Succinct Malleable NIZKs and an Application to Compact Shuffles

Melissa Chase
Microsoft Research Redmond
melissac@microsoft.com

Markulf Kohlweiss
Microsoft Research Cambridge
markulf@microsoft.com

Anna Lysyanskaya
Brown University
anna@cs.brown.edu

Sarah Meiklejohn
UC San Diego
smeiklej@cs.ucsd.edu

September 1, 2012

Abstract

Depending on the application, malleability in cryptography can be viewed as either a flaw or — especially if sufficiently understood and restricted — a feature. In this vein, Chase, Kohlweiss, Lysyanskaya, and Meiklejohn recently defined malleable zero-knowledge proofs, and showed how to *control* the set of allowable transformations on proofs. As an application, they construct the first *compact* verifiable shuffle, in which one such controlled-malleable proof suffices to prove the correctness of an entire multi-step shuffle.

Despite these initial steps, a number of natural open problems remain: (1) their construction of controlled-malleable proofs relies on the inherent malleability of Groth-Sahai proofs and is thus not based on generic primitives; (2) the classes of allowable transformations they can support are somewhat restrictive; and (3) their construction of a compactly verifiable shuffle has proof size $O(N^2 + L)$ (where N is the number of votes and L is the number of mix authorities), whereas in theory such a proof could be of size $O(N + L)$.

In this paper, we address these open problems by providing a generic construction of controlled-malleable proofs using succinct non-interactive arguments of knowledge, or SNARGs for short. Our construction has the advantage that we can support a very general class of transformations (as we no longer rely on the transformations that Groth-Sahai proofs can support), and that we can use it to obtain a proof of size $O(N + L)$ for the compactly verifiable shuffle.

1 Introduction

Recently, malleability is increasingly being viewed more as a feature than as a bug [27, 28, 17, 1, 12, 15, 6]. In this vein, earlier this year we (called CKLM in the sequel to disambiguate between our current and prior work) [7] introduced controlled-malleable non-interactive zero-knowledge proof systems (cm-NIZKs for short). At a high level, a cm-NIZK allows one, given a proof π for an instance $x \in L$, to compute a proof π' for the related instance $T(x) \in L$ for transformations T under which the language is closed. This malleability property can be additionally *controlled*, meaning there is some specified class of allowable transformations $\mathcal{T}$ such that, given the proof π for $x \in L$, a new proof π' for $T(x) \in L$ may be obtained only for $T \in \mathcal{T}$. The notion of a cm-NIZK is non-trivial when the proof system also needs to be concise or *derivation-private*; i.e., in addition to π' being the same size as π, it should be impossible to tell whether π' was obtained using a witness or by mauling a proof for a previous statement.

The notion of a derivation-private cm-NIZK is well motivated: as one application, CKLM showed that it allows for the modular design of randomizable and homomorphic chosen-ciphertext security. Another interesting application they presented is a *compactly verifiable shuffle* for an election, wherein a set of encrypted votes, submitted by N different voters, is shuffled (i.e. re-randomized and permuted), in turn, by L voting authorities. To ensure that the authorities are behaving honestly, each authority provides a non-interactive zero-knowledge proof that it has correctly shuffled the votes; if this is done using standard NIZKs, then in order to verify that the overall shuffling process was correct a verifier would need to access L separate proofs, each proving that an authority correctly performed the shuffling process. If each proof is of size $s(N)$, this means that the verifier's work is $\Theta(Ls(N))$ (here we ignore the security parameter). Using derivation-private cm-NIZKs, the verifier's workload can be reduced: each authority can, instead of producing a brand new proof, "maul" the proof of the previous authority; the proof produced by the last authority should then convince the verifier that the ciphertexts output at the end are a valid shuffling of the input ciphertexts. This makes vote shuffling a factor of L more efficient, as the verifier needs to verify a proof of size only $\Theta(s(N) + L)$. (The size of the proof is still dependent on L because each authority needs to, intuitively, add a "stamp of participation" in order for a verifier to ascertain that the shuffling process was performed correctly.)

CKLM then showed how to construct derivation-private cm-NIZK proof systems for a limited, but nevertheless expressive, class of transformations. Specifically, their approach builds heavily on the Groth-Sahai proof system [24]; this means that they can consider only relations on group elements in groups that admit bilinear pairings, and it might therefore seem as though controlled malleability were just a property of the Groth-Sahai proof system and not necessarily something that could be realized using more general building blocks. Interestingly, as a consequence of this limitation, CKLM did not fully deliver on the promise of a compactly verifiable shuffle: in order to prove that a given set of ciphertexts is a shuffle, they needed to represent everything, including the transformations applied to the set of ciphertexts, as a set of elements in the underlying group. The way they chose to do this was using a permutation matrix; since this permutation matrix needs to be extractable from the proof, the size of each proof in their construction was $\Theta(N^2 + L)$. For the usual voting scenario, in which the number of voters far exceeds the number of mix authorities, a vote shuffling scheme wherein each authority produces its own proof but the proofs are only of size $\Theta(N)$ (such as the verifiable shuffle of Groth and Lu [22]), therefore has a shorter proof overall.

Thus, the two important, and somewhat related open problems were: first, can a derivation-private controlled-malleable NIZK be realized in a modular fashion from general building blocks, and not just from the Groth-Sahai proof system? CKLM showed, for example, that cm-NIZKs give modular constructions of randomizable and homomorphic CCA security. If cm-NIZKs could be realized only from the Groth-Sahai proof system based on statements expressible as relations on group elements then these findings would not apply to other homomorphic cryptosystems, whereas if they could be based on general building blocks then we could apply the CKLM results more broadly. Second, in compact vote shuffling, can the proof be of size $\Theta(N + L)$? In this paper, we give a positive answer to both.

Our contributions. We first investigate, in Section 3, how to construct a derivation-private cm-NIZK from succinct non-interactive arguments (SNARGs) [21, 6]. We limit our attention to t-tiered languages and transformations; briefly, a language is t-tiered if each instance x can be efficiently labeled with an integer $i = \mathsf{tier}(x)$, $1 \leq i \leq t$, and a transformation T for a t-tiered language L is t-tiered if $\mathsf{tier}(T(x)) > \mathsf{tier}(x)$ for all $x \in L$ where $\mathsf{tier}(x) < t$, and $T(x) = \perp$ if $\mathsf{tier}(x) = t$. Some transformations are naturally t-tiered: for example, a vote shuffling transformation carried out by authority i should output a set of ciphertexts and stamps of approval from each authority up to i; furthermore, all transformations can be made t-tiered if one is willing to reveal how many times a transformation has been applied.

Intuitively, our construction works as follows: given a proof π for an instance $x \in L$, to provide a proof for a new instance $x' = T(x) \in L$, a user can form a "proof of a proof;" i.e., prove knowledge of this previous instance x and its proof π, as well as the transformation T from x to x', and call this proof π'. By the succinctness property of SNARGs, this new proof π' can in fact be the same size as the previous proof π, and thus this "proof of a proof" approach can be continued without incurring any blowup in size.

Although the intuition is relatively simple, going from SNARGs to cm-NIZKs is in fact quite challenging. While the outline above describes how to build malleability into SNARGs, it is still the case that SNARGs satisfy only the non-black-box notion of adaptive knowledge extraction, whereas cm-NIZKs require a much stronger (black-box) version of extractability. (This stronger notion is crucially used in the CCA encryption and the shuffle applications in CKLM.) To therefore break all these requirements up into smaller pieces, we begin with SNARGs and then slowly work our way up to cm-NIZKs in three separate constructions, with each construction incorporating an additional requirement.

We begin in Section 3.1 with a construction of a malleable SNARG. This construction closely follows the intuition above (which is itself inspired by the "targeted malleability" construction of Boneh et al. [6]): malleability is achieved by proving knowledge of either a fresh witness or a previous instance and proof, and a transformation from that instance to the current one. As observed by Bitansky et al. [3, 4], care must be taken with this kind of recursive composition of SNARGs, as the size of the extractor can quickly blow up as we continue to extract proofs from other proofs; we can therefore construct t-tiered malleable SNARGs (i.e., SNARGs malleable with respect to the class of all t-tiered transformations) for only constant t. Furthermore, a formal treatment of our particular recursive technique reveals that a stronger notion of extraction, in which the extractor gets to see not only the random tape but also the code for the adversary, is necessary for both our construction and the original one of Boneh et al.

With our construction in Section 3.1, we therefore added malleability to the SNARG while preserving succinctness. In Section 3.2, we next tackle the issue of extractability; in particular, we want to boost from the non-black-box notion of extractability supported by SNARGs to the standard black-box notion of a proof of knowledge (NIZKPoK). To do this, we in fact rely only on the soundness of the SNARG, and do not attempt to use the (non-black-box) extractor at all. Instead, we perform a sort of verifiable encryption, in which we encrypt the witness and then prove knowledge (using the malleable SNARG) of the value inside the ciphertext; in this our approach is perhaps most similar to that of Damgård et al. [10]. A black-box extractor is then simple to construct: it just decrypts the ciphertext and thus, provided the proof is sound, recovers the witness. In addition, to preserve the full generality of our t-tiered transformations one would instantiate the encryption scheme using fully homomorphic encryption, although we will also see in Section 4 that interesting classes of transformations can still be supported by more limited schemes (such as ones that are multiplicatively homomorphic).

With our construction in Section 3.2, we therefore achieved the same properties that the Groth-Sahai proof system already provided (namely, a malleable NIWIPoK), but with respect to a more general class of transformations. As such, to now construct cm-NIZKs in Section 3.3, we can follow approximately the same construction as CKLM, who also used malleable NIWIPoKs to construct their cm-NIZK. Once again, however, care must be taken in this step, as we would like to preserve the generality in the class of transformations that we supported in the previous two sections. We therefore modify the CKLM construction to allow for this, and thus achieve cm-NIZKs for all t-tiered transformations.

In summary, we show that if zero-knowledge SNARGs exist for all languages in NP and fully homomorphic encryption exists, then derivation-private cm-NIZK proof systems exist for all t-tiered classes of transformations, where t is a constant. We do this by constructing three distinct types of proofs, each of which may be of independent interest: first, a malleable SNARG, then a malleable NIZKPoK, and finally a cm-NIZK. While each of our constructions builds from the previous one, we stress that our constructions are all fully generic; e.g., any malleable SNARG can be used to construct a malleable NIZKPoK, not just the specific one we construct.

Finally, for the compact shuffle application, we show in Section 4 how to use our SNARG-based proofs for t-tiered transformation classes (using just multiplicatively homomorphic encryption rather than the heavyweight requirement of fully homomorphic encryption) to construct a shuffle with proof size $\Theta(N + L)$. As a result, we achieve the first compact verifiable shuffles for N ciphertexts and L voting authorities with proof size $\Theta(N + L)$.

2 Definitions and Notation

We recall the main security notions we use. We begin with the recent definitions for malleability due to CKLM [7], as well as their definition for compactly verifiable shuffles; we then define succinct non-interactive zero-knowledge arguments (SNARGs), which form the basis for our construction of malleable proofs in Section 3.

2.1 Malleable proofs

Let $R(\cdot, \cdot)$ be a relation such that the corresponding language $L_R = \{x \mid \exists w \text{ such that } (x, w) \in R\}$ is in NP. As defined by CKLM, the relation is *closed* with respect to a transformation $T = (T_x, T_w)$ if, for every $(x, w) \in R$, $(T_x(x), T_w(w)) \in R$ as well. We define zero knowledge and related notions formally in Appendix A, but recall briefly here that a non-interactive zero-knowledge proof (NIZK) system [5, 13, 19] is a set of algorithms $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$ for which there exists an efficient simulator (S_1, S_2) such that no adversary can distinguish between proofs formed by the prover and proofs formed by the simulator, and an efficient extractor (E_1, E_2) that can produce a witness w such that $(x, w) \in R$ from any valid proof π for x. For zero knowledge, we discuss here two additional variants: the first, *composable* zero knowledge, says that the adversary should still be unable to distinguish even give the simulation trapdoor, and the second, *statistical* zero knowledge, says that the distribution of proofs formed by the simulator and prover are indistinguishable even to an unbounded adversary; composable zero knowledge is thus implied by statistical zero knowledge, as an unbounded adversary could produce the simulator trapdoor itself.

The main definition of CKLM for controlled malleable proofs reconciles simulation soundness [29, 11] and simulation-sound extractability [20] with malleability by requiring that, for a set of transformations $\mathcal{T}$, if an adversary can produce a proof π that $x \in L_R$ then the extractor can extract from π either a witness w or a transformation $T \in \mathcal{T}$ and previously proved instance x' such that $x = T_x(x')$. This is defined more formally as:

Definition 2.1. [7] *Let* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$ *be a NIZKPoK system for an efficient relation* R*, with a simulator* (S_1, S_2) *and an extractor* (E_1, E_2)*. Let* $\mathcal{T}$ *be a set of unary transformations for the relation* R *such that membership in* $\mathcal{T}$ *is efficiently testable. Let* SE_1 *be an algorithm that, on input* 1^k*, outputs* $(\mathsf{crs}, \tau_s, \tau_e)$ *such that* (crs, τ_s) *is distributed identically to the output of* S_1*. Let* $\mathcal{A}$ *be given, and consider the following game:*

- *Step 1.* $(\mathsf{crs}, \tau_s, \tau_e) \xleftarrow{\$} SE_1(1^k)$.
- *Step 2.* $(x, \pi) \xleftarrow{\$} \mathcal{A}^{S_2(\mathsf{crs}, \tau_s, \cdot)}(\mathsf{crs}, \tau_e)$.
- *Step 3.* $(w, x', T) \leftarrow E_2(\mathsf{crs}, \tau_e, x, \pi)$.

The NIZKPoK satisfies controlled-malleable simulation-sound extractability *(CM-SSE, for short) with respect to* $\mathcal{T}$ *if for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that the probability (over the choices of* SE_1*,* $\mathcal{A}$*, and* S_2*) that* $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ *and* $(x, \pi) \notin Q$ *(where* Q *is the set of queried statements and their responses) but either (1)* $w \neq \perp$ *and* $(x, w) \notin R$*; (2)* $(x', T) \neq (\perp, \perp)$ *and either* $x' \notin Q_x$ *(the set of queried instances),* $x \neq T_x(x')$*, or* $T \notin \mathcal{T}$*; or (3)* $(w, x', T) = (\perp, \perp, \perp)$ *is at most* $\nu(k)$*.*

CKLM also defined the notion of *derivation privacy* for malleable proofs, which says that proofs should not reveal whether they were formed fresh or via transformation.

Definition 2.2. [7] *For a non-interactive proof system* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V}, \mathsf{ZKEval})$*, an efficient relation* R *malleable with respect to* $\mathcal{T}$*, an adversary* $\mathcal{A}$*, and a bit* b*, let* $p_b^{\mathcal{A}}(k)$ *be the probability of the event that* $b' = 0$ *in the following game:*

- *Step 1.* $\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$.
- *Step 2.* $(\mathsf{state}, x_1, w_1, \pi_1, \ldots, x_q, w_q, \pi_q, T) \xleftarrow{\$} \mathcal{A}(\mathsf{crs})$.
- *Step 3. If* $\mathcal{V}(\mathsf{crs}, x_i, \pi_i) = 0$ *for some* i*,* $(x_i, w_i) \notin R$ *for some* i*, or* $T \notin \mathcal{T}$*, abort and output* $\bot$*. Otherwise, form*

$$\pi \xleftarrow{\$} \begin{cases} \mathcal{P}(\mathsf{crs}, T_x(x_1, \ldots, x_q), T_w(w_1, \ldots, w_q)) & \text{if } b = 0 \\ \mathsf{ZKEval}(\mathsf{crs}, T, \{x_i, \pi_i\}) & \text{if } b = 1. \end{cases}$$

- *Step 4.* $b' \xleftarrow{\$} \mathcal{A}(\mathsf{state}, \pi)$.

Then the proof system is derivation private *if for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that* $|p_0^{\mathcal{A}}(k) - p_1^{\mathcal{A}}(k)| < \nu(k)$.

