Malleable SNARKs and Their Applications

Suvradip Chakraborty[1], Dennis Hofheinz[2], Roman Langrehr[2], Jesper Buus Nielsen[3], Christoph Striecks[4], and Daniele Venturi[5]

[1] Visa Research
suvchakr@visa.com
[2] ETH Zurich
{hofheinz,roman.langrehr}@inf.ethz.ch
[3] Aarhus University
jbn@cs.au.dk
[4] AIT Austrian Institute of Technology
Christoph.Striecks@ait.ac.at
[5] Sapienza University of Rome
venturi@di.uniroma1.it

Abstract. Succinct non-interactive arguments of knowledge (SNARKs) are variants of non-interactive zero-knowledge proofs (NIZKs) in which complex statements can be proven in a compact way. SNARKs have had tremendous impact in several areas of cryptography, including verifiable computing, blockchains, and anonymous communication. A recurring concept in many applications is the concept of recursive SNARKs, in which a proof references a previous proof to show an evolved statement.

In this work, we investigate *malleable SNARKs*, a generalization of this concept of recursion. An adaptation of the existing concept of malleable NIZKs, malleable SNARKs allow to modify SNARK proofs to show related statements, but such that such mauled proofs are indistinguishable from "properly generated" fresh proofs of the related statement. We show how to instantiate malleable SNARKs for universal languages and relations, and give a number of applications: the first post-quantum RCCA-secure rerandomizable and updatable encryption schemes, a generic construction of reverse firewalls, and an unlinkable (i.e., computation-hiding) targeted malleable homomorphic encryption scheme.

Technically, our malleable SNARK construction relies on recursive proofs, but with a twist: in order to support the strong indistinguishability properties of mauled and fresh SNARK proofs, we need to allow an unbounded recursion depth. To still allow for a reasonable notion of extractability in this setting (and in particular to guarantee that extraction eventually finishes with a "proper" witness that does not refer to a previous SNARK proof), we rely on a new and generic computational primitive called *adversarial one-way function (AOWF)* that may be of independent interest. We give an AOWF candidate and prove it secure in the random oracle model.

1 Introduction

Non-Interactive Zero-Knowledge Proofs. Non-interactive zero-knowledge (NIZK) proofs [10] allow a prover to generate non-interactive proofs for statements without revealing the corresponding witnesses. NIZK proofs and arguments[6] have found numerous applications, ranging from constructing other cryptographic primitives (e.g., [5, 59, 64]) to anonymous credential systems (e.g., [3]). While initial constructions of NIZK proofs or arguments ("NIZKs" in the following) were feasibility results and only asymptotically efficient, we now know very efficient NIZKs for restricted sets of languages, e.g., [46]. Furthermore, "SNARKs"[7] (i.e., succinct non-interactive arguments of knowledge [69, 43, 7]) are a particularly compact and efficient variant of NIZKs, in which the size of the argument is bounded by some fixed polynomial in the security parameter and is essentially independent of the exact (polynomial) size of the statement being proved and its witness.

1.1 Malleable NIZKs

Malleable Non-Interactive Zero-Knowledge. NIZKs can have a number of interesting properties. For instance, they can be proofs or arguments of knowledge [64, 29] (which means that it is possible to extract witnesses from a successful prover or proof), or they can satisfy very strong notions of soundness (like non-malleability or simulation soundness [67], or even simulation extractability [45]). In this work, we are interested in one particular interesting property of NIZKs: malleability [23, 70]. Malleability denotes the property that a proof π for a certain statement X can be transformed into a proof π' for a related statement $X' = T(X)$. Of course, this should only be possible for a restricted class of allowed statement transformations T. We also want to hide X, i.e., the fact that π' has been produced from a related proof π.

Malleable NIZKs are, e.g., useful in shuffling schemes [23, 24], when a proof certifies a property of a set of ciphertexts.[8] Essentially, a malleable NIZK now allows to rerandomize and reorder the ciphertexts, while adapting the NIZK proof suitably. (This can be seen as a special case of "incremental" or "updatable" cryptography [4, 2].)

Malleability As a Form Of Self-Reference. With malleable NIZKs, it is possible to generate proofs for related statements. A related concept is the notion of a "recursive NIZK (or SNARK)", which denotes a new proof that proves knowledge of an old proof (for a related statement). Recursion is a popular tool in the

[6] While *proofs* do not exist for false statements, *arguments* for false statements are only infeasible to find.

[7] With SNARKs, we implicitly mean "zero-knowledge SNARKs", i.e., SNARKs with simulatable proofs.

[8] That proof could certify, e.g., that the ciphertexts contain a number of encrypted votes that have been correctly counted.

construction and usage of SNARKs [69, 14, 8], in particular because with SNARKs, proofs do not grow with the depth of the recursion. In fact, recursion can also be seen as a tool to obtain a form of malleability, in particular when proofs obtained through recursion and "fresh" proofs look alike. For some applications, like reverse firewalls, this can be interesting even for the trivial transformation $T(X) = X$.

Derivation Privacy. In this work, we are particularly concerned with the concept of *derivation privacy* (initially put forward by [23] in the context of malleable NIZKs). In the context of recursion, derivation privacy requires that proofs that have been obtained through recursion (i.e., proofs that prove knowledge of a proof for a related statement) cannot be distinguished from "direct" proofs for that related statement.

Bounding Extraction Depth. Derivation privacy and recursive proofs are at odds with each other. Security of SNARKs requires that you can extract a witness from a short SNARK proof π. In case of recursive SNARKs (in which SNARK proofs may use previous SNARK proofs π' as witnesses), the extraction procedure recursively extracts π'. Every layer of extraction may require another invocation of the basic SNARK extraction assumption, which may result in an exponential blowup in extraction time. But even when making strong assumptions about extraction (such as the existence of a *fast extractor* with an additive, polynomial, extraction overhead), we still must make sure that this recursion terminates. To illustrate, note that for a trivial malleability transformation like $X = T(X)$ there is nothing in the above discussion preventing $\pi' = \pi$. Intuitively, the language definition according to which we verify proofs therefore needs to involve a "recursion level" or "modification counter" ρ ensuring that recursive extraction terminates after ρ iterations. But then π and π' would have different recursion levels and can intuitively be distinguished, violating derivation privacy.

1.2 Our Results

Warmup: a construction with limited derivation privacy. As a warmup, we recall a simple construction of malleable SNARKs in which mauled proofs do *not* look like fresh proofs. This construction is essentially the same as the SNARK construction in [24], and captures ideas implicit in previous works on recursive SNARKs (e.g., [14]). This construction avoids unbounded recursions through an explicit (and visible) counter in proofs. This counter starts with 0 in fresh proofs and gets incremented each time a related proof is produced. This construction guarantees that extraction eventually ends up with a "proper" witness (i.e., one which does not lead to recursive extraction), but also only satisfies a "leveled" form of derivation privacy (in which proofs of the same "derivation depth" are indistinguishable).

New Assumption: Adversarial One-Way Functions. To make progress and achieve full derivation privacy, we introduce a new computational assumption called

"adversarial one-way functions (AOWFs)". Intuitively, an AOWF f cannot be efficiently inverted "many times in a row" on an *adversarially generated* image y. A little simplified, we want that

- for every polynomial t and every adversary $\mathcal{A}_1$ that has only a limited amount of time t to pick an image y,
- there should be a number $p(t)$ (for a fixed polynomial p),
- such that every adversary $\mathcal{A}_2$ that gets $\mathcal{A}_1$'s state and y as input, has negligible success in finding a preimage x with $f^{p(t)}(x) = y$.

We also require a technical property called "unlinkability" which we do not discuss here. Notice that we need to restrict $\mathcal{A}_1$'s time in this assumption. Otherwise, $\mathcal{A}_1$ could simply pick an arbitrary x, compute $y := f^{p(t)}(x)$, and leave x as part of its state for $\mathcal{A}_2$. Intuitively, the assumption is that this is essentially the only attack.

It is also helpful to observe that, of course, arbitrarily long chains of f-evaluations $y_1, \ldots, y_n$ (with $y_{i+1} = f(y_i)$) exist. However, for every image y that an adversary comes up with in time t, it is infeasible to find such a chain $y_1, \ldots, y_n$ with $y_n = y$ of length $n \geq p(t)$. This last characterization will be the key for our construction of malleable and derivation-private SNARKs.[9]

We do not know how to achieve the AOWF definition from generic assumptions (such as, say, the existence of one-way functions). However, we present a plausible candidate for an AOWF based on hash-functions that we can prove secure in the random oracle model. The necessary techniques for AOWFs seem also very related to time-lock puzzles [66], proofs of work [33], and verifiable delay functions [11], in the sense that we also assume that certain computations (e.g., function inversions) require a certain amount of time. Our definition is more restrictive, however, since we allow even an adversarial choice of problem instance (y above), and the additional property of "unlinkability" we discuss in the technical overview. Both distinctions are crucial for obtaining derivation-privacy and our applications, and appear to require stronger (and, in fact, non-falsifiable, see Appendix F) assumptions.

From AOWFs to malleable SNARKs. For our construction of malleable SNARKs, we use AOWFs to bound the depth of any recursive extraction chain. Concretely, we add a fresh AOWF image $y = f(x)$ (for a freshly chosen preimage x) to any freshly generated SNARK proof π for a statement X. In case we generate a mauled proof π' for a modified statement X' from a pair (X, π), however, we add $f(y)$ (for the image y from π) to π' and use π' to prove knowledge of π. In other words, every proof carries an AOWF image that is developed (using f) upon every proof modification. One can think of y as an obfuscated modification counter, which can be incremented without knowing its value, and which cannot adversarially be brought to a high value without spending the corresponding time.

[9] We also generalize AOWFs to evaluation *chains* to *graphs*, which will be crucial for one of our applications.

When recursively extracting from a given proof π, this means we also extract (step by step) a chain $y_1, \ldots, y_n$ of f-images as above. By our AOWF assumption, the length of this chain is bounded by a polynomial $p(t)$ (where t is the time it took to produce π). Hence, extraction must terminate at some point with a "proper" witness.

Applications of malleable SNARKs. Our notion of malleable SNARKs has a number of applications. In particular, we show that malleable SNARKs imply the following:

Reverse Firewalls. The first reverse firewall (RF) [57] for NIZKs or SNARKs; this improves over a previous result by Ganesh, Magri and Venturi [40] who gave RFs for *interactive* proof systems supporting only a restricted class of "malleable Σ-protocols". An RF for a SNARK is an untrusted external entity that sits between the prover and the verifier, and that sanitizes the proof produced by the prover by eliminating any subliminal channel due to the fact that the prover's machine has been subverted (in an undetectable manner).

Rerandomizable Public-Key Encryption. We also use malleable SNARKs to give the first generic construction (and in particular the first one based on post-quantum assumptions) of rerandomizable public-key encryption (RPKE) schemes that are secure in a chosen-ciphertext sense (i.e., RCCA secure [16, 44]). RPKE schemes are PKE schemes whose ciphertexts can be publicly rerandomized (such that they look like fresh encryptions of the corresponding message). RCCA-secure RPKE schemes have numerous applications (in particular in anonymous communication [44, 63, 61]) and were previously known to exist based on group-related complexity assumptions (e.g., [44, 63, 37]), but not from post-quantum assumptions. We note that derivation-privacy of our SNARK is essential in this construction, since it guarantees that rerandomized ciphertexts appear indistinguishable from fresh ones.

Updatable Encryption. Updatable encryption (UE [13]) is a form of (symmetric) encryption that allows to periodically change (or "update") the used secret key and the associated ciphertexts using a special "update token". In its strongest form, UE enables a form of forward-secure communication (e.g., [68]) and also has applications in anonymous communication (e.g., [50]). In this work, we use malleable SNARKs to construct the first *rerandomizable* UE scheme that satisfies a strong, chosen-ciphertext UE security notion yielding the first post-quantum secure UE scheme as a special case (answering an open question posed in prior work [39] in the affirmative). Again, derivation privacy is critical here for rerandomizability.

Targeted Malleability. Targeted malleability [14] is a property of homomorphic encryption schemes that restricts homomorphic computations to a restricted set of possible computations. [14] already present a SNARK-based construction of targeted malleable homomorphic encryption, which however may reveal which (and how many) computations have been performed with a given ciphertext. Plugging in our malleable SNARK into their construction yields a targeted malleable homomorphic encryption scheme which *does* hide

which computations have been performed (thanks to derivation-privacy). This has been explicitly stated as an open problem in [14].

1.3 Technical overview

Adversarial OWFs. The core building block of our malleable SNARK construction is the notion of an adversarial one-way function (AOWF). Recall that in the simplified version outlined above, an AOWF f satisfies that for every pair of adversaries $(\mathcal{A}_1, \mathcal{A}_2)$, and

- for $(y, \mathsf{st}) \leftarrow \mathcal{A}_1$ computed by $\mathcal{A}_1$ in time t,
- and for $x \leftarrow \mathcal{A}_2(y, \mathsf{st})$, we have $f^{p(t)}(x) = y$ only negligibly often, where p is a fixed polynomial.

This AOWF requirement can be viewed as a limitation to quickly produce f-images that can be inverted many times. Such f-images will serve as "tags" in SNARK proofs which are developed (by applying f) every time a proof is referenced in another SNARK proof. The above condition then ensures that any chain of SNARK proofs generated through recursive extraction is at most polynomially long.

A simple AOWF candidate (with $p(t) = t + 1$) is $f(x) = H(x)$ for a hash function H. This property is in fact easy to prove in the random oracle model, assuming that a random oracle query to H costs at least one computation step. However, for our purposes, we will require a slightly different form of AOWF. In fact, our previously mentioned "derivation privacy" of a SNARK property states that SNARK proofs π' that have been generated by modifying other proofs π should be indistinguishable from SNARK proofs π' directly generated from a witness.

This "derivation privacy" property should (for some of our applications) even hold when both π and π' are known. Since $f(x)$ will be part of the corresponding proof, this means that π contains $y = f(x)$ (for some x), and π' contains $y' = f(y)$ *if and only if* π' was generated from π. This contradicts derivation privacy, and in fact eliminates any deterministic function f as a "tag generator".

We will resolve this with a slightly different AOWF definition that allows for *randomized functions* (modeled by functions f that get as input x and random coins r). The AOWF condition below is essentially the same as above, but adapted to randomized functions. Formally, we require that for all $(\mathcal{A}_1, \mathcal{A}_2)$ and

- for $(y, \mathsf{st}) \leftarrow \mathcal{A}_1$ computed by $\mathcal{A}_1$ in time t,
- and for $(x, r_1, \ldots, r_{p(t)}) \leftarrow \mathcal{A}_2(y, \mathsf{st})$, we have

$$f^{p(t)}(x, r_1, \ldots, r_{p(t)}) = f(\cdots f(f(x, r_1), r_2) \ldots, r_{p(t)}) = y$$

only negligibly often, where p is a fixed polynomial.

In other words, it is the task of $\mathcal{A}_2$ to output the corresponding random coins used in the $p(t)$ f-evaluations.

Additionally, we explicitly require *unlinkability*. This property states that given an (even adversarially generated) preimage x, the corresponding image $y = f(x, r)$ (for random, unknown r) looks like a random image y. Unlinkability implies derivation privacy when AOWF images are used as tags in SNARK proofs. Furthermore, we show that $f(x, r) = H(x, r)$, a simple randomized variant of the hash-based deterministic AOWF above is an (unlinkable) AOWF in the random oracle model.

Generalized AOWFs. AOWFs as informally defined above allow an extraction "along a path" of preimages $x_1, x_2 := f(x_1, r_1), \ldots, x_n := f(x_{n-1}, r_{n-1})$ (starting with x_n). We in fact define (and prove) a more generalized variant of AOWFs which allow a more flexible extraction. Specifically, we also consider AOWFs with larger a larger number of (x-)inputs, such that we can express not only paths (as above), but trees and in fact more general graphs of f-evaluations. Adversarial one-wayness then requires that $\mathcal{A}_2(y, \mathsf{st})$ (with y, st as above) cannot output a *graph* G exceeding size $p(t)$ with preimages x at leaves, and randomness values r at inner nodes, such that the computation induced by G yields y.

Informally, hence, adversarial one-wayness states that computing the result of any such a "topological computation" requires a certain amount of time. Jumping ahead, this translates to SNARKs in which newly computed SNARKs may depend on more than one given SNARK.

Malleable SNARKs. Before we describe our main construction with adversarial one-way functions, we explain a simpler construction that uses instead a counter ℓ counting the number of transformations applied so far. This approach only requires a SNARK to begin with, and is essentially the one presented in [24] (and implicit in [14]).[10] This construction has two drawbacks that our main construction will overcome: during set-up, we have to fix a bound B on the number of transformations that can be applied[11], and the construction only achieves a weaker, "leveled" form of derivation privacy, where the number of applied transformations is allowed to leak.

These counters then become part of the SNARK statement. More concretely, to build a malleable SNARK for an NP relation $\mathcal{R}$ and a set of transformations $\mathcal{T}$ we use a SNARK for statements of the format (ℓ, x) (where ℓ is the counter and x a purported statement in $\mathcal{R}$). Each transformation $T \in \mathcal{T}$ has the format $T = (T_x, T_w)$ and has to fulfill $(x, w) \in \mathcal{R} \implies (T_x(x), T_w(w)) \in \mathcal{R}$. The SNARK then allows the prover to prove statements of the form:

> *Either $\ell = 0$ and I know a witness for x, or $\ell \in [B]$ and I know a transformation $(T_x, T_w) \in \mathcal{T}$, a statement x' and a SNARK proof π' such that $x = T_x(x')$ and π' is valid proof for the statement $(\ell - 1, x')$.*

[10] Of course, we do not claim novelty for this counter-based construction. We detail this construction here to pave the way for our main SNARK construction.

[11] This can also be beneficial, for example for lattice-based encryption schemes, each transformation might increase the noise level of a ciphertext. Bounding the number of transformations can ensure that a certain noise-level is not exceeded.

To support the verification of a SNARK in the SNARK language, we need a fully-succinct SNARK for which the common reference string (CRS) does not grow with the description size of the SNARK language (since verification requires the CRS and thus the CRS has to be part of the language description).[12]

We can now transform a proof for the statement x to a proof of the statement $T(x)$ by generating a new SNARK proof and prove the right-hand side of the or-language described above. Zero-knowledge and leveled derivation privacy follow from the zero-knowledge property of the underlying SNARK.

To argue knowledge soundness in the case $\ell = 0$, we can just use the knowledge soundness property of the SNARK. This guarantees that there exists an extractor (depending on the adversary) that extracts the witness w from the adversary's inputs and random tape. In the case $\ell \geq 1$, this SNARK extractor will not give us a witness, but a transformation (T_x, T_w) and (x', π') with $x = T_x(x')$. In this case, we define a new adversary that runs this extractor and outputs $((\ell - 1, x'), \pi')$, which is another valid statement-proof pair for the SNARK. We can then apply the SNARK extractor to this statement and repeat this process until we end in the base case $\ell = 0$. If this recursive extraction outputs a witness w' for x', we can output $T_w(w')$ as witness for x. The counter guarantees that this will take at most B recursions. If B is a constant, we also have the guarantee that all these extractors run in polynomial time. If we want to allow any bound B polynomial in the security parameter, we have to assume fast extraction (meaning that the extractor for an adversary running in time t takes only time $t + \mathsf{poly}(\lambda)$ for a polynomial poly independent of the adversary) to avoid an exponential blow-up in the runtime.

We now come to our main SNARK construction, and show how we can replace the counters above with an adversarial one-way function to get full derivation privacy and remove the bound on the number of applicable transformations. For simplicity, we state this construction with the simplified variant of AOWFs that only take a single x (and r) as input. In the main body, we however prove a construction with generalized AOWFs that enables more general transformations that take *several* x (and w) as input.

In our (simplified) construction, the SNARK is for statements of the form (ξ, x) and allows to prove statements of the form

> *Either I know a witness for x, or I know ξ', r with $f(\xi', r) = \xi$ and a transformation $(T_x, T_w) \in \mathcal{T}$, a statement x' and a SNARK proof π' such that $x = T_x(x')$ and π' is valid proof for the statement (ξ', x').*

Initially, ξ is sampled uniformly at random. In this setting the Turing machine verifying the NP-relation for the SNARK also needs its own description. We solve this without using an unproven, efficient version of the recursion theorem like previous works do [8].

Zero-knowledge of this construction follows directly from zero-knowledge of the underlying SNARK. To argue derivation privacy, we additionally need the

[12] When we use our construction with counters only for a constant number of levels, we can also use a separate CRS for every level.

unlinkability of the AOWF. (Essentially, unlinkability guarantees that the value ξ in a proof does not reveal anything about its preimage ξ', and hence proofs do not reveal anything about the proofs they were potentially derived from.)

We can argue knowledge soundness with a construction analogous to the one with counters. The adversarial one-wayness guarantees that we reach the base case after at most $p(t)$ recursions, where t is the run-time of the adversary. Otherwise, we could use the adversary as first stage for the adversarial one-way function game. The second stage then runs the first $p(t)$ extractors to obtain the values $(x, r_1, \ldots, r_{p(t)})$ to win the game. Here, we have to assume fast extraction to avoid an exponential blow-up in the runtime.

If the underlying SNARK has simulation extractability, both of our constructions also achieve controlled-malleable simulation extractability, which guarantees that the adversary can only prove true statements or statements obtained by applying transformations to statements for which it queried a simulated proof.

Application: Reverse Firewalls. Reverse firewalls were introduced by Mironov and Stephens-Davidowitz [57] as a method for protecting cryptographic protocols against subversion attacks. In a subversion attack, an adversary tampers with the machines of the honest parties and leaks the honest parties' secrets through the protocol transcript. A reverse firewall (RF) is an external device that sits between a party P and the external world in order to "sanitize" the messages that are sent and received by P. The parties do not trust the firewall (and hence do not share its secret with the firewall), and while the RF itself cannot create security, the hope is for the RF to preserve security in the face of subversion. Roughly, the security properties desired from an RF are: (i) *exfiltration-resistance*: the firewall prevents the machine from leaking any information to the outside world regardless of how the user's machine behaves; (ii) *security preservation*: the protocol with the firewall is secure even when honest parties' machines are tampered. Similar to most prior works in this area, we consider the adversarial tamperings to be "functionality-maintaining", such that tampered machines still maintain the completeness of the protocol (but try to exfiltrate sensitive information by establishing covert channels).[13]

All known constructions of protocols in the RF framework [57, 30, 40, 18, 19, 22, 20, 21] rely on the malleability of the underlying operations in order for the RF to rerandomize/sanitize the transcript. Thus the existing RFs are limited to protocols that offer some structure. Therefore, in this work we ask whether this is inherent to all RF constructions. In other words, can we provide generic constructions of RFs that do not rely on specific algebraic structures of the underlying protocols? We answer this in the positive for NIZK proof systems. The previous works on RF-compatible zero-knowledge protocols [40, 18] exploit malleability properties of the underlying ZK protocols (the work of [40] relies on malleable sigma protocols, whereas [18] rely on controlled malleable NIZKs) to sanitize/rerandomize the ZK protocols.

[13] This class of tampering attacks cover most of the practical subversion attacks where the adversary attempts to stay undetectable.

In this work, we use our malleable SNARKs (NIZKs) satisfying (strong) derivation privacy in a straightforward way to construct RF for NIZK argument systems. In particular, the RF (for the prover) receives a tuple (x, π) from the prover, where x is a statement and π is a purported proof of membership of x in some language $\mathcal{L}$.[14] The RF then uses our malleable NIZK to generate a proof π' for the statement: *"Either I know a witness for x, or I know a NIZK proof π such that π is valid proof for the statement x."*[15] The RF then outputs the tuple (x, π') to the outside world. Additionally, thanks to the (recursive) malleability of our underlying NIZK, we can let the firewalls be *stackable*, so that one party may have arbitrarily many firewalls. Exfiltration-resistance of the prover (against the verifer) follows from the derivation privacy of our underlying malleable NIZK which ensures that the resulting proof π' for x looks like a fresh (and honestly generated) proof for the statement x.

Application: Rerandomizable Public-Key Encryption. We present a variant of the Naor–Yung transformation [59, 32] that turns an IND-CPA secure rerandomizable public-key encryption (PKE) into an IND-RCCA secure one. IND-RCCA security [16] guarantees indistinguishability of ciphertexts given access to a decryption oracle that decrypts every ciphertext, but returns a special symbol $\diamond$ when the message is one of the two messages used for the challenge oracle ($\mathsf{m}_0^\star$ or $\mathsf{m}_1^\star$). For a rerandomizable PKE, we can not provide a more powerful decryption oracle, since otherwise an adversary can trivially win by submitting a rerandomizable challenge ciphertext.

The idea of the Naor–Yung transformation is to encrypt each message under two different public keys and equip them with a simulation sound NIZK that both ciphertexts encrypt the same message. The reduction can now simulate the decryption oracle using one of the two secret keys while changing which of the two challenge messages is encrypted under the other public key.

For a rerandomizable PKE, we replace that NIZK with a malleable NIZK for the transformations that rerandomize of both ciphertexts. Since a malleable NIZK cannot be simulation sound (an adversary can apply transformations to the simulated proofs), we have to argue differently that the decryption oracle can still be simulated with each of the two secret keys. We use instead the weaker notion of controlled-malleable simulation soundness. This guarantees that the adversary can only produce proofs for true statements and statements obtainable by applying transformations to simulated statements. Our construction exactly gives controlled malleability. In the context of Naor–Yung, where we use in the reduction simulated proofs for pairs of ciphertexts where one of them encrypts $\mathsf{m}_0^\star$ and the other $\mathsf{m}_1^\star$ and all the allowed transformations preserve the message, we can thus still safely return $\diamond$ whenever we encounter $\mathsf{m}_0^\star$ or $\mathsf{m}_1^\star$ during the decryption of one of them.

[14] Note that the proof need not be generated honestly. However, it must be that the proof π verifies with respect to x if $x \in \mathcal{L}$ (since the tampering is functionality-maintaining).

[15] Note that, in this case the transformation (T_x, T_w) is the identity transformation.

Furthermore, we have to show that a transformed NIZK is indistinguishable from a freshly generated NIZK to guarantee that our PKE scheme is still rerandomizable. On a high level, this follows from the derivation privacy of the malleable NIZK. However, this holds only computationally and, as far as we are aware, previous works only considered statistical rerandomizablity. We propose a game-based definition of rerandomizablity and show that our construction satisfies it.

A similar result has been shown in [23]. However, they use a variant of Naor–Yung that requires a NIZK proof of knowledge (NIZKPoK) which makes it incompatible with our SNARK-based construction. Our SNARK-based construction only achieves a "white-box extractability" notion, where the extractor depends on the adversary and extracts a witness from the adversary's random tape. The Naor–Yung transformation used in [23] however requires a universal extractor that extracts the witness from the proof using a trapdoor for the CRS. This allows to answer many decryption queries during the game. The SNARK-style extraction does not allow this. One can not nest the extractors arbitrary often, since our malleable SNARK does not have the fast-extraction property described above even if the underlying SNARK has it. So we cannot simply plug our SNARK into [23]. Even more so because [23] does not show that the resulting scheme is still *re*randomizable. On the other hand, the result of [23] is more general for controlled-malleable *CCA* security, which captures for example CCA-security variants for homomorphic encryption. We believe that our approach can also be generalized in this way.

Application: Updatable Encryption. Updatable encryption (UE), first defined in [13], is a symmetric primitive that allows rotating secret keys and updating ciphertexts in a discrete-epoch-based fashion. More concretely, during the rotation of a key in epoch e, a fresh key and a so-called update token are generated for epoch $e+1$ while the epoch-e key is discarded. The update token can be used to update existing (e.g., outsourced) ciphertexts under the epoch-e key to valid ciphertexts under the rotated key in epoch $e+1$. UE has several applications, particularly, in the realm of cloud storage and anonymous routing.

A rich body of UE constructions is known from the literature where UE schemes can be either ciphertext-dependent [13, 35, 12, 26, 25] or ciphertext-independent [51, 49, 15, 47, 52, 36, 60, 39, 56, 68, 55]. Latter schemes are the most versatile ones where during the key rotation no access to any ciphertext parts is required.[16]

Moreover, ciphertext updates can be deterministic or randomized. UE schemes with randomized updates yield strong security properties, but are significantly harder to prove secure. This circumstance resulted in only *one* known randomized-update scheme which is secure under chosen-ciphertext attacks [49], but only against rather weak adversaries that are not allowed to query certain update tokens.

[16] In practice, such a feature plays a significant role [48].

The currently strongest models with randomized updates allowing the adversary to query almost all update tokens [60, 39, 56, 68] allow the construction of UE schemes that are secure under chosen-plaintext attacks. A recent work [68] enhanced the common epoch-based UE security models with so-called expiry epochs, allowing that forward security can be achieved in UE and potentially all tokens can be leaked to an adversary. (The security model in [68] implies the randomized model used in [60, 39, 56] by setting the expiry epoch to a global fixed constant 2^λ for security parameter λ; loosely speaking, it means that ciphertexts never expire.) However, stronger security in terms of indistinguishably under randomized chosen-ciphertext attacks was left unaddressed until now.

In this work, we give the first randomized UE scheme under chosen-ciphertext attacks by combining the CPA-secure construction from [68] with our malleable NIZK approach via the Naor–Yung paradigm. As a byproduct, this yields the first post-quantum RCCA-secure UE scheme under randomized updates from the learning-with-errors (LWE) assumptions; i.e., when the CPA scheme in the above transformation is instantiated with the LWE-based schemes from [60, 39] and the expiry epoch is set to 2^λ. This answers an open question posed in [39] in the affirmative.

Concretely, in the vein of Naor–Yung, we generate two independent initial keys $\mathsf{K}_{1,1}, \mathsf{K}_{1,2}$ and a CRS crs using the key and CRS generation algorithms of the UE and of the malleable NIZK schemes, respectively. The keys in epoch e are rotated independently to the next epoch $e+1$ (via key rotation of the UE scheme) while the CRS stays constant.

Encryption in epoch e generates a ciphertext under a message m for each key together with a proof from the malleable NIZK which shows that the same message is encrypted in both ciphertexts. The ciphertext update runs the update algorithms of the UE scheme on the respective token-ciphertext pair as well as the proof evaluation showing that those ciphertexts are updated correctly. The updated ciphertexts with the proof are returned and yield a valid ciphertext in epoch $e+1$. Decryption verifies if and only if the ciphertexts under the keys and the proof are consistent and returns the message m.

The security proof is mostly similar to the proof of the randomized PKE and, eventually, we arrive at a UE scheme with the desired properties.

Application: Targeted Malleable Homomorphic Encryption. It is well-known that a homomorphic (public-key) encryption (HE) cannot achieve IND-CCA security. However, the weaker IND-CCA1 security notion which only gives the adversary a decryption oracle *before* the challenge query is known to be insufficient for many applications [38, 53]. The first work that introduces a security notion beyond IND-CCA1 that is achievable for HE was [62] with the notion of HCCA security. Their notion is unfortunately unachievable for non-unary functionalities (like addition or multiplication of encrypted values). Another notion called targeted non-malleability (TNM-CPA or TNM-CCA1) was introduced in [14]. Furthermore, [1] introduce the notion of FuncCPA security, however that notion seems fairly weak and is not even known to imply IND-CCA1 security. Finally, [23] introduce CM-CCA security and [54] introduce IND-vCCA (and TNM-vCCA) security,

which are stronger than HCCA security, and thus also not achievable for non-unary functionalities.

In this work, we thus consider the notion TNM-CCA1 of [14], which is the strongest known notion that is achievable for general HE. We give a generic conversion of any IND-CPA secure HE for function class $\mathcal{F}$ to a TNM-CCA1 secure HE for any function class $\mathcal{F}' \subseteq \mathcal{F}$ by applying the Naor–Yung paradigm with our malleable NIZK construction. In contrast to [14], who propose a similar construction, our construction preserves *unlinkability* [62], due to the derivation-privacy of our malleable SNARK construction. This solves an open problem of [14], who also give applications of targeted non-malleable HE, for example voting systems based on homomorphic encryption like [28].

