{pp}_i^0, \mathsf{pp}_i^1, \mathsf{pp}_j^0, \mathsf{pp}_j^1$ with an internal wire between the i^{th} and the j^{th} gate in the circuit. In our construction, we commit to the wire value with respect to all of these parameters and thus, have four commitments $\mathbf{c}_i^0, \mathbf{c}_i^1, \mathbf{c}_j^0, \mathbf{c}_j^1$ per wire. We compute Bit Proofs with respect to each of the four commitments, and compute Gate Proofs for every gate with respect to both parameters associated with that gate.

Ensuring Soundness. Recall that the GOS NIWI consists of two independent NIZK proofs Π^0, Π^1 with respect to parameters $\mathsf{pp}^0, \mathsf{pp}^1$ respectively. Thus, the commitments, Bit Proofs and Gate Proofs with respect to both the parameters are independent of each other, and Π^0, Π^1 are verified separately. This is not the case in our setting.

Our proof contains a pair of parameters per gate, and has four commitments per wire. Thus, we need to prove that the multiple commitments per wire commit to the same value. In particular for soundness, it is sufficient to prove that among the four commitments per wire, the two commitments corresponding to the two binding parameters commit to the same value.

However the verifier does not know which of the four parameters $\mathsf{pp}_i^0, \mathsf{pp}_i^1, \mathsf{pp}_j^0, \mathsf{pp}_j^1$ are binding. All we are guaranteed is that for every gate j, one of $(\mathsf{pp}_j^0, \mathsf{pp}_j^1)$ is binding. So in our construction, we give four pairwise proofs that *each* commitment with respect to gate i commits to the same value as *each* commitment with respect to gate j. Namely, for all $b_1, b_2 \in \{0,1\}$, the commitments $(\mathbf{c}_i^{b_1}, \mathbf{c}_j^{b_2})$ with respect to $\mathsf{pp}_i^{b_1}, \mathsf{pp}_j^{b_2}$ commit to the same value. This ensures consistency with respect to the two binding commitments across gates i, j. This, along with the Bit and Gate proofs will ensure that there is a consistent boolean assignment $w_1, \ldots, w_n$ induced by the witness $\mathbf{w}$ across all the wires of the circuit, such that $C(\mathbf{w}) = \mathsf{out}$.

We emphasize that we do not provide consistency proofs between the two commitments $(\mathbf{c}_i^0, \mathbf{c}_i^1)$ for a gate i, and in fact this is crucial for achieving witness indistinguishability, as we explain later. Towards constructing such pairwise proofs, we define the language L_{TC} [2] which consists of instances of the form $(\mathbf{c}_i, \mathbf{c}_j, \mathsf{pp}_i, \mathsf{pp}_j)$ where commitment $\mathbf{c}_i$ with respect to parameters pp_i and $\mathbf{c}_j$ with respect to pp_j commit to the same bit. See Section 7.1 for a detailed description of the language.

2.2.1 Arguing Witness Indistinguishability

The main challenge is to prove that the final construction is witness indistinguishable even given the additional L_{TC} proofs for instances of the form $(\mathbf{c}_i, \mathbf{c}_j, \mathsf{pp}_i, \mathsf{pp}_j)$. We note that even if the proof system for L_{TC} satisfies WI, we do not know how to argue that the final construction is WI. Intuitively, the issue is that an L_{TC} statement may have a unique witness, in which case WI offers no secrecy. As we explain below, we need our L_{TC} proof system to have a secrecy guarantee of the flavor of strong NIWI (with respect to specific distributions).

To argue WI of our final FH NIWI construction, we prove that a proof Π_0 for $(C, \mathsf{out}) \in L_{\mathcal{U}}$ with respect to witness wit_0 is indistinguishable from a proof Π_1 with respect to witness wit_1. Let us zoom in on a wire k between gates i, j whose value changes from 0 (for wit_0) to 1 (for wit_1). Both Π_0, Π_1 will contain four commitments to the wire k with respect to parameters $\mathsf{pp}_i^0, \mathsf{pp}_i^1, \mathsf{pp}_j^0, \mathsf{pp}_j^1$, along with the four L_{TC} proofs (see Figure 3).

Denote by $\mathsf{PP} = (\mathsf{pp}_i^0, \mathsf{pp}_i^1, \mathsf{pp}_j^0, \mathsf{pp}_j^1)$. Denote by $\mathsf{W}(b)$ the four commitments to bit b on wire k, that is $\mathsf{W}(b) = (\mathbf{c}_i^0, \mathbf{c}_i^1, \mathbf{c}_j^0, \mathbf{c}_j^1)$ where all the four commitments are to the bit b. Denote by $\mathbf{\Pi}(b) = (\pi^{00}, \pi^{01}, \pi^{10}, \pi^{11})$ where for all $b_1, b_2 \in \{0,1\}$, $\pi^{b_1 b_2}$ is a proof for $(\mathbf{c}_i^{b_1}, \mathbf{c}_j^{b_2}, \mathsf{pp}_i^{b_1}, \mathsf{pp}_j^{b_2}) \in L_{\mathsf{TC}}$.

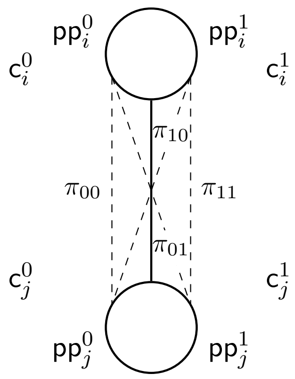

Figure 3: Zooming in on wire k of circuit C with parameters $\mathsf{PP} = (\mathsf{pp}_i^0, \mathsf{pp}_i^1, \mathsf{pp}_j^0, \mathsf{pp}_j^1)$, commitments $\mathsf{W} = (\mathbf{c}_i^0, \mathbf{c}_i^1, \mathbf{c}_j^0, \mathbf{c}_j^1)$ and L_{TC} proofs $\mathbf{\Pi} = (\pi^{00}, \pi^{01}, \pi^{10}, \pi^{11})$.

To prove WI of the final construction, in particular the following should hold:

$$\big(\mathsf{PP}, \mathsf{W}(0), \mathbf{\Pi}(0)\big) \approx \big(\mathsf{PP}, \mathsf{W}(1), \mathbf{\Pi}(1)\big) \tag{1}$$

[2] TC stands for the language of Two Commitments.

This indistinguishability requirement already implies a strong NIWI for L_{TC}, with respect to distributions $\mathcal{D}_0$ and $\mathcal{D}_1$, where $\mathcal{D}_b$ samples L_{TC} instances $(\mathbf{c}_i, \mathbf{c}_j, \mathsf{pp}_i, \mathsf{pp}_j)$ such that $\mathbf{c}_i, \mathbf{c}_j$ commit to the bit b.

For our analysis, Equation (1) is insufficient since we need Equation (1) to hold even given the rest of the proof for $(C, \mathsf{out}) \in L_{\mathcal{U}}$. In other words, we need Equation (1) to hold given some auxiliary information aux, where given aux it should be possible to efficiently compute the rest of the proof from it. One possible aux is the openings of all the four commitments so that it is then possible to compute Bit and Gate Proofs for the rest of the proof. But if we give the openings with respect to 0 and 1 respectively, then the two distributions in Equation (1) are clearly distinguishable.

So the question is, what aux can we give? Our key insight is that we can give equivocated openings for the commitments with respect to the two hiding parameters and honest openings with respect to the binding parameters, so that in both the distributions in Equation (1), two of the openings are to 0 and two of them are to 1. Without loss of generality, we think of $\mathsf{pp}_i^0, \mathsf{pp}_j^0$ as the binding parameters and $\mathsf{pp}_i^1, \mathsf{pp}_j^1$ as the hiding parameters. We strengthen the requirement in Equation (1) as follows:

$$\big(\mathsf{PP}(0), \mathsf{W}(0), \mathbf{\Pi}(0), \mathsf{O}(0)\big) \approx \big(\mathsf{PP}(1), \mathsf{W}(1), \mathbf{\Pi}(1), \mathsf{O}(1)\big) \tag{2}$$

where $\mathsf{PP}(b) = (\mathsf{pp}_i^b, \mathsf{pp}_i^{1-b}, \mathsf{pp}_j^b, \mathsf{pp}_j^{1-b})$, and $\mathsf{W}(b), \mathbf{\Pi}(b)$ are as before, and where in both the distributions, $\mathsf{O}(b)$ contains openings for the commitments $\mathsf{W}(b)$ to $(0,1,0,1)$ respectively. This is the case since in the left-hand-side parameters $\mathsf{PP}(0)$, the second and fourth parameters are hiding, and we equivocate $\mathbf{c}_i^1, \mathbf{c}_j^1$ to open to 1, whereas in the right-hand-side parameters $\mathsf{PP}(1)$, the first and third parameters are hiding, and we equivocate $\mathbf{c}_i^1, \mathbf{c}_j^1$ to open to 0. Note that the L_{TC} proofs in $\mathbf{\Pi}(b)$ are still computed using the (honest) openings to b.

This is still not sufficient for our WI analysis. In order to argue WI of the final construction, we need to invoke Equation (2) for every wire k in the circuit for which the value of wit_0 on wire k is different from value of wit_1 on wire k. These invocations are not completely independent since two different wires may be associated with the same gate, and in particular the two wires may be associated with an overlapping set of parameters. Thus, we need to further strengthen Equation (2) to as follows:

$$\big(\mathsf{PP}(0), \mathsf{W}(0), \mathbf{\Pi}(0), \mathsf{O}(0), \mathsf{W}(1), \mathbf{\Pi}(1), \mathsf{O}(1)\big) \approx \big(\mathsf{PP}(1), \mathsf{W}(1), \mathbf{\Pi}(1), \mathsf{O}(1), \mathsf{W}(0), \mathbf{\Pi}(0), \mathsf{O}(0)\big) \tag{3}$$

where $\mathsf{PP}(b), \mathsf{W}(b), \mathbf{\Pi}(b)$ and $\mathsf{O}(b)$ are as described above. We note that in the left-hand-side, $\mathsf{W}(1)$ are four commitments to 1 with respect to $\mathsf{PP}(0)$, $\mathbf{\Pi}(1)$ are the corresponding L_{TC} proofs computed using the honest openings to 1, and $\mathsf{O}(1)$ are the openings to $(1,0,1,0)$ respectively. Similarly, in the right-hand-side, $\mathsf{W}(0)$ are four commitments to 0 with respect to $\mathsf{PP}(1)$, $\mathbf{\Pi}(0)$ are the corresponding L_{TC} proofs, and again $\mathsf{O}(0)$ are the openings to $(1,0,1,0)$ respectively. We refer to the property from Equation (3) as *Strong Secrecy* of L_{TC} and describe it in detail in Section 7.1. The Strong Secrecy requirement of L_{TC} as in Equation (3) is sufficient for our WI analysis. Before explaining our WI analysis, we describe the ingredients for our FH NIWI Construction.

Recall that our NIWI proof for $(C, \mathsf{out}) \in L_{\mathcal{U}}$ is computed as follows: Choose a fresh pair of parameters per gate, commit to all the wire values with respect to all the associated parameters (2 commitments per input wire, 4 commitments per connecting wire), compute Bit Proofs (one per commitment), compute Gate Proofs (two per gate) and compute L_{TC} proofs (four per connecting wire). In order to randomize our NIWI proof, we randomize all the parameters, correspondingly update the commitments and update the proofs to be with respect to the randomized parameters and commitments. Specifically, we need the following ingredients for our final FH NIWI Construction.

Ingredients for our FH NIWI.

– A Commitment Scheme as required in the FH NIZK construction, but with the additional feature that allows for randomizing the parameters and updating the commitments to be with respect to the randomized parameters, so that the randomized parameters and commitments are distributed like fresh commitments.

– Bit Proofs and Gate Proofs as required in the FH NIZK construction, but with the following (modified) malleability property: Given a proof for commitments with respect to some pp, it is possible to efficiently randomize the parameters, correspondingly update the commitments and update the proofs to be with respect to the new parameters and commitments, such that they are all distributed like fresh ones. As in the FH NIZK, we require the Bit and Gate Proofs to satisfy WI.

– A proof system for L_{TC} with the same malleability property as Bit and Gate Proofs, and with the Strong Secrecy property as described in Equation (3).

We show (in Section 5.1) that the GOS commitment scheme ($\mathsf{C.Setup}, \mathsf{C.Commit}, \mathsf{C.Rand}$) satisfies the additional feature that we require. The malleability of Bit Proofs and Gate Proofs can be reduced to the malleability of the NP language L_{Lin} described previously (similar to the FH NIZK construction). We describe the corresponding proof systems ($\mathsf{Bit.Prove}, \mathsf{Bit.Verify}, \mathsf{Bit.GenMaul}$) and ($\mathsf{N.Prove}, \mathsf{N.Verify}, \mathsf{N.GenMaul}$) in Section 7.1.

Jumping ahead, we construct the proof system for L_{TC} also using the proof system for L_{Lin}, and the malleability of L_{TC} follows from the malleability of L_{Lin}. We then argue that the Strong Secrecy follows from our new *DLIN with Leakage* assumption (see Section 2.2.2 for an overview and Section 7.3.2 for the details).

WI Analysis. To explain our WI analysis, we describe an algorithm $\mathsf{ProofGen}$ that on input a sample from the left-hand-side distribution in Equation (3), generates an entire proof Π for $(C, \mathsf{out}) \in L_{\mathcal{U}}$ which is indistinguishable from an honest proof generated using wit_0, and on input a sample from the right-hand-side distribution, $\mathsf{ProofGen}$ generates a proof Π which is indistinguishable from an honest proof generated using wit_1.

$\mathsf{ProofGen}$ *Algorithm.* Without loss of generality, we assume that every circuit is layered; that is, all the gates of the circuit can be arranged in t layers so that for all $i \in [t]$, all the output wires of gates from layer i are input wires to gates in layer $i+1$. Fix any two witnesses wit_0 and wit_1 for $(C, \mathsf{out}) \in L_{\mathcal{U}}$.

On input $\big(\mathsf{PP}(b), \mathsf{W}(b), \mathbf{\Pi}(b), \mathsf{O}(b), \mathsf{W}(1-b), \mathbf{\Pi}(1-b), \mathsf{O}(1-b)\big)$, $\mathsf{ProofGen}$ does the following:

1. Recall that $\mathsf{PP}(b) = (\mathsf{pp}_i^b, \mathsf{pp}_i^{1-b}, \mathsf{pp}_j^b, \mathsf{pp}_j^{1-b})$. Assign parameters $(\mathsf{pp}_i^b, \mathsf{pp}_i^{1-b})$ to all the odd layer gates of the circuit and $(\mathsf{pp}_j^b, \mathsf{pp}_j^{1-b})$ to all the even layer gates of the circuit. We will refer to $\{\mathsf{pp}_i^b, \mathsf{pp}_j^b\}$ as the *Left Parameters* and $\{\mathsf{pp}_i^{1-b}, \mathsf{pp}_j^{1-b}\}$ as the *Right Parameters.*

2. For all the input wires of the circuit C, commit to wit_0 with respect to pp_i^b (Left Parameter) and commit to wit_1 with respect to pp_i^{1-b} (Right Parameter).

3. For every wire k, produce the 4 commitments and 4 L_{TC} proofs for the wire as follows: Denote by $w_{k,0}$ the value induced by wit_0 on wire k, and denote by $w_{k,1}$ the value induced by wit_1 on wire k in the circuit.

   – If $w_{k,0} = w_{k,1}$ then compute the commitments and L_{TC} proofs honestly.
   – If $w_{k,0} = 0$ and $w_{k,1} = 1$ then use $\mathsf{W}(b)$ as the commitments and $\mathbf{\Pi}(b)$ as the L_{TC} proofs.
   – If $w_{k,0} = 1$ and $w_{k,1} = 0$ then use $\mathsf{W}(1-b)$ as the commitments and $\mathbf{\Pi}(1-b)$ as the L_{TC} proofs.

4. Compute the Bit Proofs and Gate Proofs honestly: We have the openings for all the commitments to the input bits (from Step 2). We also have the openings for the commitments to every non-input wire k, namely $\mathsf{O}(b)$ for $\mathsf{W}(b)$ when $w_{k,0} = 0$ and $w_{k,1} = 1$, or $\mathsf{O}(1-b)$ for $\mathsf{W}(1-b)$ when $w_{k,0} = 1$ and

$w_{k,1} = 0$, or since we generated the commitments honestly when $w_{k,0} = w_{k,1}$. Note that the openings with respect to the Left Parameters always correspond to wit_0 and the openings with respect to the Right Parameters always correspond to wit_1.

– Bit Proofs can be computed honestly since all the openings are to 0 or 1.

– Gate Proofs can be computed honestly since all the openings with respect to the Left Parameters are consistent with wit_0 and all the openings with respect to the Right Parameters are consistent with wit_1.

5. Randomize the entire proof as follows:

   – For every gate, randomize the pair of parameters for that gate.

   – Update all the commitments (2 commitments per input wire, 4 commitments per connecting wire) to be with respect to the randomized parameters.

   – Maul all the Bit Proofs (one per commitment), all the Gate Proofs (two per gate) and all the L_{TC} proofs (four for every connecting wire) to be with respect to the updated parameters and commitments.

   Finally output this randomized proof.

So far, we described the $\mathsf{ProofGen}$ algorithm that given a sample from the distributions in Equation (3), generates an entire proof for $(C, \mathsf{out}) \in L_{\mathcal{U}}$. Let Π^0_{Gen} be a proof output by $\mathsf{ProofGen}$ on input a sample from the left-hand-side of Equation (3) and let Π^1_{Gen} be a proof output by $\mathsf{ProofGen}$ on input a sample from the right-hand-side of Equation (3).

From Equation (3), it follows that $\Pi^0_{\mathsf{Gen}} \approx \Pi^1_{\mathsf{Gen}}$. All that remains is to argue that $\Pi_0 \approx \Pi^0_{\mathsf{Gen}}$ and $\Pi_1 \approx \Pi^1_{\mathsf{Gen}}$, where Π_b is an honestly computed proof for $(C, \mathsf{out}) \in L_{\mathcal{U}}$ using witness wit_b. Note that Π_0 and Π^0_{Gen} are identical except that Π^0_{Gen} uses equivocated openings to wit_1 on the Right Parameters to compute the Bit and Gate Proofs. Hence, $\Pi_0 \approx \Pi^0_{\mathsf{Gen}}$ follows from WI of the Bit and Gate Proofs, and in addition follows by the randomizability of the commitment scheme and the malleability of the underlying proofs. By a similar argument, $\Pi_1 \approx \Pi^1_{\mathsf{Gen}}$. Thus, WI of the final construction follows form the Strong Secrecy of L_{TC}.

2.2.2 Constructing the L_{TC} Proof System

We construct a proof system for L_{TC} with the following properties:

1. Strong Secrecy: As defined in Equation (3).

2. Malleability: Given a proof π for $(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2) \in L_{\mathsf{TC}}$, one can efficiently randomize the parameters to obtain $\mathsf{pp}'_1, \mathsf{pp}'_2$, update the commitments to obtain $\mathbf{c}'_1, \mathbf{c}'_2$ which are with respect to $\mathsf{pp}'_1, \mathsf{pp}'_2$, and then maul π to a proof π' for $(\mathbf{c}'_1, \mathbf{c}'_2, \mathsf{pp}'_1, \mathsf{pp}'_2) \in L_{\mathsf{TC}}$ such that $(\mathbf{c}'_1, \mathbf{c}'_2, \mathsf{pp}'_1, \mathsf{pp}'_2)$ looks like a fresh instance and π' is distributed like a fresh proof.

3. Soundness: We require that soundness holds for all instances $(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2)$ where both $\mathsf{pp}_1, \mathsf{pp}_2$ are binding. As noted above, this is sufficient for the soundness of the final construction.

We construct such a proof system using the malleable NIWI proof system for L_{Lin} described before. Recall that L_{Lin} is a parameterized language with parameters $\mathsf{pp} = (f, h, g)$ where f, h, g are generators of a group $\mathbb{G}$, and it consists of a pair of tuples $(\mathbf{A}, \mathbf{B})$ such that one of them is of the form $(f^{a_1}, h^{a_2}, g^{a_3})$ where $a_3 = a_1 + a_2$.

We reduce proving that $(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2) \in L_{\mathsf{TC}}$ to proving that $(\mathbf{A}, \mathbf{B}) \in L_{\mathsf{Lin}}$ for some $(\mathbf{A}, \mathbf{B})$. However, we only know how to do this reduction for L_{TC} instances $(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2)$ for which $\mathsf{pp}_1 = \mathsf{pp}_2$.

Therefore, we consider an NP-relation for L_{TC} with an additional witness which lets us convert an instance $(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2)$ into an instance $(\mathbf{c}_*, \mathbf{c}_2, \mathsf{pp}_2, \mathsf{pp}_2)$. The additional witness for $(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2)$ is a hard-to-compute function of the parameters $\mathsf{pp}_1, \mathsf{pp}_2$, and we refer to it as an "intermediate parameter" pp_* of $\mathsf{pp}_1, \mathsf{pp}_2$. Using the intermediate parameter pp_* we can convert the commitment $\mathbf{c}_1$ with respect to pp_1 into a commitment $\mathbf{c}_*$ with respect to pp_2.

More specifically in our proof, pp_* helps in converting the commitment $\mathbf{c}_1$ with respect to parameters pp_1, into a commitment $\mathbf{c}_*$ (to the same value) with respect to pp_2. Then, we can reduce the instance $(\mathbf{c}_*, \mathbf{c}_2, \mathsf{pp}_2, \mathsf{pp}_2) \in L_{\mathsf{TC}}$ to a pair of tuples $(\mathbf{A}, \mathbf{B}) \in L_{\mathsf{Lin}}$. The soundness and malleability of the L_{TC} proof system follows from the corresponding properties of L_{Lin} proof system. We refer to Section 7.3.1 for a detailed description of the construction.

Strong Secrecy from DLIN with Leakage. All that remains is to show that the strong secrecy of L_{TC} follows from our new assumption of DLIN with Leakage. We first prove that Strong Secrecy of L_{TC} follows from the fact that the NIWI for L_{Lin} is strong WI with respect to the following distributions $\mathcal{D}_0$ and $\mathcal{D}_1$.

– $\mathcal{D}_0$ generates $(\mathbf{A}, \mathbf{B})$ where $\mathbf{A} = (f^{a_1}, h^{a_2}, g^{a_3})$ for random a_1, a_2, a_3 such that $a_1 + a_2 = a_3$, and $\mathbf{B} = (f^{a_1}, h^{a_2}, g^{a_3+1})$.

– $\mathcal{D}_1$ generates $(\mathbf{A}, \mathbf{B})$ where $\mathbf{A} = (f^{a_1}, h^{a_2}, g^{a_3-1})$ for random a_1, a_2, a_3 such that $a_1 + a_2 = a_3$, and $\mathbf{B} = (f^{a_1}, h^{a_2}, g^{a_3})$.

We then prove that the proof system for L_{Lin} is strong WI with respect to $\mathcal{D}_0$ and $\mathcal{D}_1$ under DLIN with Leakage assumption. We refer to Section 7.3.2 for a detailed description of the reduction.

3 Preliminaries

We denote the security parameter by λ. We use PPT to denote that an algorithm is probabilistic polynomial time. We denote by $y \leftarrow A(x)$ if y is the output of a single execution of A on input x. We denote by $y = A(x; r)$ to explicitly mention the randomness used in the execution. We denote $y \in A(x)$ if there exists randomness r such that $y = A(x; r)$.

We use $[n]$ to represent the set $\{1, \ldots, n\}$. Vectors are denoted by $\mathbf{a}$ where $\mathbf{a} = (a_1, \ldots, a_n)$ and a_i is the i th element of $\mathbf{a}$. $|\mathbf{a}|$ denotes the size of $\mathbf{a}$. $\mathbf{a} \circ \mathbf{b}$ denotes concatenation of the vectors $\mathbf{a}, \mathbf{b}$. $\{\mathcal{X}\}_{\lambda \in \mathbb{N}} \approx_c \{\mathcal{Y}\}_{\lambda \in \mathbb{N}}$ will denote that distributions $\{\mathcal{X}\}_{\lambda \in \mathbb{N}}$ and $\{Y\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable.

3.1 Definition of Proof Systems

Definition 1 (Non-interactive Zero-knowledge Proofs [BDMP91])**.** Let $L \in \mathsf{NP}$ and let R_L be the corresponding NP relation. A triplet of PPT algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ is called a *non interactive zero knowledge* (NIZK) proof system for L if it satisfies:

– **Perfect Completeness:** For all security parameters $\lambda \in \mathbb{N}$ and for all $(x, w) \in R_L$,

$$\Pr[\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)\ ;\ \pi \leftarrow \mathsf{Prove}(\mathsf{CRS}, x, w) : \mathsf{Verify}(\mathsf{CRS}, x, \pi) = 1] = 1$$

– **Adaptive Soundness:** For any all-powerful prover P^*, there exists a negligible function μ such that for all λ,

$$\Pr[\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)\ ;\ (x, \pi) = P^*(\mathsf{CRS})\ :\ \mathsf{Verify}(\mathsf{CRS}, x, \pi) = 1 \ \wedge\ x \notin L] \leq \mu(\lambda)$$

When this probability is 0, we say it is *perfectly* sound.

- **Adaptive Zero Knowledge:** There exists a PPT simulator $S = (S_1, S_2)$ where $S_1(1^\lambda)$ outputs (CRS_S, τ) and $S_2(\mathsf{CRS}_S, \tau, x)$ outputs π_s such that for all non-uniform PPT adversaries $\mathcal{A}$,

$$\{\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda) \ : \ \mathcal{A}^{\mathcal{O}_1(\mathsf{CRS},\cdot,\cdot)}(\mathsf{CRS})\} \approx_c$$
$$\{(\mathsf{CRS}_S, \tau) \leftarrow S_1(1^\lambda) \ : \ \mathcal{A}^{\mathcal{O}_2(\mathsf{CRS}_S,\tau,\cdot,\cdot)}(\mathsf{CRS}_S)\}$$

where $\mathcal{O}_1, \mathcal{O}_2$ on input (x, w) first check that $(x, w) \in R_L$, else output $\perp$. Otherwise $\mathcal{O}_1$ outputs $\mathsf{Prove}(\mathsf{CRS}, x, w)$ and $\mathcal{O}_2$ outputs $S_2(\mathsf{CRS}_S, \tau, x)$.

Definition 2 (Non interactive Witness Indistinguishable Proofs [BOV05, DN00])**.** A pair of PPT algorithms $(\mathsf{Prove}, \mathsf{Verify})$ is called a *non interactive witness indistinguishable* (NIWI) proof for an NP language L with NP relation R_L if it satisfies:

- **Completeness:** For all security parameters λ and for all $(x, w) \in R_L$,

$$\Pr[\pi \leftarrow \mathsf{Prove}(1^\lambda, x, w) \ : \ \mathsf{Verify}(1^\lambda, x, \pi) = 1] = 1$$

- **Soundness:** For any all-powerful prover P^*, if $P^*(1^\lambda) = (x, \pi)$ and $x \notin L$, then $\mathsf{Verify}(1^\lambda, x, \pi) = 0$.

- **Witness Indistinguishability:** For all non-uniform PPT adversaries $\mathcal{A}$, there exists a negligible function ν such that for every $\lambda \in \mathbb{N}$, probability that $b' = b$ in the following game is at most $1/2 + \nu(\lambda)$:

  1. $(\mathsf{state}, x, w_0, w_1) \leftarrow \mathcal{A}(1^\lambda)$.
  2. Choose $b \xleftarrow{\$} \{0, 1\}$. If $R_L(x, w_0) \neq 1$ or $R_L(x, w_1) \neq 1$ then output $\perp$. Else, if $b = 0$ then $\pi \leftarrow \mathsf{Prove}(1^\lambda, x, w_0)$, and if $b = 1$ then $\pi \leftarrow \mathsf{Prove}(1^\lambda, x, w_1)$.
  3. $b' \leftarrow \mathcal{A}(\mathsf{state}, \pi)$.

We say that a pair of PPT algorithms $(\mathsf{Prove}, \mathsf{Verify})$ is called a *non interactive proof system* for an NP language L if it satisfies completeness and adaptive soundness.

For our purposes, we will be using NIWI proofs with respect to parameterized languages of the form $L[\mathsf{pp}]$ where pp denotes some global parameters.

Definition 3 (Non interactive Witness Indistinguishability proofs for Parameterized Languages)**.** Let Setup be a PPT algorithm that takes as input the security parameter and outputs a set of parameters pp. A pair of PPT algorithms $(\mathsf{Prove}, \mathsf{Verify})$ is called a NIWI proof for a parameterized NP language $L[\mathsf{pp}]$, with NP relation $R_L[\mathsf{pp}]$ if it satisfies:

- **Completeness:** For all security parameters λ, for all $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$ and for all $(x, w) \in R_L[\mathsf{pp}]$, $\Pr[\pi \leftarrow \mathsf{Prove}(\mathsf{pp}, x, w) \ : \ \mathsf{Verify}(\mathsf{pp}, x, \pi) = 1] = 1$.

- **Adaptive Soundness:** For any all-powerful prover P^*, there exists a negligible function μ such that for all λ,

$$\Pr[\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \ : \ (x, \pi) \leftarrow P^*(\mathsf{pp}) \ : \ \mathsf{Verify}(\mathsf{pp}, x, \pi) = 1 \ \wedge \ x \notin L] \leq \mu(\lambda)$$

- **Witness Indistinguishability:** For all non-uniform PPT adversaries $\mathcal{A}$, there exists a negligible function ν such that for every $\lambda \in \mathbb{N}$, probability that $b' = b$ in the following game is at most $1/2 + \nu(\lambda)$:

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $(\mathsf{state}, x, w_0, w_1) \leftarrow \mathcal{A}(\mathsf{pp})$.
3. Choose $b \xleftarrow{\$} \{0,1\}$. If $R_L[\mathsf{pp}](x, w_0) \neq 1$ or $R_L[\mathsf{pp}](x, w_1) \neq 1$ then output $\perp$. Else if $b = 0$ then $\pi \leftarrow \mathsf{Prove}(\mathsf{pp}, x, w_0)$, else if $b = 1$ then $\pi \leftarrow \mathsf{Prove}(\mathsf{pp}, x, w_1)$. Send π to $\mathcal{A}$.
4. $b' \leftarrow \mathcal{A}(\mathsf{state}, \pi)$.

Definition 4 (Randomizable NIZK and NIWI Proofs [BCC$^+$09b])**.** A NIZK proof system for an NP language L with NP relation R_L with algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ is said to be a randomizable proof system if there exists a PPT algorithm Rand which on input a CRS, an instance x and a proof π, outputs a "randomized" proof π' for x such that for all non-uniform PPT adversaries $\mathcal{A}$, there exists a negligible function ν such that for every $\lambda \in \mathbb{N}$, the probability that $b' = b$ in the following game is at most $1/2 + \nu(\lambda)$:

1. $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $(\mathsf{state}, x, w, \pi) \leftarrow \mathcal{A}(\mathsf{CRS})$.
3. Choose $b \xleftarrow{\$} \{0,1\}$. If $\mathsf{Verify}(\mathsf{CRS}, x, \pi) \neq 1$ or $R_L(x, w) \neq 1$ then output $\perp$.
4. Else if $b = 0$ then $\pi' \leftarrow \mathsf{Prove}(\mathsf{CRS}, x, w)$, else if $b = 1$ then $\pi' \leftarrow \mathsf{Rand}(\mathsf{CRS}, x, \pi)$.
5. $b' \leftarrow \mathcal{A}(\mathsf{state}, \pi')$.

More generally, a (WI) proof system $(\mathsf{Prove}, \mathsf{Verify})$ is said to be randomizable if there exists a PPT algorithm Rand with the same description and properties as above and where $\mathsf{CRS} = 1^\lambda$.

Definition 5 (Malleable NIWI Proofs for Parameterized Languages [?])**.** Let $(\mathsf{Prove}, \mathsf{Verify})$ be a NIWI proof system for a parameterized NP language $L[\mathsf{pp}]$ with NP relation $R_L[\mathsf{pp}]$ where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ (as per Definition 3). Let $T = (T_{\mathsf{inst}}, T_{\mathsf{wit}})$ be a pair PPT transformations such that for every $(x, w) \in R_L$ and for every randomness $\sigma \in \{0,1\}^{\mathsf{poly}(\lambda)}$, $\big(T_{\mathsf{inst}}(\mathsf{pp}, x; \sigma), T_{\mathsf{wit}}(\mathsf{pp}, x, w, \sigma)\big) \in R_L$.

Such a proof system is said to be *malleable* with respect to T, if there exists a randomized PPT algorithm Maul which on input parameters pp, an instance x, randomness σ and proof π, outputs a "mauled" proof π' for $T(\mathsf{pp}, x; \sigma)$ such that the following properties hold:

Malleability For all non-uniform PPT $\mathcal{A}$, for all $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$, for all $\lambda \in \mathbb{N}$,

$$\Pr\Big[(x,\pi) \leftarrow \mathcal{A}(\mathsf{pp}) \ ; \ (\sigma, R) \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)} \ ; \ \pi' = \mathsf{Maul}(\mathsf{pp}, x, \sigma, \pi; R) \ : \\ \big(\mathsf{Verify}(\mathsf{pp}, x, \pi) = 0\big) \ \vee \ \big(\mathsf{Verify}(\mathsf{pp}, T(\mathsf{pp}, x; \sigma), \pi') = 1\big)\Big] = 1$$

Perfect Randomizability There exists a poly-time function f_T such that for all $\mathsf{pp} \in \mathsf{Setup}(1^\lambda)$ and every $(x, w) \in R_L[\mathsf{pp}]$, for every $R, \sigma \in \{0,1\}^{\mathsf{poly}(\lambda)}$,

$$\mathsf{Maul}(\mathsf{pp}, x, \sigma, \mathsf{Prove}(\mathsf{pp}, x, w; R); R') = \mathsf{Prove}(\mathsf{pp}, T_{\mathsf{inst}}(\mathsf{pp}, x; \sigma), T_{\mathsf{wit}}(\mathsf{pp}, x, w, \sigma); S)$$

where $S = f_T(\mathsf{pp}, w, R, R', \sigma)$. Moreover, if R', σ are uniform, then $f_T(w, R, R', \sigma)$ is uniformly distributed.

Definition 6 (Strong Non-interactive Witness Indistinguishability [Gol00])**.** Let Setup be a PPT algorithm that takes as input the security parameter and outputs a set of parameters pp. Let $\mathcal{D}_0 = \{\mathcal{D}_{0,\lambda}\}_{\lambda \in \mathbb{N}}, \mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ be distribution ensembles in the support of $R_L[\mathsf{pp}] \cap \{0,1\}^\lambda$ such that for every $b \in \{0,1\}$, $(x_b, w_b) \leftarrow \mathcal{D}_b$ such that $(x_b, w_b) \in R_L[\mathsf{pp}]$.

A NIWI proof system $(\mathsf{Prove}, \mathsf{Verify})$ for a parameterized NP language $L[\mathsf{pp}]$ is a *strong* non interactive witness indistinguishable (Strong NIWI) proof with respect to distributions $\mathcal{D}_0, \mathcal{D}_1$, if the following holds:

$$\text{If } \{\mathsf{pp}, x_0\} \approx \{\mathsf{pp}, x_1\} \text{ then } E_0 \approx E_1$$

where $E_b(1^\lambda)$ does the following: Sample $(x_b, w_b) \leftarrow \mathcal{D}_b(\mathsf{pp})$ and compute $\pi_b \leftarrow \mathsf{Prove}(\mathsf{pp}, x_b, w_b)$. Output $(\mathsf{pp}, x_b, \pi_b)$.

3.2 Bilinear Maps

We will be working with abelian groups $\mathbb{G}, \mathbb{G}_T$ of prime order p equipped with a symmetric bilinear map $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_T$. We let $\mathcal{G}$ be a *deterministic* polynomial time algorithm that takes as input the security parameter 1^λ and outputs $(p, \mathbb{G}, \mathbb{G}_T, e, g_p)$ such that p is a prime, $\mathbb{G}, \mathbb{G}_T$ are descriptions of groups of order p, g_p is a fixed generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_T$ is a bilinear map with the following properties:

– (Non-degenerate) For any generator g of $\mathbb{G}$, $g_T = e(g, g)$ has order p in $\mathbb{G}_T$

– (Bilinear) For all $a, b \in \mathbb{G}$, for all $x, y \in \mathbb{Z}_p$, $e(a^x, b^y) = e(a, b)^{xy}$

We require that the group operations and the bilinear operations are computable in polynomial time with respect to security parameter.