CKLM give a zero-knowledge variant of derivation privacy called *strong derivation privacy*, in which proofs output by ZKEval should be indistinguishable from those output by the simulator. The security experiment is almost the same, with the only differences being that $\mathcal{A}$ is given the simulation trapdoor, $\mathcal{A}$ is not required to output any witnesses, and S_2 is used in place of $\mathcal{P}$. More formally, CKLM provide the following definition:

Definition 2.3. [7] *For a malleable NIZK proof system* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V}, \mathsf{ZKEval})$ *with an associated simulator* (S_1, S_2)*, a given adversary* $\mathcal{A}$*, and a bit* b*, let* $p_b^{\mathcal{A}}(k)$ *be the probability of the event that* $b' = 0$ *in the following game:*

- *Step 1.* $(\sigma_{sim}, \tau_s) \xleftarrow{\$} S_1(1^k)$.
- *Step 2.* $(\mathsf{state}, x_1, \pi_1, \ldots, x_q, \pi_q, T) \xleftarrow{\$} \mathcal{A}(\sigma_{sim}, \tau_s)$.
- *Step 3. If* $\mathcal{V}(\sigma_{sim}, x_i, \pi_i) = 0$ *for some* i*,* $(x_1, \ldots, x_q)$ *is not in the domain of* T_x*, or* $T \notin \mathcal{T}$*, abort and output* $\bot$*. Otherwise, form*

$$\pi \xleftarrow{\$} \begin{cases} S_2(\sigma_{sim}, \tau_s, T_x(x_1, \ldots, x_q)) & \text{if } b = 0 \\ \mathsf{ZKEval}(\sigma_{sim}, T, \{x_i, \pi_i\}) & \text{if } b = 1. \end{cases}$$

- *Step 4.* $b' \xleftarrow{\$} \mathcal{A}(\mathsf{state}, \pi)$.

Then the proof system is strongly derivation private *if for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that* $|p_0^{\mathcal{A}}(k) - p_1^{\mathcal{A}}(k)| < \nu(k)$.

Putting these all together, if a proof system is zero knowledge, strongly derivation private, and CM-SSE, then CKLM call it a *cm-NIZK*.

2.2 Compactly verifiable shuffles

A *compact* verifiable shuffle [7] requires that a single non-interactive proof suffices to verify the correctness of an entire multi-step shuffle. We have the following definition:

Definition 2.4. [7] *For a verifiable shuffle* $(\mathsf{Setup}, \mathsf{Shuffle}, \mathsf{Verify})$ *with respect to an encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*, a given adversary* $\mathcal{A}$ *and a bit* $b \in \{0,1\}$*, let* $p_b^{\mathcal{A}}(k)$ *be the probability that* $b' = 0$ *in the following experiment:*

- *Step 1.* $(params, sk, S = \{pk_i\}, \{sk_i\}) \xleftarrow{\$} \mathsf{Setup}(1^k)$.
- *Step 2.* $\mathcal{A}$ *gets* $params$, S, *and access to the following two oracles: an initial shuffle oracle that, on input* $(\{c_i, \pi_i\}, pk_\ell)$ *for* $pk_\ell \in S$, *outputs* $(\{c'_i\}, \pi, \{pk_\ell\})$ *(if all the proofs of knowledge* π_i *verify), where* π *is a proof that the* $\{c'_i\}$ *constitute a valid shuffle of the* $\{c_i\}$ *performed by the user corresponding to* pk_ℓ *(i.e., the user who knows* sk_ℓ*); and a shuffle oracle that, on input* $(\{c_i, \pi_i\}, \{c'_i\}, \pi, \{pk_j\}, pk_m)$ *for* $pk_m \in S$, *outputs* $(\{c''_i\}, \pi', \{pk_j\} \cup pk_m)$.
- *Step 3. Eventually,* $\mathcal{A}$ *outputs a tuple* $(\{c_i, \pi_i\}, \{c'_i\}, \pi, S' = \{pk_j\})$.
- *Step 4. If* $\mathsf{Verify}(params, (\{c_i, \pi_i\}, \{c'_i\}, \pi, \{pk_j\})) = 1$ *and* $S \cap S' \neq \emptyset$ *then continue; otherwise simply abort and output* $\perp$. *If* $b = 0$ *give* $\mathcal{A}$ $\{\mathsf{Dec}(sk, c'_i)\}$, *and if* $b = 1$ *then give* $\mathcal{A}$ $\varphi(\{\mathsf{Dec}(sk, c_i)\})$, *where* φ *is a random permutation* $\varphi \xleftarrow{\$} S_n$.
- *Step 5.* $\mathcal{A}$ *outputs a guess bit* b'.

Then the shuffle is compactly verifiable *if for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that* $|p_0^{\mathcal{A}}(k) - p_1^{\mathcal{A}}(k)| < \nu(k)$.

In addition to providing this definition, CKLM also provide a generic construction [7, Section 6] of such a shuffle using as building blocks a hard relation [8], a re-randomizable encryption scheme, a proof of knowledge, and a cm-NIZK. We use this generic construction in our shuffle construction in Section 4, and for completeness we give their construction in Appendix B.

2.3 Function privacy

For their application to controlled malleable encryption, CKLM also defined the notion of function privacy for encryption, which had already been used in the literature and is analogous to the notion of derivation privacy for proofs. As we use function-private encryption as a building block in Section 3.2, we give their formal definition here.

First, we follow CKLM in defining a homomorphic encryption scheme as a tuple $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$, where if $c \xleftarrow{\$} \mathsf{Enc}(pk, m)$, $\mathsf{Eval}(pk, c, T) = \mathsf{Enc}(pk, T(m))$ (with some appropriate randomness). For the transformations, we require that $T(m_1, \ldots, m_n) = \perp$ if $m_i = \perp$ for any i, and similarly that $\mathsf{Eval}(pk, \{c_i\}, T) = \perp$ if $\mathsf{Dec}(sk, c_i) = \perp$ for any i.

Definition 2.5. [7] *For a homomorphic encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$*, a given adversary* $\mathcal{A}$*, and a bit* b*, let* $p_b^{\mathcal{A}}(k)$ *be the probability of the event* $b' = 0$ *in the following game:*

- *Step 1.* $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$.
- *Step 2.* $(\mathsf{state}, \{c_i\}, T) \xleftarrow{\$} \mathcal{A}(pk, sk)$.

- *Step 3. If $T \notin \mathcal{T}$ then abort. Otherwise, compute*

$$c' \xleftarrow{\$} \begin{cases} \mathsf{Enc}(pk, T(\{\mathsf{Dec}(sk, c_i)\})) & \text{if } b = 0 \\ \mathsf{Eval}(pk, \{c_i\}, T) & \text{if } b = 1. \end{cases}$$

- *Step 4.* $b' \xleftarrow{\$} \mathcal{A}(\mathsf{state}, c')$.

Then the encryption scheme is function private *with respect to $\mathcal{T}$ if for all PPT algorithms $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that $|p_0^{\mathcal{A}}(k) - p_1^{\mathcal{A}}(k)| < \nu(k)$.*

2.4 Succinct non-interactive arguments of knowledge

Our cm-NIZK construction in Section 3 builds on succinct non-interactive arguments of knowledge, or SNARGs (also called SNARKs) for short. Proofs of this kind were first shown to exist by Micali in 2000 [26], who used the Fiat-Shamir heuristic [14] to eliminate the interaction in previous succinct arguments. More recently, Groth provided a construction using pairings [21] which was improved by Lipmaa [25], Bitansky et al. [3] constructed designated-verifier SNARGs using the new notion of extractable collision-resistant hash functions, and Gennaro et al. [16] constructed constant-sized SNARGs with a relatively short common reference string.

Our definition is based primarily on that of Boneh et al. [6], although for the succinctness property we incorporate the definition of Gentry and Wichs [18] as well. In addition, to perform our recursive composition in Section 3.1, we require a stronger notion of extraction than the original definition provided: essentially, we require that there exists a universal "extractor generator" that when given the code for an adversary can output the code for the corresponding extractor.

Definition 2.6. *Let $0 < \gamma < 1$ be a constant. A* (strong) γ-succinct non-interactive argument of knowledge *for a language $L := \bigcup_{k\in\mathbb{N}} L(k)$ with a witness relation $R_L := \bigcup_{k\in\mathbb{N}} R_{L(k)}$ is a tuple of probabilistic polynomial-time algorithms $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$ with the following properties:*

1. *Perfect completeness. For every $k \in \mathbb{N}$ and $(x, w) \in R_{L(k)}$, and for $\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$ and $\pi \xleftarrow{\$} \mathcal{P}(\mathsf{crs}, x, w)$, the probability that $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ is 1.*

2. *(Strong) Adaptive knowledge extraction. For a PPT algorithm $\mathcal{A}$, let $E_{\mathcal{A}}$ be an associated PPT algorithm. Then consider the following game:*

   - *Step 1.* $\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$; $r \xleftarrow{\$} \{0,1\}^*$.
   - *Step 2.* $(x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}; r)$.
   - *Step 3.* $w \leftarrow E_{\mathcal{A}}(\mathsf{crs}; r)$.

   We say the argument system satisfies adaptive knowledge extraction if for all PPT $\mathcal{A}$ there exists an $E_{\mathcal{A}}$ and a negligible function $\nu(\cdot)$ such that the probability (over the choices of $\mathsf{CRSSetup}$ *and r) that $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ but $(x, w) \notin R_{L(k)}$ is at most $\nu(k)$.*

   In addition, it satisfies strong adaptive knowledge extraction if there exists a PPT algorithm $\mathcal{E}$ such that for all PPT $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that, on input the code of $\mathcal{A}$, $\mathcal{E}$ produces an extractor $E_{\mathcal{A}}$, running in time polynomial in that of $\mathcal{A}$, such that the probability (over the choices of $\mathsf{CRSSetup}$ *and r) that $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ but $(x, w) \notin R_{L(k)}$ is at most $\nu(k)$.*

3. *ϕ-succinct arguments. For every $k \in \mathbb{N}$, $(x, w) \in R_{L(k)}$, and $\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$, it holds that $\mathcal{P}(\mathsf{crs}, x, w)$ produces a distribution over strings of length at most $\phi(k, |x|, |w|)$, where $\phi(k, |x|, |w|)$ is bounded by $poly(k) polylog(|x|) + \gamma|w|$ for some constant $0 < \gamma < 1$.*

While the succinctness property of SNARGs is quite attractive for applications, it comes with a price: all known SNARG constructions are based on so-called "knowledge of exponent" assumptions [9, 2]; furthermore, a recent result due to Gentry and Wichs [18] that separates SNARGS from all falsifiable assumptions suggests that this dependence is perhaps inherent. In addition, to satisfy our stronger version of adaptive knowledge extraction, the knowledge of exponent assumption used to prove the security of existing SNARG constructions [21, 16] would have to be potentially strengthened to consider an extractor that has access to the code of $\mathcal{A}$.

The final observation we make about SNARGs is that the definition of adaptive knowledge extraction requires the extractor to have non-black-box access to the malicious prover; as we will see in Section 3.2, this can make SNARGs difficult to integrate into protocol design. Fortunately, we can easily see that this notion relates to the standard notion of soundness for proofs [13] (as used implicitly in Groth's SNARG construction [21]):

Theorem 2.7. *If a proof system* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$ *satisfies adaptive knowledge extraction then it also satisfies adaptive computational soundness.*

Proof. To show this, we take an adversary $\mathcal{A}$ that can break the soundness of the proof system with non-negligible probability ϵ and use it to construct an adversary $\mathcal{B}$ that breaks adaptive knowledge extraction with the same probability ϵ.

The code for $\mathcal{B}$ is simple: on input $(\mathsf{crs}; r)$, it gives crs to $\mathcal{A}$ (and implicitly runs it on a random tape $r' \subseteq r$), and when $\mathcal{A}$ outputs a pair (x, π) $\mathcal{B}$ outputs the same. By the definition of soundness, $\mathcal{A}$ will win if $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ but $x \notin L_R$; this implies that, for any w output by $E_{\mathcal{B}}$, it must be the case that $(x, w) \notin R$, as otherwise $x \in L_R$. $\mathcal{B}$ will therefore succeed whenever $\mathcal{A}$ does and thus succeeds with probability ϵ. □

3 A Construction of cm-NIZKs from SNARGs

In this section, we construct cm-NIZK proofs from zero-knowledge SNARGs that are malleable with respect to a wide range of transformations, namely all *t-tiered* transformation classes. Intuitively, a relation is t-tiered if each instance x lives in some tier i. We would like transformations to move up through the tiers, and we would also like ensure that at most t transformations are applied. Formally, we say that a relation $R^{(t)}$ is t-tiered if there exists an efficiently computable function $\mathsf{tier} : L_R^{(t)} \to [0, t]$ and $(\bot, \bot) \in R^{(t)}$, and that a transformation class $\mathcal{T}^{(t)}$ is t-tiered for $R^{(t)}$ if for all $T = (T_x, T_w) \in \mathcal{T}$ the following two conditions hold: (1) if $(x, w) \in R^{(t)}$ and $\mathsf{tier}(x) < t$, then $(T_x(x), T_w(w)) \in R^{(t)}$ and $\mathsf{tier}(T_x(x)) > \mathsf{tier}(x)$; and (2) if $\mathsf{tier}(x) = t$ then $T_x(x) = \bot$.

We summarize the contributions in this section in Figure 1. As discussed in the introduction, the construction in each subsection is used as a component in the next subsection's construction, with the end goal of constructing a cm-NIZK. In Section 3.1 we construct a SNARG, malleable with respect to a t-tiered transformation class, that we then use in Section 3.2 in combination with encryption to obtain a full NIZKPoK; this step seems necessary because SNARGs satisfy only the weak notion of adaptive knowledge extraction, which seems insufficient for constructing cm-NIZKs. Finally, using this NIZKPoK and a one-time and regular signature scheme, we construct in Section 3.3 a cm-NIZK that is malleable with respect to a broader class of transformations than could be supported by the construction of CKLM [7].

3.1 From SNARGs to malleable but weakly extractable proofs

We begin by constructing a derivation-private NIZK for a relation $R^{(t)}$, malleable with respect to a t-tiered transformation class $\mathcal{T}^{(t)}$, that achieves some degree of knowledge extraction. Our approach

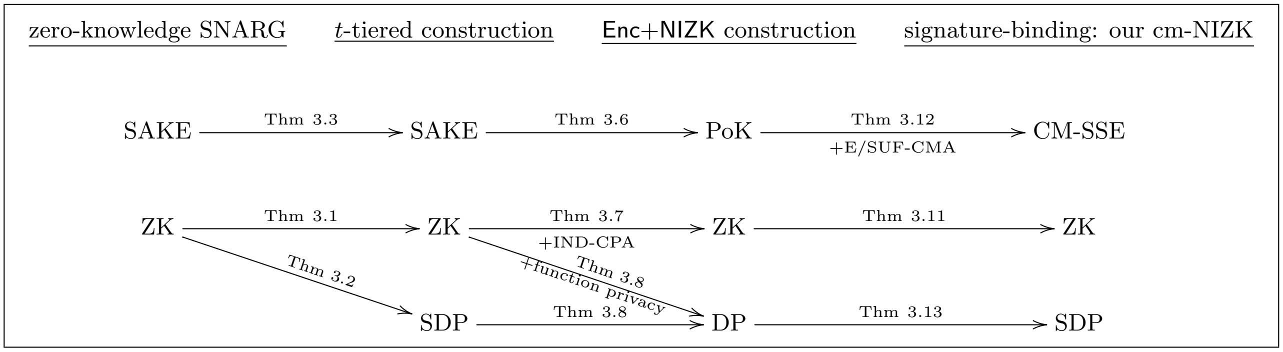

Figure 1: The various relations among our constructions in this section. The arrows indicate which properties of the previous construction are used to obtain which properties of the next one, and are labeled on the top with the theorem number that proves the relation; the labels on the bottom indicate properties of additional primitives that are used as well. For example, we prove in Theorem 3.12 that our signature-binding construction of a cm-NIZK satisfies CM-SSE if our Enc+NIZK construction is a proof of knowledge, and the additional signature and one-time signature schemes we use are, respectively, unforgeable and strongly unforgeable; this is captured by the top rightmost arrow in the diagram. Strong adaptive knowledge extraction is written as SAKE, zero knowledge as ZK, proof of knowledge as PoK, and (strong) derivation privacy as (S)DP.

in this endeavor is inspired by that of Boneh et al. [6], who use SNARGs to construct a "targeted malleable" encryption scheme. To form a proof for an instance x_0 at the bottom level, one can use the SNARG directly to obtain a proof π_0. Now, suppose we would like to further form a proof for an instance $x_1 = T_x(x_0)$; one option is to use the witness $T_w(w_0)$ and form a fresh proof just as we did for x_0. Another option, however, is to "maul" the proof π_0: this can be accomplished by forming a new proof π_1 that proves knowledge of the old proof π_0 and instance x_0, as well as a transformation T such that $x_1 = T_x(x_0)$.