We uncover several flaws in the approach of [14], one of them requires an additional assumption about *extractability in the presence of auxiliary input* both in their and our work. As shown in [9], such a notion is not achievable for arbitrary auxiliary input, which makes our construction non-generic. For our transformation, the auxiliary input has to store a public key and a ciphertext of the underlying HE scheme. One approach to build a fully generic transformation for targeted malleable HE is to construct a SNARK for *bounded* auxiliary input (where the parameters of the SNARK can depend on the size of the auxiliary input), which is not ruled out by the result in [9]. However, we are not aware of any SNARK candidate for this security notion.

Applying the Naor–Yung transform to homomorphic encryption using recursive SNARK was also studied in [17]. However, this work only gives a proof sketch that ignores the challenges that arise with the recursive usage of SNARKs.

2 Preliminaries

Notation. For an NP relation $\mathcal{R}$ we use $\mathcal{L}_{\mathcal{R}} := \{x \mid \exists w : (x, w) \in \mathcal{R}\}$ to denote the language associated to $\mathcal{R}$. We assume without loss of generality $\varepsilon \notin \mathcal{L}_{\mathcal{R}}$ to avoid problems with the notion of polynomial runtime for the empty statement.

We use $\mathbb{N}_0$ for the set of natural numbers with zero and $\mathbb{N}_+$ for the set of natural numbers without zero. For $n \in \mathbb{N}_+$ we use $[n] := \{1, \ldots, n\}$.

We use $x \xleftarrow{\$} S$ to denote the process of sampling an element x from a set S uniformly at random. For a (probabilistic) algorithm $\mathcal{A}$ we write $x \xleftarrow{t} \mathcal{A}(b; r)$ to denote the random variable x output by $\mathcal{A}$ on input b with random coins r and the runtime t that $\mathcal{A}(b)$ used for this process. When the exact runtime is not relevant, we omit t. We omit r, if independent uniformly random coins are used. Furthermore $\mathsf{Time}_{\mathcal{A}}(\lambda)$ denotes the worst-case running time of $\mathcal{A}$ on inputs of length λ.

We write $(a, \ldots, b, \_, c, \ldots, d) \in S$ as shorthand for $\exists x.(a, \ldots, b, x, c, \ldots, d) \in S$.

Definition 1 (Non-interactive argument system). *A non-interactive argument system Π for an NP relation $\mathcal{R}$ consists of the following three PPT algorithms:*

- CRSGen *inputs the unary encoded security parameter* 1^λ *and outputs a common reference string* crs*. We assume without loss of generality that* crs *contains* λ*.*
- Prove *inputs the CRS* crs*, a statement* x *and witness* w *and outputs an argument* π*.*
- Verify *is deterministic and inputs the CRS* crs*, a statement* x *and an argument* π *and outputs a bit.*

Every non-interactive argument system should satisfy the following two properties:

Definition 2 (Completeness). *A non-interactive argument system* $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *for NP relation* $\mathcal{R}$ *is* complete *if for all* $(x, w) \in \mathcal{R}$*, for* $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$*,* $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$

$$\Pr[\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1] \geq 1 - \mathsf{negl}(\lambda)$$

for a negligible function negl*. The probability is taken over the random coins of* CRSGen*,* Prove*, and* Verify*.*

Definition 3 (Soundness). *A non-interactive argument system* $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *for NP relation* $\mathcal{R}$ *is* sound *if for all PPT adversaries* $\mathcal{A}$*, for* $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$ *and* $(x, \pi) \xleftarrow{\$} \mathcal{A}(\mathsf{crs})$

$$\Pr[\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \wedge x \notin \mathcal{L}_\mathcal{R}] \leq \mathsf{negl}(\lambda)$$

for a negligible function negl*. The probability is taken over the random coins of* CRSGen*,* $\mathcal{A}$*, and* Verify*.*

A argument system that satisfies the following definition additionally is called a *non-interactive zero-knowledge argument system (NIZK).*

Definition 4 (Zero knowledge). *A non-interactive argument system* $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *for NP relation* $\mathcal{R}$ *is* zero-knowledge *for NP relation* $\mathcal{R}$ *if there exists a PPT simulator* $S = (S_1, S_2)$ *such that for every PPT adversary* $\mathcal{A}$ *we have*

$$\mathsf{Adv}^{\mathsf{zk}}_{\mathcal{A},\Pi}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{zk}}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

for a negligible function negl *where the experiment is defined in Figure 1.*

2.1 SNARKs

For SNARGs and SNARKs we fix the NP relation to be the universal relation

$$\mathcal{R}_\mathcal{U} := \{((M, x, t), w) \mid M \text{ describes a Turing machine accepting the input } (x, w) \text{ in time at most } t.\}$$

$\underline{\mathsf{Exp}^{\mathsf{zk}}_{\mathcal{A},\Pi=(\mathsf{CRSGen},\mathsf{Prove},\mathsf{Verify})}(\lambda)}$:	$\underline{\textsc{Prove}(x,w)}$:
$b \xleftarrow{\$} \{0,1\}$	**if** $(x,w) \notin \mathcal{R}$ **then return** $\bot$
if $b = 0$ **then**	**if** $b = 0$ **then**
$\quad \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$	$\quad$ **return** $\mathsf{Prove}(\mathsf{crs}, x, w)$
else	**else**
$\quad (\mathsf{crs}, \mathsf{td}) \leftarrow S_1(1^\lambda)$	$\quad$ **return** $S_2(\mathsf{crs}, \mathsf{td}, x)$
$b' \leftarrow \mathcal{A}^{\textsc{Prove}(\cdot,\cdot)}(1^\lambda, \mathsf{crs})$	
return $b \stackrel{?}{=} b'$	

Fig. 1. The zero-knowledge experiment for a NIZK for NP relation $\mathcal{R}$.

A fully-succinct non-interactive argument (SNARG) is a non-interactive argument system which satisfies the following succinctness property. We only give a definition of fully-succinct SNARGs, where "full" refers to the fact that the CRS is generated without specifying a bound on the time t and therefore also the size of the CRS is a fixed polynomial in λ.[17] The prover gets the time bound t in unary encoding to allow it to run in time polynomial in t, while the verifier gets it in binary encoding, so that it runs at most polylogarithmically in t.

Definition 5 (Full succinctness). *A non-interactive argument system Π for the NP relation $\mathcal{R}_\mathcal{U}$ is* fully-succinct *if for all* crs *in the image of* $\mathsf{CRSGen}(1^\lambda)$*, all* $((M,x,t),w) \in \mathcal{R}_\mathcal{U}$ *and all arguments* π *in the image of* $\mathsf{Prove}(\mathsf{crs}, (M,x,1^t), w)$ *we have* $|\pi| \leq \mathsf{poly}(\lambda)$ *for a polynomial* poly*. Here* $|\pi|$ *denotes the bit-length of* π*.*

A SNARG that satisfies zero-knowledge is called a zk-SNARG. A succinct non-interactive argument of knowledge (SNARK) is a SNARG that satisfies the following extractability requirement.

Definition 6 (Simulation extractable). *A proof system Π is* simulation extractable *if for every PPT adversary $\mathcal{A}$ there exists a PPT extractor $\mathcal{E}$ such that*

$$\mathsf{Adv}^{\mathsf{sse}}_{\mathcal{A},\mathcal{E},\Pi}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{sse}}_{\mathcal{A},\mathcal{E},\Pi}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

for a negligible function negl *where the experiment is defined in Figure 2.*

We say that a SNARK has fast extraction if the simulation extractability property holds with the additional restriction that $\mathsf{Time}_\mathcal{E}(\lambda) \leq \mathsf{Time}_\mathcal{A}(\lambda) + \mathsf{poly}(\lambda)$ *holds for a polynomial* poly *independent of $\mathcal{A}$. We then say that $\mathcal{E}$ is a fast extractor and Π provides fast extraction.*

A concurrent work by Cheng and Goyal [27] shows that SNARKs with fast extraction are useful to boost SNARKs with non-trivial succinctness to SNARKs with full succinctness.

[17] SNARGs where a polynomial bound on the time has to be fixed for the CRS generation are called preprocessing SNARGs, but they are not sufficient for our constructions.

$\underline{\mathsf{Exp}^{\mathsf{sse}}_{\mathcal{A},\mathcal{E},\Pi=(\mathsf{CRSGen},\mathsf{Prove},\mathsf{Verify})}(\lambda)}$:	$\underline{\textsc{Prove}(x)}$:

$Q_{\mathsf{sim}} := \emptyset$
$(\mathsf{crs}, \mathsf{td}) \leftarrow S_1(1^\lambda)$
$(x, \pi) \leftarrow \mathcal{A}^{\textsc{Prove}(\cdot)}(1^\lambda, \mathsf{crs}; r)$
$w \leftarrow \mathcal{E}(1^\lambda, \mathsf{crs}, Q_{\mathsf{sim}}, r)$
if $(x, \_) \in Q_{\mathsf{sim}} \vee (x, w) \in \mathcal{R}$ **then**
$\quad$**return** 0
else
$\quad$**return** 1
	$\pi \leftarrow S_2(\mathsf{crs}, \mathsf{td}, x)$
	$Q_{\mathsf{sim}} := Q_{\mathsf{sim}} \cup \{(x, \pi)\}$
	return π

Fig. 2. The simulation extractability experiment for a zk-SNARK $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ for NP-relation $\mathcal{R}$ with zero-knowledge simulator $S = (S_1, S_2)$.

2.2 Malleable NIZKs

We recall the definition of Malleable NIZKs [23].

Definition 7 (Admissible Transformation). *We say that an n-ary transformation $T = (T_x, T_w)$ is admissible for an NP relation $\mathcal{R}$ if*

$$\forall i \in \{1, \ldots, n\} : (x_i, w_i) \in \mathcal{R} \implies (T_x(x_1, \ldots, x_n), T_w(w_1, \ldots, w_n)) \in \mathcal{R} \ ,$$

T_x and T_w are efficiently computable and for all statements $x_1, \ldots, x_n$ we have $|T| + |x_1| + \cdots + |x_n| \leq \mathsf{poly}(|T_x(x_1, \ldots, x_n)|)$ for a fixed polynomial poly.[18]

We say that a set of transformations $\mathcal{T}$ is allowable for $\mathcal{R}$ if every $T \in \mathcal{T}$ is admissible for $\mathcal{R}$ and membership in $\mathcal{T}$ is efficiently decidable.

Definition 8 (Malleable argument system). *An argument system $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ for $\mathcal{R}$ is malleable for an allowable set of transformations $\mathcal{T}$ if there is a PPT algorithm* ZKEval *that inputs the CRS* crs*, an n-ary transformation $T = (T_x, T_w) \in \mathcal{T}$ and n statements with corresponding arguments $(x_i, \pi_i)_{1 \leq i \leq n}$ and outputs an argument π for $T_x(x_1, \ldots, x_n)$.*

For completeness, we additionally require that for all $T = (T_x, T_w) \in \mathcal{T}$ and all $(x_i, \pi_i)_{1 \leq i \leq n}$, for $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$

$$\begin{aligned} &\Pr[\forall i \in [n] : \mathsf{Verify}(\mathsf{crs}, x_i, \pi_i) = 1 \\ &\quad \wedge \mathsf{Verify}(\mathsf{crs}, T_x(x_1, \ldots, x_n), \mathsf{ZKEval}(\mathsf{crs}, T, (x_i, \pi_i)_{1 \leq i \leq n})) = 0] \leq \mathsf{negl}(\lambda) \end{aligned}$$

for a negligible function negl*. The probability is taken over the random coins of* CRSGen*,* ZKEval*, and* Verify*.*

A malleable NIZK obtained via applying a transformation should also hide the input statements (and proofs) of the transformation. The work [23] introduces two security notions for this:

[18] The last property is necessary to ensure that all our algorithms run in polynomial time in their input.

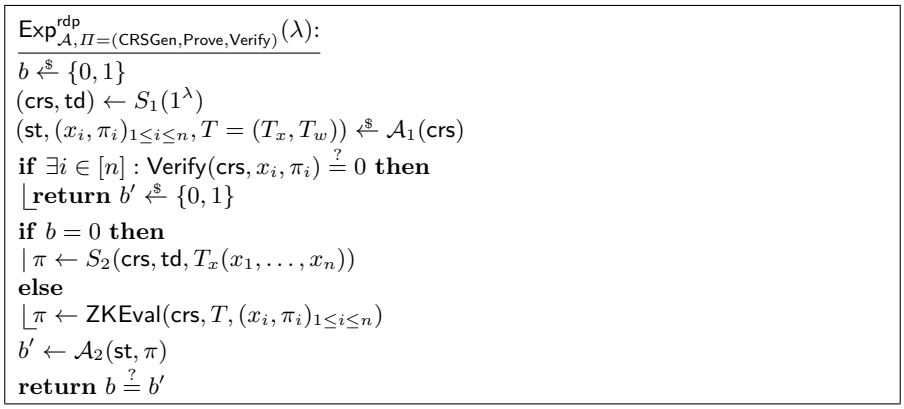

Fig. 3. The relaxed derivation privacy experiment for a NIZK.

– "Derivation privacy" guarantees this for true statements: In the game the adversary has to provide a transformation and the input statements with witnesses and proofs. The adversary wins if it can distinguish the output of ZKEval from a freshly generated proof for the same statement.
– "Strong derivation privacy" guarantees this also for (possibly false) statements with a simulated proof: The adversary gets access to the zero-knowledge trapdoor and does not have to provide a witness for the statements. The adversary wins if it can distinguish the output of ZKEval from a simulated proof for the same statement.

The version of derivation privacy we consider is similar to the strong derivation privacy of [23], but weakened in one aspect: The adversary does not get the trapdoor to the CRS.

Definition 9 (Relaxed derivation privacy). *A malleable NIZK* $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{ZKEval})$ *with associated simulator* $S = (S_1, S_2)$ *is* relaxed derivation private *if for every PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *we have*

$$\mathsf{Adv}^{\mathsf{rdp}}_{\mathcal{A},\Pi}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{rdp}}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2}\right| \le \mathsf{negl}(\lambda)$$

for a negligible function negl *where the experiment is defined in Figure 3.*

We also provide a definition of controlled-malleable simulation extractability for malleable NIZKs. This differs from definition provided by [23] in two points:

– The definition by [23] is only for the special case where $\mathcal{T}$ consists only of unary transformations and $(\mathcal{T}, \circ)$ is a monoid (where $\circ$ is the composition operator $(T_x, T_w) \circ (T'_x, T'_w) := (T_x \circ T'_x, T_w \circ T'_w)$). Ours is more general for arbitrary sets of admissible transformations.
– The definition of [23] guarantees a universal extractor that takes an extraction trapdoor for the CRS. Our definition guarantees an extractor depending on the adversary that extracts using the randomness and inputs of the adversary.

For a malleable NIZK where $\mathcal{R}$ is non-trivial ($\mathcal{L}_{\mathcal{R}}$ is not decidable in polynomial time) and $\mathcal{T} \neq \emptyset$ and $\mathcal{T} \neq \{\mathsf{id}\}$ (where id is the identity function) we can not hope to achieve the standard definition of simulation extractability, since the adversary can always ask for a simulated proof and then apply a transformation to it. Our definition captures this by relaxing the requirements on the extractor. The extractor does not have to provide a witness for the statement proven by the adversary. Instead, if suffices if the extractor provides an "explanation" of the statement where

- simulated statements require no explanation,
- a witness for the statement is an explanation, or
- a transformation that obtains the statement from other explainable statements (without introducing any circularities) is an explanation.

Definition 10 (Controlled-malleable simulation extractability). *A malleable non-interactive zero-knowledge argument system* $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *for an NP relation* $\mathcal{R}$ *and an admissible set of transformations* $\mathcal{T}$ *is* controlled-malleable simulation extractable *if for every PPT adversary* $\mathcal{A}$ *there exists a PPT extractor* $\mathcal{E}$ *such that*

$$\mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{A},\mathcal{E},\Pi}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{cm\text{-}sse}}_{\mathcal{A},\mathcal{E},\Pi}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

for a negligible function negl *where the experiment is defined in Figure 4.*

$\underline{\mathsf{Exp}^{\mathsf{cm\text{-}sse}}_{\mathcal{A},\mathcal{E},\Pi=(\mathsf{CRSGen},\mathsf{Prove},\mathsf{Verify})}(\lambda)}$:	$\underline{\mathsf{checkExplanation}(x, E)}$:
$Q_{\mathsf{sim}} := \emptyset$	**if** $(x, \_) \in Q_{\mathsf{sim}}$ **then**
$(\mathsf{crs}, \mathsf{td}) \leftarrow S_1(1^\lambda)$	**return** 1
$(x, \pi) \leftarrow \mathcal{A}^{\textsc{Prove}(\cdot)}(1^\lambda, \mathsf{crs}; r)$	**if** $\exists w : (x, w) \in E \wedge (x, w) \in \mathcal{R}$ **then**
$E \leftarrow \mathcal{E}(1^\lambda, \mathsf{crs}, Q_{\mathsf{sim}}, r)$	**return** 1
if $\mathsf{checkExplanation}(x, E) = 1$ **then**	**if** $\exists T = (T_x, T_w), x_1, \ldots, x_n : (x, (T, x_1, \ldots, x_n)) \in E$ **then**
return 0	**if** $T \in \mathcal{T} \wedge x = T_x(x_1, \ldots, x_n)$ **then**
else	$E' := E \setminus \{(x, w) \in E\}$
return 1	**return** $\bigwedge_{i=1}^{n} \mathsf{checkExplanation}(x_i, E')$
$\underline{\textsc{Prove}(x)}$:	**return** 0
$\pi \leftarrow S_2(\mathsf{crs}, \mathsf{td}, x)$	
$Q_{\mathsf{sim}} := Q_{\mathsf{sim}} \cup \{(x, \pi)\}$	
return π	

Fig. 4. The simulation extractability experiment for a malleable NIZK $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ for NP-relation $\mathcal{R}$ and set of transformations $\mathcal{T}$ with zero-knowledge simulator $S = (S_1, S_2)$.

Definition 11 (Controlled-malleable simulation soundness). *A malleable non-interactive zero-knowledge argument system* $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *for*

an NP relation $\mathcal{R}$ and an admissible set of transformations $\mathcal{T}$ is controlled-malleable simulation sound *if for every PPT adversary $\mathcal{A}$*

$$\mathsf{Adv}^{\mathsf{cm\text{-}ss}}_{\mathcal{A},\Pi}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{cm\text{-}ss}}_{\mathcal{A},\Pi}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

for a negligible function negl *where the experiment is defined in Figure 5.*

$\underline{\mathsf{Exp}^{\mathsf{cm\text{-}ss}}_{\mathcal{A},\mathcal{E},\Pi=(\mathsf{CRSGen},\mathsf{Prove},\mathsf{Verify})}(\lambda)\text{:}}$	$\underline{\textsc{Prove}(x)\text{:}}$
$Q_{\mathsf{sim}} := \emptyset$	$\pi \leftarrow S_2(\mathsf{crs}, \mathsf{td}, x)$
$(\mathsf{crs}, \mathsf{td}) \leftarrow S_1(1^\lambda)$	$Q_{\mathsf{sim}} := Q_{\mathsf{sim}} \cup \{(x, \pi)\}$
$(x, \pi) \leftarrow \mathcal{A}^{\textsc{Prove}(\cdot)}(1^\lambda, \mathsf{crs}; r)$	**return** π
return $\mathsf{Verify}(\mathsf{crs}, x, \pi) \stackrel{?}{=} 1 \wedge x \notin \mathcal{L}^\star$	

Fig. 5. The simulation soundness experiment for a malleable NIZK $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ for NP-relation $\mathcal{R}$ and set of transformations $\mathcal{T}$ with zero-knowledge simulator $S = (S_1, S_2)$.

$$\mathcal{L}^\star_0 := \{x \mid (x, \_) \in \mathcal{R} \vee (x, \_) \in Q_{\mathsf{sim}}\}$$

$$\text{For } i \in \mathbb{N}_0 : \mathcal{L}^\star_{i+1} := \{T_x(x_1, \ldots, x_n) \mid (T_x, \_) \in \mathcal{T} \wedge x_1, \ldots, x_n \in \bigcup_{i=0}^{i} \mathcal{L}^\star_i\}$$

$$\mathcal{L}^\star := \bigcup_{i=0}^{\infty} \mathcal{L}^\star_i$$

Lemma 1. *If Π is a controlled-malleable simulation extractability malleable NIZK, it is also controlled-malleable simulation sound.*

Proof. Let $\mathcal{A}$ be an adversary against the controlled-malleable simulation soundness of Π. Whenever $\mathcal{A}$ wins it outputs (x, π) with $\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1$ and $x \notin \mathcal{L}^\star$. The latter implies that there does not exist a valid explanation for the statement x, and thus no extractor $\mathcal{E}$ can extract such an explanation. Thus $\mathcal{A}$ also wins the controlled-malleable simulation extractability game. □

3 Construction without counters

Here we give a construction of a malleable NIZK that achieves derivation privacy for any NP-relation $\mathcal{R}$ and any allowable set of transformations $\mathcal{T}$ for $\mathcal{R}$. The construction uses zk-SNARKs for NP with fast extraction and a new, strong type of one-way functions that is hard to invert on adversarially chosen inputs.

3.1 Adversarial one-way functions

A family $\mathcal{F} = \{f_\lambda : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ of one-way functions satisfies that each f_λ is PPT computable but it is hard for a PPT adversary to find a preimage of given image y. This image y is sampled as $f_\lambda(x)$ for $x \xleftarrow{\$} \mathcal{X}_\lambda$. We now want to modify this definition by allowing the adversary to choose y. Of course, it then becomes easy for an adversary to find a preimage: The adversary can pick any $x \in \mathcal{X}_\lambda$ and output $y := f_\lambda(x)$ together with x as preimage. To make this definition achievable, we ask the adversary to invert the function f_λ many times, concretely, $p(t)$ times, where p is a polynomial and t is the time it took the adversary to output y. To avoid that a (non-uniform) adversary wins the game trivially by having hardcoded values x, y where $y = f_\lambda^{p(t)}(x)$ and t is the time it takes $\mathcal{A}_1$ to output y, we have a setup algorithm that generates public parameters pp that describe the function.

To achieve malleable NIZKs for transformations with arity $n > 1$, we have to generalize the AOWFs to multiple inputs. The goal of the adversary is then to output a graph, where at each node the incoming edges are ordered and each node has label that corresponds to the AOWF applied to all its inputs (in the given order). Moreover, there should exist a single node without any outgoing edges (the sink) labeled y and every node should have a path to this sink. The adversary wins if this graph has a cycle or more than $p(t)$ *distinct* nodes. Note that when restricting to arity 1, this is equivalent to the previously sketched definition.

We also give our adversarial one-way function an additional random input. This allows us to achieve a second property that we need for our construction that we call unlinkability. Informally speaking, this guarantees us that the output of the adversarial one-way function looks like a fresh sample, even given the input (but not the randomness).

Definition 12 (Adversarial one-way function generator). *An adversarial one-way function (AOWF) generator* $\mathsf{AOWFGen}$ *inputs* 1^λ *and outputs public parameters* pp *describing sets* $\mathcal{X}_{\mathsf{pp}}$ *and* $\mathfrak{R}_{\mathsf{pp}}$ *from which we can draw efficiently uniformly random samples and an efficiently computable function* $f_{\mathsf{pp}} : \mathcal{X}^*_{\mathsf{pp}} \times \mathfrak{R}_{\mathsf{pp}} \to \mathcal{X}_{\mathsf{pp}}$ *(the AOWF).*

Definition 13. *An explanation of* y *for the AOWF* $f_{\mathsf{pp}} : \mathcal{X}^*_{\mathsf{pp}} \times \mathfrak{R}_{\mathsf{pp}} \to \mathcal{X}_{\mathsf{pp}}$ *is a directed graph* $G = (V, E)$*, where each node has a main label in* $\mathcal{X}_{\mathsf{pp}}$ *and either*

1. *no parents (i.e., in-degree* 0*) or*
2. *at least one parent and auxiliary labels that describe a total order of the parents and a value* $r \in \mathfrak{R}_{\mathsf{pp}}$ *such that* $y = f_{\mathsf{pp}}(x_1, \ldots, x_n, r)$*, where* y *is the main label of this node and* $x_1, \ldots, x_n$ *are the parent's main labels in the specified order.*

Moreover, the graph has to have exactly one node without children and this node's main label has to be y*. Finally, every node should have a path to the unique child-free node. We write* $\mathsf{valid}_{\mathsf{pp}}(G, y)$ *for the event that* G *is an explanation for* y *satisfying these requirements.*

We say that an explanation $G = (V, E)$ is good for parameter s, denoted by $\mathsf{good}_{\mathsf{pp}}(G, s)$*, if*

1. *G is acyclic*[19] *and*
2. *G has strictly less than s nodes with disjoint main labels.*

Definition 14 (Adversarial one-wayness). *An AOWF generator* AOWFGen *is* adversarially one-way *if there exists a polynomial p such that for every two-stage PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*

$$\Pr[\mathsf{valid}_{\mathsf{pp}}(G, y) \wedge \neg \mathsf{good}_{\mathsf{pp}}(G, p(t))] \leq \mathsf{negl}(\lambda)$$

for a negligible function negl *where the probability is taken over* $\mathsf{pp} \leftarrow \mathsf{AOWFGen}(1^\lambda)$*,* $(y, \mathsf{st}) \xleftarrow{t} \mathcal{A}_1(1^\lambda, \mathsf{pp})$*, and $G \leftarrow \mathcal{A}_2(y, \mathsf{st})$. Here t is the runtime of $\mathcal{A}_1(1^\lambda, \mathsf{pp})$.*

Note that the polynomial p has to satisfy $p(t) > t/t_f$ for all t, where t_f is the time it takes to sample $r \xleftarrow{\$} \mathfrak{R}_{\mathsf{pp}}$ and to evaluate the function f_{pp}, because otherwise there is a trivial attack: $\mathcal{A}_1(1^\lambda, \mathsf{pp})$ can take any $x \in \mathcal{X}_{\mathsf{pp}}, r_1, \ldots r_{p(t)} \in \mathfrak{R}$ and compute and output $y := f_{\mathsf{pp}}^{p(t)}(x)$ and $\mathsf{st} := (x, r_1, \ldots, r_{p(t)})$. The second stage then simply outputs the graph with root node v_0 with label x and nodes v_i with label $x_i := f_{\mathsf{pp}}(x_{i-1}, r_i)$ and auxiliary label r_i. This graph has either $p(t)$ distinct labels, or two nodes with the same label. In the latter case it can be easily turned into an explanation graph with a cycle.

We can turn this idea into an impossibility result (under a mild additional assumption) showing that adversarial one-way functions cannot be built from falsifiable assumptions in a black-box way. The proof uses that a black-box reduction cannot "see" the run-time of an adversary. We refer the reader to Appendix F for more details.

Definition 15 (Collision-resistance). *An AOWF generator* AOWFGen *is* collision-resistance *if for every PPT adversary $\mathcal{A}$*

$$\Pr[f_{\mathsf{pp}}(x_1, \ldots, x_n, r) = f_{\mathsf{pp}}(x'_1, \ldots, x'_n, r'))] \leq \mathsf{negl}(\lambda)$$

for a negligible function negl*, where the probability is taken over* $\mathsf{pp} \leftarrow \mathsf{AOWFGen}(1^\lambda)$ *and* $(x_1, \ldots, x_n, r, x'_1, \ldots, x'_n, r') \leftarrow \mathcal{A}(\mathsf{pp})$*.*

Definition 16 (Unlinkability). *An AOWF generator* AOWFGen *is* unlinkable *if for every two-stage PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{unlink}}_{\mathsf{AOWFGen},\mathcal{B}}(\lambda) := \Pr[\mathcal{A}_2(x_1, \ldots, x_n, \mathsf{st}, f_{\mathsf{pp}}(x_1, \ldots, x_n, r)) \Rightarrow 1] \\ - \Pr[\mathcal{A}_2(x_1, \ldots, x_n, \mathsf{st}, y) \Rightarrow 1] \leq \mathsf{negl}(\lambda)\end{aligned}$$

for a negligible function negl *where the probability is taken over* $\mathsf{pp} \leftarrow \mathsf{AOWFGen}(1^\lambda)$*,* $(x_1, \ldots, x_n, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{pp})$*,* $r \xleftarrow{\$} \mathfrak{R}_{\mathsf{pp}}$*, and* $y \xleftarrow{\$} \mathcal{X}_{\mathsf{pp}}$*.*

[19] We count loops as cycles.

Hash function instantiation. We next provide a candidate constructions of adversarial one-way function generators. The first heuristic candidate is to use a cryptographic hash function, like SHA-3 (Keccak), that is designed to behave like a random function in many ways. To justify this, we prove that a random oracle is adversarial one-way and unlinkable. Note that we can not use a random oracle in place of an adversarial one-way function in our Malleable NIZK construction, because we need to use this function inside the SNARK language.

Theorem 1. *In the ROM, where the adversary gets oracle access to a random function $f_\lambda : \{0,1\}^* \to \{0,1\}^\lambda$, the random oracle is adversarial one-way for $\mathcal{X}_{\mathsf{pp}} = \mathfrak{R}_{\mathsf{pp}} = \{0,1\}^\lambda$, collision-resistant and unlinkable.*

Proof. We first prove adversarial one-wayness. Therefore, we use the polynomial $p(X) = X/\lambda + 2$.

In an explanation graph $G = (V, E)$ each non-root node v's main label is the ROM output when inputting the main label of the parent nodes and the string r contained in the auxiliary label. We call this the ROM query corresponding to v. We assume without loss of generality that $\mathcal{A}_2$ makes all the ROM queries corresponding to a node of the explanation graph. We assume that performing a random oracle query on input length n takes at least n time steps.

Let S_i be the set of all z for which there are $j, n \in \mathbb{N}_+$, $j \leq n$, $x_1, \ldots, x_{j-1}$, $x_{j+1}, \ldots, x_n, r \in \{0,1\}^\lambda$ such that $\mathcal{A}_i$ queried the RO on $x_1 \| \cdots \| x_{j-i} \| z \| x_{j+1} \| \cdots \| x_n \| r$, and set $S := S_1 \cup S_2$. Then $|S_1| \leq t/\lambda = p(\lambda) - 2$ and $|S|$ is polynomial.

We first show that $\mathcal{A}_2$ can only output an explanation graph $G = (V, E)$ with a cycle with negligible probability. Note that a root node cannot be part of a cycle and therefore each node on a cycle has a corresponding ROM query. Let q be the number of random oracle queries of $\mathcal{A}_1$ and $\mathcal{A}_2$. Then there must be an index $i \in [q]$ such that the i-th ROM query is the last ROM query corresponding to the cycle. Then the output of the ROM query must be equal to the main label of a node on that cycle, that has been part of the input of a previous or the current hash query. However, the output of the ROM query is sampled uniformly random and will therefore match any of these inputs with probability $2^{-\lambda}$. Since there are at most $|S|$ such previous inputs, the probability of a ROM query completing a cycle is at most $|S|/2^\lambda$, which is negligible.

Next, we argue that if $\mathcal{A}_2$ outputs a valid explanation graph, it will contain less than $p(t)$ nodes with disjoint main labels. Assume the adversary outputs a graph with at least $p(t)$ disjoint main labels. Then, there must be at least two labels that are not contained in S_1 and at least one of them is not y. Let v be a node with the shortest path to y that has a label $x \notin S_1$.

Now assume that y is the output of a ROM query that $\mathcal{A}_1$ made. Since $\mathcal{A}_2$ evaluates all ROM queries that are necessary to check the graph, $\mathcal{A}_2$ makes a ROM query where x is one of the inputs and the output x' satisfies $x' \in S_1$. If $x' \notin S_1$, v would not have a shortest path to y among all nodes with main label not in S_1. However, the probability that $\mathcal{A}_2$ makes a ROM query whose output is in S_1 (which is fixed by the run of $\mathcal{A}_1$) and that $\mathcal{A}_1$ did not make is at most $|S_1|/2^\lambda$, which is negligible.