Assumption 1 (Decisional Linear Assumption)**.** We say that the Decisional Linear (DLIN) Assumption holds for a bilinear group generator $\mathcal{G}$ if the following distributions are computationally indistinguishable:

$$\{(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}(1^\lambda)\ ;\ (x, y) \overset{\$}{\leftarrow} \mathbb{Z}_p^*\ :\ (r, s) \overset{\$}{\leftarrow} \mathbb{Z}_p\ :\ (p, \mathbb{G}, \mathbb{G}_T, e, g, g^x, g^y, g^{xr}, g^{ys}, g^{r+s})\} \text{ and}$$

$$\{(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathcal{G}(1^\lambda)\ ;\ (x, y) \overset{\$}{\leftarrow} \mathbb{Z}_p^*\ :\ (r, s, d) \overset{\$}{\leftarrow} \mathbb{Z}_p\ :\ (p, \mathbb{G}, \mathbb{G}_T, e, g, g^x, g^y, g^{xr}, g^{ys}, g^d)\}$$

4 Fully Homomorphic Proofs: Definition

In this section we define fully homomorphic NIZK and NIWI proofs for the NP-complete language $L_\mathcal{U}$ consisting of instances of the form (C, b) where $C : \{0,1\}^k \to \{0,1\}$ is a boolean circuit and $b \in \{0,1\}$. Formally, $L_\mathcal{U}$ is defined as:

$$L_\mathcal{U} = \{(C, b) \mid \exists\ \mathbf{w} \text{ such that } C(\mathbf{w}) = b\}$$

Let $R_\mathcal{U}$ be the corresponding NP-relation. We first define the notion of composing multiple instances of $L_\mathcal{U}$ to get a new instance in $L_\mathcal{U}$:

Composing $L_\mathcal{U}$ Instances: On input k instances $\{(C_i, b_i)\}_{i=1}^k$ where $C_i : \{0,1\}^{t_i} \to \{0,1\}$ and $C' : \{0,1\}^k \to \{0,1\}$,

$$\mathsf{Compose}(\{(C_i, b_i)\}_{i=1}^k, C') = (C, b)$$

where $C : \{0,1\}^T \to \{0,1\}$ and $T = \sum_{i=1}^k t_i$ and for all $(\mathbf{w_1}, \ldots, \mathbf{w_k}) \in \{0,1\}^{t_1} \times \cdots \times \{0,1\}^{t_k}$,

$$C(\mathbf{w_1}, \ldots, \mathbf{w_k}) = C'\big(C_1(\mathbf{w_1}), \ldots, C_k(\mathbf{w_k})\big)\ \wedge\ b = C'(b_1, \ldots, b_k).$$

4.1 Definition: Fully Homomorphic NIZK and NIWI Proofs

We now define fully homomorphic NIZK and NIWI proofs for the language $L_\mathcal{U}$ defined above.

Definition 7 (Fully Homomorphic NIZK Proofs)**.** A randomizable NIZK proof system ($\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Rand}$) is a fully homomorphic proof system if there exists a PPT algorithm Eval with the following input-output behavior:

$((C, b), \Pi) \leftarrow \mathsf{Eval}(\mathsf{CRS}, \{(C_i, b_i), \Pi_i\}_{i=1}^k, C')$: The Eval algorithm takes as input the CRS, k instances $\{(C_i, b_i)\}_{i=1}^k$ along with their proofs $\{\Pi_i\}_{i=1}^k$, and a circuit $C' : \{0,1\}^k \to \{0,1\}$. It outputs the composed instance $(C, b) = \mathsf{Compose}(\{(C_i, b_i)\}_{i=1}^k, C')$ and a corresponding proof Π such that the following properties hold:

Completeness of Eval: We require that evaluating on valid proofs (proofs that verify), should result in a proof that verifies. More concretely, we require that for all non-uniform PPT $\mathcal{A}$ and for all $\lambda \in \mathbb{N}$,

$$\Pr\left[\begin{array}{c} \mathsf{CRS}\leftarrow\mathsf{Setup}(1^\lambda)\ ;\ (\{(C_i,b_i,\Pi_i)\}_{i=1}^k,C')\leftarrow\mathcal{A}(\mathsf{CRS})\ ; \\ ((C,b),\Pi)\leftarrow\mathsf{Eval}(\mathsf{CRS},\{(C_i,b_i),\Pi_i\}_{i=1}^k,C')\ : \\ \Big(\mathsf{Valid}(C')=0\Big)\ \vee\ \Big(\exists\ i\in[k]\ \text{s.t.}\mathsf{Verify}(\mathsf{CRS},(C_i,b_i),\Pi_i)=0\Big)\ \vee \\ \Big((\mathsf{Verify}(\mathsf{CRS},(C,b),\Pi)=1)\ \wedge\ (C,b)=\mathsf{Compose}(\{(C_i,b_i)\}_{i=1}^k,C')\Big) \end{array}\right] = 1$$

where $\mathsf{Valid}(C') = 1$ if and only if $C' : \{0,1\}^k \to \{0,1\}$.

Unlinkability: We require that a proof for $(C,b) \in L_{\mathcal{U}}$ obtained by Eval should be indistinguishable from a fresh proof for the same instance. Namely, for any non-uniform PPT adversary $\mathcal{A}$, there exists a negligible function ν such that for every λ the probability that $\mathsf{bit} = \mathsf{bit}'$ in the following game is at most $1/2 + \nu(\lambda)$:

<u>GAME$_{\mathsf{Eval}}$:</u>

1. $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $(\mathsf{state}, \{((C_i,b_i),\mathbf{w}_i,\Pi_i)\}_{i=1}^k, C') \leftarrow \mathcal{A}(\mathsf{CRS})$
3. Choose $\mathsf{bit} \overset{\$}{\leftarrow} \{0,1\}$. If for any $i \in [k]$, $\mathsf{Verify}(\mathsf{CRS},(C_i,b_i),\Pi_i) \neq 1$ or $((C_i,b_i),\mathbf{w}_i) \notin R_{\mathcal{U}}$, output $\perp$.
4. Else if $\mathsf{bit} = 0$ then $((C,b),\Pi) \leftarrow \mathsf{Eval}(\mathsf{CRS},\{(C_i,b_i),\Pi_i\}_{i=1}^k, C')$. Else if $\mathsf{bit} = 1$ then compute $(C,b) = \mathsf{Compose}(\{(C_i,b_i)\}_{i=1}^k, C')$ and
   $\Pi \leftarrow \mathsf{Prove}(\mathsf{CRS},(C,b),\mathbf{w})$ where $\mathbf{w} = \mathbf{w}_1 \circ \cdots \circ \mathbf{w}_k$. Send (C,b,Π) to $\mathcal{A}$.
5. $\mathsf{bit}' \leftarrow \mathcal{A}(\mathsf{state},(C,b,\Pi))$.

Definition 8 (Fully Homomorphic NIWI Proofs)**.** A randomizable NIWI proof system $(\mathsf{Prove}, \mathsf{Verify}, \mathsf{Rand})$ is a fully homomorphic NIWI proof system if there exists a PPT algorithm Eval with the same description and properties as in Definition 7 and where $\mathsf{CRS} = 1^\lambda$.

5 Building Blocks for Fully Homomorphic Proofs

In this section we describe the building blocks for our fully homomorphic (FH) NIZK and NIWI constructions. In Section 5.1, we define a commitment scheme with additional properties, which we will use in our FH NIZK and NIWI constructions, and we then instantiate it from DLIN.

In Section 5.2, we describe a NIWI proof system for the NP language L_{Lin} (defined in Definition 12) based on DLIN. This proof system is the main ingredient in constructing FH NIZK and FH NIWI proofs.

For our FH NIWI construction, we need the NIWI proof for L_{Lin} to have additional properties of malleability and strong WI with respect to specific distributions. We prove that the proof system is malleable and we prove that strong WI holds under a new assumption on bilinear groups: *DLIN with Leakage*. We describe the corresponding bilinear assumption in Section 5.3.

5.1 Randomizable Commitment Scheme

Definition 9 (Randomizable Commitment Scheme)**.** A Randomizable commitment scheme for message space $\mathcal{M}$ consists of PPT algorithms $\mathsf{COM} = (\mathsf{C.Setup}, \mathsf{C.Commit}, \mathsf{C.Rand})$ with the following descriptions and properties:

$\mathsf{pp} \leftarrow \mathsf{C.Setup}(1^\lambda)$: On input the security parameter, the setup algorithm outputs public parameters pp.

$\mathsf{com} = \mathsf{C.Commit}(\mathsf{pp}, b; o)$: Using the public parameters pp, the commit algorithm produces commitment com to message $b \in \{0,1\}$ using randomness $o \leftarrow \{0,1\}^{p(\lambda)}$ for some polynomial p. We will refer to o as "opening" for the commitment com.

$\mathsf{com}' = \mathsf{C.Rand}(\mathsf{pp}, \mathsf{com}; o')$: On input parameters pp, commitment com, randomness o', $\mathsf{C.Rand}$ outputs a randomized commitment com' to the same value.

We require the following properties from the commitment scheme:

Perfectly Binding: For all $(m_0, m_1) \in \mathcal{M}$ such that $m_0 \neq m_1$ and for all $o_0, o_1 \in \{0,1\}^{\mathsf{poly}(\lambda)}$

$$\Pr[\mathsf{pp} \leftarrow \mathsf{C.Setup}(1^\lambda) \; : \; \mathsf{C.Commit}(\mathsf{pp}, m_0; o_0) = \mathsf{C.Commit}(\mathsf{pp}, m_1; o_1)] = 0$$

Computationally Hiding: Let $\mathsf{pp} \leftarrow \mathsf{C.Setup}(1^\lambda)$. For all $(m_0, m_1) \in \mathcal{M}$ and $o_0, o_1 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$,

$$\big(\mathsf{C.Commit}(\mathsf{pp}, m_0; o_0)\big) \approx_c \big(\mathsf{C.Commit}(\mathsf{pp}, m_1; o_1)\big)$$

Perfect Randomizability: Let $\mathsf{pp} \leftarrow \mathsf{C.Setup}(1^\lambda)$. There exists an efficient function f_{com} such that for any randomness o, the following holds:

- For every $o' \in \{0,1\}^{\mathsf{poly}(\lambda)}$, $\mathsf{C.Rand}(\mathsf{pp}, \mathsf{C.Commit}(\mathsf{pp}, m; o); o') = \mathsf{C.Commit}(\mathsf{pp}, m; s)$ where $s = f_{\mathsf{com}}(o, o')$.
- If o' is chosen uniformly at random, then $f_{\mathsf{com}}(o, o')$ is uniformly distributed.

We now describe additional properties that we require from our commitment scheme for our FH NIZK construction:

- **Additive Homomorphism:** We require that if $\mathbf{c}_1$ and $\mathbf{c}_2$ are commitments to m_1 and m_2 respectively, then there exists an efficient function f_{add} such that $\mathbf{c} = f_{\mathsf{add}}(\mathbf{c}_1, \mathbf{c}_2)$ is a commitment to $(m_1 + m_2)$.
- **Perfect Equivocation:** There exists a PPT algorithm $\mathsf{C.Setup}'$ and a polynomial time algorithm $\mathsf{C.Equivocate}$ such that
  - $\mathsf{C.Setup}'$ on input the security parameter, outputs pp', such that

    $$\{\mathsf{pp} \leftarrow \mathsf{C.Setup}(1^\lambda) \; : \; \mathsf{pp}\} \approx_c \{\mathsf{pp}' \leftarrow \mathsf{C.Setup}'(1^\lambda) \; : \; \mathsf{pp}'\}.$$

  - Fix any $r_{\mathsf{pp}} \in \{0,1\}^{\mathsf{poly}(\lambda)}$, any $m, m' \in \mathcal{M}$ and any randomness $o \in \{0,1\}^{\mathsf{poly}(\lambda)}$. Let $\mathsf{pp}' = \mathsf{C.Setup}'(1^\lambda; r_{\mathsf{pp}})$ and $\mathbf{c} = \mathsf{C.Commit}(\mathsf{pp}', m; o)$. Algorithm $\mathsf{C.Equivocate}$ on input $(\mathsf{pp}', r_{\mathsf{pp}}, \mathbf{c}, o, m')$ outputs o' such that $\mathbf{c} = \mathsf{C.Commit}(\mathsf{pp}', m'; o')$. Also, for truly random o, $(\mathbf{c}, o')$ is distributed identically to $(\mathbf{c}'', o'')$ where o'' is chosen at random and $\mathbf{c}'' = \mathsf{C.Commit}(\mathsf{pp}', m'; o'')$.

  Note that the parameters output by $\mathsf{C.Setup}(1^\lambda)$ are *binding* and the parameters output by $\mathsf{C.Setup}'(1^\lambda)$ are *hiding*.

We will denote a randomizable commitment which is also additively homomorphic (aH) and equivocable (E) as described above, by a RaHE-commitment scheme.

Remark 1. *We will denote by* ***1*** *and* ***0*** *the canonical commitments to* $1, 0$ *respectively, namely the commitments computed with randomness* $o = 0$*. Given such a commitment it is possible to verify, that the commitment is indeed to* 0 *or* 1*.*

Additional Functionalities for FH NIWI. In our FH NIWI construction, we use a RaHE-commitment scheme which has additional functionalities ($\mathsf{OutParam}, \mathsf{ValidParam}, \mathsf{RParam}, \mathsf{ChangeCom}$) with properties described below:

- **Outputting hiding parameters**: The deterministic algorithm $\mathsf{OutParam}$ takes as input parameters pp^0 and outputs pp^1 such that for all r_{pp}, if $\mathsf{pp}^0 = \mathsf{C.Setup}(1^\lambda; r_{\mathsf{pp}})$, then $\mathsf{pp}^1 = \mathsf{C.Setup}'(1^\lambda; r_{\mathsf{pp}})$.
- **Verifying if two parameters are valid**: The algorithm $\mathsf{ValidParam}$ is an efficient predicate that outputs 1 if $\mathsf{pp}^0 \in \mathsf{C.Setup}(1^\lambda)$ and $\mathsf{pp}^1 = \mathsf{OutParam}(\mathsf{pp}^0)$. It outputs 0 if both parameters are hiding, namely if $\mathsf{pp}^0, \mathsf{pp}^1 \in \mathsf{C.Setup}'(1^\lambda)$.
- **Randomization of parameters**: The RParam algorithm takes as input parameters pp, randomness r'_{pp}, and outputs new parameters pp' such that for all r_{pp} and for $\mathsf{pp} = \mathsf{C.Setup}(1^\lambda; r_{\mathsf{pp}})$, the following properties hold:
  - There exists an efficient function f_{pp}: $f_{\mathsf{pp}}(r_{\mathsf{pp}}, r'_{\mathsf{pp}}) = \sigma$ and $\mathsf{pp}' = \mathsf{RParam}(\mathsf{pp}; r'_{\mathsf{pp}}) = \mathsf{C.Setup}(1^\lambda; \sigma)$.
  - $\mathsf{RParam}(\mathsf{OutParam}(\mathsf{pp}); r'_{\mathsf{pp}}) = \mathsf{OutParam}(\mathsf{RParam}(\mathsf{pp}; r'_{\mathsf{pp}}))$.
- **Transformation of commitments with respect to new parameters**: The $\mathsf{ChangeCom}$ algorithm takes in parameters pp, randomness r'_{pp}, commitment $\mathbf{c}$, and outputs commitment $\mathbf{c}'$ to the same value, with respect to the parameters $\mathsf{pp}' = \mathsf{RParam}(\mathsf{pp}; r'_{\mathsf{pp}})$.

5.1.1 Instantiation from DLIN

We will be using the additively homomorphic commitment scheme used in GOS [GOS06a]. We show that the scheme is randomizable. The scheme is as follows:

$\mathsf{C.Setup}(1^\lambda; r_{\mathsf{pp}})$: Compute $\mathcal{G}(1^\lambda) = (p, \mathbb{G}, \mathbb{G}_T, e, g_p)$ (as defined in Section 3.2). Parse $r_{\mathsf{pp}} = (x, y, z, R, S)$ for $x, y, z, R, S \in \mathbb{Z}_p^*$. Compute $f = g_p^x, h = g_p^y, g = g_p^z$; $(u, v, w) = (f^R, h^S, g^{R+S+1})$. Output

$$\mathsf{pp} = [p, \mathbb{G}, \mathbb{G}_T, e, g_p, f, h, g, u, v, w]^3$$

$\mathsf{C.Commit}(\mathsf{pp}, m)$: Choose $r, s \leftarrow \mathbb{Z}_p^*$ and let opening $o = (r, s)$. Output

$$\mathbf{c} = (c_1, c_2, c_3) = (u^m f^r, v^m h^s, w^m g^{r+s}).$$

$\mathsf{C.Rand}(\mathsf{pp}, \mathbf{c})$: Parse $\mathbf{c} = (c_1, c_2, c_3)$. Choose $r', s' \leftarrow \mathbb{Z}_p^*$ and output

$$\mathbf{c}' = (c_1 f^{r'}, c_2 h^{s'}, c_3 g^{r'+s'}) = (u^m f^{r+r'}, v^m h^{s+s'}, w^m g^{(r+s)+r'+s'}).$$

Proposition 1. *Assuming DLIN, the commitment scheme described above is an additively homomorphic randomizable commitment scheme as per Definition 9.*

Proof. The fact that this commitment scheme is perfectly binding and computationally hiding was proven in GOS [GOS06a]. Their commitment scheme is also **additively homomorphic**: Let $\mathbf{c} = (c_1, c_2, c_3)$ and $\mathbf{c}' = (c'_1, c'_2, c'_3)$ be commitments to (m, m') respectively. Then, $(c_1c'_1, c_2c'_2, c_3c'_3)$ is a commitment to $(m + m')$.

We now prove perfect randomizability: Define $f_{\mathsf{com}}(o, o') = (r + r', s + s')$, where $o = (r, s)$ and $o' = (r', s')$. We have that if $\mathbf{c} = (u^m f^r, v^m h^s, w^m g^{r+s})$ and $\mathbf{c}' = \mathsf{C.Rand}(\mathsf{pp}, \mathbf{c}; o')$, then $\mathbf{c}' = (c'_1, c'_2, c'_3) = (u^m f^{r+r'}, v^m h^{s+s'}, w^m g^{r+r'+s+s'})$. Also for $o, o' \leftarrow (\mathbb{Z}_p^*)^2$, $f_{\mathsf{com}}(o, o')$ is uniformly distributed.

[3] We sometimes write $\mathsf{pp} = [f, h, g, u, v, w]$ and omit $(p, \mathbb{G}, \mathbb{G}_T, e, g_p)$ when it is obvious from the context.

We now prove perfect equivocation: $\mathsf{C.Setup}'(1^\lambda)$ is defined exactly as $\mathsf{C.Setup}(1^\lambda)$ except that it outputs $w = g^{R+S}$, as opposed to g^{R+S+1}. By DLIN, $(f, h, g, f^R, h^S, g^{R+S+1}) \approx_c (f, h, g, f^R, h^S, g^{R+S})$. Also $\mathsf{C.Equivocate}(\mathsf{pp}, r_{\mathsf{pp}}, \mathbf{c}, (r, s), m')$ for $r_{\mathsf{pp}} = (x, y, z, R, S)$, outputs $o' = (r - Rm' + Rm, s - Sm' + Sm)$ which is distributed as fresh o.

□

We now instantiate the additional functionalities and observe that they satisfy the desired properties:

- $\mathsf{OutParam}(\mathsf{pp}^0)$: Parse $\mathsf{pp}^0 = [f, h, g, u, v, w]$. Output $\mathsf{pp}^1 = [f, h, g, u, v, w/g]$. Note that if $\mathsf{pp}^0 \in \mathsf{C.Setup}(1^\lambda)$ then $\mathsf{pp}^1 \in \mathsf{C.Setup}'(1^\lambda)$.
- $\mathsf{ValidParam}(\mathsf{pp}^0, \mathsf{pp}^1)$: Parse $\mathsf{pp}^b = [f, h, g, u, v, w_b]$ for all $b \in \{0, 1\}$. Output 1 if and only if $w_0 = w_1 \cdot g$.
- $\mathsf{RParam}(\mathsf{pp}, r_{\mathsf{pp}})$: Parse $r_{\mathsf{pp}} = (x', y', z', R', S')$ and parse $\mathsf{pp} = [f, h, g, u, v, w]$ for all $b \in \{0, 1\}$. Compute $f' = f^{x'}, h' = h^{y'}, g' = g^{z'}$, $(u', v', w') = \big((uf^{R'})^{x'}, (vh^{S'})^{y'}, (w_b g^{R'+S'})^{z'}\big)$. Output $\mathsf{pp}' = [f', h', g', u', v', w']$.

  We now show the function f_{pp} as required by RParam. Let $\sigma = (x, y, z, R, S)$ such that $\mathsf{pp} = \mathsf{C.Setup}(1^\lambda; \sigma)$. Define

  $$f_{\mathsf{pp}}((x, y, z, R, S), (x', y', z', R', S')) = (xx', yy', zz', R + R', S + S').$$

  Note that $\mathsf{pp}' = \mathsf{C.Setup}(1^\lambda; \sigma')$ where $\sigma' = (xx', yy', zz', R + R', S + S')$.
- $\mathsf{ChangeCom}(\mathsf{pp}, \mathbf{c}, r_{\mathsf{pp}})$: Parse $r_{\mathsf{pp}} = (x', y', z', R', S')$ and parse $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$. Output $\mathbf{c}' = \big((\mathbf{c}_1)^{x'}, (\mathbf{c}_2)^{y'}, (\mathbf{c}_3)^{z'}\big)$.

5.1.2 Dual Tuples and More Functionalities for FH NIWI

We now define the notion of *Dual Tuples* and *Intermediate Parameter* over a group $\mathbb{G}$. We then describe efficient algorithms ($\mathsf{ValidInter}, \mathsf{InterParam}$) that we will be using in our FH NIWI construction. Specifically, these algorithms are used in the construction of proof system for L_{TC}, to prove that two commitments with respect to two different parameters commit to the same value.

Let $(p, \mathbb{G}, \mathbb{G}_T, e, g_p) = \mathcal{G}(1^\lambda)$ and let $f_1, h_1, g_1, f_2, h_2, g_2 \in \mathbb{G}$.

Definition 10 (Dual Tuples)**.** *Tuples* $(f_1, h_1, g_1, f_1^{a_1}, h_1^{a_2}, g_1^{a_3})$ *and* $(f_2, h_2, g_2, f_2^{b_1}, h_2^{b_2}, g_2^{b_3})$ *are said to be* Dual Tuples *if $a_i = b_i$ for all $i \in [3]$.*

Let $\mathsf{pp}_1, \mathsf{pp}_2, \mathsf{pp}^* \in \mathbb{G}^6$ and denote $\mathsf{pp}_i = (f_i, h_i, g_i, u_i, v_i, w_i)$ for all $i \in [2]$.

Definition 11 (Intermediate Parameter)**.** *A tuple* pp^* *is said to be an* Intermediate Parameter *between* $(\mathsf{pp}_1, \mathsf{pp}_2)$ *if* $\mathsf{pp}^* = (f_j, h_j, g_j, u^*, v^*, w^*)$ *for some* $j \in [2]$ *and* $(\mathsf{pp}^*, \mathsf{pp}_{3-j})$ *are dual tuples.*

Remark 2. *If* $\mathsf{pp}_1, \mathsf{pp}_2$ *are such that* $(f_1, h_1, g_1) = (f_2, h_2, g_2)$*, then* pp_1 *is an intermediate parameter between* $(\mathsf{pp}_1, \mathsf{pp}_2)$ *since each tuple is trivially a dual tuple of itself. This is the case for* $\mathsf{pp}_1 \leftarrow \mathsf{C.Setup}(1^\lambda)$ *and* $\mathsf{pp}_2 = \mathsf{OutParam}(\mathsf{pp}_1)$ *as instantiated in Section 5.1.1.*

We now define efficient algorithms ($\mathsf{ValidInter}, \mathsf{InterParam}$) which we will use in conjunction with the RaHE-commitment scheme:

$\mathsf{bit} = \mathsf{ValidInter}(\mathsf{pp}_1, \mathsf{pp}_2, \mathsf{pp}^*)$: The $\mathsf{ValidInter}$ is a efficient predicate which on input $(\mathsf{pp}_1, \mathsf{pp}_2, \mathsf{pp}^*)$ outputs 1 if and only if pp^* is an intermediate parameter between $\mathsf{pp}_1, \mathsf{pp}_2$.

Note that $\mathsf{ValidInter}$ can be efficiently checked using the bilinear map: Check that for some $i \in [2]$, $e(u^*, f_i) = e(u_i, f^*)$, $e(v^*, h_i) = e(v_i, h^*)$ and $e(w^*, g_i) = e(w_i, g^*)$.

Similarly, we define the following algorithm that given $\mathsf{pp}_1, \mathsf{pp}_2$, and randomness $r_{\mathsf{pp}} = (x, y, z, R, S)$ that generates pp_1, outputs the intermediate parameter pp^*.

$\mathsf{pp}^* = \mathsf{InterParam}(\mathsf{pp}_1, \mathsf{pp}_2, r_{\mathsf{pp}})$: $\mathsf{InterParam}$ is an efficient function which takes as input $(\mathsf{pp}_1, \mathsf{pp}_2, r_{\mathsf{pp}})$ which could be of the following forms:

– For all $i \in [2]$, $\mathsf{pp}_i \in \mathsf{C.Setup}(1^\lambda)$ or $\mathsf{pp}_i \in \mathsf{C.Setup}'(1^\lambda)$.

– $\mathsf{pp}_i \in \mathsf{C.Setup}(1^\lambda)$ or $\mathsf{pp}_i \in \mathsf{C.Setup}'(1^\lambda)$ for some $i \in [2]$ and where $\mathsf{pp}_{3-i} = \mathsf{RParam}(\mathsf{pp}_i; r_{\mathsf{pp}})$.

In both cases, it outputs pp^* such that $\mathsf{ValidInter}(\mathsf{pp}_1, \mathsf{pp}_2, \mathsf{pp}^*) = 1$.

$\mathsf{InterParam}(\mathsf{pp}_1, \mathsf{pp}_2, r_{\mathsf{pp}})$ can be instantiated as follows: Parse $\mathsf{pp}_i = [f_i, h_i, g_i, u_i, v_i, w_i]$ for all $i \in [2]$ and parse $r_{\mathsf{pp}} = (x, y, z, R, S)$. Suppose that $\mathsf{pp}_1 = \mathsf{C.Setup}(1^\lambda; r_{\mathsf{pp}})$ (binding case) or $\mathsf{pp}_1 = \mathsf{C.Setup}'(1^\lambda; r_{\mathsf{pp}})$ (hiding case). Compute $(u^*, v^*, w^*) = (f_2^R, h_2^S, g_2^T)$ where $T = R + S + 1$ for the binding case and $T = R + S$ for the hiding case. Output $\mathsf{pp}^* = [f_2, h_2, g_2, u^*, v^*, w^*]$. The case where $\mathsf{pp}_2 = \mathsf{C.Setup}(1^\lambda; r_{\mathsf{pp}})$ or $\mathsf{pp}_2 = \mathsf{C.Setup}'(1^\lambda; r_{\mathsf{pp}})$ is analogous.

Alternatively, if $\mathsf{pp}_{3-j} = \mathsf{RParam}(\mathsf{pp}_j; r_{\mathsf{pp}})$ for some $j \in [2]$, first parse $\mathsf{pp}_i = [f_i, h_i, g_i, u_i, v_i, w_i]$ for all $i \in [2]$ and parse $r_{\mathsf{pp}} = (x, y, z, R, S)$. Output $\mathsf{pp}^* = [f_i^x, h_i^y, g_i^z, u_i^x, v_i^y, w_i^z]$.

5.2 Proofs of Linearity.

In this section we describe the main ingredient for our fully homomorphic proofs, which is a NIWI proof system with additional properties for the parameterized language $L_{\mathsf{Lin}}[\mathsf{pp}]$.

Definition 12 (Linear Tuples). *Let $(p, \mathbb{G}, \mathbb{G}_T, e, g_p) = \mathcal{G}(1^\lambda)$ and let f, h, g be any three generators of $\mathbb{G}$. A tuple $\mathbf{A} = (f^{a_1}, h^{a_2}, g^{a_3})$ is said to be* linear *with respect to (f, h, g) if $a_1 + a_2 = a_3$.*

Before describing the parameterized language $L_{\mathsf{Lin}}[\mathsf{pp}]$, we describe the corresponding setup algorithm for the parameters of the language, given by $\mathsf{Lin.Setup}$.

$\mathsf{Lin.Setup}(1^\lambda)$: Compute $\mathcal{G}(1^\lambda) = (p, \mathbb{G}, \mathbb{G}_T, e, g_p)$. Choose at random $x, y, z \leftarrow \mathbb{Z}_p^*$. Compute $f = g_p^x, h = g_p^y, g = g_p^z$. Output $\mathsf{pp} = [p, \mathbb{G}, \mathbb{G}_T, e, g_p, f, h, g]$.

We abuse notation and let pp denote the output of $\mathsf{Lin.Setup}$ as well as the output of $\mathsf{C.Setup}$. Note that $\mathsf{pp} \leftarrow \mathsf{Lin.Setup}(1^\lambda)$ is a subset of $\mathsf{pp} \leftarrow \mathsf{C.Setup}(1^\lambda)$.

We now define the language $L_{\mathsf{Lin}}[\mathsf{pp}]$ where $\mathsf{pp} \leftarrow \mathsf{Lin.Setup}(1^\lambda)$. $L_{\mathsf{Lin}}[\mathsf{pp}]$ is the language consisting of a pair of tuples such that one of them is linear. It is defined as follows:

$$L_{\mathsf{Lin}}[\mathsf{pp}] = \big\{ \big(\mathbf{A}, \mathbf{B}\big) \mid \exists\, (w_1, w_2, w_3)\ \big((w_1 + w_2 = w_3\big)\ \wedge\ \big(\mathbf{A} = (f^{w_1}, h^{w_2}, g^{w_3})\ \vee\ \mathbf{B} = (f^{w_1}, h^{w_2}, g^{w_3})\big)\big\}$$

5.2.1 NIWI Proof from GOS

We first describe the NIWI proof $(\mathsf{Lin.Prove}, \mathsf{Lin.Verify})$ for $L_{\mathsf{Lin}}[\mathsf{pp}]$ from GOS [GOS06a]:

$\mathsf{Lin.Prove}(\mathsf{pp}, (A_1, A_2, A_3), (B_1, B_2, B_3), (a_1, a_2, a_3))$: Without loss of generality, let (a_1, a_2, a_3) be such that $(A_1, A_2, A_3) = (f^{a_1}, h^{a_2}, g^{a_3})$ and $a_1 + a_2 = a_3$. Choose $t \xleftarrow{\$} \mathbb{Z}_p^*$ and output proof Π which consists of the following matrix:

$$\begin{bmatrix} \pi_{11} = B_1^{a_1} & \pi_{12} = B_2^{a_1} h^{-t} & \pi_{13} = B_3^{a_1} g^{-t} \\ \pi_{21} = B_1^{a_2} f^t & \pi_{22} = B_2^{a_2} & \pi_{23} = B_3^{a_2} g^t \end{bmatrix}$$

$\mathsf{Lin.Verify}(\mathsf{pp}, (A_1, A_2, A_3), (B_1, B_2, B_3), \Pi)$:

– Compute $\pi_{31} = \pi_{11}\pi_{21}$ and $\pi_{32} = \pi_{12}\pi_{22}$ and $\pi_{33} = \pi_{13}\pi_{23}$.
– Check $e(A_1, B_1) = e(f, \pi_{11})$, $e(A_2, B_2) = e(h, \pi_{22})$, $e(A_3, B_3) = e(g, \pi_{33})$.
– Finally check $e(A_1, B_2)e(A_2, B_1) = e(f, \pi_{12})e(h, \pi_{21})$, $e(A_2, B_3)e(A_3, B_2) = e(h, \pi_{23})e(g, \pi_{32})$ and $e(A_1, B_3)e(A_3, B_1) = e(f, \pi_{13})e(g, \pi_{31})$.

Proposition 2 ([GOS06a])**.** *Assuming DLIN, the proof system described above is a perfectly sound witness indistinguishable proof system for the language $L_{\mathsf{Lin}}[\mathsf{pp}]$ (as per Definition 3).*

Remark 3. *If $\Pi = [\pi_{11}, \ldots, \pi_{33}]$ is a valid proof for $((A_1, A_2, A_3), (B_1, B_2, B_3)) \in L_{\mathsf{Lin}}[\mathsf{pp}]$, then $\Pi^{-1} = [\pi_{11}^{-1}, \ldots, \pi_{33}^{-1}]$ is a valid proof for $((A_1^{-1}, A_2^{-1}, A_3^{-1}), (B_1, B_2, B_3)) \in L_{\mathsf{Lin}}[\mathsf{pp}]$ and for $((A_1, A_2, A_3), (B_1^{-1}, B_2^{-1}, B_3^{-1})) \in L_{\mathsf{Lin}}[\mathsf{pp}]$.*

GOS [GOS06a] provided a NIWI proof for $L_{\mathsf{Lin}}[\mathsf{pp}]$ as described above. In our work, we need the NIWI proof system to satisfy two additional properties: The first is malleability with respect to randomization, namely given a tuple $(\mathbf{A}, \mathbf{B}) \in L_{\mathsf{Lin}}[\mathsf{pp}]$ with NIWI proof Π, it is possible to randomize $(\mathbf{A}, \mathbf{B})$ to a new tuple $(\mathbf{A}', \mathbf{B}') \in L_{\mathsf{Lin}}[\mathsf{pp}]$ and maul the proof Π to be proof Π' with respect to $(\mathbf{A}', \mathbf{B}')$. As a second property, we require that the proof system satisfies strong witness indistinguishability with respect to specific distributions (which we describe later in the section).

5.2.2 Malleable Proofs for L_{Lin}

We now show that $(\mathsf{Lin.Prove}, \mathsf{Lin.Verify})$ is malleable with respect to the transformation $\mathsf{Lin.T} = (\mathsf{Lin.Transform}, \mathsf{Lin.WitTrans})$ defined as follows:

$$\mathsf{Lin.Transform}(\mathsf{pp}, \mathbf{A}, \mathbf{B}; (r_1, r_2, s_1, s_2)) \triangleq \left((A_1 f^{r_1}, A_2 h^{r_2}, A_3 g^{r_1+r_2}), (B_1 f^{s_1}, B_2 h^{s_2}, B_3 g^{s_1+s_2})\right)$$

where $\mathsf{pp} = [p, \mathbb{G}, \mathbb{G}_T, e, g_p, f, h, g]$, $\mathbf{A} = (A_1, A_2, A_3)$ and $\mathbf{B} = (B_1, B_2, B_3)$.

$$\mathsf{Lin.WitTrans}(\mathsf{pp}, (\mathbf{A}, \mathbf{B}), (w_1, w_2, w_3); (r_1, r_2, s_1, s_2)) \triangleq (w_1 + z_1, w_2 + z_2, w_3 + z_1 + z_2) \text{ where}$$
$$(z_1, z_2) = (r_1, r_2) \text{ if } \mathbf{A} = (f^{w_1}, h^{w_2}, g^{w_3}) \text{ else } (z_1, z_2) = (s_1, s_2) \text{ if } \mathbf{B} = (f^{w_1}, h^{w_2}, g^{w_3})$$

Mauled proof for $\mathsf{Lin.Transform}(\mathsf{pp}, \mathbf{A}, \mathbf{B}, (r_1, r_2, s_1, s_2)) = (A_1 f^{r_1}, A_2 h^{r_2}, A_3 g^{r_3}), (B_1 f^{s_1}, B_2 h^{s_2}, B_3 g^{s_3})$ is given by $\mathsf{Lin.Maul}(\mathsf{pp}, (\mathbf{A}, \mathbf{B}), (r_1, r_2, s_1, s_2), \Pi)$: Choose $t \leftarrow \mathbb{Z}_p^*$, and output a proof Π' consisting of the following matrix:

$$\begin{bmatrix} \pi'_{11} = \pi_{11} A_1^{s_1} B_1^{r_1} f^{r_1 s_1} & \pi'_{12} = \pi_{12} A_2^{s_1} B_2^{r_1} h^{r_1 s_2 - t} & \pi'_{13} = \pi_{13} A_3^{s_1} B_3^{r_1} g^{r_1 s_3 - t} \\ \pi'_{21} = \pi_{21} A_1^{s_2} B_1^{r_2} f^{r_2 s_1 + t} & \pi'_{22} = \pi_{22} A_2^{s_2} B_2^{r_2} h^{r_2 s_2} & \pi'_{23} = \pi_{23} A_3^{s_2} B_3^{r_2} g^{r_2 s_3 + t} \end{bmatrix}$$

Proposition 3. *Assuming DLIN, the proof system* $(\mathsf{Lin.Prove}, \mathsf{Lin.Verify}, \mathsf{Lin.Maul})$ *is a malleable NIWI for $L_{\mathsf{Lin}}[\mathsf{pp}]$ as per Definition 5, with respect to transformation* $\mathsf{Lin.T} = (\mathsf{Lin.Transform}, \mathsf{Lin.WitTrans})$.