The reason why SNARGs are attractive for this application is that, because the extraction procedure is non-black-box and therefore the proofs can be succinct, the proof π_1 can in fact be the same size as the proof π_0. Continuing in this fashion, we can see that at the i-th level, a proof for x_i can be proved using either knowledge of a witness w_i for the relation $R^{(t)}$, or knowledge of a proof π_{i-1} for x_{i-1} and a transformation T such that $x_i = T_x(x_{i-1})$.

It turns out that, if the SNARG proof system used is zero knowledge (or even just witness indistinguishable), then the resulting proof system is derivation private. As mentioned above, however, the notion of extractability we can satisfy is still only the weak notion of adaptive knowledge extraction that SNARGs provide. In the next section, we show how to bootstrap this construction to obtain a proof system that satisfies the standard notion of extractability for proofs of knowledge (and still satisfies all the malleability and derivation privacy requirements).

To begin our construction, we first formalize the intuition developed above by defining the languages we use: at the bottom level at $i = 0$ we have $L_0 := \left\{x \mid \exists\, w \text{ s.t. } (x, w) \in R^{(t)}\right\}$, and for i such that $1 \leq i \leq t$, we have

$$L_i := \left\{ (x, \mathsf{crs}_{i-1}, \ldots, \mathsf{crs}_0) \;\middle|\; \begin{array}{c} \exists\, (w, x', \pi', T) \text{ s.t } (x, w) \in R^{(t)} \text{ or} \\ \mathcal{V}_{i-1}(\mathsf{crs}_{i-1}, (x', \mathsf{crs}_{i-2}, \ldots, \mathsf{crs}_0), \pi') = 1, \\ T_x(x') = x, \text{ and } T \in \mathcal{T}^{(t)} \end{array} \right\}$$

Before giving our proof construction, we mention that these languages and our subsequent arguments can be generalized to accommodate n-ary transformations, in which the language L_i would be defined as

$$L_i := \left\{ (x, \mathsf{crs}_{i-1}, \ldots, \mathsf{crs}_0) \;\middle|\; \begin{array}{c} \exists\ (w, \{x'_j\}, \{\pi'_j\}, T) \text{ s.t } (x, w) \in R^{(t)} \text{ or} \\ \mathcal{V}_{i-1}(\mathsf{crs}_{i-1}, (x'_j, \mathsf{crs}_{i-2}, \ldots, \mathsf{crs}_0), \pi'_j) = 1\ \forall j, \\ T_x(\{(x'_j)\}) = x, \text{ and } T \in \mathcal{T}^{(t)} \end{array} \right\}$$

Because our shuffle application that we present in Section 4 involves only unary operations, we choose to focus only on the case of $n = 1$ for ease of exposition. We can, however, support n-ary transformations for constant n.

Using these languages and $t + 1$ SNARG systems $(\mathsf{CRSSetup}_i, \mathcal{P}_i, \mathcal{V}_i)$, we now define our malleable t-tiered construction for $R^{(t)}$ as seen in Figure 2.

- $\mathsf{CRSSetup}(1^k)$: Generate $\mathsf{crs}_i \xleftarrow{\$} \mathsf{CRSSetup}_i(1^k)$ for all i, $0 \leq i \leq t$. Output $\mathsf{crs} := (\mathsf{crs}_0, \ldots, \mathsf{crs}_t)$.
- $\mathcal{P}(\mathsf{crs}, x, w)$: Compute $i := \mathsf{tier}(x)$ and output $\pi \xleftarrow{\$} \mathcal{P}_i(\mathsf{crs}_i, (x, \mathsf{crs}_{i-1}, \ldots, \mathsf{crs}_0), (w, \perp, \perp, \perp))$.
- $\mathcal{V}(\mathsf{crs}, x, \pi)$: Compute $i := \mathsf{tier}(x)$ and output $\mathcal{V}_i(\mathsf{crs}_i, (x, \mathsf{crs}_{i-1}, \ldots, \mathsf{crs}_0), \pi)$.
- $\mathsf{ZKEval}(\mathsf{crs}, T, x, \pi)$: Compute $i := \mathsf{tier}(x)$, define $x' := T_x(x)$, and output $\pi \xleftarrow{\$} \mathcal{P}_{i+1}(\mathsf{crs}_{i+1}, (x', \mathsf{crs}_i, \ldots, \mathsf{crs}_0), (\perp, x, \pi, T))$.

Figure 2: Our t-tiered construction of a malleable SNARG.

Recall that there are three properties we would like this proof system to satisfy: (1) zero knowledge, (2) derivation privacy, and (3) strong adaptive knowledge extraction; we deal with each of these in turn. For the first, zero knowledge, if we assume that our underlying proof systems are zero knowledge then we get a proof of the following theorem for free:

Theorem 3.1. *If the SNARG systems* $(\mathsf{CRSSetup}_i, \mathcal{P}_i, \mathcal{V}_i)$ *are zero knowledge, then the t-tiered construction is zero knowledge.*

We next turn to derivation privacy. At first glance, it would seem impossible that our construction could meet derivation privacy: after all, $\mathsf{tier}(x)$ openly reveals exactly how many times a transformation has been applied! Looking at the definition of the prover $\mathcal{P}$, however, we see that for x such that $\mathsf{tier}(x) = i$ it does in fact output a proof that "looks like" i transformations have been applied, even though it is using a fresh witness; as this is what the definition of derivation privacy requires (i.e., that the proof, rather than the instance, not reveal the transformation), we therefore use the witness indistinguishability of the SNARGs (which trivially follows from zero knowledge) to show that derivation privacy does hold. In addition, to show that strong derivation privacy holds, we require our SNARGs to be composable zero knowledge (as the adversary in the strong derivation privacy game gets to see the simulation trapdoor, and thus the zero knowledge adversary needs to as well); this requirement is met by the SNARG constructions of Groth [21] and Gennaro et al. [16], both of which actually satisfy the significantly stronger property of statistical zero knowledge.

Theorem 3.2. *If the SNARG systems* $(\mathsf{CRSSetup}_i, \mathcal{P}_i, \mathcal{V}_i)$ *satisfy witness indistinguishability, then the t-tiered construction satisfies derivation privacy for transformations in $\mathcal{T}^{(t)}$. Furthermore, if the SNARG systems* $(\mathsf{CRSSetup}_i, \mathcal{P}_i, \mathcal{V}_i)$ *satisfy composable zero-knowledge, then the t-tiered construction satisfies both derivation privacy and strong derivation privacy for transformations in $\mathcal{T}^{(t)}$.*

Proof. To show both derivation privacy and strong derivation privacy, we first observe a relation between an adversary's overall advantage and its advantage at a specific tier. In particular, suppose there exists an adversary $\mathcal{A}$ that wins either the derivation privacy or strong derivation privacy game with advantage ϵ. If we let p_i be the probability that $\mathcal{A}$ outputs an instance in L_i and a_i be the advantage of $\mathcal{A}$ when it outputs an instance in L_i, then there must be some level $i_0 \in \{0, \ldots, t\}$ such that $p_{i_0} a_{i_0} \geq \epsilon/t$. To see this, note that there are no valid transformations for levels $i \geq t$, so if the adversary outputs an instance in L_i for $i \geq t$ it will have advantage 0; $\mathcal{A}$'s overall advantage ϵ is thus at most $\sum_{i=0}^{t} p_i a_i$ by the triangle inequality, and our inequality follows by an averaging argument.

Now, using this fact, we first show that if there exists an adversary $\mathcal{A}$ that wins at the derivation privacy game with non-negligible advantage ϵ, then we can construct an adversary $\mathcal{B}$ that can distinguish between witnesses at level i_0 (where i_0 is the level such that $p_{i_0} a_{i_0} \geq \epsilon/t$) with related non-negligible advantage. To start, $\mathcal{B}$ will receive as input a CRS crs_{i_0} (which will be the output of $\mathsf{CRSSetup}_{i_0}(1^k)$). It can then form $\mathsf{crs}_j \xleftarrow{\$} \mathsf{CRSSetup}_j(1^k)$ for all j, $0 \leq j \leq t$ such that $j \neq i_0$, and give $\mathsf{crs} := (\mathsf{crs}_0, \ldots, \mathsf{crs}_t)$ to $\mathcal{A}$. At some point, $\mathcal{A}$ will give back to $\mathcal{B}$ a tuple of the form $(\mathsf{state}, x, w, \pi, T)$. $\mathcal{B}$ can then check that $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$, $(x, w) \in R^{(t)}$, and $T \in \mathcal{T}$; if any of these checks fail then $\mathcal{A}$ is not behaving properly and $\mathcal{B}$ can return $\perp$ to $\mathcal{A}$. Let $\mathsf{tier}(x) = i$. $\mathcal{B}$ can then check further that $i + 1 = i_0$; if this fails then $\mathcal{B}$ outputs a random bit. Otherwise, $\mathcal{B}$ can compute $x' := (T_x(x), \mathsf{crs}_i, \ldots, \mathsf{crs}_0)$, $w_0 := (T_w(w), \perp, \perp, \perp)$, and $w_1 := (\perp, x, \pi, T)$ and output (x', w_0, w_1) as its query to get back a proof π. It can then return π to $\mathcal{A}$, and when $\mathcal{A}$ outputs its guess bit $\mathcal{B}$ will output the same bit.

To see that interactions with $\mathcal{B}$ are indistinguishable from honest interactions, we observe first that the CRS given to $\mathcal{A}$ is identical to the one it was expecting. As for the proof, note that if π is formed using w_0 then it is the output of $\mathcal{P}_{i+1}(\mathsf{crs}_{i+1}, (T_x(x), \mathsf{crs}_i, \ldots, \mathsf{crs}_0), (T_w(w), \perp, \perp, \perp)) = \mathcal{P}(\mathsf{crs}, T'_x(x), T_w(w))$, which is exactly what $\mathcal{A}$ was expecting in the case that $b = 0$. If instead π is formed using w_1 then it is the output of $\mathcal{P}_{i+1}(\mathsf{crs}_{i+1}, (T_x(x), \mathsf{crs}_i, \ldots, \mathsf{crs}_0), (\perp, x, \pi, T)) = \mathsf{ZKEval}(\mathsf{crs}, T, x, \pi)$, which is exactly what it is expecting in the case that $b = 1$. As $\mathcal{B}$ will therefore succeed with advantage a_{i_0} whenever $\mathcal{A}$ outputs an instance at level i_0, and $\mathcal{A}$'s view is identical to that in the derivation privacy game so it outputs an instance at level i_0 with probability p_{i_0}, $\mathcal{B}$ will succeed with probability $p_{i_0} a_{i_0} \geq \epsilon/t$.

Next, we show that if the SNARG systems $(\mathsf{CRSSetup}_i, \mathcal{P}_i, \mathcal{V}_i)$ satisfy composable zero-knowledge, then the t-tiered construction satisfies strong derivation privacy. To show this, we assume there exists an adversary $\mathcal{A}$ that can win at the strong derivation privacy game with non-negligible advantage ϵ and use it to construct an adversary $\mathcal{B}$ that can distinguish between real and simulated proofs at level i_0 with related non-negligible advantage.

First, recall that the simulator for the t-tiered proof system simply runs the simulator for the appropriate SNARG system $(\mathsf{CRSSetup}_i, \mathcal{P}_i, \mathcal{V}_i)$; for each $i \in \{0, \ldots, t\}$, let $S_1^{(i)}, S_2^{(i)}$ be the simulator corresponding to $(\mathsf{CRSSetup}_i, \mathcal{P}_i, \mathcal{V}_i)$. To start, $\mathcal{B}$ will receive as input a pair $(\mathsf{crs}_{i_0}, \tau_{i_0})$ (which will be the output of the simulator $S_1^{(i_0)}$ corresponding to $(\mathsf{CRSSetup}_{i_0}, \mathcal{P}_{i_0}, \mathcal{V}_{i_0})$). It can then form $(\mathsf{crs}_j, \tau_j) \xleftarrow{\$} S_1^{(j)}(1^k)$ for all j, $0 \leq j \leq t$ such that $j \neq i_0$, and give $\mathsf{crs} := (\mathsf{crs}_0, \ldots, \mathsf{crs}_t)$ and $\tau_s := (\tau_0 \ldots \tau_t)$ to $\mathcal{A}$. At some point, $\mathcal{A}$ will give back to $\mathcal{B}$ a tuple of the form $(\mathsf{state}, x, \pi, T)$. Let $\mathsf{tier}(x) = i$. $\mathcal{B}$ can then check that $T \in \mathcal{T}$; if this check fails then $\mathcal{A}$ is not behaving properly and $\mathcal{B}$ can return $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ can then check further that $i + 1 = i_0$; if this fails then $\mathcal{B}$ outputs a random bit. Otherwise, $\mathcal{B}$ can compute $x' := (T_x(x), \mathsf{crs}_i, \ldots, \mathsf{crs}_0)$, and $w' = (\perp, x, \pi, T)$ and query its oracle on (x', w') to get back a proof π'. It can then return π' to $\mathcal{A}$, and when $\mathcal{A}$ outputs its guess bit $\mathcal{B}$ will output the same bit.

To see that interactions with $\mathcal{B}$ are indistinguishable from the strong derivation privacy game, we observe first that the CRS given to $\mathcal{A}$ is identical to the one it was expecting. As for the proof, note that if $\mathcal{B}$'s proof oracle produces a real proof using $\mathcal{P}_{i_0}$, then π is the output of $\mathcal{P}_{i+1}(\mathsf{crs}_{i+1}, (T_x(x), \mathsf{crs}_i, \ldots, \mathsf{crs}_0), (\perp, x, \pi, T)) = \mathsf{ZKEval}(\mathsf{crs}, T, x, \pi)$, which is exactly what $\mathcal{A}$ was expecting in the case that $b = 1$. If instead $\mathcal{B}$'s proof oracle produces a simulated proof π using using $S_2^{(i)}(x, \mathsf{crs}_i, \ldots, \mathsf{crs}_0)$, then this is identical to the output of the t-tiered simulator $S_2(x)$, which is exactly what $\mathcal{A}$ was expecting in the case that $b = 0$. As $\mathcal{B}$ will therefore succeed with advantage a_i whenever $\mathcal{A}$ outputs an instance at level i, and $\mathcal{A}$'s view is identical to that in the strong derivation privacy game so it outputs an instance at level i with probability p_i, $\mathcal{B}$ will succeed with probability $p_{i_0} a_{i_0} \geq \epsilon/t$. □

Next, we turn to adaptive knowledge extraction; here, we can show that if the number of times the "proof of a proof" method has been applied is constant, then the t-tiered construction is strongly adaptive knowledge extractable. As do Boneh et al. [6], we require t be constant so the runtime of the extractor does not blow up: if $\mathcal{A}$ runs in time τ, and we require the runtime of the extractor to be only polynomial in the runtime of $\mathcal{A}$, then the extraction of the t-th nested proof (i.e., if $\mathcal{A}$ has formed a proof of a proof t times) might take time $a^t\tau + tb$ for some constants a and b, which for arbitrary t could be exponential. To ensure that the time taken to extract from these nested proofs instead remains polynomial, we therefore require that t be constant. Furthermore, as we will see in the proof we rely on strong adaptive knowledge extraction to perform our recursive extraction (again, as do Boneh et al.)

Theorem 3.3. *If the SNARG systems $(\mathsf{CRSSetup}_i, \mathcal{P}_i, \mathcal{V}_i)$ satisfy strong adaptive knowledge extraction (as defined in Definition 2.6), then the t-tiered construction satisfies strong adaptive knowledge extraction for constant t.*

Proof. To show this, we proceed inductively. We will first show that if we use only a single tier ($t = 0$), then the t-tiered construction satisfies strong adaptive knowledge extraction whenever the underlying SNARG does. Then we will show that for any constant ℓ, if the $(\ell - 1)$-tiered construction and the SNARG used in the ℓ-th tier both satisfy strong adaptive knowledge extraction, then the ℓ-tiered construction satisfies it as well.

Base case: $t = 0$. At the first level, $t = 0$, we can show that if the SNARG system $(\mathsf{CRSSetup}_0, \mathcal{P}_0, \mathcal{V}_0)$ satisfies strong adaptive knowledge extraction, then the 0-tiered construction does as well. As the two primitives are essentially equivalent, this is quite straightforward: if we have an adversary $\mathcal{A}_{0\text{-}tiered}$ against the 0-tiered construction that takes in a CRS crs_0 and a random tape r and outputs a pair (x, π), then we can construct an adversary $\mathcal{A}_0$ against the SNARG that, on input a CRS crs_0 and a random tape r, runs $(x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}_0; r)$ and outputs (x, π).