What remains is the case where y was not chosen as the output of a ROM query of $\mathcal{A}_1$ but $\mathcal{A}_2$ made a ROM query with output y. For each ROM query that $\mathcal{A}_2$ makes, it will get output y with probability $1/2^\lambda$ and thus the probability that this case happens is at most $q/2^\lambda$.

To prove collision-resistance of this construction, it suffices to use the birthday-bound to show that an adversary making q ROM queries can output a collision at most with probability $q^2/2^\lambda$.

Finally, we show that this construction is unlinkable. Therefore, let $(x_1, \ldots, x_n, \mathsf{st}) \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{pp})$ and $r \xleftarrow{\$} \{0,1\}^\lambda$. The probability that $\mathcal{A}_1$ or $\mathcal{A}_2$ makes a random oracle query on $(x_1, \ldots, x_n, r)$ is at most $q/2^\lambda$ and thus negligible, where q is the number of random oracle queries of $\mathcal{A}_1$ and $\mathcal{A}_2$. If $\mathcal{A}_2$ does not make such a query, $f_\lambda(x_1, \ldots, x_n, r)$ is statistically indistinguishable from $y \xleftarrow{\$} \{0,1\}^\lambda$. □

3.2 Construction

In Figure 6 we give a construction of a malleable NIZK Π_{aowf} that achieves derivation privacy for any NP-relation $\mathcal{R}$ and any allowable set of transformations $\mathcal{T}$ for $\mathcal{R}$. The construction uses a (zk-)SNARK for NP with fast extraction and an adversarial one-way function.

The construction defines a Turing machine M (the NP-Verifier) that gets as one input the description of another Turing machine. When we use this machine, this input will always be the description of the Turing machine itself ($\widetilde{M} = M$). This way, we avoid the use of the recursion theorem (which has been used in this situation for example in [8]), which also avoids any additional assumptions about the resulting Turing machine running in polynomial time.

We next describe how to set the time bound $\tau(\lambda, m)$ for the above construction. Let $n_{\max}$ be the maximum arity of the transformations in $\mathcal{T}$. The run time of M excluding the SNARK verification in the last step for inputs (ξ, x, M) is polynomial in λ and $|x|$. The i-th SNARK verification is polynomial in λ, $|x_i|$ (which is again polynomial in $|x|$ by Definition 7), and $\log(\tau(\lambda, m))$. This leads to the following recursive relation that τ must satisfy in order to be a valid time bound:

$$\tau(\lambda, m) \geq n_{\max}\mathsf{poly}(\lambda, m, \log(\tau(\lambda, m)))$$

for a suitable polynomial poly. We set $\tau(\lambda, m) := n_{\max}\mathsf{poly}(\lambda, m, \lambda \cdot m)$. This satisfies the above recursive relation asymptotically: Clearly, τ grows polynomial in λ and m and thus $2^{\lambda \cdot m}$ is an asymptotic upper bound. Plugging in this upper bound on the right-hand side of the recursive relation proves the claim.

Theorem 2 (Completeness). *The malleable NIZK Π_{aowf} is complete if the underlying SNARK is complete.*

Proof. Let $\mathsf{crs} \xleftarrow{\$} \mathsf{SNARK.CRSGen}(1^\lambda)$ and $\mathsf{pp} \leftarrow \mathsf{AOWFGen}(1^\lambda)$. First, we show that proofs generated directly via $\Pi_{\mathsf{aowf}}.\mathsf{Prove}$ verify. Let $(x, w) \in \mathcal{R}$ and $\pi' = (\xi, \pi) \leftarrow \Pi_{\mathsf{aowf}}.\mathsf{Prove}(\mathsf{crs}' := (\mathsf{crs}, \mathsf{pp}), x, (w, \perp))$. Since $(x, w) \in \mathcal{R}$, $M((\xi, x, M), (w, \perp))$ will return 1 and, if the security parameter is large enough, this will happen

$\underline{\Pi_{\mathsf{aowf}}.\mathsf{CRSGen}(1^\lambda):}$
$\mathsf{crs} \leftarrow \mathsf{SNARK.CRSGen}(1^\lambda)$
$\mathsf{pp} \leftarrow \mathsf{AOWFGen}(1^\lambda)$
return $\mathsf{crs}' := (\mathsf{crs}, \mathsf{pp})$

$M :=$
input: $(\xi, x, \widetilde{M}), w' = (w, (T = (T_x, T_w), (x_i, \pi'_i = (\xi_i, \pi_i))_{1 \le i \le n}, r))$
//Return 0 if witness does not have the right format
if $\xi \notin \mathcal{X}_{\mathsf{pp}}$ **then**
  return 0
if $(x, w) \in \mathcal{R}$ **then**
  return 1
if $x \neq T_x(x_1, \ldots, x_n) \vee \xi \neq f_{\mathsf{pp}}((x_1, \pi'_1), \ldots, (x_n, \pi'_n), r) \vee r \notin \mathfrak{R}_{\mathsf{pp}}$ **then**
  return 0
for $i \in [n]$ **do**
  if $\mathsf{SNARK.Verify}(\mathsf{crs}, (\widetilde{M}, (\xi_i, x_i), \tau(\lambda, |x_i|)), \pi_i) = 0$ **then**
    return 0
return 1

$\underline{\Pi_{\mathsf{aowf}}.\mathsf{Prove}(\mathsf{crs}' = (\mathsf{crs}, \mathsf{pp}), x, w):}$
$\xi \xleftarrow{\$} \mathcal{X}_{\mathsf{pp}}$
$\pi := \mathsf{SNARK.Prove}(\mathsf{crs}, (M, (\xi, x, M), 1^{\tau(\lambda, |x|)}), (w, \bot))$
return $\pi' := (\xi, \pi)$

$\underline{\Pi_{\mathsf{aowf}}.\mathsf{ZKEval}(\mathsf{crs}' = (\mathsf{crs}, \mathsf{pp}), T = (T_x, T_w), (x_i, \pi'_i = (\xi_i, \pi_i))_{1 \le i \le n}):}$
$r \xleftarrow{\$} \mathfrak{R}_{\mathsf{pp}}$
$\xi := f_{\mathsf{pp}}((x_1, \pi'_1), \ldots, (x_n, \pi'_n), r)$
$\pi \leftarrow \mathsf{SNARK.Prove}(\mathsf{crs}, (M, (\xi, T_x(x_1, \ldots, x_n), M), 1^{\tau(\lambda, |x|)}), (\bot, (T, (x_i, \pi'_i)_{1 \le i \le n}), r))$
return $\pi' := (\xi, \pi)$

$\underline{\Pi_{\mathsf{aowf}}.\mathsf{Verify}(\mathsf{crs}, x, \pi' = (\xi, \pi)):}$
if $\ell \le B(\lambda)$ **then**
  return $\mathsf{SNARK.Verify}(\mathsf{crs}, (M, (\xi, x, M), \tau(\lambda, |x|)), \pi)$
else
  return 0

Fig. 6. Our malleable NIZK construction $\Pi_{\mathsf{aowf}} = (\Pi_{\mathsf{aowf}}.\mathsf{CRSGen}, \Pi_{\mathsf{aowf}}.\mathsf{Prove}, \Pi_{\mathsf{aowf}}.\mathsf{Verify}, \Pi_{\mathsf{aowf}}.\mathsf{ZKEval}, \Pi_{\mathsf{aowf}}.\mathsf{Level})$ with counters from a zk-SNARK $\mathsf{SNARK} = (\mathsf{SNARK.CRSGen}, \mathsf{SNARK.Prove}, \mathsf{SNARK.Verify})$ and an adversarial one-way function with generator $\mathsf{AOWFGen}$. The construction works for any NP relation $\mathcal{R}$ and any set of admissible transformations $\mathcal{T}$. The time bound τ is defined later.

within the time bound $\tau(\lambda, m)$ defined before. Thus, by completeness of the underlying SNARK, the proof will verify.

Next, let $T = (T_x, T_w) \in \mathcal{T}$ be an n-ary admissible transformation and $\pi' := (\xi, \pi) \xleftarrow{\$} \Pi_{\mathsf{aowf}}.\mathsf{ZKEval}(\mathsf{crs}, T = (T_x, T_w), (x_i, \pi'_i)_{1 \le i \le n})$. Let us assume $\mathsf{Verify}(\mathsf{crs}, x_i, \pi'_i) = 1$ holds for all $i \in [n]$ with overwhelming probability. Then for $\pi'_i = (\xi_i, \pi_i)$ we have $\mathsf{SNARK.Verify}(\mathsf{crs}, (M, (\xi_i, x_i), \tau(\lambda, |x_i|)), \pi_i) = 1$ holds for all $i \in [n]$ with overwhelming probability. Furthermore, $\xi = f_{\mathsf{pp}}((x_1, \pi'_1), \ldots, (x_n, \pi'_n), r)$. This shows that $M((\xi, x, M), (\bot, (T = (T_x, T_w), (x_i, \pi'_i)_{1 \le i \le n}), r)) = 1$ and, if the security parameter is large enough, this will happen within the time bound $\tau(\lambda, m)$ defined before. Thus, by completeness of the underlying SNARK, the proof will verify. □

Theorem 3 (Full succinctness). *The malleable NIZK Π_{aowf} is fully succinct if the underlying SNARK is fully succinct.*

Proof. A proof $\pi' = (\xi, \pi)$ of Π_{aowf} consists of a proof π for the underlying SNARK and a value ξ from the domain of the AOWF. Both have a fixed polynomial length that is independent of the statement or witness length. □

Theorem 4 (Soundness). *The malleable NIZK Π_{aowf} is controlled-malleable simulation extractable if the underlying SNARK is simulation extractable with fast extraction and* AOWFGen *is adversarial one-way and collision-resistant.*

Proof. Let $\mathcal{A}$ be an algorithm that inputs $\mathsf{crs}' = (\mathsf{crs}, \mathsf{pp})$, has access to a proving oracle $\textsc{Prove}(x)$ that generates simulated proofs for arbitrary statements x and outputs $(x, \pi' = (\xi, \pi))$. We recursively show existence of an extractor $\mathcal{E}'_{\mathcal{A}}$. This extractor inputs $\mathsf{crs}' = (\mathsf{crs}, \mathsf{pp})$, $\mathcal{A}$'s randomness r, the list of simulated proofs Q_{sim}, and a cache of extracted proofs C and outputs an updated cache of extracted proofs C'. The final extractor $\mathcal{E}_{\mathcal{A}}$ just calls $\mathcal{E}'_{\mathcal{A}}$ with all its inputs and $C = \emptyset$ to get C'. It then simulates $\mathcal{A}$ on randomness r until it outputs a statement proof pair (x, π). It then searches the cache for an explanation E with $(\pi, E) \in C$ and returns E. The definition of $\mathcal{E}'_{\mathcal{A}}$ guarantees that there always exists a (unique) E with $(\pi, E) \in C$.

We now describe $\mathcal{E}'_{\mathcal{A}}$ on input $(\mathsf{crs}' = (\mathsf{crs}, \mathsf{pp}), r, Q_{\mathsf{sim}}, C)$. First, it simulates $\mathcal{A}$ on randomness r until it outputs a statement/proof pair (x, π). If there exists an explanation E with $(x, E) \in C$, it just returns $C' = C$. Otherwise, if $(x, \_) \in Q_{\mathsf{sim}}$, the extractor outputs $C' := C \cup \{(x, E)\}$ where $E := \{(x, \bot)\}$. Otherwise, it runs the extractor for the underlying SNARK for $\mathcal{A}$ on its own inputs (except for the cache). This extractor outputs with overwhelming probability a witness $w' = (w, (T = (T_x, T_w), (x_i, \pi'_i = (\xi_i, \pi_i))_{1 \le i \le n}, r)$ such that $M((\xi, x, M), w') = 1$. This implies (1) $(x, w) \in \mathcal{R}$ or (2) $x = T_x(x_1, \ldots, x_n)$ and $\mathsf{SNARK.Verify}(\mathsf{crs}, (\widetilde{M}, (\xi_i, x_i), \tau(\lambda, |x_i|)), \pi_i) = 1$ for all $i \in [n]$. In case (1), the extractor $\mathcal{E}'_{\mathcal{A}}$ outputs the explanation $E := \{(x, w)\}$. In case (2), let $\mathcal{A}_i$ be an algorithm that proceeds exactly like $\mathcal{E}'_{\mathcal{A}}$ up to this point and then outputs $(x_i, (\xi_i, \pi_i))$. We can recursively prove existence of an extractor $\mathcal{E}'_{\mathcal{A}_i}$ that extracts an explanation E_i for x_i. The extractor $\mathcal{E}'_{\mathcal{A}}$ runs in this case these extractors as follows: It runs each $\mathcal{E}'_i$ with its respective inputs and cache C_{i-1} to get its output

C_i for $i \in [n]$ in increasing order and with $C_0 = C$. The extractor then computes the explanation $E := \{(x, (T, (x_1, \ldots, x_n)))\} \cup \bigcup_{i=1}^{n} E_i$, where E_i is the (unique) explanation with $(x_i, E_i) \in C_n$. $\mathcal{E}'_{\mathcal{A}}$ then outputs $C' := C_n \cup \{(x, E)\}$.

First, we show that the extractor $\mathcal{E}_{\mathcal{A}}$ outputs a valid explanation. Therefore, we show that $\mathcal{E}'_{\mathcal{A}}$ outputs a cache C' that contains a unique, valid explanation for the statement x that $\mathcal{A}$ proved when run with the given inputs and a cache C that contains at most one explanation for each statement, and this explanation is valid. This is trivially true in the initial extractor call with $C = \emptyset$. In the case where $\mathcal{E}'_{\mathcal{A}}$ does not modify the cache, because it already contains an explanation, the statement holds trivially. In the base case where the statement is a simulated statement, this is trivial. In the other base case, where the SNARK extractor outputs a witness w with $(x, w) \in \mathcal{R}$, this follows directly from knowledge soundness of the underlying SNARK. In the recursive step, if all the extractors $\mathcal{E}_i$ output a valid explanation E_i for x_i, the explanation $E := \{(x, (T = (T_x, T_w), (x_1, \ldots, x_n)))\} \cup \bigcup_{i=1}^{n} E_i$ is a valid explanation for $x = T_x(x_1, \ldots, x_n)$ since T is an admissible transformation by the knowledge soundness of the underlying SNARK.

Next, we bound the runtime of the extractor by using the adversarial one-wayness of $\mathsf{AOWFGen}$. Let $p(t)$ be the polynomial from Definition 12 for $\mathsf{AOWFGen}$ and t be the run time of the adversary $\mathcal{A}_1$ that

- inputs $(1^\lambda, \mathsf{pp})$ where pp are parameters for the adversarial one-way function,
- samples $(\mathsf{crs}, \mathsf{td}) \leftarrow S_1(1^\lambda)$, where (S_1, S_2) is the zero-knowledge simulator for SNARK,
- runs $(x, \pi' = (\xi, \pi)) \leftarrow \mathcal{A}^{\textsc{Prove}}(\mathsf{crs}' = (\mathsf{crs}, \mathsf{pp}))$, where $\textsc{Prove}(x)$ is answered with $S_2(\mathsf{td}, x)$, and
- outputs ξ.

If our extractor needs at least $p(t)$ SNARK extractions, we can use it to break the security of $\mathsf{AOWFGen}$. Therefore, let $G = (V, E)$ be the directed graph that contains a node for each statement-proof pair $(x, \pi' = (\xi, \pi))$ that we encounter during the recursive extraction procedure and and an edge (u, v) if extraction for the node v leads to recursive extraction of the node u (or a cache lookup for the explanation belonging to u). Our recursive extraction procedure proceeds like a depth-first search on this graph, where the cache avoids that we extract an explanation for the same node more than once.

There are two cases where our recursive extraction would perform more than $p(t)$ SNARK extractions. Case 1: The graph G contains a cycle (in this case our extraction would not terminate). Case 2: The graph G is a acyclic (a DAG), but it contains at least $p(t)$ different nodes.

We can turn the graph G into an explanation graph G' for the AOWF. This graph will have for each root node of G with label $(x, \pi' = (\xi, \pi))$ a root node with main label ξ and for each non-root node of G with label $(x, \pi' = (\xi, \pi))$ a non-root node with main label ξ and auxiliary label r that has for input to the transformation $(x_i, \pi'_i = (\xi_i, \pi_i))$ three parent nodes with main labels x_i, π_i, and ξ_i. The nodes with labels x_i and π_i are always root nodes.

If G has a cycle, then so has G'. If G has at least $p(t)$ different nodes, but G' has less than $p(t)$ different nodes with disjoint main labels, we could break collision-resistance of the AOWF. Thus, with overwhelming probability, we can construct in the event that the extractor would perform more than $p(t)$ SNARK extractions an explanation graph that breaks the adversarial one-wayness of $\mathsf{AOWFGen}$.

Thus, we can make our extractor for Π_{aowf} abort after $p(t)$ SNARK extractions and it will still be successful with overwhelming probability. Since the underlying extractor has fast extraction, there exists a polynomial poly such that for every PPT adversary $\mathcal{A}$ the corresponding extractor $\mathcal{E}$ (and also any algorithm that runs $\mathcal{E}$ and then deletes some parts of its output) runs in time $\mathsf{Time}_{\mathcal{E}}(\lambda) \leq \mathsf{Time}_{\mathcal{A}}(\lambda) + \mathsf{poly}(\lambda)$. This leads to a total runtime of

$$p(t)(\mathsf{Time}_{\mathcal{A}}(\lambda) + p(t)\mathsf{poly}(\lambda)),$$

which is polynomial in λ. □

Theorem 5 (Zero-knowledge). *The malleable proof system Π_{aowf} is zero-knowledge, if the underlying SNARK is zero-knowledge.*

The proof is analogous to Theorem 10.

Theorem 6 (Derivation privacy). *The malleable proof system Π_{aowf} is relaxed derivation private, if the underlying SNARK is zero-knowledge and $\mathsf{AOWFGen}$ is unlinkable.*

Proof. Throughout the proof, let $(\mathsf{st}, (x_i, \pi'_i(\xi_i, \pi_i))_{1 \leq i \leq n}, T = (T_x, T_w))$ be the output of the first stage of the adversary.

The security reduction proceeds via a hybrid argument. Let G_0 be the real relaxed derivation privacy game for Π_{aowf}. The first hybrid G_1 differs to G_0 in one aspect: If $b = 0$, the game samples the SNARK CRS as $\mathsf{crs} \leftarrow \mathsf{SNARK.CRSGen}(1^\lambda)$ (instead of using the simulator).

Lemma 2 ($\mathsf{G}_0 \rightsquigarrow \mathsf{G}_1$). *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ with*

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{B},\mathsf{SNARK}}(\lambda).$$

Proof. If $b = 1$, the reduction uses the CRS obtained from the zero-knowledge game for SNARK. The reduction samples everything else by itself, in particular it does not make use of the PROVE oracle. □

The next hybrid G_2 is identical to G_1, except that if $b = 0$ the SNARK CRS is sampled via $(\mathsf{crs}, \mathsf{td}) \leftarrow S_1(1^\lambda)$ and the adversary gets (ξ, π) where (as before) $\xi := f_{\mathsf{pp}}((x_1, \pi'_1), \ldots, (x_n, \pi'_n), r)$ for $r \xleftarrow{\$} \mathcal{X}_{\mathsf{pp}}$ and $\pi \leftarrow S_2(\mathsf{crs}, \mathsf{td}, T_x(x_1, \ldots, x_n))$ (instead of computing it honestly via $\mathsf{SNARK.Prove}$).

Lemma 3 ($\mathsf{G}_1 \rightsquigarrow \mathsf{G}_2$). *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ with*

$$\left|\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{B},\mathsf{SNARK}}(\lambda).$$

The proof is a straightforward reduction to the zero-knowledge property of SNARK.

In the next hybrid G_3, if $b = 0$ we pick $\xi \xleftarrow{\$} \mathcal{X}_{\mathsf{pp}}$ instead of $\xi := f_{\mathsf{pp}}((x_1, \pi'_1), \ldots, (x_n, \pi'_n), r)$ for $r \xleftarrow{\$} \mathcal{X}_{\mathsf{pp}}$.

Lemma 4 ($\mathsf{G}_2 \rightsquigarrow \mathsf{G}_3$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ with*

$$\left|\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{unlink}}_{\mathsf{AOWFGen},\mathcal{B}}(\lambda).$$

The proof is a straightforward reduction to the unlinkability property of $\mathsf{AOWFGen}$.

The next hybrid G_4 is identical to G_5, except that also if $b = 1$ the adversary gets (ξ, π) where (as before) $\xi := f_{\mathsf{pp}}((x_1, \pi'_1), \ldots, (x_n, \pi'_n), r)$ for $r \xleftarrow{\$} \mathcal{X}_{\mathsf{pp}}$ and $\pi \leftarrow S_2(\mathsf{crs}, \mathsf{td}, T'_x(x'_1, \ldots, x'_n))$ (instead of computing it honestly via $\mathsf{SNARK.Prove}$).

Lemma 5 ($\mathsf{G}_3 \rightsquigarrow \mathsf{G}_4$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ with*

$$\left|\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{B},\mathsf{SNARK}}(\lambda).$$

The proof is a straightforward reduction to the zero-knowledge property of SNARK.

In the next hybrid G_5, if $b = 1$ we pick $\xi \xleftarrow{\$} \mathcal{X}_{\mathsf{pp}}$ instead of $\xi := f_{\mathsf{pp}}((x_1, \pi'_1), \ldots, (x_n, \pi'_n), r)$ for $r \xleftarrow{\$} \mathcal{X}_{\mathsf{pp}}$

Lemma 6 ($\mathsf{G}_4 \rightsquigarrow \mathsf{G}_5$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ with*

$$\left|\Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{unlink}}_{\mathsf{AOWFGen},\mathcal{B}}(\lambda).$$

The proof is a straightforward reduction to the unlinkability property of $\mathsf{AOWFGen}$.

Lemma 7 (G_5)**.**

$$\Pr[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

Proof. If $T_x(x_1, \ldots, x_n) \neq T'_x(x'_1, \ldots, x'_n)$, the adversary violated the non-trivial win condition and thus wins with probability $1/2$ by the definition of the game. Otherwise, the view of the adversary is statistically independent of b and thus it wins with probability $1/2$. □

Combining Lemmata 2–7 yields Theorem 6. □

Acknowledgments. Christoph Striecks' research on this work was co-funded by the European Union. Views and opinions expressed are however those of the author only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. Moreover, Christoph Striecks was supported by the Internet Privatstiftung Austria (IPA) under the netidee SCIENCE program under FWF project number P 31621-N38 ("PROFET"). Daniele Venturi is member of the Gruppo Nazionale Calcolo Scientifico Istituto Nazionale di Alta Matematica (GNCS-INdAM). His research was supported by project SERICS (PE00000014) and by project PARTHENON

(B53D23013000006), under the MUR National Recovery and Resilience Plan funded by the European Union—NextGenerationEU, and by project BEAT, funded by Sapienza University of Rome. Jesper Buus Nielsen was funded by the Danish Independent Research Council under Grant-ID DFF-3103-00077B (CryptoDigi). Jesper Buus Nielsen is grateful to Mathias Hall-Andersen for helpful input to the project.

References

[1] Adi Akavia, Craig Gentry, Shai Halevi, and Margarita Vald. "Achievable CCA2 Relaxation for Homomorphic Encryption". In: *TCC 2022, Part II*. Ed. by Eike Kiltz and Vinod Vaikuntanathan. Vol. 13748. LNCS. Springer, Cham, Nov. 2022, pp. 70–99. DOI: 10.1007/978-3-031-22365-5_3.

[2] Prabhanjan Ananth, Aloni Cohen, and Abhishek Jain. "Cryptography with Updates". In: *EUROCRYPT 2017, Part II*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. LNCS. Springer, Cham, 2017, pp. 445–472. DOI: 10.1007/978-3-319-56614-6_15.

[3] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. "P-signatures and Noninteractive Anonymous Credentials". In: *TCC 2008*. Ed. by Ran Canetti. Vol. 4948. LNCS. Springer, Berlin, Heidelberg, Mar. 2008, pp. 356–374. DOI: 10.1007/978-3-540-78524-8_20.

[4] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. "Incremental Cryptography: The Case of Hashing and Signing". In: *CRYPTO'94*. Ed. by Yvo Desmedt. Vol. 839. LNCS. Springer, Berlin, Heidelberg, Aug. 1994, pp. 216–233. DOI: 10.1007/3-540-48658-5_22.

[5] Mihir Bellare and Shafi Goldwasser. "New Paradigms for Digital Signatures and Message Authentication Based on Non-Interactive Zero Knowledge Proofs". In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, New York, Aug. 1990, pp. 194–211. DOI: 10.1007/0-387-34805-0_19.

[6] Mihir Bellare and Amit Sahai. "Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization". In: *CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, Berlin, Heidelberg, Aug. 1999, pp. 519–536. DOI: 10.1007/3-540-48405-1_33.

[7] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. "The Hunting of the SNARK". In: *Journal of Cryptology* 30.4 (Oct. 2017), pp. 989–1066. DOI: 10.1007/s00145-016-9241-9.

[8] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. "Recursive composition and bootstrapping for SNARKS and proof-carrying data". In: *45th ACM STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 111–120. DOI: 10.1145/2488608.2488623.

[9] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. "On the existence of extractable one-way functions". In: *46th ACM STOC*. Ed. by David B. Shmoys. ACM Press, 2014, pp. 505–514. DOI: 10.1145/2591796.2591859.

[10] Manuel Blum, Paul Feldman, and Silvio Micali. "Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract)". In: *20th ACM STOC*. ACM Press, May 1988, pp. 103–112. DOI: 10.1145/62212.62222.

[11] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. "Verifiable Delay Functions". In: *CRYPTO 2018, Part I*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10991. LNCS. Springer, Cham, Aug. 2018, pp. 757–788. DOI: 10.1007/978-3-319-96884-1_25.

[12] Dan Boneh, Saba Eskandarian, Sam Kim, and Maurice Shih. "Improving Speed and Security in Updatable Encryption Schemes". In: *ASIACRYPT 2020, Part III*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12493. LNCS. Springer, Cham, Dec. 2020, pp. 559–589. DOI: 10.1007/978-3-030-64840-4_19.

[13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. "Key Homomorphic PRFs and Their Applications". In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Berlin, Heidelberg, Aug. 2013, pp. 410–428. DOI: 10.1007/978-3-642-40041-4_23.

[14] Dan Boneh, Gil Segev, and Brent Waters. "Targeted malleability: homomorphic encryption for restricted computations". In: *ITCS 2012*. Ed. by Shafi Goldwasser. ACM, Jan. 2012, pp. 350–366. DOI: 10.1145/2090236.2090264.

[15] Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang. "Fast and Secure Updatable Encryption". In: *CRYPTO 2020, Part I*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12170. LNCS. Springer, Cham, Aug. 2020, pp. 464–493. DOI: 10.1007/978-3-030-56784-2_16.

[16] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. "Relaxing Chosen-Ciphertext Security". In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Berlin, Heidelberg, Aug. 2003, pp. 565–582. DOI: 10.1007/978-3-540-45146-4_33.

[17] Ran Canetti, Srinivasan Raghuraman, Silas Richelson, and Vinod Vaikuntanathan. "Chosen-Ciphertext Secure Fully Homomorphic Encryption". In: *PKC 2017, Part II*. Ed. by Serge Fehr. Vol. 10175. LNCS. Springer, Berlin, Heidelberg, Mar. 2017, pp. 213–240. DOI: 10.1007/978-3-662-54388-7_8.

[18] Suvradip Chakraborty, Stefan Dziembowski, and Jesper Buus Nielsen. "Reverse Firewalls for Actively Secure MPCs". In: *CRYPTO 2020, Part II*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Cham, Aug. 2020, pp. 732–762. DOI: 10.1007/978-3-030-56880-1_26.

[19] Suvradip Chakraborty, Chaya Ganesh, Mahak Pancholi, and Pratik Sarkar. "Reverse Firewalls for Adaptively Secure MPC Without Setup". In: *ASIACRYPT 2021, Part II*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13091. LNCS. Springer, Cham, Dec. 2021, pp. 335–364. DOI: 10.1007/978-3-030-92075-3_12.

[20] Suvradip Chakraborty, Chaya Ganesh, and Pratik Sarkar. "Reverse Firewalls for Oblivious Transfer Extension and Applications to Zero-Knowledge". In: *EUROCRYPT 2023, Part I*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14004. LNCS. Springer, Cham, Apr. 2023, pp. 239–270. DOI: 10.1007/978-3-031-30545-0_9.

[21] Suvradip Chakraborty, Lorenzo Magliocco, Bernardo Magri, and Daniele Venturi. *Key Exchange in the Post-Snowden Era: UC Secure Subversion-Resilient PAKE*. Cryptology ePrint Archive, Paper 2023/1827 (to appear at Asiacrypt 2024). https://eprint.iacr.org/2023/1827. 2024. URL: https://eprint.iacr.org/2023/1827.

[22] Suvradip Chakraborty, Bernardo Magri, Jesper Buus Nielsen, and Daniele Venturi. "Universally Composable Subversion-Resilient Cryptography". In: *EUROCRYPT 2022, Part I*. Ed. by Orr Dunkelman and Stefan Dziembowski. Vol. 13275. LNCS. Springer, Cham, 2022, pp. 272–302. DOI: 10.1007/978-3-031-06944-4_10.

[23] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. "Malleable Proof Systems and Applications". In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Berlin, Heidelberg, Apr. 2012, pp. 281–300. DOI: 10.1007/978-3-642-29011-4_18.

[24] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. "Succinct Malleable NIZKs and an Application to Compact Shuffles". In: *TCC 2013*. Ed. by Amit Sahai. Vol. 7785. LNCS. Springer, Berlin, Heidelberg, Mar. 2013, pp. 100–119. DOI: 10.1007/978-3-642-36594-2_6.

[25] Huanhuan Chen, Yao Jiang Galteland, and Kaitai Liang. "CCA-1 Secure Updatable Encryption with Adaptive Security". In: *ASIACRYPT 2023, Part V*. Ed. by Jian Guo and Ron Steinfeld. Vol. 14442. LNCS. Springer, Singapore, Dec. 2023, pp. 374–406. DOI: 10.1007/978-981-99-8733-7_12.

[26] Long Chen, Yanan Li, and Qiang Tang. "CCA Updatable Encryption Against Malicious Re-encryption Attacks". In: *ASIACRYPT 2020, Part III*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12493. LNCS. Springer, Cham, Dec. 2020, pp. 590–620. DOI: 10.1007/978-3-030-64840-4_20.

[27] Jiaqi Cheng and Rishab Goyal. *Boosting SNARKs and Rate-1 Barrier in Arguments of Knowledge*. Cryptology ePrint Archive, Report 2024/1603. https://eprint.iacr.org/2024/1603. 2024.

[28] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. "A Secure and Optimally Efficient Multi-Authority Election Scheme". In: *EUROCRYPT'97*. Ed. by Walter Fumy. Vol. 1233. LNCS. Springer, Berlin, Heidelberg, May 1997, pp. 103–118. DOI: 10.1007/3-540-69053-0_9.

[29] Alfredo De Santis and Giuseppe Persiano. "Zero-Knowledge Proofs of Knowledge Without Interaction (Extended Abstract)". In: *33rd FOCS*. IEEE Computer Society Press, Oct. 1992, pp. 427–436. DOI: 10.1109/SFCS.1992.267809.

[30] Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. "Message Transmission with Reverse Firewalls—Secure Communication on Corrupted Machines". In: *CRYPTO 2016, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. LNCS. Springer, Berlin, Heidelberg, Aug. 2016, pp. 341–372. DOI: 10.1007/978-3-662-53018-4_13.