Proof. Malleability: Fix any PPT adversary $\mathcal{A}$ and fix any $\mathsf{pp} \in \mathsf{Lin.Setup}(1^\lambda)$. Let $(\mathbf{A}, \mathbf{B}, \Pi) \leftarrow \mathcal{A}(\mathsf{pp})$. Let $\Pi' = \mathsf{Lin.Maul}(\mathsf{pp}, (\mathbf{A}, \mathbf{B}), (r_1, r_2, s_1, s_2), \Pi; t')$ for randomly chosen r_1, r_2, s_1, s_2, t'.

We prove that $\mathsf{Lin.Verify}(\mathsf{pp}, (\mathbf{A}', \mathbf{B}'), \Pi') = 1$ if and only if $\mathsf{Lin.Verify}(\mathsf{pp}, (\mathbf{A}, \mathbf{B}), \Pi) = 1$. Let $g_1 = f, g_2 = h$ and $g_3 = g$. For $i \in \{1, 2, 3\}$,

$$e(A_i g_i^{r_i}, B_i g_i^{s_i}) = e(A_i, B_i)e(A_i, g_i^{s_i})e(g_i^{r_i}, B_i)e(g_i^{r_i}, g_i^{s_i}) = e(g_i, \pi_{i,i})e(A_i^{r_i} B_i^{s_i} g_i^{r_i s_i}, g_i) = e(g_i, \pi'_{i,i})$$

if and only if $e(A_i, B_i) = e(g_i, \pi_{i,i})$ which are the first three verification check for Π.

For $i \neq j$ and $i, j \in \{1, 2, 3\}$,

$$\begin{aligned}
&e(A_i g_i^{r_i}, B_j g_j^{s_j}).e(A_j g_j^{r_j}, B_i g_i^{s_i}) \\
=&e(A_i, B_j)e(A_j, B_i)e(A_i, g_j^{s_j})e(g_i^{r_i}, B_j)e(A_j, g_i^{s_i})e(g_j^{r_j}, B_i)e(g_i, g_j)^{r_i s_j + r_j s_i} \\
=&e(g_j, \pi_{i,j})e(g_i, \pi_{j,i})e(A_i^{s_j} B_i^{r_j} g_i^{r_i s_j + t'}, g_j)e(B_j^{r_i} A_j^{s_i} g_j^{r_j s_i - t'}, g_i) \text{ for } t' = (b_i r_j - b_j r_i) \\
=&e(g_j, \pi'_{i,j})e(g_i, \pi'_{j,i})
\end{aligned}$$

if and only if $e(A_i, B_j) = e(g_j, \pi_{i,j})e(g_i, \pi_{j,i})$, which are the verification checks for Π.

Perfect Randomizability: Fix any $\mathsf{pp} \in \mathsf{Lin.Setup}(1^\lambda)$ and $(\mathbf{A}, \mathbf{B}) \in L_{\mathsf{Lin}}[\mathsf{pp}]$ with witness $\mathbf{a} = (a_1, a_2, a_3)$. Fix $r_1, r_2, s_1, s_2, t, t'$ such that $\Pi = \mathsf{Lin.Prove}(\mathsf{pp}, \mathbf{A}, \mathbf{B}, \mathbf{a}; t)$, $(\mathbf{A}', \mathbf{B}') = \mathsf{Transform}(\mathsf{pp}, (\mathbf{A}, \mathbf{B}), (r_1, r_2, s_1, s_2))$ and $\Pi' = \mathsf{Lin.Maul}(\mathsf{pp}, (\mathbf{A}, \mathbf{B}), (r_1, r_2, s_1, s_2), \Pi; t')$. Let $r_3 = r_1 + r_2$ and $s_3 = s_1 + s_2$.

Function f_{Lin} is given by: $f_{\mathsf{Lin}}(\mathsf{pp}, \mathbf{a}, t, t', (r_1, r_2, s_1, s_2)) = t + t' + \sigma$ where $\sigma = a_1 s_2 - a_2 s_1 = a_1 s_3 - s_3 a_1 = a_3 s_2 - a_2 s_3$. Also if t' is uniform, $t'' = f_{\mathsf{Lin}}(\mathsf{pp}, \mathbf{a}, t, t', (r_1, r_2, s_1, s_2))$ is distributed as uniform. □

Remark 4. *We denote by* $\mathsf{Lin.Transform}(\mathsf{pp}, (\mathbf{A}, \mathbf{B}), (r_1, r_2))$ *the transformation given by* $\mathsf{Lin.Transform}(\mathsf{pp}, (\mathbf{A}, \mathbf{B}), (r_1, r_2, r_1, r_2))$.

5.2.3 Strong NIWI for L_{Lin}.

For our FH NIWI construction, we require that the NIWI proofs for $(\mathbf{A}, \mathbf{B}) \in L_{\mathsf{Lin}}[\mathsf{pp}]$ satisfy strong witness indistinguishability with respect to distributions $\mathcal{D}_0(\mathsf{pp}), \mathcal{D}_1(\mathsf{pp})$ for $\mathsf{pp} \leftarrow \mathsf{Lin.Setup}(1^\lambda)$. For every $b \in \{0, 1\}$, distribution $\mathcal{D}_b(\mathsf{pp})$ is defined as follows:

Parse $\mathsf{pp} = [p, \mathbb{G}, \mathbb{G}_T, e, g_p, f, h, g]$. Choose $a_1, a_2 \leftarrow \mathbb{Z}_p^*$, let $a_3 = a_1 + a_2$. Let $\mathbf{A}_b = (f^{a_1}, h^{a_2}, g^{a_3 - b})$ and let $\mathbf{B}_b = (f^{a_1}, h^{a_2}, g^{a_3 - b + 1})$. Output $(\mathbf{A}_b, \mathbf{B}_b)$.

Recall that $(\mathsf{Lin.Prove}, \mathsf{Lin.Verify}, \mathsf{Lin.Maul})$ is said to be strong NIWI with respect to distributions $\mathcal{D}_0(\mathsf{pp})$, $\mathcal{D}_1(\mathsf{pp})$ (as per Definition 6), if the following holds:

$$\{\mathsf{pp}, (\mathbf{A}_0, \mathbf{B}_0), \pi_0\} \approx \{\mathsf{pp}, (\mathbf{A}_1, \mathbf{B}_1), \pi_1\}$$

where $(\mathbf{A}_b, \mathbf{B}_b) \leftarrow \mathcal{D}_b(\mathsf{pp})$ and where $\pi_b \leftarrow \mathsf{Lin.Prove}(\mathsf{pp}, \mathbf{A}_b, \mathbf{B}_b, (a_1, a_2, a_3))$.

5.3 Assumption: DLIN with Leakage

In this subsection, we state our new assumption on bilinear maps: *DLIN with Leakage.*
Let $\mathsf{pp} \leftarrow \mathsf{Lin.Setup}(1^\lambda)$ and parse $\mathsf{pp} = [p, \mathbb{G}, \mathbb{G}_T, e, f, h, g]$. DLIN with Leakage states that $\mathcal{D}'_0(1^\lambda) \approx_c \mathcal{D}'_1(1^\lambda)$ where $\mathcal{D}'_b(1^\lambda)$ is as follows:

- $\mathcal{D}'_0(1^\lambda)$: Choose $R, S, t \leftarrow \mathbb{Z}_p^*$ and output pp along with the following matrix:

$$\begin{bmatrix} f^R & h^S & g^{R+S} \\ f^{R^2} & h^{RS-t} & g^{R(R+S+1)-t} \\ f^{RS+t} & h^{S^2} & g^{S(R+S+1)+t} \end{bmatrix}$$

- $\mathcal{D}'_1(1^\lambda)$: Choose $R, S, t \leftarrow \mathbb{Z}_p^*$ and output pp along with the following matrix:

$$\begin{bmatrix} f^R & h^S & g^{R+S-1} \\ f^{R^2} & h^{RS-t} & g^{R(R+S-1)-t} \\ f^{RS+t} & h^{S^2} & g^{S(R+S-1)+t} \end{bmatrix}$$

Proposition 4. *The DLIN with Leakage assumption is secure in the generic group model.*

Proof. We defer the proof to Appendix A.1. □

Proposition 5. *Assuming DLIN with Leakage,* $(\mathsf{Lin.Prove}, \mathsf{Lin.Verify})$ *is strong NIWI for* $L_{\mathsf{Lin}}[\mathsf{pp}]$ *with respect to* $\mathcal{D}_0, \mathcal{D}_1$ *(as described in Section 5.2.3).*

6 Fully Homomorphic NIZK Proofs

In this section, we construct fully homomorphic NIZK proofs for NP from DLIN. Our construction uses certain NIWI proof systems as ingredients; we describe them in Section 6.1. In Section 6.2, we present our FH NIZK construction from these ingredients. In Section 6.3, we instantiate the ingredients from DLIN.

6.1 Ingredients for the FH NIZK Construction

Recall that the generic template for NIZK proofs for $L_{\mathcal{U}}$ from GOS [GOS06b, GOS06a] is as follows:

GOS Template. An $L_{\mathcal{U}}$ instance is of the form (C, out) where $C : \{0,1\}^t \to \{0,1\}$ and $\mathsf{out} \in \{0,1\}$. Denote by n the number of wires in C (including t input wires and excluding the output wire) and denote by m the number of gates. The NIZK proof is computed as follows:

1. Let $w_1, \ldots, w_n$ be the values induced by witness $\mathbf{w} \in \{0,1\}^t$ on all the wires of the circuit C (except the output wire). Commit to all wire values in the circuit C using an additively homomorphic commitment scheme. Let $\mathbf{c}_i$ denote the commitment to wire i for $i \in [n]$.
2. For each commitment $\mathbf{c}_i$ to wire i in C, give a NIWI proof that $\mathbf{c}_i$ is a commitment to a boolen value. We denote such a proof by π^i_{bit} for $i \in [n]$.
3. For each gate in C, give a NIWI proof that the input and the output values to the gate are consistent with the gate functionality. We denote such a proof by π^j_{gate} for $j \in [m]$.
4. Give a canonical commitment to the output value out, denoted by $\mathbf{c}_{\mathsf{out}}$.

A NIZK proof for (C, out) is of the form:

$$\Pi = \left[\{\mathbf{c}_i\}_{i=1}^n, \{\pi^i_{\mathsf{bit}}\}_{i=1}^n, \{\pi^j_{\mathsf{gate}}\}_{j=1}^m, \mathbf{c}_{\mathsf{out}}\right].$$

Our FH NIZK construction follows a similar template and will use three ingredients: A RaHE-commitment scheme $(\mathsf{C.Setup}, \mathsf{C.Commit}, \mathsf{C.Rand})$ as per Definition 9, a NIWI proof system to prove that committed bit is boolean (denoted by *Bit Proofs*), a NIWI proof system to prove that committed bits are consistent with the gate functionality (denoted by *Gate Proofs*).

We note that GOS [GOS06a] defined and constructed NIWI proof systems $(\mathsf{Bit.Prove}, \mathsf{Bit.Verify})$ and $(\mathsf{N.Prove}, \mathsf{N.Verify})$ for bit proofs and gate consistency proofs respectively. This was sufficient for their NIZK construction. However, to achieve full homomorphism, we require the NIWI proof systems to be malleable. At a high level, we need to be able to randomize the commitments in the proofs and maul the bit proofs and gate consistency proofs with respect to the new commitments. In what follows, we describe the concrete transformations for which malleability is needed.

Malleability of Bit Proofs. Let $(\mathsf{C.Setup}, \mathsf{C.Commit}, \mathsf{C.Rand})$ be RaHE-commitment scheme as per Definition 9. For Bit Proofs, we consider the following language parameterized by $\mathsf{pp} \leftarrow \mathsf{C.Setup}(1^\lambda)$.

$$L_{\mathsf{com}}[\mathsf{pp}] = \{\mathbf{c} \mid \exists\ (b, o) \text{ s.t. } \mathbf{c} = \mathsf{C.Commit}(\mathsf{pp}, b; o)\ \wedge\ b \in \{0,1\}\}$$

We require a malleable NIWI proof system $(\mathsf{Bit.Prove}, \mathsf{Bit.Verify}, \mathsf{Bit.Maul})$ for $L_{\mathsf{com}}[\mathsf{pp}]$ (as per Definition 5), with respect to the transformation: $\mathsf{Bit.T} = (\mathsf{Bit.Transform}, \mathsf{Bit.WitTrans})$ given by

$$\mathsf{Bit.Transform}(\mathsf{pp}, \mathbf{c}, o') = \mathsf{C.Rand}(\mathsf{pp}, \mathbf{c}; o') \text{ and } \mathsf{Bit.WitTrans}(\mathsf{pp}, \mathbf{c}, (b, o), o') = f_{\mathsf{com}}(\mathsf{pp}, o, o')$$

where o' is fresh randomness.

Malleability of Gate Consistency Proofs. Without loss of generality, we assume that the circuit is made up of NAND gates. Boolean values b_1, b_2, b_3 satisfy NAND relation that is, $b_3 = b_1 \bar{\wedge} b_2$ if and only if $b_1 + b_2 + 2b_3 - 2 \in \{0,1\}$. For Gate Proofs, we consider the following language parameterized by $\mathsf{pp} \leftarrow \mathsf{C.Setup}(1^\lambda)$.

$$L_{\mathsf{N}}[\mathsf{pp}] = \left\{\{\mathbf{c}_i\}_{i\in[3]} \mid \exists\ \{b_i, o_i\}_{i\in[3]} \text{ s.t. } \mathbf{c}_i = \mathsf{C.Commit}(b_i; o_i)\ \wedge\ (b_3 = b_1 \bar{\wedge} b_2)\ \wedge\ \{b_i \in \{0,1\}\}_{i\in[3]}\right\}$$

We require a malleable NIWI proof system $(\mathsf{N.Prove}, \mathsf{N.Verify}, \mathsf{N.Maul})$ for $L_{\mathsf{N}}[\mathsf{pp}]$ (as per Definition 5), with respect to the transformation: $\mathsf{N.T} = (\mathsf{N.Transform}, \mathsf{N.WitTrans})$ given by

$$\mathsf{N.Transform}(\mathsf{pp}, \{\mathbf{c}_i\}_{i\in[3]}, \{o'_i\}_{i\in[3]}) = \{\mathbf{c}'_i\}_{i\in[3]} \text{ and } \mathsf{N.WitTrans}(\mathsf{pp}, \{\mathbf{c}_i, b_i, o_i, o'_i\}_{i\in[3]}) = f_{\mathsf{com}}(\mathsf{pp}, o, o')$$

where $\mathbf{c}'_i = \mathsf{C.Rand}(\mathsf{pp}, \mathbf{c}_i, o'_i)$ for fresh randomness (o'_1, o'_2, o'_3) and where $o = o_1 + o_2 + 2o_3 - 2$ and $o' = o'_1 + o'_2 + 2o'_3 - 2$.

6.2 FH NIZK Construction

We use the following ingredients for our FH NIZK construction:

- Randomizable commitment scheme as per Definition 9, which is additively homomorphic and equivocable, denoted by
$$(\mathsf{C.Setup}, \mathsf{C.Commit}, \mathsf{C.Rand})$$
- Malleable NIWI proof system for $L_{\mathsf{com}}[\mathsf{pp}]$ with respect to transformation $\mathsf{Bit.Transform}$ from Section 6.1, denoted by
$$(\mathsf{Bit.Prove}, \mathsf{Bit.Verify}, \mathsf{Bit.Maul}).$$
- Malleable NIWI proof system for $L_{\mathsf{N}}[\mathsf{pp}]$ with respect to transformation $\mathsf{N.Transform}$ as described in Section 6.1, denoted by
$$(\mathsf{N.Prove}, \mathsf{N.Verify}, \mathsf{N.Maul}).$$

We now describe our construction:

$\mathsf{NIZK.Setup}(1^k)$: Output $\mathsf{pp} \leftarrow \mathsf{C.Setup}(1^\lambda)$.

$\mathsf{NIZK.Prove}(\mathsf{CRS}, (C, \mathsf{out}), \mathbf{w})$: Let $C : \{0,1\}^t \to \{0,1\}$ consist of n wires (including input wires and excluding output wire), one output wire and m NAND gates. Let $w_1, \ldots, w_n, w_{\mathsf{out}}$ be the boolean values induced by $\mathbf{w} \in \{0,1\}^t$ on all (input and internal) the wires of circuit C and where w_{out} is the output wire ($w_{\mathsf{out}} = \mathsf{out}$).

1. For wire i, commit to the value w_i as follows: Choose o_i at random and compute

$$\mathbf{c}_i = \mathsf{C.Commit}(w_i; o_i).$$

For the output wire w_{out}, use canonical commitments so that $\mathbf{c}_{\mathsf{out}} = \mathbf{1}$ if $\mathsf{out} = 1$ and $\mathbf{c}_{\mathsf{out}} = \mathbf{0}$ if $\mathsf{out} = 0$.

2. For each wire i (except output), generate a proof that commitment $\mathbf{c}_i$ commits to a bit. Namely, compute

$$\pi^i_{\mathsf{bit}} = \mathsf{Bit.Prove}(\mathsf{pp}, \mathbf{c}_i, o_i)$$

where o_i is the opening for commitment $\mathbf{c}_i$.

3. For each NAND gate j, let j_1, j_2 be the input wires and j_3 be the output wire with corresponding commitments $\mathbf{c}_{j_i}$ for $i \in [3]$. Compute

$$\pi^j_{\mathsf{gate}} = \mathsf{N.Prove}(\mathsf{pp}, \{\mathbf{c}_{j_i}\}_{i\in[3]}, \{o_{j_i}\}_{i\in[3]}).$$

Finally output

$$\Pi = \left[\{\mathbf{c}_i\}^n_{i=1}, \{\pi^i_{\mathsf{bit}}\}^n_{i=1}, \{\pi^j_{\mathsf{gate}}\}^m_{j=1}, \mathbf{c}_{\mathsf{out}}\right]$$

$\mathsf{NIZK.Verify}(\mathsf{CRS}, (C, \mathsf{out}), \Pi)$: Parse $\Pi = \left[\{\mathbf{c}_i\}^n_{i=1}, \{\pi^i_{\mathsf{bit}}\}^n_{i=1}, \{\pi^j_{\mathsf{gate}}\}^m_{j=1}, \mathbf{c}_{\mathsf{out}}\right]$.

1. For each wire $i \in [n]$, check whether $\mathsf{Bit.Verify}(\mathsf{pp}, \mathbf{c}_i, \pi^i_{\mathsf{bit}}) = 1$. Else output 0.
2. For each NAND gate $j \in [m]$, with input wires j_1, j_2 and output wire j_3 and with corresponding commitments $\mathbf{c}_{j_i}$, for $i = 1, 2, 3$. Check that $\mathsf{N.Verify}(\mathsf{CRS}, \{\mathbf{c}_{j_i}\}^3_{i=1}, \pi^j_{\mathsf{gate}}) = 1$. Else output 0.
3. Finally check that $\pi_{\mathsf{out}} = \mathbf{1}$ for $\mathsf{out} = 1$ and $\pi_{\mathsf{out}} = \mathbf{0}$ for $\mathsf{out} = 0$.

$\mathsf{NIZK.Rand}(\mathsf{CRS}, (C, \mathsf{out}), \Pi))$: Parse $\Pi = [\{\mathbf{c}_i\}^n_{i=1}, \{\pi^i_{\mathsf{bit}}\}^n_{i=1}, \{\pi^j_{\mathsf{gate}}\}^m_{j=1}, \mathbf{c}_{\mathsf{out}}]$.

1. For each wire i, choose o'_i at random and compute $\mathbf{c}'_i = \mathsf{C.Rand}(\mathsf{pp}, \mathbf{c}_i, o'_i)$.
2. Compute $\pi^{i'}_{\mathsf{bit}} \leftarrow \mathsf{Bit.Maul}(\mathsf{pp}, \mathbf{c}_i, o'_i, \pi^i_{\mathsf{bit}})$.
3. For each NAND gate j, with input wires j_1, j_2 and output wire j_3, compute $\pi^{j'}_{\mathsf{gate}} \leftarrow \mathsf{N.Maul}(\mathsf{pp}, \{\{\mathbf{c}_{j_i}, o'_{j_i}\}_{i\in[3]}, \pi^j_{\mathsf{gate}})$.
4. Finally keep the output proof $\mathbf{c}_{\mathsf{out}}$ same as before. Output

$$\Pi' = \left[\{\mathbf{c}'_i\}^n_{i=1}, \{\pi^{i'}_{\mathsf{bit}}\}^n_{i=1}, \{\pi^{j'}_{\mathsf{gate}}\}^m_{j=1}, \mathbf{c}_{\mathsf{out}}\right]$$

$\mathsf{NIZK.Eval}(\mathsf{CRS}, \{(C_i, b_i, \Pi_i)\}^k_{i=1}, C')$:

1. Compute $(C, \mathsf{out}^*) = \mathsf{Compose}(\{(C_i, b_i, \Pi_i)\}^k_{i=1}, C')$.
2. Let $\pi^i_{\mathsf{out}} \in \Pi'_i$ be the gate consistency proof for the output gate out^i of circuit C_i for $i \in [k]$. Compute $\widehat{\Pi_i}$ as the proof Π'_i without the proof π^i_{out}, namely $\widehat{\Pi_i} = \Pi'_i \setminus \pi^i_{\mathsf{out}}$.
3. Compute a proof for C' with witness $(b_1, \ldots, b_k)$ by computing: $\Pi^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{CRS}, (C', \mathsf{out}^*), (b_1, \ldots, b_k))$ where $\mathsf{out}^* = C'(b_1, \ldots, b_k)$.

4. For each output gate out^i for C_i, $i \in [k]$, let i_1, i_2 be the input wires to the gate and i_3 be the output wire (with value b_i). Let o'_{i_3} be the randomness used in step 2 such that $\mathbf{c}'_{i_3} \in \Pi'$ and $\mathbf{c}'_{i_3} = \mathsf{C.Commit}(\mathsf{pp}, b_i, o'_{i_3})$. Compute $(\pi^i_{\mathsf{out}})' = \mathsf{N.Maul}(\mathsf{pp}, \{\{\mathbf{c}'_{j_i}, o'_{j_i}\}_{i\in[3]}, \pi^i_{\mathsf{out}})$ where $o'_{i_k} = 0$ for $k \in [2]$.
5. Let $\Pi = \left[\widehat{\Pi_1}, \ldots, \widehat{\Pi_k}, \Pi^*, (\pi^1_{\mathsf{out}})', \ldots, (\pi^k_{\mathsf{out}})'\right]$. Compute $\Pi' \leftarrow \mathsf{NIZK.Rand}(\mathsf{CRS}, (C, \mathsf{out}^*), \Pi)$. Finally output $(C, \mathsf{out}^*, \Pi')$.

Theorem 3. *The construction as described above is a fully homomorphic NIZK proof system for $L_{\mathcal{U}}$ as per Definition 7.*

Proof. The algorithms ($\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify}$) are identical to the NIZK construction in Groth-Ostrovsky-Sahai [GOS06a]. Hence completeness, statistical soundness[4] and perfect zero-knowledge follows. It is left to prove the following claims.

Claim 1. Π_{FHNIZK} *satisfies completeness of evaluation.*

Proof. Completeness of evaluation follows from completeness of randomizability and malleability of Bit proofs and Gate consistency proofs. ☐

Claim 2. Π_{FHNIZK} *satisfies randomizability.*

Proof. We show that for any instance (C, b) with witness w and any proof Π such that $\mathsf{NIZK.Verify}(\mathsf{CRS}, (C, b), \Pi) = 1$, the following distributions are identical:

$$\left\{(C, b), w, \Pi, \mathbf{R}, \Pi_f\right\} \text{ and } \left\{(C, b), w, \Pi, \mathbf{R}, \Pi'\right\}$$

where Π_f is a fresh proof obtained as $\Pi_f \leftarrow \mathsf{NIZK.Prove}(\mathsf{CRS}, (C, b), w)$, Π' is a randomized proof obtained as $\Pi' \leftarrow \mathsf{NIZK.Rand}(\mathsf{CRS}, (C, b), \Pi)$ and $\mathbf{R}$ is such that $\Pi = \mathsf{NIZK.Prove}(\mathsf{CRS}, (C, b), w; \mathbf{R})$. Parse

$$\Pi = [\{\mathbf{c}_i\}_{i=1}^n, \{\pi^i_{\mathsf{bit}}\}_{i=1}^n, \{\pi^j_{\mathsf{gate}}\}_{j=1}^m, \mathbf{c}_{\mathsf{out}}].$$

We will parse

$$\mathbf{R} = [o_1, \ldots, o_n, t_1, \ldots, t_n, s_1, \ldots, s_m]$$

where o_i is such that $\mathbf{c}_i = \mathsf{C.Commit}(\mathsf{pp}, w_i, o_i)$ where w_i is the value on wire i, t_i is such that $\pi^i_{\mathsf{bit}} = \mathsf{Bit.Prove}(\mathsf{pp}, \mathbf{c}_i, o_i; t_i)$ for $i \in [n]$. Finally s_j is such that
$\pi^j_{\mathsf{gate}} = \mathsf{N.Prove}(\mathsf{pp}, \{\mathbf{c}_{j_i}, o_{j_i}\}_{i\in[3]}; s_j)$.

Similarly, let $\Pi' = \mathsf{NIZK.Rand}(\mathsf{CRS}, (C, b), \Pi; \mathbf{R}')$ and parse

$$\mathbf{R}' = [o'_1, \ldots, o'_n, t'_1, \ldots, t'_n, s'_1, \ldots, s'_m]$$

where o'_i is such that $\mathbf{c}'_i = \mathsf{C.Rand}(\mathsf{pp}, \mathbf{c}, o'_i)$ for $i \in [n]$, t'_i is such that $\pi^{i'}_{\mathsf{bit}} = \mathsf{Bit.Maul}(\mathsf{pp}, \mathbf{c}_i, o'_i, \pi^i_{\mathsf{bit}}; t'_i)$ for $i \in [n]$. Finally s'_j is such that $\pi^{j'}_{\mathsf{gate}} = \mathsf{N.Maul}(\mathsf{pp}, \{\mathbf{c}_{j_i}, o'_{j_i}\}_{i\in[3]}, s'_j, \pi^j_{\mathsf{N}})$.

By perfect randomizability of the commitment scheme, there exists f_{com} such that $o''_i = f_{\mathsf{com}}(o_i, o'_i)$ and $\mathbf{c}'_i = \mathsf{C.Commit}(\mathsf{pp}, w_i; o''_i)$ and o''_i is uniformly distributed. By perfect randomizability of bit proofs, there exists f_{bit} such that $f_{\mathsf{bit}}(t_i, o_i, o'_i, t'_i) = t''_i$ such that $\pi^{i'}_{\mathsf{bit}} = \mathsf{Bit.Prove}(\mathsf{pp}, \mathbf{c}'_i, o''_i; t''_i)$ and t''_i is uniformly distributed.

Similarly, perfect randomizability of gate consistency proofs, there exists f_{gate} such that $f_{\mathsf{gate}}(s_j, s''_j, \{o_{j_i}, o'_{j_i}\}_{i\in[3]}) = s''_j$ and $\pi^{j'}_{\mathsf{gate}} = \mathsf{N.Prove}(\mathsf{pp}, \{\mathbf{c}_{j_i}, o''_{j_i}\}_{i\in[3]}; s''_j)$ where s''_j is distributed as uniform.

[4]The GOS NIZK construction achieves statistical soundness in the common random string model.

We can now identify $\mathbf{R}''$ such that $\Pi' = \mathsf{NIZK.Prove}(\mathsf{CRS}, (C, b), w; \mathbf{R}'')$ as follows:

$$\mathbf{R}'' = [o''_1, \dots, o''_n, t''_1, \dots, t''_n, s''_1, \dots, s''_m]$$

which is distributed as uniform. It follows that $\{(C, b), w, \Pi, \mathbf{R}, \Pi_f\}$, $\{(C, b), w, \Pi, \mathbf{R}, \Pi'\}$ are identical, where $\Pi_f = \mathsf{NIZK.Prove}(\mathsf{CRS}, (C, b), w; \mathbf{S})$ and $\Pi' = \mathsf{NIZK.Prove}(\mathsf{CRS}, (C, b), w; \mathbf{R}'')$ for truly random $\mathbf{S}, \mathbf{R}''$.

□

Claim 3. Π_{FHNIZK} *satisfies unlinkability.*

Proof. Follows from completeness of underlying primitives and randomizability of the NIZK. □

□

6.3 Instantiating the Ingredients

We now give concrete instantiations of the two malleable NIWI proof systems described in Section 6.1: bit proofs ($\mathsf{Bit.Prove}, \mathsf{Bit.Verify}, \mathsf{Bit.Maul}$) and gate consistency proofs ($\mathsf{N.Prove}, \mathsf{N.Verify}, \mathsf{N.Maul}$) from DLIN.

Bit Proofs. Let ($\mathsf{C.Setup}, \mathsf{C.Commit}, \mathsf{C.Rand}$) be the RaHE-commitment scheme from Section 5.1. Recall the language,

$$L_{\mathsf{com}}[\mathsf{pp}] = \{(c_1, c_2, c_3) \mid \exists\ (b, r, s) \text{ s.t. } ((c_1, c_2, c_3) = (u^b f^r, v^b h^s, w^b g^{r+s}))\ \wedge\ b \in \{0, 1\}\}$$

Let $\mathbf{c} = (c_1, c_2, c_3)$, define $\delta_{\mathsf{pp}}(\mathbf{c}) \triangleq (c_1/u, c_2/v, (c_3/w))$. We drop pp when it is obvious from the context. Observe that if $\mathbf{c} \in L_{\mathsf{com}}[\mathsf{pp}]$ then $\mathbf{c}$ or $\delta(\mathbf{c})$ is linear. In particular, $\mathbf{c}$ is linear if $\mathbf{c}$ commits to 0 and $\delta(\mathbf{c})$ is linear if $\mathbf{c}$ commits to 1.
Let ($\mathsf{Lin.Prove}, \mathsf{Lin.Verify}, \mathsf{Lin.Maul}$) be the malleable NIWI proof system for $L_{\mathsf{Lin}}[\mathsf{pp}]$ with respect to transformation $\mathsf{Lin.Transform}$. ($\mathsf{Bit.Prove}, \mathsf{Bit.Verify}, \mathsf{Bit.Maul}$) is as follows:

$\mathsf{Bit.Prove}(\mathsf{pp}, \mathbf{c}, o)$: Let $\mathsf{pp} = [p, \mathbb{G}, \mathbb{G}_T, e, g, f, h, u, v, w]$. Parse $\mathbf{c} = (c_1, c_2, c_3)$ and $o = (r, s)$. Compute $\delta_{\mathsf{pp}}(\mathbf{c}) = (c_1/u, c_2/v, (c_3/w)^{-1})$. Output $\pi_{\mathsf{bit}} \leftarrow \mathsf{Lin.Prove}(\mathsf{pp}, (\mathbf{c}, \delta(\mathbf{c})), (r, s, (r + s)))$.

$\mathsf{Bit.Verify}(\mathsf{pp}, \mathbf{c}, \pi_{\mathsf{bit}})$: Output $\mathsf{Lin.Verify}(\mathsf{pp}, (\mathbf{c}, \delta(\mathbf{c})), \pi_{\mathsf{bit}})$.

$\mathsf{Bit.Maul}(\mathsf{pp}, \mathbf{c}, o', \pi_{\mathsf{bit}})$: Parse $o' = (r', s')$. Output $\mathsf{Lin.Maul}(\mathsf{pp}, (\mathbf{c}, \delta(\mathbf{c})), (r', s'), \pi_{\mathsf{bit}})$.

Gate Proofs. Recall that boolean values b_1, b_2, b_3 satisfy NAND relation that is, $b_3 = b_1 \bar{\wedge} b_2$ if and only if $b_1 + b_2 + 2b_3 - 2 \in \{0, 1\}$. We use this observation along with the homomorphic properties of the underlying commitment to prove gate consistency for every NAND gate. This approach was also used in GOS [GOS06a].

Definition 13 (NAND Function). *Function f_{N} takes as input three commitments $\{\mathbf{c}_i\}_{i=1}^3$ where $\mathbf{c}_i = (c_1^i, c_2^i, c_3^i)$ along with (f, h, g), and outputs a homomorphically computed commitment as follows:*

$$f_{\mathsf{N}}(\{\mathbf{c}_i\}_{i=1}^3, f, h, g) \triangleq (c_1^1 c_1^2 (c_1^3)^2 f^{-2}, c_2^1 c_2^2 (c_2^3)^2 h^{-2}, c_3^1 c_3^2 (c_3^3)^2 g^{-2})$$

We also define function $\mathcal{R}_{\mathsf{N}}$ which on input randomness to commitments $\{\mathbf{c}_i\}_{i=1}^3$, outputs the randomness corresponding to the commitment $f_{\mathsf{N}}\big(\{\mathbf{c}^i\}_{i=1}^3, f, h, g\big)$.

$$\mathcal{R}_{\mathsf{N}}(\{(r_i, s_i)\}_{i=1}^3) \triangleq (r_1 + r_2 + 2r_3 - 2, s_1 + s_2 + 2s_3 - 2)$$

Recall the language, parameterized by $\mathsf{pp} = [p, \mathbb{G}, \mathbb{G}_T, e, g, f, h, u, v, w]$:

$$L_{\mathsf{N}}[\mathsf{pp}] = \Big\{\{\mathbf{c}_i\}_{i=1}^3 \mid \exists\ (b_1, b_2, b_3) \text{ and } \{(r_i, s_i)\}_{i=1}^3 \text{ s.t.}$$

$$((c_1^i, c_2^i, c_3^i) = (u^{b_i} f^{r_i}, v^{b_i} h^{s_i}, w^{b_i} g^{r_i+s_i}))\ \wedge\ (b_3 = b_1 \bar{\wedge}\, b_2)\Big\}$$

The NIWI proof construction for $L_{\mathsf{N}}[\mathsf{pp}]$ is as follows:

$\mathsf{N.Prove}(\mathsf{pp}, \{\mathbf{c}_i\}_{i=1}^3, \{b_i, o_i\}_{i=1}^3)$: Compute a commitment to $b = (b_1 + b_2 + 2b_3 - 2)$ homomorphically. Namely, compute $\mathbf{d} = f_{\mathsf{N}}\big(\{(c_1^i, c_2^i, c_3^i)\}_{i=1}^3, f, h, g\big)$. Note that $(d_1, d_2, d_3) \in L_{\mathsf{com}}$ with witness $b = (b_1 + b_2 + 2b_3 - 2)$ and $(r, s) = \mathcal{R}_{\mathsf{N}}(\{(r_i, s_i)\}_{i=1}^3)$ where $o_i = (r_i, s_i)$. Output L_{com} proof given by:

$$\Pi_{\mathsf{N}} = \mathsf{Bit.Prove}\big(\mathsf{pp}, \mathbf{d}, \delta(\mathbf{d}), (r, s, (r + s))\big).$$

$\mathsf{N.Verify}(\mathsf{pp}, \{\mathbf{c}_i\}_{i=1}^3, \Pi_{\mathsf{N}})$: Compute $\mathbf{d} = f_{\mathsf{N}}(\{\mathbf{c}_i\}_{i=1}^3, f, h, g)$ and $\mathbf{d}' = \delta(\mathbf{d})$. Output $\mathsf{Bit.Verify}(\mathsf{pp}, \mathbf{d}, \mathbf{d}', \Pi_{\mathsf{N}})$.

$\mathsf{N.Maul}(\mathsf{pp}, \{\mathbf{c}_i, o_i\}_{i=1}^3, \pi_{\mathsf{N}})$: Parse $o_i' = (r_i', s_i')$ for $i \in [3]$ and compute $(r', s') = \mathcal{R}_{\mathsf{N}}(\{(r_i', s_i')\}_{i=1}^3)$. Output $\mathsf{Lin.Maul}\big(\mathsf{pp}, (\mathbf{d}, \mathbf{d}'), (r', s'), \pi_{\mathsf{N}}\big)$.

We again note that ($\mathsf{Bit.Prove}, \mathsf{Bit.Verify}$) and ($\mathsf{N.Prove}, \mathsf{N.Verify}$) are taken verbatim from GOS [GOS06a]. In this work, we show that both the proof systems are malleable with respect to $\mathsf{Bit.T}$ and $\mathsf{N.T}$ respectively. For both the proof systems, malleability follows from malleability of $L_{\mathsf{Lin}}[\mathsf{pp}]$ as per Proposition 3.

7 Fully Homomorphic NIWI Proofs

In this section, we construct a fully homomorphic (FH) NIWI proof system. In Section 7.1, we describe our main ingredient: a malleable proof system (with additional properties) for proving that two commitments with respect to different parameters commit to the same value. We defer the construction of this malleable proof system to Section 7.3. In Section 7.2, we describe our construction for FH NIWI and prove security.