By strong adaptive knowledge extraction, we also have an extractor generator $\mathcal{E}_0$ that, given the code of $\mathcal{A}_0$, outputs a corresponding extractor E_0. To use this to construct an extractor generator $\mathcal{E}_{0\text{-}tiered}$ for the 0-tiered construction, on input the adversary $\mathcal{A}_{0\text{-}tiered}$, $\mathcal{E}_{0\text{-}tiered}$ will construct the adversary $\mathcal{A}_0$ described above, and then output $E_{0\text{-}tiered} := E_0 \leftarrow \mathcal{E}_0(\mathcal{A}_0)$.

We now need to show two things: first, that $\mathcal{E}_{0\text{-}tiered}$ runs in polynomial time and produces extractors whose running time is a fixed polynomial of the corresponding adversaries, and next that these extractors succeed in producing valid witnesses whenever the adversaries produce valid proofs.

First, we observe that as $\mathcal{E}_{0\text{-}tiered}$ just outputs whatever $\mathcal{E}_0$ does, by the assumption that $\mathcal{E}_0$ runs in polynomial time, $\mathcal{E}_{0\text{-}tiered}$ will as well. Furthermore, by assumption there also exists some polynomial $poly_0$ such that for any SNARG adversary $\mathcal{A}_0$ running in time t, the resulting extractor E_0 produced by $\mathcal{E}_0$ runs in time $poly_0(t)$. Then, since our SNARG adversary $\mathcal{A}_0$ just runs the 0-tiered adversary $\mathcal{A}_{0\text{-}tiered}$, if $\mathcal{A}_{0\text{-}tiered}$ runs in time t then $\mathcal{A}_0$ will as well. Thus, the resulting $E_{0\text{-}tiered} = E_0$ will have running time $poly_0(t)$.

Next, note that by construction, (x, π) is accepted by the 0-tiered verifier if and only if (x, π) is accepted by the SNARG verifier. Thus, whenever $\mathcal{A}_{0\text{-}tiered}$ produces an accepting proof, $\mathcal{A}_0$ also produces an accepting proof. Now, suppose there exists an adversary $\mathcal{A}_{0\text{-}tiered}$ such that, for $E_{0\text{-}tiered} \leftarrow \mathcal{E}_{0\text{-}tiered}$, with non-negligible probability $\mathcal{A}_{0\text{-}tiered}(\mathsf{crs}_0; r)$ outputs (x, π) and $E_{0\text{-}tiered}(\mathsf{crs}_0; r)$ outputs w such that $\mathcal{V}(\mathsf{crs}_0, x, \pi) = 1$ but $(x, w) \notin R^{(t)}$; i.e., suppose $\mathcal{A}_{0\text{-}tiered}$ wins at its game. Then if we consider the $\mathcal{A}_0$ constructed from $\mathcal{A}_0$-*tiered* as above, by our construction of $\mathcal{E}_{0\text{-}tiered}$ it must be the case that $\mathcal{E}_0$ on input $\mathcal{A}_0$ produces E_0 such that with non-negligible probability $\mathcal{A}_0(\mathsf{crs}_0; r)$ outputs (x, π) and $E_0(\mathsf{crs}_0; r)$ outputs w such that $\mathcal{V}(\mathsf{crs}_0, x, \pi) = 1$ but $(x, w) \notin R_0$. This would contradict the strong adapative extraction property of the SNARG.

Inductive step: the ℓ-tiered construction. We now proceed to our inductive step, in which we show that, if the $(\ell - 1)$-tiered construction and the SNARG system $(\mathsf{CRSSetup}_\ell, \mathcal{P}_\ell, \mathcal{V}_\ell)$ both satisfy strong adaptive knowledge extraction, then the ℓ-tiered construction satisfies strong adaptive knowledge extraction. To show this, we proceed in three steps: first, we show how, given any adversary $\mathcal{A}$ against the ℓ-tiered construction, we can construct adversaries $\mathcal{A}_\ell$ against the ℓ-th SNARG system and $\mathcal{A}_{base}$ against the $(\ell - 1)$-tiered construction. Now, by assumption, there exist extractor generators $\mathcal{E}_\ell$ and $\mathcal{E}_{base}$ for the ℓ-th SNARG and $(\ell - 1)$-tiered construction respectively, which we then use to construct an extractor generator $\mathcal{E}$ for the ℓ-tiered construction; we furthermore show that $\mathcal{E}$ runs in polynomial time and that the extractors produced by $\mathcal{E}$ have a runtime that is a fixed polynomial of the runtime of the corresponding adversaries. Finally, we show that if there exists an $\mathcal{A}$ such that $E \leftarrow \mathcal{E}(\mathcal{A})$ fails to produce a valid witness (i.e., $\mathcal{A}$ wins at the strong adaptive knowledge extraction game) then either $E_\ell \leftarrow \mathcal{E}_\ell(\mathcal{A}_\ell)$ or $E_{base} \leftarrow \mathcal{E}_{base}(\mathcal{A}_{base})$ must have failed as well, which contradicts our assumption about either the $(\ell - 1)$-tiered construction or the ℓ-th SNARG system and thus E cannot fail with more than negligible probability, meaning the ℓ-tiered construction satisfies strong adaptive knowledge extraction as well.

To define $\mathcal{A}_{base}$ and $\mathcal{A}_\ell$ given $\mathcal{A}$, we recall that $\mathcal{A}$ is given $\mathsf{crs} = (\mathsf{crs}_0, \ldots, \mathsf{crs}_\ell)$ and outputs a value of the form (x, π) with $\mathsf{tier}(x) = i$, $0 \leq i \leq \ell$, but that both $\mathcal{A}_{base}$ and $\mathcal{A}_\ell$ will receive as input only part of this crs. Rather than have them form the missing CRS values themselves (as this would cause a problem for the recursive extraction), we instead give the remaining CRS values as *hard-coded* inputs, and denote the hard-coded adversaries as $\mathcal{A}_{base}^{\mathsf{crs}_\ell}$ and $\mathcal{A}_\ell^{\mathsf{crs}_0, \ldots, \mathsf{crs}_{\ell-1}}$ respectively. Then $\mathcal{A}_\ell^{\mathsf{crs}_0, \ldots, \mathsf{crs}_{\ell-1}}$ proceeds as follows:

1. On input crs_ℓ and a random tape r, define $\mathsf{crs} := (\mathsf{crs}_0, \ldots, \mathsf{crs}_\ell)$ using the hard-coded values of $\mathsf{crs}_0, \ldots, \mathsf{crs}_{\ell-1}$.
2. Compute $(x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}; r)$. If $\mathsf{tier}(x) = \ell$ then output $(\hat{x} := (x, \mathsf{crs}_{\ell-1}, \ldots, \mathsf{crs}_0), \pi)$, and otherwise output $\perp$.

In terms of efficiency, we observe that the runtime of $\mathcal{A}_\ell^{\mathsf{crs}_0, \ldots, \mathsf{crs}_{\ell-1}}$ will be very close to the runtime of $\mathcal{A}$, as its only overhead beyond $\mathcal{A}$ is checking a condition and modifying the output; its runtime will therefore certainly be polynomial in the runtime of $\mathcal{A}$. We can also define $\mathcal{A}_{base}^{\mathsf{crs}_\ell}$ as follows:

1. On input $(\mathsf{crs}_0, \ldots, \mathsf{crs}_{\ell-1})$ and r, define $\mathsf{crs} := (\mathsf{crs}_0, \ldots, \mathsf{crs}_\ell)$ using the hard-coded value of crs_ℓ.
2. Compute $(x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}; r)$. If $\mathsf{tier}(x) < \ell$, output (x, π). Otherwise, if $\mathsf{tier}(x) = \ell$, then construct the adversary $\mathcal{A}_\ell^{\mathsf{crs}_0, \ldots, \mathsf{crs}_{\ell-1}}$ defined above and generate $E_\ell \leftarrow \mathcal{E}_\ell(\mathcal{A}_\ell^{\mathsf{crs}_0, \ldots, \mathsf{crs}_{\ell-1}})$; run $(w_\ell, x_{\ell-1}, \pi_{\ell-1}, T_\ell) \leftarrow E_\ell(\mathsf{crs}; r)$. If $(x, w_\ell) \notin R^{(t)}$ then output $(x_{\ell-1}, \pi_{\ell-1})$, and otherwise output $\perp$.

Again, the runtime of $\mathcal{A}_{base}^{\mathsf{crs}_\ell}$ is polynomial in the runtime of $\mathcal{A}$, as by assumption the runtime of both $\mathcal{E}_\ell$ and E_ℓ is polynomial in the runtime of $\mathcal{A}_\ell^{\mathsf{crs}_0, \ldots, \mathsf{crs}_{\ell-1}}$ which, as just argued, is itself polynomial in the runtime of $\mathcal{A}$.

Next, we define $\mathcal{E}$ given the underlying extractor generators $\mathcal{E}_\ell$ and $\mathcal{E}_{base}$; given the code of an adversary $\mathcal{A}$, $\mathcal{E}$ produces an extractor E that behaves as follows:

On input $\mathsf{crs} = (\mathsf{crs}_0, \ldots, \mathsf{crs}_\ell)$ and r, run $(x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}; r)$.

1. If $\mathsf{tier}(x) < \ell$ then output $w' \leftarrow E_{base}((\mathsf{crs}_0, \ldots, \mathsf{crs}_{\ell-1}); r)$.
2. If $\mathsf{tier}(x) = \ell$, compute $\hat{w} = (w_\ell, x_{\ell-1}, \pi_{\ell-1}, T_\ell) \leftarrow E_\ell(\mathsf{crs}_\ell; r)$. Let $\hat{x} := (x, \mathsf{crs}_{\ell-1}, \ldots, \mathsf{crs}_0)$.
   (a) If $(\hat{x}, \hat{w}) \notin R_\ell$, output $\perp$.
   (b) If $(x, w_\ell) \in R^{(t)}$ output w_ℓ.
   (c) If $(x, w_\ell) \notin R^{(t)}$, compute $w' \leftarrow E_{base}((\mathsf{crs}_0, \ldots, \mathsf{crs}_{\ell-1}); r)$ and output $T_\ell(w')$.

First, we show that both $\mathcal{E}$ and E are efficient. For $\mathcal{E}$, to provide E with the code for E_{base}, it must first generate $\mathcal{A}_{base}^{\mathsf{crs}_\ell}$ and run $E_{base} \leftarrow \mathcal{E}_{base}(\mathcal{A}_{base}^{\mathsf{crs}_\ell})$; similarly, to provide the code for E_ℓ it must generate $\mathcal{A}_\ell^{\mathsf{crs}_0,\ldots,\mathsf{crs}_{\ell-1}}$ and run $E_\ell \leftarrow \mathcal{E}_\ell(\mathcal{A}_\ell^{\mathsf{crs}_0,\ldots,\mathsf{crs}_{\ell-1}})$ (and perform some basic additional operations at a constant cost). By assumption, both the extractor generators have a runtime polynomial in the runtime of their respective adversaries; furthermore, by our earlier discussions, both these adversaries have a runtime polynomial in the runtime of $\mathcal{A}$. The runtime of $\mathcal{E}$ can therefore be some fixed polynomial in the runtime of $\mathcal{A}$.

Moving on to E, once again by assumption the runtime of E_{base} is polynomial in the runtime of $\mathcal{A}_{base}^{\mathsf{crs}_\ell}$, and the runtime of E_ℓ is polynomial in the runtime of $\mathcal{A}_\ell^{\mathsf{crs}_0,\ldots,\mathsf{crs}_{\ell-1}}$. Again, because the runtimes of $\mathcal{A}_{base}^{\mathsf{crs}_\ell}$ and $\mathcal{A}_\ell^{\mathsf{crs}_0,\ldots,\mathsf{crs}_{\ell-1}}$ are themselves polynomial in the runtime of $\mathcal{A}$, the overall runtime of E can also be polynomial in the runtime of $\mathcal{A}$.

Finally, we show that if with non-negligible probabilty when run on $\mathsf{crs} = (\mathsf{crs}_0, \ldots, \mathsf{crs}_\ell)$ and r, $\mathcal{A}$ outputs (x, π) and $E \leftarrow \mathcal{E}(\mathcal{A})$ outputs w such that $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ but $(x, w) \notin R^{(t)}$, we break either the strong adaptive knowledge extraction of the SNARG for level ℓ or the $\ell - 1$-tiered SNARG.

To show this, we consider four cases (based on the four cases used by E): either (1) $\mathsf{tier}(x) < \ell$, (2.a) $\mathsf{tier}(x) = \ell$ and $(\hat{x}, \hat{w}) \notin R_\ell$, (2.b) $\mathsf{tier}(x) = \ell$ and $(x, w_\ell) \in R^{(t)}$, or (2.c) $\mathsf{tier}(x) = \ell$ and $(x, w_\ell) \notin R^{(t)}$.

1. If $\mathsf{tier}(x) < \ell$, then by definition $\mathcal{A}_{base}^{\mathsf{crs}_\ell}$ outputs the same (x, π) as $\mathcal{A}$, and E outputs the same w as E_{base}. If it is therefore the case that $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ but $(x, w) \notin R^{(t)}$, then it is also the case that $\mathcal{V}(\mathsf{crs}, x', \pi') = 1$ but $(x', w') \notin R^{(t)}$ too, as they are all just the same values. Thus, if with non-negligible probability $(x, w) \notin R^{(t)}$ and this case occurs, then $\mathcal{A}_{base}^{\mathsf{crs}_\ell}$ breaks the strong extractability of the $(\ell - 1)$-tiered SNARG.

2.a If $\mathsf{tier}(x) = \ell$ and $(\hat{x}, \hat{w}) \notin R_\ell$, then by definition $\mathcal{A}_\ell^{\mathsf{crs}_0,\ldots,\mathsf{crs}_{\ell-1}}$ outputs $(\hat{x}, \pi)$, and E_ℓ outputs $\hat{w}$ such that $(\hat{x}, \hat{w}) \notin R_\ell$. By construction, $\mathcal{V}_\ell(\mathsf{crs}_\ell, \hat{x}, \pi) = 1$ whenever $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$, so if the proof output by $\mathcal{A}$ is accepted then the proof output by $\mathcal{A}_\ell^{\mathsf{crs}_0,\ldots,\mathsf{crs}_{\ell-1}}$ will be accepted too. Thus if this case occurs with non-negligible probability then $\mathcal{A}_\ell^{\mathsf{crs}_0,\ldots,\mathsf{crs}_{\ell-1}}$ contradicts the strong extractability of the ℓ-th SNARG.

2.b The case $\mathsf{tier}(x) = \ell$ and $(x, w_\ell) \in R^{(t)}$ cannot lead to a successful $\mathcal{A}$, since in this case E produces w_ℓ, which is by definition a valid witness w such that $(x, w) \in R^{(t)}$.

2.c If, on the other hand, $\mathsf{tier}(x) = \ell$ and $(x, w_\ell) \notin R^{(t)}$, then we know that $\mathcal{A}_{base}^{\mathsf{crs}_\ell}$ outputs $(x_{\ell-1}, \pi_{\ell-1})$ (where we recall these values are pulled from the output of E_ℓ), and that E output $w = T_\ell(w')$. Since we know that $(\hat{x}, \hat{w}) \in R_\ell$, that means T_ℓ will map a valid witness w' for $x_{\ell-1}$ to a valid witness for x. Thus we know by the way w was computed that $(x, w) \notin R^{(t)}$ implies $(x_{\ell-1}, w') \notin R^{(t)}$. Furthermore, $(\hat{x}, \hat{w}) \in R_\ell$ also implies that $\pi_{\ell-1}$ is an accepting proof for $x_{\ell-1}$. Thus if with non-negligible probability $(x, w) \notin R^{(t)}$ and this case occurs, then $\mathcal{A}_{base}^{\mathsf{crs}_\ell}$ breaks the strong extractability of the $(\ell - 1)$-tiered SNARG.