[31] Danny Dolev, Cynthia Dwork, and Moni Naor. "Non-malleable Cryptography". In: *SIAM Journal on Computing* 30.2 (2000), pp. 391–437. DOI: 10.1137/S0097539795291562.

[32] Danny Dolev, Cynthia Dwork, and Moni Naor. "Nonmalleable Cryptography". In: *SIAM Journal on Computing* 30.2 (2000), pp. 391–437.

[33] Cynthia Dwork and Moni Naor. "Pricing via Processing or Combatting Junk Mail". In: *CRYPTO'92*. Ed. by Ernest F. Brickell. Vol. 740. LNCS. Springer, Berlin, Heidelberg, Aug. 1993, pp. 139–147. DOI: 10.1007/3-540-48071-4_10.

[34] Cynthia Dwork, Moni Naor, and Omer Reingold. "Immunizing Encryption Schemes from Decryption Errors". In: *EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. LNCS. Springer, Berlin, Heidelberg, May 2004, pp. 342–360. DOI: 10.1007/978-3-540-24676-3_21.

[35] Adam Everspaugh, Kenneth G. Paterson, Thomas Ristenpart, and Samuel Scott. "Key Rotation for Authenticated Encryption". In: *CRYPTO 2017, Part III*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10403. LNCS. Springer, Cham, Aug. 2017, pp. 98–129. DOI: 10.1007/978-3-319-63697-9_4.

[36] Andrés Fabrega, Ueli Maurer, and Marta Mularczyk. *A Fresh Approach to Updatable Symmetric Encryption*. Cryptology ePrint Archive, Report 2021/559. https://eprint.iacr.org/2021/559. 2021.

[37] Antonio Faonio, Dario Fiore, Javier Herranz, and Carla Ràfols. "Structure-Preserving and Re-randomizable RCCA-Secure Public Key Encryption and Its Applications". In: *ASIACRYPT 2019, Part III*. Ed. by Steven D. Galbraith and Shiho Moriai. Vol. 11923. LNCS. Springer, Cham, Dec. 2019, pp. 159–190. DOI: 10.1007/978-3-030-34618-8_6.

[38] Prastudy Fauzi, Martha Norberg Hovd, and Håvard Raddum. "On the IND-CCA1 Security of FHE Schemes". In: *Cryptography* 6.1 (2022). ISSN: 2410-387X. DOI: 10.3390/cryptography6010013.

[39] Yao Jiang Galteland and Jiaxin Pan. "Backward-Leak Uni-Directional Updatable Encryption from (Homomorphic) Public Key Encryption". In: *PKC 2023, Part II*. Ed. by Alexandra Boldyreva and Vladimir Kolesnikov. Vol. 13941. LNCS. Springer, Cham, May 2023, pp. 399–428. DOI: 10.1007/978-3-031-31371-4_14.

[40] Chaya Ganesh, Bernardo Magri, and Daniele Venturi. "Cryptographic Reverse Firewalls for Interactive Proof Systems". In: *ICALP 2020*. Ed. by Artur Czumaj, Anuj Dawar, and Emanuela Merelli. Vol. 168. LIPIcs. Schloss Dagstuhl, July 2020, 55:1–55:16. DOI: 10.4230/LIPIcs.ICALP.2020.55.

[41] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions". In: *40th ACM STOC*. Ed. by Richard E. Ladner and Cynthia Dwork. ACM Press, May 2008, pp. 197–206. DOI: 10.1145/1374376.1374407.

[42] Craig Gentry and Daniel Wichs. "Separating succinct non-interactive arguments from all falsifiable assumptions". In: *43rd ACM STOC*. Ed. by Lance Fortnow and Salil P. Vadhan. ACM Press, June 2011, pp. 99–108. DOI: 10.1145/1993636.1993651.

[43] Jens Groth. "On the Size of Pairing-Based Non-interactive Arguments". In: *EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. LNCS. Springer, Berlin, Heidelberg, May 2016, pp. 305–326. DOI: 10.1007/978-3-662-49896-5_11.

[44] Jens Groth. "Rerandomizable and Replayable Adaptive Chosen Ciphertext Attack Secure Cryptosystems". In: *TCC 2004*. Ed. by Moni Naor. Vol. 2951. LNCS. Springer, Berlin, Heidelberg, Feb. 2004, pp. 152–170. DOI: 10.1007/978-3-540-24638-1_9.

[45] Jens Groth. "Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures". In: *ASIACRYPT 2006*. Ed. by Xuejia Lai and Kefei Chen. Vol. 4284. LNCS. Springer, Berlin, Heidelberg, Dec. 2006, pp. 444–459. DOI: 10.1007/11935230_29.

[46] Jens Groth and Amit Sahai. "Efficient Non-interactive Proof Systems for Bilinear Groups". In: *EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, Berlin, Heidelberg, Apr. 2008, pp. 415–432. DOI: 10.1007/978-3-540-78967-3_24.

[47] Yao Jiang. "The Direction of Updatable Encryption Does Not Matter Much". In: *ASIACRYPT 2020, Part III*. Ed. by Shiho Moriai and Huaxiong Wang. Vol. 12493. LNCS. Springer, Cham, Dec. 2020, pp. 529–558. DOI: 10.1007/978-3-030-64840-4_18.

[48] Stefan Kölbl, Anvita Pandit, Rafael Misoczki, and Sophie Schmieg. "Crypto Agility and Post-Quantum Cryptography at Google". In: *Real-World Crypto Symposium* (2023). URL: https://www.youtube.com/watch?v=IAOWRO9Qn10&t=107s.

[49] Michael Klooß, Anja Lehmann, and Andy Rupp. "(R)CCA Secure Updatable Encryption with Integrity Protection". In: *EUROCRYPT 2019, Part I*. Ed. by

Yuval Ishai and Vincent Rijmen. Vol. 11476. LNCS. Springer, Cham, May 2019, pp. 68–99. DOI: 10.1007/978-3-030-17653-2_3.

[50] Christiane Kuhn, Dennis Hofheinz, Andy Rupp, and Thorsten Strufe. "Onion Routing with Replies". In: *ASIACRYPT 2021, Part II*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Vol. 13091. LNCS. Springer, Cham, Dec. 2021, pp. 573–604. DOI: 10.1007/978-3-030-92075-3_20.

[51] Anja Lehmann and Björn Tackmann. "Updatable Encryption with Post-Compromise Security". In: *EUROCRYPT 2018, Part III*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10822. LNCS. Springer, Cham, 2018, pp. 685–716. DOI: 10.1007/978-3-319-78372-7_22.

[52] Françoise Levy-dit-Vehel and Maxime Roméas. *A Composable Look at Updatable Encryption*. Cryptology ePrint Archive, Report 2021/538. https://eprint.iacr.org/2021/538. 2021.

[53] Jake Loftus, Alexander May, Nigel P. Smart, and Frederik Vercauteren. "On CCA-Secure Somewhat Homomorphic Encryption". In: *SAC 2011*. Ed. by Ali Miri and Serge Vaudenay. Vol. 7118. LNCS. Springer, Berlin, Heidelberg, Aug. 2012, pp. 55–72. DOI: 10.1007/978-3-642-28496-0_4.

[54] Mark Manulis and Jérôme Nguyen. "Fully Homomorphic Encryption Beyond IND-CCA1 Security: Integrity Through Verifiability". In: *EUROCRYPT 2024, Part II*. Ed. by Marc Joye and Gregor Leander. Vol. 14652. LNCS. Springer, Cham, May 2024, pp. 63–93. DOI: 10.1007/978-3-031-58723-8_3.

[55] Jonas Meers and Doreen Riepel. "CCA Secure Updatable Encryption from Non-mappable Group Actions". In: *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part I*. Ed. by Markku-Juhani Saarinen and Daniel Smith-Tone. Springer, Cham, June 2024, pp. 137–169. DOI: 10.1007/978-3-031-62743-9_5.

[56] Peihan Miao, Sikhar Patranabis, and Gaven J. Watson. "Unidirectional Updatable Encryption and Proxy Re-encryption from DDH". In: *PKC 2023, Part II*. Ed. by Alexandra Boldyreva and Vladimir Kolesnikov. Vol. 13941. LNCS. Springer, Cham, May 2023, pp. 368–398. DOI: 10.1007/978-3-031-31371-4_13.

[57] Ilya Mironov and Noah Stephens-Davidowitz. "Cryptographic Reverse Firewalls". In: *EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. LNCS. Springer, Berlin, Heidelberg, Apr. 2015, pp. 657–686. DOI: 10.1007/978-3-662-46803-6_22.

[58] Moni Naor. "On Cryptographic Assumptions and Challenges (Invited Talk)". In: *CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, Berlin, Heidelberg, Aug. 2003, pp. 96–109. DOI: 10.1007/978-3-540-45146-4_6.

[59] Moni Naor and Moti Yung. "Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks". In: *22nd ACM STOC*. ACM Press, May 1990, pp. 427–437. DOI: 10.1145/100216.100273.

[60] Ryo Nishimaki. "The Direction of Updatable Encryption Does Matter". In: *PKC 2022, Part II*. Ed. by Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe. Vol. 13178. LNCS. Springer, Cham, Mar. 2022, pp. 194–224. DOI: 10.1007/978-3-030-97131-1_7.

[61] Olivier Pereira and Ronald L. Rivest. "Marked Mix-Nets". In: *FC 2017 Workshops*. Ed. by Michael Brenner, Kurt Rohloff, Joseph Bonneau, Andrew Miller, Peter Y. A. Ryan, Vanessa Teague, Andrea Bracciali, Massimiliano Sala, Federico Pintore, and Markus Jakobsson. Vol. 10323. LNCS. Springer, Cham, Apr. 2017, pp. 353–369. DOI: 10.1007/978-3-319-70278-0_22.

[62] Manoj Prabhakaran and Mike Rosulek. "Homomorphic Encryption with CCA Security". In: *ICALP 2008, Part II*. Ed. by Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz. Vol. 5126. LNCS. Springer, Berlin, Heidelberg, July 2008, pp. 667–678. DOI: 10.1007/978-3-540-70583-3_54.

[63] Manoj Prabhakaran and Mike Rosulek. "Rerandomizable RCCA Encryption". In: *CRYPTO 2007*. Ed. by Alfred Menezes. Vol. 4622. LNCS. Springer, Berlin, Heidelberg, Aug. 2007, pp. 517–534. DOI: 10.1007/978-3-540-74143-5_29.

[64] Charles Rackoff and Daniel R. Simon. "Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack". In: *CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. LNCS. Springer, Berlin, Heidelberg, Aug. 1992, pp. 433–444. DOI: 10.1007/3-540-46766-1_35.

[65] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *37th ACM STOC*. Ed. by Harold N. Gabow and Ronald Fagin. ACM Press, May 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.

[66] Ronald L. Rivest, Adi Shamir, and David A. Wagner. *Time-lock puzzles and timed-release Crypto*. Tech. rep. (MIT/LCS/TR)-684. MIT, Feb. 1996. URL: https://people.csail.mit.edu/rivest/pubs/RSW96.pdf.

[67] Amit Sahai. "Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security". In: *40th FOCS*. IEEE Computer Society Press, Oct. 1999, pp. 543–553. DOI: 10.1109/SFFCS.1999.814628.

[68] Daniel Slamanig and Christoph Striecks. "Revisiting Updatable Encryption: Controlled Forward Security, Constructions and a Puncturable Perspective". In: *TCC 2023, Part II*. Ed. by Guy N. Rothblum and Hoeteck Wee. Vol. 14370. LNCS. Springer, Cham, 2023, pp. 220–250. DOI: 10.1007/978-3-031-48618-0_8.

[69] Paul Valiant. "Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency". In: *TCC 2008*. Ed. by Ran Canetti. Vol. 4948. LNCS. Springer, Berlin, Heidelberg, Mar. 2008, pp. 1–18. DOI: 10.1007/978-3-540-78524-8_1.

[70] Mikhail Volkhov. "Malleable Zero-Knowledge Proofs and Applications". PhD thesis. University of Edinburgh, 2023.

$\underline{\mathsf{Exp}^{\mathsf{ldp}}_{\mathcal{A},\Pi=(\mathsf{CRSGen},\mathsf{Prove},\mathsf{Verify})}(\lambda)}$:
$b \xleftarrow{\$} \{0,1\}$
$(\mathsf{crs},\mathsf{td}) \leftarrow S_1(1^\lambda)$
$(\mathsf{st},(x_i,\pi_i)_{1\le i\le n}, T=(T_x,T_w),(x'_i,\pi'_i)_{1\le i\le n}, T'=(T'_x,T'_w)) \xleftarrow{\$} \mathcal{A}_1(\mathsf{crs})$
if $\exists i\in[n] : (\mathsf{Verify}(\mathsf{crs},x_i,\pi_i) \stackrel{?}{=} 0 \vee \mathsf{Verify}(\mathsf{crs},x'_i,\pi'_i) \stackrel{?}{=} 0) \vee T_x(x_1,\ldots,x_n) \neq T'_x(x'_1,\ldots,x'_n) \vee \max_{i\in[n]} \mathsf{Level}(\pi_i) \neq \max_{i\in[n]} \mathsf{Level}(\pi'_i)$ **then**
  return $b' \xleftarrow{\$} \{0,1\}$
if $b=0$ **then**
  $\pi \leftarrow \mathsf{ZKEval}(\mathsf{crs},T,(x_i,\pi_i)_{1\le i\le n})$
else
  $\pi \leftarrow \mathsf{ZKEval}(\mathsf{crs},T',(x'_i,\pi'_i)_{1\le i\le n})$
$b' \leftarrow \mathcal{A}_2(\mathsf{st},\pi)$
return $b \stackrel{?}{=} b'$

Fig. 7. The leveled derivation privacy experiment for a NIZK.

A Construction with counters

In this section we present a construction that uses counters instead of the AOWF and and achieves a weaker notion of derivation privacy, defined below. We stress that this construction serves as a warmup for our main construction, and appears already (with insignificant differences) in [24].

We introduce a relaxation of derivation privacy where arguments are allowed to leak the number of applied transformations. For this notion we assume the existence of a PPT algorithm Level that takes an argument π and outputs a number corresponding to the number of applied transformation (the "level"). We can ensure that such an algorithm exists, by including a counter of the number of applied transformations in the arguments. Formally, we require:

- $\mathsf{Level}(\pi) = 0$ for all arguments π in the image of Prove.
- $\mathsf{Level}(\pi) = \max_{i\in[n]} \mathsf{Level}(\pi_i) + 1$ for all arguments $\pi \xleftarrow{\$} \mathsf{ZKEval}(\mathsf{crs},T,(x_i,\pi_i)_{1\le i\le n})$

We say that an argument system is B-bounded malleable if it satisfies Definition 9 except that we relax the additional completeness requirement to hold only for all $(x_i,\pi_i)_{1\le i\le n}$ with $\mathsf{Level}(\pi_i) \le B-1$ for all $i\in[n]$.

Definition 17 (Leveled derivation privacy). *A malleable NIZK* $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{ZKEval})$ *with associated simulator* $S=(S_1,S_2)$ *is* leveled derivation private *if for every PPT adversary* $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)$ *we have*

$$\mathsf{Adv}^{\mathsf{ldp}}_{\mathcal{A},\Pi}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{ldp}}_{\mathcal{A},\Pi}(\lambda)=1] - \frac{1}{2}\right| \le \mathsf{negl}(\lambda)$$

for a negligible function negl *where the experiment is defined in Figure 7.*

In Figure 8 we give a construction of a $B(\lambda)$-bounded malleable NIZK Π_{ctr} (where B can be any a priori fixed polynomial) that achieves leveled derivation

privacy for any NP-relation $\mathcal{R}$ and any allowable set of transformations $\mathcal{T}$ for $\mathcal{R}$. We stress that this construction serves as a warmup for our main construction, and appears already (with insignificant differences) in [24]. Let $n_{\max}$ be the maximum arity of the transformations in $\mathcal{T}$. The construction uses a zk-SNARK for NP. We require that one of the following restrictions hold:

- $n_{\max}$ and $B(\lambda)$ are both constants (independent of the security parameter).
- $B(\lambda)n_{\max}^{B(\lambda)}$ grows polynomially in λ (that includes the cases that $n_{\max}$ is constant and $B(\lambda)$ grows logarithmic in λ or $n_{\max} = 1$ and $B(\lambda)$ is polynomial), the underlying zk-SNARK offers fast extraction and there is a universal polynomial bound on the time it takes to evaluate T_x and T_w for all $(T_x, T_w) \in \mathcal{T}$.

If $B(\lambda)$ is a constant, we can also use a preprocessing SNARK (where the CRS size grows with the size of the statements) and generate a separate CRS for each level.

The construction makes use of the following time bounds:

- $\tau_0(\lambda, m)$ is a time bound on the run time of the NP-verifier M_0 for $\mathcal{R}$ and for statements of length m. The argument λ is ignored here.
- For $\ell \in \mathbb{N}_+ : \tau_\ell(\lambda, m) := \mathsf{poly}(\log B, m) + n_{\max} \cdot \mathsf{poly}'(\lambda, m, \log(B), \log(\tau_{\ell-1}(\lambda, m)))$ for suitable polynomials poly and poly' defined below.

We argue that this is an upper bound on the runtime of M_ℓ. The case $\ell = 0$ is trivial. For $\ell \in \mathbb{N}_+$, the Turing machine M_ℓ first checks whether $T \in \mathcal{T}$ and whether $x = T_x(x_1, \ldots, x_n)$. These checks can be performed in time polynomial in $|T| + |x_1| + \cdots + |x_n|$ which is, by definition of an admissible transformation, also polynomial in $|x|$. The check that $\ell_i \geq \ell$ can be done in time $\mathcal{O}(\log \ell) \subseteq \mathcal{O}(\log B)$ and this check is done $n \leq |x_1| + \cdots + |x_n|$ times (since we excluded empty statements). Thus, all checks except for the SNARK verifications can be done in time $\mathsf{poly}(\log B, |x|)$ for a suitable, fixed polynomial poly.

The i-th SNARK verification is only executed if the previous checks passed. It then takes time polynomial in λ, $|x_i|$, $\log(\ell_i) \leq B$, and $\log(\tau_{\ell_i}(\lambda, |x_i|))$. Since $|x_i| \leq \mathsf{poly}''(|x|)$ for a polynomial poly'' by Definition 7 and $\ell_i \leq \ell - 1$, this can be upper bounded by $\mathsf{poly}'(\lambda, |x|, \log(B), \log(\tau_{\ell-1}(\lambda, |x|)))$ for a suitable polynomial poly'.

We prove that τ_ℓ for $\ell \in \mathbb{N}_0$ grows asymptotically at most like a polynomial by induction over ℓ. It follows immediately from the definition of τ_0 that $\tau_0(\lambda, m)$ thus grows polynomial in m (and λ). Now fix $\ell \in \mathbb{N}_+$ and assume that all $\tau_{\ell'}$ for $\ell' < \ell$ grow asymptotically at most like a polynomial. Then $\tau_{\ell'}(\lambda, m) \leq 2^{\lambda \cdot m}$ if λ is sufficiently large and with this bound we get

$$\begin{aligned}
\tau_\ell(\lambda, m) &= \mathsf{poly}(\log B, |x|) + \sum_{i=1}^{n} \mathsf{poly}'(\lambda, |x_i|, \log(B), \log(\tau_{\ell_i}(\lambda, |x_i|))) \\
&\leq \mathsf{poly}(\log B, |x|) + \sum_{i=1}^{n} \mathsf{poly}'(\lambda, |x_i|, \log(B), \lambda \cdot m) \\
&\leq \mathsf{poly}(\log B, |x|) + n \cdot \mathsf{poly}'(\lambda, |x_1| + \cdots + |x_n|, \log(B), \lambda \cdot m).
\end{aligned}$$

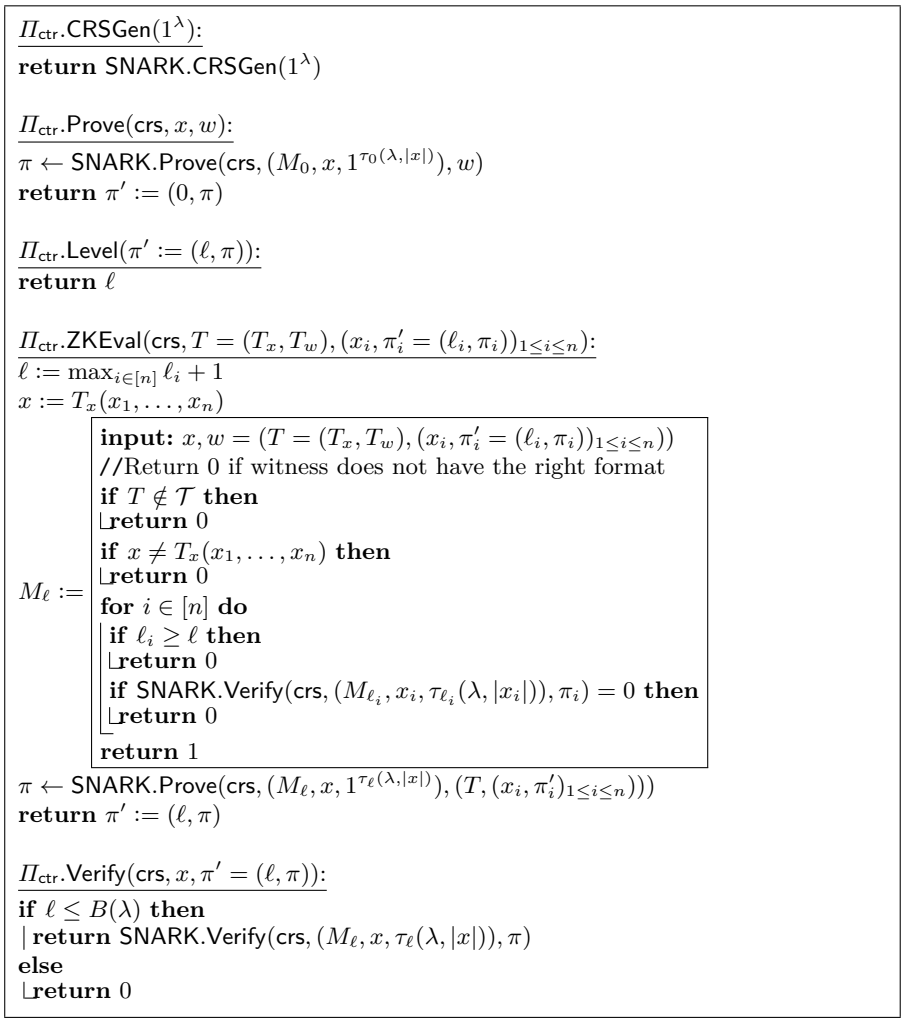

Fig. 8. Our malleable NIZK construction $\Pi_{\mathsf{ctr}} = (\Pi_{\mathsf{ctr}}.\mathsf{CRSGen}, \Pi_{\mathsf{ctr}}.\mathsf{Prove}, \Pi_{\mathsf{ctr}}.\mathsf{Verify}, \Pi_{\mathsf{ctr}}.\mathsf{ZKEval}, \Pi_{\mathsf{ctr}}.\mathsf{Level})$ with counters from a zk-SNARK $\mathsf{SNARK} = (\mathsf{SNARK.CRSGen}, \mathsf{SNARK.Prove}, \mathsf{SNARK.Verify})$. The construction works for any NP relation $\mathcal{R}$ where M_0 is a Turing machine verifying the NP relation and any set of admissible transformations $\mathcal{T}$. The time bounds τ_ℓ are defined later.

Since, by definition of an admissible transformation, n grows polynomial in λ, $\tau_\ell(\lambda, m)$ grows polynomially in λ and m.

Theorem 7 (Completeness). *The malleable NIZK Π_{ctr} is complete if the underlying SNARK is complete.*

Proof. Let $\mathsf{crs} \xleftarrow{\$} \Pi_{\mathsf{ctr}}.\mathsf{CRSGen}(1^\lambda)$. First, we show that proofs generated directly via $\Pi_{\mathsf{ctr}}.\mathsf{Prove}$ verify. Let $(x, w) \in \mathcal{R}$ and $\pi' = (0, \pi) \leftarrow \Pi_{\mathsf{ctr}}.\mathsf{Prove}(\mathsf{crs}, x, w)$. Since M_0 is an NP-Verifier for $\mathcal{R}$ and halts within $\tau_0(\lambda, |x|)$ steps, $(M_0, x, \tau_0(\lambda, |x|)) \in \mathcal{R}_{\mathcal{U}}$. Thus, by completeness of the SNARK,

$$\Pr[\Pi_{\mathsf{ctr}}.\mathsf{Verify}(\mathsf{crs}, x, \pi') = 1] \geq 1 - \mathsf{negl}(\lambda)$$

for a negligible function negl.

Next, let $T = (T_x, T_w) \in \mathcal{T}$ be an n-ary admissible transformation and $\pi' := (\ell, \pi) \xleftarrow{\$} \Pi_{\mathsf{ctr}}.\mathsf{ZKEval}(\mathsf{crs}, T = (T_x, T_w), (x_i, \pi'_i = (\ell_i, \pi_i))_{1 \leq i \leq n})$. Let us assume that for all $i \in [n]$ $\mathsf{Verify}(\mathsf{crs}, x_i, \pi'_i) = 1$ holds with overwhelming probability. Then for $\pi'_i = (\ell_i, \pi_i)$ we have $\mathsf{SNARK.Verify}(\mathsf{crs}, (M_{\ell_i}, x_i, \tau_{\ell_i}(\lambda, |x_i|)), \pi_i) = 1$ holds for all $i \in [n]$ with overwhelming probability. Furthermore, $\ell_i < \ell = \max_{i \in [n]} \ell_i + 1$. This shows $(M, x, 1^{\tau_\ell(\lambda, |x|)}), (T, (x_i, \pi'_i)_{1 \leq i \leq n}) \in \mathcal{U}_{\mathcal{R}}$ and thus, by completeness of the SNARK,

$$\Pr[\Pi_{\mathsf{ctr}}.\mathsf{Verify}(\mathsf{crs}, T_x(x_1, \ldots, x_n), \pi') = 1] \geq 1 - \mathsf{negl}(\lambda)$$

for a negligible function negl. □

Theorem 8 (Full succinctness). *The malleable NIZK Π_{ctr} is fully succinct if the underlying SNARK is fully succinct.*

Proof. A proof $\pi' = (\ell, \pi)$ of Π_{ctr} consists of a proof π for the underlying SNARK, that has a fixed polynomial length, and the counter ℓ, that has a fixed logarithmic length. □

Theorem 9 (Soundness). *The malleable NIZK Π_{ctr} is controlled-malleable simulation extractable if the underlying SNARK is simulation extractable.*

Proof. Let $\mathcal{A}$ be an algorithm that inputs crs, has access to a proving oracle $\textsc{Prove}(x)$ that generates simulated proofs for arbitrary statements x and outputs (x, π). We recursively show existence of the extractor $\mathcal{E}$. The extractor inputs crs, $\mathcal{A}$'s randomness r, and the list of simulated proofs Q_{sim}. If $(x, \_) \in Q_{\mathsf{sim}}$, the extractor outputs the explanation $E := \{(x, \perp)\}$. Otherwise, it runs the extractor for the underlying SNARK for $\mathcal{A}$ on its own inputs. If $\Pi_{\mathsf{ctr}}.\mathsf{Level}(\pi) = 0$, this extractor outputs with overwhelming probability a witness w for x and $\mathcal{E}$ also just outputs the explanation $E := \{(x, w)\}$. If $\Pi_{\mathsf{ctr}}.\mathsf{Level}(\pi) = \ell \in \mathbb{N}_+$, this extractor outputs $w = (T = (T_x, T_w), (x_i, \pi'_i = (\ell_i, \pi_i))_{1 \leq i \leq n})$ such that $M_\ell(x, w) = 1$. That implies $T \in \mathcal{T}$, $x = T_x(x_1, \ldots, x_n)$, and for all $i \in [n]$ $\ell_i < \ell$ and $\mathsf{SNARK.Verify}(\mathsf{crs}, (M_{\ell_i}, x_i, \tau_{\ell_i}(\lambda, |x_i|)), \pi_i) = 1$. Now let $\mathcal{A}_i$ be an algorithm that proceeds exactly like $\mathcal{E}$ up to this point and then outputs $(x_i, (\ell_i, \pi_i))$. For

each $i \in [n]$, we can recursively assume existence of an extractor $\mathcal{E}_i$ that extracts an explanation E_i for x_i. The extractor $\mathcal{E}$ runs all these extractors to compute E_i. The extractor then outputs $E := \bigcup_{i=1}^{n} E_i \cup \{(x, (T, (x_i)_{1 \le i \le n}))\}$.

First, we show that the extractor outputs a valid explanation. In the base case where the statement is a simulated statement, this is trivial. In the other base case (where $\Pi_{\mathsf{ctr}}.\mathsf{Level}(\pi) = 0$), this follows directly from knowledge soundness of the underlying SNARK. In the recursive step, if all the extractors $\mathcal{E}_i$ output a valid explanation E_i for x_i, the explanation $E := \bigcup_{i=1}^{n} E_i \cup \{(x, (T = (T_x, T_w), (x_i)_{1 \le i \le n}))\}$ is a valid explanation for $x = T_x(x_1, \ldots, x_n)$ since T is an admissible transformation by the knowledge soundness of the underlying SNARK.

Next, we analyze the runtime of the extractor. The bound on the level $B(\lambda)$ is an upper bound on the recursion depth of the extractor and $n_{\max}$ is an upper bound on the number of recursions.

We first focus on the case where $n_{\max}$ and $B(\lambda)$ are constants. Clearly, the runtime of each extractor in the base case (where $\Pi_{\mathsf{ctr}}.\mathsf{Level}(\pi) = 0$) is polynomial. For the recursive step, the runtime can be expressed as a sum of a constant number of polynomials (the runtime of the $n_{\max}$ recursively invoked extractors plus some polynomial overhead to evaluate the transformations). Since the maximum number of recursions is also a constant, this results in a polynomial runtime.

Next, we focus on the case where $B(\lambda) n_{\max}^{B(\lambda)}$ is a polynomial and the underlying zk-SNARK offers fast extraction. Therefore, let poly be the polynomial such that for every PPT adversary $\mathcal{A}$ the corresponding extractor $\mathcal{E}$ (and also any algorithm that runs $\mathcal{E}$ and then deletes some parts of its output) runs in time $\mathsf{Time}_{\mathcal{E}}(\lambda) \le \mathsf{Time}_{\mathcal{A}}(\lambda) + \mathsf{poly}(\lambda)$.

The above construction uses SNARK extractors for adversaries consisting of the adversary $\mathcal{A}$ and then applying the up to $B(\lambda) - 1$ SNARK extractors to it. Thus, the maximum runtime of each extractor is $\mathsf{Time}_{\mathcal{A}}(\lambda) + B(\lambda)\mathsf{poly}(\lambda)$. This leads to a total runtime of

$$B(\lambda) n_{\max}^{B(\lambda)} (\mathsf{Time}_{\mathcal{A}}(\lambda) + B(\lambda)\mathsf{poly}(\lambda)),$$

which is polynomial in λ if $B(\lambda) n_{\max}^{B(\lambda)}$ is a polynomial λ. □

Theorem 10 (Zero-knowledge). *The malleable proof system Π_{ctr} is zero-knowledge, if the underlying SNARK is zero-knowledge.*

Proof. The reduction is straightforward: It obtains a CRS for the underlying SNARK SNARK and forwards it to the zero-knowledge adversary. Whenever the adversary asks for a proof for a statement x with witness w, the reduction uses the proof oracle for SNARK to obtain a proof π for (x, w) and sends $\pi' := (0, \pi)$ to the adversary. In the end, it outputs the same bit as the adversary.