7.1 Ingredients for the FH NIWI Construction

Our <u>first</u> ingredient is ($\mathsf{C.Setup}, \mathsf{C.Commit}, \mathsf{C.Rand}$), a RaHE-commitment scheme with the additional functionalities ($\mathsf{OutParam}, \mathsf{ValidParam}, \mathsf{RParam}, \mathsf{ChangeCom}, \mathsf{ValidInter}, \mathsf{InterParam}$) as defined in Section 5.1.

Our <u>second</u> ingredient is a malleable proof system ($\mathsf{TC.Prove}, \mathsf{TC.Verify}, \mathsf{TC.Maul}$) for the language L_{TC} defined as follows:

$$L_{\mathsf{TC}} = \Big\{(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2) \mid \exists\ (b, \mathsf{pp}_*, o_1, o_2) \text{ s.t.}$$

$$\{\mathbf{c}_i = \mathsf{C.Commit}(\mathsf{pp}_i, b; o_i)\}_{i \in [2]}\ \wedge\ \big(\mathsf{ValidInter}(\mathsf{pp}_1, \mathsf{pp}_2, \mathsf{pp}_*) = 1\big)\Big\}$$

Recall that pp_* is the intermediate parameter between $\mathsf{pp}_1, \mathsf{pp}_2$. It is a hard-to-compute function of the parameters which we require as an additional witness for the language.

The malleability is with respect to the transformation $\mathsf{TC.T} = (\mathsf{TC.Transform}, \mathsf{TC.WitTrans})$. $\mathsf{TC.Transform}$ takes as input an instance $(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2)$, randomness $(r^1_{\mathsf{pp}}, r^2_{\mathsf{pp}}, o_1, o_2)$ and outputs transformed instance $(\mathbf{c}'_1, \mathbf{c}'_2, \mathsf{pp}'_1, \mathsf{pp}'_2)$.

In detail, $\mathsf{TC.Transform}$ on input $(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2)$, does the following:

- Randomize the parameters as follows: For all $i \in [2]$, compute $\mathsf{pp}'_i = \mathsf{RParam}(\mathsf{pp}_i; r^i_{\mathsf{pp}})$.
- Change the commitment $\mathbf{c}_i$ to be with respect to the new parameters pp'_i, by computing $z_i = \mathsf{ChangeCom}(\mathsf{pp}_i, \mathbf{c}_i; r^i_{\mathsf{pp}})$ for all $i \in [2]$.
- Randomize the commitments as follows: For all $i \in [2]$, compute $\mathbf{c}'_i = \mathsf{C.Rand}(\mathsf{pp}'_i, z_i; o_i)$. Output $(\mathbf{c}'_1, \mathbf{c}'_2, \mathsf{pp}'_1, \mathsf{pp}'_2)$.

Correspondingly,

$$\mathsf{TC.WitTrans}\big((\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2), (b, \mathsf{pp}_*, o_1, o_2), (r^1_{\mathsf{pp}}, r^2_{\mathsf{pp}}, o'_1, o'_2)\big) = (b, \widehat{\mathsf{pp}}, r_1, r_2)$$

where $\widehat{\mathsf{pp}} = \mathsf{InterParam}(\mathsf{pp}_1, \mathsf{pp}_2, r^1_{\mathsf{pp}})$ and where for every $i \in [2]$, $r_i = f_{\mathsf{com}}(o_i, o'_i)$. Recall that $\mathsf{InterParam}$ and f_{com} are as per the definition of the RaHE-commitment scheme described in Section 5.1.

Let us look at the soundness and secrecy requirements from this proof system. We weaken the soundness requirement of our NIWI proof system and require a stronger secrecy property from the proof system. We now describe both of these properties:

1. *Weak Soundness:* Rather than requiring soundness to hold for every $(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2) \in L_{\mathsf{TC}}$, we only require soundness to hold for all instances for which $\mathsf{pp}_1, \mathsf{pp}_2 \in \mathsf{C.Setup}(1^\lambda)$ (when both parameters are binding).

   Note that our construction for NIWI proof of L_{TC} achieves standard soundness, however for the FH NIWI construction it suffices for the proof system to have weak soundness.

2. *Strong Secrecy:* We require that the distributions $\mathcal{D}_{\mathsf{Bind}}$ and $\mathcal{D}_{\mathsf{Hide}}$ (described below) are computationally indistinguishable. Both the distributions output two parameters $\mathsf{pp}, \mathsf{pp}'$, four commitments $\mathbf{c}_0, \mathbf{c}'_0, \mathbf{c}_1, \mathbf{c}'_1$ where $\mathbf{c}_0, \mathbf{c}_1$ are with respect to pp and $\mathbf{c}'_0, \mathbf{c}'_1$ are with respect to pp'. The distributions also output openings to the four commitments and two L_{TC} proofs. We now explain in detail.

   In the output of $\mathcal{D}_{\mathsf{Bind}}$, pp is the binding parameter and pp' is hiding. The commitments $\mathbf{c}_0, \mathbf{c}'_0$ commit to 0 and $\mathbf{c}_1, \mathbf{c}'_1$ commit to 1. Honest L_{TC} proofs $\Pi^0_{\mathsf{TC}}, \Pi^1_{\mathsf{TC}}$ are computed with respect to $(\mathbf{c}_0, \mathbf{c}'_0, \mathsf{pp}, \mathsf{pp}')$ and $(\mathbf{c}_1, \mathbf{c}'_1, \mathsf{pp}, \mathsf{pp}')$ respectively. Finally, the commitments $\mathbf{c}'_0, \mathbf{c}'_1$ (with respect to the hiding parameter pp') are equivocated to obtain openings to the compliment bit and $\mathcal{D}_{\mathsf{Bind}}$ outputs honest openings o_0, o_1 for $\mathbf{c}_0, \mathbf{c}_1$ (openings to 0, 1 respectively) along with equivocated openings o'_0, o'_1 for $\mathbf{c}'_0, \mathbf{c}'_1$ (openings to 1, 0 respectively).

   In the output of $\mathcal{D}_{\mathsf{Hide}}$, pp is the hiding parameter and pp' is binding. The commitments $\mathbf{c}_0, \mathbf{c}'_0$ now commit to 1 and $\mathbf{c}_1, \mathbf{c}'_1$ commit to 0. Honest L_{TC} proofs $\Pi^0_{\mathsf{TC}}, \Pi^1_{\mathsf{TC}}$ are computed with respect to $(\mathbf{c}_0, \mathbf{c}'_0, \mathsf{pp}, \mathsf{pp}')$ and $(\mathbf{c}_1, \mathbf{c}'_1, \mathsf{pp}, \mathsf{pp}')$ respectively. Finally, the commitments $\mathbf{c}_0, \mathbf{c}_1$ (with respect to the hiding parameter pp) are equivocated to obtain openings to the compliment bit and $\mathcal{D}_{\mathsf{Hide}}$ outputs equivocated openings o_0, o_1 for $\mathbf{c}_0, \mathbf{c}_1$ (openings to 0, 1 respectively) and honest openings o'_0, o'_1 for $\mathbf{c}'_0, \mathbf{c}'_1$ (openings to 1, 0 respectively).

   Note that in both the distributions, the openings o_0, o'_0, o_1, o'_1 are with respect to the values $0, 1, 1, 0$ respectively. Formally, the distributions are as follows:

- $\mathcal{D}_{\mathsf{Bind}}(1^\lambda)$: Choose r_{pp} at random and compute $\mathsf{pp} = \mathsf{C.Setup}(1^\lambda; r_{\mathsf{pp}})$. Compute $\mathsf{pp}' = \mathsf{OutParam}(\mathsf{pp})$. For every $d \in \{0,1\}$, do the following:
  - Choose o_d, o''_d at random and compute $\mathbf{c}_d = \mathsf{C.Commit}(\mathsf{pp}, d\,; o_d)$, $\mathbf{c}'_d = \mathsf{C.Commit}(\mathsf{pp}', d; o''_d)$.
  - Compute $\Pi^d_{\mathsf{TC}} \leftarrow \mathsf{TC.Prove}((\mathbf{c}_d, \mathbf{c}'_d, \mathsf{pp}, \mathsf{pp}'), (d, \mathsf{pp}, o_d, o''_d))$.[5]
  - Compute $o'_d = \mathsf{C.Equivocate}(\mathsf{pp}', r_{\mathsf{pp}}, \mathbf{c}'_d, o''_d, 1-d)$.

  Output $\big(\mathsf{pp}, \mathsf{pp}', \mathbf{c}_0, \mathbf{c}'_0, \mathbf{c}_1, \mathbf{c}'_1, o_0, o'_0, o_1, o'_1, \Pi^0_{\mathsf{TC}}, \Pi^1_{\mathsf{TC}}\big)$.

- $\mathcal{D}_{\mathsf{Hide}}(1^\lambda)$: Choose r_{pp} at random and compute $\mathsf{pp} = \mathsf{C.Setup}'(1^\lambda; r_{\mathsf{pp}})$. Compute $\mathsf{pp}' = \mathsf{OutParam}(\mathsf{pp})$. For every $d \in \{0,1\}$, do the following:
  - Choose o'_d, o''_d at random. Compute $\mathbf{c}_d = \mathsf{C.Commit}(\mathsf{pp}, 1-d\,; o''_d)$ and compute $\mathbf{c}'_d = \mathsf{C.Commit}(\mathsf{pp}', 1-d; o'_d)$.
  - Compute $\Pi^d_{\mathsf{TC}} \leftarrow \mathsf{TC.Prove}((\mathbf{c}_d, \mathbf{c}'_d, \mathsf{pp}, \mathsf{pp}'), (1-d, \mathsf{pp}, o''_d, o'_d))$.
  - Compute $o_d = \mathsf{C.Equivocate}(\mathsf{pp}, r_{\mathsf{pp}}, \mathbf{c}_d, o''_d, d)$.

  Output $\big(\mathsf{pp}, \mathsf{pp}', \mathbf{c}_0, \mathbf{c}'_0, \mathbf{c}_1, \mathbf{c}'_1, o_0, o'_0, o_1, o'_1, \Pi^0_{\mathsf{TC}}, \Pi^1_{\mathsf{TC}}\big)$.

Remark 5. Note that strong secrecy implies plain witness indistinguishability for L_{TC} instances with more than one witnesses. In particular, it implies that the following two distributions are indistinguishable for any $d \in \{0,1\}$:

- E^d_0: Choose r_1, r_2 at random and for all $i \in [2]$, compute $\mathsf{pp}_i = \mathsf{C.Setup}'(1^\lambda; r_i)$. Compute $\mathsf{pp}_* = \mathsf{InterParam}(\mathsf{pp}_1, \mathsf{pp}_2, r_1)$. Choose o_1, o_2 at random and for all $i \in [2]$, compute $\mathbf{c}_i = \mathsf{C.Commit}(\mathsf{pp}_i, d; o_i)$. Finally compute $\pi^0 \leftarrow \mathsf{TC.Prove}\big((\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2), (d, \mathsf{pp}_*, o_1, o_2)\big)$. Output $\big(\mathsf{pp}_1, \mathsf{pp}_2, \mathbf{c}_1, \mathbf{c}_2, \pi^0\big)$.
- E^d_1: Choose r_1, r_2 at random and for all $i \in [2]$, compute $\mathsf{pp}_i = \mathsf{C.Setup}'(1^\lambda; r_i)$. Compute $\mathsf{pp}_* = \mathsf{InterParam}(\mathsf{pp}_1, \mathsf{pp}_2, r_1)$. Choose o_1, o_2 at random and for all $i \in [2]$, compute $\mathbf{c}_i = \mathsf{C.Commit}(\mathsf{pp}_i, d; o_i)$. For all $i \in [2]$, compute $s_i = \mathsf{C.Equivocate}(\mathsf{pp}_i, r_i, \mathbf{c}_i, o_i, 1-d)$. Finally compute $\pi^1 \leftarrow \mathsf{TC.Prove}\big((\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2), (1-d, \mathsf{pp}_*, s_1, s_2)\big)$. Output $\big(\mathsf{pp}_1, \mathsf{pp}_2, \mathbf{c}_1, \mathbf{c}_2, \pi^1\big)$.

Additional Procedures. We now describe procedures ($\mathsf{OutCom}, \mathsf{VerCom}, \mathsf{RCom}$) that will help describe the FH NIWI construction succinctly. These procedures use the RaHE-commitment scheme and the algorithms ($\mathsf{TC.Prove}, \mathsf{TC.Verify}, \mathsf{TC.Maul}$) as subroutines.

Notation: We denote by $(\mathsf{pp}^0_j, \mathsf{pp}^1_j)$ the public parameters associated with gate j where pp^0_j denotes the binding parameters and pp^1_j are the hiding parameters. For any $b \in \{0,1\}$, we denote by $(\mathsf{pp}^b_j)'$ the randomized parameters corresponding to pp^b_j.

$(\sigma_c, \sigma_\pi, \mathsf{st}) \leftarrow \mathsf{OutCom}(\mathsf{pp}^0_1, \mathsf{pp}^0_2, r^1_{\mathsf{pp}}, b)$: The OutCom algorithm takes as input two pairs of parameters $\mathsf{pp}^0_1, \mathsf{pp}^0_2 \in \mathsf{C.Setup}(1^\lambda)$, randomness r^1_{pp} such that $\mathsf{pp}^0_1 = \mathsf{C.Setup}(1^\lambda; r^1_{\mathsf{pp}})$ and a bit b, and does the following:

- For all $i \in [2]$, compute $\mathsf{pp}^1_i = \mathsf{OutParam}(\mathsf{pp}^0_i)$. For all $i \in [2]$, $d \in \{0,1\}$, choose at random o^d_i and compute $\mathbf{c}^d_i = \mathsf{C.Commit}(\mathsf{pp}^d_i, b; o^d_i)$. Denote by $\sigma_c = (\mathbf{c}^0_1, \mathbf{c}^0_2, \mathbf{c}^1_1, \mathbf{c}^1_2)$ and $\mathsf{st} = (o^0_1, o^0_2, o^1_1, o^1_2)$.
- Compute $\mathsf{pp}^0_* = \mathsf{InterParam}(\mathsf{pp}^0_1, \mathsf{pp}^0_2, r^1_{\mathsf{pp}})$ and $\mathsf{pp}^1_* = \mathsf{InterParam}(\mathsf{pp}^1_1, \mathsf{pp}^1_2, r^1_{\mathsf{pp}})$. For all $b_1, b_2 \in \{0,1\}$, compute

$$\pi^{b_1 b_2} \leftarrow \mathsf{TC.Prove}\big((\mathbf{c}^{b_1}_1, \mathbf{c}^{b_2}_2, \mathsf{pp}^{b_1}_1, \mathsf{pp}^{b_2}_2), (b, \mathsf{pp}^{b_1}_*, o^{b_1}_1, o^{b_2}_2)\big).$$

  Denote by $\sigma_\pi = (\pi^{00}, \pi^{01}, \pi^{10}, \pi^{11})$. Output $(\sigma_c, \sigma_\pi, \mathsf{st})$.

[5] Recall that for parameters $\mathsf{pp}, \mathsf{pp}'$ such that $\mathsf{pp}' = \mathsf{OutParam}(\mathsf{pp})$, pp itself is an intermediate parameter between $\mathsf{pp}, \mathsf{pp}'$; see Remark 2 (Section 5.1.2).

$\{0,1\} \leftarrow \mathsf{VerCom}\big(\mathsf{pp}_1^0, \mathsf{pp}_2^0, \sigma_c, \sigma_\pi\big)$: The verification algorithm takes as input two pairs of parameters, four commitments σ_c and four proofs σ_π, and outputs 1 if and only if all the following checks are successful:

For every $i \in [2]$, compute $\mathsf{pp}_i^1 = \mathsf{OutParam}(\mathsf{pp}_i^0)$. Parse $\sigma_c = (\mathbf{c}_1^0, \mathbf{c}_2^0, \mathbf{c}_1^1, \mathbf{c}_2^1)$ and $\sigma_\pi = (\pi^{00}, \pi^{01}, \pi^{10}, \pi^{11})$. For all $b_1, b_2 \in \{0,1\}$, check that

$$\mathsf{TC.Verify}\big((\mathbf{c}_1^{b_1}, \mathbf{c}_2^{b_2}, \mathsf{pp}_1^{b_1}, \mathsf{pp}_2^{b_2}), \pi^{b_1 b_2}\big) = 1.$$

$\big(\sigma_c', \sigma_\pi', \mathsf{st}'\big) \leftarrow \mathsf{RCom}\big(\{\mathsf{pp}_i^0, (\mathsf{pp}_i^0)', r_{\mathsf{pp}}^i\}_{i\in[2]}, \sigma_c, \sigma_\pi\big)$: The RCom algorithm takes as input two parameters $\mathsf{pp}_1^0, \mathsf{pp}_2^0$, randomized parameters $(\mathsf{pp}_1^0)', (\mathsf{pp}_2^0)'$, randomness $r_{\mathsf{pp}}^1, r_{\mathsf{pp}}^2$ used in randomizing the parameters $\mathsf{pp}_1^0, \mathsf{pp}_2^0$, commitments σ_c and proofs σ_π, and does the following:

- For all $i \in [2]$, check that $(\mathsf{pp}_i^0)' = \mathsf{RParam}(\mathsf{pp}_i^0; r_{\mathsf{pp}}^i)$ and compute $(\mathsf{pp}_i^1)' = \mathsf{OutParam}(\ (\mathsf{pp}_i^0)'\ )$.
- Parse $\sigma_c = (\mathbf{c}_1^0, \mathbf{c}_2^0, \mathbf{c}_1^1, \mathbf{c}_2^1)$. For all $i \in [2]$ and $d \in \{0,1\}$, compute $z_i^d = \mathsf{ChangeCom}(\mathsf{pp}_i^d, \mathbf{c}_i^d, r_{\mathsf{pp}}^i)$, choose randomness o_i^d and compute $(\mathbf{c}_i^d)' = \mathsf{C.Rand}(\mathsf{pp}_i^{d'}, z_i^d; o_i^d)$. Denote by $\sigma_c' = \big((\mathbf{c}_1^0)', (\mathbf{c}_2^0)', (\mathbf{c}_1^1)', (\mathbf{c}_2^1)'\big)$ and $\mathsf{st}' = (o_1^0, o_2^0, o_1^1, o_2^1)$.
- Parse $\sigma_\pi = (\pi^{00}, \pi^{01}, \pi^{10}, \pi^{11})$. For all $b_1, b_2 \in \{0,1\}$, compute

$$(\pi^{b_1 b_2})' \leftarrow \mathsf{TC.Maul}\big((\mathbf{c}_1^{b_1}, \mathbf{c}_2^{b_2}, \mathsf{pp}_1^{b_1}, \mathsf{pp}_2^{b_2}), (r_{\mathsf{pp}}^1, r_{\mathsf{pp}}^2, o_1^{b_1}, o_2^{b_2}), \pi^{b_1 b_2}\big).$$

  Denote by $\sigma_\pi' = \big((\pi^{00})', (\pi^{01})', (\pi^{10})', (\pi^{11})'\big)$. Output $(\sigma_c', \sigma_\pi', \mathsf{st}')$.

In Section 7.3, we construct the proof system ($\mathsf{TC.Prove}, \mathsf{TC.Verify}, \mathsf{TC.Maul}$), prove weak soundness, and prove the strong secrecy property assuming *DLIN with Leakage* as described in Section 5.3.

The **<u>third</u>** ***ingredient in our FH NIWI construction*** is a malleable NIWI proof system ($\mathsf{Bit.Prove}$, $\mathsf{Bit.Verify}, \mathsf{Bit.GenMaul}$) for $L_{\mathsf{com}}[\mathsf{pp}]$. Recall that $L_{\mathsf{com}}[\mathsf{pp}]$ is parameterized by $\mathsf{pp} \leftarrow \mathsf{C.Setup}(1^\lambda)$ and is defined as follows:

$$L_{\mathsf{com}}[\mathsf{pp}] = \{\mathbf{c} \mid \exists\ (b, o) \text{ s.t. } \mathbf{c} = \mathsf{C.Commit}(\mathsf{pp}, b; o)\ \wedge\ b \in \{0,1\}\}$$

In Section 6.1, we described a malleable NIWI ($\mathsf{Bit.Prove}, \mathsf{Bit.Verify}, \mathsf{Bit.Maul}$) for $L_{\mathsf{com}}[\mathsf{pp}]$ with respect to transformation $\mathsf{Bit.T}$. For the FH NIWI construction, we need the malleability to be with respect to a more general transformation $\mathsf{Bit.GenT} = (\mathsf{Bit.GenTrans}, \mathsf{Bit.GWitTrans})$. The transformation $\mathsf{Bit.GenTrans}$ randomizes and transforms a commitment $\mathbf{c} \in L_{\mathsf{com}}[\mathsf{pp}]$ to a new commitment $\mathbf{c}' \in L_{\mathsf{com}}[\mathsf{pp}']$. Formally, $\mathsf{Bit.GenTrans}(\mathbf{c}; o, r_{\mathsf{pp}})$ works as follows:

1. Compute $\mathsf{pp}' = \mathsf{RParam}(\mathsf{pp}, r_{\mathsf{pp}})$ and output $z = \mathsf{ChangeCom}(\mathsf{pp}', \mathbf{c}, r_{\mathsf{pp}})$.
2. Compute $\mathbf{c}' = \mathsf{C.Rand}(\mathsf{pp}, z; o')$. Output $\mathbf{c}'$.

Recall that the syntax of the associated mauling algorithm is as follows:

$$\pi_{\mathsf{bit}}' \leftarrow \mathsf{Bit.GenMaul}(\mathsf{pp}, \mathbf{c}, o', r_{\mathsf{pp}}, \pi_{\mathsf{bit}})$$

The **<u>fourth</u>** ***ingredient in our FH NIWI construction*** is the NIWI proof system ($\mathsf{N.Prove}, \mathsf{N.Verify}$, $\mathsf{N.GenMaul}$) for $L_{\mathsf{N}}[\mathsf{pp}]$. Recall that $L_{\mathsf{N}}[\mathsf{pp}]$ is parameterized by $\mathsf{pp} \leftarrow \mathsf{C.Setup}(1^\lambda)$ and is defined as follows:

$$L_{\mathsf{N}}[\mathsf{pp}] = \big\{\{\mathbf{c}_i\}_{i\in[3]} \mid \exists\ \{b_i, o_i\}_{i\in[3]} \text{ s.t. } \mathbf{c}_i = \mathsf{C.Commit}(b_i; o_i)\ \wedge\ (b_3 = b_1 \bar{\wedge} b_2)\ \wedge\ \{b_i \in \{0,1\}\}_{i\in[3]}\big\}$$

In Section 6.1, we described a malleable NIWI ($\mathsf{N.Prove}, \mathsf{N.Verify}, \mathsf{N.Maul}$) for $L_{\mathsf{N}}[\mathsf{pp}]$ with respect to transformation $\mathsf{N.T}$. For the FH NIWI construction, we need the malleability to be with respect to a more general transformation $\mathsf{N.GenT} = (\mathsf{N.GenTrans}, \mathsf{N.GWitTrans})$. The transformation $\mathsf{N.GenTrans}(\mathsf{pp}, \{\mathbf{c}_i\}_{i\in[3]}$, $\{o_i'\}_{i\in[3]}, r_{\mathsf{pp}})$ works as follows:

1. Compute $\mathsf{pp}' = \mathsf{RParam}(\mathsf{pp}, r_{\mathsf{pp}})$.
2. For every $i \in [3]$, output $z_i = \mathsf{ChangeCom}(\mathsf{pp}', \mathbf{c}_i, r_{\mathsf{pp}})$.
3. Compute $\mathbf{c}'_i = \mathsf{C.Rand}(z_i, o'_i)$ for $i \in [3]$. Output $(\mathbf{c}'_1, \mathbf{c}'_2, \mathbf{c}'_3)$

Recall that the syntax of the associated mauling algorithm is as follows:

$$\pi'_{\mathsf{N}} \leftarrow \mathsf{N.GenMaul}(\mathsf{pp}, \{\mathbf{c}_i\}_{i\in[3]}, \{o'_i\}_{i\in[3]}, r_{\mathsf{pp}}, \pi_{\mathsf{N}})$$

Note that the third and fourth ingredients can be instantiated similar to the instantiations described in Section 6.3, again using the malleable NIWI proof system $(\mathsf{Lin.Prove}, \mathsf{Lin.Verify}, \mathsf{Lin.Maul})$ for $L_{\mathsf{Lin}}[\mathsf{pp}]$ with respect to transformation $\mathsf{Lin.T}$.

7.2 FH NIWI Construction

In this section we construct a FH NIWI for the language $L_{\mathcal{U}}$. Recall that

$$L_{\mathcal{U}} = \{(C, \mathsf{out}) \mid \exists\ \mathbf{w} \text{ such that } C(\mathbf{w}) = \mathsf{out}\}.$$

We start by defining a *connecting wire* for a circuit C.

Definition 14 (Connecting Wire)**.** A wire k in a circuit C is said to be a *connecting wire* for a pair of NAND gates (i, j) if it is an output wire of gate i and an input wire to gate j.

Without loss of generality, we assume that every circuit C in an $L_{\mathcal{U}}$ instance (C, out) is a layered circuit; namely, the circuit consists of t layers of gates such that any output wire from a gate at layer $i \in [t]$ is an input wire to a gate in layer $i + 1$.[6] We also assume without loss of generality that the circuit consists of NAND gates where each gate has fan-in 2 and fan-out at most 2.

We will use the following ingredients in our FH NIWI construction:

– A RaHE-commitment scheme $(\mathsf{C.Setup}, \mathsf{C.Commit}, \mathsf{C.Rand})$ with the additional functionalities

$$(\mathsf{OutParam}, \mathsf{ValidParam}, \mathsf{RParam}, \mathsf{ChangeCom}, \mathsf{ValidInter}, \mathsf{InterParam})$$

as defined in Section 5.1.

– Malleable proof system for L_{TC} with weak soundness and strong secrecy, with respect to the transformation $\mathsf{TC.T} = (\mathsf{TC.Transform}, \mathsf{TC.WitTrans})$ as described in Section 7.1, denoted by

$$(\mathsf{TC.Prove}, \mathsf{TC.Verify}, \mathsf{TC.Maul}).$$

– Malleable NIWI proof system for $L_{\mathsf{com}}[\mathsf{pp}]$ with respect to the transformation $\mathsf{Bit.GenT} = (\mathsf{Bit.GenTrans}, \mathsf{Bit.GWitTrans})$ as described in Section 7.1, denoted by

$$(\mathsf{Bit.Prove}, \mathsf{Bit.Verify}, \mathsf{Bit.GenMaul}).$$

– Malleable NIWI proof system for $L_{\mathsf{N}}[\mathsf{pp}]$ with respect to the transformation $\mathsf{N.GenT} = (\mathsf{N.GenTrans}, \mathsf{N.GWitTrans})$ as described in Section 7.1, denoted by

$$(\mathsf{N.Prove}, \mathsf{N.Verify}, \mathsf{N.GenMaul}).$$

[6]Any circuit can be converted into a layered circuit by adding dummy gates.

Theorem 4. *Assuming the existence of the ingredients as described above, the following construction* Π_{FHNIWI} *is a Fully Homomorphic NIWI proof system as per Definition 8.*

We instantiate the first, third and fourth ingredients from DLIN and instantiate the second ingredient from DLIN with Leakage as we describe in Section 7.3. This gives the following corollary:

Corollary 1. *Assuming DLIN with Leakage, the following construction* Π_{FHNIWI} *is a Fully Homomorphic NIWI proof system as per Definition 8.*

Construction: We now describe our construction of fully homomorphic NIWI proofs for $L_{\mathcal{U}}$.

$\mathsf{NIWI.Prove}((C, \mathsf{out}), \mathbf{w})$: For $C : \{0,1\}^n \to \{0,1\}$, denote by m the number of NAND gates, and by ℓ the number of connecting wires. Let $\mathbf{w} = w_1, \ldots, w_{n+\ell}$ be the values induced by $\mathbf{w}$ on all the wires excluding the output wire (but including the input wires).

1. For each gate $j \in [m]$, choose randomness r^j_{pp} and compute $\mathsf{pp}^0_j = \mathsf{C.Setup}(1^\lambda; r^j_{\mathsf{pp}})$ and $\mathsf{pp}^1_j = \mathsf{OutParam}(\mathsf{pp}^1_j)$. Denote by $\overrightarrow{\mathsf{pp}}_j = (\mathsf{pp}^0_j, \mathsf{pp}^1_j)$.

2. For each input wire $k \in [n]$, denote by j the gate for which wire k is an input. For every $b \in \{0,1\}$, choose at random $o^b_{k,j}$ and compute $\mathbf{c}^b_{k,j} = \mathsf{C.Commit}(\mathsf{pp}^b_j, w_k; o^b_{k,j})$. Let $\mathbf{c}_k = (\mathbf{c}^0_{k,j}, \mathbf{c}^1_{k,j})$.

   For the output wire wout (which is the output wire of gate m) and for every $b \in \{0,1\}$, let $\mathbf{c}^b_{\mathsf{wout},m} = \mathbf{1}$ for $\mathsf{out} = 1$ and let $\mathbf{c}^b_{\mathsf{wout},m} = \mathbf{0}$ for $\mathsf{out} = 0$. Recall that $\mathbf{1}, \mathbf{0}$ are the canonical commitments to 1 and 0 respectively (see Remark 1). Let $\mathbf{c}_{\mathsf{wout}} = (\mathbf{c}^0_{\mathsf{wout},m}, \mathbf{c}^1_{\mathsf{wout},m})$.

3. For each connecting wire $k \in \{n+1, \ldots, n+\ell\}$ that connects gates $i, j \in [m]$ $(i < j)$, compute

   $$\left(\sigma^k_c, \sigma^k_\pi, \mathsf{st}^k\right) \leftarrow \mathsf{OutCom}(\mathsf{pp}^0_i, \mathsf{pp}^0_j, r^j_{\mathsf{pp}}, w_k)$$

   where $\sigma^k_c = (\mathbf{c}^0_{k,i}, \mathbf{c}^0_{k,j}, \mathbf{c}^1_{k,i}, \mathbf{c}^1_{k,j})$, $\sigma^k_\pi = (\pi^{00}_k, \pi^{01}_k, \pi^{10}_k, \pi^{11}_k)$ and where $\mathsf{st}^k = (o^0_{k,i}, o^0_{k,j}, o^1_{k,i}, o^1_{k,j})$. We will denote $(\sigma^k_c, \sigma^k_\pi)$ by Φ_k.

   Note that a commitment $\mathbf{c}^b_{k,j}$ commits to w_k with respect to parameters pp^b_j, and where wire k is either an input or output wire for gate j. Also note that there are two commitments for each input wire and four commitments for each connecting wire.

4. Denote by S the set of all pairs (k, j) for $k \in [n+\ell], j \in [m]$, such that wire k is an input or output to gate j. For all $(k, j) \in S$ and for every $b \in \{0,1\}$, generate a proof that the commitment $\mathbf{c}^b_{k,j}$ commits to a bit. Namely, compute

   $$\pi_{\mathsf{bit}}[k, j]^b \leftarrow \mathsf{Bit.Prove}(\mathsf{pp}^b_j, \mathbf{c}^b_{k,j}, o^b_{k,j})$$

   where $o^b_{k,j}$ is the opening for commitment $\mathbf{c}^b_{k,j}$ as computed in step 2 (for input wires) or as part of st^k output by OutCom (for connecting wires) in step 3. Let $\pi_{\mathsf{bit}}[k, j] = \left(\pi_{\mathsf{bit}}[k, j]^0, \pi_{\mathsf{bit}}[k, j]^1\right)$.

5. For each gate $j \in [m]$, denote by k_1, k_2 the input wires of the gate j and by k_3, k_4 the output wires to gate j. For each $t \in \{3,4\}$ and $b \in \{0,1\}$, compute a gate consistency proof as follows:

   $$\pi^{j,b}_{\mathsf{gate}}[t] \leftarrow \mathsf{N.Prove}\big(\mathsf{pp}^b_j, \{\mathbf{c}^b_{k_i,j}\}_{i \in \{1,2,t\}}, \{w^b_{k_i}, o^b_{k_i,j}\}_{i \in \{1,2,t\}}\big)$$

   Let $\pi^j_{\mathsf{gate}} = \left(\pi^{j,0}_{\mathsf{gate}}[3], \pi^{j,1}_{\mathsf{gate}}[3], \pi^{j,0}_{\mathsf{gate}}[4], \pi^{j,1}_{\mathsf{gate}}[4]\right)$.

6. Finally output

   $$\Pi_{\mathsf{NIWI}} = \left[\{\overrightarrow{\mathsf{pp}}_j\}_{j \in [m]}, \{\mathbf{c}_k\}_{k \in [n]}, \{\Phi_k\}_{k \in [\ell]}, \left\{\pi_{\mathsf{bit}}[i, j]\right\}_{(i,j) \in S}, \{\pi^j_{\mathsf{gate}}\}_{j \in [m]}, \mathbf{c}_{\mathsf{wout}}\right].$$

$\mathsf{NIWI.Verify}((C, \mathsf{out}), \Pi)$: Parse

$$\Pi_{\mathsf{NIWI}} = \left[\{\overrightarrow{\mathsf{pp}}_j\}_{j\in[m]}, \{\mathbf{c}_k\}_{k\in[n]}, \{\Phi_k\}_{k\in[\ell]}, \left\{\pi_{\mathsf{bit}}[i,j]\right\}_{(i,j)\in S}, \{\pi^j_{\mathsf{gate}}\}_{j\in[m]}, \mathbf{c}_{\mathsf{wout}}\right]$$

1. For each $j \in [m]$, parse $\overrightarrow{\mathsf{pp}}_j = (\mathsf{pp}^0_j, \mathsf{pp}^1_j)$ and check that $\mathsf{ValidParam}(\mathsf{pp}^0_j, \mathsf{pp}^1_j) = 1$.
2. For each connecting wire k that connects gates $i, j \in [m]$, check that $\mathsf{VerCom}(\mathsf{pp}^0_i, \mathsf{pp}^0_j, \Phi_k) = 1$.
3. For each $(k, j) \in S$, parse $\pi_{\mathsf{bit}}[k,j] = \left(\pi_{\mathsf{bit}}[k,j]^0, \pi_{\mathsf{bit}}[k,j]^1\right)$ and for every $b \in \{0,1\}$, check that $\mathsf{Bit.Verify}(\mathsf{pp}^b_j, \mathbf{c}^b_{k,j}, \pi_{\mathsf{bit}}[k,j]^b) = 1$.
4. For each gate $j \in [m]$, parse $\pi^j_{\mathsf{gate}} = \left(\pi^{j,0}_{\mathsf{gate}}[3], \pi^{j,1}_{\mathsf{gate}}[3], \pi^{j,0}_{\mathsf{gate}}[4], \pi^{j,1}_{\mathsf{gate}}[4]\right)$. Denote by k_1, k_2 the input wires to gate j and by k_3, k_4 the output wires of gate j. For each $t \in \{3,4\}$ and $b \in \{0,1\}$, check that $\mathsf{N.Verify}(\mathsf{pp}^b_j, \{\mathbf{c}^b_{k_i,j}\}_{i\in\{1,2,t\}}, \pi^{j,b}_{\mathsf{gate}}[t]) = 1$ where $\pi^j_{\mathsf{gate}} = \left(\pi^{j,0}_{\mathsf{gate}}[3], \pi^{j,1}_{\mathsf{gate}}[3], \pi^{j,0}_{\mathsf{gate}}[4], \pi^{j,1}_{\mathsf{gate}}[4]\right)$.
5. Parse $\mathbf{c}_{\mathsf{wout}} = (\mathbf{c}^0_{\mathsf{wout},m}, \mathbf{c}^1_{\mathsf{wout},m})$ and check that for all $b \in \{0,1\}$, $\mathbf{c}^b_{\mathsf{wout},m} = \mathbf{1}$ if $\mathsf{out} = 1$ and $\mathbf{c}^b_{\mathsf{wout},m} = \mathbf{0}$ if $\mathsf{out} = 0$.

$\mathsf{NIWI.Rand}((C, \mathsf{out}), \Pi)$: Parse

$$\Pi_{\mathsf{NIWI}} = \left[\{\overrightarrow{\mathsf{pp}}_j\}_{j\in[m]}, \{\mathbf{c}_k\}_{k\in[n]}, \{\Phi_k\}_{k\in[\ell]}, \left\{\pi_{\mathsf{bit}}[i,j]\right\}_{(i,j)\in S}, \{\pi^j_{\mathsf{gate}}\}_{j\in[m]}, \mathbf{c}_{\mathsf{wout}}\right]$$

Randomize the proof Π as follows:

1. For each $j \in [m]$, parse $\overrightarrow{\mathsf{pp}}_j = (\mathsf{pp}^0_j, \mathsf{pp}^1_j)$, choose randomness r^j_{pp} and compute $(\mathsf{pp}^0_j)' \leftarrow \mathsf{RParam}(\mathsf{pp}^0_j; r^j_{\mathsf{pp}})$ and $(\mathsf{pp}^1_j)' = \mathsf{OutParam}(\mathsf{pp}^{0'}_j)$.
2. For each $k \in [n]$, denote by j the gate for which wire k is an input. For each $b \in \{0,1\}$,
   - Compute $z^b_{k,j} = \mathsf{ChangeCom}((\mathsf{pp}^b_j)', \mathbf{c}^b_{k,j}, r^j_{\mathsf{pp}})$.
   - Choose fresh randomness $o^b_{k,j}$ and compute $(\mathbf{c}^b_{k,j})' = \mathsf{C.Rand}(\mathsf{pp}^b_j, z^b_{k,j}; o^b_{k,j})$.