As we have therefore demonstrated that any winning case for $\mathcal{A}$ results in either $\mathcal{A}_\ell^{\mathsf{crs}_0,\ldots,\mathsf{crs}_{\ell-1}}$ or $\mathcal{A}_{base}^{\mathsf{crs}_\ell}$ winning as well, which is a contradiction, it must be the case that the ℓ-tiered construction satisfies strong adaptive knowledge extraction as well. □

Finally, we discuss the size of the proofs. Looking at the language L_i for some level, we see that an instance for the next language L_{i+1} consists of the same elements as an instance of L_i, with the addition of the CRS crs_i. Following Boneh et al. [6], we use for concreteness the SNARG construction of Groth [21], which, if we define $x^{(i)}$ to be an instance of L_i, requires the size of crs_i to be $O(|x^{(i)}|^2)$. Let f be the function that computes the size of the instance at level $i+1$ given the size of the instance x at level i. Then, because an element of size $|x|^2$ is added to obtain the instance for the next level up, we have that $f(f(|x|)) = |x|^4$, and, after t transformations, that $f^t(|x_0|) > |x_0|^{2^t}$. If t is constant, the fact that we require SNARGs to be of size polylog($|x|$) accounts for every such polynomial factor. Considering next the witness, we observe that the size of the witness $w^{(i)}$ for $i > 0$ is $|w_i| + |x_{i-1}| + |\pi_{i-1}| + |T_i|$. In order for our proofs to be succinct, we require that $|\pi_i| \leq |\pi_{i-1}|$. If we assume that $|w_i| \leq |w_{i-1}|$, $|x_i| \leq |x_{i-1}|$, and $|T_i| \leq |T_{i-1}|$ and that $w^{(i)} = |w_i| + |x_{i-1}| + |\pi_{i-1}| + |T_i| \leq 4|\pi_{i-1}|$, then a $\text{poly}(k)\text{polylog}(|x|) + \gamma|w|$ succinct SNARG with $\gamma = 1/4$ is sufficient for our construction.

3.2 From weak malleable proofs to malleable proofs of knowledge

With our malleable NIZK in place, we might now try to use it to directly construct a cm-NIZK or, because we can satisfy only adaptive knowledge extraction, a weakened notion of cm-NIZK that accomodates this weaker extractability property. Looking back at the definition of controlled malleability (CM-SSE) in Definition 2.1, however, we can see that $\mathcal{A}$ is given access to a simulation oracle S_2. This oracle access seems to be fundamental to the definition: to achieve any kind of simulation soundness, in which we want $\mathcal{A}$ to be unable to produce its own proofs of false statements even after seeing many such proofs, we must give it an oracle that can produce false proofs. If we attempt to then use any non-black-box notion of extractability in conjunction with such an oracle, it is not clear how such an extractor would even be defined, as it cannot simply run the code for $\mathcal{A}$ (in particular, because the oracle's ability to produce false proofs must be presumably unavailable to $\mathcal{A}$ and therefore $E_{\mathcal{A}}$).

To avoid this obstacle altogether, we instead augment the construction from the previous section to achieve full extractability. To do this, our proofs consist of a ciphertext encrypting the witness, and a malleable zero-knowledge SNARG proving knowledge of the value inside of this ciphertext. Now, rather than require the use of the non-black-box extractor to prove any kind of extractability, we can instead give an extractor the secret key, and it can extract by decrypting the ciphertext. As we will see in our proof of Theorem 3.6, this means that all is required of the SNARG is soundness (which, we recall by Theorem 2.7, is implied by adaptive knowledge extraction).

In more detail, to construct a malleable NIZKPoK for a relation $R^{(pok)}$ and transformation class $\mathcal{T}^{(pok)}$, we use an encryption scheme and a proof system for the relation $R^{(t)}$ such that

$$((pk, x, c), (w, r)) \in R^{(t)} \iff c = \mathsf{Enc}(pk, w; r) \wedge (x, w) \in R^{(pok)}.$$

As for malleability, suppose we want to be able to transform the proofs for $R^{(pok)}$ with respect to some transformation class $\mathcal{T}^{(pok)}$. In order to implement ZKEval for a transformation $T = (T_x, T_w) \in \mathcal{T}^{(pok)}$, we will need to be able to transform the proof for $R^{(t)}$ and the ciphertext c. For the latter, this means we need to be able to apply a transformation T_c on the ciphertext that produces an encryption of $T_w(w)$; i.e., the homomorphic property of the encryption scheme must be robust enough to allow us to apply T_w to the encrypted message. For the proof, we also require a transformation T_r on the randomness r of the ciphertext, as we require a transformation that maps (pk, x, c) to $(pk, T_x(x), T_c(c))$ and (w, r) to $(T_w(w), T_r(r))$.

A bit more formally, for every $T = (T_x, T_w) \in \mathcal{T}^{(pok)}$ and r' from the randomness space $\mathcal{R}$, let T_c be the transformation which maps $c = \mathsf{Enc}(w; r)$ to $\mathsf{Eval}(c, T_w; r') = \mathsf{Enc}(T_w(w); r \circ r')$ (where $\circ$ denotes the operation that composes the randomness, and Eval denotes the homomorphic operation on ciphertexts), let T_r be the resulting transformation on the randomness, and let $\tau(T, r')$ be the transformation that maps instances (x, c) to new instances $(T_x(x), T_c(c))$, and witnesses (w, r) to new witnesses $(T_w(w), T_r(r))$ (i.e., the exact transformation we need for the proof). Finally, let $\mathcal{T}^{(t)}$ be the set of transformations that includes $\tau(T, r')$ for all $T \in \mathcal{T}^{(pok)}$, $r' \in \mathcal{R}$, and let $\mathcal{T}^{(E)}$ be the set of all T_w.

To give our Enc+NIZK construction for $R^{(pok)}$, let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a function-private homomorphic encryption scheme (as defined in Definition 2.5) with randomness space $\mathcal{R}$ and let $(\mathsf{CRSSetup}', \mathcal{P}', \mathcal{V}', \mathsf{ZKEval}')$ be a malleable zero-knowledge SNARG for the relation $R^{(t)}$ with transformation set $\mathcal{T}^{(t)}$. See Figure 3 for our construction of a NIZKPoK using these primitives.

- $\mathsf{CRSSetup}(1^k)$: Generate $\mathsf{crs}' \xleftarrow{\$} \mathsf{CRSSetup}'(1^k)$ and $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ and output $\mathsf{crs} := (\mathsf{crs}', pk)$.
- $\mathcal{P}(\mathsf{crs}, x, w)$: Parse $\mathsf{crs} = (\mathsf{crs}', pk)$ and pick randomness $r \xleftarrow{\$} \mathcal{R}$. Then compute $c \leftarrow \mathsf{Enc}(pk, w; r)$ and $\pi' \xleftarrow{\$} \mathcal{P}'(\mathsf{crs}', (pk, x, c), (w, r))$ and output $\pi := (\pi', c)$.
- $\mathcal{V}(\mathsf{crs}, x, \pi)$: Parse $\mathsf{crs} = (\mathsf{crs}', pk)$ and $\pi = (\pi', c)$. Then output $\mathcal{V}'(\mathsf{crs}', (pk, x, c), \pi')$.
- $\mathsf{ZKEval}(\mathsf{crs}, T, x, \pi)$: Parse $\mathsf{crs} = (\mathsf{crs}', pk)$, $\pi = (\pi', c)$, and $T = (T_x, T_w)$. Then choose random $r' \xleftarrow{\$} \mathcal{R}$, compute $T' := \tau(T, r')$, and compute $\pi_T \xleftarrow{\$} \mathsf{ZKEval}'(\mathsf{crs}', T', (pk, x, c), \pi')$ and $c_T := \mathsf{Eval}(pk, T_w, c; r')$. Output (π_T, c_T).

Figure 3: Our Enc+NIZK construction of a NIZKPoK.

We make the following requirements on the underlying SNARG to obtain the completeness and malleability properties; note that both of them follow directly from the Enc+NIZK construction:

Theorem 3.4. *Let $\mathbb{W}^{(E+N)}$ be the witness space for $R^{(pok)}$. If the SNARG is complete for $R^{(t)}$ and the encryption scheme has message space $\mathcal{M}$ such that $\mathbb{W}^{(E+N)} \subseteq \mathcal{M}$, then the Enc+NIZK construction is complete.*

Theorem 3.5. *The Enc+NIZK construction is malleable with respect to $\mathcal{T}^{(pok)}$ whenever the SNARG is malleable with respect to the corresponding set $\mathcal{T}^{(t)} = \tau(\mathcal{T}^{(pok)}, \mathcal{R})$ and the encryption scheme is malleable with respect to $\mathcal{T}^{(E)}$ (as defined above).*

Note that, if $\mathcal{T}^{(pok)}$ is a t-tiered class of transformations on $R^{(pok)}$, then $\tau(\mathcal{T}^{(pok)})$ will also be t-tiered on $R^{(t)}$. Thus, if we instantiate $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ using a fully homomorphic encryption scheme and we use the SNARGs constructed in the previous section, we can obtain a malleable proof system for any t-tiered $\mathcal{T}^{(pok)}$ with constant t. (On the other hand, we will see in Section 4 that there are interesting relations and transformation classes we can obtain without fully homomorphic encryption as well.) As for size efficiency, we know by the succinctness property of SNARGs that the size of π' will not grow through transformation. For the ciphertext c, if we assume that T_w does not increase the size of the witness, then the size of c will stay the same as well and thus the proof will remain compact even as it is transformed.

We would now like to show that if the SNARG satisfies adaptive knowledge extraction then the Enc+NIZK construction satisfies extractability; i.e., is an argument of knowledge. We also must show that the construction retains the original zero knowledge and derivation privacy properties as well.

Theorem 3.6. *If the SNARG satisfies adaptive knowledge extraction with respect to $R^{(t)}$ then the Enc+NIZK construction is a proof of knowledge with respect to $R^{(pok)}$.*

Proof. To show this, we first define our extractor (E_1, E_2). E_1 will generate $\mathsf{crs}' \xleftarrow{\$} \mathsf{CRSSetup}'(1^k)$ and $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$; it will then output $\mathsf{crs} := (\mathsf{crs}', pk)$ and $\tau_e := sk$, so that its output crs is distributed identically to the output of $\mathsf{CRSSetup}$. When E_2 receives a value π, it will parse $\pi = (\pi', c)$ and output $w := \mathsf{Dec}(sk, c)$.

Now, we show that if there exists an adversary $\mathcal{A}$ that breaks extractability using this extractor (E_1, E_2) with some non-negligible probability ϵ then we can use it to construct an adversary $\mathcal{B}$ that breaks soundness for the SNARG with the same probability ϵ; by Theorem 2.7, which states that adaptive knowledge extraction implies soundness, the result will follow. To start, the adversary $\mathcal{B}$ will get as input a CRS crs'. It can then form $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ and give (crs', pk) and $\tau_e := sk$ to $\mathcal{A}$. At some point, $\mathcal{A}$ will output (x, π); if π parses as $\pi = (\pi', c)$, then $\mathcal{B}$ will output $((pk, x, c), \pi')$.

As $\mathcal{B}$ generates (pk, sk) honestly and its input crs' is assumed to be drawn from $\mathsf{CRSSetup}'$, the CRS given to $\mathcal{A}$ is distributed identically to what $\mathcal{A}$ expects. To see that $\mathcal{B}$ will also succeed whenever $\mathcal{A}$ does, we observe that $\mathcal{A}$ will break extractability only if $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ but $E_2(\mathsf{crs}, \tau_e, \pi) = w$ and $(x, w) \notin R^{(pok)}$. If $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ then we know, by how the verifier is defined, that $\mathcal{V}'(\mathsf{crs}', (pk, x, c), \pi') = 1$ as well, so that the output of $\mathcal{B}$ will pass verification. Similarly, by the definition of the extractor and the perfect decryption of the encryption scheme, we know that c is not an encryption of a witness w such that $(x, w) \in R^{(pok)}$ and therefore $(pk, x, c) \notin L_{R^{(t)}}$. $\mathcal{B}$ will therefore succeed in breaking soundness whenever $\mathcal{A}$ breaks extractability, so $\mathcal{B}$ will succeed with probability ϵ. □

Theorem 3.7. *If the SNARG is zero knowledge and the encryption scheme is IND-CPA secure, then the* Enc+NIZK *construction is zero knowledge.*

Proof. To show this, we first define our simulator (S_1, S_2) based on the underlying simulator (S_1', S_2') for the SNARG. The simulator S_1 will generate $(\mathsf{crs}', \tau_s') \xleftarrow{\$} S_1'(1^k)$ and $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$; it will then output $\mathsf{crs} := (\mathsf{crs}', pk)$ and $\tau_s := \tau_s'$. When S_2 is queried on an instance x, it will form $c \xleftarrow{\$} \mathsf{Enc}(pk, 0)$ and $\pi' \xleftarrow{\$} S_2'(\mathsf{crs}', \tau_s', (pk, x, c))$. It will then return $\pi := (\pi', c)$.

To show that the outputs of this simulator are indistinguishable from those of $(\mathsf{CRSSetup}, \mathcal{P})$, we first describe a hybrid simulator S_2 that is given both the instance x and the witness w such that $(x, w) \in R^{(pok)}$. This simulator will form $c \xleftarrow{\$} \mathsf{Enc}(pk, w)$ and $\pi' \xleftarrow{\$} S_2'(\mathsf{crs}', \tau_s', (pk, x, c))$. If there exists some adversary $\mathcal{A}$ that distinguishes between the outputs of this hybrid simulator and the prover with some non-negligible advantage ϵ, then we show that we can use it to construct an adversary $\mathcal{B}$ that breaks the zero knowledge property of the SNARG with the same advantage ϵ. The behavior of $\mathcal{B}$ is straightforward: when it is given a crs', it generates $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ and gives $\mathsf{crs} := (\mathsf{crs}', pk)$ to $\mathcal{A}$. When $\mathcal{A}$ issues an oracle query (x, w), $\mathcal{B}$ picks randomness $r \xleftarrow{\$} \mathcal{R}$, sets $c := \mathsf{Enc}(pk, w; r)$, and sends $((pk, x, c), (w, r))$ to its own oracle to get back a proof π'; it then returns (π', c) to $\mathcal{A}$. At the end, $\mathcal{B}$ will output the same guess bit as $\mathcal{A}$.

To see that $\mathcal{B}$ will succeed whenever $\mathcal{A}$ does, we first argue that interactions with $\mathcal{B}$ are identical to those that $\mathcal{A}$ expects. For the CRS, we observe that if crs' was generated by $\mathsf{CRSSetup}'$ then this corresponds to the honest case for $\mathcal{A}$, while if crs' was generated by S_1' then this corresponds to the simulated case. For the proofs, if $\mathcal{B}$'s oracle uses the prover $\mathcal{P}'$ to compute π' then the (π', c) returned to $\mathcal{A}$ will be distributed identically to the output of $\mathcal{P}$, while if it uses S_2' then the values given to $\mathcal{A}$ will be distributed identically to those computed by the hybrid simulator. As the winning cases for $\mathcal{A}$ and $\mathcal{B}$ therefore align perfectly, we can see that $\mathcal{B}$ will succeed whenever $\mathcal{A}$ does.

Now, we move on to our real simulator S_2 described at the beginning of the proof, and argue that if there exists some adversary $\mathcal{A}$ that can distinguish between the outputs of this simulator and the hybrid simulator with some non-negligible advantage ϵ then we can use it to construct a $\mathcal{B}$ that breaks the IND-CPA security of the underlying encryption scheme with the same advantage ϵ. Again, the behavior of $\mathcal{B}$ is straightforward: on an input pk, it will generate $(\mathsf{crs}', \tau_s') \xleftarrow{\$} S_1(1^k)$ and give to $\mathcal{A}$ $\mathsf{crs} := (\mathsf{crs}', pk)$. On queries of the form (x, w), $\mathcal{B}$ will query its own left-right oracle on $(w, 0)$ to get back a ciphertext c. It will then compute $\pi' \xleftarrow{\$} S_2'(\mathsf{crs}', \tau_s', (pk, x, c))$ and return (π', c) to $\mathcal{A}$. At the end, $\mathcal{B}$ will output the same guess bit as $\mathcal{A}$.

To see that $\mathcal{B}$ will succeed whenever $\mathcal{A}$ does, we first argue that interactions with $\mathcal{B}$ are identical to those that $\mathcal{A}$ expects. First, we observe that the CRS will be identical to the one that $\mathcal{A}$ expects. For the proofs, if $\mathcal{B}$'s oracle uses the value w to encrypt then the (π', c) returned to $\mathcal{A}$ will be distributed identically to the output of the hybrid simulator, while if it uses 0 then the values given to $\mathcal{A}$ will be distributed identically to those computed by the real simulator. As the winning cases for $\mathcal{A}$ and $\mathcal{B}$ therefore again align perfectly, we can see that $\mathcal{B}$ will succeed whenever $\mathcal{A}$ does. □

Theorem 3.8. *If the SNARG is zero knowledge and strongly derivation private with respect to the class of transformations $\mathcal{T}^{(t)}$ and the encryption scheme is function private with respect to $\mathcal{T}^{(E)}$ then the* Enc+NIZK *construction is derivation private with respect to $\mathcal{T}^{(pok)}$.*

Proof. To show this, we progress through a series of game transformations. The first game, G_0, will be the honest derivation privacy game using $b = 0$, in which $\mathcal{P}$ and Enc are used. In the next game, G_1, we switch to using simulated proofs; we argue that this change will go undetected by zero knowledge. Next, in Game G_2, we switch to using Eval instead of Enc, which we argue will go unnoticed by function privacy. Finally in Game G_3 we switch to using ZKEval, which we argue will go unnoticed by strong derivation privacy; note that here we are now using ZKEval and Eval and so are in the derivation privacy game for $b = 1$. If each game is indistinguishable from the previous one, then in particular G_0 will be indistinguishable from G_3 and so we will be done.

