Malleable Proof Systems and Applications

Melissa Chase[1], Markulf Kohlweiss[2], Anna Lysyanskaya[3], and Sarah Meiklejohn[⋆4]

[1] Microsoft Research Redmond
melissac@microsoft.com
[2] Microsoft Research Cambridge
markulf@microsoft.com
[3] Brown University
anna@cs.brown.edu
[4] UC San Diego
smeiklej@cs.ucsd.edu

Abstract. Malleability for cryptography is not necessarily an opportunity for attack; in many cases it is a potentially useful feature that can be exploited. In this work, we examine notions of malleability for non-interactive zero-knowledge (NIZK) proofs. We start by defining a malleable proof system, and then consider ways to meaningfully *control* the malleability of the proof system, as in many settings we would like to guarantee that only certain types of transformations can be performed.
As our motivating application, we consider a shorter proof for verifiable shuffles. Our controlled-malleable proofs allow us for the first time to use one compact proof to prove the correctness of an entire multi-step shuffle. Each authority takes as input a set of encrypted votes and a controlled-malleable NIZK proof that these are a shuffle of the original encrypted votes submitted by the voters; it then permutes and re-randomizes these votes and updates the proof by exploiting its controlled malleability. As another application, we generically use controlled-malleable proofs to realize a strong notion of encryption security.
Finally, we examine malleability in existing proof systems and observe that Groth-Sahai proofs are malleable. We then go beyond this observation by characterizing all the ways in which they are malleable, and use them to efficiently instantiate our generic constructions from above; this means we can instantiate our proofs and all their applications using only the Decision Linear (DLIN) assumption.

1 Introduction

Let L be a language in NP. For concreteness, consider the language of Diffie-Hellman tuples: $(G, g, X, Y, Z) \in L_{DH}$ if there exist (x, y) such that g, X, Y, Z are elements of the group G, $X = g^x$, $Y = g^y$, and $Z = g^{xy}$. Suppose that we have a polynomial time prover P, and a verifier V, and P wants to convince V that $(G, g, X, Y, Z) \in L_{DH}$. Does the efficient prover need to know the values

⋆ Work done as an intern at Microsoft Research Redmond

(x, y) in order to convince the verifier? Not necessarily. Suppose that P is in possession of a non-interactive zero-knowledge (NIZK) proof π' that another tuple, $(G, g, X', Y', Z') \in L_{DH}$; suppose in addition that P happens to know (a, b) such that $X = (X')^a$, $Y = (Y')^b$, and $Z = (Z')^{ab}$. Can he, using the fact that he knows (a, b), transform π' into a NIZK π for the related instance (G, g, X, Y, Z)? In the sequel, we say that a proof system is *malleable* if it allows a prover to derive proofs of statements (such as $(G, g, X, Y, Z) \in L_{DH}$) not just from witnesses for their truth, but also from proofs of related statements (such as the proof π' that $(G, g, X', Y', Z') \in L_{DH}$).

In this paper, we consider malleability for non-interactive zero-knowledge proof systems. Our contributions are threefold: (1) definitions; (2) constructions; and (3) applications.

Motivating application. Why is malleability for non-interactive zero-knowledge proof systems an interesting feature? Let us present, as a motivating application, a verifiable vote shuffling scheme that becomes much more efficient if constructed using malleable proofs.

In a vote shuffling scheme, we have a set of encrypted votes $(v_1, \ldots, v_n)$ submitted by n voters; each vote v_i is an encryption of the voter's ballot under some trusted public key pk. The set of encrypted votes is then re-randomized[5] and shuffled, in turn, by several shuffling authorities. More precisely, let $(v_1^{(0)}, \ldots, v_n^{(0)}) = (v_1, \ldots, v_n)$; then each authority A_j takes as input $(v_1^{(j-1)}, \ldots, v_n^{(j-1)})$, picks a random permutation ρ and outputs $(v_1^{(j)}, \ldots, v_n^{(j)}) = (\tilde{v}_{\rho(1)}^{(j)}, \ldots, \tilde{v}_{\rho(n)}^{(j)})$, where $\tilde{v}_i^{(j)}$ is a randomization of $v_i^{(j-1)}$. At the end, the final set of encrypted votes $(v_1^{(\ell)}, \ldots, v_n^{(\ell)})$ is decrypted (for example, by a trustee who knows the decryption key corresponding to pk, or via a threshold decryption protocol) and the election can be tallied.

It is easy to see that, if we are dealing with an honest-but-curious adversary, this scheme guarantees both correctness and privacy as long as at least one of the authorities is honest. To make it withstand an active adversary, however, it is necessary for all participants (both the voters and the shuffling authorities) to prove (using a proof system with appropriate, technically subtle soundness and zero-knowledge guarantees) that they are correctly following the protocol. If these proofs are non-interactive, then the protocol gets the added benefit of being universally verifiable: anyone with access to the original encrypted votes and the output and proofs of each authority can verify that the votes were shuffled correctly. Thus, any party wishing to verify an election with n voters and ℓ shuffling authorities (and ℓ can potentially be quite large, for example a large polynomial in n for cases where a small group is voting on a very sensitive issue) will have to access $\Omega(n\ell)$ data just to read all the proofs.

[5] It is therefore important that the encryption scheme used is randomizable, so that on input a ciphertext $c = \mathsf{Enc}_{pk}(m; r)$ and randomness r' one can compute $c' = \mathsf{Enc}_{pk}(m; r * r')$, where $*$ is some group operation.

Can the proof that the verifier needs to read be shorter than that? The statement that needs to be verified is that the ciphertexts $(v_1^{(\ell)}, \ldots, v_n^{(\ell)})$ can be obtained by randomizing and permuting the original votes $(v_1, \ldots, v_n)$. The witness for this statement is just some permutation (that is obtained by composing the permutations applied by individual authorities) and randomness that went into randomizing each ciphertext (that can be obtained by applying the group operation repeatedly to the randomness used by each authority); thus, ignoring the security parameter, the length of the witness can potentially be only $O(n)$.[6]

Of course, no individual authority knows this witness. But each authority A_j is given a proof π_{j-1} that, up until now, everything was permuted and randomized correctly. Using controlled malleable proofs, from this π_{j-1} and its own secret permutation ρ_j and vector of random values $(r_1^{(j)}, \ldots, r_n^{(j)})$, A_j should be able to compute the proof π that his output is a permutation and randomization of the original votes.

In this paper, we give a construction that roughly corresponds to this outline, and prove its security. We must stress that even though this construction is a more or less direct consequence of the new notion of controllable malleability, and therefore may seem obvious in hindsight, it is actually a significant breakthrough as far as the literature on efficient shuffles is concerned: for the first time, we obtain a non-interactive construction in which the complexity of verifying the tally with ℓ authorities is not ℓ times the complexity of verifying the tally with one authority!