It is obvious that when the proof oracle returns honest SNARK proofs, the reduction also returns honest Π_{ctr} proofs and when the proof oracle returns proofs simulated without the witness, the reduction also returns simulated proofs. □

Theorem 11 (Leveled derivation privacy). *The malleable proof system Π_{ctr} is leveled derivation private, if the underlying SNARK is zero-knowledge.*

Proof. Throughout the proof, let $(\mathsf{st}, (x_i, (\ell_i, \pi_i))_{1 \le i \le n}, T = (T_x, T_w), (x'_i, (\ell'_i, \pi'_i))_{1 \le i \le n}, T' = (T'_x, T'_w))$ be the output of the first stage of the adversary.

The security reduction proceeds via a hybrid argument. Let G_0 be the real leveled derivation privacy game for Π_{ctr}. The first hybrid G_1 differs from G_0 in one aspect: If $b = 0$, the game samples the SNARK CRS as $\mathsf{crs} \leftarrow \mathsf{SNARK.CRSGen}(1^\lambda)$ (instead of using the simulator).

Lemma 8 ($\mathsf{G}_0 \rightsquigarrow \mathsf{G}_1$). *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ with*

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]\right| \le \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{B},\mathsf{SNARK}}(\lambda)$$

Proof. If $b = 1$, the reduction uses the CRS obtained from the zero-knowledge game for SNARK. The reduction samples everything else by itself, in particular it does not make use of the PROVE oracle. □

The hybrid G_2 is identical to G_1, except that if $b = 0$ the SNARK CRS is sampled via $(\mathsf{crs}, \mathsf{td}) \leftarrow S_1(1^\lambda)$ and the adversary gets (ℓ, π) where $\pi \leftarrow S_2(\mathsf{crs}, \mathsf{td}, T_x(x_1, \ldots, x_n))$ (instead of computing it honestly via SNARK.Prove) and $\ell := \max_{i \in [n]} \ell_i + 1$.

Lemma 9 ($\mathsf{G}_1 \rightsquigarrow \mathsf{G}_2$). *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ with*

$$\left|\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]\right| \le \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{B},\mathsf{SNARK}}(\lambda)$$

The proof is a straightforward reduction to the zero-knowledge property of SNARK.

The hybrid G_3 is identical to G_2, except that also if $b = 1$ the adversary gets (ℓ', π) where $\pi \leftarrow S_2(\mathsf{crs}, \mathsf{td}, T'_x(x'_1, \ldots, x'_n))$ (instead of computing it honestly via SNARK.Prove) and $\ell' := \max_{i \in [n]} \ell'_i + 1$.

Lemma 10 ($\mathsf{G}_2 \rightsquigarrow \mathsf{G}_3$). *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ with*

$$\left|\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1]\right| \le \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{B},\mathsf{SNARK}}(\lambda)$$

The proof is a straightforward reduction to the zero-knowledge property of SNARK.

Lemma 11 (G_3).

$$\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}$$

Proof. If $\ell \neq \ell'$ or $T_x(x_1, \ldots, x_n) \neq T'_x(x'_1, \ldots, x'_n)$, the adversary violated the non-trivial win condition and thus wins with probability $1/2$ by the definition of the game. Otherwise, the view of the adversary is statistically independent of b, and thus they win with probability $1/2$. □

Combining Lemmata 8–11 yields Theorem 11. □

B Reverse Firewalls for NIZKs

In this section, we are interested in a scenario in which the implementation of honest parties can be subverted by the adversary in an undetectable manner; this is sometimes known as specious subversion. A cryptographic reverse firewall (RF) is an external party that can be attached to the honest parties in order to make sure that a subverted implementation does not compromise security. Below, we formalize this notion in the setting of non-interactive argument systems. Our treatment follows closely that of Ganesh et al. [40], which however focus on the more general setting of interactive arguments.

Let $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ be a non-interactive argument system for an NP-relation $\mathcal{R}$, as in Definition 1. A RF for Π is a Turing machine W that takes as input a proof π and outputs a sanitized proof $\widehat{\pi}$; sometimes we refer to the algorithm $\widehat{\mathsf{Prove}}(\mathsf{crs}, x, w) = \mathsf{W}(\mathsf{Prove}(\mathsf{crs}, x, w))$ as the sanitized prover. Importantly, W shares no state with Prove, but it is allowed to toss its own coins.

B.1 Completeness Preservation

The most basic requirement is that the RF should not ruin the protocol's functionality in case both parties are honest. This requirement is captured by the definition below.

Definition 18 (Completeness preservation). *Let $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ be a non-interactive argument system for an NP-relation $\mathcal{R}$. We say that a RF W for Π preserves completeness if the sanitized non-interactive argument system $\widehat{\Pi} = (\mathsf{CRSGen}, \widehat{\mathsf{Prove}}, \mathsf{Verify})$ satisfies completeness.*

A stronger flavor of completeness preservation is to require that sanitized arguments are computationally indistinguishable from honestly produced ones. This property is sometimes known as transparency.

Definition 19 (Transparency). *Let $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ be a non-interactive argument system for an NP-relation $\mathcal{R}$. We say that a RF W for Π is transparent if the following two ensembles are computationally indistinguishable:*

$$\{\pi : \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda); \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\}_{\lambda \in \mathbb{N}, (x,w) \in \mathcal{R}}$$
$$\{\widehat{\pi} : \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda); \widehat{\pi} \leftarrow \mathsf{W}(\mathsf{Prove}(\mathsf{crs}, x, w))\}_{\lambda \in \mathbb{N}, (x,w) \in \mathcal{R}}.$$

B.2 Exfiltration Resistance

Intuitively, a RF for a non-interactive argument system preserves security if a subverted prover's implementation leaks nothing about the witness. We call this property exfiltration resistance. For this notion to be feasible, even assuming RFs, we need to require[20] the subversion to be specious (a.k.a. functionality maintaining).

[20] As shown in [40], an arbitrarily subverted prover could just output garbage making it impossible to have a RF that preserves both completeness and zero knowledge for non-trivial languages.

Definition 20 (Specious subversion of the prover). *Let* $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *be a non-interactive argument system for an NP-relation* $\mathcal{R}$*. We say that* $\widetilde{\mathsf{Prove}}$ *is a specious subversion of the prover if the non-interactive argument system* $\widetilde{\Pi} = (\mathsf{CRSGen}, \widetilde{\mathsf{Prove}}, \mathsf{Verify})$ *satisfies completeness.*

Definition 21 (Exfiltration resistance). *Let* $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ *be a non-interactive argument system for an NP-relation* $\mathcal{R}$*. We say that a RF* W *for* Π *is exfiltration resistant if for all PPT specious subversions* $\widetilde{\mathsf{Prove}}$ *of the prover the following two ensembles are computationally indistinguishable:*

$$\{\widetilde{\pi} : \ \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda); \pi \leftarrow \mathsf{W}(\widetilde{\mathsf{Prove}}(\mathsf{crs}, x, w))\}_{\lambda \in \mathbb{N}, (x,w) \in \mathcal{R}}$$
$$\{\widehat{\pi} : \ \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda); \widehat{\pi} \leftarrow \mathsf{W}(\mathsf{Prove}(\mathsf{crs}, x, w))\}_{\lambda \in \mathbb{N}, (x,w) \in \mathcal{R}}.$$

Discussion. Alternatively, one can define zero knowledge preservation instead of exfiltration resistance where the latter means that the zero-knowledge property still holds for all PPT specious subversions of the prover. Similarly to [40], one can show that the two notions are equivalent (and that both notions become impossible without assuming the subversion to be specious).

Ganesh et al. [40] also consider (specious) subversion of the verifier, but we note that this is not very interesting in the setting of non-interactive argument systems.

B.3 Our Reverse Firewall

In Figure 9 we give a construction of a reverse firewall for the prover of a NIZK argument system Π for any NP-relation $\mathcal{R}$. We require Π to satisfy *strong derivation privacy* for $\mathcal{R}$ and any allowable set of *unary* transformations $\mathcal{T}$ for $\mathcal{R}$. For our construction we instantiate any allowable transformation $T = (T_x, T_w) \in \mathcal{T}$ to be an identity transformation, i.e., $T_x(x) = x$ and $T_w(w) = w$.

Theorem 12. *Let* $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{ZKEval})$ *be a malleable NIZK argument system for any NP relation* $\mathcal{R}$ *with respect to any unary transformation* $\mathcal{T}$ *satisfying completeness and strong derivation privacy (see Definition 9). Then the reverse firewall presented in Figure 9 preserves completeness, is transparent, and is also exfiltration-resistant.*

Proof. We start by showing that the firewall preserves completeness in the above protocol.

Completeness Preservation. It is easy to see that the firewall maintains functionality of the protocol. If the prover honestly follows the protocol, the completeness of Π stipulates that the proof π verifies with respect to the statement x. Completeness preservation then follows from the completeness requirement of the ZKEval algorithm, namely, if $\mathsf{Verify}(\mathsf{crs}, x, \pi) = 1$, then it follows that $\mathsf{Verify}(\mathsf{crs}, T_x(x), \mathsf{ZKEval}(\mathsf{crs}, T, (x, \pi))) = 1$. In our case, $T_x(x) = \widehat{x} = x$.

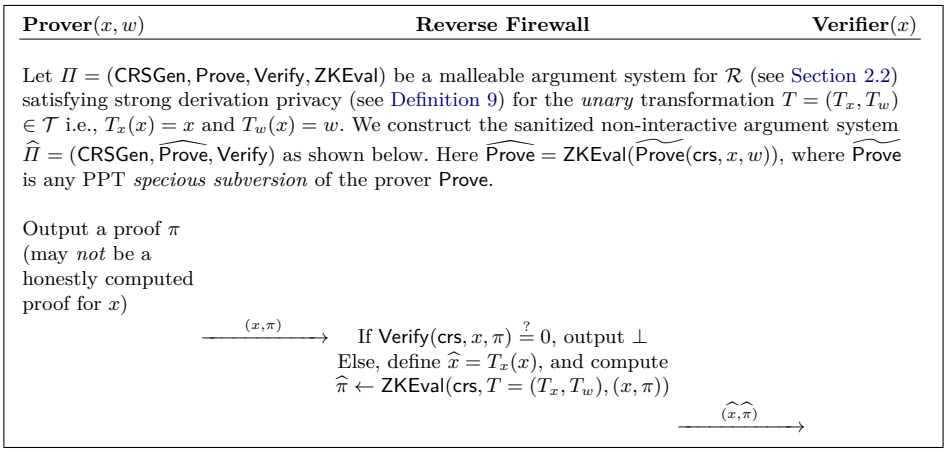

Fig. 9: Reverse Firewall satisfying exfiltration resistance for the prover of a NIZK argument Π.

Exfiltration Resistance. First note that, if $\mathsf{Verify}(\mathsf{crs}, x, \pi) = 0$, the firewall outputs $\perp$ and aborts. This is consistent with the real world behaviour of the protocol. In particular, for any specious subversion of the prover $\widetilde{\mathsf{Prove}}$, the proof π with respect to the statement x should verify with overwhelming probability (since a specious corruption must satisfy completeness). The exfiltration resistance of the firewall then follows from the *strong derivation privacy* and the *zero-knowledge* property of the malleable NIZK argument Π. In particular, the strong derivation privacy (see Definition 9) requires that the proofs generated by ZKEval (for statement $T_x(x) = x$) be indistinguishable from simulated proofs of the same statement. Finally, the zero-knowledge property satisfied by Π stipulates that the simulated proofs (for statement $T_x(x) = x$) are indistinguishable from the real proofs generated by the prover (by knowing the witness $T_w(w) = w$). Since, the proofs generated by the honest prover do not exfiltrate (by definition), the proofs generated by the firewall also do not exfiltrate.

Transparency. By a similar argument as above, it is easy to see that, if Π satisfies strong derivation privacy and the zero-knowledge property, the protocol shown in Figure 9 is transparent. □

C Rerandomizable RCCA-secure PKE

C.1 Preliminaries

Definition 22 (PKE). *A* public key encryption (PKE) *scheme* PKE *for message space* $\mathcal{M}$ *with ciphertext space* $\mathcal{C}$ *consist of the following three probabilistic algorithms:*

$\mathsf{Gen}(1^\lambda)$**:** *Given a unary encoded security parameter* λ *as input, it outputs a public key* pk *and a secret key* sk*.*

$\mathsf{Enc}(\mathsf{pk}, \mathsf{m})$**:** *Given a public key* pk *and a message* $\mathsf{m} \in \mathcal{M}$ *as input, it outputs a ciphertext* $\mathsf{ct} \in \mathcal{C}$*.*

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$**:** *Given a secret key* sk *and a ciphertext* $\mathsf{ct} \in \mathcal{C}$ *as input, it outputs a message* $\mathsf{m} \in \mathcal{M}$ *or* $\perp$ *(indicating a failure).*

We use $\mathfrak{R}$ *to denote the set of all possible random coins for* Enc*.*

We require perfect correctness, that is, for every $\mathsf{m} \in \mathcal{M}$*, all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*, and all* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m})$ *we have* $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \mathsf{m}$*.*

We also require that, given the public key, one can efficiently check membership in $\mathcal{C}$*. Note that* $\mathcal{C}$ *can be larger than the range of the* $\mathsf{Enc}(\mathsf{pk}, \cdot)$*.*

Definition 23 (Rerandomizability). *A* rerandomizable *PKE is a PKE with the following additional probabilistic algorithm:*

$\mathsf{Rerand}(\mathsf{pk}, \mathsf{ct})$**:** *Given a public key* pk *and a ciphertext* $\mathsf{ct} \in \mathcal{C}$ *as input, it outputs a ciphertext* $\mathsf{ct}' \in \mathcal{C}$*.*

For correctness we additionally require that for every $\mathsf{m} \in \mathcal{M}$*, all* $n \in \mathbb{N}$*, all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*, and all* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m})$*, we have* $\mathsf{Dec}(\mathsf{sk}, \mathsf{Rerand}^n(\mathsf{pk}, \mathsf{ct})) = \mathsf{m}$*, where* $\mathsf{Rerand}^n(\mathsf{pk}, \mathsf{ct})$ *stands for applying the* Rerand *algorithm with* pk *n times consecutively on* ct*. If this correctness requirement holds only for* $n \leq u$ *where* $u \in \mathbb{N}$ *is an additional input for the* Gen *algorithm we say that the scheme is* bounded rerandomizable*.*

We use $\mathfrak{R}'$ *to denote the set of random coins for* Rerand*.*

To simplify our construction, we also assume that there exists an efficient and deterministic algorithm $\mathsf{RerandR}$ *that inputs random coins* $r \in \mathfrak{R}$ *and* $r' \in \mathfrak{R}'$ *and satisfies for all* $\mathsf{pk} \in \mathcal{PK}$ *and all* $\mathsf{m} \in \mathcal{M}$

$$\mathsf{Rerand}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{m}; r); r') = \mathsf{Enc}(\mathsf{pk}, \mathsf{m}; \mathsf{RerandR}(r, r')).$$

The above definition of rerandomizability is meaningless without an additional requirement that the output of $\mathsf{Rerand}(\mathsf{pk}, \mathsf{ct})$ "looks independent" of ct. Two different versions of such a requirement are defined next.

Definition 24. *We say that a rerandomizable PKE is* RAND-CPA*-secure or* RAND-CCA*-secure respectively, if every PPT adversary* $\mathcal{A}$ *has advantage*

$$\mathsf{Adv}^{\mathsf{rand\text{-}cpa}\ \textit{resp.}\ \text{-}\mathsf{cca}}_{\mathcal{A},\mathsf{PKE}}(\lambda) := 2\left|\Pr[\mathsf{Exp}^{(\mathsf{lvl})\mathsf{rand\text{-}cca}}_{\mathcal{A},\mathsf{PKE}}(\lambda) = 1] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

for a negligible function negl*. The security games are defined in Figure 10.*

We next recall the definition of replay-CCA (RCCA) security. This security notion has been introduced by Canetti et al. [16] as a relaxation of CCA security that can be achieved by rerandomizable encryption schemes.

Definition 25. *We say that a PKE* PKE *is* IND-RCCA*-secure if every PPT adversary* $\mathcal{A}$ *has advantage*

$$\mathsf{Adv}^{\mathsf{ind\text{-}rcca}}_{\mathcal{A},\mathsf{PKE}}(\lambda) := 2\left|\Pr[\mathsf{Exp}^{\mathsf{ind\text{-}rcca}}_{\mathcal{A},\mathsf{PKE}}(\lambda) = 1] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

for a negligible function negl*. The* IND-RCCA *game is defined in Figure 11.*

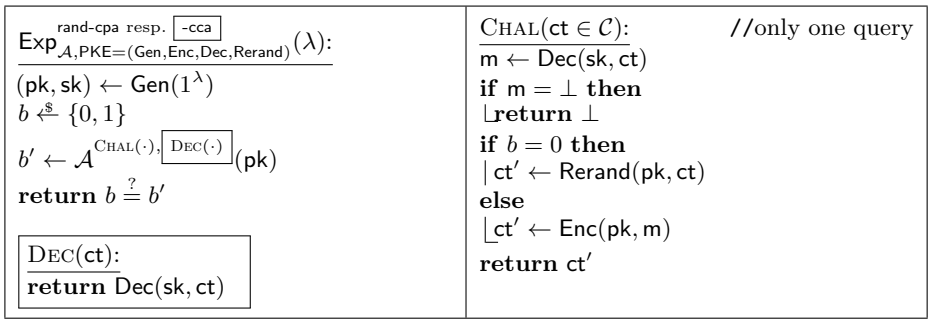

Fig. 10. The games for the rerandomizablility notions RAND-CPA and RAND-CCA. The boxed code segments are only executed in the RAND-CCA game.

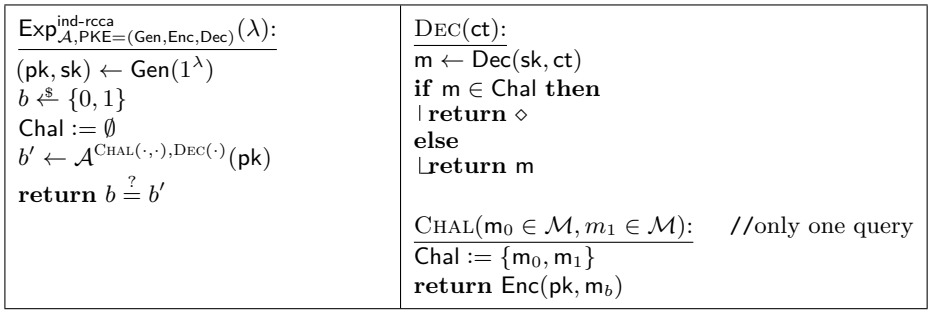

Fig. 11. The game for IND-RCCA security.

The standard security notion IND-CPA for PKE is defined like IND-RCCA-security, except that the adversary does not get access to the decryption oracle.

C.2 Naor–Yung transformation for rerandomizable PKE

We present a generic transformation – reminiscent to the Naor–Yung transformation [59, 32] – from IND-CPA-secure rerandomizable encryption schemes to IND-RCCA secure rerandomizable encryption schemes using a malleable NIZK for the NP relation

$$\mathcal{R}_{\mathsf{ny}} := \{((\mathsf{pk}_1,\mathsf{ct}_1,\mathsf{pk}_2,\mathsf{ct}_2),(\mathsf{m},r_1,r_2)) \mid \mathsf{ct}_1 = \mathsf{Enc}(\mathsf{pk}_1,\mathsf{m};r_1) \wedge \mathsf{ct}_2 = \mathsf{Enc}(\mathsf{pk}_2,\mathsf{m};r_2)\}$$

and the set of transformations

$$\mathcal{T}_{\mathsf{ny}} := \{T_{r_1',r_2'} = (T_{x,r_1',r_2'}, T_{w,r_1',r_2'}) \mid r_1', r_2' \in \mathfrak{R}'\} \quad \text{where}$$

$\underline{\mathsf{Gen}'(1^\lambda,):}$
$(\mathsf{pk}_1,\mathsf{sk}_1) \leftarrow \mathsf{Gen}(1^\lambda)$
$(\mathsf{pk}_2,\mathsf{sk}_2) \leftarrow \mathsf{Gen}(1^\lambda)$
$\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$
return $(\mathsf{pk}' := (\mathsf{pk}_1,\mathsf{pk}_2,\mathsf{crs}), \mathsf{sk}' := (\mathsf{pk}_1,\mathsf{pk}_2,\mathsf{sk}_1,\mathsf{crs}))$

$\underline{\mathsf{Enc}'(\mathsf{pk}' = (\mathsf{pk}_1,\mathsf{pk}_2,\mathsf{crs}),\mathsf{m}):}$
$\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{pk}_1,\mathsf{m};r_1)$
$\mathsf{ct}_2 \leftarrow \mathsf{Enc}(\mathsf{pk}_2,\mathsf{m};r_2)$
$\pi \leftarrow \mathsf{Prove}(\mathsf{crs},(\mathsf{pk}_1,\mathsf{ct}_1,\mathsf{pk}_2,\mathsf{ct}_2),(\mathsf{m},r_1,r_2))$
return $\mathsf{ct}' := (\mathsf{ct}_1,\mathsf{ct}_2,\pi)$

$\underline{\mathsf{Dec}'(\mathsf{sk}',\mathsf{ct}' = (\mathsf{ct}_1,\mathsf{ct}_2,\pi)):}$
parse $(\mathsf{pk}_1,\mathsf{pk}_2,\mathsf{sk}_1,\mathsf{crs}) := \mathsf{sk}'$
if $\mathsf{Verify}(\mathsf{crs},(\mathsf{pk}_1,\mathsf{ct}_1,\mathsf{pk}_2,\mathsf{ct}_2,\pi) \stackrel{?}{=} 1$
then
| **return** $\mathsf{Dec}(\mathsf{sk}_1,\mathsf{ct}_1)$
else
└**return** $\perp$

$\underline{\mathsf{Rerand}(\mathsf{pk}',\mathsf{ct}' := (\mathsf{ct}_1,\mathsf{ct}_2,\pi)):}$
parse $(\mathsf{pk}_1,\mathsf{pk}_2,\mathsf{sk}_1,\mathsf{crs}) := \mathsf{sk}'$
$\widehat{\mathsf{ct}_1} \leftarrow \mathsf{Rerand}(\mathsf{pk}_1,\mathsf{ct}_1;r_1')$
$\widehat{\mathsf{ct}_2} \leftarrow \mathsf{Rerand}(\mathsf{pk}_2,\mathsf{ct}_2;r_2')$
$\widehat{\pi} \leftarrow \mathsf{ZKEval}(\mathsf{crs},T_{r_1',r_2'},((\mathsf{pk}_1,\mathsf{ct}_1,\mathsf{pk}_2,\mathsf{ct}_2),\pi))$
return $(\widehat{\mathsf{ct}_1},\widehat{\mathsf{ct}_2},\widehat{\pi})$

Fig. 12. The Naor–Yung transformation that turns a rerandomizable RAND-CPA and IND-CPA-secure PKE $\mathsf{PKE} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec},\mathsf{Rerand})$ and a malleable NIZK $\Pi = (\mathsf{CRSGen},\mathsf{Prove},\mathsf{Verify})$ into a rerandomizable RAND-CCA and IND-CCA-secure PKE $\mathsf{PKE}'[\mathsf{PKE},\Pi] = (\mathsf{Gen}',\mathsf{Enc}',\mathsf{Dec}',\mathsf{Rerand}')$.

$$
\begin{aligned}
T_{x,r_1',r_2'} : (\mathcal{PK}\times\mathcal{C})^2 &\to (\mathcal{PK}\times\mathcal{C})^2\\
(\mathsf{pk}_1,\mathsf{ct}_1,\mathsf{pk}_2,\mathsf{ct}_2) &\mapsto (\mathsf{pk}_1,\mathsf{Rerand}(\mathsf{pk}_1,\mathsf{ct}_1;r_1'),\mathsf{pk}_2,\mathsf{Rerand}(\mathsf{pk}_2,\mathsf{ct}_2;r_2'))\\
T_{w,r_1',r_2'} : (\mathcal{M}\times\mathfrak{R}\times\mathfrak{R}) &\to (\mathcal{M}\times\mathfrak{R}\times\mathfrak{R})\\
(\mathsf{m},r_1,r_2) &\mapsto (\mathsf{m},\mathsf{RerandR}(r_1,r_1'),\mathsf{RerandR}(r_2,r_2')).
\end{aligned}
$$

The definition of $\mathsf{RerandR}$ implies that $\mathcal{T}_{\mathsf{ny}}$ is an admissible set of transformations.

A similar result is shown in [23]. However, they use a malleable NIZKPoK where an explanation of a statement can be extracted using a trapdoor for the CRS. We do not know how to achieve such an extraction property for our SNARK-based construction. The generic transformation of a NIZK to a NIZKPoK by encrypting the witness under a public key stored in the CRS [64] does not generalize to malleable NIZKs (unless one assumes that the encryption scheme is homomorphic for the transformations applied to the witness). Our malleable NIZK constructions only achieves a different variant of controlled malleable simulation extractability where the extractor can depend on the adversary, but does not get a trapdoor. Moreover, the extractor is not a fast extractor, even if the underlying SNARK had a fast extractor. To answer many decryption queries, we would have to nest these extractors which would lead to an exponential run-time. Therefore, we prove our results here using the classical variant of Naor–Yung with two ciphertexts and a simulation sound NIZK that does not need any extractability. The transformation is given in Figure 12.

Clearly, the resulting $\mathsf{PKE}'[\mathsf{PKE},\Pi]$ is correct, if PKE is correct and Π is complete.

Theorem 13 (RCCA-security). *If* PKE *is perfectly*[21] *correct and* $\mathsf{IND\text{-}CPA}$*-secure, and* Π *is zero-knowledge and controlled-malleable simulation sound, then* $\mathsf{PKE}'[\mathsf{PKE}, \Pi]$ *is* $\mathsf{IND\text{-}RCCA}$*-secure. More precisely, for every PPT adversary* $\mathcal{A}$ *against the* $\mathsf{IND\text{-}RCCA}$ *security of* $\mathsf{PKE}'[\mathsf{PKE}, \Pi]$*, there exist PPT adversaries* $\mathcal{B}$*,* $\mathcal{C}$*, and* $\mathcal{D}$

$$\mathsf{Adv}^{\mathsf{ind\text{-}rcca}}_{\mathcal{A},\mathsf{PKE}'[\mathsf{PKE},\Pi]}(\lambda) \leq 2\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{B},\mathsf{PKE}}(\lambda) + \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{C},\Pi}(\lambda) + \mathsf{Adv}^{\mathsf{cm\text{-}ss}}_{\mathcal{D},\Pi}(\lambda).$$

Proof. The proof proceeds via a hybrid argument. The game G_0 is the real $\mathsf{IND\text{-}RCCA}$ game for $\mathsf{PKE}'[\mathsf{PKE}, \Pi]$. Let (S_1, S_2) be the zero-knowledge simulator for Π. The game G_1 is identical to G_0, except that crs is generated via $(\mathsf{crs}, \mathsf{td}) \leftarrow S_1(1^\lambda)$ (instead of $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$). Furthermore, the proof returned by the CHAL oracle is computed as $S_2(\mathsf{crs}, \mathsf{td}, (\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{pk}_2, \mathsf{ct}_2))$ (instead of $\mathsf{Prove}(\mathsf{crs}, (\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{pk}_2, \mathsf{ct}_2), (\mathsf{m}_b, r_1, r_2))$).

Lemma 12 ($\mathsf{G}_0 \rightsquigarrow \mathsf{G}_1$). *For every PPT adversary* $\mathcal{A}$ *there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{B},\Pi}(\lambda).$$

Proof. The reduction is straightforward: It obtains crs from the zero-knowledge challenger and computes all components belonging to PKE itself. To compute the proof in $\mathcal{A}$'s CHAL query, the reduction uses the Prove oracle from the zero-knowledge challenger which will either return an honest proof or a simulated proof. □

In the next hybrid, G_2, the challenge ciphertext is changed as follows: The component ct_2 is always an encryption of m_0 (instead of m_b).

Lemma 13 ($\mathsf{G}_1 \rightsquigarrow \mathsf{G}_2$). *For every PPT adversary* $\mathcal{A}$ *there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\left|\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{B},\mathsf{PKE}}.$$

Proof. Note that in G_1 and G_2, we never use sk_2. The reduction is thus straightforward: It uses the public key it gets from the $\mathsf{IND\text{-}CPA}$-security game for PKE as pk_2. In the challenge query, the reduction submits the challenge messages $m'_0 = m_b$ and $m'_1 = m_0$ to its own challenge oracle and uses the resulting ciphertext as ct_2. An adversary that wins with different probability in G_1 than in G_2 can thus be turned into an adversary against $\mathsf{IND\text{-}CPA}$-security game for PKE. □

The next hybrid G_3 answers all DEC queries by decrypting with sk_2 (instead of sk_1).

Lemma 14 ($\mathsf{G}_2 \rightsquigarrow \mathsf{G}_3$). *For every PPT adversary* $\mathcal{A}$ *there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\left|\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{cm\text{-}ss}}_{\mathcal{D},\Pi}(\lambda).$$

[21] For non-perfect correctness, we can use the derandomziation technique of [34] to get a similar result.

Proof. The reduction proceeds as follows: It receives crs from the controlled-malleable simulation soundness game and computes the simulated proof for the challenge query using the PROVE oracle. If the adversary makes a DEC query on $(\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ where $\mathsf{Verify}(\mathsf{crs}, (\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{pk}_2, \mathsf{ct}_2), \pi) = 1$, it computes $\mathsf{m}_1 \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_1)$ and $\mathsf{m}_2 \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_2)$. By perfect correctness, the messages m_1 and m_2 are the same messages used during encryption for ct_1 and ct_2. If $\mathsf{m}_1 \neq \mathsf{m}_2$ and $\{\mathsf{m}_1, \mathsf{m}_2\} \not\subseteq \mathsf{Chal}$, the reduction outputs $((\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{pk}_2, \mathsf{ct}_2), \pi)$ and wins the controlled-malleable simulation soundness experiment. To see this, note that all allowed transformations keep the messages inside ct_1 and ct_2 intact. The relation only allows ciphertexts which encrypt the same message and the reduction asked for a simulated proof where ct_1 encrypts m_b and ct_2 encrypts m_0. Thus, a valid proof for a ciphertext pair that encrypts two different messages and at least one of them is not m_0 or m_1 will win the controlled-malleable simulation soundness game.

If the adversary does not make such a DEC query that makes the reduction win, the games G_2 and G_3 are identical in its view, because

- DEC queries where the proof does not verify are answered with $\perp$ in both games,
- DEC queries where ct_1 and ct_2 encrypt the same message are answered identical, and
- DEC queries where ct_1 and ct_2 both encrypt a message from Chal are answered with $\diamond$ and thus identical.

□

In the final game G_4 in the challenge query ct_1 is always an encryption of m_0 (instead of m_b).

Lemma 15 ($\mathsf{G}_3 \rightsquigarrow \mathsf{G}_4$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{B},\mathsf{PKE}}.$$

Since in $\mathsf{G}_3/\mathsf{G}_4$ sk_1 is never used, the reduction is analogous to Lemma 13.