To show that G_0 is indistinguishable from G_1, we argue that if there exists an adversary $\mathcal{A}$ that distinguishes between the two games with some non-negligible advantage ϵ then we can use it to construct an adversary $\mathcal{B}$ that breaks the zero knowledge of the SNARG with the same advantage ϵ. To start, $\mathcal{B}$ will get as input a CRS crs'. It will then generate $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ and give $\mathsf{crs} := (\mathsf{crs}', pk)$ to $\mathcal{A}$. When $\mathcal{A}$ returns its challenge query (x_A, w_A, π_A, T), $\mathcal{B}$ will parse $\pi_A = (\pi, c)$ and $T = (T_x, T_w)$, pick randomness $r' \xleftarrow{\$} \mathcal{R}$, and form $c' := \mathsf{Enc}(pk, T_w(w_A); r')$. It will then query $((pk, T_x(x_A), c'), (T_w(w_A), r'))$ to its own oracle to get back a proof π' and return (π', c') to $\mathcal{A}$. If $\mathcal{A}$ guesses that it is in G_0 then $\mathcal{B}$ guesses it is interacting with the prover, and if $\mathcal{A}$ guesses that it is in G_1 then $\mathcal{B}$ guesses it is interacting with the simulator. As $\mathcal{B}$ is executing the code of either game exactly, the success cases for $\mathcal{A}$ and $\mathcal{B}$ therefore line up perfectly and $\mathcal{B}$ will succeed with the same advantage as $\mathcal{A}$.

To next show that G_1 is indistinguishable from G_2, we argue that if there exists an adversary $\mathcal{A}$ that distinguishes between the two games with non-negligible advantage ϵ then we can use it to construct an adversary $\mathcal{B}$ that breaks function privacy with the same advantage ϵ. To start, $\mathcal{B}$ will get as input a keypair (pk, sk); it will then generate $(\mathsf{crs}', \tau_s) \xleftarrow{\$} S_1'(1^k)$ and give $\mathsf{crs} := (\mathsf{crs}', pk)$ to $\mathcal{A}$. When $\mathcal{A}$ returns its challenge query (x_A, w_A, π_A, T), $\mathcal{B}$ will parse $\pi_A = (\pi, c)$ and $T = (T_x, T_w)$. It will then query $(c, (T_c, T_w))$ to its own oracle to get back a ciphertext c'. Finally, it will compute $\pi' \xleftarrow{\$} S_2(\mathsf{crs}', \tau_s, (pk, T_x(x_A), c'))$ and return (π', c') to $\mathcal{A}$. If $\mathcal{A}$ guesses it is in G_1 then $\mathcal{B}$ will guess that $b = 0$ (i.e., c' was formed using Enc), and if $\mathcal{A}$ guesses it is in G_2 then $\mathcal{B}$ will guess that $b = 1$ (i.e., c' was formed using Eval). Again, if $b = 0$ then $\mathcal{B}$ is executing the exact code of G_1, while if $b = 1$ then $\mathcal{B}$ is executing the exact code of G_2; $\mathcal{B}$'s guesses will therefore be right exactly when $\mathcal{A}$'s are, so $\mathcal{B}$ will succeed with the same advantage as $\mathcal{A}$.

Finally, we can argue that G_2 is indistinguishable from G_3; to do this, we show that if there exists an adversary $\mathcal{A}$ that distinguishes between the two games with non-negligible advantage ϵ then we can use it to construct an adversary $\mathcal{B}$ that breaks strong derivation privacy of the SNARG with the same advantage ϵ. To start, $\mathcal{B}$ will get as input a CRS crs' and a simulation trapdoor τ_s; it can then generate $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ and give $\mathsf{crs} := (\mathsf{crs}', pk)$ to $\mathcal{A}$. When $\mathcal{A}$ returns its challenge query (x_A, w_A, π_A, T), $\mathcal{B}$ can pick randomness $r' \xleftarrow{\$} \mathcal{R}$, derive $T' := \tau(T, r')$ from T, and compute $c' := \mathsf{Eval}(pk, (T_c, T_w), c; r'))$. It then outputs $((pk, x, c), \pi, T')$ as its own challenge query to get back a proof π' and returns (π', c') to $\mathcal{A}$. $\mathcal{B}$ will then guess $b = 0$ if $\mathcal{A}$ guesses it is in G_2, and $b = 1$ if $\mathcal{A}$ guesses it is in G_3. To see that interactions with $\mathcal{B}$ will be indistinguishable to those that $\mathcal{A}$ expects, observe that if $b = 0$ and $\mathcal{B}$'s oracle is returning proofs from S_2, then the values given to $\mathcal{A}$ will be identical to those that it expects in G_2; similarly, if $\mathcal{B}$'s oracle is instead using ZKEval, then the values given to $\mathcal{A}$ will be identical to those that it expects in G_3. $\mathcal{B}$ therefore succeeds whenever $\mathcal{A}$ does, and thus succeeds with advantage ϵ. □

3.3 From malleable NIWIPoKs to cm-NIZKs

With our malleable NIZKPoK in place, we are finally ready to construct cm-NIZKs (although, as we will see, we require only witness indistinguishability rather than full zero knowledge). We first recall the construction of CKLM, who used a relation R' such that $((x, vk), (w, x', T, \sigma)) \in R'$ if $(x, w) \in R$ or $\mathsf{Verify}(vk, \sigma, x') = 1$, $x = T_x(x')$, and $T \in \mathcal{T}$, where σ was a signature for a secure signature scheme. The reason for using the signature scheme was their simulation strategy: on queries x, their simulator could use as a trapdoor the signing key sk to create witnesses of the form $(\bot, x, \mathrm{id}, \sigma)$ on queries x.

Unlike CKLM, we want to consider classes of transformations that may not contain the identity (for example, the t-tiered transformation classes). If we want to guarantee that the adversary cannot always apply the identity transformation, we need to modify the construction.

Suppose we want to construct a cm-NIZK for relation $R^{(cm)}$ and transformation class $\mathcal{T}^{(cm)}$. We will use a NIWIPoK for an augmented relation $R^{(pok)}$ such that $((x, vk, vk_{\mathsf{ot}}), (w, x', vk'_{\mathsf{ot}}, T, \sigma)) \in R^{(pok)}$ if (1) $(x, w) \in R^{(cm)}$ or (2) $\mathsf{Verify}(vk, \sigma, (x', vk'_{\mathsf{ot}})) = 1$ and either $x = T_x(x')$ for $T = (T_x, T_w) \in \mathcal{T}^{(cm)}$, or $x' = x$ and $vk'_{\mathsf{ot}} = vk_{\mathsf{ot}}$, where vk_{ot} is a verification key for a secure one-time signature scheme.

Intuitively, our simulator can now use this last type of witness; i.e., on a query x, it can use sk as a trapdoor to sign (x, vk_{ot}) and produce a signature σ, and then form a proof using $(\bot, x, vk_{\mathsf{ot}}, \bot, \sigma)$ as a witness. To ensure that an adversary cannot simply reuse this proof and claim it as its own (i.e., apply the identity transformation), proofs are accompanied by a one-time signature, on both the instance and the proof, to indicate that the proof was formed fresh for this instance. Because the one-time signature thus binds together the instance and the proof, we call this construction "signature binding."

Now, if we want to allow transformations $(\widehat{T}_x, \widehat{T}_w) \in \mathcal{T}^{(cm)}$ for our cm-NIZK, we will have to be able to transform the underlying NIWIPoK accordingly. To do this for any $\widehat{T} = (\widehat{T}_x, \widehat{T}_w) \in \mathcal{T}^{(cm)}$, and any $\widehat{vk}_{\mathsf{ot}} \in VK_{\mathsf{ot}}$ (where VK_{ot} is the set of all possible verification keys), let $\rho(\widehat{T}, \widehat{vk}_{\mathsf{ot}})$ be a transformation that maps $(x, vk, vk_{\mathsf{ot}})$ to $(\widehat{T}_x(x), vk, \widehat{vk}_{\mathsf{ot}})$ and $(w, x', vk'_{\mathsf{ot}}, T, \sigma)$ to $(\widehat{T}_w(w), x', vk'_{\mathsf{ot}}, \widehat{T} \circ T, \sigma)$. We require the underlying NIWIPoK to be malleable with respect to this class $\mathcal{T}^{(pok)}$.

More formally, let $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be an unforgeable signature scheme, let $(\mathsf{KeyGen}_{\mathsf{ot}}, \mathsf{Sign}_{\mathsf{ot}}, \mathsf{Verify}_{\mathsf{ot}})$ be a strongly unforgeable one-time signature scheme, and let $(\mathsf{CRSSetup}_{\mathsf{WI}}, \mathcal{P}_{\mathsf{WI}}, \mathcal{V}_{\mathsf{WI}})$ be a malleable derivation-private NIWIPoK for $R^{(pok)}$. We give our construction of a cm-NIZK using these primitives in Figure 4.

Although in using $\widehat{T} \circ T$ we require that $\mathcal{T}^{(cm)}$ be closed under composition, we note that this is not a strong restriction. Indeed, if $\mathcal{T}^{(cm)}$ is not closed under composition, then we can define the closure of $\mathcal{T}^{(cm)}$ to be the class of transformations $\mathcal{T}^{(cm)'}$ such that $T \in \mathcal{T}^{(cm)'}$ if and only if $T = T_1 \circ \ldots \circ T_j$ for $j < t$ and $T_1, \ldots, T_j \in \mathcal{T}^{(cm)}$. In this case, if we construct the NIWIPoK using our $\mathsf{Enc}{+}\mathsf{NIZK}$ construction, our proofs have to increase in size by a factor of t. (The encryption scheme used will have to have message space large enough to represent $T_1 \circ \ldots \circ T_t$ as $(T_1, \ldots, T_t)$.) On the other hand, this size increase is unavoidable for general transformations if we want to obtain a definition (like CM-SSE) in which a non-interactive black-box extractor must be able to extract the entire transformation

- $\mathsf{CRSSetup}(1^k)$: Generate $\mathsf{crs}_{\mathsf{WI}} \xleftarrow{\$} \mathsf{CRSSetup}_{\mathsf{WI}}(1^k)$ and $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$. Output $\mathsf{crs} := (\mathsf{crs}_{\mathsf{WI}}, vk)$.

- $\mathcal{P}(\mathsf{crs}, x, w)$: Generate $(vk_{\mathsf{ot}}, sk_{\mathsf{ot}}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{ot}}(1^k)$ and parse $\mathsf{crs} = (\mathsf{crs}_{\mathsf{WI}}, vk)$. Compute $\pi' \xleftarrow{\$} \mathcal{P}_{\mathsf{WI}}(\mathsf{crs}_{\mathsf{WI}}, (x, vk, vk_{\mathsf{ot}}), (w, \bot, \bot, \bot, \bot))$ and $\sigma_{\mathsf{ot}} \xleftarrow{\$} \mathsf{Sign}_{\mathsf{ot}}(sk_{\mathsf{ot}}, (x, \pi'))$, and return $\pi := (\pi', \sigma_{\mathsf{ot}}, vk_{\mathsf{ot}})$.

- $\mathcal{V}(\mathsf{crs}, x, \pi)$: Parse $\mathsf{crs} = (\mathsf{crs}_{\mathsf{WI}}, vk)$ and $\pi = (\pi', \sigma_{\mathsf{ot}}, vk_{\mathsf{ot}})$. Check that $\mathsf{Verify}_{\mathsf{ot}}(vk_{\mathsf{ot}}, \sigma_{\mathsf{ot}}, (x, \pi')) = 1$; if this fails then output 0. Otherwise, return $\mathcal{V}_{\mathsf{WI}}(\mathsf{crs}_{\mathsf{WI}}, (x, vk, vk_{\mathsf{ot}}), \pi')$

- $\mathsf{ZKEval}(crs, T, x, \pi)$: Parse $\mathsf{crs} = (\mathsf{crs}_{\mathsf{WI}}, vk)$; then generate $(vk_{\mathsf{ot}}, sk_{\mathsf{ot}}), (\widehat{vk}_{\mathsf{ot}}, \widehat{sk}_{\mathsf{ot}}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{ot}}(1^k)$, and compute $\pi' \xleftarrow{\$} \mathsf{ZKEval}_{\mathsf{WI}}(\mathsf{crs}_{\mathsf{WI}}, \rho(T, \widehat{vk}_{\mathsf{ot}}), (x, vk, vk_{\mathsf{ot}}), \pi)$. Compute $\sigma_{\mathsf{ot}} \xleftarrow{\$} \mathsf{Sign}_{\mathsf{ot}}(\widehat{sk}_{\mathsf{ot}}, (x, \pi'))$, and return $\pi := (\pi', \sigma_{\mathsf{ot}}, \widehat{vk}_{\mathsf{ot}})$.

Figure 4: Our signature-binding construction of a cm-NIZK.

performed.

By construction, we directly obtain the following theorems:

Theorem 3.9. *If the NIWIPoK is complete for relation $R^{(pok)}$, and the one-time signature is correct, then the above signature-binding construction is complete for relation $R^{(cm)}$.*

Theorem 3.10. *If the NIWIPoK is malleable with respect to transformation class $\mathcal{T}^{(pok)} = \rho(\mathcal{T}^{(cm)}, VK_{\mathsf{ot}})$ (as defined above), then the above signature-binding construction is malleable for transformation class $\mathcal{T}^{(cm)}$.*

Now, if we want to instantiate the NIWIPoK using our Enc+NIZK construction from the previous section, we must first ensure that $R^{(pok)}$ and $\mathcal{T}^{(pok)}$ satisfy the constraints discussed therein. In particular, we required that $\mathcal{T}^{(pok)}$ be a t-tiered transformation class for $R^{(pok)}$, and that there is an encryption scheme whose message space contains the witness space for $R^{(pok)}$ that is homomorphic with respect to the class of transformations $\{T_w\}$ for all $(T_x, T_w) \in \mathcal{T}^{(pok)}$.

Expanding on this last requirement, as our witnesses for $R^{(pok)}$ are of the form $(w, x', vk'_{\mathsf{ot}}, T, \sigma)$, we need to use an encryption scheme in which the message space subsumes the space of all of these values; i.e., the witness, instance, and transformation spaces, as well as the space of possible one-time verification keys and signatures. We also need the encryption scheme to be homomorphic with respect to the set of transformations that map $(w, x', vk'_{\mathsf{ot}}, T, \sigma)$ to $(\widehat{T}_w(w), x', vk'_{\mathsf{ot}}, \widehat{T} \circ T, \sigma)$ for any $(\widehat{T}_x, \widehat{T}_w) \in \mathcal{T}^{(cm)}$. Finally, we require that $\mathcal{T}^{(cm)}$ is t-tiered for $R^{(cm)}$, as this will guarantee that $\mathcal{T}^{(pok)}$ is t-tiered for $R^{(pok)}$. If we assume SNARGs for general languages and fully homomorphic encryption, then we can obtain a cm-NIZK for any t-tiered transformation class as long as t is constant; in Section 4, we will also see that we can construct cm-NIZKs for interesting relations using only multiplicatively homomorphic encryption. Moreover, if we continue our assumption from the previous section that $\widehat{T}_w$ does not increase the size of w, then the size of proofs will not grow by transformation here either.

Finally, in order to show that this is a cm-NIZK, we need to show that it satisfies zero knowledge, CM-SSE, and strong derivation privacy.

Theorem 3.11. *If the proof system $(\mathsf{CRSSetup}_{\mathsf{WI}}, \mathcal{P}_{\mathsf{WI}}, \mathcal{V}_{\mathsf{WI}})$ is witness indistinguishable then the signature-binding construction is zero knowledge.*

Proof. To show this, we first define our simulator (S_1, S_2). The simulator S_1 is simple: it will honestly generate $\mathsf{crs}_{\mathsf{WI}} \xleftarrow{\$} \mathsf{CRSSetup}_{\mathsf{WI}}(1^k)$ and $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$; it will then output $\mathsf{crs} := (\mathsf{crs}_{\mathsf{WI}}, vk)$ and $\tau_s := sk$. As for S_2, when it is asked to provide a proof for an instance x, it will compute $(vk_{\mathsf{ot}}, sk_{\mathsf{ot}}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{ot}}(1^k)$ and then use the signing key sk to compute $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, (x, vk_{\mathsf{ot}}))$. It will then compute the proof $\pi' \xleftarrow{\$} \mathcal{P}_{\mathsf{WI}}(\mathsf{crs}_{\mathsf{WI}}, (x, vk, vk_{\mathsf{ot}}), (\bot, x, vk_{\mathsf{ot}}, \bot, \sigma))$, form the one-time signature $\sigma_{\mathsf{ot}} \xleftarrow{\$} \mathsf{Sign}(sk_{\mathsf{ot}}, (x, \pi'))$, and return $(\pi', \sigma_{\mathsf{ot}}, vk_{\mathsf{ot}})$.