Our definitions. Care needs to be taken when defining malleable NIZKs suitable for the above application. We first need malleability itself: from an instance x' and a proof π' that $x' \in L$, we want to have an efficient algorithm ZKEval that computes another instance $x = T(x')$ and a proof π that $x \in L$, where T is some transformation (in the above example, x' is a set of ciphertexts, and T is a re-randomization and permutation of these ciphertexts). We want the resulting proof to be *derivation private*, so that, from x and π, it is impossible to tell from which T and x' they were derived. (In the above example, it should be impossible to tell how the ciphertexts were shuffled.) Finally, we want to ensure that the proof system is sound, even in the presence of a zero-knowledge simulator that provides proofs of adversarially chosen statements (so that we can relate the real-world experiment where the adversary participates in shuffling the ciphertexts to an ideal-world process that only has access to the final tally). To this end, we define *controlled malleability* (as opposed to malleability that is out of control!) that guarantees that, from proofs computed by an adversary, an extractor (with a special extracting trapdoor) can compute either a witness to the truth of the

[6] In our concrete construction we use a very simple approach to proving a shuffle in which we represent the permutation as a matrix, thus the length of a single shuffle proof is $O(n^2)$. This could potentially be improved using more sophisticated verifiable shuffle techniques as we will mention later. Additionally, because we want to be able to verify the fact that each authority participated in the shuffle, we will include a public key for each authority involved and the size will actually grow to $O(n^2 + \ell)$.

statement, or the transformation T and some statement for which the simulator had earlier provided a proof.

Our definitional approach to derivation privacy is inspired by circuit privacy for fully homomorphic encryption [27, 40, 39, 13], also called function privacy or unlinkability. Our definitional approach to controlled malleability is inspired by the definitions of HCCA (homomorphic-CCA) secure encryption due to Prabhakaran and Rosulek [36]; it is also related to the recently proposed notion of targeted malleability due to Boneh, Segev, and Waters [12]. (See the full version of our paper [15] for more detailed comparison with these notions.)

Our construction. Our construction of controlled-malleable and derivation-private NIZK proof systems consists of two steps. First, in Section 3, we show how to construct a controlled-malleable derivation-private NIZK from any derivation-private non-interactive witness-indistinguishable (NIWI) proof system and secure signature scheme. Then, in Section 4.1 we show how to instantiate the appropriate NIWI proof system and signature scheme using the Groth-Sahai proof system [33] and a recent structure-preserving signature due to Chase and Kohlweiss [14]; this combination means we can instantiate our proofs (and in fact all of the constructions in our paper) using the Decision Linear (DLIN) assumption [11]. The size of the resulting proof is linear in the size of the statement, although the size of the structure-preserving signature does make it admittedly much less efficient than Groth-Sahai proofs alone.

At the heart of our construction is the observation that the Groth-Sahai (GS) proof system is malleable in ways that can be very useful. This feature of GS proofs has been used in prior work in a wide variety of applications: Belenkiy et al. [8] use the fact that the GS proof system can be randomized in order to construct delegatable anonymous credentials; Dodis et al. [20] uses homomorphic properties of GS proofs in order to create a signature scheme resilient to continuous leakage; Acar and Nguyen [7] use malleability to delegate and update non-membership proofs for a cryptographic accumulator in their implementation of a revocation mechanism for delegatable anonymous credentials; and Fuchsbauer [24] uses malleability to transform a proof about the contents of a commitment into a proof of knowledge of a signature on the committed message in his construction of commuting signatures.

Compact verifiable shuffles. Armed with a construction of a controlled-malleable and derivation-private NIZK, we proceed, in Section 6, to consider the problem of obtaining a verifiable shuffle with compact proofs. We formally define this concept, describe a generic construction from a semantically-secure encryption scheme and a controlled-malleable and derivation-private NIZK following the outline above, and finally argue that we can in fact construct such a proof system for the appropriate set of transformations based on the instantiation described in Section 4.1.

An application to encryption. Can controlled malleability of NIZKs give us controlled malleability for encryption? That is to say, can we achieve a meaningful notion of adaptively secure encryption, even while allowing computations on encrypted data? Similarly to controlled malleability for proofs, we define in Section 5 controlled malleability for encryption (directly inspired by the notion of HCCA security; in this, our work can be considered closely related to that of Prabhakaran and Rosulek), and show a *general* method for realizing it for broad classes of unary transformations, using a semantically secure encryption scheme with appropriate homomorphic properties and a controlled-malleable and derivation-private NIZK for an appropriate language as building blocks. Our construction follows easily from these properties, resulting in a much simpler proof of security than was possible in previous works. (We note that our methods do not extend to n-ary transformations for $n > 1$, because the same limitations that apply for HCCA security, pointed out by Prabhakaran and Rosulek, also apply here. The work of Boneh et al. overcomes this and allows for binary transformations as well, with the sacrifice that, unlike both our scheme and the Prabhakaran-Rosulek scheme, the encryption scheme can no longer satisfy function privacy.)

Related work on shuffling ciphertexts. Shuffles and mixing in general were introduced by Chaum in 1981 [16], and the problem of verifiable shuffles was introduced by Sako and Kilian in 1995 [38]; the work on verifiable shuffles in the ensuing sixteen years has been extensive and varied [2, 26, 6, 34, 29, 25, 41, 31]. In 1998, Abe [1] considered the problem of compact proofs of shuffles. Unlike our non-interactive solution, his solution is based on an interactive protocol[7] wherein all mixing authorities must jointly generate a proof with size independent of ℓ; in comparison, our solution allows each authority to be offline before and after it performs its shuffling of the ciphertexts. In terms of approaches most similar to our own, Furukawa and Sako [26] use a permutation matrix to shuffle the ciphertexts; they then prove that the matrix used was in fact a permutation matrix, and that it was applied properly. Most recently, Groth and Lu [31] give a verifiable shuffle that is non-interactive (the only one to do so without use of the Fiat-Shamir heuristic [23]), uses pairing-based verifiability, and obtains $O(n)$ proof size for a single shuffle. The advantage, as outlined above, that our construction has over all of these is that one proof suffices to show the security of the entire shuffle; we do not require a separate proof from each mix server. An interesting open problem is to see if there is some way to combine some of these techniques with an appropriate controlled-malleable proof system to obtain a multi-step shuffle with optimal proof size $O(n + \ell)$.

2 Definitions and Notation

Our definitional goal is to formulate what it means to construct a proof of a particular statement using proofs of related statements. Let $R(\cdot, \cdot)$ be some relation that is polynomial-time computable in the size of its first input; in the

[7] The protocol could in fact be made non-interactive, but only using the Fiat-Shamir heuristic [23] and thus the random oracle model.

sequel we call such a relation an *efficient* relation. Associated with R, there is an NP language $L_R = \{x \mid \exists\, w \textit{ such that } R(x, w) = \textit{TRUE}\}$.[8] For example, let $R(x, w)$ be a relation that holds if the witness $w = (a, b)$ demonstrates that the instance $x = (G, g, A, B, C)$ is a Diffie-Hellman tuple; i.e. it holds if $g, A, B, C \in G$ and $A = g^a$, $B = g^b$, $C = g^{ab}$. Then the language associated with R is L_{DH} defined in the introduction. We often write $(x, w) \in R$ to denote that $R(x, w) = \textit{TRUE}$.