Lemma 16 (G_4)**.**

$$\Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

Proof. In G_4, the adversaries view is statistically independent of the challenge bit b. □

Combining Lemmata 12–16 yields Theorem 13. □

Theorem 14 (Rand-CCA-security). *If* PKE *is perfectly correct and* RAND-CPA*-secure, and Π is derivation private, zero-knowledge and controlled-malleable simulation sound, then* $\mathsf{PKE}'[\mathsf{PKE}, \Pi]$ *is* RAND-CCA*-secure. More precisely, for every PPT adversary $\mathcal{A}$ against the* RAND-CCA *security of* $\mathsf{PKE}'[\mathsf{PKE}, \Pi]$*, there exist PPT adversaries $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{F}$ such that*

$$\mathsf{Adv}^{\mathsf{rand\text{-}cca}}_{\mathcal{A},\mathsf{PKE}'[\mathsf{PKE},\Pi]}(\lambda) \leq 2\mathsf{Adv}^{\mathsf{rand\text{-}cpa}}_{\mathcal{B},\mathsf{PKE}}(\lambda) + \frac{1}{2}\mathsf{Adv}^{\mathsf{zk}}_{\mathcal{C},\Pi}(\lambda) + \frac{1}{2}\mathsf{Adv}^{\mathsf{rdp}}_{\mathcal{D},\Pi}(\lambda) + \mathsf{Adv}^{\mathsf{cm\text{-}ss}}_{\mathcal{F},\Pi}(\lambda).$$

Proof. The proof proceeds via a hybrid argument. The game G_0 is the real RAND-CCA game for $\mathsf{PKE}'[\mathsf{PKE}, \Pi]$. Let (S_1, S_2) be the zero-knowledge simulator for Π. The game G_1 is identical to G_0, except that crs is generated via $(\mathsf{crs}, \mathsf{td}) \leftarrow S_1(1^\lambda)$ (instead of $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$). Furthermore, the proof returned by the CHAL in the $b = 1$ case is computed as $S_2(\mathsf{crs}, \mathsf{td}, (\mathsf{pk}_1, \mathsf{ct}'_1, \mathsf{pk}_2, \mathsf{ct}'_2))$ (instead of $\mathsf{Prove}(\mathsf{crs}, (\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{pk}_2, \mathsf{ct}_2), (\mathsf{m}, r_1, r_2))$).

Lemma 17 ($\mathsf{G}_0 \rightsquigarrow \mathsf{G}_1$). *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]\right| \leq \frac{1}{2}\mathsf{Adv}^{\mathsf{zk}}_{\mathcal{B},\Pi}(\lambda).$$

Proof. The reduction is straightforward: It obtains crs from the zero-knowledge challenger and computes all components belonging to PKE itself. To compute the proof in $\mathcal{A}$'s CHAL query if $b = 1$, the reduction uses the Prove oracle from the zero-knowledge challenger which will either return an honest proof or a simulated proof. If $b = 0$, which happens with probability $\frac{1}{2}$, the games are identical. □

The next hybrid G_2 is identical to G_1, except that in the CHAL query in the $b = 0$ case the proof is computed as $S_2(\mathsf{crs}, \mathsf{td}, (\mathsf{pk}_1, \mathsf{ct}'_1, \mathsf{pk}_2, \mathsf{ct}'_2))$ (instead of $\mathsf{ZKEval}(\mathsf{crs}, T_{r'_1,r'_2}, ((\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{pk}_2, \mathsf{ct}_2), \pi))$).

Lemma 18 ($\mathsf{G}_1 \rightsquigarrow \mathsf{G}_2$). *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]\right| \leq \frac{1}{2}\mathsf{Adv}^{\mathsf{rdp}}_{\mathcal{B},\Pi}(\lambda).$$

Proof. The reduction is straightforward: It obtains crs from the derivation-privacy challenger and computes all components belonging to PKE itself. To compute the proof in $\mathcal{A}$'s CHAL query if $b = 0$, the reduction uses the Prove oracle from the derivation privacy challenger which will either return a proof obtained via ZKEval or a simulated proof. If $b = 1$, which happens with probability $\frac{1}{2}$, the games are identical. □

In the next hybrid, G_3, the challenge ciphertext is changed as follows: The component ct'_2 is always a fresh encryption of m (instead of being a fresh encryption of m or a rerandomization of ct_2, depending on the challenge bit).

Lemma 19 ($\mathsf{G}_2 \rightsquigarrow \mathsf{G}_3$). *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{rand\text{-}cpa}}_{\mathcal{B},\mathsf{PKE}}.$$

Proof. Note that in G_2 and G_3, we never use sk_2. The reduction is thus straightforward: It uses the public key it gets from the RAND-CPA-security game for PKE as pk_2. When the adversary makes its challenge query on (ct_1, ct_2, π) and this ciphertexts decrypts to $\mathsf{m} \neq \perp$, the reduction answers if $b = 1$ with a ciphertext where ct'_2 is a fresh encryption of m. If $b = 0$, it submits ct_2 to its own challenge oracle and uses the resulting ciphertext as ct'_2. An adversary that wins with

different probability in G_2 than in G_3 can thus be turned into an adversary against RAND-CPA-security game for PKE. □

The next hybrid G_4 answers all DEC queries by decrypting with sk_2 (instead of sk_1).

Lemma 20 ($\mathsf{G}_3 \rightsquigarrow \mathsf{G}_4$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{cm\text{-}ss}}_{\mathcal{D},\Pi}(\lambda).$$

Proof. The reduction proceeds as follows: It receives crs from the controlled-malleable simulation soundness game and computes the simulated proof for the challenge query using the PROVE oracle. If the adversary makes a DEC query on $(\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ where $\mathsf{Verify}(\mathsf{crs}, (\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{pk}_2, \mathsf{ct}_2), \pi) = 1$, it computes $\mathsf{m}_1 \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_1)$ and $\mathsf{m}_2 \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_2)$. By perfect correctness, the messages m_1 and m_2 are the same messages used during encryption for ct_1 and ct_2. If $\mathsf{m}_1 \neq \mathsf{m}_2$, the reduction outputs $((\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{pk}_2, \mathsf{ct}_2), \pi)$ and wins the controlled-malleable simulation soundness experiment. To see this, note that all allowed transformations keep the messages inside ct_1 and ct_2 intact. The relation only allows ciphertexts which encrypt the same message and the reduction also asks only for a simulated proof where ct_1 and ct_2 encrypt the same message. Thus, a valid proof for a ciphertext pair that encrypts two different messages will win the controlled-malleable simulation soundness game.

If the adversary does not make such a DEC query that makes the reduction win, the games G_3 and G_4 are identical in its view, because DEC queries where the proof does not verify are answered with $\perp$ in both games and DEC queries where ct_1 and ct_2 encrypt the same message are answered identical. □

In the final game G_5 in the challenge query ct_1 is always an encryption of m_0 (instead of m_b).

Lemma 21 ($\mathsf{G}_4 \rightsquigarrow \mathsf{G}_5$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{B},\mathsf{PKE}}.$$

Since in $\mathsf{G}_4/\mathsf{G}_5$ sk_1 is never used, the reduction is analogous to Lemma 19.

Lemma 22 (G_5)**.**

$$\Pr[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

Proof. In G_5, the adversaries view is statistically independent of the challenge bit b. □

Combining Lemmata 17–22 yields Theorem 14. □

On lattice-based instantiations. In lattice-based encryptions schemes like Regev [65] and dual Regev [41], the size of the noise increases with each rerandomization. Therefore, these schemes do not fit directly in the malleable NIZK framework where applying a transformation (rerandomizing a ciphertext) has to result in

a statement in the initial relation (a ciphertext with the initial noise level). Nevertheless, we can relax the notion for admissible transformations to allow transformations as follows: A directly generate proof shows that a statement is in a relation $\mathcal{R}_0$ and each transformation guarantees that when the transformation is applied to a statement in $\mathcal{R}_i$, the resulting statement is in $\mathcal{R}_{i+1}$. For lattice-based RCCA-secure Rerandomizable PKE, the relations $\mathcal{R}_0, \mathcal{R}_1, \ldots$ would allow increasingly larger noise levels.

We also need to ensure that the noise level does not become too large, otherwise the ciphertexts would no longer decrypt correctly. Lattice-based encryption schemes with polynomial modulus-to-noise ratio can support only a fixed polynomial number of rerandomizations. When combined with a bounded malleable NIZK like our counter-based construction in Appendix A, this is ensured. Since this malleable NIZK is only leveled derivation private, it also achieves only a leveled variant of RAND-CCA security where the number of rerandomizations is leaked. When a super-polynomial modulus-to-noise ratio is used, we can get schemes that can (hypothetically) tolerate a super-polynomial number of reradomizations and therefore in particular any polynomial number of rerandomizations (without an *a priori* fixed bound). In this setting, we can use any simulation extractable, malleable NIZK (like our AOWF-based construction in Section 3). The simulation extractability game guarantees that any statement with proof the adversary outputs is obtained by applying only polynomially many transformations to an initial (or simulated) statement, since the extractor in this game has to output an explanation (that grows with every applied transformation) for every statement with proof *in polynomial time*.

D RCCA-Secure Updatable Encryption

We define randomized RCCA-secure updatable encryption (UE) by adapting the recent IND-CPA UE model from [68] which itself implies the randomized IND-CPA model used in [60, 39, 56]. Currently, randomized RCCA-secure UE schemes are absent in such a strong model.

UE schemes consists of the commonly known algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ for key generation, encryption and decryption, but adapted to discrete epochs $1, 2, 3, \ldots$ where Gen outputs an initial secret key K_1 (for epoch 1). Additionally, the UE scheme has a PPT algorithm RotKey which takes a key K_e and outputs a next-epoch key K_{e+1} along with a update token Δ_{e+1}. This update token can be used by a third party to update ciphertexts under key K_e for epoch e to ciphertexts for epoch $e+1$ under key K_{e+1} via an algorithm Update. We recap from [68]:

Definition 26. *A UE scheme* UE *with message space* $\mathcal{M}$ *consist of the PPT algorithms* $(\mathsf{Gen}, \mathsf{RotKey}, \mathsf{Enc}, \mathsf{Update}, \mathsf{Dec})$*:*

$\mathsf{Gen}(1^\lambda)$*: on input security parameter* λ*, key generation outputs an initial (symmetric) key* K_1*.*

$\mathsf{RotKey}(\mathsf{K}_e)$*: on input key* K_e*, key rotation outputs an updated key* K_{e+1} *for the next epoch together with an update token* Δ_{e+1}*.*

$\mathsf{Enc}(\mathsf{K}_e, \mathsf{m}, e_{\mathsf{exp}})$*: on input key* K_e*, a message* $\mathsf{m} \in \mathcal{M}$*, and expiry epoch* e_{exp}*, encryption outputs a ciphertext* $\mathsf{ct}_{e,e_{\mathsf{exp}}}$ *or* $\perp$*.*

$\mathsf{Update}(\Delta_{e+1}, \mathsf{ct}_{e,e_{\mathsf{exp}}})$*: on input an update token* Δ_{e+1} *and a ciphertext* $\mathsf{ct}_{e,e_{\mathsf{exp}}}$*, update outputs an updated ciphertext* $\mathsf{ct}_{e+1,e_{\mathsf{exp}}}$ *or* $\perp$*.*

$\mathsf{Dec}(\mathsf{K}_e, \mathsf{ct}_{e,e_{\mathsf{exp}}})$*: on input key* K_e *and a ciphertext* $\mathsf{ct}_{e,e_{\mathsf{exp}}}$*, decryption outputs* $\mathsf{m} \in \mathcal{M} \cup \{\perp\}$*.*

Correctness essentially guarantees that the decryption of updated ciphertexts with a rotated key yield the desired encrypted message. Concretely:

Correctness. For all $\lambda \in \mathbb{N}$, for all $e \in [\lfloor \mathsf{poly}(\lambda) \rfloor]$, for $\mathsf{K}_1 \leftarrow \mathsf{Gen}(1^\lambda)$, for all $i \in \{1, \ldots, e\}$, for all $(\mathsf{K}_{i+1}, \Delta_{i+1}) \leftarrow \mathsf{RotKey}(\mathsf{K}_i)$, for all $\mathsf{m} \in \mathcal{M}$, for all $e_{\mathsf{exp}} \in \mathbb{N}$, for all $j \in \{1, \ldots, e+1\}$, for all $\mathsf{ct}_{j,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(\mathsf{K}_j, \mathsf{m}, e_{\mathsf{exp}})$, we require that $\mathsf{m} = \mathsf{Dec}(\mathsf{K}_j, \mathsf{ct}_{j,e_{\mathsf{exp}}})$ holds if $e_{\mathsf{exp}} \geq j$. Moreover, for all $j \in \{1, \ldots, e\}$, for all $\mathsf{ct}_{j,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(\mathsf{K}_j, \mathsf{m}, e_{\mathsf{exp}})$, for all $i \in \{j, \ldots, e\}$, for $\mathsf{ct}'_{j,e_{\mathsf{exp}}} := \mathsf{ct}_{j,e_{\mathsf{exp}}}$, for all $\mathsf{ct}'_{i+1,e_{\mathsf{exp}}} \leftarrow \mathsf{Update}(\Delta_{i+1}, \mathsf{ct}'_{i,e_{\mathsf{exp}}})$, we require that $\mathsf{m} = \mathsf{Dec}(\mathsf{K}_{e+1}, \mathsf{ct}'_{e+1,e_{\mathsf{exp}}})$ holds if $e_{\mathsf{exp}} \geq e+1$.

The following security definition is taken from [68] and adapted to yield chosen ciphertext security next.

Definition 27. *A UE scheme* UE *is* ee-IND-UE-CPA*-secure iff for any PPT adversary* $\mathcal{A}$*, the advantage function*

$$\mathsf{Adv}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathsf{Exp}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},\mathcal{A}}(\lambda) = 1\right] - 1/2 \right|$$

is negligible in λ*, where* $\mathsf{Exp}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}cpa}}_{\mathsf{UE},\mathcal{A}}$ *is defined as in Figure 13.*

To achieve chosen ciphertext security, we adapt the security model from [68] and add two additional oracles, namely, one for decryption and one for arbitrary ciphertext updates. As an intuition, the adversary is now capable of querying decryption of chosen ciphertexts as well as updates of chosen ciphertexts. Since we deal with randomized updates, we only allow decryptions and updates of ciphertexts which do not correspond to the challenge messages (i.e., randomized chosen-ciphertext security or RCCA). We define:

Definition 28. *A UE scheme* UE *is* ee-IND-UE-RCCA*-secure iff for any PPT adversary* $\mathcal{A}$*, the advantage function*

$$\mathsf{Adv}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}rcca}}_{\mathsf{UE},\mathcal{A}}(\lambda) := \left| \Pr\left[\mathsf{Exp}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}rcca}}_{\mathsf{UE},\mathcal{A}}(\lambda) = 1\right] - 1/2 \right|$$

is negligible in λ*, where* $\mathsf{Exp}^{\mathsf{ee\text{-}ind\text{-}ue\text{-}rcca}}_{\mathsf{UE},\mathcal{A}}$ *is defined as in Figure 13.*

Rerandomization properties. Similarly to the randomized PKE construction above, we require a deterministic PPT algorithm $\mathsf{RerandR}'$ in the UE setting that on input random coins $r \in \mathfrak{R}$ and $r' \in \mathfrak{R}'$, for all $\mathsf{K}_1 \leftarrow \mathsf{Gen}(1^\lambda)$, for all $e, e_{\mathsf{exp}} \in \mathbb{N}$, for all $(\mathsf{K}_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{RotKey}(\mathsf{K}_e)$, and for all $\mathsf{m} \in \mathcal{M}$, the algorithm satisfies

$$\mathsf{Update}(\Delta_{e+1}, \mathsf{Enc}(\mathsf{K}_e, \mathsf{m}, e_{\mathsf{exp}}; r); r') = \mathsf{Enc}(\mathsf{K}_{e+1}, \mathsf{m}, e_{\mathsf{exp}}; \mathsf{RerandR}'(r, r')).$$

Experiments $\mathsf{Exp}^{\text{ee-ind-ue-cpa resp. } \boxed{\text{rcca}}}_{\mathsf{UE},\mathcal{A}}(\lambda)$
$\mathsf{K}_1 \leftarrow \mathsf{Gen}(\lambda), \mathsf{phase} = 0, e = 1, c = 0, \Delta_1 = \bot$
$\mathcal{M}^* := \emptyset, \mathcal{L}^* := \emptyset, \mathcal{C}^* := \emptyset, \mathcal{K}^* := \emptyset, \mathcal{D}^* := \emptyset, b \leftarrow \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc}',\mathsf{RotKey}',\mathsf{Update}',\mathsf{Corrupt},\mathsf{Chall},\mathsf{GetUpdct}^*,\boxed{\mathsf{ArbUpdate},\mathsf{Dec}'}}(1^\lambda)$
if $\mathcal{A}$ is not valid, then return $b'' \leftarrow \{0,1\}$
if $b = b'$, then return 1 else return 0

Oracles
$\mathsf{Enc}'(\mathsf{m}, e_{\mathsf{exp}})$: run $\mathsf{ct}_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{Enc}(\mathsf{K}_e, \mathsf{m}, e_{\mathsf{exp}})$ and set $\mathcal{L}^* := \mathcal{L}^* \cup \{(c, e, \mathsf{m}, \mathsf{ct}_{e,e_{\mathsf{exp}}})\}$, $c = c + 1$. Return $\mathsf{ct}_{e,e_{\mathsf{exp}}}$.
RotKey' : run $(\mathsf{K}_{e+1}, \Delta_{e+1}) \leftarrow \mathsf{RotKey}(\mathsf{K}_e)$. If $\mathsf{phase} = 1$, run $\mathsf{ct}^*_{e+1,b} \leftarrow \mathsf{Update}(\Delta_{e+1}, \mathsf{ct}^*_{e,b})$. Set $e = e + 1$.
$\mathsf{Update}'(\mathsf{ct}_{e-1,e_{\mathsf{exp}}})$: if $(\cdot, e-1, \mathsf{m}, \mathsf{ct}_{e-1,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, for some $\mathsf{m} \in \mathcal{M}$, return $\bot$. Run $\mathsf{ct}_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{Update}(\Delta_e, \mathsf{ct}_{e-1,e_{\mathsf{exp}}})$ and set $\mathcal{L}^* := \mathcal{L}^* \cup (c, e, \mathsf{m}, \mathsf{ct}_{e,e_{\mathsf{exp}}}), c = c + 1$. Return $\mathsf{ct}_{e,e_{\mathsf{exp}}}$.
$\boxed{\mathsf{ArbUpdate}(\mathsf{ct}_{e-1,e_{\mathsf{exp}}})}$: if $\mathsf{phase} = 0$ and $(\cdot, e-1, \cdot, \mathsf{ct}_{e-1,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, return $\bot$. Run $\mathsf{m} := \mathsf{Dec}(\mathsf{K}_{e-1}, \mathsf{ct}_{e-1,e_{\mathsf{exp}}})$ and $\mathsf{ct}_{e,e_{\mathsf{exp}}} \leftarrow \mathsf{Update}(\Delta_e, \mathsf{ct}_{e-1,e_{\mathsf{exp}}})$. If $\mathsf{phase} = 1$ and $\mathsf{m} \in \mathcal{M}^*$, set $\mathcal{C}^* := \mathcal{C}^* \cup \{e\}$, else set $\mathcal{L}^* := \mathcal{L}^* \cup \{(c, e, \mathsf{m}, \mathsf{ct}_{e,e_{\mathsf{exp}}})\}$, $c = c + 1$. Return $\mathsf{ct}_{e,e_{\mathsf{exp}}}$.
$\mathsf{Corrupt}(\mathsf{inp}, e')$: if $e' > e$, return $\bot$. If $\mathsf{inp} = \mathsf{key}$, set $\mathcal{K}^* = \mathcal{K}^* \cup \{e'\}$ and return $\mathsf{K}_{e'}$. If $\mathsf{inp} = \mathsf{token}$, set $\mathcal{D}^* = \mathcal{D}^* \cup \{e'\}$ and return $\Delta_{e'}$.
$\mathsf{Chall}(\mathsf{m}_0, \mathsf{ct}_{e-1,e_{\mathsf{exp}}})$: if $\mathsf{phase} = 1$, return $\bot$. Set $\mathsf{phase} = 1$. If $(\cdot, e-1, \mathsf{m}_1, \mathsf{ct}_{e-1,e_{\mathsf{exp}}}) \notin \mathcal{L}^*$, for some $\mathsf{m}_1 \in \mathcal{M}$, return $\bot$. If $b = 0$, set $\mathsf{ct}^*_{e,0} \leftarrow \mathsf{Enc}(\mathsf{K}_e, \mathsf{m}_0, e_{\mathsf{exp}})$, else $\mathsf{ct}^*_{e,1} \leftarrow \mathsf{Update}(\Delta_e, \mathsf{ct}_{e-1,e_{\mathsf{exp}}})$. Set $\mathcal{M}^* := \{\mathsf{m}_0, \mathsf{m}_1\}$, $\mathcal{C}^* = \mathcal{C}^* \cup \{e\}$, $e^* = e$, $e^*_{\mathsf{exp}} = e_{\mathsf{exp}}$, and return $\mathsf{ct}^*_{e,b}$.
$\mathsf{GetUpdct}^*$: If $\mathsf{phase} = 0$, return $\bot$. Set $\mathcal{C}^* := \mathcal{C}^* \cup \{e\}$ and return $\mathsf{ct}^*_{e,b}$.
$\boxed{\mathsf{Dec}'(\mathsf{ct})}$: Run $\mathsf{m} = \mathsf{Dec}(\mathsf{K}_e, \mathsf{ct})$. If $\mathsf{phase} = 1$ and $\mathsf{m} \in \mathcal{M}^*$, return $\bot$. Return m.

$\mathcal{A}$ **is valid iff**:
1) For all $e' \in \mathcal{K}^*$, $e' \notin \mathcal{C}^*$ holds. (No trivial win via retrieved keys.)
2) For all $e' \in \mathcal{K}^*$ with $e^* < e' \leq e^*_{\mathsf{exp}}$ and $e' - 1 \in \mathcal{C}^*$, $e' - 1 \notin \mathcal{D}^*$ holds. (No trivial win via retrieved update token.)

Fig. 13. The EE-IND-UE-CPA resp. RCCA security notions for UE schemes with expiry epochs.

Naor–Yung Transformation for UE. We show how to apply the Naor–Yung paradigm [59, 32] for UE adapting the above IND-CPA-secure to IND-RCCA-secure transformation for rerandomizable PKE schemes using a malleable NIZK for the NP relation

$$\mathcal{R}_{\mathsf{ny}} := \{((\mathsf{K}_1, \mathsf{ct}_1, \mathsf{K}_2, \mathsf{ct}_2), (\mathsf{m}, r_1, r_2)) \mid \mathsf{ct}_1 = \mathsf{Enc}(\mathsf{K}_1, \mathsf{m}, e_{\mathsf{exp}}; r_1) \wedge \mathsf{ct}_2 = \mathsf{Enc}(\mathsf{K}_2, \mathsf{m}, e_{\mathsf{exp}}; r_2)\}$$

$\underline{\mathsf{Gen}'(1^\lambda)}$:
$\mathsf{K}_{1,1} \leftarrow \mathsf{Gen}(1^\lambda)$
$\mathsf{K}_{1,2} \leftarrow \mathsf{Gen}(1^\lambda)$
$\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$
return $\mathsf{K}'_1 := (\mathsf{K}_{1,1}, \mathsf{K}_{1,2}, \mathsf{crs})$

$\underline{\mathsf{RotKey}'(\mathsf{K}'_e = (\mathsf{K}_{e,1}, \mathsf{K}_{e,2}, \mathsf{crs}))}$:
$(\mathsf{K}_{e+1,1}, \Delta_{e+1,1}) \leftarrow \mathsf{RotKey}(\mathsf{K}_{e,1})$
$(\mathsf{K}_{e+1,2}, \Delta_{e+1,2}) \leftarrow \mathsf{RotKey}(\mathsf{K}_{e,2})$
return $(\mathsf{K}'_{e+1} := (\mathsf{K}_{e+1,1}, \mathsf{K}_{e+1,2}, \mathsf{crs}),$
$\Delta'_{e+1} := (\Delta_{e+1,1}, \Delta_{e+1,2}, \mathsf{crs}))$

$\underline{\mathsf{Enc}'(\mathsf{K}'_e = (\mathsf{K}_{e,1}, \mathsf{K}_{e,2}, \mathsf{crs}), \mathsf{m}, e_{\mathsf{exp}})}$:
$r_1, r_2 \leftarrow \mathfrak{R}$
$\mathsf{ct}_{e,e_{\mathsf{exp}},1} \leftarrow \mathsf{Enc}(\mathsf{K}_{e,1}, \mathsf{m}, e_{\mathsf{exp}}; r_1)$
$\mathsf{ct}_{e,e_{\mathsf{exp}},2} \leftarrow \mathsf{Enc}(\mathsf{K}_{e,2}, \mathsf{m}, e_{\mathsf{exp}}; r_2)$
$\pi_e \leftarrow \mathsf{Prove}(\mathsf{crs}, (\mathsf{K}_{e,1}, \mathsf{ct}_{e,e_{\mathsf{exp}},1}, \mathsf{K}_{e,2}, \mathsf{ct}_{e,e_{\mathsf{exp}},2}), (\mathsf{m}, r_1, r_2))$
return $\mathsf{ct}'_{e,e_{\mathsf{exp}}} := (\mathsf{ct}_{e,e_{\mathsf{exp}},1}, \mathsf{ct}_{e,e_{\mathsf{exp}},2}, \pi_e)$

$\underline{\mathsf{Update}'(\Delta'_e, \mathsf{ct}'_{e,e_{\mathsf{exp}}})}$:
$(\Delta_{e,1}, \Delta_{e,2}, \mathsf{crs}) := \Delta'_e$
$(\mathsf{ct}_{e,e_{\mathsf{exp}},1}, \mathsf{ct}_{e,e_{\mathsf{exp}},2}, \pi_e) := \mathsf{ct}'_{e,e_{\mathsf{exp}}}$
$r'_1, r'_2 \leftarrow \mathfrak{R}'$
$\mathsf{ct}_{e+1,e_{\mathsf{exp}},1} \leftarrow \mathsf{Update}(\Delta_{e,1}, \mathsf{ct}_{e,e_{\mathsf{exp}},1}; r'_1)$
$\mathsf{ct}_{e+1,e_{\mathsf{exp}},2} \leftarrow \mathsf{Update}(\Delta_{e,2}, \mathsf{ct}_{e,e_{\mathsf{exp}},2}; r'_2)$
$\pi_{e+1} \leftarrow \mathsf{ZKEval}(\mathsf{crs}, T_{r'_1,r'_2}, (\Delta_{e,1}, \mathsf{ct}_{e,e_{\mathsf{exp}},1}, \Delta_{e,2}, \mathsf{ct}_{e,e_{\mathsf{exp}},2}, \pi_e))$
return $\mathsf{ct}'_{e+1,e_{\mathsf{exp}}} := (\mathsf{ct}_{e+1,e_{\mathsf{exp}},1}, \mathsf{ct}_{e+1,e_{\mathsf{exp}},2}, \pi_{e+1})$

$\underline{\mathsf{Dec}'(\mathsf{K}'_e, \mathsf{ct}'_{e,e_{\mathsf{exp}}})}$:
$(\mathsf{K}_{e,1}, \mathsf{K}_{e,2}, \mathsf{crs}) := \mathsf{K}'_e$
$(\mathsf{ct}_{e,e_{\mathsf{exp}},1}, \mathsf{ct}_{e,e_{\mathsf{exp}},2}, \pi_e) := \mathsf{ct}'_{e,e_{\mathsf{exp}}}$
if $\mathsf{Verify}(\mathsf{crs}, (\mathsf{K}_{e,1}, \mathsf{ct}_{e,e_{\mathsf{exp}},1}, \mathsf{K}_{e,2}, \mathsf{ct}_{e,e_{\mathsf{exp}},2}, e_{\mathsf{exp}}, \pi_e)) \stackrel{?}{=} 1$ **then**
  return $\mathsf{Dec}(\mathsf{K}_{e,1}, \mathsf{ct}_{e,e_{\mathsf{exp}},1})$
else
  return $\perp$

Fig. 14. Given an EE-IND-UE-CPA-secure UE $\mathsf{UE} = (\mathsf{Gen}, \mathsf{RotKey}, \mathsf{Enc}, \mathsf{Update}, \mathsf{Dec})$ and a malleable NIZK $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$, we construct an EE-IND-UE-RCCA-secure UE $\mathsf{UE}'[\mathsf{UE}, \Pi] = (\mathsf{Gen}', \mathsf{RotKey}', \mathsf{Enc}', \mathsf{Update}', \mathsf{Dec}')$ via the Naor–Yung transformation applied to UE. The resulting $\mathsf{UE}'[\mathsf{UE}, \Pi]$ is correct, if UE is correct and Π is complete.

and the set of transformations

$$\begin{aligned}
\mathcal{T}_{\mathsf{UE,ny}} &:= \{T_{r'_1,r'_2} = (T_{x,r'_1,r'_2}, T_{w,r'_1,r'_2}) \mid r'_1, r'_2 \in \mathfrak{R}'\} \text{ where} \\
T_{x,r'_1,r'_2} &: (\mathcal{K} \times \mathcal{C})^2 \to (\mathcal{K} \times \mathcal{C})^2 \\
(\mathsf{K}_{e,1}, \mathsf{ct}_{e,1}, \mathsf{K}_{e,2}, \mathsf{ct}_{e,2}) &\mapsto (\mathsf{K}_{e+1,1}, \mathsf{Update}(\Delta_{e+1,1}, \mathsf{ct}_{e,1}; r'_1), \mathsf{K}_{e+1,2}, \mathsf{Update}(\Delta_{e+1,2}, \mathsf{ct}_{e,2}; r'_2)) \\
&\quad \text{with } (\mathsf{K}_{e+1,1}, \Delta_{e+1,1}) \leftarrow \mathsf{RotKey}(\mathsf{K}_{e,1}), (\mathsf{K}_{e+1,2}, \Delta_{e+1,2}) \leftarrow \mathsf{RotKey}(\mathsf{K}_{e,2}) \\
T_{w,r'_1,r'_2} &: (\mathcal{M} \times \mathfrak{R} \times \mathfrak{R}) \to (\mathcal{M} \times \mathfrak{R} \times \mathfrak{R}) \\
(\mathsf{m}, r_1, r_2) &\mapsto (\mathsf{m}, \mathsf{RerandR}(r_1, r'_1), \mathsf{RerandR}(r_2, r'_2)).
\end{aligned}$$

$\mathcal{T}_{\mathsf{UE,ny}}$ is an admissible set of transformations due to the definitions of $\mathsf{RerandR}$.

Theorem 15 (RCCA-security). *If* UE *is* ee-IND-UE-CPA*-secure and* Π *is zero-knowledge and controlled-malleable simulation sound, then* $\mathsf{UE}'[\mathsf{UE}, \Pi]$ *is* ee-IND-UE-RCCA*-secure. More precisely, for every PPT adversary* $\mathcal{A}$ *against the* ee-IND-UE-RCCA *security of* $\mathsf{UE}'[\mathsf{UE}, \Pi]$*, there exist PPT adversaries* $\mathcal{B}$*,* $\mathcal{C}$*,* $\mathcal{D}$*, and* $\mathcal{E}$

$$\begin{aligned}
\mathsf{Adv}^{\text{ee-IND-UE-RCCA}}_{\mathcal{A},\mathsf{UE}'[\mathsf{UE},\Pi]}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\text{ee-IND-UE-CPA}}_{\mathcal{B},\mathsf{UE}}(\lambda) + \frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{C},\Pi}(\lambda) \\
+ \frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{rdp}}_{\mathcal{D},\Pi}(\lambda) + \frac{1}{2} \cdot \mathsf{Adv}^{\text{cm-ss}}_{\mathcal{E},\Pi}(\lambda),
\end{aligned}$$

where q_{dec} *is the number of decryption queries* $\mathcal{A}$ *makes.*

Proof. The proof proceeds along the lines of the proof for RAND-CCA security for PKE. Analogously, the game G_0 is the real ee-IND-UE-RCCA game for $\mathsf{UE}'[\mathsf{UE}, \Pi]$. Let (S_1, S_2) be the zero-knowledge simulator for Π. The game G_1 is identical to G_0, except that crs is generated via $(\mathsf{crs}, \mathsf{td}) \leftarrow S_1(1^\lambda)$ instead of $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$ and the proof during Enc in Chall for the case $b = 0$ is computed as $S_2(\mathsf{crs}, \mathsf{td}, (\mathsf{K}_{e,1}, \mathsf{ct}_{e,e_{\mathsf{exp}},1}, \mathsf{K}_{e,2}, \mathsf{ct}_{e,e_{\mathsf{exp}},2}))$ instead of $\mathsf{Prove}(\mathsf{crs}, (\mathsf{K}_{e,1}, \mathsf{ct}_{e,e_{\mathsf{exp}},1}, \mathsf{K}_{e,2}, \mathsf{ct}_{e,e_{\mathsf{exp}},2}), (\mathsf{m}_0, r_1, r_2))$.