Using this simulator, we can show that if there exists an adversary $\mathcal{A}$ that distinguishes between the outputs of S_1 and S_2 and the outputs of $\mathsf{CRSSetup}$ and $\mathcal{P}$ with some non-negligible advantage ϵ then we can construct an adversary $\mathcal{B}$ that distinguishes between witnesses in the underlying proof system with the same advantage.

The adversary $\mathcal{B}$ will now begin by getting as input a CRS crs'. It will then generate $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ and give $\mathsf{crs} := (\mathsf{crs}', vk)$ to $\mathcal{A}$. When $\mathcal{A}$ outputs a query (x, w), $\mathcal{B}$ will first generate $(vk_{\mathsf{ot}}, sk_{\mathsf{ot}}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{ot}}(1^k)$ and then form $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, (x, vk_{\mathsf{ot}}))$. It will then set $x' := (x, vk, vk_{\mathsf{ot}})$, $w_0 := (w, \bot, \bot, \bot, \bot)$, and $w_1 := (\bot, x, vk_{\mathsf{ot}}, \bot, \sigma)$, and output (x', w_0, w_1) as its own query to get back a proof π'. $\mathcal{B}$ can then form $\sigma_{\mathsf{ot}} \xleftarrow{\$} \mathsf{Sign}(sk_{\mathsf{ot}}, (x, \pi'))$ and return $(\pi', \sigma_{\mathsf{ot}}, vk_{\mathsf{ot}})$ to $\mathcal{A}$. At the end of the game, $\mathcal{B}$ outputs the same guess bit as $\mathcal{A}$.

As the underlying CRS and signing keypair were generated honestly, the crs returned by S_1 will be distributed identically to an honest one, and so the CRS that $\mathcal{B}$ gives to $\mathcal{A}$ will be distributed identically to both the output of S_1 and the output of $\mathsf{CRSSetup}$. As for the proofs, we note that σ_{ot} and vk_{ot} are always formed the same way, so the only potential difference is in the underlying proof π'. If the left witness is used (i.e., $b = 0$ for the WI game) then $\mathcal{A}$ gets a proof of the form $\pi' \xleftarrow{\$} \mathcal{P}_{\mathsf{WI}}(\mathsf{crs}', (x, vk, vk_{\mathsf{ot}}), (w, \bot, \bot, \bot, \bot))$, which is exactly what $\mathcal{A}$ would get when interacting with the prover. If instead the right witness is used then $\mathcal{A}$ gets a proof of the form $\pi' \xleftarrow{\$} \mathcal{P}_{\mathsf{WI}}(\mathsf{crs}', (x, vk, vk_{\mathsf{ot}}), (\bot, x, vk_{\mathsf{ot}}, \bot, \sigma))$ which, looking back to the description of S_2, is exactly what $\mathcal{A}$ would get when interacting with the simulator. As the interactions with $\mathcal{B}$ are therefore identical to the interactions that $\mathcal{A}$ expects and $\mathcal{B}$ will guess correctly whenever $\mathcal{A}$ does, we can conclude that $\mathcal{B}$ will succeed with advantage ϵ as well. □

Theorem 3.12. *If the signature scheme* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is unforgeable (i.e., EUF-CMA secure), the one-time signature* $(\mathsf{KeyGen}_{\mathsf{ot}}, \mathsf{Sign}_{\mathsf{ot}}, \mathsf{Verify}_{\mathsf{ot}})$ *is strongly unforgeable (i.e., SUF-CMA secure), and the proof* $(\mathsf{CRSSetup}_{\mathsf{WI}}, \mathcal{P}_{\mathsf{WI}}, \mathcal{V}_{\mathsf{WI}})$ *is an argument of knowledge, the signature-binding construction satisfies the CM-SSE property.*

Proof. To show this, we can take an adversary $\mathcal{A}$ that breaks CM-SSE with some non-negligible probability ϵ and use it to construct an adversary $\mathcal{B}$ that can break either the unforgeability of the signature scheme, the security of the one-time signature scheme, or the extractability of the proof (at some level i) with probability $\epsilon/3q_s$, where q_s is the number of queries made to S_2. To start, $\mathcal{B}$ will get as input public signing keys vk and vk^*_{ot}, and (crs', τ'_e). It will then pick at random which scheme it wants to try to break: it can choose (1) unforgeability, in a strategy we call $\mathcal{B}_1$, (2) one-time unforgeability, in a strategy we call $\mathcal{B}_2$, or (3) extractability, in a strategy we call $\mathcal{B}_3$.

The path $\mathcal{B}_1$ will give to $\mathcal{A}$ $(\mathsf{crs} := (\mathsf{crs}', vk), \tau'_e)$. On S_2 oracle queries, $\mathcal{B}_1$ will generate $(vk_{\mathsf{ot}}, sk_{\mathsf{ot}}) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ and give the query (x, vk_{ot}) to its own signing oracle to get back a value σ. It will then compute $\pi' \xleftarrow{\$} \mathcal{P}_{\mathsf{WI}}(\mathsf{crs}', (x, vk, vk_{\mathsf{ot}}), (\bot, x, vk_{\mathsf{ot}}, \bot, \sigma))$ and $\sigma_{\mathsf{ot}} \xleftarrow{\$} \mathsf{Sign}(sk_{\mathsf{ot}}, (x, \pi'))$ and return $(\pi', \sigma_{\mathsf{ot}}, vk_{\mathsf{ot}})$ to $\mathcal{A}$.

The path $\mathcal{B}_2$, on the other hand, will first generate on its own $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ and give to $\mathcal{A}$ $(\mathsf{crs} := (\mathsf{crs}', vk), \tau'_e)$. It will then pick at random which S_2 query it thinks $\mathcal{A}$ might use to form its proof.

For this particular query, $\mathcal{B}_2$ will form $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, (x, vk^*_{\mathsf{ot}}))$ and $\pi' \xleftarrow{\$} \mathcal{P}_{\mathsf{WI}}(\mathsf{crs}', (x, vk, vk^*_{\mathsf{ot}}), (\bot, x, vk^*_{\mathsf{ot}}, \bot, \sigma))$. $\mathcal{B}_2$ will then query its oracle on (x, π') to get back a signature σ^*_{ot} and return to $\mathcal{A}$ $(\pi', \sigma^*_{\mathsf{ot}}, vk^*_{\mathsf{ot}})$. On all other queries, $\mathcal{B}_2$ will instead generate fresh $(vk_{\mathsf{ot}}, sk_{\mathsf{ot}}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{ot}}(1^k)$ and form the one-time signature itself (and otherwise keep everything the same).

Finally, the path $\mathcal{B}_3$ will generate $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ and give to $\mathcal{A}$ $((\mathsf{crs}', vk), \tau'_e)$. It can now behave exactly as S_2 in responding to oracle queries.

With all of the paths, $\mathcal{A}$ will eventually output a pair (x, π); if $\mathcal{V}(\mathsf{crs}, x, \pi) = 0$ or $(x, \pi) \in Q$ then $\mathcal{A}$ is not behaving properly and they can simply abort. Otherwise, all three paths can parse $\pi = (\pi', \sigma_{\mathsf{ot}}, vk_{\mathsf{ot}})$. If $vk_{\mathsf{ot}} = vk^*_{\mathsf{ot}}$ then $\mathcal{B}_2$ can output $((x, \pi'), \sigma_{\mathsf{ot}})$ to win at its game. To see why this must be a forgery, we consider the possibilities. If x was not queried to $\mathcal{B}_2$ then (x, π') is clearly different from the value $\mathcal{B}_2$ queried to its oracle. If instead x was queried, then either π' is the same one as $\mathcal{B}_2$ used or it is different; in the latter case we trivially see that (x, π') is different from the query used by $\mathcal{B}_2$. If we therefore have that x and π' are both the same, then we turn to the constraint that $(x, \pi) \notin Q$. As $\pi = (\pi', \sigma_{\mathsf{ot}}, vk_{\mathsf{ot}})$ and we are assuming that π' and vk_{ot} are the same ones used by $\mathcal{B}_2$, it must be the case that $\sigma_{\mathsf{ot}} \neq \sigma^*_{\mathsf{ot}}$ and so, because we are using strong unforgeability, $((x, \pi'), \sigma_{\mathsf{ot}})$ will still suffice to win the game.

The other two paths can continue by acting as E'_2 to extract a tuple $(w, x', vk'_{\mathsf{ot}}, T, \sigma)$ from π'. If $\mathsf{Verify}(vk, \sigma, (x', vk'_{\mathsf{ot}})) = 1$ but x' was not queried to S_2 or vk'_{ot} was not the value chosen when x' was queried, then $\mathcal{B}_1$ can output $((x', vk'_{\mathsf{ot}}), \sigma)$ to win the unforgeability game. If instead either $w \neq \bot$ but $(x, w) \notin R$, $(x', vk'_{\mathsf{ot}}, T, \sigma) \neq (\bot, \bot, \bot, \bot)$ but $\mathsf{Verify}(vk, \sigma, (x', vk'_{\mathsf{ot}})) = 0$ and either $x \neq T_x(x')$, $T \notin \mathcal{T}$ or $x \neq x'$, $vk \neq vk_{\mathsf{ot}}$, or $(w, x', vk'_{\mathsf{ot}}, T, \sigma) = (\bot, \bot, \bot, \bot, \bot)$ then $\mathcal{B}_3$ can output (x, π) to win at its game.

As each path gives honestly formed inputs to $\mathcal{A}$ and behaves honestly on both simulation and extraction queries, interactions along each path will be distributed identically to what $\mathcal{A}$ expects. Furthermore, as argued above, each of the winning conditions for $\mathcal{A}$ allows some path taken by $\mathcal{B}$ to win at its respective game, so $\mathcal{B}$ should succeed every time $\mathcal{A}$ succeeds and it guesses the correct path; in the case of $\mathcal{B}_2$, however, $\mathcal{B}$ also needs to guess the correct S_2 query. If we say that $\mathcal{A}$ makes at most q_s queries to S_2, then overall $\mathcal{B}$ should succeed with probability at least $\epsilon/3q_s$. □

Theorem 3.13. *If the proof system* $(\mathsf{CRSSetup}_{\mathsf{WI}}, \mathcal{P}_{\mathsf{WI}}, \mathcal{V}_{\mathsf{WI}})$ *is derivation private for* $\mathcal{T}^{(pok)}$ *then the signature-binding construction is strongly derivation private for* $\mathcal{T}^{(cm)}$.

Proof. To show this, we take an adversary $\mathcal{A}$ that breaks strong derivation privacy for our construction with some non-negligible advantage ϵ and use it to construct an adversary $\mathcal{B}$ that breaks derivation privacy for the underlying proof with the same advantage. To start, $\mathcal{B}$ will receive as input some $\mathsf{crs}_{\mathsf{WI}}$. It then generates $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ and gives $\mathsf{crs} := (\mathsf{crs}_{\mathsf{WI}}, vk)$ to $\mathcal{A}$; it also keeps sk for itself. On S_2 queries, $\mathcal{B}$ will use the signing key to behave exactly as S_2 would. On $\mathcal{A}$'s challenge query (x_A, π_A, T_A), $\mathcal{B}$ can generate $(vk_{\mathsf{ot}}, sk_{\mathsf{ot}}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{ot}}(1^k)$ and form $w := (\bot, x_A, vk_{\mathsf{ot}}, \bot, \mathsf{Sign}(sk, (x_A, vk_{\mathsf{ot}})))$; i.e., the same kind of witness that it uses for simulation. $\mathcal{B}$ then generates a new pair $(\widehat{vk}_{\mathsf{ot}}, \widehat{sk}_{\mathsf{ot}}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{ot}}(1^k)$ and queries its own oracle on $((x_A, vk, vk_{\mathsf{ot}}), w, \pi_A, \rho(T_A, \widehat{vk}_{\mathsf{ot}}))$ to receive a proof π'. It then gives to $\mathcal{A}$ $\pi := (\pi', \mathsf{Sign}(vk_{\mathsf{ot}}, (T_x(x), \pi')), vk_{\mathsf{ot}})$. At the end of the game, $\mathcal{B}$ will output the same guess bit as $\mathcal{A}$.

To see that interactions with $\mathcal{B}$ are indistinguishable from the honest interactions that $\mathcal{A}$ expects, we first note that $\mathcal{B}$ behaves completely honestly as S_1 and S_2, so we need only focus on the challenge query and its response. As $\mathcal{B}$ is here computing the same valid witnesses for the proof system that S_2 uses, there are two options. If its response π is coming from $\mathcal{P}_{\mathsf{WI}}$, then the proof it returns to $\mathcal{A}$ is distributed identically to a proof from S_2. If the query is instead answered by $\mathsf{ZKEval}_{\mathsf{WI}}$, then the proof it returns is trivially distributed identically to a proof from ZKEval, as ZKEval is defined using $\mathsf{ZKEval}_{\mathsf{WI}}$. As $\mathcal{B}$ outputs the same guess bit as $\mathcal{A}$, it will therefore succeeds whenever $\mathcal{A}$ does. □

4 A Compactly Verifiable Shuffle Using SNARGs

Now that we have just constructed our SNARG-based cm-NIZK, we consider how to use it to construct a compactly verifiable shuffle, as defined in Definition 2.4.

4.1 Relations and transformations for shuffles

We start by defining formally the relation and transformations we want to use for shuffles. Abstractly, instances for the correctness of a shuffle are of the form $x = (pk, \{c_i\}, \{c'_i\})$, where pk is a public key for a re-randomizable encryption scheme, $\{c_i\}$ are the original ciphertexts, and $\{c'_i\}$ are the shuffled ciphertexts. Similarly, witnesses are of the form $w = (\varphi, \{R_i\})$, where φ is a permutation and $\{R_i\}$ are the re-randomization factors. The relation R is such that

$$((pk, \{c_i\}, \{c'_i\}), (\varphi, \{R_i\})) \in R \iff \{c'_i\} = \{\mathsf{ReRand}(pk, \varphi(c_i); R_i)\}. \quad (1)$$

Valid transformations $\mathcal{T}$ should be shuffles, which means they have the same form as witnesses. We define transformations on instances as $T_x(x) = T_{(\varphi', \{R'_i\})}(pk, \{c_i\}, \{c'_i\}) := (pk, \{c_i\}, \{\mathsf{ReRand}(pk, \varphi'(c_i); R'_i)\})$ and on witnesses as $T_w(w) = T_{(\varphi', \{R'_i\})}(\varphi, \{R_i\}) := (\varphi' \circ \varphi, \{\varphi'(R_i) * R'_i\})$, where $*$ is the operation used to compose the randomness (i.e., $\mathsf{ReRand}(pk, \mathsf{ReRand}(pk, c; R), R') = \mathsf{ReRand}(pk, c; R * R')$).

4.2 Our construction

Recall from Appendix B that the shuffle construction of CKLM [7] used four building blocks: a hard relation R_{pk}, a re-randomizable encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReRand})$, a proof of knowledge $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$, and a cm-NIZK $(\mathsf{CRSSetup}', \mathcal{P}', \mathcal{V}')$. We extend the relation from Equation 1 to be

$$((pk, \{c_i\}, \{c'_i\}, \{pk_j\}), (\varphi, \{R_i\}, \{sk_j\})) \in R \Leftrightarrow \{c'_i\} = \{\mathsf{ReRand}(pk, \varphi(c_i); R_i)\} \wedge (pk_j, sk_j) \in R_{pk} \ \forall j.$$

As we just constructed a cm-NIZK, we can simply plug it into this generic construction, which CKLM already proved secure. What it remains to show is that the requirements placed on transformations in Sections 3.2 and Section 3.3 are met by the shuffle transformations.

Recall the general requirement for transformations from Section 3.3: because we must encrypt values of the form $(w, x', vk'_{\mathsf{ot}}, T, \sigma)$, we need an encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ that is homomorphic with respect to the set of transformations that map $(w, x', vk'_{\mathsf{ot}}, T, \sigma)$ to $(\widehat{T}_w(w), x', vk'_{\mathsf{ot}}, \widehat{T} \circ T, \sigma)$ for any $(\widehat{T}_x, \widehat{T}_w) \in \mathcal{T}^{(cm)}$.