   Let $\mathbf{c}'_k = \left((\mathbf{c}^0_{k,j})', (\mathbf{c}^1_{k,j})'\right)$.
3. For each connecting wire k between gates $i, j \in [m]$, compute
   $$\left((\sigma^k_c)', (\sigma^k_\pi)', \mathsf{st}'_k\right) \leftarrow \mathsf{RCom}(\mathsf{pp}^0_i, (\mathsf{pp}^0_i)', r^i_{\mathsf{pp}}, \mathsf{pp}^0_j, (\mathsf{pp}^0_j)', r^j_{\mathsf{pp}}, \sigma^k_c, \sigma^k_\pi)$$
   Let $\mathsf{st}'_k = (o^0_{k,i}, o^0_{k,j}, o^1_{k,i}, o^1_{k,j})$. Denote by $\phi'_k = \left((\sigma^k_c)', (\sigma^k_\pi)'\right)$.
4. For all $(k, j) \in S$, for every $b \in \{0,1\}$, compute
   $$(\pi_{\mathsf{bit}}[k,j]^b)' \leftarrow \mathsf{Bit.GenMaul}(\mathsf{pp}^b_j, \mathbf{c}^b_{k,j}, o^b_{k,j}, r^j_{\mathsf{pp}}, \pi_{\mathsf{bit}}[k,j]^b)$$
   where $o^b_{k,j}$ is the randomizing factor for commitment $\mathbf{c}^b_i$ as computed in the step 2 (for input wires) or as part of st'_k output by RCom (for connecting wires) computed in step 3. Let $\pi_{\mathsf{bit}}[k,j]' = \left[(\pi_{\mathsf{bit}}[k,j]^0)', (\pi_{\mathsf{bit}}[k,j]^1)'\right]$.

5. For each gate $j \in [m]$, let k_1, k_2 be the input wires to gate j, and k_3, k_4 be the output wires. For each $t \in \{3,4\}$ and for every $b \in \{0,1\}$, compute

$$(\pi^{j,b}_{\mathsf{gate}}[t])' \leftarrow \mathsf{N.GenMaul}(\mathsf{pp}^b_j, \{\mathbf{c}^b_{k_i,j}\}_{i\in\{1,2,t\}}, \{o^b_{k_i,j}\}_{i\in\{1,2,t\}}, r^j_{\mathsf{pp}}, \pi^{j,b}_{\mathsf{gate}}[t])$$

and where $o^b_{k_i,j}$ is the randomizing factor for commitment $\mathbf{c}^b_{k_i,j}$ as computed in the step 2 (for input wires) or as part of st'_k output by RCom (for connecting wires) computed in step 3. Let $(\pi^j_{\mathsf{gate}})' = \big((\pi^{j,0}_{\mathsf{gate}}[3])', (\pi^{j,1}_{\mathsf{gate}}[3])', (\pi^{j,0}_{\mathsf{gate}}[4])', (\pi^{j,1}_{\mathsf{gate}}[4])'\big)$

Finally output randomized proof as:

$$\Pi'_{\mathsf{NIWI}} = \big[\{\overrightarrow{\mathsf{pp}}'_j\}_{j\in[m]}, \{\mathbf{c}'_i\}_{i\in[n]}, \{\Phi'_k\}_{k\in[\ell]}, \big\{\pi_{\mathsf{bit}}[k,j]'\big\}_{(k,j)\in S}, \{(\pi^j_{\mathsf{gate}})'\}_{j\in[m]}, \mathbf{c}_{\mathsf{wout}}\big].$$

$\mathsf{NIWI.Eval}(\{(C_q, b_q, \Pi_q)\}^K_{q=1}, C')$:

1. Check that $\mathsf{Valid}(C') = 1$, else output $\perp$. Recall that $\mathsf{Valid}(C') = 1$ if and only if $C' : \{0,1\}^k \to \{0,1\}$. Compute $(C, \mathsf{out}') = \mathsf{Compose}(\{(C_q, b_q, \Pi_q)\}^K_{i=q}, C')$.
2. For each $q \in [K]$, let $\pi^{\mathsf{out}}_{\mathsf{gate},q} \in \Pi_q$ be the gate consistency proofs of the output gate q_{out} of circuit C_q and let $\mathbf{c}_{\mathsf{wout}q} \in \Pi_q$ be the canonical output commitments to output b_q of circuit C_q. Let $\widehat{\Pi_q} = \Pi_q \setminus \{\pi^{\mathsf{out}}_{\mathsf{gate},q}, \mathbf{c}_{\mathsf{wout}q}\}$.
3. Compute a proof $\widehat{\Pi'}$ for C' with witness $(b_1, \ldots, b_K)$ by evaluating $\mathsf{NIWI.Prove}((C', \mathsf{out}'), (b_1, \ldots, b_K))$ (where out' is from Step 1) with the following modification:

   For each $q \in [K]$, denote by q_{out} the output gate of C_q, by W_q the output wire and by q_{in} the gate to which wire W_q is an input in the composed circuit C.

   – For every $q \in [K]$, choose randomness $r_{q_{\mathsf{in}}}$ and compute $\mathsf{pp}^0_{q_{\mathsf{in}}} = \mathsf{C.Setup}(1^\lambda; r_{q_{\mathsf{in}}})$ and $\mathsf{pp}^1_{q_{\mathsf{in}}} = \mathsf{OutParam}(\mathsf{pp}^0_{q_{\mathsf{in}}})$.
   – Instead of fresh commitment for the wires $W_q \in [K]$, compute

   $$\big(\sigma^q_c, \sigma^q_\pi, \mathsf{st}^q\big) \leftarrow \mathsf{OutCom}(\mathsf{pp}^0_{q_{\mathsf{out}}}, \mathsf{pp}^0_{q_{\mathsf{in}}}, r_{q_{\mathsf{in}}}, b_q)$$

4. For each output gate q_{out} for C_q and for $b \in \{0,1\}$, let $\mathbf{c}^b_{1,q_{\mathsf{out}}}, \mathbf{c}^b_{2,q_{\mathsf{out}}}$ be the commitments to the input wire values to the gate q_{out}. Let o^b_q be the randomness used to commit to b_q with respect to $\mathsf{pp}^b_{q_{\mathsf{out}}}$, computed as part of st^q in step 3. Recall $\pi^{\mathsf{out}}_{\mathsf{gate},q} = (\pi^{\mathsf{out},0}_{\mathsf{gate},q}, \pi^{\mathsf{out},1}_{\mathsf{gate},q})$ and $\mathbf{c}_{\mathsf{wout}q} = (\mathbf{c}^0_{W_q,q_{\mathsf{out}}}, \mathbf{c}^1_{W_q,q_{\mathsf{out}}})$. For every $b \in \{0,1\}$, compute

   $$(\pi^{\mathsf{out},b}_{\mathsf{gate},q})' \leftarrow \mathsf{N.GenMaul}(\mathsf{pp}^b_{q_{\mathsf{out}}}, (\mathbf{c}^b_{1,q_{\mathsf{out}}}, \mathbf{c}^b_{2,q_{\mathsf{out}}}, \mathbf{c}^b_{W_q,q_{\mathsf{out}}}), (1,1,o^b_q), 1, \pi^{\mathsf{out},b}_{\mathsf{gate},q})$$

5. Note that $\Pi = \big[\widehat{\Pi}_1, \ldots, \widehat{\Pi}_n, \widehat{\Pi'}, \big\{(\pi^{\mathsf{out},b}_{\mathsf{gate},q})'\big\}_{i\in[K],b\in\{0,1\}}\big]$ is a complete proof for (C, out). Randomize the proof by computing $\Pi' \leftarrow \mathsf{NIWI.Rand}((C, \mathsf{out}'), \Pi)$. Finally output (C, out', Π').

Proof of Theorem 4: Completeness follows from the completeness of underlying primitives.

Claim 4. Π_{FHNIWI} *satisfies perfect soundness.*

Proof. Suppose for contradiction, there exists P^* and an infinite set $\Lambda \subseteq \mathbb{N}$ such that for all $\lambda \in \Lambda$,

$$\Pr[((C, \mathsf{out}), \Pi) \leftarrow P^*(1^\lambda) \ : \ \mathsf{NIWI.Verify}((C, \mathsf{out}), \Pi) = 1 \ \wedge \ (C, \mathsf{out}) \notin L_{\mathcal{U}}] > 0. \tag{4}$$

For $C : \{0,1\}^n \to \{0,1\}$, denote by m the number of NAND gates, by ℓ the number of connecting wires, and by S the set of all pairs (k, j) for $k \in [n+\ell], j \in [m]$, such that wire k is an input or output to gate j. Parse $\Pi = \left[\{\overrightarrow{\mathsf{pp}}_j\}_{j\in[m]}, \{\mathbf{c}_k\}_{k\in[n]}, \{\Phi_k\}_{k\in[\ell]}, \{\pi_{\mathsf{bit}}[i,j]\}_{(i,j)\in S}, \{\pi^j_{\mathsf{gate}}\}_{j\in[m]}, \mathbf{c}_{\mathsf{wout}}\right]$.

Let E_1 be the event that $\mathsf{NIWI.Verify}((C, \mathsf{out}), \Pi) = 1$ where $((C, \mathsf{out}), \Pi) \leftarrow P^*(1^\lambda)$. Let E_2 be the event that $(C, \mathsf{out}) \notin L_{\mathcal{U}}$ where $((C, \mathsf{out}), \Pi) \leftarrow P^*(1^\lambda)$.

– E_1 implies that for all $j \in [m]$, $\mathsf{ValidParam}(\overrightarrow{\mathsf{pp}}_j) = 1$. Namely, for all $j \in [m]$, there exists $b_j \in \{0,1\}$ such that $\mathsf{pp}_j^{b_j} \in \mathsf{C.Setup}(1^\lambda)$. Let $\{\widehat{\mathsf{pp}}_j\}_{j\in[m]}$ be the set of all binding parameters such that for each $j \in [m]$, $\widehat{\mathsf{pp}}_j = \mathsf{pp}_j^{b_j}$.

– Let $\mathbf{w} = (w_1, \dots, w_{n+\ell})$ be the values committed with respect to $\{\widehat{\mathsf{pp}}_j\}_{j\in[m]}$ on all the wires excluding the output wire (but including the input wires). E_1 implies that for all $(k, j) \in S$, $\mathsf{Bit.Verify}(\widehat{\mathsf{pp}}_j, \mathbf{c}_{k,j}, \pi_{\mathsf{bit}}[k,j]) = 1$ where $\mathbf{c}_{k,j}$ is the commitment to wire $k \in [n+\ell]$ with respect to $\widehat{\mathsf{pp}}_j$ and $\pi_{\mathsf{bit}}[k,j]$ is the corresponding $L_{\mathsf{com}}[\widehat{\mathsf{pp}}_j]$ proof. This along with perfect soundness of $(\mathsf{Bit.Prove}, \mathsf{Bit.Verify})$, implies that for all $k \in [n+\ell]$, $w_k \in \{0,1\}$.

– E_1 implies that for any connecting wire $k \in [\ell]$ between gates i, j, $\mathsf{VerCom}\big(\widehat{\mathsf{pp}}_i, \widehat{\mathsf{pp}}_j, \Phi_k\big) = 1$. This in turn implies that $\mathsf{TC.Verify}((\widehat{\mathsf{pp}}_i, \widehat{\mathsf{pp}}_j, \mathbf{c}_{k,i}, \mathbf{c}_{k,j}), \pi_{\mathsf{TC}})$ where $\mathbf{c}_{k,i}, \mathbf{c}_{k,j}$ are commitments to wire value k under $\widehat{\mathsf{pp}}_i, \widehat{\mathsf{pp}}_j$ respectively and π_{TC} is the corresponding L_{TC} proof. This along with the soundness of $(\mathsf{TC.Prove}, \mathsf{TC.Verify})$ implies that $\mathbf{c}_{k,i}, \mathbf{c}_{k,j}$ commit to the same value w_k.

– For any gate $j \in [m]$, let k_{j_1}, k_{j_2} be the input wires and k_{j_3} be the output wire. E_1 implies that for all $j \in [m]$, $\mathsf{N.Verify}(\widehat{\mathsf{pp}}_j, \widehat{\mathsf{pp}}_j, \{\mathbf{c}_{k_{j_i},j}\}_{i\in[3]}, \pi^j_{\mathsf{gate}}) = 1) = 1$ where $\{\mathbf{c}_{k_{j_i},j}\}_{i\in[3]}$ are the commitment to wires $k_{j_1}, k_{j_2}, k_{j_3}$ with respect to $\widehat{\mathsf{pp}}_j$ and π^j_{gate} is the corresponding $L_{\mathsf{N}}[\widehat{\mathsf{pp}}_j]$ proof. This along with perfect soundness of $(\mathsf{N.Prove}, \mathsf{N.Verify})$, implies that for all $j \in [m]$, $w_{k_{j_1}} \barwedge w_{k_{j_2}} = w_{k_{j_3}}$. In particular, $w_{n+\ell-1} \barwedge w_{n+\ell} = \mathsf{out}$ where $w_{n+\ell-1}, w_{n+\ell}$ are the values of the two input wires to the output gate of C.

Thus, E_1 implies that $\mathbf{w} = (w_1, \dots, w_{n+\ell})$ defines a consistent boolean assignment across the entire circuit C such that $C(\mathbf{w}) = \mathsf{out}$. However E_2 implies that that $C(\mathbf{w}) \neq \mathsf{out}$. Hence we contradict Equation (4) which says that $\Pr[E_1 \ \wedge \ E_2] > 0$. □

Claim 5. Π_{FHNIWI} *is a randomizable non-interactive proof system as per Definition 4.*

Proof. We show that for any instance $(C, b) \in L_{\mathcal{U}}$ with witness $w \in \{0,1\}^n$ and any proof Π such that $\mathsf{NIWI.Verify}((C, b), \Pi) = 1$, the following distributions are identical:

$$\big\{(C, b), w, \Pi, \mathbf{R}, \Pi_f\big\} \text{ and } \big\{(C, b), w, \Pi, \mathbf{R}, \Pi'\big\}$$

where Π_f is a fresh proof obtained by $\Pi_f \leftarrow \mathsf{NIWI.Prove}((C, b), w)$, Π' is a randomized proof obtained by $\Pi' \leftarrow \mathsf{NIWI.Rand}((C, b), \Pi)$ and $\mathbf{R}$ is randomness such that $\Pi = \mathsf{NIWI.Prove}((C, b), w; \mathbf{R})$. Let $\mathbf{w} = w_1, \dots, w_{n+\ell}$ be the values induced by $\mathbf{w}$ on all the wires excluding the output wire (but including the input wires). Parse

$$\Pi = \left[\{\overrightarrow{\mathsf{pp}}_j\}_{j\in[m]}, \{\mathbf{c}_k\}_{k\in[n]}, \{\Phi_k\}_{k\in[\ell]}, \{\pi_{\mathsf{bit}}[i,j]\}_{(i,j)\in S}, \{\pi^j_{\mathsf{gate}}\}_{j\in[m]}, \mathbf{c}_{\mathsf{wout}}\right]$$

and parse

$$\mathbf{R} = \left[\{r^j_{\mathsf{pp}}, s_j\}_{j\in[m]}, \{S_k\}_{k\in[\ell]}, \{o_i, t_i\}_{i\in[n+2\ell]}\right]$$

where $\mathsf{pp}_j^0 = \mathsf{C.Setup}(1^\lambda; r_{\mathsf{pp}}^j)$, $\mathsf{pp}_j^1 = \mathsf{OutParam}(\mathsf{pp}_j^1)$, $(\phi_k, \mathsf{st}_k) \leftarrow \mathsf{OutCom}(\mathsf{pp}_i^0, \mathsf{pp}_j^0, r_{\mathsf{pp}}^j, w_k; S_k)$ for $k \in \{n+1, \ldots, n+\ell\}$, where $\mathbf{c}_i = \mathsf{C.Commit}(\mathsf{pp}_i, w_i; o_i)$ for $i \in [n]$, $\pi_{\mathsf{bit}}^i = \mathsf{Bit.Prove}(\mathsf{pp}, \mathbf{c}_i, o_i; t_i)$ for $i \in [n+2\ell]$. Finally $\pi_{\mathsf{gate}}^j = \mathsf{N.Prove}(\mathsf{pp}_j, \{\mathbf{c}_{j_i}, o_{j_i}\}_{i\in[3]}; s_j)$ for $j \in [m]$.

Similarly, let $\Pi' = \mathsf{NIWI.Rand}((C, b), \Pi; \mathbf{R}')$ and parse

$$\mathbf{R}' = \left[\{r_{\mathsf{pp}}^{j'}, s_j'\}_{j\in[m]}, \{S_k'\}_{k\in[\ell]}, \{o_i', t_i'\}_{i\in[n+2\ell]}\right]$$

where $(\mathsf{pp}_j^0)' = \mathsf{RParam}(\mathsf{pp}_j^0; r_{\mathsf{pp}}^{j'})$, $(\phi_k', \mathsf{st}_k') \leftarrow \mathsf{RCom}(\phi_k; S_k')$, $\mathbf{c}_i' = \mathsf{C.Rand}(\mathsf{pp}_i, \mathbf{c}_i; o_i')$ for $i \in [n+2\ell]$, $\pi_{\mathsf{bit}}^{i'} = \mathsf{Bit.Maul}(\mathsf{pp}, \mathbf{c}_i, o_i', \pi_{\mathsf{bit}}^i; t_i')$. Finally, $\pi_{\mathsf{gate}}^{j'} = \mathsf{N.Maul}(\mathsf{pp}, \{\mathbf{c}_{j_i}, o_{j_i}'\}_{i\in[3]}, \pi_{\mathsf{N}}^j; s_j')$.

By perfect randomizability of $\mathsf{TC.Maul}$, there exists function f_σ given by $(R_i', R_j', S_k'') = f_\sigma(r_{\mathsf{pp}}^i, r_{\mathsf{pp}}^j, r_{\mathsf{pp}}^{i'}, r_{\mathsf{pp}}^{j'}, \mathsf{st}_k')$ such that $(\mathsf{pp}_q^0)' \leftarrow \mathsf{C.Setup}(1^\lambda; R_q')$ for $q \in \{i, j\}$ and $(\phi_k', \mathsf{st}_k') \leftarrow \mathsf{OutCom}(\mathsf{pp}_i, \mathsf{pp}_i,\ R_i, b; S_k'')$.

By perfect randomizability of commitment scheme and of underlying proof systems, there exists $o_i'' = f_{\mathsf{com}}(o_i, o_i')$ such that $\mathbf{c}_i' = \mathsf{C.Commit}(\mathsf{pp}, w_i; o_i'')$ and o_i'' is distributed as uniform, there exists $f_{\mathsf{bit}}(t_i', t_i, o_i, o_i') = t_i''$ such that $\pi_{\mathsf{bit}}^{i'} = \mathsf{Bit.Prove}(\mathsf{pp}, \mathbf{c}_i', o_i''; t_i'')$ and t_i'' is distributed as uniform.

Similarly, $f_{\mathsf{gate}}(s_j', s_j, \{o_{j_i}, o_{j_i}'\}_{i\in[3]}) = s_j'$ and $\pi_{\mathsf{gate}}^{j'} = \mathsf{N.Prove}(\mathsf{pp}, \{\mathbf{c}_{j_i}, o_{j_i}''\}_{i\in[3]}; s_j'')$ where s_j'' is distributed as uniform. We can now identify $\mathbf{R}''$ such that $\Pi' = \mathsf{NIWI.Prove}((C, b), w; \mathbf{R}'')$ as follows:

$$\mathbf{R}'' = \left[\{R_{j'}, s_j''\}_{j\in[m]}, \{S_k''\}_{k\in[\ell]}, \{o_i'', t_i''\}_{i\in[n+2\ell]}\right]$$

which is distributed as uniform. It follows that $\left\{(C, b), w, \Pi, \mathbf{R}, \Pi_f\right\}$ and $\left\{(C, b), w, \Pi, \mathbf{R}, \Pi'\right\}$ are identical distributions, where $\Pi_f = \mathsf{NIWI.Prove}((C, b), w; \mathbf{S})$ and $\Pi' = \mathsf{NIWI.Prove}((C, b), w; \mathbf{R}'')$ for truly random $\mathbf{S}, \mathbf{R}''$.

□

Claim 6. Π_{FHNIWI} *satisfies unlinkability.*

Proof. Follows from the completeness of the underlying primitives and randomizability of the NIWI. □

Claim 7. Π_{FHNIWI} *satisfies witness indistinguishabiilty.*

Proof. Fix any $(C, \mathsf{out}), \mathsf{wit}_0, \mathsf{wit}_1$ such that $((C, \mathsf{out}), \mathsf{wit}_0) \in R_{\mathcal{U}}$ and $((C, \mathsf{out}), \mathsf{wit}_1) \in R_{\mathcal{U}}$. We will prove that $\{\Pi_0\} \approx \{\Pi_1\}$ where $\Pi_b \leftarrow \mathsf{NIWI.Prove}((C, \mathsf{out}), \mathsf{wit}_b)$ for $b \in \{0, 1\}$.

For $C : \{0,1\}^n \to \{0,1\}$, denote by m the number of NAND gates, by ℓ the number of connecting wires, and by S the set of all pairs (k, j) for $k \in [n+\ell], j \in [m]$, such that wire k is an input or output to gate j. Let $\mathbf{w}^b = (w_1^b, \ldots, w_{n+\ell}^b)$ be the values induced by wit_b on all the wires excluding the output wire (but including the input wires). We will proceed through the following hybrids:

Hyb_0**:** Compute $\Pi \leftarrow \mathsf{NIWI.Prove}((C, \mathsf{out}), \mathsf{wit}_0)$. Output proof Π.

Hyb_1: This is exactly as Hyb_0 with the following changes: Instead of choosing fresh $\mathsf{pp}_j \leftarrow \mathsf{C.Setup}(1^\lambda)$ for each gate $j \in [m]$, compute only two parameters $\mathsf{pp}_1, \mathsf{pp}_2$ by running $\mathsf{C.Setup}(1^\lambda)$ twice independently. Recall that C is a layered circuit. Use parameters pp_1 for all the gates on odd layers of C and pp_2 for all the gates on even layers of C. Compute the rest of the proof honestly and at the end, randomize the resulting proof. In more detail, Hyb_1 is as follows:

1. Denote by $\mathsf{layer}_1, \ldots, \mathsf{layer}_t$ the t layers of gates in C. Choose at random $r_1, r_2 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
   - $\underline{\text{For each } i \in [2] \text{, compute } \mathsf{pp}_i^0 = \mathsf{C.Setup}(1^\lambda; r_i) \text{ and compute } \mathsf{pp}_i^1 = \mathsf{OutParam}(\mathsf{pp}_i^0).}$

– For all $j \in [t]$ and for each gate $v_j \in \mathsf{layer}_j$, let $\overrightarrow{\mathsf{pp}}_{v_j} = (\mathsf{pp}_1^0, \mathsf{pp}_1^1)$ if j is odd, else $\overrightarrow{\mathsf{pp}}_{v_j} = (\mathsf{pp}_2^0, \mathsf{pp}_2^1)$ if j is even.

2. For each input wire $k \in [n]$, denote by j the gate for which wire k is an input. For every $b \in \{0,1\}$, choose at random $o_{k,j}^b$ and compute $\mathbf{c}_{k,j}^b = \mathsf{C.Commit}(\mathsf{pp}_j^b, w_k^0; o_{k,j}^b)$. Let $\mathbf{c}_k = (\mathbf{c}_{k,j}^0, \mathbf{c}_{k,j}^1)$.

   For the output wire wout and for every $b \in \{0,1\}$, if $\mathsf{out} = 1$, $\mathbf{c}_{\mathsf{wout},m}^b = \mathbf{1}$ and if $\mathsf{out} = 0$, $\mathbf{c}_{\mathsf{wout},m}^b = \mathbf{0}$. Let $\mathbf{c}_{\mathsf{wout}} = (\mathbf{c}_{\mathsf{wout},m}^0, \mathbf{c}_{\mathsf{wout},m}^1)$.

3. For each connecting wire $k \in \{n+1, \ldots, n+\ell\}$ that connects gates $i, j \in [m]$, compute

$$\left(\sigma_c^k, \sigma_\pi^k, \mathsf{st}^k\right) \leftarrow \mathsf{OutCom}(\mathsf{pp}_i^0, \mathsf{pp}_j^0, r_j, w_k^0)$$

   where $\sigma_c^k = (\mathbf{c}_{k,i}^0, \mathbf{c}_{k,j}^0, \mathbf{c}_{k,i}^1, \mathbf{c}_{k,j}^1)$, $\sigma_\pi^k = (\pi_k^{00}, \pi_k^{01}, \pi_k^{10}, \pi_k^{11})$ and where $\mathsf{st}^k = (o_{k,i}^0, o_{k,j}^0, o_{k,i}^1, o_{k,j}^1)$. We will denote $(\sigma_c^k, \sigma_\pi^k)$ by Φ_k.

4. For all $(k,j) \in S$ and for every $b \in \{0,1\}$, generate a proof that the commitment $\mathbf{c}_{k,j}^b$ commits to a bit. Namely, compute

$$\pi_{\mathsf{bit}}[k,j]^b \leftarrow \mathsf{Bit.Prove}(\mathsf{pp}_j^b, \mathbf{c}_{k,j}^b, o_{k,j}^b)$$

   where $o_{k,j}^b$ is the opening for commitment $\mathbf{c}_{k,j}^b$ as computed in step 2 (for input wires) or as part of st^k output by OutCom (for connecting wires) in step 3. Let $\pi_{\mathsf{bit}}[k,j] = \left(\pi_{\mathsf{bit}}[k,j]^0, \pi_{\mathsf{bit}}[k,j]^1\right)$.

5. For each gate $j \in [m]$, denote by k_1, k_2 the input wires of the gate j and by k_3, k_4 the output wires of the gate j. For each $t \in \{3,4\}$ and $b \in \{0,1\}$, compute a gate consistency proof as follows:

$$\pi_{\mathsf{gate}}^{j,b}[t] \leftarrow \mathsf{N.Prove}\left(\mathsf{pp}_j^b, \{\mathbf{c}_{k_i,j}^b\}_{i\in\{1,2,t\}}, \{w_{k_i}^0, o_{k_i,j}^b\}_{i\in\{1,2,t\}}\right)$$

   Let $\pi_{\mathsf{gate}}^j = \left(\pi_{\mathsf{gate}}^{j,0}[3], \pi_{\mathsf{gate}}^{j,1}[3], \pi_{\mathsf{gate}}^{j,0}[4], \pi_{\mathsf{gate}}^{j,1}[4]\right)$.

6. Let $\Pi = \left[\{\overrightarrow{\mathsf{pp}}_j\}_{j\in[m]}, \{\mathbf{c}_k\}_{k\in[n]}, \{\Phi_k\}_{k\in[\ell]}, \left\{\pi_{\mathsf{bit}}[i,j]\right\}_{(i,j)\in S}, \{\pi_{\mathsf{gate}}^j\}_{j\in[m]}, \mathbf{c}_{\mathsf{wout}}\right]$.
   Finally compute $\Pi' \leftarrow \mathsf{NIWI.Rand}((C, \mathsf{out}), \Pi)$. Output Π'.

Hyb_2: This is exactly as Hyb_1 with the following changes: Recall that for each $j \in [m]$ and each $\overrightarrow{\mathsf{pp}}_j \in \Pi$ where $\overrightarrow{\mathsf{pp}}_j = (\mathsf{pp}_j^0, \mathsf{pp}_j^1)$, pp_j^0 is the binding parameter and pp_j^1 is the hiding parameter. We now equivocate the commitments with respect to the hiding parameters to obtain the openings with respect to wit_1. We then compute the bit consistency and gate consistency proofs for the hiding parameters using the equivocated openings with respect to wit_1. Note that the L_{TC} proofs output by OutCom (step 3) are still with respect to wit_0. In more detail, Hyb_2 is as follows:

1. Denote by $\mathsf{layer}_1, \ldots, \mathsf{layer}_t$ the t layers of gates in C. Choose at random $r_1, r_2 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
   – For each $i \in [2]$, compute $\mathsf{pp}_i^0 = \mathsf{C.Setup}(1^\lambda; r_i)$ and compute $\mathsf{pp}_i^1 = \mathsf{OutParam}(\mathsf{pp}_i^0)$.
   – For all $j \in [t]$ and for each gate $v_j \in \mathsf{layer}_j$, let $\overrightarrow{\mathsf{pp}}_{v_j} = (\mathsf{pp}_1^0, \mathsf{pp}_1^1)$ if j is odd, else $\overrightarrow{\mathsf{pp}}_{v_j} = (\mathsf{pp}_2^0, \mathsf{pp}_2^1)$ if j is even.

2. For each input wire $k \in [n]$, denote by j the gate for which wire k is an input. For every $b \in \{0,1\}$, choose at random $o^b_{k,j}$ and compute $\mathbf{c}^b_{k,j} = \mathsf{C.Commit}(\mathsf{pp}^b_j, w^0_k; o^b_{k,j})$. Let $\mathbf{c}_k = (\mathbf{c}^0_{k,j}, \mathbf{c}^1_{k,j})$.

   For the output wire wout and for every $b \in \{0,1\}$, if $\mathsf{out} = 1$, $\mathbf{c}^b_{\mathsf{wout},m} = \mathbf{1}$ and if $\mathsf{out} = 0$, $\mathbf{c}^b_{\mathsf{wout},m} = \mathbf{0}$. Let $\mathbf{c}_{\mathsf{wout}} = (\mathbf{c}^0_{\mathsf{wout},m}, \mathbf{c}^1_{\mathsf{wout},m})$.

3. For each connecting wire $k \in \{n+1, \ldots, n+\ell\}$ that connects gates $i, j \in [m]$, compute

$$\left(\sigma^k_c, \sigma^k_\pi, \mathsf{st}^k\right) \leftarrow \mathsf{OutCom}(\mathsf{pp}^0_i, \mathsf{pp}^0_j, r_j, w^0_k)$$

   where $\sigma^k_c = (\mathbf{c}^0_{k,i}, \mathbf{c}^0_{k,j}, \mathbf{c}^1_{k,i}, \mathbf{c}^1_{k,j})$, $\sigma^k_\pi = (\pi^{00}_k, \pi^{01}_k, \pi^{10}_k, \pi^{11}_k)$ and where $\mathsf{st}^k = (o^0_{k,i}, o^0_{k,j}, o^1_{k,i}, o^1_{k,j})$. We will denote $(\sigma^k_c, \sigma^k_\pi)$ by Φ_k.

4. <u>For all $(k,j) \in S$, first compute $s^1_{k,j} = \mathsf{C.Equivocate}(\mathsf{pp}^1_j, r_j, \mathbf{c}^1_{k,j}, o^1_{k,j}, w^1_k)$ where r_j is the randomness used to generate pp^0_j in step 1. Note that if $w^0_k = w^1_k$, then $s^1_{k,j} = o^1_{k,j}$</u> since equivocation is to the same bit as the committed bit.

   Compute $\pi_{\mathsf{bit}}[k,j]^0 \leftarrow \mathsf{Bit.Prove}(\mathsf{pp}^0_j, \mathbf{c}^0_{k,j}, o^0_{k,j})$ as before where $o^0_{k,j}$ is the opening for commitment $\mathbf{c}^0_{k,j}$ as computed in step 2 (for input wires) or as part of st^k output by OutCom (for connecting wires) in step 3. <u>Compute $\pi_{\mathsf{bit}}[k,j]^1 \leftarrow \mathsf{Bit.Prove}(\mathsf{pp}^1_j, \mathbf{c}^1_{k,j}, s^1_{k,j})$</u> where $s^1_{k,j}$ is the opening of $\mathbf{c}^1_{k,j}$ with respect to wit_1 as computed before. Let $\pi_{\mathsf{bit}}[k,j] = \left(\pi_{\mathsf{bit}}[k,j]^0, \pi_{\mathsf{bit}}[k,j]^1\right)$.

5. For each gate $j \in [m]$, denote by k_1, k_2 the input wires of the gate j and by k_3, k_4 the output wires to gate j. For each $t \in \{3,4\}$, compute gate consistency proofs as follows:

$$\pi^{j,0}_{\mathsf{gate}}[t] \leftarrow \mathsf{N.Prove}\big(\mathsf{pp}^0_j, \{\mathbf{c}^0_{k_i,j}\}_{i\in\{1,2,t\}}, \{w^0_{k_i}, o^0_{k_i,j}\}_{i\in\{1,2,t\}}\big)$$

   as before and <u>$\pi^{j,1}_{\mathsf{gate}}[t] \leftarrow \mathsf{N.Prove}\big(\mathsf{pp}^1_j, \{\mathbf{c}^1_{k_i,j}\}_{i\in\{1,2,t\}}, \{w^1_{k_i}, s^1_{k_i,j}\}_{i\in\{1,2,t\}}\big)$.</u>
   Let $\pi^j_{\mathsf{gate}} = \left(\pi^{j,0}_{\mathsf{gate}}[3], \pi^{j,1}_{\mathsf{gate}}[3], \pi^{j,0}_{\mathsf{gate}}[4], \pi^{j,1}_{\mathsf{gate}}[4]\right)$.

6. Let $\Pi = \left[\{\overrightarrow{\mathsf{pp}}_j\}_{j\in[m]}, \{\mathbf{c}_k\}_{k\in[n]}, \{\Phi_k\}_{k\in[\ell]}, \{\pi_{\mathsf{bit}}[i,j]\}_{(i,j)\in S}, \{\pi^j_{\mathsf{gate}}\}_{j\in[m]}, \mathbf{c}_{\mathsf{wout}}\right]$. Finally compute $\Pi' \leftarrow \mathsf{NIWI.Rand}((C, \mathsf{out}), \Pi)$. Output $\left\{(C, \mathsf{out}), \mathsf{wit}_0, \mathsf{wit}_1, \Pi'\right\}$.

Hyb_3**:** This is exactly as hybrid 2 and the only change is in step 3 where we use $\mathsf{OutCom}_{\mathsf{Bind}}$ instead of using OutCom. Recall that OutCom outputs four commitments $(\mathbf{c}^0_1, \mathbf{c}^0_2, \mathbf{c}^1_1, \mathbf{c}^1_2)$ with respect to $(\mathsf{pp}^0_1, \mathsf{pp}^0_2, \mathsf{pp}^1_1, \mathsf{pp}^1_2)$ respectively, four L_{TC} proofs and openings for the four commitments. $\mathsf{OutCom}_{\mathsf{Bind}}$ is same as OutCom except that it computes the L_{TC} proof for $(\mathbf{c}^1_1, \mathbf{c}^1_2, \mathsf{pp}^1_1, \mathsf{pp}^1_2)$ differently. In more detail,

$(\sigma_c, \sigma_\pi, \mathsf{st}) \leftarrow \mathsf{OutCom}_{\mathsf{Bind}}(\mathsf{pp}^0_1, \mathsf{pp}^0_2, r_1, r_2, \mathsf{bit})$: The $\mathsf{OutCom}_{\mathsf{Bind}}$ algorithm takes as input two pairs of parameters $\mathsf{pp}^0_1, \mathsf{pp}^0_2 \in \mathsf{C.Setup}(1^\lambda)$, randomness r_1, r_2 such that for all $i \in [2]$, $\mathsf{pp}^0_i = \mathsf{C.Setup}(1^\lambda; r_i)$ and a bit, and does the following:

– For all $i \in [2]$, compute $\mathsf{pp}^1_i = \mathsf{OutParam}(\mathsf{pp}^0_i)$.

– For all $i \in [2]$, for all $d \in \{0,1\}$, choose at random o^d_i and compute $\mathbf{c}^d_i = \mathsf{C.Commit}(\mathsf{pp}^d_i, \mathsf{bit}; o^d_i)$. Denote by $\sigma_c = (\mathbf{c}^0_1, \mathbf{c}^0_2, \mathbf{c}^1_1, \mathbf{c}^1_2)$.