Lemma 23 ($\mathsf{G}_0 \rightsquigarrow \mathsf{G}_1$)**.** *For every PPT adversary* $\mathcal{A}$ *there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]\right| \leq \frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{B},\Pi}(\lambda).$$

Proof. Analogously to Lemma 12, but for the case $b = 0$ in the UE-Chall oracle. Concretely, $\mathcal{B}$ obtains crs from the zero-knowledge challenger and computes all components belonging to UE. To compute the proof in $\mathcal{A}$'s Chall query if $b = 0$, the reduction uses the Prove oracle from the zero-knowledge challenger which will either return an honest proof or a simulated proof. If $b = 1$, which happens with probability $\frac{1}{2}$, the games are identical. □

The next hybrid G_2 is identical to G_1, except that during Update in Chall in the case for $b = 1$, the proof is computed as $S_2(\mathsf{crs}, \mathsf{td}, (\mathsf{K}_{e,1}, \mathsf{ct}_{e-1,e_{\mathsf{exp}},1}, \mathsf{K}_{e,2}, \mathsf{ct}_{e-1,e_{\mathsf{exp}},2}))$ instead of $\mathsf{ZKEval}(\mathsf{crs}, T_{r'_1,r'_2}, ((\Delta_{e,1}, \mathsf{ct}_{e-1,e_{\mathsf{exp}},1}, \Delta_{e,2}, \mathsf{ct}_{e-1,e_{\mathsf{exp}},2}), \pi_e)))$. See that the keys $(\mathsf{K}_{e,1}, \mathsf{K}_{e,2})$ are already available since those are generated together with $(\Delta_{e,1}, \Delta_{e,2})$.

Lemma 24 ($\mathsf{G}_1 \rightsquigarrow \mathsf{G}_2$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]\right| \leq \frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{rdp}}_{\mathcal{B},\Pi}(\lambda).$$

Proof. Analogously to Lemma 18, but for the case $b = 1$ in the UE-Chall oracle. Concretely, $\mathcal{B}$ obtains crs from the strong derivation-privacy challenger and computes all components belonging to UE. To compute the proof in $\mathcal{A}$'s Chall query if $b = 1$, the reduction uses the Prove oracle from the strong derivation-privacy challenger which will either return a proof obtained via ZKEval or a simulated proof. If $b = 0$, which happens with probability $\frac{1}{2}$, the games are identical. □

In the next hybrid G_3, the challenge ciphertext is changed as follows. The ciphertext component $\mathsf{ct}_{e,e_{\mathsf{exp}},2}$ is always a fresh encryption of m_0 instead of being a fresh encryption of m_1 or an update of $\mathsf{ct}_{e-1,e_{\mathsf{exp}},2}$ depending on the challenge bit.

Lemma 25 ($\mathsf{G}_2 \rightsquigarrow \mathsf{G}_3$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{C}$ such that*

$$\left|\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{ee\text{-}IND\text{-}UE\text{-}CPA}}_{\mathcal{C},\mathsf{UE}}.$$

Proof. Analogously to Lemma 19. Note that in G_2 and G_3, we never use $\mathsf{K}_{e^*,2}$. $\mathcal{C}$ is not allowed to query the key from its ee-IND-UE-CPA-security challenger. When the adversary makes its challenge query on its ciphertext part with ($\mathsf{ct}_{e-1,e_{\mathsf{exp}},1}$, $\mathsf{ct}_{e-1,e_{\mathsf{exp}},2}, \pi_{e-1}$) and this ciphertexts decrypts to $\mathsf{m}_1 \neq \perp$, the reduction answers if $b = 0$ with a ciphertext where $\mathsf{ct}^*_{e,b,2}$ is a fresh encryption of m_1. If $b = 1$, it submits $\mathsf{ct}_{e-1,e_{\mathsf{exp}},2}$ to its own challenge oracle and uses the resulting ciphertext as $\mathsf{ct}^*_{e,b,2}$. An adversary that wins with different probability in G_2 than in G_3 can thus be turned into an adversary against ee-IND-UE-CPA-security game for UE. □

The next hybrid G_4 answers all Dec' queries by decrypting with $\mathsf{K}_{e^*,2}$ instead of $\mathsf{K}_{e^*,1}$.

Lemma 26 ($\mathsf{G}_3 \rightsquigarrow \mathsf{G}_4$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{D}$ such that*

$$\left|\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{cm\text{-}ss}}_{\mathcal{D},\Pi}(\lambda).$$

Proof. $\mathcal{D}$ receives crs from the controlled-malleable simulation soundness game and computes the simulated proof for the challenge query using the Prove oracle. If the adversary makes a Dec' query on ($\mathsf{ct}_{e-1,e_{\mathsf{exp}},1}, \mathsf{ct}_{e-1,e_{\mathsf{exp}},2}, \pi_{e-1}$) where $\mathsf{Verify}(\mathsf{crs}, (\mathsf{K}_{e-1,1}, \mathsf{ct}_{e-1,e_{\mathsf{exp}},1}, \mathsf{K}_{e-1,2}, \mathsf{ct}_{e-1,e_{\mathsf{exp}},2}), \pi_{e-1}) = 1$, it computes $\mathsf{m}_1 \leftarrow \mathsf{Dec}(\mathsf{K}_{e-1}, \mathsf{ct}_{e-1,e_{\mathsf{exp}},1})$ and $\mathsf{m}_2 \leftarrow \mathsf{Dec}(\mathsf{K}_{e-1}, \mathsf{ct}_{e-1,e_{\mathsf{exp}},2})$. Due to perfect correctness, no corrrectness error occurred in either of these decryptions (over all Dec' queries). If $\mathsf{m}_1 \neq \mathsf{m}_2$, the reduction outputs (($\mathsf{K}_{e-1,1}, \mathsf{ct}_{e-1,e_{\mathsf{exp}},1}, \mathsf{K}_{e-1,2}$, $\mathsf{ct}_{e-1,2}$), π_e) and wins the controlled-malleable simulation soundness experiment. The argumentation is analogously to Lemma 14. □

In game G_5, the challenge ciphertext component $\mathsf{ct}^*_{e-1,e_{\mathsf{exp}},1}$ is always an encryption of m_0 instead of m_b.

Lemma 27 ($\mathsf{G}_4 \rightsquigarrow \mathsf{G}_5$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{ee\text{-}IND\text{-}UE\text{-}CPA}}_{\mathcal{B},\mathsf{UE}}.$$

The proof is analogous to Lemma 21; note that $\mathsf{K}_{e^*,1}$ is not allowed to be queried.

Lemma 28 (G_5)**.**

$$\Pr[\mathsf{G}_5^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

Proof. G_5 is statistically independent of the challenge bit b. □

Taking all Lemmata 23—28 together, shows Theorem 15. □

E Targeted homomorphic encryption

In this section we present a transformation to achieve the notion of unlinkability and targeted malleability for homomorphic encryption by [14]. This result requires a stronger, non-standard variant of (simulation sound) extractability that gives the adversary and the extractor the same auxiliary input (common auxiliary input model). Looking forward, in the construction this auxiliary input will contain a public key and a ciphertext for the underlying homomorphic encryption scheme.

We emphasize that this security notion is not achievable for arbitrary auxiliary input (assuming iO exists), as shown in [9], and therefore this transformation is *not generic*. Whether this assumption is reasonable therefore has to be decided for each combination of homomorphic encryption and SNARK (or assumption the SNARK is based on) separately.

The proof in [14] claims to achieve the notion of targeted malleability without such auxiliary inputs. However, their proof contains several gaps that we explain below in more detail. The only way we see to repair their proof is to assume simulation extractability *in the common auxiliary input model*.

E.1 Preliminaries

Definition 29 (Controlled-malleable simulation extractability in the common auxiliary input model). *A malleable non-interactive zero-knowledge argument system $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ for an NP relation $\mathcal{R}$ and an admissible set of transformations $\mathcal{T}$ is* controlled-malleable simulation extractable in the common auxiliary input model *for auxiliary input distribution $\mathcal{Z}$ if for every PPT adversary $\mathcal{A}$ there exists a PPT extractor $\mathcal{E}$ such that*

$$\mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{A},\mathcal{E},\mathcal{Z},\Pi}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{cm\text{-}sse}}_{\mathcal{A},\mathcal{E},\mathcal{Z},\Pi}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

for a negligible function negl *where the* $\mathsf{cm\text{-}sse}$ *experiment is defined in Figure 15.*

$\underline{\mathsf{Exp}^{\text{cm-sse}}_{\mathcal{A},\mathcal{E},\mathcal{Z},\Pi=(\mathsf{CRSGen},\mathsf{Prove},\mathsf{Verify})}(\lambda)}$:
$Q_{\mathsf{sim}} := \emptyset$
$(\mathsf{crs}, \mathsf{td}) \leftarrow S_1(1^\lambda)$
$z \xleftarrow{\$} \mathcal{Z}$
$(x, \pi) \leftarrow \mathcal{A}^{\textsc{Prove}(\cdot)}(1^\lambda, \mathsf{crs}, z; r)$
$E \leftarrow \mathcal{E}(1^\lambda, \mathsf{crs}, Q_{\mathsf{sim}}, z, r)$
if $\mathsf{checkExplanation}(x, E) = 1$ **then**
  return 0
else
  return 1

$\underline{\textsc{Prove}(x)}$:
$\pi \leftarrow S_2(\mathsf{crs}, \mathsf{td}, x)$
$Q_{\mathsf{sim}} := Q_{\mathsf{sim}} \cup \{(x, \pi)\}$
return π

$\underline{\mathsf{checkExplanation}(x, E)}$:
if $(x, \_) \in Q_{\mathsf{sim}}$ **then**
  return 1
if $\exists w : (x, w) \in E \wedge (x, w) \in \mathcal{R}$ **then**
  return 1
if $\exists T = (T_x, T_w), x_1, \ldots, x_n : (x, (T, x_1, \ldots, x_n)) \in E$ **then**
  if $T \in \mathcal{T} \wedge x = T_x(x_1, \ldots, x_n)$ **then**
    $E' := E \setminus \{(x, w) \in E\}$
    return $\bigwedge_{i=1}^{n} \mathsf{checkExplanation}(x_i, E')$
return 0

Fig. 15. The simulation extractability experiment for a malleable NIZK $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Verify})$ for NP-relation $\mathcal{R}$ and set of transformations $\mathcal{T}$ with zero-knowledge simulator $S = (S_1, S_2)$ in the common auxiliary input model. Differences to the game in Figure 4 are highlighted gray.

Next, we recall the definition of homomorphic encryption and the desired properties. For definitions of PKE and IND-CPA security, see Section C.1.

Definition 30 (Homorphic encryption). *A* homorphic encryption (HE) *scheme for circuit class $\mathfrak{C}$ is a PKE with the following additional algorithm:*

- Eval *inputs a public key* pk*, a list of ciphertext* $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ *and a circuit* $C \in \mathfrak{C}$ *computing a function* $f_C : \mathcal{M}^n \to \mathcal{M}$ *and outputs a new ciphertext* ct'*.*

Let $\mathfrak{R}$ be the randomness space for Enc and $\mathfrak{R}'$ the randomness space for Eval. To simplify our construction, we also assume that there exists an efficient and deterministic algorithm EvalR that inputs random coins $r_1, \ldots, r_n \in \mathfrak{R}$ and $r' \in \mathfrak{R}'$ and satisfies for all $\mathsf{pk} \in \mathcal{PK}$, all $\mathsf{m}_1, \ldots, \mathsf{m}_n \in \mathcal{M}$, and all $C \in \mathfrak{C}$

$$\mathsf{Eval}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_1; r_1), \ldots, \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_n; r_n); r') = \mathsf{Enc}(\mathsf{pk}, f_C(\mathsf{m}_1, \ldots, \mathsf{m}_n); \mathsf{EvalR}(r_1, \ldots, r_n, r')).$$

We always require the following two properties to hold for an HE scheme.

Definition 31 (Correctness). *An HE scheme* $\mathsf{HE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *is* correct *if we have*

$$|\Pr[\forall \mathsf{m} \in \mathcal{M} : \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = \mathsf{m} \mid (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m})]| \geq 1 - \mathsf{negl}(\lambda)$$

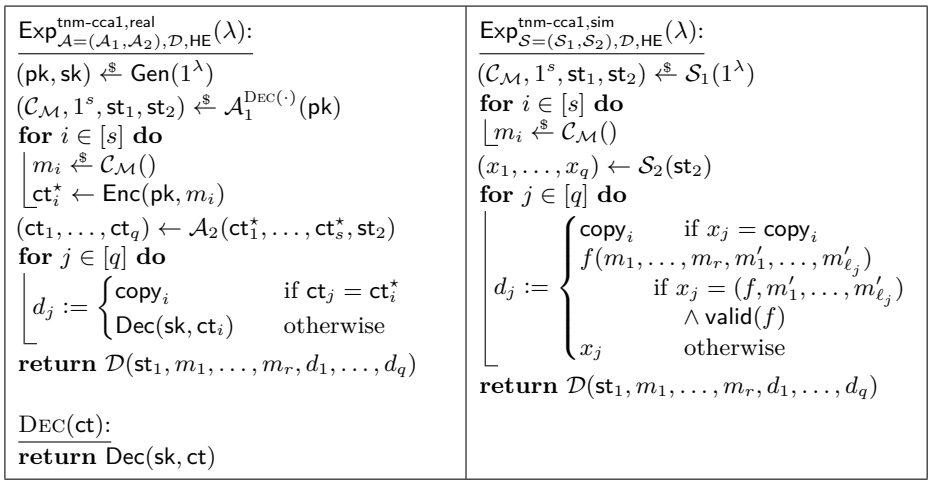

Fig. 16. The security experiments for TNM-CCA1 security for a homomorphic encryption scheme $\mathsf{HE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$. In the experiment, $\mathcal{C}_\mathcal{M}$ is a circuit that samples messages.

and

$$\begin{aligned}&\Pr[\forall \mathsf{ct}_1,\ldots,\mathsf{ct}_n \in \mathcal{C} \forall C \in \mathfrak{C} : (\forall i \in [n] : \mathsf{Dec}(\mathsf{sk},\mathsf{ct}_i) = \mathsf{m}_i \neq \bot) \\ &\implies \mathsf{Dec}(\mathsf{sk},\mathsf{Eval}(\mathsf{pk},\mathsf{ct}_1,\ldots,\mathsf{ct}_n,C)) = f_C(\mathsf{m}_1,\ldots,\mathsf{m}_n) \mid (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)]| \\ &\qquad\qquad \geq 1 - \mathsf{negl}(\lambda).\end{aligned}$$

Next, we recall the definition of TNM-CCA1 security by [14], adapted to functions of arbitrary arity. For this definition, we say that a function f is valid, denoted by $\mathsf{valid}(f)$, if f can be described by a composition of (polynomially many) functions computed by circuits in $\mathfrak{C}$ (possibly ignoring parts of its input). We simplified the game by having the simulator S_2 directly output messages, not ciphertexts that are then decrypted.

Definition 32 (TNM-CCA1 security). *An HE scheme* HE *for function class* $\mathfrak{C}$ *is TNM-CCA1 secure iff for every two-stage PPT algorithm (the adversary)* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a two-stage PPT algorithm (the simulator)* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *such that for every PPT algorithm (the distinguisher)* $\mathcal{D}$*.*

$$\mathsf{Adv}^{\mathsf{tnm\text{-}cca1}}_{\mathcal{A},\mathcal{S},\mathcal{D},\mathsf{HE}}(\lambda) := 2\left|\Pr[\mathsf{Exp}^{\mathsf{tnm\text{-}cca1,real}}_{\mathcal{A},\mathcal{D},\mathsf{HE}}(\lambda) \Rightarrow 1] - \Pr[\mathsf{Exp}^{\mathsf{tnm\text{-}cca1,sim}}_{\mathcal{S},\mathcal{D},\mathsf{HE}}(\lambda) \Rightarrow 1]\right| \leq \mathsf{negl}(\lambda)$$

for a negligible function negl*. The corresponding experiments are given in Figure 16.*

Finally, we recall the definition of unlinkability by [62].

$\underline{\mathsf{Exp}^{\mathsf{unlink}}_{\mathcal{A},\mathsf{HE}}(\lambda)}$:	$\underline{\textsc{Chal}(\mathsf{ct}_1,\ldots,\mathsf{ct}_n, C\in\mathfrak{C})}$: //only one query

$(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{Gen}(1^\lambda)$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\textsc{Chal}(\cdot,\cdot),\textsc{Dec}(\cdot)}(\mathsf{pk})$
return $b = b'$
	for $i \in [n]$ **do**
	  $\mathsf{m}_i \leftarrow \mathsf{Dec}(\mathsf{sk},\mathsf{ct}_i)$
	  if $\mathsf{m}_i = \perp$ **then return** $\perp$
	if $b = 0$ **then**
	  return $\mathsf{ct}^\star \leftarrow \mathsf{Eval}(\mathsf{pk},\mathsf{ct}_1,\ldots,\mathsf{ct}_n,C)$
	else
	  return $\mathsf{ct}^\star \leftarrow \mathsf{Enc}(\mathsf{pk}, f_C(\mathsf{m}_1,\ldots,\mathsf{m}_n))$

$\underline{\textsc{Dec}(\mathsf{ct})}$: $\mathsf{m} \leftarrow \mathsf{Dec}(\mathsf{sk},\mathsf{ct})$ **return** m	

Fig. 17. The security experiments for unlinkability for a homomorphic encryption scheme $\mathsf{HE} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Eval},\mathsf{Dec})$ for function class $\mathfrak{C}$.

Definition 33 (Unlinkability). *An HE scheme* HE *for function class* $\mathfrak{C}$ *is* unlinkable *iff for all PPT adversaries* $\mathcal{A}$

$$\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{A},\mathsf{HE}}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{unlink}}_{\mathcal{A},\mathsf{HE}}(\lambda) \Rightarrow 1] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

for a negligible function negl. *The experiment is given in Figure 17.*

We present the Naor–Yung transform for HE in Figure 18. The transformation uses a malleable NIZK for the NP relation

$$\mathcal{R}_{\mathsf{ny}} := \{((\mathsf{pk}_1,\mathsf{ct}_1,\mathsf{pk}_2,\mathsf{ct}_2),(\mathsf{m},r_1,r_2)) \mid \mathsf{ct}_1 = \mathsf{Enc}(\mathsf{pk}_1,\mathsf{m};r_1) \wedge \mathsf{ct}_2 = \mathsf{Enc}(\mathsf{pk}_2,\mathsf{m};r_2)\}$$

and the set of transformations

$$\mathcal{T}_{\mathsf{ny}} := \{T_{C,r,r'} = (T_{x,C,r,r'}, T_{w,C,r,r'}) \mid C \in \mathfrak{C}, r, r' \in \mathfrak{R}'\} \quad \text{where}$$

$$T_{x,r,r',C} : ((\mathcal{PK}\times\mathcal{C})^2)^n \to (\mathcal{PK}\times\mathcal{C})^2$$

$$(\mathsf{pk}_i,\mathsf{ct}_i,\mathsf{pk}'_i,\mathsf{ct}'_i)_{1\leq i\leq n} \mapsto \begin{cases} (\mathsf{pk}_1, \mathsf{Eval}(\mathsf{pk}_1,(\mathsf{ct}_i)_{1\leq i\leq n},C;r), \mathsf{pk}'_1, \mathsf{Eval}(\mathsf{pk}'_1,(\mathsf{ct}'_i)_{1\leq i\leq n},C;r')) \\ \qquad \text{if } \mathsf{pk}_1 = \cdots = \mathsf{pk}_n \wedge \mathsf{pk}'_1 = \cdots = \mathsf{pk}'_n \\ \perp \qquad \text{otherwise} \end{cases}$$

$$T_{w,r,r',C} : (\mathcal{M}\times\mathfrak{R}\times\mathfrak{R})^n \to (\mathcal{M}\times\mathfrak{R}\times\mathfrak{R})$$

$$(\mathsf{m}_i,r_i,r'_i)_{1\leq i\leq n} \mapsto (f_C(\mathsf{m}_1,\ldots,\mathsf{m}_n), \mathsf{EvalR}(r_1,\ldots,r_n,r), \mathsf{EvalR}(r'_1,\ldots,r'_n,r')).$$

Clearly, the resulting $\mathsf{HE}'[\mathsf{HE},\Pi]$ is correct, if HE is correct and Π is complete.

Theorem 16 (TNM-CCA1-security). *If* HE *is perfectly*[22] *correct and* IND-CPA*-secure, and* Π *is zero-knowledge and controlled-malleable simulation extractable in the common auxiliary input model for a distribution* $\mathcal{Z}$ *explained below,*

[22] For non-perfect correctness, we can use the derandomization technique of [34] to get a similar result.

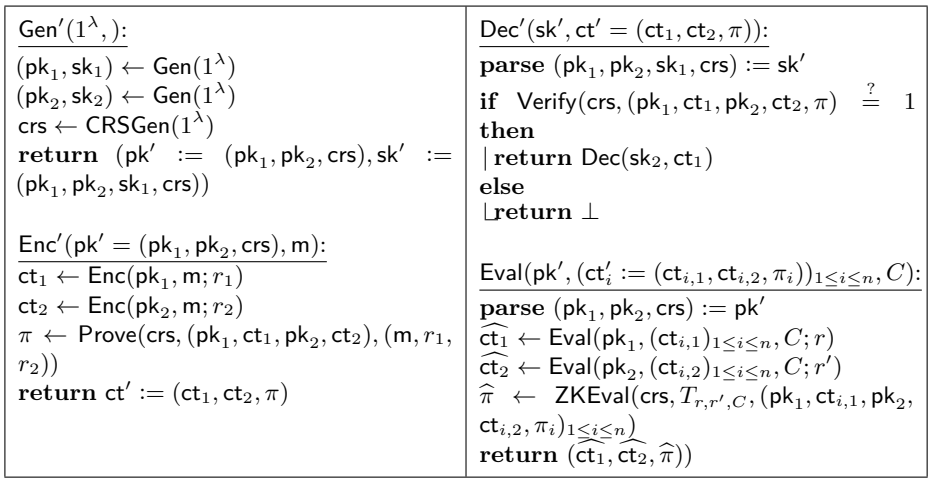

Fig. 18. The Naor–Yung transformation that turns a HE $\mathsf{HE} = (\mathsf{Gen},\mathsf{Enc},\mathsf{Eval},\mathsf{Dec})$ and a malleable NIZK $\Pi = (\mathsf{CRSGen},\mathsf{Prove},\mathsf{Verify})$ into a TNM-CCA1 secure and unlinkable HE $\mathsf{HE}'[\mathsf{HE},\Pi] = (\mathsf{Gen}',\mathsf{Enc}',\mathsf{Eval}',\mathsf{Dec}')$.

then $\mathsf{HE}'[\mathsf{HE},\Pi]$ *is* TNM-CCA1*-secure. More precisely, for every two-stage PPT adversary* $\mathcal{A} = (\mathcal{A}_1,\mathcal{A}_2)$ *against the* TNM-CCA1 *security of* $\mathsf{HE}'[\mathsf{HE},\Pi]$*, there exists a two-stage simulator* $\mathcal{S} = (\mathcal{S}_1,\mathcal{S}_2)$ *such that for every PPT distinguisher* $\mathcal{D}$ *there exist PPT adversaries* $\mathcal{B}$*,* $\mathcal{C}$*, and* $\mathcal{H}$ *such that for all PPT extractors* $\mathcal{E}$

$$\mathsf{Adv}^{\mathsf{tnm\text{-}cca1}}_{\mathcal{A},\mathcal{S},\mathcal{D},\mathsf{HE}'[HE,\Pi]}(\lambda) \le 2s\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{B},\mathsf{HE}}(\lambda) + \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{C},\Pi}(\lambda) + (2sq^2+2)\mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{H},\mathcal{E},\mathcal{Z},\Pi}(\lambda),$$

where s *is the number of challenge ciphertexts that* $\mathcal{A}_1$ *requested and* q *is the number of ciphertexts that* $\mathcal{A}_2$ *outputs.*

The proof uses ideas of [14], who in turn build upon [6, 31].

Proof. We start by describing the algorithm $\mathcal{S} = (\mathcal{S}_1,\mathcal{S}_2)$. Therefore let $S = (S_1, S_2)$ be the zero-knowledge simulator for Π.

The algorithm $\mathcal{S}_1$ samples $(\mathsf{pk}',\mathsf{sk}') = ((\mathsf{pk}_1,\mathsf{pk}_2,\mathsf{crs}),(\mathsf{sk}_1,\mathsf{sk}_2))$ via $(\mathsf{pk}_1, \mathsf{sk}_1),(\mathsf{pk}_2,\mathsf{sk}_2) \leftarrow \mathsf{Gen}(1^\lambda;r_1)$ and $(\mathsf{crs},\mathsf{td}) \leftarrow S_1(1^\lambda;r_2)$. It then simulates $\mathcal{A}_1^{\mathrm{Dec}(\cdot)}(\mathsf{pk}';r_3)$ and answers all $\mathrm{Dec}(\cdot)$ queries with sk_1. When $\mathcal{A}_1$ outputs $(\mathcal{C}_\mathcal{M}, 1^s,\mathsf{st}_1,\mathsf{st}_2)$, $\mathcal{S}_1$ outputs $(\mathcal{C}_\mathcal{M},1^s,\mathsf{st}_1,\mathsf{st}'_2 := (\mathcal{C}_\mathcal{M},1^s,r_1,r_2,r_3,\mathsf{pk},\mathsf{pk}',\mathsf{sk}',\mathsf{st}_2))$.

The algorithm $\mathcal{S}_2$ inputs $\mathsf{st}'_2 := (\mathcal{C}_\mathcal{M},r_1,r_2,r_3,\mathsf{pk},\mathsf{pk}',\mathsf{sk}',\mathsf{st}_2))$ and samples for all $i \in [r]$ a message $\mathsf{m}'_i \leftarrow \mathcal{C}_\mathcal{M}$ and encrypts it as $\mathsf{ct}^\star_i = (\mathsf{ct}^\star_{i,1},\mathsf{ct}^\star_{i,2},\pi^\star_i)$ with $\mathsf{ct}^\star_{i,1} \leftarrow \mathsf{Enc}(\mathsf{pk}_1,\mathsf{m}'_i)$, $\mathsf{ct}^\star_{i,2} \leftarrow \mathsf{Enc}(\mathsf{pk}_2,\mathsf{m}'_i)$ and $\pi^\star_i \leftarrow S_2(\mathsf{crs},\mathsf{td},(\mathsf{pk}_1,\mathsf{ct}^\star_{i,1},\mathsf{pk}_2,\mathsf{ct}^\star_{i,2}))$. It then runs $\mathcal{A}_2(\mathsf{ct}^\star_i,\ldots,\mathsf{ct}^\star_s,\mathsf{st}_2)$ to obtain $(\mathsf{ct}'_1,\ldots,\mathsf{ct}'_q)$ with $\mathsf{ct}'_i = (\mathsf{ct}_{i,1},\mathsf{ct}_{i,2},\pi_i)$.

Now let $\mathcal{A}^{\mathrm{Prove}}_{\mathsf{sse}}$ be the algorithm that inputs 1^λ and crs' and then simulates the behavior of $\mathcal{S}_1$, except that it replaces the crs in $\mathsf{pk}' = (\mathsf{pk}_1,\mathsf{pk}_2,\mathsf{crs})$ with its own input and computing the simulated proofs $\pi^\star_i$ with its Prove oracle. It

then simulates $\mathcal{S}_2$ up to the point described so far, samples a random index $j \in [q]$ and outputs $((\mathsf{pk}_1, \mathsf{ct}_{j,1}, \mathsf{pk}_2, \mathsf{ct}_{j,2}), \pi_j)$. The controlled-malleable simulation extractability game for Π guarantees the existence of a PPT extractor $\mathcal{E}_{\mathsf{sse}}$ with $\mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{A}_{\mathsf{sse}},\mathcal{E}_{\mathsf{sse}},\Pi}(\lambda) \leq \mathsf{negl}(\lambda)$.

Next, we define for each $j \in [q]$ the adversary $\mathcal{A}_j$ that works like $\mathcal{A}_{\mathsf{sse}}$, except that it uses the fixed value j instead of sampling it at random. Now consider the extractor $\mathcal{E}_j$ that inputs crs and a random tape r for $\mathcal{A}_j$. $\mathcal{E}_j$ runs $\mathcal{E}'_{\mathsf{sse}}(\mathsf{crs}, r')$, where r' is a random tape that is identical to r, except for filling in random coins at the appropriate positions that make $\mathcal{E}'_{\mathsf{sse}}(\mathsf{crs}, r')$ choose j as the random index. The union bound now guarantees

$$\mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{A}_j,\mathcal{E}_j,\mathcal{Z},\Pi}(\lambda) \leq q\mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{A}_{\mathsf{sse}},\mathcal{E}_{\mathsf{sse}},\mathcal{Z},\Pi}(\lambda).$$

This construction guarantees that all extractors $\mathcal{E}_j$ have the same runtime and success probability bound. This is necessary to ensure that running all of them is still in polynomial time and that all of them succeed with overwhelming probability.

The simulator $\mathcal{S}_2$ now proceeds by concatenating $r_1 \| r_2 \| r_3$ with the random coins that $\mathcal{S}_2$ has used so far and removing all coins that where used to generate simulated proofs, we obtain a random tape r for $\mathcal{A}_j$. The simulation extractability of the SNARK guarantees the existence of an extractor $\mathcal{E}_j$ for $\mathcal{A}_j$ that inputs 1^λ, crs and r and outputs an explanation E_j. The simulator runs for all $j \in [q]$ $E_j \leftarrow \mathcal{E}_j(1^\lambda, \mathsf{crs}, \emptyset, r)$. $\mathcal{S}_2$ then outputs $x_1, \ldots, x_q$ where

- $x_j = \perp$, if E_j is not a valid explanation
- $x_j = \mathsf{copy}_i$, if $\exists i \in [s] : \mathsf{ct}_j = \mathsf{ct}^\star_i$
- $x_j = \mathsf{m}$ if $\exists w = (\mathsf{m}, r_1, r_2) : ((\mathsf{pk}_1, \mathsf{ct}_{j,1}, \mathsf{pk}_2, \mathsf{ct}_{j,2}), w) \in E_j$
- $x_j = (f, m'_1, \ldots, m'_{\ell_j})$ otherwise.

In the last case, the explanation E_j explains recursively how the ciphertext was computed by allowed homomorphic operations. The function f is obtained by concatenating these operations. For leafs that contain a ciphertext $\mathsf{ct}^\star_i$, we use the i-th input of f. For leafs that contain a different ciphertext with witness (m, r_1, r_2), we add m to the list of additional inputs and use this as input for f.

Next, we show via a hybrid argument that the distribution $\mathcal{G}_0$ produced by the adversary in the real game is computationally indistinguishable from the distribution $\mathcal{G}_{5,s}$ produced by the simulator in the simulated game.

Therefore, let us define the distribution $\mathcal{G}_1$ as the distribution that is sampled as in the game $\mathcal{G}_0$, except that crs is sampled via $(\mathsf{crs}, \mathsf{td}) \leftarrow S_1(1^\lambda)$, where (S_1, S_2) is the zero-knowledge simulator for Π. Moreover, for all $i \in [s]$ we generate in i-th challenge ciphertext $(\mathsf{ct}^\star_i = (\mathsf{ct}^\star_{i,1}), \mathsf{ct}^\star_{i,2}, \pi^\star_i)$ the proof $\pi^\star_i$ via $\pi^\star_i \xleftarrow{\$} S_2(\mathsf{crs}, \mathsf{td}, x = (\mathsf{pk}_1, \mathsf{ct}^\star_{i,1}), \mathsf{pk}_2, \mathsf{ct}^\star_{i,2}))$.