Let $T = (T_x, T_w)$ be a pair of efficiently computable n-ary functions, where $T_x : \{\{0,1\}^*\}^n \to \{0,1\}^*$, $T_w : \{\{0,1\}^*\}^n \to \{0,1\}^*$. In what follows, we refer to such a tuple T as an n-ary *transformation*.

Definition 2.1. *An efficient relation R is* closed *under an n-ary transformation $T = (T_x, T_w)$ if for any n-tuple $\{(x_1, w_1), \ldots, (x_n, w_n)\} \in R^n$, the pair $(T_x(x_1, \ldots, x_n), T_w(w_1, \ldots, w_n)) \in R$. If R is closed under T, then we say that T is* admissible *for R. Let $\mathcal{T}$ be some set of transformations; if for every $T \in \mathcal{T}$, T is admissible for R, then $\mathcal{T}$ is an* allowable *set of transformations.*

For example, for the DH relation R described above, consider $T = (T_x, T_w)$ where for some (a', b'), $T_x(G, g, A, B, C) = (G, g, A^{a'}, B^{b'}, C^{a'b'})$ and $T_w(a, b) = (aa', bb')$; then the Diffie-Hellman relation R is closed under transformation T, and additionally the set $\mathcal{T}$ of transformations of this form (i.e., where there is a transformation T corresponding to any pair (a', b')) is an allowable set of transformations.

Our goal is to define non-interactive zero-knowledge and witness-indistinguishable proof systems for efficient relations R that are (1) malleable with respect to an allowable set of transformations $\mathcal{T}$; that is to say, for any $T \in \mathcal{T}$, given proofs for $x_1, \ldots x_n \in L_R$, they can be transformed into a proof that $T_x(x_1, \ldots, x_n) \in L_R$; and (2) derivation-private; that is to say, the resulting proof cannot be distinguished from one freshly computed by a prover on input $(T_x(x_1, \ldots, x_n), T_w(w_1, \ldots, w_n))$. Before we can proceed, however, we need to recall the definition of a non-interactive zero-knowledge proof system.

A proof system for an efficient relation R allows a prover to prove that a value x is in the associated language L_R. A non-interactive (NI) proof system with efficient provers [10, 21] consists of three PPT algorithms: the algorithm $\mathsf{CRSSetup}(1^k)$ that generates a common reference string (CRS) σ_{crs}, the algorithm $\mathcal{P}(\sigma_{\text{crs}}, x, w)$ that outputs a proof π that $x \in L_R$, and the algorithm $\mathcal{V}(\sigma_{\text{crs}}, x, \pi)$ that verifies the proof; such a proof system must be complete (meaning the verifier will always accept an honestly generated proof) and sound (meaning that a verifier cannot be fooled into accepting a proof for a false statement). A NI zero-knowledge proof (NIZK) [28, 10], additionally requires the existence of a simulator S that can generate proofs without access to a witness, while a NI witness-indistinguishable proof system [22] has the requirement that proofs generated using two different witnesses for the same x are indistinguishable from

[8] Without the restriction that R is efficient in its first input, the resulting language won't necessarily be in NP.

each other. A NI proof of knowledge [28, 9] additionally requires an efficient extractor algorithm E that, on input a proof that $x \in L_R$, finds a witness for the instance x.

We use the original definitions for completeness and soundness of NI proof systems in the common-reference-string model [10]. The version of the definition of zero-knowledge for NIZK we give is originally due to Feige, Lapidot and Shamir (FLS) [21]; they call it "adaptive multi-theorem NIZK." We also use the FLS definition of witness indistinguishability. The version of knowledge extraction we use is a generalization of the definition of knowledge extraction given by Groth, Ostrovsky and Sahai (GOS) [32]: they defined the notion of *perfect* knowledge extraction, while here we find it useful to generalize their definition, in the straightforward way, to the case when extraction is not perfect. Due to space constraints, a formal definition for non-interactive zero-knowledge proofs of knowledge (NIZKPoK) systems and non-interactive witness-indistinguishable proofs of knowledge (NIWIPoK) systems combining all of these concepts can be found in the full version of our paper [15].

Next, we define a malleable proof system; i.e., one in which, from proofs $(\pi_1, \ldots, \pi_n)$ that $(x_1, \ldots, x_n) \in L$, one can compute a proof π that $T_x(x_1, \ldots, x_n) \in L$, for an admissible transformation $T = (T_x, T_w)$:

Definition 2.2 (Malleable non-interactive proof system). *Let* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$ *be a non-interactive proof system for a relation R. Let $\mathcal{T}$ be an allowable set of transformations for R. Then this proof system is* malleable with respect to $\mathcal{T}$ *if there exists an efficient algorithm* ZKEval *that on input* $(\sigma_{crs}, T, \{x_i, \pi_i\})$, *where $T \in \mathcal{T}$ is an n-ary transformation and $\mathcal{V}(\sigma_{crs}, x_i, \pi_i) = 1$ for all i, $1 \leq i \leq n$, outputs a valid proof π for the statement $x = T_x(\{x_i\})$ (i.e., a proof π such that $\mathcal{V}(\sigma_{crs}, x, \pi) = 1$).*

Going back to our above example, the algorithm ZKEval will take as input the transformation T (which is equivalent to taking as input the values a' and b'), and a proof π_1 that $x_1 = (G, g, A, B, C)$ is a DH tuple, and output a proof π that $x = T_x(x_1) = (G, g, A^{a'}, B^{b'}, C^{a'b'})$ is a DH tuple.

2.1 Derivation privacy for proofs

In addition to malleability, we must also consider a definition of derivation privacy analogous to the notion of function privacy for encryption. (In the encryption setting this is also called unlinkability [36]; for a formal definition see the full version [15].) We have the following definition:

Definition 2.3 (Derivation privacy). *For a NI proof system* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V}, \mathsf{ZKEval})$ *for 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.* $\sigma_{crs} \xleftarrow{\$} \mathsf{CRSSetup}(1^k)$.

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

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

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

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

In some cases, we would like to work with a stronger definition that applies only for NIZKs. In this case, the adversary will not be asked to provide witnesses or distinguish between the outputs of the prover and ZKEval, but instead between the zero-knowledge simulator and ZKEval. It will also be given the simulation trapdoor so that it can generate its own simulated proofs.

Definition 2.4 (Strong derivation privacy). *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* $\perp$. *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.* $b' \xleftarrow{\$} \mathcal{A}(\mathsf{state}, \pi)$.

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

As we will see in Section 3, schemes that satisfy the weaker notion of derivation privacy can in fact be generically "boosted" to obtain schemes that satisfy the stronger notion. We can also show a generic way to obtain derivation privacy using malleability and the notion of *randomizability* for proofs, defined by Belenkiy et al. [8]; this can be found in the full version of the paper [15].