In order to meet this requirement for shuffles, we must therefore consider how to encrypt and appropriately transform all of these values. For all of the values except w and T, however, they are unchanged by the transformation; our only requirement here is therefore that they can be encrypted, meaning the spaces they live in are subsumed by the message space. As for the values that do get transformed, w and T, as they are defined for the shuffle we must consider how to transform the permutation φ, the re-randomization values $\{R_i\}$, and the secret keys $\{sk_j\}$. We deal with each of these in turn.

To encrypt a permutation $\varphi \in S_n$, we essentially represent it as its component-wise action on indices. Formally, we first consider the collection $(c_1, \dots, c_n)$ in which $c_i \xleftarrow{\$} \mathsf{Enc}(pk, i)$ for all i; i.e., the collection of ciphertexts encrypting their own index within the set. Now, to represent φ, we compute $c_i^{(\varphi)} \xleftarrow{\$} \mathsf{Enc}(pk, \varphi(i))$ for all i, $1 \leq i \leq n$; the set $\{c_i^{(\varphi)}\}$ is then equal to $\varphi(\{c_i\})$. When we need to compose this φ with a new permutation φ' (e.g., to compute $T_w(w)$), we can compute $\{c_i^{(\varphi' \circ \varphi)}\} = \varphi'(\{c_i^{(\varphi)}\}) = \varphi'(\varphi(\{c_i\}))$, which does represent the composed permutation $\varphi' \circ \varphi$ as desired.

Moving on to the re-randomization values $\{R_i\}$, we start in the same vein as with the permutations: for all i, we compute $c_i^{(r)} \xleftarrow{\$} \mathsf{Enc}(pk, R_i)$. We now place our only requirement on the encryption scheme

$(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$, which is that it must be homomorphic with respect to the $*$ operation (i.e., the operation used to compose randomness); namely that there exist a corresponding operation $\circledast$ on ciphertexts such that if c_1 is an encryption of m_1 and c_2 is an encryption of m_2 then $c_1 \circledast c_2$ is an encryption of $m_1 * m_2$. With such an operation in place, when we want to permute using φ and add in new randomness $\{R_i'\}$, we can compute $c_i^{(r*r')} := \varphi(c_i^{(r)}) \circledast \mathsf{Enc}(pk, R_i')$. By the homomorphic properties of $\circledast$, $c_i^{(r*r')}$ will then be an encryption of $\varphi(R_i) * R_i'$.

Finally, for the keys, we note that as long as all values of sk_j lie in the message space then we are fine, as these values are simply appended to a list and thus do not need to be transformed.

As for the size of the resulting shuffle, we know that the CRS for the t-tiered NIWI construction in Section 3.1 consisted of t common reference strings for the underlying SNARG, each of size $O(n^2)$; as t is constant, the total size of the CRS is $O(n^2)$. At the next level, in the Enc+NIZK construction, we add a public key pk, and at the next level, in the signature-binding construction, we add a verification key vk. If the size of each of these values is constant with respect to n (or even of size $O(n^2)$, although this seems unlikely), then we obtain an overall shuffle parameter size of $O(n^2)$. For the proofs, we know from our discussion in Section 3 that their size will depend on the representation of the witnesses w, instances x, and transformations T. As we've defined things here, the representation of φ and $\{R_i\}$ require n ciphertexts each, which means the representations of w and T are $O(n + \ell)$, as they each also contain ℓ secret keys. Similarly, the size of the instance x is $O(n + \ell)$, as it contains two sets of n ciphertexts and a set of ℓ public keys. The overall size of the proof is therefore $O(n + \ell)$.

Acknowledgments

Anna Lysyanskaya was supported by NSF grants 1012060, 0964379, 0831293, and by a Sloan Foundation fellowship, and Sarah Meiklejohn was supported in part by a MURI grant administered by the Air Force Office of Scientific Research and in part by a graduate fellowship from the Charles Lee Powell Foundation.

References

[1] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Delegatable anonymous credentials. In *Proceedings of Crypto 2009*, volume 5677 of *LNCS*, pages 108–125. Springer-Verlag, 2009.

[2] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *Proceedings of Crypto 2004*, pages 273–289, 2004.

[3] N. Bitanksy, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactve arguments of knowledge, and back again. In *Proceedings of ITCS 2012*, 2012.

[4] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. Recursive composition and bootstrapping for SNARKs and proof-carrying data. Cryptology ePrint Archive, Report 2012/095, 2012. `http://eprint.iacr.org/2012/095`.

[5] M. Blum, A. de Santis, S. Micali, and G. Persiano. Non-interactive zero-knowledge. *SIAM Journal of Computing*, 20(6):1084–1118, 1991.

[6] D. Boneh, G. Segev, and B. Waters. Targeted malleability: homomorphic encryption for restricted computations. In *Proceedings of ITCS 2012*, 2012.

[7] M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn. Malleable proof systems and applications. In *Proceedings of Eurocrypt 2012*, pages 281–300, 2012.

[8] I. Damgård. On sigma protocols. `http://www.daimi.au.dk/~ivan/Sigma.pdf`.

[9] I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *Proceedings of Crypto 1991*, pages 445–456, 1991.

[10] I. Damgård, S. Faust, and C. Hazay. Secure two-party computation with low communication. In *Proceedings of TCC 2012*, volume 7194 of *LNCS*, pages 54–74, 2012.

[11] A. de Santis, G. di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *Proceedings of Crypto 2001*, volume 2139 of *LNCS*, pages 566–598. Springer-Verlag, 2001.

[12] Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Cryptography against continuous memory attacks. In *Proceedings of FOCS 2010*, pages 511–520, 2010.

[13] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal of Computing*, 29(1):1–28, 1999.

[14] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings of Crypto 1986*, volume 263 of *LNCS*, pages 186–194. Springer-Verlag, 1986.

[15] G. Fuchsbauer. Commuting signatures and verifiable encryption. In *Proceedings of Eurocrypt 2011*, pages 224–245, 2011.

[16] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. Cryptology ePrint Archive, Report 2012/215, 2012. `http://eprint.iacr.org/2012/215`.

[17] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of STOC 2009*, pages 169–178, 2009.

[18] C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of STOC 2011*, pages 99–108, 2011.

[19] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *Proceedings of STOC 1985*, pages 186–208, 1985.

[20] J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Proceedings of Asiacrypt 2006*, volume 4284 of *LNCS*, pages 444–459. Springer-Verlag, 2006.

[21] J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Proceedings of Asiacrypt 2010*, pages 321–340, 2010.

[22] J. Groth and S. Lu. A non-interactive shuffle with pairing-based verifiability. In *Proceedings of Asiacrypt 2007*, volume 4833 of *LNCS*, pages 51–67. Springer-Verlag, 2007.

[23] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero-knowledge for NP. In *Proceedings of Eurocrypt 2006*, volume 4004 of *LNCS*, pages 339–358. Springer-Verlag, 2006.

[24] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Proceedings of Eurocrypt 2008*, volume 4965 of *LNCS*, pages 415–432. Springer-Verlag, 2008.

[25] H. Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. To appear in TTC 2012, 2012.

[26] S. Micali. Computationally sound proofs. *SIAM Journal of Computing*, 30(4):1253–1298, 2000.

[27] M. Prabhakaran and M. Rosulek. Rerandomizable RCCA encryption. In *Proceedings of Crypto 2007*, volume 4622 of *LNCS*, pages 517–534. Springer-Verlag, 2007.

[28] M. Prabhakaran and M. Rosulek. Homomorphic encryption with CCA security. In *Proceedings of ICALP 2008*, pages 667–678, 2008.

[29] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proceedings of FOCS 1999*, pages 543–553, 1999.

A Non-interactive Proof Systems

Our formal definition of zero-knowledge proofs is the same as that of CKLM, with the exception that we add the composable zero knowledge variant that we use in Section 3.1.

Definition A.1. **[7]** *A set of algorithms* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$ *constitute a non-interactive (NI) proof system for an efficient relation* R *with associated language* L_R *if completeness and soundness below are satisfied. A NI proof system is extractable if, in addition, the extractability property below is satisfied. A NI proof system is witness-indistinguishable (NIWI) if the witness-indistinguishability property below is satisfied. An NI proof system is zero-knowledge (NIZK) if the zero-knowledge property is satisfied. A NIZK proof system that is also extractable constitutes a non-interactive zero-knowledge proof of knowledge (NIZKPoK) system. A NIWI proof system that is also extractable constitutes a non-interactive witness-indistinguishable proof of knowledge (NIWIPoK) system.*

1. *Completeness [5]. For all* $\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$ *and* $(x,w) \in R$, $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ *for all proofs* $\pi \xleftarrow{\$} \mathcal{P}(\mathsf{crs}, x, w)$.

2. *Soundness [5]. For all PPT* $\mathcal{A}$*, and for* $\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$*, the probability that* $\mathcal{A}(\mathsf{crs})$ *outputs* (x, π) *such that* $x \notin L$ *but* $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$*, is negligible.* Perfect *soundness is achieved when this probability is 0.*

3. *Extractability [23]. There exists polynomial-time extractor algorithms* $E = (E_1, E_2)$ *such that* $E_1(1^k)$ *outputs* (σ_{ext}, τ_e)*, and* $E_2(\sigma_{ext}, \tau_e, x, \pi)$ *outputs a value* w *such that (1) a* σ_{ext} *output by* $E_1(1^k)$ *is indistinguishable from* crs *output by* $\mathsf{CRSSetup}(1^k)$*; (2) for all PPT* $\mathcal{A}$*, the probability that* $\mathcal{A}(\sigma_{ext}, \tau_e)$ *(where* $(\sigma_{ext}, \tau_e) \xleftarrow{\$} E_1(1^k)$*) outputs* (x, π) *such that* $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$ *and* $R(x, E_2(\sigma_{ext}, \tau_e, x, \pi)) = 0$*, is negligible.* Perfect *extractability is achieved if this probability is 0, and* σ_{ext} *is distributed identically to* crs.

4. *Witness indistinguishability [13]. For all* (x, w_1, w_2) *such that* $(x, w_1), (x, w_2) \in R$*, the tuple* (crs, π_1) *is indistinguishable from* (crs, π_2) *where* $\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$*, and for* $i \in \{1,2\}$, $\pi_i \xleftarrow{\$} \mathcal{P}(\mathsf{crs}, x, w_i)$. Perfect *witness indistinguishability is achieved when these two distributions are identical.*

5. *Zero knowledge [13]. There exists a polynomial-time simulator algorithm* $S = (S_1, S_2)$ *such that* $S_1(1^k)$ *outputs* (σ_{sim}, τ_s)*, and* $S_2(\sigma_{sim}, \tau_s, x)$ *outputs a value* π_s *such that for all* $(x, w) \in R$ *and PPT adversaries* $\mathcal{A}$*, the following two interactions are indistinguishable: in the first, we compute* $\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$ *and give* $\mathcal{A}$ crs *and oracle access to* $\mathcal{P}(\mathsf{crs}, \cdot, \cdot)$ *(where* $\mathcal{P}$ *will output* $\perp$ *on input* (x, w) *such that* $(x, w) \notin R$*); in the second, we compute* (σ_{sim}, τ_s) *and give* $\mathcal{A}$ σ_{sim} *and oracle access to* $S(\sigma_{sim}, \tau_s, \cdot, \cdot)$*, where, on input* (x, w)*,* S *outputs* $S_2(\sigma_{sim}, \tau_s, x)$ *if* $(x, w) \in R$ *and* $\perp$ *otherwise.* Perfect *zero knowledge is achieved if for all* $(x, w) \in R$*, these interactions are distributed identically, and* statistical *zero knowledge is achieved if for all* $(x, w) \in R$*, these interactions produce distributions that are statistically close (and thus indistinguishable to an unbounded adversary).* Composable *zero knowledge [20] is achieved if* $\mathcal{A}$ *is given* σ_{sim} *produced by* S_1 *in both games and is also allowed to see the associated* τ_s.

B The Shuffle Construction of CKLM

As defined by CKLM [7], a verifiable shuffle consists of three algorithms: Setup, which outputs parameters and the public keys for the (honest) mix authorities; $\mathsf{Shuffle}$, which takes in a set of ciphertexts and outputs a set of shuffled ciphertext and a proof of the correctness of the shuffle; and Verify, which checks the validity of the proofs. Their generic construction of a compactly verifiable shuffle (defined in Definition 2.4) combines a hard relation with generator $\mathcal{G}$, a re-randomizable encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReRand})$, a proof of knowledge $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$, and a cm-NIZK $(\mathsf{CRSSetup}', \mathcal{P}', \mathcal{V}')$ as follows:

- $\mathsf{Setup}(1^k)$: Generate $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$, $\mathsf{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$, and $\mathsf{crs}' \xleftarrow{\$} \mathsf{CRSSetup}'(1^k)$. For each mix server i, generate $(pk_i, sk_i) \xleftarrow{\$} \mathcal{G}(1^k)$, and output $(params := (pk, \mathsf{crs}, \mathsf{crs}'), sk, S := \{pk_i\}, \{sk_i\})$.
- $\mathsf{Enc}(params, \{m_i\}_{i=1}^n)$. Parse $params = (pk, \mathsf{crs}, \mathsf{crs}')$; then each user i can pick randomness r_i and encrypt his message m_i as $c_i \xleftarrow{\$} \mathsf{Enc}(pk, m_i; r_i)$ and form a proof of knowledge $\pi_i \xleftarrow{\$} \mathcal{P}(\mathsf{crs}, (pk, c_i), (m_i, r_i))$. This produces a collection $\{(c_i, \pi_i)\}_{i=1}^n$.

- $\mathsf{Shuffle}(params, \{c_i, \pi_i\}, \{c'_i\}, \pi, \{pk_j\})$: First check if this is the initial shuffle by seeing if $\pi = \bot$ and $\{c'_i\} = \{pk_j\} = \emptyset$. If it is the initial shuffle, check that $\mathcal{V}(\mathsf{crs}, c_i, \pi_i) = 1$ for all i, $1 \leq i \leq n$. If this check fails for some value of i, abort and output $\bot$. Otherwise, pick a random permutation $\varphi \leftarrow S_n$ and compute $c'_i \xleftarrow{\$} \mathsf{ReRand}(pk, \varphi(c_i))$ for all i. Now form a proof π for the shuffle performed by the user in possession of the secret key corresponding to pk_1 (i.e., the initial mix server). Output the tuple $(\{c_i, \pi_i\}, \{c'_i\}, \pi, \{pk_1\})$.

  Otherwise, if this is not the initial mix server, again check that $\mathcal{V}(\mathsf{crs}, c_i, \pi_i) = 1$ for all i, $1 \leq i \leq n$; check also that $\mathcal{V}'(\mathsf{crs}', (pk, \{c_i\}, \{c'_i\}, \{pk_j\}), \pi) = 1$. If any of these proofs do not pass verification abort and output $\bot$. Otherwise, continue by choosing a random permutation $\varphi \xleftarrow{\$} S_n$ and randomness $\{R_i\}$ for the encryption scheme, and computing $c''_i \xleftarrow{\$} \mathsf{ReRand}(pk, \varphi(c'_i); R_i)$ for all i. Finally, if the public key for the current mix server is pk_k then define $T := T_{(\varphi, \{R_i\}, \{sk_k, pk_k\}, \emptyset)}$ and run $\pi' \xleftarrow{\$} \mathsf{ZKEval}(\mathsf{crs}', T, (pk, \{c_i\}, \{c'_i\}, \{pk_j\}), \pi))$. Output the tuple $(\{c_i, \pi_i\}, \{c''_i\}, \pi', \{pk_j\} \cup pk_k)$.

- $\mathsf{Verify}(params, \{c_i, \pi_i\}, \{c'_i\}, \pi, \{pk_j\})$: Check that $\mathcal{V}(\mathsf{crs}, c_i, \pi_i) = 1$ for all i, $1 \leq i \leq n$; if this fails for any value of i abort and return 0. Otherwise, check that $\mathcal{V}'(\mathsf{crs}', (\{c_i\}, \{c'_i\}, \{pk_j\}), \pi) = 1$; again, if this fails output 0 and otherwise output 1.

CKLM proved that this generic construction yields a secure compactly verifiable shuffle, as defined in Definition 2.4. In Section 4, we use this framework to achieve specific efficiency guarantees (namely, a proof size of $O(n + \ell)$) by plugging in the cm-NIZK constructed in Section 3.