– Compute $\mathsf{pp}^0_* = \mathsf{InterParam}(\mathsf{pp}^0_1, \mathsf{pp}^0_2, r_1)$ and $\mathsf{pp}^1_* = \mathsf{InterParam}(\mathsf{pp}^1_1, \mathsf{pp}^1_2, r_1)$. For all $b_1, b_2 \in \{0,1\}$

except for $b_1 = b_2 = 1$, compute

$$\pi^{b_1 b_2} \leftarrow \mathsf{TC.Prove}\big((\mathbf{c}_1^{b_1}, \mathbf{c}_2^{b_2}, \mathsf{pp}_1^{b_1}, \mathsf{pp}_2^{b_2}), (\mathsf{bit}, \mathsf{pp}_*^{b_1}, o_1^{b_1}, o_2^{b_2})\big).$$

$\underline{\text{For all } i \in [2] \text{, compute } s_i^1 = \mathsf{C.Equivocate}(\mathsf{pp}_i^1, r_i, \mathbf{c}_i^1, o_i^1, 1 - \mathsf{bit}). \text{ Denote by } \mathsf{st} = (o_1^0, o_2^0, s_1^1, s_2^1).}$
$\underline{\text{Compute } \pi^{11} \leftarrow \mathsf{TC.Prove}\big((\mathbf{c}_1^1, \mathbf{c}_2^1, \mathsf{pp}_1^1, \mathsf{pp}_2^1), (1 - \mathsf{bit}, \mathsf{pp}_*^1, s_1^1, s_2^1)\big).}$
Denote by $\sigma_\pi = (\pi^{00}, \pi^{01}, \pi^{10}, \pi^{11})$. Output $(\sigma_c, \sigma_\pi, \mathsf{st})$.

Concretely, the changed step 3 in Hyb_3 will be as follows:

For each connecting wire $k \in \{n+1, \ldots, n+\ell\}$ that connects gates $i, j \in [m]$, if $w_k^0 \neq w_k^1$ then compute $\big(\sigma_c^k, \sigma_\pi^k, \mathsf{st}^k\big) \leftarrow \mathsf{OutCom}_{\mathsf{Bind}}(\mathsf{pp}_i^0, \mathsf{pp}_j^0, r_i, r_j, w_k^0)$ else compute $\big(\sigma_c^k, \sigma_\pi^k, \mathsf{st}^k\big) \leftarrow \mathsf{OutCom}(\mathsf{pp}_i^0, \mathsf{pp}_j^0, r_i, w_k^0)$.

Hyb_4**:** In this hybrid, compute $\mathsf{pp}_1^0 \leftarrow \mathsf{C.Setup}'(1^\lambda)$ and $\mathsf{pp}_2^0 \leftarrow \mathsf{C.Setup}'(1^\lambda)$. As before, compute $\mathsf{pp}_i^1 = \mathsf{OutParam}(\mathsf{pp}_i^0)$ for all $i \in [2]$. Note that $\mathsf{pp}_1^0, \mathsf{pp}_2^0$ are now the hiding parameters and $\mathsf{pp}_1^1, \mathsf{pp}_2^1$ are the binding parameters.

In addition, all the commitments are now with respect to wit_1 but the bit consistency and gate consistency proofs for the hiding parameters, are with respect to wit_0. These are computed by using the equivocations to wit_0, with respect to hiding parameters (similar to $\mathsf{Hyb}_2, \mathsf{Hyb}_3$).

Also in step 3, use $\mathsf{OutCom}_{\mathsf{Hide}}$ instead of using $\mathsf{OutCom}_{\mathsf{Bind}}$. $\mathsf{OutCom}_{\mathsf{Hide}}$ is similar to OutCom except that it computes the L_{TC} proof for $(\mathbf{c}_1^0, \mathbf{c}_2^0, \mathsf{pp}_1^0, \mathsf{pp}_2^0)$ differently. In more detail,

$\big(\sigma_c, \sigma_\pi, \mathsf{st}\big) \leftarrow \mathsf{OutCom}_{\mathsf{Hide}}(\mathsf{pp}_1^0, \mathsf{pp}_2^0, r_1, r_2, \mathsf{bit})$: The $\mathsf{OutCom}_{\mathsf{Hide}}$ algorithm takes as input two pairs of parameters $\mathsf{pp}_1^0, \mathsf{pp}_2^0 \in \mathsf{C.Setup}'(1^\lambda)$, randomness r_1, r_2 such that for all $i \in [2]$, $\mathsf{pp}_i^0 = \mathsf{C.Setup}'(1^\lambda; r_i)$ and a bit, and does the following:

- For all $i \in [2]$, compute $\mathsf{pp}_i^1 = \mathsf{OutParam}(\mathsf{pp}_i^0)$.
- For all $i \in [2]$, for all $d \in \{0,1\}$, choose at random o_i^d and compute $\mathbf{c}_i^d = \mathsf{C.Commit}(\mathsf{pp}_i^d, \mathsf{bit}; o_i^d)$. Denote by $\sigma_c = (\mathbf{c}_1^0, \mathbf{c}_2^0, \mathbf{c}_1^1, \mathbf{c}_2^1)$.
- Compute $\mathsf{pp}_*^0 = \mathsf{InterParam}(\mathsf{pp}_1^0, \mathsf{pp}_2^0, r_1)$ and $\mathsf{pp}_*^1 = \mathsf{InterParam}(\mathsf{pp}_1^1, \mathsf{pp}_2^1, r_1)$. For all $b_1, b_2 \in \{0,1\}$ except for $b_1 = b_2 = 0$, compute

$$\pi^{b_1 b_2} \leftarrow \mathsf{TC.Prove}\big((\mathbf{c}_1^{b_1}, \mathbf{c}_2^{b_2}, \mathsf{pp}_1^{b_1}, \mathsf{pp}_2^{b_2}), (\mathsf{bit}, \mathsf{pp}_*^{b_1}, o_1^{b_1}, o_2^{b_2})\big).$$

  $\underline{\text{For all } i \in [2] \text{, compute } s_i^0 = \mathsf{C.Equivocate}(\mathsf{pp}_i^0, r_i, \mathbf{c}_i^1, o_i^0, 1 - \mathsf{bit}). \text{ Denote by } \mathsf{st} = (s_1^0, s_2^0, o_1^1, o_2^1).}$
  $\underline{\text{Compute } \pi^{00} \leftarrow \mathsf{TC.Prove}\big((\mathbf{c}_1^0, \mathbf{c}_2^0, \mathsf{pp}_1^0, \mathsf{pp}_2^0), (1 - \mathsf{bit}, \mathsf{pp}_*^1, s_1^0, s_2^0)\big).}$
  Denote by $\sigma_\pi = (\pi^{00}, \pi^{01}, \pi^{10}, \pi^{11})$. Output $(\sigma_c, \sigma_\pi, \mathsf{st})$.

We now describe the hybrid in detail:

1. Denote by $\mathsf{layer}_1, \ldots, \mathsf{layer}_t$ the t layers of gates in C. Choose at random $r_1, r_2 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
   - $\underline{\text{For each } i \in [2] \text{, compute } \mathsf{pp}_i^0 = \mathsf{C.Setup}'(1^\lambda; r_i) \text{ and compute } \mathsf{pp}_i^1 = \mathsf{OutParam}(\mathsf{pp}_i^0).}$
   - For all $j \in [t]$ and for each gate $v_j \in \mathsf{layer}_j$, let $\overrightarrow{\mathsf{pp}}_{v_j} = (\mathsf{pp}_1^0, \mathsf{pp}_1^1)$ if j is odd, else $\overrightarrow{\mathsf{pp}}_{v_j} = (\mathsf{pp}_2^0, \mathsf{pp}_2^1)$ if j is even.

2. For each input wire $k \in [n]$, denote by j the gate for which wire k is an input. For every $b \in \{0,1\}$, choose at random $o^b_{k,j}$ and $\underline{\text{compute } \mathbf{c}^b_{k,j} = \mathsf{C.Commit}(\mathsf{pp}^b_j, w^1_k; o^b_{k,j})}$. Let $\mathbf{c}_k = (\mathbf{c}^0_{k,j}, \mathbf{c}^1_{k,j})$.

   For the output wire wout and for every $b \in \{0,1\}$, if $\mathsf{out} = 1$, $\mathbf{c}^b_{\mathsf{wout},m} = \mathbf{1}$ and if $\mathsf{out} = 0$, $\mathbf{c}^b_{\mathsf{wout},m} = \mathbf{0}$. Let $\mathbf{c}_{\mathsf{wout}} = (\mathbf{c}^0_{\mathsf{wout},m}, \mathbf{c}^1_{\mathsf{wout},m})$.

3. For each connecting wire $k \in \{n+1, \dots, n+\ell\}$ that connects gates $i,j \in [m]$, $\underline{\text{if } w^0_k \neq w^1_k \text{ then}}$ $\underline{\text{compute } (\sigma^k_c, \sigma^k_\pi, \mathsf{st}^k) \leftarrow \mathsf{OutCom}_{\mathsf{Hide}}(\mathsf{pp}^0_i, \mathsf{pp}^0_j, r_i, r_j, w^1_k)}$ else compute $(\sigma^k_c, \sigma^k_\pi, \mathsf{st}^k) \leftarrow$ $\underline{\mathsf{OutCom}(\mathsf{pp}^0_i, \mathsf{pp}^0_j, r_i, w^1_k)}$ where $\sigma^k_c = (\mathbf{c}^0_{k,i}, \mathbf{c}^0_{k,j}, \mathbf{c}^1_{k,i}, \mathbf{c}^1_{k,j})$, $\sigma^k_\pi = (\pi^{00}_k, \pi^{01}_k, \pi^{10}_k, \pi^{11}_k)$ and where $\mathsf{st}^k = (o^0_{k,i}, o^0_{k,j}, o^1_{k,i}, o^1_{k,j})$. We will denote $(\sigma^k_c, \sigma^k_\pi)$ by Φ_k.

4. For all $(k,j) \in S$, $\underline{\text{first compute } s^0_{k,j} = \mathsf{C.Equivocate}(\mathsf{pp}^0_j, r_j, \mathbf{c}^0_{k,j}, o^0_{k,j}, w^0_k)}$ where r_j is the randomness used to generate pp^0_j in step 1. Compute $\pi_{\mathsf{bit}}[k,j]^0 \leftarrow \mathsf{Bit.Prove}(\mathsf{pp}^0_j, \mathbf{c}^0_{k,j}, s^0_{k,j})$.

   For all $(k,j) \in S$, compute $\pi_{\mathsf{bit}}[k,j]^1 \leftarrow \mathsf{Bit.Prove}(\mathsf{pp}^1_j, \mathbf{c}^1_{k,j}, o^1_{k,j})$, where $o^1_{k,j}$ is the opening for commitment $\mathbf{c}^0_{k,j}$ as computed in step 2 (for input wires) or as part of st^k output by OutCom (for connecting wires) in step 3. Let $\pi_{\mathsf{bit}}[k,j] = (\pi_{\mathsf{bit}}[k,j]^0, \pi_{\mathsf{bit}}[k,j]^1)$.

5. For each gate $j \in [m]$, denote by k_1, k_2 the input wires of the gate j and by k_3, k_4 the output wires to gate j. For each $t \in \{3,4\}$, compute gate consistency proofs as follows:

$$\pi^{j,0}_{\mathsf{gate}}[t] \leftarrow \mathsf{N.Prove}\big(\mathsf{pp}^0_j, \{\mathbf{c}^0_{k_i,j}\}_{i\in\{1,2,t\}}, \{w^0_{k_i}, s^0_{k_i,j}\}_{i\in\{1,2,t\}}\big)$$

   and

$$\pi^{j,1}_{\mathsf{gate}}[t] \leftarrow \mathsf{N.Prove}\big(\mathsf{pp}^1_j, \{\mathbf{c}^1_{k_i,j}\}_{i\in\{1,2,t\}}, \{w^1_{k_i}, o^1_{k_i,j}\}_{i\in\{1,2,t\}}\big)$$

   Let $\pi^j_{\mathsf{gate}} = \big(\pi^{j,0}_{\mathsf{gate}}[3], \pi^{j,1}_{\mathsf{gate}}[3], \pi^{j,0}_{\mathsf{gate}}[4], \pi^{j,1}_{\mathsf{gate}}[4]\big)$.

6. Let $\Pi = \big[\{\overrightarrow{\mathsf{pp}}_j\}_{j\in[m]}, \{\mathbf{c}_k\}_{k\in[n]}, \{\Phi_k\}_{k\in[\ell]}, \{\pi_{\mathsf{bit}}[i,j]\}_{(i,j)\in S}, \{\pi^j_{\mathsf{gate}}\}_{j\in[m]}, \mathbf{c}_{\mathsf{wout}}\big]$. Finally compute $\Pi' \leftarrow \mathsf{NIWI.Rand}((C,\mathsf{out}),\Pi)$. Output $\big\{(C,\mathsf{out}), \mathsf{wit}_0, \mathsf{wit}_1, \Pi'\big\}$.

Hyb_5**:** This hybrid is same as Hyb_4 except that in step 3 it uses OutCom instead of $\mathsf{OutCom}_{\mathsf{Hide}}$. More specifically step 3 is as follows:

For each connecting wire $k \in \{n+1, \dots, n+\ell\}$ that connects gates $i,j \in [m]$, compute

$$\big(\sigma^k_c, \sigma^k_\pi, \mathsf{st}^k\big) \leftarrow \mathsf{OutCom}(\mathsf{pp}^0_i, \mathsf{pp}^0_j, r_j, w^0_k)$$

Hyb_6**:** This hybrid is same as Hyb_5 except that it uses wit_1 for all the bit consistency and gate consistency proofs in steps 4,5. In detail, the hybrid is as follows:

1. Denote by $\mathsf{layer}_1, \dots, \mathsf{layer}_t$ the t layers of gates in C. Choose at random $r_1, r_2 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.
   - For each $i \in [2]$, compute $\mathsf{pp}^0_i = \mathsf{C.Setup}(1^\lambda; r_i)$ and compute $\mathsf{pp}^1_i = \mathsf{OutParam}(\mathsf{pp}^0_i)$.
   - For all $j \in [t]$ and for each gate $v_j \in \mathsf{layer}_j$, let $\overrightarrow{\mathsf{pp}}_{v_j} = (\mathsf{pp}^0_1, \mathsf{pp}^1_1)$ if j is odd, else $\overrightarrow{\mathsf{pp}}_{v_j} = (\mathsf{pp}^0_2, \mathsf{pp}^1_2)$ if j is even.

2. For each input wire $k \in [n]$, denote by j the gate for which wire k is an input. For every $b \in \{0,1\}$, choose at random $o^b_{k,j}$ and compute $\mathbf{c}^b_{k,j} = \mathsf{C.Commit}(\mathsf{pp}^b_j, w^1_k; o^b_{k,j})$. Let $\mathbf{c}_k = (\mathbf{c}^0_{k,j}, \mathbf{c}^1_{k,j})$.

   For the output wire wout and for every $b \in \{0,1\}$, if $\mathsf{out} = 1$, $\mathbf{c}^b_{\mathsf{wout},m} = \mathbf{1}$ and if $\mathsf{out} = 0$, $\mathbf{c}^b_{\mathsf{wout},m} = \mathbf{0}$. Let $\mathbf{c}_{\mathsf{wout}} = (\mathbf{c}^0_{\mathsf{wout},m}, \mathbf{c}^1_{\mathsf{wout},m})$.

3. For each connecting wire $k \in \{n+1, \dots, n+\ell\}$ that connects gates $i, j \in [m]$, compute

$$\left(\sigma^k_c, \sigma^k_\pi, \mathsf{st}^k\right) \leftarrow \mathsf{OutCom}(\mathsf{pp}^0_i, \mathsf{pp}^0_j, r_j, w^1_k)$$

   where $\sigma^k_c = (\mathbf{c}^0_{k,i}, \mathbf{c}^0_{k,j}, \mathbf{c}^1_{k,i}, \mathbf{c}^1_{k,j})$, $\sigma^k_\pi = (\pi^{00}_k, \pi^{01}_k, \pi^{10}_k, \pi^{11}_k)$ and where $\mathsf{st}^k = (o^0_{k,i}, o^0_{k,j}, o^1_{k,i}, o^1_{k,j})$. We will denote $(\sigma^k_c, \sigma^k_\pi)$ by Φ_k.

4. For all $(k,j) \in S$ and $b \in \{0,1\}$, compute

$$\underline{\pi_{\mathsf{bit}}[k,j]^b \leftarrow \mathsf{Bit.Prove}(\mathsf{pp}^b_j, \mathbf{c}^b_{k,j}, o^b_{k,j})}$$

   where $o^b_{k,j}$ is the opening for commitment $\mathbf{c}^b_{k,j}$ as computed in step 2 (for input wires) or as part of st^k output by OutCom (for connecting wires) in step 3. Let $\pi_{\mathsf{bit}}[k,j] = \left(\pi_{\mathsf{bit}}[k,j]^0, \pi_{\mathsf{bit}}[k,j]^1\right)$.

5. For each gate $j \in [m]$, denote by k_1, k_2 the input wires of the gate j and by k_3, k_4 the output wires to gate j. For each $t \in \{3,4\}$ and $b \in \{0,1\}$, compute a gate consistency proof as follows:

$$\underline{\pi^{j,b}_{\mathsf{gate}}[t] \leftarrow \mathsf{N.Prove}\big(\mathsf{pp}^b_j, \{\mathbf{c}^b_{k_i,j}\}_{i \in \{1,2,t\}}, \{w^1_{k_i}, o^b_{k_i,j}\}_{i \in \{1,2,t\}}\big)}$$

   Let $\pi^j_{\mathsf{gate}} = \left(\pi^{j,0}_{\mathsf{gate}}[3], \pi^{j,1}_{\mathsf{gate}}[3], \pi^{j,0}_{\mathsf{gate}}[4], \pi^{j,1}_{\mathsf{gate}}[4]\right)$.

6. Let $\Pi = \left[\{\overrightarrow{\mathsf{pp}}_j\}_{j\in[m]}, \{\mathbf{c}_k\}_{k\in[n]}, \{\Phi_k\}_{k\in[\ell]}, \left\{\pi_{\mathsf{bit}}[i,j]\right\}_{(i,j)\in S}, \{\pi^j_{\mathsf{gate}}\}_{j\in[m]}, \mathbf{c}_{\mathsf{wout}}\right]$. Finally compute $\Pi' \leftarrow \mathsf{NIWI.Rand}((C, \mathsf{out}), \Pi)$. Output $\left\{(C, \mathsf{out}), \mathsf{wit}_0, \mathsf{wit}_1, \Pi'\right\}$.

Hyb_7: Exactly as Hyb_6 except in step 1, compute parameters $\mathsf{pp}^0_1, \mathsf{pp}^0_2$ using $\mathsf{C.Setup}$ instead of using $\mathsf{C.Setup}'$ so that they are binding again. In detail, step 1 will be as follows:

Denote by $\mathsf{layer}_1, \dots, \mathsf{layer}_t$ the t layers of gates in C. Choose at random $r_1, r_2 \leftarrow \{0,1\}^{\mathsf{poly}(\lambda)}$.

– For each $i \in [2]$, compute $\mathsf{pp}^0_i = \mathsf{C.Setup}(1^\lambda; r_i)$ and compute $\mathsf{pp}^1_i = \mathsf{OutParam}(\mathsf{pp}^0_i)$.

– For all $j \in [t]$ and for each gate $v_j \in \mathsf{layer}_j$, let $\overrightarrow{\mathsf{pp}}_{v_j} = (\mathsf{pp}^0_1, \mathsf{pp}^1_1)$ if j is odd, else $\overrightarrow{\mathsf{pp}}_{v_j} = (\mathsf{pp}^0_2, \mathsf{pp}^1_2)$ if j is even.

Hyb_8: Compute $\Pi \leftarrow \mathsf{NIWI.Prove}((C, \mathsf{out}), \mathsf{wit}_1)$. Output proof Π.

We now prove indistinguishability of all the hybrids. We note that the main challenge is proving that hybrids $3, 4$ are indistinguishable (we prove this at the end), the proof of which uses the strong secrecy of L_{TC}.

Proposition 6. $\mathsf{Hyb}_0 \approx \mathsf{Hyb}_1$

Proof. Follows directly from randomizability of the proof system (as proved in Claim 5). □

Proposition 7. $\mathsf{Hyb}_1 \approx \mathsf{Hyb}_2$

Proof. Follows by witness indistinguishability of $(\mathsf{Bit.Prove}, \mathsf{Bit.Verify})$ and of $(\mathsf{N.Prove}, \mathsf{N.Verify})$. □

Proposition 8. $\mathsf{Hyb}_2 \approx \mathsf{Hyb}_3$

Proof. Follows by witness indistinguishability (WI) of $(\mathsf{TC.Prove}, \mathsf{TC.Verify})$. Recall that strong secrecy of L_{TC} implies plain WI (see Remark 5). □

Proposition 9. $\mathsf{Hyb}_4 \approx \mathsf{Hyb}_5$

Proof. Follows by witness indistinguishability of $(\mathsf{TC.Prove}, \mathsf{TC.Verify})$. □

Proposition 10. $\mathsf{Hyb}_5 \approx \mathsf{Hyb}_6$

Proof. Follows by witness indistinguishability of $(\mathsf{Bit.Prove}, \mathsf{Bit.Verify})$ and of $(\mathsf{N.Prove}, \mathsf{N.Verify})$. □

Proposition 11. $\mathsf{Hyb}_6 \approx \mathsf{Hyb}_7$

Proof. Follows from the perfect equivocation of the RaHE-commitment scheme. Recall that $\mathsf{C.Setup}'$ outputs pp', such that $\{\mathsf{pp} \leftarrow \mathsf{C.Setup}(1^\lambda) \ : \ \mathsf{pp}\} \approx \{\mathsf{pp}' \leftarrow \mathsf{C.Setup}'(1^\lambda) \ : \ \mathsf{pp}'\}$. □

Proposition 12. $\mathsf{Hyb}_7 \approx \mathsf{Hyb}_8$

Proof. Follows directly from randomizability of the proof system. □

Proposition 13. $\mathsf{Hyb}_3 \approx \mathsf{Hyb}_4$

Proof. We will prove this via intermediate hybrids Hyb_3' and Hyb_4'. Hyb_3' is generated using a sample drawn from $\mathcal{D}_{\mathsf{Bind}}$ and its output is distributed identically to Hyb_3. Similarly, Hyb_4' is generated using a sample drawn from $\mathcal{D}_{\mathsf{Hide}}$ and its output is distributed identically to Hyb_4. The proposition then follows directly from the strong secrecy of L_{TC}.

Recall that by strong secrecy of L_{TC}, the following two distributions are computationally indistinguishable.

- $\mathcal{D}_{\mathsf{Bind}}(1^\lambda)$: Choose r at random and compute $\mathsf{pp}_1^0 = \mathsf{C.Setup}(1^\lambda; r)$. Compute $\mathsf{pp}_1^1 = \mathsf{OutParam}(\mathsf{pp}_1^0)$. For every $d \in \{0,1\}$, do the following:
  - Choose o_d, o_d'' at random and compute $\mathbf{c}_d = \mathsf{C.Commit}(\mathsf{pp}_1^0, d\ ; o_d)$, $\mathbf{c}_d' = \mathsf{C.Commit}(\mathsf{pp}_1^1, d; o_d'')$.
  - Compute $\Pi^d \leftarrow \mathsf{TC.Prove}((\mathbf{c}_d, \mathbf{c}_d', \mathsf{pp}_1^0, \mathsf{pp}_1^1), (d, \mathsf{pp}_1^0, o_d, o_d''))$.
  - Compute $o_d' = \mathsf{C.Equivocate}(\mathsf{pp}_1^1, r, \mathbf{c}_d', o_d'', 1-d)$.

  Output $\big(\mathsf{pp}_1^0, \mathsf{pp}_1^1, \mathbf{c}_0, \mathbf{c}_0', \mathbf{c}_1, \mathbf{c}_1', o_0, o_0', o_1, o_1', \Pi^0, \Pi^1\big)$.

- $\mathcal{D}_{\mathsf{Hide}}(1^\lambda)$: Choose r at random and compute $\mathsf{pp}_1^0 = \mathsf{C.Setup}'(1^\lambda; r)$. Compute $\mathsf{pp}_1^1 = \mathsf{OutParam}(\mathsf{pp})$. For every $d \in \{0,1\}$, do the following:
  - Choose o_d', o_d'' at random. Compute $\mathbf{c}_d = \mathsf{C.Commit}(\mathsf{pp}_1^0, 1-d\ ; o_d'')$ and compute $\mathbf{c}_d' = \mathsf{C.Commit}(\mathsf{pp}_1^1, 1-d; o_d')$.
  - Compute $\Pi^d \leftarrow \mathsf{TC.Prove}((\mathbf{c}_d, \mathbf{c}_d', \mathsf{pp}_1^0, \mathsf{pp}_1^1), (1-d, \mathsf{pp}_1^0, o_d'', o_d'))$.
  - Compute $o_d = \mathsf{C.Equivocate}(\mathsf{pp}_1^0, r, \mathbf{c}_d, o_d'', d)$.

  Output $\big(\mathsf{pp}_1^0, \mathsf{pp}_1^1, \mathbf{c}_0, \mathbf{c}_0', \mathbf{c}_1, \mathbf{c}_1', o_0, o_0', o_1, o_1', \Pi_{\mathsf{TC}}^0, \Pi_{\mathsf{TC}}^1\big)$.

Before describing the hybrids Hyb'_3 and Hyb'_4, we describe intermediate procedures $\mathsf{SCom}_0, \mathsf{SCom}_1$ that take as input a sample from $\mathcal{D}_{\mathsf{Bind}}$ or $\mathcal{D}_{\mathsf{Hide}}$ and output four commitments σ_c, four proofs σ_π and state st.

In detail, for every $d \in \{0,1\}$, SCom_d on input $(\mathsf{pp}_1^0, \mathsf{pp}_1^1, \mathbf{c}_0, \mathbf{c}'_0, \mathbf{c}_1, \mathbf{c}'_1, o_0, o'_0, o_1, o'_1, \Pi^0, \Pi^1)$ uses only a part of its input as follows:

SCom_d uses $(\mathsf{pp}_1^0, \mathsf{pp}_1^1, \mathbf{c}_d, \mathbf{c}'_d, o_d, o'_d, \Pi^d)$ and does the following:

– Choose randomness r' and compute $\mathsf{pp}_2^0 = \mathsf{RParam}(\mathsf{pp}_1^0; r')$. Compute $\mathsf{pp}_2^1 = \mathsf{OutParam}(\mathsf{pp}_2^0)$. For every $b \in \{0,1\}$, compute $\mathsf{pp}_*^b = \mathsf{InterParam}(\mathsf{pp}_1^b, \mathsf{pp}_2^b, r')$.

– Choose randomness o^0, o^1 and compute $\mathbf{c}_2^0 = \mathsf{C.Rand}(\mathsf{pp}_1^0, \mathbf{c}_d; o^0)$, $\mathbf{c}_2^1 = \mathsf{C.Rand}(\mathsf{pp}_1^1, \mathbf{c}'_d; o^1)$. For every $b \in \{0,1\}$, let o_2^b be the new opening of $\mathbf{c}_2^b$ computed as $o_2^0 = f_{\mathsf{com}}(o_d, o^0)$ and $o_2^1 = f_{\mathsf{com}}(o'_d, o^1)$. Denote by $\sigma_c = (\mathbf{c}_d, \mathbf{c}_2^0, \mathbf{c}'_d, \mathbf{c}_2^1)$ and $\mathsf{st} = (o_d, o_2^0, o'_d, o_2^1)$.

– Compute two fresh L_{TC} proofs as follows:

  – $\pi^{00} \leftarrow \mathsf{TC.Prove}\big((\mathbf{c}_d, \mathbf{c}_2^0, \mathsf{pp}_1^0, \mathsf{pp}_2^0), (d, \mathsf{pp}_*^0, o_d, o_2^0)\big)$.
  – $\pi^{11} \leftarrow \mathsf{TC.Prove}\big((\mathbf{c}'_d, \mathbf{c}_2^1, \mathsf{pp}_1^1, \mathsf{pp}_2^1), (1-d, \mathsf{pp}_*^1, o'_d, o_2^1)\big)$.

– Compute two mauled L_{TC} proofs as follows:

  – $\pi^{01} \leftarrow \mathsf{TC.Maul}\big((\mathbf{c}_d, \mathbf{c}_1^1, \mathsf{pp}_1^0, \mathsf{pp}_1^1), (1, r', 1, o^1), \pi\big)$. Note that mauled proof π^{01} is a proof that $(\mathbf{c}_d, \mathbf{c}_2^1, \mathsf{pp}_1^0, \mathsf{pp}_2^1) \in L_{\mathsf{TC}}$.
  – $\pi^{10} \leftarrow \mathsf{TC.Maul}\big((\mathbf{c}'_d, \mathbf{c}_1^1, \mathsf{pp}_1^0, \mathsf{pp}_1^1), (r', 1, o^0, 1), \pi\big)$. Note that mauled proof π^{10} is a proof that $(\mathbf{c}_2^0, \mathbf{c}'_d, \mathsf{pp}_2^0, \mathsf{pp}_1^1) \in L_{\mathsf{TC}}$.

Denote by $\sigma_\pi = (\pi^{00}, \pi^{01}, \pi^{10}, \pi^{11})$. Output $(\sigma_c, \sigma_\pi, \mathsf{st})$.

Claim 8. *Let $\Sigma_{\mathsf{Bind}} \leftarrow \mathcal{D}_{\mathsf{Bind}}(1^\lambda)$. Let r_1, r_2 be chosen at random and for all $i \in [2]$, let $\mathsf{pp}_i = \mathsf{C.Setup}(1^\lambda; r_i)$. Then, for all $d \in \{0,1\}$, the following distributions are identical:*

$$\big(\mathsf{SCom}_d(\Sigma_{\mathsf{Bind}})\big) \text{ and } \big(\mathsf{OutCom}_{\mathsf{Bind}}(\mathsf{pp}_1, \mathsf{pp}_2, r_1, r_2, d)\big)$$

Proof. Let us look at the difference in the two distributions: $\mathsf{SCom}_d(\Sigma_{\mathsf{Bind}})$ and $\mathsf{OutCom}_{\mathsf{Bind}}(\mathsf{pp}_1, \mathsf{pp}_2, r_1, r_2, d)$. Recall that $\mathsf{OutCom}_{\mathsf{Bind}}(\mathsf{pp}_1, \mathsf{pp}_2, r_1, r_2, d)$ computes four fresh commitments with respect to d to obtain $\sigma_c = (\mathbf{c}_1^0, \mathbf{c}_2^0, \mathbf{c}_1^1, \mathbf{c}_2^1)$. Proofs $\pi^{00}, \pi^{01}, \pi^{10}$ are computed honestly using the openings with respect to d and randomness r_1 such that $\mathsf{pp}_1 = \mathsf{C.Setup}(1^\lambda; r_1)$. Proof π^{11} is computed using equivocated openings of $\mathbf{c}_1^1$ and $\mathbf{c}_2^1$ with respect to $1-d$. Denote $\sigma_\pi = (\pi^{00}, \pi^{01}, \pi^{10}, \pi^{11})$. Finally, $\mathsf{OutCom}_{\mathsf{Bind}}$ outputs $(\sigma_c, \sigma_\pi, \mathsf{st})$ where st consists of openings of $\mathbf{c}_1^0, \mathbf{c}_2^0$ wrt. d and openings of $\mathbf{c}_1^1, \mathbf{c}_2^1$ wrt. $1-d$.

$\mathsf{SCom}_d(\Sigma_{\mathsf{Bind}})$ outputs $\sigma_c = (\mathbf{c}_1^0, \mathbf{c}_2^0, \mathbf{c}_1^1, \mathbf{c}_2^1)$ where $\mathbf{c}_2^0, \mathbf{c}_2^1$ are obtained by randomizing $\mathbf{c}_1^0, \mathbf{c}_1^1$ respectively and, their openings are also computed using openings of $\mathbf{c}_1^0, \mathbf{c}_1^1$ and randomization values. Again, st consists of openings of $\mathbf{c}_1^0, \mathbf{c}_2^0$ wrt. d and openings of $\mathbf{c}_1^1, \mathbf{c}_2^1$ wrt. $1-d$. In this distribution, π^{00}, π^{11} are computed identically as in $\mathsf{OutCom}_{\mathsf{Bind}}$ whereas, proofs π^{01}, π^{10} are both computed by mauling proof π from Σ_{Bind} with respect to different transformations.

Indistinguishability of the distributions follows from the perfect randomizability and equivocability of the commitment scheme, perfect randomizability of RParam and by malleability of the proof system $(\mathsf{TC.Prove}, \mathsf{TC.Verify}, \mathsf{TC.Maul})$ for L_{TC} with respect to the transformation $\mathsf{TC.T}$. □

Similarly, we have the following claim.

Claim 9. *Let $\Sigma_{\mathsf{Hide}} \leftarrow \mathcal{D}_{\mathsf{Hide}}(1^\lambda)$. Let r_1, r_2 be chosen at random and for all $i \in [2]$, let $\mathsf{pp}_i = \mathsf{C.Setup}'(1^\lambda; r_i)$. Then, for all $d \in \{0,1\}$, the following distributions are identical:*

$$\big(\mathsf{SCom}_d(\Sigma_{\mathsf{Hide}})\big) \text{ and } \big(\mathsf{OutCom}_{\mathsf{Hide}}(\mathsf{pp}_1, \mathsf{pp}_2, r_1, r_2, d)\big)$$

We are now ready to describe hybrids $\mathsf{Hyb}'_3, \mathsf{Hyb}'_4$. Hyb'_3 differs from Hyb_3 in the following ways:

- Parameters $(\mathsf{pp}_2^0, \mathsf{pp}_2^1)$ are computed as randomization of $(\mathsf{pp}_1^0, \mathsf{pp}_1^1)$ rather than as fresh parameters.
- For all connecting wires, SCom_d is used instead of $\mathsf{OutCom}_{\mathsf{Bind}}$. All commitments (including to input wires) with respect to $\mathsf{pp}_1^0, \mathsf{pp}_2^0$ (binding parameters) are with respect to wit_0 and all commitments with respect to $\mathsf{pp}_1^1, \mathsf{pp}_2^1$ (binding parameters) are with respect to wit_1.
- For bit-proofs and gate-proofs, instead of using equivocated openings use the inconsistent openings output by SCom_d. Note that in Hyb'_3, we do not have the randomness used in the generation of $\mathsf{pp}_1, \mathsf{pp}_2$ to equivocate the commitments. But we get the openings (distributed identically as Hyb_3) through the sample Σ_{Bind}.

Concretely, Hyb'_3 does the following:

1. Sample $(\mathsf{pp}_1^0, \mathsf{pp}_1^1, \mathbf{c}_0, \mathbf{c}'_0, \mathbf{c}_1, \mathbf{c}'_1, o_0, o'_0, o_1, o'_1, \Pi^0_{\mathsf{TC}}, \Pi^1_{\mathsf{TC}}) \leftarrow \mathcal{D}_{\mathsf{Bind}}(1^\lambda)$.
2. $\underline{\text{Choose randomness } r' \text{ and compute } \mathsf{pp}_2^0 = \mathsf{RParam}(\mathsf{pp}_1^0; r')}$. Compute $\mathsf{pp}_2^1 = \mathsf{OutParam}(\mathsf{pp}_2^0)$.
3. Denote by $\mathsf{layer}_1, \dots, \mathsf{layer}_t$ the t layers of gates in C. For all $j \in [t]$ and for each gate $v_j \in \mathsf{layer}_j$, let $\overrightarrow{\mathsf{pp}}_{v_j} = (\mathsf{pp}_1^0, \mathsf{pp}_1^1)$ if j is odd, else $\overrightarrow{\mathsf{pp}}_{v_j} = (\mathsf{pp}_2^0, \mathsf{pp}_2^1)$ if j is even.
4. For each input wire $k \in [n]$, denote by j the gate for which wire k is an input. For every $b \in \{0,1\}$, choose at random $o^b_{k,j}$ and $\underline{\text{compute } \mathbf{c}^b_{k,j} = \mathsf{C.Commit}(\mathsf{pp}^b_j, w^b_k; o^b_{k,j})}$. Let $\mathbf{c}_k = (\mathbf{c}^0_{k,j}, \mathbf{c}^1_{k,j})$.