3 Controlled Malleability for NIZKs

Is the notion of malleability compatible with the notion of a proof of knowledge or with strong notions like simulation soundness? Recall that to achieve simulation soundness, as defined by Sahai and de Santis et al. [37, 19], we intuitively want an adversary $\mathcal{A}$ to be unable to produce a proof of a new false statement even if it can request many such proofs from the simulator; for the even stronger notion of simulation-extractability as defined by de Santis et al. and Groth [19, 30], a proof system must admit an efficient extractor that finds witnesses to all statements proved by an adversary, again even when the adversary has access to a simulator.

Malleability, in contrast, explicitly allows an adversary to take as input the values x', π', apply some admissible transformation T to x' to obtain $x = T_x(x')$, and compute a proof π that $x \in L_R$; importantly, the adversary can do all this without knowing the original witness w'. Suppose, for a malleable proof system, that the adversary is given as input a *simulated* proof π' that was generated without access to the witness w' for x', and for concreteness let T be the identity transformation. Then requiring that, on input (x, π), the extractor should output w, implies that membership in L_R can be tested for a given x by computing a simulated proof, mauling it, and then extracting the witness from the resulting proof (formally, this would mean that $L_R \in \mathbf{RP}$). Thus, seemingly, one cannot reconcile the notion of malleability with that of a simulation-extractable proof of knowledge.

Surprisingly, however, under a relaxed but still meaningful extractability requirement, we can have a proof system that is both malleable and simulation-extractable to a satisfactory extent; we call this notion *controlled malleability.* Essentially this definition will require that the extractor can extract either a valid witness, or a previously proved statement x' and a transformation T in our allowed set $\mathcal{T}$ that could be used to transform x' into the new statement x. To demonstrate that our definition is useful, we will show in Section 5 that it can be used to realize a strong notion of encryption security, and in Section 6 that it can also be used to reduce the overall size of proofs for verifiable shuffles.

Definition 3.1 (Controlled-malleable simulation sound extractability). *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 an allowable 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 $(\sigma_{crs}, \tau_s, \tau_e)$ such that (σ_{crs}, τ_s) is distributed identically to the output of S_1. Let $\mathcal{A}$ be given, and consider the following game:*

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

We say that the NIZKPoK satisfies controlled-malleable simulation-sound extractability *(CM-SSE, for short) 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 $\mathcal{V}(\sigma_{crs}, x, \pi) = 1$ and $(x, \pi) \notin Q$ (where Q is the set of queried statements and their responses) but either (1) $w \neq \bot$ and $(x, w) \notin R$; (2) $(x', T) \neq (\bot, \bot)$ and either $x' \notin Q_x$ (the set of queried instances), $x \neq T_x(x')$, or $T \notin \mathcal{T}$; or (3) $(w, x', T) = (\bot, \bot, \bot)$ is at most $\nu(k)$.*

This definition is actually closely related to simulation-extractability; in fact, if we restrict our set of transformations to be $\mathcal{T} = \emptyset$, we obtain exactly Groth's notion of simulation-sound extractability. Note also that this definition does not require that a proof system actually be malleable, it only requires that, should it happen to be malleable, this malleability be limited in a controlled way. Thus, a simulation-sound extractable proof system would also satisfy our definition, for any set $\mathcal{T}$, even though it is not malleable. We refer to a proof system that is both strongly derivation private and controlled-malleable simulation-sound extractable as a *controlled-malleable NIZK* (cm-NIZK).

Finally, note that our definition applies only to unary transformations. This is because our requirement that we can extract the transformation T means we cannot hope to construct cm-NIZKs for n-ary transformations where $n > 1$, as this would seem to necessarily expand the size of the proof (similarly to what Prabhakaran and Rosulek show for HCCA encryption [36]). We therefore achieve cm-NIZKs for classes of unary transformations that are closed under composition (i.e., $T' \circ T \in \mathcal{T}$ for all $T, T' \in \mathcal{T}$). In addition, our simulation strategy depends on the identity transformation being a member of $\mathcal{T}$, so we can achieve cm-NIZKs only for classes of transformations that include the identity transformation.

A generic construction

Let R be an efficient relation, and suppose $\mathcal{T}$ is an allowable set of transformations for R that contains the identity transformation; suppose further that membership in $\mathcal{T}$ is efficiently testable. Let $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a secure signature scheme. Let $(\mathsf{CRSSetup}_{\mathsf{WI}}, \mathcal{P}_{\mathsf{WI}}, \mathcal{V}_{\mathsf{WI}})$ be a NIWIPoK for the following relation R_{WI}: $((x, vk), (w, x', T, \sigma)) \in R_{\mathsf{WI}}$ if $(x, w) \in R$ or $\mathsf{Verify}(vk, \sigma, x') = 1$, $x = T_x(x')$, and $T \in \mathcal{T}$. Consider the proof system $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$ defined as follows:

- $\mathsf{CRSSetup}(1^k)$: First generate $\sigma_{\mathsf{WI}crs} \xleftarrow{\$} \mathsf{CRSSetup}_{\mathsf{WI}}(1^k)$ and $(vk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$; then output $\sigma_{\mathrm{crs}} := (\sigma_{\mathsf{WI}crs}, vk)$.
- $\mathcal{P}(\sigma_{\mathrm{crs}}, x, w)$: Output $\pi \xleftarrow{\$} \mathcal{P}_{\mathsf{WI}}(\sigma_{\mathsf{WI}crs}, x_{\mathsf{WI}}, w_{\mathsf{WI}})$, where $x_{\mathsf{WI}} = (x, vk)$ and $w_{\mathsf{WI}} = (w, \perp, \perp, \perp)$.
- $\mathcal{V}(\sigma_{\mathrm{crs}}, x, \pi)$: Output $\mathcal{V}_{\mathsf{WI}}(\sigma_{\mathsf{WI}crs}, x_{\mathsf{WI}}, \pi)$ where $x_{\mathsf{WI}} = (x, vk)$.

To obtain strong derivation privacy with respect to R and $\mathcal{T}$ we also require the NIWIPoK to be derivation private with respect to R_{WI} and a set of transformations $\mathcal{T}_{\mathsf{WI}}$ such that for every $T' = (T'_x, T'_w) \in \mathcal{T}$ there exists a $T_{\mathsf{WI}}(T') \in \mathcal{T}_{\mathsf{WI}}$. For $T_{\mathsf{WI}}(T') = (T_{\mathsf{WI},x}, T_{\mathsf{WI},w})$ we require that $T_{\mathsf{WI},x}(x, vk) = (T'_x(x), vk)$, and $T_{\mathsf{WI},w}(w, x', T, \sigma) = (T'_w(w), x', T' \circ T, \sigma)$. Assuming our underlying NIWI is malleable, we can define ZKEval in terms of $\mathsf{ZKEval}_{\mathsf{WI}}$:

- $\mathsf{ZKEval}(\sigma_{\mathrm{crs}}, T, x, \pi)$: Output $\mathsf{ZKEval}_{\mathsf{WI}}(\sigma_{\mathsf{WI}crs}, T_{\mathsf{WI}}(T), x_{\mathsf{WI}}, \pi)$ where $x_{\mathsf{WI}} = (x, vk)$.