   For the output wire wout and for every $b \in \{0,1\}$, if $\mathsf{out} = 1$, $\mathbf{c}^b_{\mathsf{wout},m} = \mathbf{1}$ and if $\mathsf{out} = 0$, $\mathbf{c}^b_{\mathsf{wout},m} = \mathbf{0}$. Let $\mathbf{c}_{\mathsf{wout}} = (\mathbf{c}^0_{\mathsf{wout},m}, \mathbf{c}^1_{\mathsf{wout},m})$.
5. $\underline{\text{For each connecting wire } k \in \{n+1, \dots, n+\ell\} \text{ that connects gates } i, j \in [m],}$
   - If $w^0_k \neq w^1_k$ and $w^0_k = 0$ then
   $$(\sigma^k_c, \sigma^k_\pi, \mathsf{st}^k) \leftarrow \mathsf{SCom}_0(\mathsf{pp}_1^0, \mathsf{pp}_1^1, \mathbf{c}_0, \mathbf{c}'_0, \mathbf{c}_1, \mathbf{c}'_1, o_0, o'_0, o_1, o'_1, \Pi^0_{\mathsf{TC}}, \Pi^1_{\mathsf{TC}}; r', \cdot)$$
   - If $w^0_k \neq w^1_k$ and $w^0_k = 1$ then
   $$(\sigma^k_c, \sigma^k_\pi, \mathsf{st}^k) \leftarrow \mathsf{SCom}_1(\mathsf{pp}_1^0, \mathsf{pp}_1^1, \mathbf{c}_0, \mathbf{c}'_0, \mathbf{c}_1, \mathbf{c}'_1, o_0, o'_0, o_1, o'_1, \Pi^0_{\mathsf{TC}}, \Pi^1_{\mathsf{TC}}; r', \cdot)$$

   Note that we use same r' in $\mathsf{SCom}_0, \mathsf{SCom}_1$ as used in step 2 so that the parameters $(\mathsf{pp}_2^0, \mathsf{pp}_2^1)$ used in $\mathsf{SCom}_0, \mathsf{SCom}_1$ are consistent with step 2.
   - Finally if $w^k_0 = w^k_1$ compute
   $$(\sigma^k_c, \sigma^k_\pi, \mathsf{st}^k) \leftarrow \mathsf{OutCom}(\mathsf{pp}_i, \mathsf{pp}_j, r', w^k_0)$$

   Note here that r' is the randomization factor between $\mathsf{pp}_i, \mathsf{pp}_j$ and that is sufficient for computing the intermediate parameter pp_* required for L_{TC} proofs output by OutCom. See description of $\mathsf{InterParam}$ in Section 5.1.2.

where $\sigma_c^k = (\mathbf{c}_{k,i}^0, \mathbf{c}_{k,j}^0, \mathbf{c}_{k,i}^1, \mathbf{c}_{k,j}^1)$, $\sigma_\pi^k = (\pi_k^{00}, \pi_k^{01}, \pi_k^{10}, \pi_k^{11})$ and where $\mathsf{st}^k = (o_{k,i}^0, o_{k,j}^0, o_{k,i}^1, o_{k,j}^1)$. We will denote $(\sigma_c^k, \sigma_\pi^k)$ by Φ_k.

6. For all $(k,j) \in S$ and $b \in \{0,1\}$, compute

$$\underline{\pi_{\mathsf{bit}}[k,j]^b \leftarrow \mathsf{Bit.Prove}(\mathsf{pp}_j^b, \mathbf{c}_{k,j}^b, o_{k,j}^b)}$$

where $o_{k,j}^b$ is the opening for commitment $\mathbf{c}_{k,j}^b$ as computed in step 2 (for input wires) or as part of st^k output by OutCom (for connecting wires) in step 3. Let $\pi_{\mathsf{bit}}[k,j] = \big(\pi_{\mathsf{bit}}[k,j]^0, \pi_{\mathsf{bit}}[k,j]^1\big)$.

7. For each gate $j \in [m]$, denote by k_1, k_2 the input wires of the gate j and by k_3, k_4 the output wires to gate j. For each $t \in \{3,4\}$ and $b \in \{0,1\}$, compute a gate consistency proof as follows:

$$\underline{\pi_{\mathsf{gate}}^{j,b}[t] \leftarrow \mathsf{N.Prove}\big(\mathsf{pp}_j^b, \{\mathbf{c}_{k_i,j}^b\}_{i\in\{1,2,t\}}, \{w_{k_i}^b, o_{k_i,j}^b\}_{i\in\{1,2,t\}}\big)}$$

Let $\pi_{\mathsf{gate}}^j = \big(\pi_{\mathsf{gate}}^{j,0}[3], \pi_{\mathsf{gate}}^{j,1}[3], \pi_{\mathsf{gate}}^{j,0}[4], \pi_{\mathsf{gate}}^{j,1}[4]\big)$.

8. Let $\Pi = \big[\{\overrightarrow{\mathsf{pp}}_j\}_{j\in[m]}, \{\mathbf{c}_k\}_{k\in[n]}, \{\Phi_k\}_{k\in[\ell]}, \{\pi_{\mathsf{bit}}[i,j]\}_{(i,j)\in S}, \{\pi_{\mathsf{gate}}^j\}_{j\in[m]}, \mathbf{c}_{\mathsf{wout}}\big]$. Finally compute $\Pi' \leftarrow \mathsf{NIWI.Rand}((C, \mathsf{out}), \Pi)$. Output $\big\{(C,\mathsf{out}), \mathsf{wit}_0, \mathsf{wit}_1, \Pi'\big\}$.

Hyb_4' is exactly the same as Hyb_3' except that in step 1, we sample from $\mathcal{D}_{\mathsf{Hide}}$ instead of $\mathcal{D}_{\mathsf{Bind}}$. Concretely, step 1 will be:

Sample $(\mathsf{pp}_1^0, \mathsf{pp}_1^1, \mathbf{c}_0, \mathbf{c}_0', \mathbf{c}_1, \mathbf{c}_1', o_0, o_0', o_1, o_1', \Pi_{\mathsf{TC}}^0, \Pi_{\mathsf{TC}}^1) \leftarrow \mathcal{D}_{\mathsf{Hide}}(1^\lambda)$.

Hyb_4' differs from Hyb_4 in the following ways:

– Parameters $(\mathsf{pp}_2^0, \mathsf{pp}_2^1)$ are computed as randomization of $(\mathsf{pp}_1^0, \mathsf{pp}_1^1)$ rather than as fresh parameters.

– For all connecting wires, SCom_d is used instead of $\mathsf{OutCom}_{\mathsf{Bind}}$. All commitments (including to input wires) with respect to $\mathsf{pp}_1^0, \mathsf{pp}_2^0$ (binding parameters) are with respect to wit_0 and all commitments with respect to $\mathsf{pp}_1^1, \mathsf{pp}_2^1$ (binding parameters) are with respect to wit_1.

– For bit-proofs and gate-proofs, instead of using equivocated openings use the inconsistent openings output by SCom_d.

$\mathsf{Hyb}_3, \mathsf{Hyb}_3'$ are identically distributed by Claim 8, by the hiding property of the commitment and by the perfect randomizability of RParam. Similarly, $\mathsf{Hyb}_4, \mathsf{Hyb}_4'$ are identically distributed by Claim 9, by the hiding property of the commitment and by the perfect randomizability of RParam. Hence, Proposition 13 follows from strong secrecy of L_{TC}. □

□

7.3 Constructing Malleable Proof System for L_{TC}

In this section, we construct the malleable proof system $(\mathsf{TC.Prove}, \mathsf{TC.Verify}, \mathsf{TC.Maul})$ for L_{TC}, with respect to the transformation $\mathsf{TC.T}$ as described in Section 7.1. We also prove that it satisfies weak soundness and satisfies strong secrecy assuming DLIN with Leakage. Recall that

$$L_{\mathsf{TC}} = \Big\{(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2) \mid \exists\ (b, \mathsf{pp}_*, o_1, o_2) \text{ s.t.}$$
$$\{\mathbf{c}_i = \mathsf{C.Commit}(\mathsf{pp}_i, b; o_i)\}_{i\in[2]} \ \wedge\ \big(\mathsf{ValidInter}(\mathsf{pp}_1, \mathsf{pp}_2, \mathsf{pp}_*) = 1\big)\Big\}$$

We now describe the proof system ($\mathsf{TC.Prove}, \mathsf{TC.Verify}$) for L_{TC}. At a high level, a proof for $(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2) \in L_{\mathsf{TC}}$ is computed by first converting the commitment $\mathbf{c}_1$ with respect to pp_1 to a commitment $\mathbf{c}_*$ with respect to pp_2 (using the intermediate parameter pp_* which is part of the witness). The next step is to prove that the homomorphically computed commitment $(\mathbf{c}_2 \cdot \mathbf{c}_*)$ is a commitment to 0 or 2, which can be reduced to an L_{Lin} statement with respect to pp_2.

Let ($\mathsf{Lin.Prove}, \mathsf{Lin.Verify}, \mathsf{Lin.Transform}$) be the NIWI proof system for $L_{\mathsf{Lin}}[\mathsf{pp}]$ from Section 5.2. Concretely, the proof system for L_{TC} is as follows.

$\mathsf{TC.Prove}\big((\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2), (b, \mathsf{pp}_*, o_1, o_2)\big)$: For $i \in [2]$, parse $\mathsf{pp}_i = [f_i, h_i, g_i, u_i, v_i, w_i]$. Parse $\mathsf{pp}_* = [f_*, h_*, g_*, u_*, v_*, w_*]$. Without loss of generality, let $(f_*, h_*, g_*) = (f_2, h_2, g_2)$. For $i \in [2]$, parse $\mathbf{c}_i = (c_1^i, c_2^i, c_3^i)$, parse $o_i = (r_i, s_i)$, and compute

$$\mathbf{c}_* = (u_*^b f_2^{r_1}, v_*^b h_2^{s_1}, w_*^b g_2^{r_1+s_1}).$$

Compute $\mathbf{A}, \mathbf{B}$ as follows:

$$\mathbf{A} = \Big(c_1^* \cdot c_1^2, c_2^* \cdot c_2^2, c_3^* \cdot c_3^2\Big),\ \mathbf{B} = \Big(\frac{c_1^* \cdot c_1^2}{u_* u_2}, \frac{c_2^* \cdot c_2^2}{v_* v_2}, \frac{c_3^* \cdot c_3^2}{w_* w_2}\Big)$$

Compute $\Pi_{\mathsf{Lin}} = \mathsf{Lin.Prove}\big((f_2, h_2, g_2), (\mathbf{A}, \mathbf{B}), (r, s, u)\big)$ and where $(r, s, u) = (r_1+r_2, s_1+s_2, (r_1+s_1+r_2+s_2))$.

Finally output $\Pi_{\mathsf{TC}} = [\mathsf{pp}_*, \mathbf{c}_*, \Pi_{\mathsf{Lin}}]$.

$\mathsf{TC.Verify}\big((\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2), \Pi_{\mathsf{TC}}\big)$: Parse $\Pi_{\mathsf{TC}} = [\mathsf{pp}_*, \mathbf{c}_*, \Pi_{\mathsf{Lin}}]$. Make the following checks:

- Check that $\mathsf{ValidInter}(\mathsf{pp}_1, \mathsf{pp}_2, \mathsf{pp}_*) = 1$.
- Check $e(c_1^*, f_1) = e(c_1^1, f_2)$, $e(c_2^*, h_1) = e(c_2^1, h_2)$ and $e(c_3^*, g_1) = e(c_3^1, g_2)$.

Finally check that $\mathsf{Lin.Verify}((f_2, h_2, g_2), \mathbf{A}, \mathbf{B}, \Pi_{\mathsf{Lin}}) = 1$ where $\mathbf{A} = (c_1^* \cdot c_1^2, c_2^* \cdot c_2^2, c_3^* \cdot c_3^2)$ and $\mathbf{B} = (\frac{c_1^* \cdot c_1^2}{u_* u_2}, \frac{c_2^* \cdot c_2^2}{v_* v_2}, \frac{c_3^* \cdot c_3^2}{w_* w_2})$.

It is easy to see that completeness holds for all parameters. We now prove *weak soundness*, namely that this proof system is sound if both parameters are binding.

Proposition 14. *For $i \in [2]$, let $\mathsf{pp}_i = [f_i, h_i, g_i, u_i, v_i, w_i]$ and let $\mathbf{c}_i = (c_1^i, c_2^i, c_3^i)$. Let Π_{TC} be a proof for $(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2) \in L_{\mathsf{TC}}$. If,*

$$\mathsf{TC.Verify}\big((\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2), \Pi_{\mathsf{TC}}\big) = 1\ \wedge\ \big(\{\mathsf{pp}_i \in \mathsf{C.Setup}(1^\lambda)\}_{i \in [2]}\big)\ \textit{then,}\ (\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2) \in L_{\mathsf{TC}}$$

Proof. If $\mathsf{TC.Verify}\big((\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2), \Pi_{\mathsf{TC}}\big) = 1$, bilinear checks ensure that there exists $\eta, R_1, S_1 \in \mathbb{Z}_p^*$ such that $(u_*, v_*, w_*) = (f_2^{R_1}, h_2^{S_1}, g_2^{R_1+S_1+\eta})$ and $(u_1, v_1, w_1) = (f_1^{R_1}, h_1^{S_1}, g_1^{R_1+S_1+\eta})$. Also there exists b, r_1, s_1 such that $\mathbf{c}_* = (u_*^b f_2^{r_1}, v_*^b h_2^{s_1}, w_*^b g_2^{r_1+s_1})$ and $\mathbf{c}_1 = (u_1^b f_1^{r_1}, v_1^b h_1^{s_1}, w_1^b g_1^{r_1+s_1})$. By perfect soundness of ($\mathsf{Lin.Prove}, \mathsf{Lin.Verify}$),

$$\Pr\big[\exists\ (a_1, a_2, a_3)\ \text{ s.t. }\ (a_1 + a_2 = a_3)\ \wedge\ \big(\mathbf{A} = (f^{a_1}, h^{a_2}, g^{a_3})\ \vee\ \big(\mathbf{B} = (f^{a_1}, h^{a_2}, g^{a_3})\big)\big] = 1$$

where $\mathbf{A} = (c_1^* \cdot c_1^2, c_2^* \cdot c_2^2, c_3^* \cdot c_3^2)$ and $\mathbf{B} = (\frac{c_1^* \cdot c_1^2}{u_* u_2}, \frac{c_2^* \cdot c_2^2}{v_* v_2}, \frac{c_3^* \cdot c_3^2}{w_* w_2})$.

Also since $\mathsf{pp}_i \in \mathsf{C.Setup}(1^\lambda)$ for $i \in [2]$, there exists $r_2 = a_1 - r_1, s_2 = a_2 - s_1$ such that, $\mathbf{c}_2 = (u_2^b f_2^{r_2}, v_2^b h_2^{s_2}, w_2^b g_2^{r_2+s_2})$. Hence for $b = 0$, $\mathbf{A}$ is linear and for $b = 1$, $\mathbf{B}$ is linear, and we conclude that the proposition follows.

□

7.3.1 Malleability of the L_{TC} Proof System

Let $(\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2) \in L_{\mathsf{TC}}$ and let Π_{TC} be the corresponding proof as described above. Recall that $\mathsf{TC.Transform}$ outputs a randomized instance $(\mathbf{c}'_1, \mathbf{c}'_2, \mathsf{pp}'_1, \mathsf{pp}'_2) \in L_{\mathsf{TC}}$ (as described in Section 7.1). The transformation is defined by randomness $\{o'_k, r^k_{\mathsf{pp}}\}_{k\in[2]}$.

For $r_{\mathsf{pp}} = (x', y', z', R', S')$, and for any proof for $(\mathbf{A}, \mathbf{B}) \in L_{\mathsf{Lin}}[\mathsf{pp}]$ where $\Pi = [\pi_{11}, \ldots, \pi_{23}]$, define

$$\mathsf{ChangeGen}(\Pi, r_{\mathsf{pp}}) \triangleq [\pi_{11}^{x'}, \pi_{12}^{y'}, \pi_{13}^{z'}, \pi_{21}^{x'}, \pi_{22}^{y'}, \pi_{23}^{z'}].$$

$\mathsf{TC.Maul}\big((\mathbf{c}_1, \mathbf{c}_2, \mathsf{pp}_1, \mathsf{pp}_2), \{o'_k, r^k_{\mathsf{pp}}\}_{k\in[2]}, \Pi_{\mathsf{TC}}\big)$ works as follows:

1. Parse $\Pi_{\mathsf{TC}} = [\mathsf{pp}_*, \mathbf{c}_*, \Pi_{\mathsf{Lin}}]$. For $k \in [2]$, parse $r^k_{\mathsf{pp}} = (x'_k, y'_k, z'_k, R'_k, S'_k)$ and parse $o'_k = (q_k, t_k)$.
2. Randomize pp_* by computing $\mathsf{pp}''_* = (u''_*, v''_*, w''_*) = (u_* f_2^{R'_1}, v_* h_2^{S'_1}, w_* g_2^{R'_1+S'_1})$.
3. Randomize commitment $\mathbf{c}_*$ by computing $\mathbf{c}''_* = (\mathbf{c}^*_1 f_2^{q_1}, \mathbf{c}^*_2 h_2^{t_1}, \mathbf{c}^*_3 g_2^{q_1+t_1})$.
4. Π_{Lin} is a linearity proof for $(\mathbf{A}, \mathbf{B})$ where $\mathbf{A} = (c^*_1 \cdot c^2_1, c^*_2 \cdot c^2_2, c^*_3 \cdot c^2_3)$ and $\mathbf{B} = (\frac{c^*_1 \cdot c^2_1}{u_* u_2}, \frac{c^*_2 \cdot c^2_2}{v_* v_2}, \frac{c^*_3 \cdot c^2_3}{w_* w_2})$. Let $(\mathbf{A}', \mathbf{B}')$ be the transformed L_{Lin} statement with respect to the randomized commitments. Namely, $(\mathbf{A}', \mathbf{B}') = \mathsf{Lin.Transform}(\mathsf{pp}, \mathbf{A}, \mathbf{B}; (q_1 + q_2, t_1 + t_2, q_1 + q_2 - R'_1 - R'_2, t_1 + t_2 - S'_1 - S'_2))$. Compute
$$\Pi''_{\mathsf{Lin}} \leftarrow \mathsf{Lin.Maul}(\mathsf{pp}, (\mathbf{A}, \mathbf{B}), \mathbf{r}, \mathbf{s}, \Pi_{\mathsf{Lin}})$$
where $\mathbf{r} = (q_1 + q_2, t_1 + t_2)$ and $\mathbf{s} = (q_1 + q_2 - R'_1 - R'_2, t_1 + t_2 - S'_1 - S'_2)$.
5. Compute $\mathbf{c}'_* = \mathsf{ChangeCom}(\mathbf{c}''_*, r^2_{\mathsf{pp}})$, $\Pi'_{\mathsf{Lin}} = \mathsf{ChangeGen}(\Pi''_{\mathsf{Lin}}, r^2_{\mathsf{pp}})$, and $\mathsf{pp}'_* = \big((u''_*)^{x'_2}, (v''_*)^{y'_2}, (w''_*)^{z'_2}\big)$
6. Finally output $\Pi'_{\mathsf{TC}} = [\mathsf{pp}'_*, \mathbf{c}'_*, \Pi'_{\mathsf{Lin}}]$.

Proposition 15. *The proof system* $(\mathsf{TC.Prove}, \mathsf{TC.Verify}, \mathsf{TC.Maul})$ *is a malleable proof system for* L_{TC} *as per Definition 5, with respect to the transformation* $\mathsf{TC.Transform}$.

Proof. Follows from malleability of $(\mathsf{Lin.Prove}, \mathsf{Lin.Verify}, \mathsf{Lin.Maul})$.

□

7.3.2 Strong Secrecy from the DLIN with Leakage Assumption

In Section 7.1, we described the strong secrecy property required from the NIWI proof system ($\mathsf{TC.Prove}$, $\mathsf{TC.Verify}$). In particular, strong secrecy states that the distributions $\mathcal{D}_{\mathsf{Hide}}, \mathcal{D}_{\mathsf{Bind}}$ are indistinguishable.

We now show that strong secrecy for the proof system $(\mathsf{TC.Prove}, \mathsf{TC.Verify}, \mathsf{TC.Maul})$ constructed above, follows from the Strong NIWI for $L_{\mathsf{Lin}}[\mathsf{pp}]$ with respect to specific distributions as described in Section 5.2.3. Strong NIWI for $L_{\mathsf{Lin}}[\mathsf{pp}]$ in turn, follows from DLIN with Leakage (as per Proposition 5).

Recall that strong NIWI for $L_{\mathsf{Lin}}[\mathsf{pp}]$ states that:

$$\{\mathsf{pp}, (\mathbf{A}_0, \mathbf{B}_0), \pi_0\} \approx \{\mathsf{pp}, (\mathbf{A}_1, \mathbf{B}_1), \pi_1\}$$

where $\mathbf{A}_b = (f^{a_1}, h^{a_2}, g^{a_3-b})$ for $a_1, a_2 \leftarrow \mathbb{Z}_p^*$ and $a_3 = a_1 + a_2$, where $\mathbf{B}_b = (f^{a_1}, h^{a_2}, g^{a_3-b+1})$, and where $\pi_b \leftarrow \mathsf{Lin.Prove}(\mathsf{pp}, (\mathbf{A}_b, \mathbf{B}_b), (a_1, a_2, a_3))$.

We now describe a reduction S that takes as input $(\mathsf{pp}, \mathbf{A}, \mathbf{B}, \pi)$ where $(\mathbf{A}, \mathbf{B}, \pi) = (\mathbf{A}_0, \mathbf{B}_0, \pi_0)$ or $(\mathbf{A}, \mathbf{B}, \pi) = (\mathbf{A}_1, \mathbf{B}_1, \pi_1)$ as described above, and does the following:

1. Parse $\mathsf{pp} = [p, \mathbb{G}, \mathbb{G}_T, e, g_p, f, h, g]$. Denote by $(u, v, w') = \mathbf{A}$ and $(u, v, w) = \mathbf{B}$. Let $\mathsf{pp}_D = [f, h, g, u, v, w]$ and $\mathsf{pp}'_D = [f, h, g, u, v, w']$.
2. For every $d \in \{0, 1\}$, choose r_d, s_d, r'_d, s'_d at random and compute commitments $\mathbf{c}_d = (u^d f^{r_d}, v^d h^{s_d}, w^d g^{r_d+s_d}))$ and $\mathbf{c}'_d = (u^{1-d} f^{r'_d}, v^{1-d} h^{s'_d}, (w')^{1-d} g^{r'_d+s'_d}))$. Let $o_d = (r_d, s_d)$ and $o'_d = (r'_d, s'_d)$.
3. For every $d \in \{0, 1\}$, compute $\Pi^d_{\mathsf{Lin}} \leftarrow \mathsf{Lin.Maul}(\mathsf{pp}, \mathbf{A}, \mathbf{B}, (t_d, z_d), \pi^{-1})$ where $t = (r_d + r'_d), z = (s_d + s'_d)$. Let $\Pi^b_{\mathsf{TC}} = [\mathsf{pp}'_D, \mathbf{c}'_d, \Pi^d_{\mathsf{Lin}}]$.

Output $(\mathsf{pp}_D, \mathsf{pp}'_D, \mathbf{c}_0, \mathbf{c}'_0, \mathbf{c}_1, \mathbf{c}'_1, o_0, o'_0, o_1, o'_1, \Pi^0_{\mathsf{TC}}, \Pi^1_{\mathsf{TC}})$.

Note that o_d, o'_d are openings with respect to d and $1 - d$ respectively. Let $\mathbf{c}_d = (c^d_1, c^d_2, c^d_3)$ and $\mathbf{c}'_d = (c^{d'}_1, c^{d'}_2, c^{d'}_3)$. The main observation is that when $\mathbf{c}_d$ and $\mathbf{c}'_d$ commit to different bits, then the L_{Lin} statement in Π_{TC} given by

$$\left(c^d_1 \cdot c^{d'}_1, c^d_2 \cdot c^{d'}_2, c^d_3 \cdot c^{d'}_3\right), \left(\frac{c^d_1 \cdot c^{d'}_1}{u^2}, \frac{c^d_2 \cdot c^{d'}_2}{v^2}, \frac{c^d_3 \cdot c^{d'}_3}{ww'}\right)$$

is a transformed instance of $(\mathbf{A}^{-1}, \mathbf{B})$ for $d = 1$ and $(\mathbf{A}, \mathbf{B}^{-1})$ for $d = 0$ where $\mathbf{A} = (u, v, w')$ and $\mathbf{B} = (u, v, w)$, and where the instance is transformed by (t_d, z_d) for $t = (r_d + r'_d), z = (s_d + s'_d)$. Recall from Remark 3 that given linearity proof for $(\mathbf{A}, \mathbf{B})$, it is possible to compute linearity proof for $(\mathbf{A}^{-1}, \mathbf{B})$ or $(\mathbf{A}, \mathbf{B}^{-1})$ by inverting all the terms in proof π to obtain the proof π^{-1}.

It is easy to see that when reduction S gets as input $(\mathsf{pp}, \mathbf{A}_0, \mathbf{B}_0, \pi_0)$, it outputs a sample from $\mathcal{D}_{\mathsf{Hide}}$ since $\mathbf{B}_0 \in \mathsf{C.Setup}'(1^\lambda)$, and when it gets as input $(\mathsf{pp}, \mathbf{A}_1, \mathbf{B}_1, \pi_1)$, it outputs a sample from $\mathcal{D}_{\mathsf{Bind}}$ since $\mathbf{B}_1 \in \mathsf{C.Setup}'(1^\lambda)$.

8 Commit-and-Compute Paradigm

In this section, we define and instantiate *commit-and-compute* proof systems. Our motivation is the setting where users commit to their private data (say on a public ledger), and then may wish to prove statements about their private committed data. We want the ability to evaluate an arbitrary function (in the form of a circuit) over individual proofs to obtain proofs on inferred statements about the committed data.

We first define a commit-and-compute NIZK and NIWI proof system. We then use the construction ideas in the previous sections to achieve this notion.

8.1 Definition of Commit-and-Compute

Definition 15 (Commit-and-Compute NIZK Proofs)**.** A commit-and-compute NIZK proof system consists of the PPT algorithms ($\mathsf{CnC.Setup}$, $\mathsf{CnC.Commit}$, $\mathsf{CnC.Prove}$, $\mathsf{CnC.Verify}$, $\mathsf{CnC.Eval}$) with the following input-output behavior:

$\mathsf{CRS} \leftarrow \mathsf{CnC.Setup}(1^\lambda)$: The setup algorithm takes as input the security parameter and outputs a common random string CRS.

$\mathbf{c} \leftarrow \mathsf{CnC.Commit}(\mathsf{CRS}, d; r)$: The commit algorithm takes as input the CRS, bit d, randomness r, and outputs a commitment $\mathbf{c}$. We denote a vector of commitments $(\mathbf{c}_1, \ldots, \mathbf{c}_n)$ by $\mathbf{com}$.

We are now ready to define our language L_{COM}:

$$L_{\mathsf{COM}} = \big\{(C, \mathbf{com}, b) \mid \exists (\mathbf{w}, \mathbf{r}) \text{ s.t.} \big(C(w_1, \ldots, w_n) = b\big) \;\wedge\; \big(\{\mathbf{c}_i = \mathsf{CnC.Commit}(\mathsf{CRS}, w_i; r_i)\}_{i \in [n]}\big)\big\}$$

$\Pi \leftarrow \mathsf{CnC.Prove}(\mathsf{CRS}, (C, \mathbf{com}, b), (\mathbf{w}, \mathbf{r}))$: The prove algorithm takes as input the CRS, an instance $(C, \mathbf{com}, b)$ along with its witness $(\mathbf{w}, \mathbf{r})$ and outputs a proof Π.

$0/1 \leftarrow \mathsf{CnC.Verify}(\mathsf{CRS}, (C, \mathbf{com}, b), \Pi)$: The verification algorithm takes as input the CRS, an instance $(C, \mathbf{com}, b)$ and a proof Π, and outputs a boolean value indicating success or failure.

$((C, \mathbf{com}, b), \Pi) \leftarrow \mathsf{CnC.Eval}(\mathsf{CRS}, \{(C_i, \mathbf{com}_i, b_i), \Pi_i\}_{i=1}^{K}, C')$: The evaluation algorithm takes as input the CRS, K instances $\{(C_i, \mathbf{com}_i, b_i)\}_{i=1}^{K}$ of L_{COM} with their proofs $\{\Pi_i\}_{i=1}^{K}$ and a circuit C', and outputs the composed instance $(C, \mathbf{com}, b)$ and a corresponding proof Π.

Composing for L_{COM} instances: Recall that in Section 4, we defined the $\mathsf{Compose}()$ operation that takes as input $\{(C_i, b_i)\}_{i=1}^{k}$ where $C_i : \{0,1\}^{n_i} \to \{0,1\}$ and a circuit $C' : \{0,1\}^k \to \{0,1\}$, and outputs (C, b) where $C : \{0,1\}^N \to \{0,1\}$ for $N = n_1 + \cdots + n_k$. In this case, all the circuits $\{C_i\}_{i=1}^{k}$ were with respect to independent inputs.

We now consider the case where different circuits $\{C_i\}_{i=1}^{k}$ may have overlapping inputs. In particular, given k instances $\{(C_i, \mathbf{com}_i, b_i)\}_{i \in [k]} \in L_{\mathsf{COM}}$, we want to support the case that different $\mathbf{com}_i$ given as input to $\mathsf{CnC.Eval}$ are overlapping; namely, there is a commitment $\mathbf{c}$ such that $\mathbf{c}$ is part of $\mathbf{com}_i$ as well as part of $\mathbf{com}_j$ for some $i, j \in [k]$. We define the composed $\mathbf{com}$ as the sequence $(\mathbf{com}_1, \ldots, \mathbf{com}_k)$ where we delete a commitment $\mathbf{c}$ if it has previously appeared. We formalize the compose operation as follows:

$\mathsf{Compose}(\{(C_i, \mathbf{com}_i, b_i)\}_{i=1}^{k}, C')$: Let $\mathbf{com}$ be the vector of commitments obtained as follows: Instantiate $\mathbf{com}$ with $\mathbf{com}_1$. For each commitment $\mathbf{c}$ in subsequent $\mathbf{com}_i$ for $i \in \{2, \ldots, k\}$, append $\mathbf{c}$ to $\mathbf{com}$ only if the string $\mathbf{c}$ does not already appear in $\mathbf{com}$. Hence we finally obtain $\mathbf{com}$ such that each commitment in $\mathbf{com}_1, \ldots, \mathbf{com}_k$ appears in $\mathbf{com}$ exactly once. Thus, $\mathbf{com}$ is the union of all the commitments in $\mathbf{com}_1, \ldots, \mathbf{com}_k$. Let $M = |\mathbf{com}|$.

We will now think of each $C_i : \{0,1\}^M \to \{0,1\}$ where C_i might use only a part of the M inputs. The compose algorithm outputs circuit $C : \{0,1\}^M \to \{0,1\}$ such that for all $\mathbf{w} \in \{0,1\}^M$, we have $C(\mathbf{w}) = C'(C_1(\mathbf{w}), \ldots, C_k(\mathbf{w}))$ and $b = C'(b_1, \ldots, b_k)$.
If $\mathsf{Compose}()$ is given as input witnesses $(\mathbf{w_i}, \mathbf{r_i})$ for the k instances then it outputs the composed witness $(\mathbf{w}, \mathbf{r})$ corresponding to commitments in the vector $\mathbf{com}$, where $\mathbf{com}$ is computed as explained before.

We require the following properties from Commit-and-Compute NIZK proof system:

- **NIZK:** $(\mathsf{CnC.Setup}, \mathsf{CnC.Prove}, \mathsf{CnC.Verify})$ is a non-interactive zero-knowledge proof system for L_{COM} as per Definition 1.

- **Completeness of Eval:** We require that for all non-uniform PPT $\mathcal{A}$ and for all $\lambda \in \mathbb{N}$,

$$\Pr\left[\begin{array}{c} \mathsf{CRS} \leftarrow \mathsf{CnC.Setup}(1^\lambda) \;;\; (\{(C_i, \mathbf{com}_i, b_i, \Pi_i)\}_{i=1}^{k}, C') \leftarrow \mathcal{A}(\mathsf{CRS}) \;; \\ ((C, \mathbf{com}, b), \Pi) \leftarrow \mathsf{CnC.Eval}(\mathsf{CRS}, \{(C_i, \mathbf{com}_i, b_i), \Pi_i\}_{i=1}^{k}, C') \;: \\ \big(\mathsf{Valid}(C') = 0\big) \;\vee\; \big(\exists\; i \in [k] \text{ s.t.} \mathsf{CnC.Verify}(\mathsf{CRS}, (C_i, \mathbf{com}_i, b_i), \Pi_i) = 0\big) \;\vee \\ \big((\mathsf{CnC.Verify}(\mathsf{CRS}, (C, \mathbf{com}, b), \Pi) = 1) \;\wedge\; (C, \mathbf{com}, b) = \mathsf{Compose}(\{(C_i, \mathbf{com}_i, b_i)\}_{i=1}^{k}, C')\big) \end{array}\right] = 1$$

where $\mathsf{Valid}(C') = 1$ if and only if $C' : \{0,1\}^k \to \{0,1\}$.

– **Unlinkability:** For all PPT adversaries $\mathcal{A}$, there exists a negligible function ν such that for all λ, the probability that $\mathsf{bit}' = \mathsf{bit}$ in the following game is at most $1/2 + \nu(\lambda)$:

  <u>GAME$_{\mathsf{Eval}}$:</u>

  1. $\mathsf{CRS} \leftarrow \mathsf{CnC.Setup}(1^\lambda)$.
  2. $(\mathsf{state}, \{((C_i, \mathbf{com}_i, b_i), (\mathbf{w}_i, \mathbf{r}_i), \Pi_i)\}_{i=1}^k, C') \leftarrow \mathcal{A}(\mathsf{CRS})$
  3. Choose $\mathsf{bit} \leftarrow \{0,1\}$. If for any $i \in [k]$, $\mathsf{CnC.Verify}(\mathsf{CRS}, (C_i, \mathbf{com}_i, b_i), \Pi_i) \neq 1$ or $\big((C_i, \mathbf{com}_i, b_i), (\mathbf{w}_i, \mathbf{r}_i)\big) \notin R_{\mathsf{COM}}$, then output $\perp$.
  4. If $\mathsf{bit} = 0$ then $((C, \mathbf{com}, b), \Pi) \leftarrow \mathsf{CnC.Eval}(\mathsf{CRS}, \{(C_i, \mathbf{com}_i, b_i), \Pi_i\}_{i=1}^k, C')$.
     If $\mathsf{bit} = 1$ then $\Pi \leftarrow \mathsf{CnC.Prove}(\mathsf{CRS}, (C, \mathbf{com}, b), (\mathbf{w}, \mathbf{r}))$ where $((C, \mathbf{com}, b), (\mathbf{w}, \mathbf{r})) \leftarrow \mathsf{Compose}$ $(\{(C_i, \mathbf{com}_i, b_i), \Pi_i, C', (\mathbf{w}_i, \mathbf{r}_i)\}_{i=1}^k)$. Send (C, b, Π) to $\mathcal{A}$.
  5. $\mathsf{bit}' \leftarrow \mathcal{A}(\mathsf{state}, (C, b, \Pi))$.

Definition 16 (Commit-and-Compute NIWI Proofs)**.** $(\mathsf{CnC.Commit}, \mathsf{CnC.Prove}, \mathsf{CnC.Verify}, \mathsf{CnC.Eval})$ is a Commit-and-Compute NIWI if it has the same description as in Definition 15 where $\mathsf{CRS} = 1^\lambda$. Additionally, $(\mathsf{CnC.Prove}, \mathsf{CnC.Verify})$ is a NIWI proof system for L_{COM} as per Definition 2, and it also satisfies the completeness of evaluation and unlinkability properties as in Definition 15.

8.2 Construction Overview

Our commit-and-compute (NIZK or NIWI) proof system is very similar to that of a fully homomorphic (NIZK or NIWI) proof system. The main difference is that there is an explicit commitment vector $\mathbf{com}$ as part of the instance, and the proofs are with respect to the values committed in this specific vector.

Recall that in our constructions, an FH NIZK as well as FH NIWI proof for $(C, b) \in L_{\mathcal{U}}$ contains commitments to all the wire values in C. One idea is to use the commitments in the instance $\mathbf{com}$ directly in the proof for $(C, \mathbf{com}, b)$. However if we do that, it will make an evaluated proof distinguishable from a fresh proof when circuits share input variables.

For example, let $(C_1, \mathbf{com}_1, b) \in L_{\mathsf{COM}}$ and $(C_2, \mathbf{com}_2, b) \in L_{\mathsf{COM}}$ such that circuits C_1, C_2 share some input variable. Let $\mathbf{c}$ be the commitment corresponding to the common input such that $\mathbf{c}$ is part of both $\mathbf{com}_1$ and $\mathbf{com}_2$. An evaluated proof for $C_1 \wedge C_2$ will contain the commitment $\mathbf{c}$ twice (the proofs for C_1 and C_2 will each contain $\mathbf{c}$), whereas a fresh proof for $C_1 \wedge C_2$ will contain the commitment $\mathbf{c}$ only once.

We deal with this issue by keeping the commitments $\mathbf{com}$ in the instance separate from the commitments in the proof. We compute the FH NIZK or NIWI proof for $(C, b) \in L_{\mathcal{U}}$ as before. We then add proofs of consistency between the values in $\mathbf{com}$ and the commitments to the input values in the proof.