To see that this construction gives us the desired properties, we have the following three theorems; due to space constraints, the proofs can be found in the full version of our paper [15]:

Theorem 3.1. *If the underlying non-interactive proof system is witness indistinguishable, the scheme described above is zero knowledge.*

Theorem 3.2. *If the underlying signature scheme is EUF-CMA secure and the underlying NIWIPoK is extractable, the scheme described above satisfies controlled-malleable simulation-sound extractability.*

Theorem 3.3. *If the underlying NIWIPoK is derivation private for $\mathcal{T}_{\mathsf{WI}}$ (as defined in Definition 2.3), then the scheme described above is strongly derivation private for $\mathcal{T}$ (as defined in Definition 2.4).*

In addition, we would like to ensure that this construction can in fact be instantiated efficiently for many useful sets $\mathcal{T}$ with a derivation-private NIWIPoK; it turns out that this can be done by combining Groth-Sahai proofs [33] with a special type of signature called a *structure-preserving signature.* For more details, we defer to Section 4.2.

4 Instantiating cm-NIZKs Using Groth-Sahai Proofs

In this section, we explore the malleability of Groth-Sahai (GS) proofs [33]. This will allow us to efficiently instantiate controlled-malleable proofs for a large class of transformations.

4.1 Malleability for Groth-Sahai Proofs

We aim to fully characterize the class of transformations with respect to which GS proofs can be made malleable. First, we recall that GS proofs allow a prover to prove knowledge of a satisfying assignment to a list of (homogeneous) *pairing product equations* eq of the form $\prod_{i,j\in[1..n]} e(x_i,x_j)^{\gamma_{ij}} = 1$ concerning the set of variables $x_1,\ldots,x_n \in \mathbb{G}$. Furthermore, some of the variables in these equations may be fixed to be specific constant values (for example, the public group generator g). In what follows we will use $a, b, c, \ldots$ to denote fixed constants, and $\mathbf{x}, \mathbf{y}, \mathbf{z}, \ldots$ to denote unconstrained variables. An instance x of such a *pairing product statement* consists of the list of equations $\mathsf{eq}_1,\ldots,\mathsf{eq}_\ell$ (fully described by their exponents $\{\gamma_{ij}^{(1)}\},\ldots,\{\gamma_{ij}^{(\ell)}\}$) and the values of the constrained variables (fully described by the list $a_1,\ldots,a_{n'} \in \mathbb{G}$ for $n' \leq n$).

In the existing literature, there are already various examples [20, 7, 24] of ways in which pairing product statements and the accompanying Groth-Sahai proofs can be mauled. Here, we attempt to generalize these previous works by providing a characterization of *all* the ways in which GS proofs of pairing product statements can be mauled; we then show, in the full version of our paper [15], how these previous examples can be obtained as special cases of our general characterization.

To start, we describe transformations on pairing product instances in terms of a few basic operations. We will say that any transformation that can be described as a composition of these operations is a *valid transformation.* For each valid transformation we show, in the full version, that there is a corresponding ZKEval procedure that updates the GS proof to prove the new statement. Finally, we present in the full version some other convenient operations that can be derived from our minimal set.

To help illustrate the usage of our basic transformations, we consider their effect on the pairing product instance $(\mathsf{eq}_1, \mathsf{eq}_2, a, b)$, where $\mathsf{eq}_1 := e(\mathbf{x}, b)e(a, b) = 1$ and $\mathsf{eq}_2 := e(a, \mathbf{y}) = 1$. Note that here we will describe the transformations in terms of their effect on the instances, but in all of these operations the corresponding witness transformations T_w are easily derived from the instance transformations T_x.

Definition 4.1. *(Informal) A* valid transformation *is one that can be expressed as some combination of (a polynomial number of) the following six operations:*

1. *Merge equations:* $\mathsf{MergeEq}(\mathsf{eq}_i, \mathsf{eq}_j)$ *adds the product of* eq_i *and* eq_j *as a new equation.*
   Ex. $\mathsf{MergeEq}(\mathsf{eq}_1, \mathsf{eq}_2)$ *adds the equation* $e(\mathbf{x}, b)e(a, b)e(a, \mathbf{y}) = 1$
2. *Merge variables:* $\mathsf{MergeVar}(x, y, z, S)$ *generates a new variable z. If x and y are both constants, z will have value xy. Otherwise $\mathbf{z}$ will be unconstrained. For every variable w in the set S, we add the equation $e(xy, w)^{-1}e(z, w) = 1$.*[9]
   Ex. $\mathsf{MergeVar}(x, a, z, \{x, b, z\})$ *adds the variable z and the equations* $e(\mathbf{x}a, \mathbf{x})^{-1}e(\mathbf{z}, \mathbf{x}) = 1$, $e(\mathbf{x}a, b)^{-1}e(\mathbf{z}, b) = 1$, *and* $e(\mathbf{x}a, \mathbf{z})^{-1}e(\mathbf{z}, \mathbf{z}) = 1$.
3. *Exponentiate variable:* $\mathsf{ExpVar}(x, \delta, z, S)$ *generates a new variable z. If x is a constant, $z = x^\delta$, otherwise $\mathbf{z}$ will be unconstrained. For every variable $w \in S$, we add the equation $e(x, w)^{-\delta}e(z, w) = 1$.*
   Ex. $\mathsf{ExpVar}(x, \delta, z, \{x, b, z\})$ *adds the variable z and the equations* $e(\mathbf{x}, \mathbf{x})^{-\delta}e(\mathbf{z}, \mathbf{x}) = 1$, $e(\mathbf{x}, b)^{-\delta}e(\mathbf{z}, b) = 1$, *and* $e(\mathbf{x}, \mathbf{z})^{-\delta}e(\mathbf{z}, \mathbf{z}) = 1$.
4. *Add constant equation:* $\mathsf{Add}(\{a_i\}, \{b_j\}, \{\gamma_{ij}\})$ *takes a set of constants a_i, b_i, satisfying a pairing product equation $\prod e(a_i, b_j)^{\gamma_{ij}} = 1$ and adds these variables and the new equation to the statement.*
   Ex. $\mathsf{Add}(\{g\}, \{1\}, \{1\})$ *adds the variables $g, 1$ and equation* $\mathsf{eq}_3 := e(g, 1) = 1$. *We often write as a shorthand* $\mathsf{Add}(\mathsf{eq}_3 := e(g, 1) = 1)$.
5. *Remove equation:* $\mathsf{RemoveEq}(\mathsf{eq}_i)$ *simply removes equation* eq_i *from the list.*
   Ex. $\mathsf{RemoveEq}(\mathsf{eq}_2)$ *removes the equation $e(a, \mathbf{y}) = 1$ from the equation list.*
6. *Remove variable:* $\mathsf{RemoveVar}(x)$ *removes the variable x from the variable set iff x does not appear in any of the listed equations.*
   Ex. We cannot remove any of the variables from the example statement. However, we could do $\mathsf{RemoveEq}(\mathsf{eq}_2)$ *and then* $\mathsf{RemoveVar}(y)$*, which would remove the equation $e(a, \mathbf{y}) = 1$ from the equation list and the variable y from the set of variables.*

A proof of the following lemma appears in the full version:

Lemma 4.1. *There exists an efficient procedure* ZKEval *such that given any pairing product instance x, any valid transformation T, and any accepting Groth-Sahai proof π for x,* $\mathsf{ZKEval}(x, \pi, T)$ *produces an accepting proof for $T(x)$.*

[9] This is shorthand for $e(x, w)^{-1}e(y, w)^{-1}e(z, w) = 1$.

4.2 An efficient instantiation of controlled malleable NIZKs

Looking back at Section 3 we see that there are two main components needed to efficiently instantiate a controlled-malleable NIZK proof system: appropriately malleable proofs and signatures that can be used in conjunction with these proofs.

First we consider the set of relations and tranformations for which we can use Groth-Sahai proofs to construct the necessary malleable NIWIPoKs.

Definition 4.2. *For a relation R and a class of transformations $\mathcal{T}$, we say $(R, \mathcal{T})$ is* CM-friendly *if the following six properties hold: (1) representable statements: any instance and witness of R can be represented as a set of group elements; (2) representable transformations: any transformation in $\mathcal{T}$ can be represented as a set of group elements; (3) provable statements: we can prove the statement $(x, w) \in R$ using pairing product equations; (4) provable transformations: we can prove the statement "$T_x(x') = x$ for $T \in \mathcal{T}$" using pairing product equations; (5) transformable statements: for any $T \in \mathcal{T}$ there is a valid transformation from the statement "$(x, w) \in R$" to the statement "$(T_x(x), T_w(w)) \in R$"; and (6) transformable transformations: for any $T, T' \in \mathcal{T}$ there is a valid transformation from the statement "$T_x(x') = x$ for $T = (T_x, T_w) \in \mathcal{T}$" to the statement "$T'_x \circ T_x(x') = T'_x(x)$ for $T' \circ T \in \mathcal{T}$."*

In order for the signatures to be used within our construction, we know that they need to have pairing-based verifiability (i.e., we can represent the Verify algorithm in terms of a set of GS equations), and that the values being signed must be group elements so that they can be efficiently extracted from the proof (as GS proofs are extractable for group elements only, not exponents). These requirements seem to imply the need for *structure-preserving signatures* [3], which we can define for the symmetric setting as follows:

Definition 4.3. *A signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ over a bilinear group (p, G, G_T, g, e) is said to be* structure preserving *if the verification key, messages, and signatures all consist of group elements in G, and the verification algorithm evaluates membership in G and pairing product equations.*

Since their introduction, three structure-preserving signature schemes have emerged that would be suitable for our purposes; all three have advantages and disadvantages. The first, due to Abe, Haralambiev, and Ohkubo [5, 3] is quite efficient but uses a slightly strong q-type assumption. The second, due to Abe et al. [4], is optimally efficient but provably secure only in the generic group model. The third and most recent, due to Chase and Kohlweiss [14], is significantly less efficient than the previous two, but relies for its security on Decision Linear (DLIN) [11], which is already a relatively well-established assumption.

Because we can also instantiate GS proofs using DLIN, we focus on this last structure-preserving signature, keeping in mind that others may be substituted in for the sake of efficiency (but at the cost of adding an assumption). Putting these signatures and GS proofs together, we can show our main result of this section: given any CM-friendly relation and set of transformations $(R, \mathcal{T})$, we can combine structure-preserving signatures and malleable proofs to obtain a cm-NIZK. This can be stated as the following theorem (a proof of which can be found in the full version of our paper):

Theorem 4.1. *Given a derivation private NIWIPoK for pairing product statements that is malleable for the set of all valid transformations, and a structure preserving signature scheme, we can construct a cm-NIZK for any CM-friendly relation and transformation set* $(R, \mathcal{T})$.

In the full version of our paper, we show that Groth-Sahai proofs are malleable for the set of all valid transformations (as outlined in Definition 4.1). As Groth-Sahai proofs and structure-preserving signatures can both be constructed based on DLIN, we obtain the following theorem:

Theorem 4.2. *If DLIN holds, then we can construct a cm-NIZK that satisfies strong derivation privacy for any CM-friendly relation and transformation set* $(R, \mathcal{T})$.

5 Controlled Malleability for Encryption

As we mentioned earlier, malleability can also be an attractive feature for a cryptosystem: it allows computation on encrypted data. On the other hand, it seems to be in conflict with security: if a ciphertext can be transformed into a ciphertext for a related message, then the encryption scheme is clearly not secure under an adaptive chosen ciphertext attack, which is the standard notion of security for encryption.

Prabhakaran and Rosulek [35, 36] were the first to define and realize a meaningful notion of security in this context. They introduced re-randomizable CCA security (RCCA) [35] and homomorphic CCA security (HCCA) [36]. In a nutshell, their definition of security is given as a game between a challenger and an adversary; the adversary receives a public key and a challenge ciphertext and can query the challenger for decryptions of ciphertexts. The challenger's ciphertext c^* is either a valid encryption of some message, or a dummy ciphertext; in the former case, the challenger answers the decryption queries honestly; in the latter case, the challenger may decide that a decryption query is a "derivative" ciphertext computed from c^* using some transformation T; if this is an allowed transformation, the challenger responds with $T(m)$, else it rejects the query. The adversary wins if it correctly guesses whether its challenge ciphertext was meaningful.[10] Prabhakaran and Rosulek achieve their notion of security under the decisional Diffie-Hellman assumption using ad-hoc techniques reminiscent of the Cramer-Shoup [17] cryptosystem.

In this section, we show that controlled-malleable NIZKs can be used as a general tool for achieving RCCA and HCCA security. Our construction is

[10] A formal definition and more detailed explanation of their notion of homomorphic-CCA (HCCA) security can be found in the full version of our paper [15].

more modular than that of Prabhakaran and Rosulek: we construct a controlled-malleable-CCA-secure encryption scheme generically from a semantically secure one and a cm-NIZK for an appropriate language; where controlled-malleable-CCA security is our own notion of security that is, in some sense, a generalization of RCCA security and also captures the security goals of HCCA security. We then show how our construction can be instantiated using Groth-Sahai proofs, under the DLIN assumption in groups with bilinear maps.

5.1 Definition of Controlled-Malleable CCA Security

Our definitional goals here are (1) to give a definition of controlled malleability for encryption that closely mirrors our definition of controlled malleability for proofs, and (2) to give a definition that can be easily related to previous notions such as CCA, RCCA, and HCCA. We call this notion of security *controlled-malleable CCA* (CM-CCA) security.

Following Prabhakaran and Rosulek [36], CM-CCA requires the existence of two algorithms, SimEnc and SimExt. SimEnc creates ciphertexts that are distributed indistinguishably from regular ciphertexts (those generated using the encryption algorithm Enc), but contain no information about the queried message; this is modeled by having SimEnc not take any message as input. SimExt allows the challenger to track "derivative" ciphertexts. That is to say, on input a ciphertext c, SimExt determines if it was obtained by transforming some ciphertext c' previously generated using SimEnc; if so, SimExt outputs the corresponding transformation T.