Commit-and-Compute NIZK Proofs. We want to compute a proof for $(C, \mathbf{com}, b) \in L_{\mathsf{COM}}$. As described above, we first compute a FH NIZK proof Π for $(C, b) \in L_{\mathcal{U}}$ with witness $(\mathbf{w}, \mathbf{r})$. We additionally need to prove consistency between commitments $\{\mathsf{com}_1, \ldots, \mathsf{com}_t\}$ part of $\mathbf{com}$, and the commitments $\{\mathbf{c}_1, \ldots, \mathbf{c}_t\}$ to the input values. Namely, for each $i \in [t]$, we need to prove that com_i and $\mathbf{c}_i$ commit to the same value.

This is done as follows: Using the homomorphic properties of the commitment, we can prove the statement that the commitment $\mathsf{com}_i \cdot \mathbf{c}_i$ is either a commitment to 0 or a commitment to 2. This can be reduced to a statement of $L_{\mathsf{Lin}}[\mathsf{pp}]$ where $\mathsf{pp} = \mathsf{CRS}$, similar to the reduction in Bit Proofs and Gate Proofs of FH NIZK. Our final commit-and-compute NIZK proof for $(C, \mathbf{com}, b)$ will consist of an FH NIZK proof Π for $(C, b) \in L_{\mathcal{U}}$ along with t WI proofs that for each $i \in [t]$, com_i and $\mathbf{c}_i$ commit to the same value. The completeness of evaluation and unlinkability follow from the corresponding properties of the underlying FH NIZK and malleability properties of $L_{\mathsf{Lin}}[\mathsf{pp}]$.

Commit-and-Compute NIWI Proofs. We start be explaining how we generate the commitments in this setting. Note that since we have no CRS in a NIWI, we need to slightly modify our commitment scheme to be without any public parameters. Similar to our FH NIWI construction, we choose two parameters

$(\mathsf{pp}^0, \mathsf{pp}^1)$ such that one of them is verifiably binding and commit with respect to both the parameters. Thus, our **com** will be of the form $\mathbf{com} = (\mathsf{pp}^0, \mathsf{pp}^1, (\mathsf{com}_1^0, \mathsf{com}_1^1), \ldots, (\mathsf{com}_n^0, \mathsf{com}_n^1))$.

Again, we first compute FH NIWI proof Π for $(C, b) \in L_{\mathcal{U}}$. We also need to add consistency proofs with respect to **com** and commitments in the FH NIWI proof. Consider an input wire k to circuit C; the proof Π contains $(\mathsf{pp}_k^0, \mathsf{pp}_k^1, \mathbf{c}_k^0, \mathbf{c}_k^1)$ corresponding to wire k. Proving consistency between these and some $(\mathsf{pp}^0, \mathsf{pp}^1, \mathsf{com}_i^0, \mathsf{com}_i^1) \in \mathbf{com}$ boils down to proving four L_{TC} statements exactly as in the procedure OutCom (described in Section 7.1). Our final commit-and-compute NIWI proof for $(C, \mathbf{com}, b)$ will consist of an FH NIWI proof Π for $(C, b) \in L_{\mathcal{U}}$ along with L_{TC} proofs for consistency of commitments. The completeness of evaluation and unlinkability follows from the corresponding properties of the underlying FH NIWI and malleability properties of L_{TC}.

Thus, commit-and-compute NIZK and NIWI proofs can be directly instantiated using our FH NIZK and FH NIWI constructions and its building blocks, and we get the following theorems.

Theorem 5. *There exists Commit-and-Compute NIZK proof system as per Definition 15, assuming DLIN.*

Theorem 6. *There exists a commit-and-compute NIWI proof system as per Definition 16, assuming DLIN with Leakage.*

References

[ACJ17] Prabhanjan Ananth, Aloni Cohen, and Abhishek Jain. Cryptography with Updates. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 445–472. Springer, 2017.

[AGM18] Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. Non-interactive zero-knowledge proofs for composite statements. In *IACR Cryptology ePrint Archive*, 2018.

[AGP14] Prabhanjan Ananth, Vipul Goyal, and Omkant Pandey. Interactive proofs under continual memory leakage. In *International Cryptology Conference*, pages 164–182. Springer, 2014.

[AN11] Tolga Acar and Lan Nguyen. Homomorphic proofs and applications. `https://www.microsoft.com/en-us/research/wp-content/uploads/2011/03/rac.pdf`, 2011.

[BCC+09a] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In *Advances in Cryptology-CRYPTO 2009*, pages 108–125. Springer, 2009.

[BCC+09b] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In *Advances in Cryptology-CRYPTO 2009*, pages 108–125. Springer, 2009.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 111–120. ACM, 2013.

[BCG+17] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: optimizations and applications. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2105–2122. ACM, 2017.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Guiseppe Persiano. Non-interactive zero-knowledge. *SIAM Journal of Computing*, 20(6):1084–1118, 1991.

[BF11] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 149–168. Springer, 2011.

[BGI$^+$18] Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of homomorphic secret sharing. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 94. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[BOV05] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. *IACR Cryptology ePrint Archive*, 2005:365, 2005.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871, 2014.

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–300. Springer, 2012.

[CKLM13a] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable signatures: Complex unary transformations and delegatable anonymous credentials. http://eprint.iacr.org/2013/179, 2013.

[CKLM13b] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Succinct malleable nizks and an application to compact shuffles. In *Theory of Cryptography*, pages 100–119. Springer, 2013.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 283–293. IEEE, 2000.

[Gen09] Craig Gentry. A fully homomorphic encryption scheme [ph. d. thesis]. *International Journal of Distributed Sensor Networks, Stanford University*, 2009.

[Gol00] Oded Goldreich. *Foundations of Cryptography: Basic Tools.* Cambridge University Press, New York, NY, USA, 2000.

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for nizk. In *CRYPTO*, volume 4117, pages 97–111. Springer, 2006.

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for np. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 339–358. Springer, 2006.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 469–477. ACM, 2015.

[KW18] Sam Kim and David J Wu. Multi-theorem preprocessing nizks from lattices. In *Annual International Cryptology Conference*, pages 733–765. Springer, 2018.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[NT16] Assa Naveh and Eran Tromer. Photoproof: Cryptographic image authentication for any set of permissible transformations. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 255–271. IEEE, 2016.

[oSBS15] Conference of State Bank Supervisors. State regulatory requirements for virtual currency activities. *CSBS Model Regulatory Framework*, September, 2015.

[RAD78] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *Theory of Cryptography Conference*, pages 1–18. Springer, 2008.

A Bilinear Generic Group Model

Consider groups $\mathbb{G}, \mathbb{G}_T$ of prime order q and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. One popular method to justify computational assumptions on $(\mathbb{G}, \mathbb{G}_T, e)$ is by showing that the assumption holds unconditionally in the bilinear generic group model.

The bilinear generic group model is an idealized model where the computationally unbounded adversary is given access to randomly chosen encodings (called "handles") of elements in the group. The handles themselves reveal no information about the group elements they are associated with; this means that the adversary cannot perform any computation on the associated group elements using the handles alone. However, the adversary is given access to an oracle $\mathcal{O}$ that performs bilinear computations on the group elements and also allows the adversary to test if a handle corresponds to the unit element in the group.

In more detail, suppose the bilinear assumption is of the form $\mathcal{D}_0 \approx_c \mathcal{D}_1$, where $\mathcal{D}_0$ and $\mathcal{D}_1$ are distributions over the group elements in $\mathbb{G}$. For $b \in \{0, 1\}$, suppose $\mathcal{D}_b$ generates an N-tuple of group elements $g^{\xi_1^b(x_1,\ldots,x_k)}, \ldots, g^{\xi_N^b(x_1,\ldots,x_k)}$, where g is a generator of $\mathbb{G}$, $\xi_1^b, \ldots, \xi_N^b$ are k-variate polynomials and $x_1, \ldots, x_k$ are random elements in $\mathbb{Z}_q$. Associated with the distribution $\mathcal{D}_b$ is the oracle $\mathcal{O}$ initialized with a list L_b which consists of $(h_1^b, \xi_1^b), \ldots, (h_N^b, \xi_N^b)$, where h_i^b is a random handle assigned to ξ_i^b, for every $i \in [N]$. We describe the interaction of the adversary $\mathcal{A}$ with the oracle $\mathcal{O}$, initialized with L_b. The oracle $\mathcal{O}$, initialized with L_b, sends the handles $h_1^b, \ldots, h_N^b$ to the adversary. The adversary can then submit n handles $h_{i_1}, \ldots, h_{i_n}$ and an n-variate polynomial p. The oracle $\mathcal{O}$ first checks if for every $j \in [n]$, the handle h_{i_j} is in its internal list. If one of the h_{i_j} is not in the list, it returns $\perp$. Then it checks if $p(\xi_{i_1}, \ldots, \xi_{i_n})$ was queried before, where ξ_{i_j} is the polynomial associated with the handle h_{i_j}. If it is in the list then it returns the handle h where $(h, p(\xi_{i_1}, \ldots, \xi_{i_n}))$ is the entry in its internal list. If it is not in the list then it returns h, picked uniformly at random, and stores $(h, p(\xi_{i_1}, \ldots, \xi_{i_n}))$ in its internal list. The adversary can also check if a handle h^* corresponds to a zero polynomial by submitting h^* to $\mathcal{O}$. As before, $\mathcal{O}$ returns $\perp$ if h^* is not in the list. Otherwise it returns 0 if the polynomial ξ^* associated with h^* (i.e., (h^*, ξ^*) is an entry in the internal list of $\mathcal{O}$) satisfies $\xi^*(x_1, \ldots, x_n) = 0$ for every $(x_1, \ldots, x_n) \in \mathbb{Z}_q$, else it returns 1.

We emphasize that even though the adversary is computationally unbounded, it can only make poly-logarithmic (in the order of the group) queries to the oracle.

A.1 DLIN with Leakage in the Bilinear Generic Group Model

DLINwithLeakage **with respect to** $(\mathbb{G}, \mathbb{G}_T, e)$**.** *Suppose $\mathbb{G}, \mathbb{G}_T$ be groups of prime order $q(\lambda)$. For every non-uniform probabilistic polynomial time adversary $\mathcal{A}$, the following holds for some negligible function* negl*,*

$$\Bigg|\mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(g, f, h, f^R, h^S, g^{R+S}, \begin{bmatrix} f^{R^2} & h^{RS-t} & g^{R(R+S+1)-t} \\ f^{RS+t} & h^{S^2} & g^{S(R+S+1)+t} \end{bmatrix}\right) \;:\; (g, f, h) \xleftarrow{\$} \mathbb{G}\backslash\{1\}, (R, S, t) \xleftarrow{\$} \mathbb{Z}_q\right] -$$

$$\mathsf{Pr}\left[1 \leftarrow \mathcal{A}\left(g, f, h, f^R, h^S, g^{R+S-1}, \begin{bmatrix} f^{R^2} & h^{RS-t} & g^{R(R+S-1)-t} \\ f^{RS+t} & h^{S^2} & g^{S(R+S-1)+t} \end{bmatrix}\right) \;:\; (g, f, h) \xleftarrow{\$} \mathbb{G}\backslash\{1\}, (R, S, t) \xleftarrow{\$} \mathbb{Z}_q\right]\Bigg|$$

$$\leq \mathsf{negl}(\lambda)$$

We prove that the above assumption holds in the generic group model. Before we show this, we define two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$, where:

$$\mathcal{D}_b = \left(g, (f = g^x), (h = g^y), g^{xR}, g^{yS}, g^{\alpha_b} \begin{bmatrix} g^{xR^2} & g^{y(RS-t)} & g^{\beta_b} \\ g^{x(RS+t)} & g^{yS^2} & g^{\gamma_b} \end{bmatrix} \right)$$

with $x, y, R, S, t \xleftarrow{\$} \mathbb{Z}_q$ and $\alpha_0 = (R+S), \beta_0 = (R(R+S+1)-t), \gamma_0 = (S(R+S+1)-t)$ and $\alpha_1 = (R+S-1), \beta_1 = (R(R+S-1)-t), \gamma_1 = (S(R+S-1)+t)$. Note that the assumption states that $\mathcal{D}_0 \approx_c \mathcal{D}_1$.

Theorem 7. *DLIN with Leakage holds in the bilinear generic group model.*

Proof. Suppose $\mathcal{A}$ is a computationally unbounded adversary with access to the oracle $\mathcal{O}$. Consider the following polynomials over the formal variables $\mathbf{x}, \mathbf{y}, \mathbf{R}, \mathbf{S}, \mathbf{t}$:

- $\xi_1^0 = \xi_1^1 = \mathbf{1}$
- $\xi_2^0 = \xi_2^1 = \mathbf{x}$
- $\xi_3^0 = \xi_3^1 = \mathbf{y}$
- $\xi_4^0 = \xi_4^1 = \mathbf{xR}$
- $\xi_5^0 = \xi_5^1 = \mathbf{yS}$
- $\xi_6^0 = \mathbf{R} + \mathbf{S}$, $\xi_6^1 = \mathbf{R} + \mathbf{S} - 1$
- $\xi_7^0 = \xi_7^1 = \mathbf{xR}^2$
- $\xi_8^0 = \xi_8^1 = \mathbf{y}(\mathbf{RS} - \mathbf{t})$
- $\xi_9^0 = \mathbf{R}(\mathbf{R} + \mathbf{S} + 1) - \mathbf{t}$, $\xi_9^1 = \mathbf{R}(\mathbf{R} + \mathbf{S} - 1) - \mathbf{t}$
- $\xi_{10}^0 = \xi_{10}^1 = \mathbf{x}(\mathbf{RS} + \mathbf{t})$
- $\xi_{11}^0 = \xi_{11}^1 = \mathbf{yS}^2$
- $\xi_{12}^0 = \mathbf{S}(\mathbf{R} + \mathbf{S} + 1) + \mathbf{t}$, $\xi_{12}^1 = \mathbf{S}(\mathbf{R} + \mathbf{S} - 1) + \mathbf{t}$

For any $b \in \{0,1\}$, denote by h_i^b the handle generated by $\mathcal{O}$, corresponding to the polynomial ξ_i^b. Let L_b be the list defined by $\{(h_i^b, \xi_i^b)\}_{i \in [12]}$. To prove the theorem it suffices to argue that:

$$\left| \Pr\left[1 \leftarrow \mathcal{A}^{\mathcal{O}(L_0)}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathcal{O}(L_1)}\right] \right| \leq \frac{\text{polylog}(q)}{q},$$

where $\mathcal{A}$ can make at most polylog(q) queries to $\mathcal{O}$.

We start with the following observation: it suffices to prove that if the adversary submits two polynomials p_i and p_j such that $p_i(\xi_1^0, \ldots, \xi_{12}^0) = p_j(\xi_1^0, \ldots, \xi_{12}^0)$ then it should hold that $p_i(\xi_1^1, \ldots, \xi_{12}^1) = p_j(\xi_1^1, \ldots, \xi_{12}^1)$ (and vice versa). More generally, we prove that for every degree-2 polynomial p, $p(\xi_1^0, \ldots, \xi_{12}^0)$ is a zero polynomial if and only if $p(\xi_1^1, \ldots, \xi_{12}^1)$ is a zero polynomial.

Lemma 1. *Let $p \in \mathbb{Z}_q[\mathbf{x}, \mathbf{y}, \mathbf{R}, \mathbf{S}, \mathbf{t}]$ be a degree 2 polynomial. $p(\xi_1^0, \ldots, \xi_{12}^0)$ is a zero polynomial if and only if $p(\xi_1^1, \ldots, \xi_{12}^1)$ is a zero polynomial.*

Proof. We prove only one direction; the proof for the other direction follows symmetrically.

Suppose $p(\xi_1^0, \ldots, \xi_{12}^0)$ is a zero polynomial. Let,

$$\begin{aligned} p(\xi_1^0, \ldots, \xi_{12}^0) \;=\; & Q_1(\xi_1^0, \ldots, \xi_{12}^0) + Q_2(\xi_1^0, \ldots, \xi_{12}^0) \cdot \mathbf{x} + Q_3(\xi_1^0, \ldots, \xi_{12}^0) \cdot \mathbf{y} \\ & + Q_4(\xi_1^0, \ldots, \xi_{12}^0) \cdot \mathbf{x}^2 + Q_5(\xi_1^0, \ldots, \xi_{12}^0) \cdot \mathbf{y}^2 + Q_6(\xi_1^0, \ldots, \xi_{12}^0) \cdot \mathbf{xy} \end{aligned}$$

where $Q_1(\xi_1^0, \ldots, \xi_{12}^0), \ldots, Q_6(\xi_1^0, \ldots, \xi_{12}^0)$ are polynomials over $\mathbf{R}, \mathbf{S}, \mathbf{t}$. Since $p(\xi_1^0, \ldots, \xi_{12}^0)$ is a zero polynomial, the polynomials $\{Q_i(\xi_1^0, \ldots, \xi_{12}^0)\}$ are zero polynomials. We now prove that $Q_i(\xi_1^1, \ldots, \xi_{12}^1)$ is a zero polynomial for every $i \in [6]$. Once we do this, the proof of the lemma will be complete. We handle each of these polynomials separately.

Claim 10. $Q_1(\xi_1^1, \ldots, \xi_{12}^1) = \mathbf{0}$.

Proof. Observe that $Q_1(\xi_1^b, \ldots, \xi_{12}^b)$ is of the following form:

$$c_1 + c_2\xi_6^b + c_3\xi_9^b + c_4\xi_{12}^b + c_5\xi_6^b\xi_9^b + c_6\xi_6^b\xi_{12}^b + c_7\xi_9^b\xi_{12}^b + c_8(\xi_6^b)^2 + c_9(\xi_9^b)^2 + c_{10}(\xi_{12}^b)^2,$$

where $c_1, \ldots, c_{10} \in \mathbb{Z}_q$. We omit writing the terms of the form $c_1'\xi_1^b$ and $c_1''\left(\xi_1^b\right)^2$ since $\xi_1^b = 1$, $\left(\xi_1^b\right)^2 = 1$, and hence are subsumed in the term c_1, where $c_1 = c_1' + c_1''$. Similarly, we also omit writing the terms $\xi_1^b\xi_6^b, \xi_1^b\xi_9^b, \xi_1^b\xi_{12}^b$ since $\xi_6^b = \xi_1^b\xi_6^b$, $\xi_9^b = \xi_1^b\xi_9^b$ and $\xi_{12}^b = \xi_1^b\xi_{12}^b$. We first focus on the case when $b = 0$. First observe that $c_{10} = 0$; this is because $(\xi_{12}^0)^2$ has the term $\mathbf{S}^4$ that cannot be canceled by other polynomials. Similarly $c_9 = 0$, since $(\xi_9^0)^2$ has the term $\mathbf{R}^4$. In addition, it holds that $c_7 = 0$ since $(\xi_9^0\xi_{12}^0)$ has the term $\mathbf{R}^3\mathbf{S}$, that cannot be canceled by other polynomials. Also $c_5 = 0$, since $\xi_6^0\xi_9^0$ contains the term $\mathbf{R}^3$ that cannot be canceled by other polynomials and, similarly $c_6 = 0$, $\xi_6^0\xi_{12}^0$ contains the term $\mathbf{S}^3$ that cannot be canceled by other polynomials. So far, we have shown that $c_5, c_6, c_7, c_9, c_{10} = 0$. We now expand Q_1.

$$\begin{aligned} Q_1(\xi_1^0, \ldots, \xi_{12}^0) &= c_1 + c_2(\mathbf{R} + \mathbf{S}) + c_3(\mathbf{R}^2 + \mathbf{RS} + \mathbf{R} - \mathbf{t}) + c_4(\mathbf{S}^2 + \mathbf{RS} + \mathbf{S} + \mathbf{t}) + c_8(\mathbf{R}^2 + \mathbf{S}^2 + 2\mathbf{RS}) \\ &= c_1 + (c_2 + c_3)\mathbf{R} + (c_2 + c_4)\mathbf{S} + (c_3 + c_8)\mathbf{R}^2 + (c_4 + c_8)\mathbf{S}^2 + (c_3 + c_4 + 2c_8)\mathbf{RS} \\ &\quad + (-c_3 + c_4)\mathbf{t} \end{aligned}$$

Since Q_1 is a zero polynomial, we have $c_1 = 0$, $c_2 = -c_3$,$c_2 = -c_4$, $c_3 = -c_8$, $c_3 + c_4 + 2c_8 = 0$ and $c_3 = c_4$. That is, $c_1 = 0$, $c_2 = -c_3$, $c_3 = c_4$ and, $c_2 = c_8$.

Now, we focus on $Q_1(\xi_1^1, \ldots, \xi_{12}^1)$. Expanding $Q_1(\xi_1^1, \ldots, \xi_{12}^1)$ we get:

$$\begin{aligned} Q_1(\xi_1^1, \ldots, \xi_{12}^1) &= c_2(\mathbf{R} + \mathbf{S} - 1) + c_3(\mathbf{R}^2 + \mathbf{RS} - \mathbf{R} - \mathbf{t}) + c_4(\mathbf{S}^2 + \mathbf{RS} - \mathbf{S} + \mathbf{t}) \\ &\quad + c_8(\mathbf{R}^2 + \mathbf{S}^2 + 2\mathbf{RS} - 2\mathbf{R} - 2\mathbf{S} + 1) \\ &= (c_2 - c_3 - 2c_8)\mathbf{R} + (c_2 - c_4 - 2c_8)\mathbf{S} + (c_3 + c_8)\mathbf{R}^2 + (c_4 + c_8)\mathbf{S}^2 \\ &\quad + (c_3 + c_4 + 2c_8)\mathbf{RS} + (-c_3 + c_4)\mathbf{t} + (-c_2 + c_8) \\ &= \mathbf{0} \end{aligned}$$

The last equality follows from the fact that $c_2 = -c_3 = -c_4 = c_8$. □

Claim 11. $Q_2(\xi_1^1, \ldots, \xi_{12}^1) = \mathbf{0}$.

Proof. For $b \in \{0, 1\}$, observe that $Q_2(\xi_1^b, \ldots, \xi_{12}^b)$ is of the following form:

$$\mathbf{x}^{-1} \cdot \left(\sum_{\substack{i \in \{1,6,9,12\}, \\ j \in \{2,4,7,10\}}} c_{i,j}\xi_i^b\xi_j^b \right),$$

where $c_{i,j} \in \mathbb{Z}_q$.

We first make some initial observations about the coefficients:

- $c_{9,7} = 0$; because $\xi_9^b\xi_7^b$ contains the term $\mathbf{xR}^4$ that cannot be canceled by other terms.
- $c_{12,10} = 0$; because $\xi_{12}^b\xi_{10}^b$ contains the term $\mathbf{xRS}^3$ that cannot be canceled by other terms.
- $c_{9,10} = 0$; because $\xi_9^b\xi_{10}^b$ contains the term $\mathbf{xt}^2$ that can only be canceled by the corresponding term contained in $\xi_{12}^b\xi_{10}^b$, but since $c_{12,10} = 0$ we have $c_{9,10} = 0$.
- $c_{12,7} = 0$; $\xi_{12}^b\xi_7^b$ contains the term $\mathbf{xR}^2\mathbf{S}^2$ that can only be canceled by the corresponding terms contained in the polynomials $\xi_{12}^b\xi_{10}^b, \xi_9^b\xi_{10}^b$, but since $c_{12,10}, c_{9,10} = 0$ we have $c_{12,7} = 0$.

- $c_{6,10} = 0$; because $\xi_6^b \xi_{10}^b$ contains the term $\mathbf{xSt}$ that cannot be canceled by other terms.
- $c_{12,4} = 0$; because $\xi_{12}^b \xi_4^b$ contains the term $\mathbf{xRS}^2$ that can only be canceled by the corresponding terms in the polynomials $\xi_{12}^b \xi_{10}^b, \xi_6^b \xi_{10}^b$, but since $c_{12,10}, c_{6,10} = 0$ we have $c_{12,4} = 0$.
- $c_{12,2} = 0$; because $\xi_{12}^b \xi_1^b$ contains the term $\mathbf{xS}^2$ that cannot be canceled by other terms.
- $c_{6,2} = 0$; because $\xi_6^b \xi_2^b$ contains $\mathbf{xS}$ that can only be canceled by the corresponding term contained in $\xi_{12}\xi_2$, but since $c_{12,2} = 0$ we have $c_{6,2} = 0$.
- $c_{9,4} = 0$; because $\xi_9^b \xi_4^b$ contains $\mathbf{xRt}$ that can only be canceled by corresponding terms contained in the polynomials $\xi_6^b \xi_{10}^b, \xi_9^b \xi_{10}^b, \xi_{12}^b \xi_4^b$, but since $c_{6,10}, c_{9,10}, c_{12,4} = 0$ we have $c_{9,4} = 0$.
- $c_{6,7} = 0$; because $\xi_6^b \xi_7^b$ contains $\mathbf{xR}^3$ that can only be canceled by the corresponding term contained in $\xi_9^b \xi_4^b, \xi_9^b \xi_7^b$, but since $c_{9,4}, c_{9,7} = 0$ we have $c_{6,7} = 0$.

Rewriting $Q_2(\xi_1^b, \ldots, \xi_{12}^b)$, we have:

$$\mathbf{x}^{-1} \cdot \left(c_{1,2}\xi_1^b\xi_2^b + c_{1,4}\xi_1^b\xi_4^b + c_{1,7}\xi_1^b\xi_7^b + c_{1,10}\xi_1^b\xi_{10}^b + c_{6,4}\xi_6^b\xi_4^b + c_{9,2}\xi_9^b\xi_2^b\right)$$

We now analyze the cases for $b = 0$ and $b = 1$ separately.

We start with $b = 0$.

$$\begin{aligned} Q_2(\xi_1^0, \ldots, \xi_{12}^0) &= c_{1,2} + (c_{1,4} + c_{9,2})\mathbf{R} + (c_{1,7} + c_{6,4} + c_{9,2})\mathbf{R}^2 \\ &\quad + (c_{1,10} + c_{6,4} + c_{9,2})\mathbf{RS} + (c_{1,10} - c_{9,2})\mathbf{t} \\ &= \mathbf{0} \end{aligned}$$

From the above equation, the following holds:

- $c_{1,2} = 0$.
- $c_{1,4} = -(c_{9,2})$
- $c_{1,7} + c_{6,4} + c_{9,2} = 0$
- $c_{1,10} + c_{6,4} + c_{9,2} = 0$
- $c_{1,10} = c_{9,2}$

Now, we consider the case when $b = 1$.

$$\begin{aligned} Q_2(\xi_1^1, \ldots, \xi_{12}^1) &= c_{1,2} + (c_{1,4} - c_{9,2} - c_{6,4})\mathbf{R} + (c_{1,7} + c_{6,4} + c_{9,2})\mathbf{R}^2 \\ &\quad + (c_{1,10} + c_{6,4} + c_{9,2})\mathbf{RS} + (c_{1,10} - c_{9,2})\mathbf{t} \\ &= (c_{1,4} - c_{9,2} - c_{6,4})\mathbf{R} + (c_{1,7} + c_{6,4} + c_{9,2})\mathbf{R}^2 \\ &\quad + (c_{1,10} + c_{6,4} + c_{9,2})\mathbf{RS} + (c_{1,10} - c_{9,2})\mathbf{t} \ (\because c_{1,2} = 0) \\ &= (-2c_{9,2} - (-2c_{9,2}))\mathbf{R} + (c_{1,7} + c_{6,4} + c_{9,2})\mathbf{R}^2 \\ &\quad + (c_{1,10} + c_{6,4} + c_{9,2})\mathbf{RS} + (c_{1,10} - c_{9,2})\mathbf{t} \ (\because c_{1,4} = -c_{9,2}, c_{6,4} = -2c_{9,2}) \\ &= (c_{1,7} + c_{6,4} + c_{9,2})\mathbf{R}^2 + (c_{1,10} + c_{6,4} + c_{9,2})\mathbf{RS} + (c_{1,10} - c_{9,2})\mathbf{t} \\ &= \mathbf{0} \ (\because \text{from the last three bullets described above}) \end{aligned}$$

□

Claim 12. $Q_3(\xi_1^1, \ldots, \xi_{12}^1) = \mathbf{0}$.

Proof. For $b \in \{0,1\}$, observe that $Q_3(\xi_1^b, \ldots, \xi_{12}^b)$ is of the following form:

$$\mathbf{y}^{-1} \cdot \left(\sum_{\substack{i \in \{1,6,9,12\}, \\ j \in \{3,5,8,11\}}} c_{i,j} \xi_i^b \xi_j^b \right),$$

where $c_{i,j} \in \mathbb{Z}_q$ and $b \in \{0,1\}$.

We first make some initial observations about the coefficients:

- $c_{12,11} = 0$; because $\xi_{12}^b \xi_{11}^b$ contains the term $\mathbf{yS}^4$ that cannot be canceled by other terms.
- $c_{9,8} = 0$; because $\xi_9^b \xi_8^b$ contains the term $\mathbf{ySR}^3$ that cannot be canceled by other terms.
- $c_{12,8} = 0$; because $\xi_{12}^b \xi_8^b$ contains the term $\mathbf{yt}^2$ that can only be canceled by the corresponding terms contained in $\xi_9^b \xi_8^b$, but since $c_{9,8} = 0$ we have $c_{12,8} = 0$.
- $c_{9,11} = 0$; because $\xi_9^b \xi_{11}^b$ contains the term $\mathbf{yR}^2\mathbf{S}^2$ that can only be canceled by the corresponding terms contained in the polynomials $\xi_9^b \xi_8^b, \xi_{12}^b \xi_8^b$, but since $c_{9,8}, c_{12,8} = 0$ we have $c_{9,11} = 0$.
- $c_{6,8} = 0$; because $\xi_6^b \xi_8^b$ contains the term $\mathbf{yRt}$ that cannot be canceled by other terms.
- $c_{6,11} = 0$; because $\xi_6^b \xi_{11}^b$ contains the term $\mathbf{ySR}^2$ that can only be canceled by the corresponding terms in the polynomials $\xi_6^b \xi_8^b, \xi_{12}^b \xi_8^b$, but since $c_{6,8}, c_{12,8} = 0$ we have $c_{6,11} = 0$.
- $c_{9,3} = 0$; because $\xi_9^b \xi_3^b$ contains the term $\mathbf{yR}^2$ that cannot be canceled by other terms.
- $c_{6,3} = 0$; because $\xi_6^b \xi_3^b$ contains $\mathbf{yR}$ that can only be canceled by the corresponding term contained in $\xi_{9,3}$, but since $c_{9,3} = 0$ we have $c_{6,3} = 0$.
- $c_{12,5} = 0$; because $\xi_{12}^b \xi_5^b$ contains $\mathbf{yS}^3$ that can only be canceled by the corresponding term contained in $\xi_{12}^b \xi_{11}^b, \xi_6^b \xi_{11}^b$, but since $c_{12,11}, c_{6,11} = 0$ we have $c_{12,5} = 0$.
- $c_{9,5} = 0$; because $\xi_9^b \xi_5^b$ contains $\mathbf{ySt}$ that can only be canceled by corresponding terms contained in the polynomials $\xi_{12}^b \xi_8^b, \xi_6^b \xi_8^b, \xi_{12}^b \xi_5^b$, but since $c_{12,8}, c_{6,8}, c_{12,5} = 0$ we have $c_{9,5} = 0$.

Rewriting $Q_3(\xi_1^b, \ldots, \xi_{12}^b)$, we have:

$$\mathbf{y}^{-1} \cdot \left(c_{1,3}\xi_1^b\xi_3^b + c_{1,5}\xi_1^b\xi_5^b + c_{1,8}\xi_1^b\xi_8^b + c_{1,11}\xi_1^b\xi_{11}^b + c_{6,5}\xi_6^b\xi_5^b + c_{12,3}\xi_{12}^b\xi_3^b \right)$$

We now analyze the cases for $b = 0$ and $b = 1$ separately.

We start with $b = 0$.

$$\begin{aligned} Q_3(\xi_1^0, \ldots, \xi_{12}^0) &= c_{1,3} + (c_{1,5} + c_{12,3})\mathbf{S} + (c_{1,11} + c_{6,5} + c_{12,3})\mathbf{S}^2 \\ &\quad + (c_{1,8} + c_{6,5} + c_{12,3})\mathbf{RS} + (c_{1,8} - c_{12,3})\mathbf{t} \\ &= \mathbf{0} \end{aligned}$$

From the above equation, the following holds:

- $c_{1,3} = 0$.
- $c_{1,5} = -(c_{12,3})$
- $c_{1,11} + c_{6,5} + c_{12,3} = 0$
- $c_{1,8} + c_{6,5} + c_{12,3} = 0$

– $c_{1,8} - c_{12,3} = 0$

Now, we consider the case when $b = 1$.

$$\begin{aligned}
Q_3(\xi_1^1, \ldots, \xi_{12}^1) &= c_{1,3} + (c_{1,5} - c_{12,3} - c_{6,5})\mathbf{S} + (c_{1,11} + c_{6,5} + c_{12,3})\mathbf{S}^2 \\
&\quad +(c_{1,8} + c_{6,5} + c_{12,3})\mathbf{RS} + (c_{1,8} - c_{12,3})\mathbf{t} \\
&= (c_{1,5} - c_{12,3} - c_{6,5})\mathbf{S} + (c_{1,11} + c_{6,5} + c_{12,3})\mathbf{S}^2 \\
&\quad +(c_{1,8} + c_{6,5} + c_{12,3})\mathbf{RS} + (c_{1,8} - c_{12,3})\mathbf{t}(\because c_{1,3} = 0) \\
&= (-2c_{12,3} - (-2c_{12,3}))\mathbf{S} + (c_{1,11} + c_{6,5} + c_{12,3})\mathbf{S}^2 \\
&\quad +(c_{1,8} + c_{6,5} + c_{12,3})\mathbf{RS} + (c_{1,8} - c_{12,3})\mathbf{t}(\because c_{1,5} = -c_{12,3}, c_{6,5} = -2c_{12,3}) \\
&= (c_{1,11} + c_{6,5} + c_{12,3})\mathbf{S}^2 + (c_{1,8} + c_{6,5} + c_{12,3})\mathbf{RS} + (c_{1,8} - c_{12,3})\mathbf{t} \\
&= \mathbf{0} \text{ } (\because \text{from the last three bullets described above})
\end{aligned}$$

□

Claim 13. $Q_4(\xi_1^1, \ldots, \xi_{12}^1) = \mathbf{0}$.

Proof. For $b \in \{0, 1\}$, observe that $Q_4(\xi_1^b, \ldots, \xi_{12}^b)$ is of the following form:

$$\mathbf{x}^{-2} \cdot \left(\sum_{i,j \in \{2,4,7,10\}, j \geq i} c_{i,j} \xi_i^b \xi_j^b \right),$$

where $c_{i,j} \in \mathbb{Z}_q$ and $b \in \{0, 1\}$. Moreover, $\xi_i^0 = \xi_i^1$ for $i \in \{2, 4, 7, 10\}$. Thus, $Q_4(\xi_1^0, \ldots, \xi_{12}^0) = \mathbf{0}$ implies that $Q_4(\xi_1^1, \ldots, \xi_{12}^1) = \mathbf{0}$. □

Claim 14. $Q_5(\xi_1^1, \ldots, \xi_{12}^1) = \mathbf{0}$.

Proof. For $b \in \{0, 1\}$, observe that $Q_5(\xi_1^b, \ldots, \xi_{12}^b)$ is of the following form:

$$\mathbf{y}^{-2} \cdot \left(\sum_{i,j \in \{3,5,8,11\}, j \geq i} c_{i,j} \xi_i^b \xi_j^b \right),$$

where $c_{i,j} \in \mathbb{Z}_q$ and $b \in \{0, 1\}$. Moreover, $\xi_i^0 = \xi_i^1$ for $i \in \{3, 5, 8, 11\}$. Thus, $Q_5(\xi_1^0, \ldots, \xi_{12}^0) = \mathbf{0}$ implies that $Q_5(\xi_1^1, \ldots, \xi_{12}^1) = \mathbf{0}$. □

Claim 15. $Q_6(\xi_1^1, \ldots, \xi_{12}^1) = \mathbf{0}$.

Proof. Observe that $Q_6(\xi_1^b, \ldots, \xi_{12}^b)$ is of the following form:

$$(\mathbf{xy})^{-1} \cdot \left(\sum_{i \in \{2,4,7,10\}, j \in \{3,5,8,11\}} c_{i,j} \xi_i^b \xi_j^b \right),$$

where $c_{i,j} \in \mathbb{Z}_q$ and $b \in \{0, 1\}$. Moreover, $\xi_j^0 = \xi_j^1$ for $j \in \{3, 5, 8, 11\}$ and $\xi_i^0 = \xi_i^1$ for $i \in \{2, 4, 7, 10\}$. Thus, $Q_6(\xi_1^0, \ldots, \xi_{12}^0) = \mathbf{0}$ implies that $Q_6(\xi_1^1, \ldots, \xi_{12}^1) = \mathbf{0}$. □

Thus, we proved that $Q_i(\xi_1^1, \ldots, \xi_{12}^1)$ is a zero polynomial for every $i \in [6]$. This proves that $p(\xi_1^1, \ldots, \xi_{12}^1)$ is also a zero polynomial. As remarked before, the other direction also can be argued symmetrically.

□

□