The game between the challenger and the adversary in the definition of security is somewhat different from that in the definition by Prabhakaran and Rosulek. Specifically, we do not have a single challenge ciphertext c^*; instead, the adversary has access to encryption and decryption oracles. Intuitively, for our definition we would like to say that an adversary cannot distinguish between two worlds: the real world in which it is given access to honest encryption and decryption oracles, and an ideal world in which it is given access to an ideal encryption oracle (which outputs ciphertexts containing no information about the queried message) and a decryption oracle that outputs a special answer for ciphertexts derived from the ideal ciphertexts (by using SimExt to track such ciphertexts) and honestly decrypts otherwise.

Let us consider transformations more closely. Recall that, for proofs of language membership, a transformation $T \in \mathcal{T}$ consists of a pair of transformations (T_x, T_w), where T_x acts on the instances, and T_w on the witnesses. What is the analogue for ciphertexts? A legal transformation T_x on a ciphertext implies some legal transformation T_m on an underlying message and a corresponding transformation T_r on the underlying randomness. Thus, here we view transformations as tuples $T = (T_x, (T_m, T_r))$, where T_x acts on the ciphertexts, T_m acts on the plaintexts, and T_r acts on the randomness.

In the full version of our paper [15], we relate CM-CCA security to CCA, RCCA and HCCA security. Specifically, we show that (1) when the class of allowed transformation $\mathcal{T}$ is the empty set, CM-CCA implies regular CCA security; (2) when the class of allowed transformations is as follows: $T \in \mathcal{T}$ if $T = (T_x, (T_m, T_r))$ where T_m is the identity transformation, then CM-CCA security implies RCCA security; (3) in more general cases we show that it implies the notion of targeted malleability introduced by Boneh et al. [12]; in addition, we show that our notion satisfies the UC definition given by Prabhakaran and Rosulek, so that it captures the desired HCCA security goals, even if it does not satisfy their definition of HCCA security (which is in fact a stronger notion).

Finally, because our cm-NIZK is malleable only with respect to unary transformations, we inherit the limitation that our encryption scheme is malleable only with respect to unary transformations as well; as our security definition is closely related to HCCA security and Prabharakan and Rosulek in fact prove HCCA security (combined with unlinkability) is impossible with respect to binary transformations, this is perhaps not surprising.

Definition 5.1. *For an encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*, a class of transformations* $\mathcal{T}$*, an 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: first* $(pk, sk) \xleftarrow{\$} K(1^k)$*, and next* $b' \xleftarrow{\$} \mathcal{A}^{E_{pk}(\cdot), D_{sk}(\cdot)}(pk)$*, where* (K, E, D) *are defined as* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *if* $b = 0$*, and the following algorithms (defined for a state set* $Q = Q_m \times Q_c = \{(m_i, c_i)\}$*) if* $b = 1$*:*

Procedure $K(1^k)$	*Procedure* $E(pk, m)$	*Procedure* $D(sk, c)$
(pk, sk, τ_1, τ_2)	$c \xleftarrow{\$} \mathsf{SimEnc}(pk, \tau_1)$	$(c', T) \leftarrow \mathsf{SimExt}(sk, \tau_2, c)$
$\xleftarrow{\$} \mathsf{SimKeyGen}(1^k)$	*add* (m, c) *to* Q	*if* $\exists i$ *s.t.* $c' = c_i \in Q_c$ *and* $T \neq \bot$
return pk	*return* c	*return* $T_m(m_i)$
		else
		return $\mathsf{Dec}(sk, c)$

We say that the encryption scheme is controlled-malleable-CCA secure *(or CM-CCA secure for short) if there exist PPT algorithms* $\mathsf{SimKeyGen}$*,* SimEnc*, and* SimExt *as used above such that for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that* $|p_0^{\mathcal{A}}(k) - p_1^{\mathcal{A}}(k)| < \nu(k)$*.*

As mentioned earlier, we can obtain an encryption scheme that achieves this notion of security; we do this by combining a cm-NIZK $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V})$ for the relation R such that $((pk, c), (m, r)) \in R$ iff $c := \mathsf{Enc}'(pk, m; r)$ and an IND-CPA-secure encryption scheme $(\mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$. Due to space constraints, our construction and efficient instantiation of this scheme can be found in the full version of our paper [15].

6 Compactly Proving Correctness of a Shuffle

As described in the introduction, we achieve a notion of verifiability for shuffles that does not require each mix server to output its own proof of correctness;

instead, using the malleability of our proofs, each mix server can maul the proof of the previous one. One point that it is important to keep in mind with this approach is that the soundness of the scheme does not follow directly from the soundness of each of the individual proofs anymore; instead, one proof must somehow suffice to prove the validity of the entire series of shuffles, yet still remain compact. To capture this requirement, we define a new notion for the security of a shuffle, that we call *compact verifiability*.

To define our notion, we assume that a verifiable shuffle consists of three algorithms: a Setup algorithm that outputs the parameters for the shuffle and the identifying public keys for the honest mix servers, a $\mathsf{Shuffle}$ algorithm that takes in a set of ciphertexts and outputs both a shuffle of these ciphertexts and a proof that the shuffle was done properly, and finally a Verify algorithm that checks the validity of the proofs.

In our definition, the adversary is given the public keys of all the honest shuffling authorities, as well as an honestly generated public key for the encryption scheme. It can then provide a list of ciphertexts and ask that they be shuffled by one of the honest authorities (we call this an initial shuffle), or provide a set of input ciphertexts, a set of shuffled ciphertexts, and a proof, and ask one of the honest authorities to shuffle the ciphertexts again and update the proof. Finally, the adversary produces challenge values consisting of a set of input ciphertexts, a set of shuffled ciphertexts and a proof that includes the public key of at least one of the honest authorities. If this proof verifies, it receives either the decryption of the shuffled ciphertexts, or a random permutation of the decryptions of the initial ciphertexts. Our definition requires that it should be hard for the adversary to distinguish which of the two it is given.

We also require that the input ciphertexts are always accompanied by a proof that they are well-formed; i.e., a proof of knowledge of a valid message and the randomness used in encryption. This is usually necessary in many applications (for example in voting when each voter must prove that he has encrypted a valid vote), and in our construction it means that we can easily handle an adversary who produces the input ciphertexts in invalid ways; e.g., by mauling ciphertexts from a previous shuffle, or by submitting malformed ciphertexts.

Definition 6.1. *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'.*

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

Our compactly-verifiable shuffle construction will utilize four building blocks: a *hard relation R_{pk}* (as defined by Damgård [18, Definition 3]), a re-randomizable IND-CPA-secure encryption scheme ($\mathsf{KeyGen}, \mathsf{ReRand}, \mathsf{Enc}, \mathsf{Dec}$), a proof of knowledge ($\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V}$), and a cm-NIZK ($\mathsf{CRSSetup}', \mathcal{P}', \mathcal{V}'$). The hard relation will be used to ensure that the secret key sk_j known to the j-th mix server cannot be derived from its public key pk_j,[11] the proof of knowledge will be created by the users performing the initial encryptions to prove knowledge of their votes, and the cm-NIZK will be used to prove that a given collection $\{c_i'\}$ is a valid shuffle of a collection $\{c_i\}$, performed by the mix servers corresponding to a set of public keys $\{pk_j\}$. This means that the instances are of the form $x = (pk, \{c_i\}, \{c_i'\}, \{pk_j\})$, witnesses are of the form $w = (\varphi, \{r_i\}, \{sk_j\})$ (where φ is the permutation used, $\{r_i\}$ is the randomness used to re-randomize the ciphertexts, and $\{sk_j\}$ are the secret keys corresponding to $\{pk_j\}$), and the relation R is $((pk, \{c_i\}, \{c_i'\}, \{pk_j\}_{i=1}^{\ell'}), (\varphi, \{r_i\}, \{sk_j\})) \in R$ iff $\{c_i'\} = \{\mathsf{ReRand}(pk, \varphi(c_i); r_i)\} \wedge (pk_j, sk_j) \in R_{pk}\ \forall j \in [1..\ell']$. The valid transformations are then $T_{(\varphi', \{r_i'\}, \{sk_j^+, pk_j^+\}, \{pk_j^-\})} = (T_x, T_w)$, where $T_x(pk, \{c_i\}, \{c_i'\}, \{pk_j\}) := (pk, \{c_i\}, \{\mathsf{ReRand}(pk, \varphi'(c_i); r_i')\}, \{pk_j\} \cup (\{pk_j^+\} \setminus \{pk_j^-\}))$ and T_w transforms the witness accordingly. Due to space constraints, a formal outline of how these primitives are combined can be found in the full version of our paper, along with a proof of the following theorem:

Theorem 6.1. *If the encryption scheme is re-randomizable and IND-CPA secure, R_{pk} is a hard relation, the proofs π_i are NIZKPoKs, and the proof π is a cm-NIZK, then the above construction gives a compactly verifiable shuffle.*

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. Abe. Universally verifiable mix-net with verification work indendent of the number of mix-servers. In *Proceedings of EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 437–447. Springer, 1998.

[11] It is worth mentioning that generically we can use a one-way function to obtain this property, but that we cannot efficiently instantiate this in our setting and so use instead a hard relation (for more on this see the full version of our paper).

2. M. Abe. Mix-networks on permutation networks. In *Proceedings of Asiacrypt 1999*, pages 258–273, 1999.
3. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In *Proceedings of Crypto 2010*, volume 6223 of *LNCS*, pages 209–236, 2010.
4. M. Abe, J. Groth, K. Haralambiev, and M. Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In *Proceedings of Crypto 2011*, volume 6841 of *LNCS*, pages 649–666, 2011.
5. M. Abe, K. Haralambiev, and M. Ohkubo. Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive, Report 2010/133, 2010. `http://eprint.iacr.org/2010/133`.
6. M. Abe and F. Hoshino. Remarks on mix-networks based on permutation networks. In *Proceedings of PKC 2001*, pages 317–324, 2001.
7. T. Acar and L. Nguyen. Revocation for delegatable anonymous credentials. In *Proceedings of PKC 2011*, pages 423–440, 2011.
8. 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.
9. M. Bellare and O. Goldreich. On defining proofs of knowledge. In *Proceedings of Crypto 1992*, volume 740 of *LNCS*, pages 390–420. Springer-Verlag, 1992.
10. M. Blum, A. de Santis, S. Micali, and G. Persiano. Non-interactive zero-knowledge. *SIAM Journal of Computing*, 20(6):1084–1118, 1991.
11. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 41–55. Springer-Verlag, 2004.
12. D. Boneh, G. Segev, and B. Waters. Targeted malleability: homomorphic encryption for restricted computations. In *Proceedings of ITCS 2012*, 2012. To appear.
13. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Proceedings of Crypto 2011*, pages 505–524, 2011.
14. M. Chase and M. Kohlweiss. A domain transformations for structure-preserving signatures on group elements. Cryptology ePrint Archive, Report 2011/342, 2011. `http://eprint.iacr.org/2011/342`.
15. M. Chase, M. Kohlweiss, A. Lysyanskaya, and S. Meiklejohn. Malleable proof systems and applications. Cryptology ePrint Archive, Report 2012/012, 2012. `http://eprint.iacr.org/2012/012`.
16. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.
17. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Proceedings of Crypto 1998*, pages 13–25, 1998.
18. I. Damgård. On sigma protocols. `http://www.daimi.au.dk/~ivan/Sigma.pdf`.
19. 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.
20. Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Cryptography against continuous memory attacks. In *Proceedings of FOCS 2010*, pages 511–520, 2010.
21. 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.
22. U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of STOC 1990*, pages 416–426, 1990.

23. 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.
24. G. Fuchsbauer. Commuting signatures and verifiable encryption. In *Proceedings of Eurocrypt 2011*, pages 224–245, 2011.
25. J. Furukawa. Efficient and verifiable shuffling and shuffle-decryption. *IEICE Transactions on Fundamentals of Electronic, Communications and Computer Science*, 88(1):172–188, 2005.
26. J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *Proceedings of Crypto 2001*, pages 368–387, 2001.
27. C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of STOC 2009*, pages 169–178, 2009.
28. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *Proceedings of STOC 1985*, pages 186–208, 1985.
29. J. Groth. A verifiable secret shuffle of homomorphic encryptions. In *Proceedings of PKC 2003*, volume 2567 of *LNCS*, pages 145–160. Springer-Verlag, 2003.
30. 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.
31. 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.
32. 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.
33. 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.
34. A. Neff. A verifiable secret shuffle and its applications to e-voting. In *Proceedings of CCS 2001*, pages 116–125, 2001.
35. M. Prabhakaran and M. Rosulek. Rerandomizable RCCA encryption. In *Proceedings of Crypto 2007*, volume 4622 of *LNCS*, pages 517–534. Springer-Verlag, 2007.
36. M. Prabhakaran and M. Rosulek. Homomorphic encryption with CCA security. In *Proceedings of ICALP 2008*, pages 667–678, 2008.
37. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proceedings of FOCS 1999*, pages 543–553, 1999.
38. K. Sako and J. Kilian. Receipt-free mix-type voting scheme. In *Proceedings of Eurocrypt 1995*, volume 921 of *LNCS*, pages 393–403. Springer-Verlag, 1995.
39. N. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Proceedings of PKC 2010*, pages 420–443, 2010.
40. M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proceedings of Eurocrypt 2010*, volume 6110 of *LNCS*, pages 24–43. Springer-Verlag, 2010.
41. D. Wikström. A sender verifiable mix-net and a new proof of a shuffle. In *Proceedings of Asiacrypt 2005*, pages 273–292, 2005.