Lemma 29. *There exists a PPT adversary $\mathcal{C}$ such that*

$$\left| \Pr_{z \xleftarrow{\$} \mathcal{G}_0} [\mathcal{D}(z) \Rightarrow 1] - \Pr_{z \xleftarrow{\$} \mathcal{G}_1} [\mathcal{D}(z) \Rightarrow 1] \right| \leq \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{C},\Pi}(\lambda)$$

Proof. The reduction embeds the crs it receives in pk', and computes the proofs for all the challenge ciphertexts with the PROVE oracle of the zero-knowledge game (which either returns a real or a simulated proof). □

The distribution $\mathcal{G}_2$ is identical to $\mathcal{G}_1$, except that the values d_j for new ciphertext (i.e. where $d_j \neq \mathsf{copy}_i$) are computed via SSE extractors of the underlying malleable SNARK instead of using decryption. To compute d_j, we run the extractor $\mathcal{E}_j$ we defined in the definition of $\mathcal{S}_2$ on crs and the random coins used to generate $\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{m}_i, \mathsf{ct}_i^\star$ for all $i \in [r]$ and to simulate the adversary to obtain an explanation E_j. The value d_j is then computed from E_j as described for the simulator $\mathcal{S}_2$.

Lemma 30. *There exists a PPT adversary $\mathcal{H}$ such that for all PPT extractors $\mathcal{E}$*

$$\left| \Pr_{z \xleftarrow{\$} \mathcal{G}_1} [\mathcal{D}(z) \Rightarrow 1] - \Pr_{z \xleftarrow{\$} \mathcal{G}_2} [\mathcal{D}(z) \Rightarrow 1] \right| \leq q^2 \mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{H},\mathcal{E},\Pi}(\lambda)$$

Proof. In $\mathcal{G}_1$, we run the q Π-SSE extractors $\mathcal{E}_1, \ldots, \mathcal{E}_q$ for the underlying malleable NIZKs. In each of these runs we use an honestly sampled witness and uniformly random coins, thus each extractor will return a valid explanation except with probability at most $q\mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{H},\mathcal{E},\Pi}(\lambda)$ if the proof π_j was a valid proof for the statement $(\mathsf{pk}_1, \mathsf{ct}_{j,1}, \mathsf{pk}_2, \mathsf{ct}_{j,2})$. If the proof was not valid, we have $d_j = \perp$ in both $\mathcal{G}_0$ and $\mathcal{G}_1$ We need to argue that if the run of the extractor $\mathcal{E}_j$ yields a valid explanation E_j for the statement $(\mathsf{pk}_1, \mathsf{ct}_{j,1}, \mathsf{pk}_2, \mathsf{ct}_{j,2})$, than the value d_j is distributed as in $\mathcal{G}_0$. The language $\mathcal{R}_{\mathsf{ny}}$ and the set of transformation $\mathcal{T}_{\mathsf{ny}}$ guarantee that $\mathsf{ct}_{j,1}$ and $\mathsf{ct}_{j,2}$ are encryptions of the same message m and obtained by encryption and homomorphic evaluation operations. By perfect correctness of HE, we have $d_j = \mathsf{m}$ in both cases. □

The next distribution $\mathcal{G}_{3,i}$ for $i \in [0, s]$ is identical to $\mathcal{G}_2$, except that for $\hat{\imath} \in [i]$ we sample $\mathsf{m}'_{\hat{\imath}} \xleftarrow{\$} \mathcal{C}_{\mathcal{M}}$ (in addition to the messages $\mathsf{m}_1, \ldots, \mathsf{m}_s \xleftarrow{\$} \mathcal{C}_{\mathcal{M}}$) and we compute the $\hat{\imath}$-th challenge ciphertext $\mathsf{ct}^\star_{\hat{\imath}} = (\mathsf{ct}^\star_{\hat{\imath},1}, \mathsf{ct}^\star_{\hat{\imath},2}, \pi^\star_{\hat{\imath}})$ as $\mathsf{ct}^\star_{\hat{\imath},1} \leftarrow \mathsf{Enc}(\mathsf{pk}_1, \mathsf{m}'_{\hat{\imath}})$ (in $\mathcal{G}_2$ this was done as $\mathsf{ct}^\star_{\hat{\imath},1} \leftarrow \mathsf{Enc}(\mathsf{pk}_1, \mathsf{m}_{\hat{\imath}})$), $\mathsf{ct}^\star_{\hat{\imath},2} \leftarrow \mathsf{Enc}(\mathsf{pk}_2, \mathsf{m}_{\hat{\imath}})$ (unchanged), and $\pi^\star_{\hat{\imath}}$ via a call to PROVE (unchanged).

Clearly, the distributions $\mathcal{G}_2$ and $\mathcal{G}_{3,0}$ are identical.

Lemma 31. *There exist PPT adversaries $\mathcal{B}$ and $\mathcal{H}$ such that for all PPT extractors $\mathcal{E}$ and for all $i \in [s]$*

$$\left| \Pr_{z \xleftarrow{\$} \mathcal{G}_{3,i-1}} [\mathcal{D}(z) \Rightarrow 1] - \Pr_{z \xleftarrow{\$} \mathcal{G}_{3,i}} [\mathcal{D}(z) \Rightarrow 1] \right| \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{B},\mathsf{HE}}(\lambda) + sq^2 \mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{H},\mathcal{E},\mathcal{Z},\Pi}(\lambda).$$

Proof. Before we describe the reduction we construct an extractor for the SNARK that uses the auxiliary input.

Therefore, we define algorithm $\mathcal{A}'^{\text{PROVE}}_{\mathsf{sse}}$ which will serve as the adversary for the simulation extractability in the common auxiliary input model for Π as follows. The algorithm gets as input a crs crs and the auxiliary input $z = (\mathsf{pk}_1, \mathsf{ct}^\star)$. The algorithm then samples a public key $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{Gen}(1^\lambda)$ and runs $(\mathcal{C}_{\mathcal{M}}, 1^s, \mathsf{st}_1, \mathsf{st}_2) \leftarrow \mathcal{A}_1^{\text{DEC}(\cdot)}(\mathsf{pk}' = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs}))$. Note that $\mathcal{A}'^{\text{PROVE}}_{\mathsf{sse}}$ can answer

the decryption queries with just knowing sk_2. It then samples a random index $i^\star \xleftarrow{\$} [s]$. For all $\hat{\imath} \in [i^\star]$ it samples $\mathsf{m}_{\hat{\imath}}, \mathsf{m}'_{\hat{\imath}} \xleftarrow{\$} \mathcal{C}_{\mathcal{M}}$, $\mathsf{ct}^\star_{\hat{\imath},1} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_1, \mathsf{m}'_{\hat{\imath}})$, except when $\hat{\imath} = i^\star$. In this case it sets $\mathsf{ct}^\star_{i^\star,1} := \mathsf{ct}^\star$. For all $\hat{\imath} \in [i^\star + 1, s]$ it samples $\mathsf{ct}^\star_{\hat{\imath},1} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_1, \mathsf{m}_{\hat{\imath}})$. It continues to sample for all $\hat{\imath} \in [s]$ values $\mathsf{ct}^\star_{\hat{\imath},2} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_2, \mathsf{m}_{\hat{\imath}})$, $\pi^\star_{\hat{\imath}} \xleftarrow{\$} \textsc{Prove}(\mathsf{crs}, (\mathsf{pk}_1, \mathsf{ct}^\star_{\hat{\imath},1}, \mathsf{pk}_2, \mathsf{ct}^\star_{\hat{\imath},2}))$, and stores $\mathsf{ct}^\star_{\hat{\imath}} := (\mathsf{ct}^\star_{\hat{\imath},1}, \mathsf{ct}^\star_{\hat{\imath},2}, \pi^\star_{\hat{\imath}})$. Next, it runs $(\mathsf{ct}_1, \ldots, \mathsf{ct}_q) \leftarrow \mathcal{A}_2(\mathsf{ct}^\star_1, \ldots, \mathsf{ct}^\star_s, \mathsf{st}_2)$. Finally, it samples a random $j \in [q]$ and outputs ct_j.

Now let $\mathcal{Z}$ be the distribution that samples $\mathsf{pk}_0 \leftarrow \mathsf{Gen}(1^\lambda)$, $\mathsf{m} \xleftarrow{\$} \mathcal{C}_{\mathcal{M}}$ and $\mathsf{ct}^\star \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m})$.

The controlled-malleable simulation extractable in the common auxiliary input model of Π now guarantees the existence of an extractor $\mathcal{E}'_{\mathsf{sse}}$ that inputs crs and a random tape r for $\mathcal{A}'_{\mathsf{sse}}$ with $\mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{A}'_{\mathsf{sse}},\mathcal{E}'_{\mathsf{sse}},\mathcal{Z},\Pi}(\lambda) \leq \mathsf{negl}(\lambda)$.

Next, we define the adversary $\mathcal{A}_{i^\star,j}$ that works like $\mathcal{A}'_{\mathsf{sse}}$, except that it uses the fixed values $i^\star$ and j instead of sampling them at random. Now consider the extractor $\mathcal{E}_{i^\star,j}$ that inputs crs and a random tape r for $\mathcal{A}_{i^\star,j}$. The extractor runs $\mathcal{E}'_{\mathsf{sse}}(\mathsf{crs}, r')$, where r' is a random tape that is identical to r, except for filling in random coins at the appropriate positions that make $\mathcal{E}'_{\mathsf{sse}}(\mathsf{crs}, r')$ choose $i^\star$ and j as the random indices. The union bound now guarantees

$$\mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{A}_{i^\star,j},\mathcal{E}_{i^\star,j},\mathcal{Z},\Pi}(\lambda) \leq sq\mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{A}'_{\mathsf{sse}},\mathcal{E}'_{\mathsf{sse}},\mathcal{Z},\Pi}(\lambda).$$

Next, we can describe the reduction to IND-CPA security of HE. The reduction receives a public key pk_1 which is used to compute $\mathsf{pk}' = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{crs})$ ($(\mathsf{pk}_2, \mathsf{sk}_2)$ and crs are sampled as in the game). The security reduction then continues to simulate the game by running $(\mathcal{C}_{\mathcal{M}}, 1^s, \mathsf{st}_1, \mathsf{st}_2) \leftarrow \mathcal{A}_1^{\textsc{Dec}(\cdot)}(\mathsf{pk}')$, hereby simulating the $\textsc{Dec}(\cdot)$ queries with sk_2, and sampling $\mathsf{m}_1, \ldots, \mathsf{m}_s, \mathsf{m}'_1, \ldots \mathsf{m}_i \leftarrow \mathcal{C}_{\mathcal{M}}$. The reduction then makes its challenge query as $\mathsf{ct}^\star \leftarrow \textsc{Chal}(\mathsf{m}_i, \mathsf{m}'_i)$ and embeds the resulting ciphertext in the challenge ciphertexts for the TNM-CCA1 game as $\mathcal{A}'_{\mathsf{sse}}$: For all $\hat{\imath} \in [i^\star]$ it samples $\mathsf{ct}^\star_{\hat{\imath},1} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_1, \mathsf{m}'_{\hat{\imath}})$, except when $\hat{\imath} = i^\star$. In this case it sets $\mathsf{ct}^\star_{i^\star,1} := \mathsf{ct}^\star$. For all $\hat{\imath} \in [i^\star + 1, s]$ it samples $\mathsf{ct}^\star_{\hat{\imath},1} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_1, \mathsf{m}_{\hat{\imath}})$. It continues to sample for all $\hat{\imath} \in [s]$ $\mathsf{ct}^\star_{\hat{\imath},2} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_2, \mathsf{m}_{\hat{\imath}})$, $\pi^\star_{\hat{\imath}} \xleftarrow{\$} \textsc{Prove}(\mathsf{crs}, (\mathsf{pk}_1, \mathsf{ct}^\star_{\hat{\imath},1}, \mathsf{pk}_2, \mathsf{ct}^\star_{\hat{\imath},2}))$ and stores $\mathsf{ct}^\star_{\hat{\imath}} := (\mathsf{ct}^\star_{\hat{\imath},1}, \mathsf{ct}^\star_{\hat{\imath},2}, \pi^\star_{\hat{\imath}})$.

Next, it runs $(\mathsf{ct}_1, \ldots, \mathsf{ct}_q) \leftarrow \mathcal{A}_2(\mathsf{ct}^\star_1, \ldots, \mathsf{ct}^\star_s, \mathsf{st}_2)$. Now, both $\mathcal{G}_{3,i-1}$ and $\mathcal{G}_{3,i}$ would use for all $j \in [q]$ the extractor $\mathcal{E}_j$ to extract an explanation for the proof contained in ct_j. However, the reduction is *unable* to do so, because it does not know the randomness used to generate pk_1 and $\mathsf{ct}^\star$. However, the extractor $\mathcal{E}_j$ requires this randomness as input.

Therefore, the reduction uses the extractor $\mathcal{E}_{i,j}$ with auxiliary input $z = (\mathsf{pk}_1, \mathsf{ct}^\star)$. For this extractor, it can compute a random tape by taking its own random tape and removing the random coins used to generate the crs and the simulated proofs (The extractor is not expecting random coins for these values, because $\mathcal{A}'_{\mathsf{sse}}$ gets the crs as input and generates the simulated proofs with its oracle.) The reduction then runs for each $j \in [s]$ the extractors $\mathcal{E}_{i,j}$ and computes the output value d_j from the explanation, like the simulator does.

It remains to show that the usage of the extractor $\mathcal{E}_{i,j}$ (instead of $\mathcal{E}_j$) does not change the values d_j given to the distinguisher. Assume that proof in the

ciphertext ct_j is valid and both $\mathcal{E}_j$ (on its respective input, which include the random coins used to sample pk and $\mathsf{ct}^\star$) and $\mathcal{E}_{i,j}$ output both a valid explanation for the statement.

Therefore, we argue that d_j is equivalently computed by performing the same homomorphic operations as $\mathcal{A}_2(\mathsf{ct}_1^\star, \ldots, \mathsf{ct}_s^\star, \mathsf{st}_2)$, but on the plaintexts $\mathsf{m}_1, \ldots, \mathsf{m}_s$ (even if some $\mathsf{ct}_i^\star$ encrypt a different message or are invalid). Clearly, this is true for the base case where the explanation provides a valid witness, due to perfect correctness of HE and the definition of $\mathcal{R}_{\mathsf{ny}}$. This is also true in the base case where the explanation refers to a simulated proof. In this case, we either have $d_j = \mathsf{copy}_i$ when the ciphertext was copied without modification or we insert the message m_i at this position, when recursively extracting a ciphertext. For the recursive case, we can argue that this property is preserved because of the perfect (homomorphic) correctness of HE and the definition of $\mathcal{T}_{\mathsf{ny}}$. □

The next distribution $\mathcal{G}_4$ is identical to $\mathcal{G}_{3,s}$, except that we simulate the $\textsc{Dec}(\cdot)$ oracle for $\mathcal{A}_1$ with sk_1 instead of sk_2.

Lemma 32. *There exists a PPT adversary $\mathcal{H}$ such that for all PPT extractors $\mathcal{E}$*

$$\left| \Pr_{z \xleftarrow{\$} \mathcal{G}_{3,s}} [\mathcal{D}(z) \Rightarrow 1] - \Pr_{z \xleftarrow{\$} \mathcal{G}_4} [\mathcal{D}(z) \Rightarrow 1] \right| \leq \mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{H},\mathcal{E},\Pi}(\lambda)$$

Proof. For this argument, we only need the proof system to be controlled-malleable simulation sound (c.f. Definition 11) without any simulated proof queries.

If the adversary $\mathcal{A}_1$ submits to its $\textsc{Dec}$ oracle a ciphertext $\mathsf{ct}' = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$ such that π verifies for the statement $x = (\mathsf{pk}_1, \mathsf{ct}_1, \mathsf{pk}_2, \mathsf{ct}_2)$, but $x \notin \mathcal{L}_{\mathcal{R}_{\mathsf{ny}}}$. Then we can use $\mathcal{A}_1$ to win the controlled-malleable simulation soundness game. If this even does not occur, the answer of $\textsc{Dec}$ are identical in both games due to perfect correctness of HE. □

The next distribution $\mathcal{G}_{5,i}$ for $i \in [0, s]$ is identical to $\mathcal{G}_4$, except that we introduce the same changes as with $\mathcal{G}_{3,i}$ but in the second instance of the underlying HE. More precisely, for $\hat{\imath} \in [s]$ we sample $\mathsf{m}_{\hat{\imath}}, \mathsf{m}'_{\hat{\imath}} \xleftarrow{\$} \mathcal{C}_{\mathcal{M}}$ and we compute for all for $\hat{\imath} \in [i]$ the $\hat{\imath}$-th challenge ciphertext $\mathsf{ct}_{\hat{\imath}}^\star = (\mathsf{ct}_{\hat{\imath},1}^\star, \mathsf{ct}_{\hat{\imath},2}^\star, \pi_{\hat{\imath}}^\star)$ as $\mathsf{ct}_{\hat{\imath},1}^\star \leftarrow \mathsf{Enc}(\mathsf{pk}_1, \mathsf{m}'_{\hat{\imath}})$ (unchanged), $\mathsf{ct}_{\hat{\imath},2}^\star \leftarrow \mathsf{Enc}(\mathsf{pk}_2, \mathsf{m}'_{\hat{\imath}})$ (unchanged)(in $\mathcal{G}_2$ this was done as $\mathsf{ct}_{\hat{\imath},2}^\star \leftarrow \mathsf{Enc}(\mathsf{pk}_2, \mathsf{m}_{\hat{\imath}})$), and $\pi_{\hat{\imath}}^\star$ via a call to $\textsc{Prove}$ (unchanged).

Lemma 33. *There exist PPT adversaries $\mathcal{B}$ and $\mathcal{H}$ such that for all PPT extractors $\mathcal{E}$ and for all $i \in [s]$*

$$\left| \Pr_{z \xleftarrow{\$} \mathcal{G}_{5,i-1}} [\mathcal{D}(z) \Rightarrow 1] - \Pr_{z \xleftarrow{\$} \mathcal{G}_{5,i}} [\mathcal{D}(z) \Rightarrow 1] \right| \leq \mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{B},\mathsf{HE}}(\lambda) + sq^2 \mathsf{Adv}^{\mathsf{cm\text{-}sse}}_{\mathcal{H},\mathcal{E},\mathcal{Z},\Pi}(\lambda).$$

The proof of the this lemma proceeds identical to Lemma 31, except for swapping the roles of the two instances of the underlying HE.

The game distribution $\mathcal{G}_{5,s}$ is identical to the simulators distribution. Thus by combing Lemmata 29–33 Theorem 16 follows. □

Gaps in [14]. The work [14] claim to prove a very similar result, but only for a bounded number of homomorphic operations and without achieving unlinkability (which we prove next) with a proof system that is similar to our counter-based construction. However, there proof contains several gaps:

1. They do not include the randomness that the first stage of the simulator used in the state for the second stage of the simulator. Without this randomness the second stage of the simulator cannot run the extractors.
2. In the TNM-CPA variant of their proof, the simulator uses an externally given public key instead of generating it itself. This way, the simulator does not have the random coins used to generate the public key and thus cannot run the extractors. The TNM-CCA1 security variant is not affected by this problem, since there the simulator generates the public key.
3. They perform reductions to the IND-CPA security of the underlying encryption scheme while using extractors without auxiliary inputs. In these steps, the reduction does not have access to the random coins used to generate the public key and the challenge ciphertext (and with these random coins, it could trivially break IND-CPA security), and without these random coins it cannot run the extractors. We solve this problem by assuming controlled-malleable simulation extractability in the common auxiliary input model for an auxiliary input that provides the public key and challenge ciphertext. We are not aware of any solutions to this problem that do not require stronger assumptions.
4. Their simulator (and several reductions) run polynomially many extractors that might have different polynomial runtime bounds. Since in general the sum of polynomially many polynomials is no longer a polynomial, their simulation is not guaranteed to run in polynomial time. A similar problem exists when summing up their failure probability, which is in general not negligible even when every single adversary has a negligible failure probability. We solve this problem by carefully constructing the extractors in such a way that they all have the *same* polynomial runtime bound and the *same* negligible function bounding their advantage.

Theorem 17 (Unlinkability). *If* HE *is unlinkable, and* Π *is zero-knowledge and relaxed derivation private, then* $\mathsf{HE}'[\mathsf{HE}, \Pi]$ *is unlinkable. More precisely, for every PPT adversary* $\mathcal{A}$ *there exists PPT adversaries* $\mathcal{B}$*,* $\mathcal{C}$ *and* $\mathcal{D}$ *with*

$$\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{A},\mathsf{HE}'[HE,\Pi]}(\lambda) \leq \mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{B},\mathsf{HE}}(\lambda) + \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{C},\Pi}(\lambda) + \frac{1}{2}\mathsf{Adv}^{\mathsf{rdp}}_{\mathcal{D},\Pi}(\lambda).$$

Proof. The proof proceeds with a hybrid argument. Let G_0 be the real unlinkability game. Let (S_1, S_2) be the zero-knowledge simulator for Π. The game G_1 is defined identically, except that the crs, which is part of the public key, is generated with $(\mathsf{crs}, \mathsf{td}) \leftarrow S_1(1^\lambda)$ (instead of $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$). Moreover, if $b = 1$, the proof in the ciphertext $\mathsf{ct}^\star = (\mathsf{ct}^\star_1, \mathsf{ct}^\star_2, \pi^\star)$ that is returned by the CHAL oracle is generated as $\pi^\star \leftarrow S_2(\mathsf{crs}, \mathsf{td}, (\mathsf{pk}_1, \mathsf{ct}^\star_1, \mathsf{pk}_2, \mathsf{ct}^\star_2))$ (instead of via Prove).

Lemma 34 ($\mathsf{G}_0 \rightsquigarrow \mathsf{G}_1$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1]\right| \leq \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{B},\Pi}(\lambda).$$

Proof. The reduction embeds the crs it gets as input in pk and generates everything else itself. If $b = 1$, the reduction generates the proof for ct' via its PROVE oracle.

If the zero-knowledge experiment is played with $b = 0$, the reduction simulates G_0 and otherwise G_1. □

The next game G_2 is identical to G_1, except that if $b = 0$ in the ciphertext $\mathsf{ct}^\star = (\mathsf{ct}_1^\star, \mathsf{ct}_2^\star, \pi^\star)$ that is used to answer the CHAL oracle query, the proof is generated as $\pi \leftarrow S_2(\mathsf{crs}, \mathsf{td}, (\mathsf{pk}_1, \mathsf{ct}_1^\star, \mathsf{pk}_2, \mathsf{ct}_2^\star))$ (instead of via ZKEval)

Lemma 35 ($\mathsf{G}_1 \rightsquigarrow \mathsf{G}_2$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{D}$ such that*

$$\left|\Pr[\mathsf{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1]\right| \leq \frac{1}{2}\mathsf{Adv}^{\mathsf{rdp}}_{\mathcal{D},\Pi}(\lambda).$$

Proof. If $b = 1$, the games are identical. If $b = 0$, which happens with probability $1/2$, the reduction computes everything as in G_1, except that it uses the crs it got as input for pk and generates the proof and in the CHAL query for input $(\mathsf{ct}_1, \ldots, \mathsf{ct}_n, C)$ with $\mathsf{ct}_i = (\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}, \pi_i)$ it computes the proof $\pi^\star$ for the challenge ciphertext $\mathsf{ct}^\star$ by submitting $(x_i, \pi_i)_{1 \leq i \leq n}$ with $x_i = (\mathsf{pk}_1, \mathsf{ct}_{i,1}, \mathsf{pk}_2, \mathsf{ct}_{i,2})$ and a transformation $T = (T_x, T_w)$ describing the homomorphic evaluation of C to the relaxed derivation privacy game and using the resulting proof. □

The next hybrid G_3 is identical to G_2 except that, if $b = 0$, in the challenge ciphertext $\mathsf{ct}^\star = (\mathsf{ct}_1^\star, \mathsf{ct}_2^\star, \pi^\star)$ the component $\mathsf{ct}_1^\star$ is generated as a fresh encryption $\mathsf{ct}_1^\star \leftarrow \mathsf{Enc}(\mathsf{pk}_1, f_C(\mathsf{m}_1, \ldots, \mathsf{m}_n))$ (instead of being computed via Eval).

Lemma 36 ($\mathsf{G}_2 \rightsquigarrow \mathsf{G}_3$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_2^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1]\right| \leq \frac{1}{2}\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{B},\mathsf{HE}}(\lambda).$$

Proof. If $b = 1$, the games are identical. If $b = 0$, which happens with probability $1/2$, the reduction uses its input public key as pk_1 and computes in the CHAL query for input $(\mathsf{ct}_1, \ldots, \mathsf{ct}_n, C)$ with $\mathsf{ct}_i = (\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}, \pi_i)$ the first component $\mathsf{ct}_1^\star$ of the challenge ciphertext by making a CHAL on $(\mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{n,1}, C)$. If the relaxed derivation privacy game for HE is played with $b = 0$, the reduction simulates G_2 and otherwise G_3. □

The final hybrid G_4 is identical to G_3 except that, if $b = 0$, in the challenge ciphertext $\mathsf{ct}^\star = (\mathsf{ct}_1^\star, \mathsf{ct}_2^\star, \pi^\star)$ the component $\mathsf{ct}_2^\star$ is generated as a fresh encryption $\mathsf{ct}_2^\star \leftarrow \mathsf{Enc}(\mathsf{pk}_2, f_C(\mathsf{m}_1, \ldots, \mathsf{m}_n))$ (instead of being computed via Eval).

Lemma 37 ($\mathsf{G}_3 \rightsquigarrow \mathsf{G}_4$)**.** *For every PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{B}$ such that*

$$\left|\Pr[\mathsf{G}_3^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1]\right| \leq \frac{1}{2}\mathsf{Adv}^{\mathsf{unlink}}_{\mathcal{B},\mathsf{HE}}(\lambda).$$

The proof of this Lemma proceeds identical to Lemma 36, except for swapping the roles of the two instances of the underlying HE.

Lemma 38 (G_4).

$$\Pr[\mathsf{G}_4^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

Proof. In G_4, the adversaries view is statistically independent of the challenge bit b, because the CHAL oracle always generates $\mathsf{ct}_1^\star$ and $\mathsf{ct}_2^\star$ as fresh encryptions and $\pi^\star$ via the zero-knowledge simulator. □

Combining Lemmata 34—38 yields Theorem 17. □

F Separating AOWFs from falsifiable assumptions

We recall the notion of a falsifiable assumption. Falsifiable assumptions where first defined in [58] to classify cryptographic assumptions. We use the (more inclusive) definition of [42], that they used to separate SNARGs from falsifiable assumptions.

Definition 34 (Falsifiable assumption). *A falsifiable assumption $(\mathcal{C}, c)$ consist of an interactive PPT algorithm, the challenger $\mathcal{C}$, and a threshold $c \in [0,1)$. The challenger inputs 1^λ, interacts with an interactive PPT adversary $\mathcal{A}(1^\lambda)$ and outputs a bit $b \in \{0,1\}$, indicating whether the adversary has won the game. We use $\langle \mathcal{C}(1^\lambda) \rightleftarrows \mathcal{A}(1^\lambda) \rangle$ to denote the random variable representing the output of $\mathcal{C}(1^\lambda)$ when interacting with $\mathcal{A}(1^\lambda)$.*

The assumption $(\mathcal{C}, c)$ holds if for all PPT adversaries $\mathcal{A}$ we have

$$\Pr[\langle \mathcal{C}(1^\lambda) \rightleftarrows \mathcal{A}(1^\lambda) \rangle = 1] \leq c + \mathsf{negl}(\lambda)$$

for a negligible function negl, where the probability is taken over the random coins of $\mathcal{C}$ and $\mathcal{A}$.

We consider the following black-box variant for two-stage adversaries, where the reduction has no access to to the state st of the two adversaries. Whenever the reduction calls the second stage, it has to provide an index of the first stage run that is supposed to be used. The reduction also gets access to the exact runtime of the first stage, to be able to simulate the security game for adversarial one-way functions. For the second stage, there is no run-time restriction.

Definition 35 (Black-box reduction). *Let $(\mathcal{A}_1, \mathcal{A}_2)$ be an adversary, where $\mathcal{A}_1$ is PPT and $\mathcal{A}_2$ can be unbounded, for the security game for $\mathsf{AOWFGen}$. With this, we define the following oracles:*

- *$\mathcal{O}_1(\cdot,\cdot)$ inputs 1^λ and pp, runs $(y, \mathsf{st}) \xleftarrow{t} \mathcal{A}_1(1^\lambda, \mathsf{pp})$, outputs y and the runtime t and stores on its i-th call $y_i := y$ and $\mathsf{st}_i := \mathsf{st}$.*
- *$\mathcal{O}_2(\cdot)$ inputs an index $i \in [n]$ where n is the number of calls to the oracle $\mathcal{A}_1$ so far and outputs $x \leftarrow \mathcal{A}_2(y_i, \mathsf{st}_i)$.*

A black-box reduction for an adversarial one-way function generator AOWFGen *to a falsifiable assumption* $(\mathcal{C}, c)$ *is an algorithm* $\mathcal{R}$ *with access to the oracles* $\mathcal{O}_1(\cdot, \cdot)$ *and* $\mathcal{O}_2(\cdot)$ *such that whenever* $(\mathcal{A}_1, \mathcal{A}_2)$ *wins the security game for* AOWFGen*, then* $\mathcal{R}^{\mathcal{O}_1(\cdot,\cdot),\mathcal{O}_2(\cdot)}$ *wins the game* $\mathcal{C}$ *with probability* $c + \varepsilon(\lambda)$ *where* $\varepsilon(\lambda)$ *is non-negligible.*

We show that our impossibility results holds even if we restrict ourselves to AOWF for arity 1. Moreover, since we do not require unlinkability, we omit the random input. In this simplified setting, a bad but valid explanation can be described by x with $f_{\mathsf{pp}}^{p(t)}(x) = y$, because the graph obtained by applying $f_p p$ $p(t)$ to x will either contain $p(t)$ nodes with distinct labels or two nodes with the same label, which can be turned into an explanation graph with a cycle.

Theorem 18. *If* AOWFGen *is an adversarial one-way function generator and assume for every sufficiently large polynomial* p *there exists distributions* $\mathcal{C}_{p,0}(\mathsf{pp})$ *and* $\mathcal{C}_{p,1}(\mathsf{pp})$ *on* $\mathcal{X}_{\mathsf{pp}}$ *such that we can sample from* $\mathcal{C}_{p,1}(\mathsf{pp})$ *in time* t *and*

$$\mathcal{C}_{p,1}(\mathsf{pp}) \approx_c f^{p(t)}(\mathcal{C}_{p,0}(\mathsf{pp})).$$

Then the security of AOWFGen *cannot be black-box reduced (as defined in Definition 35) to any non-false falsifiable assumption* $(\mathcal{C}, c)$*.*

Proof. Consider the following adversary, where the second stage is inefficient and that wins the security game for AOWFGen with probability 1:

- $\mathcal{A}_1(1^\lambda, \mathsf{pp})$ returns $(y, \mathsf{st} = \varepsilon)$ where $y \leftarrow \mathcal{C}_{p,1}(\mathsf{pp})$
- $\mathcal{A}_2(y, \mathsf{st} = \varepsilon)$ samples $x \leftarrow \mathcal{C}_{p,0}(\mathsf{pp})$ until $f_{\mathsf{pp}}^{p(t)}(x) = y$, where t is the runtime of $\mathcal{A}_1$. If such a sample is found, it returns x.

If $\mathcal{R}^{\mathcal{O}_1(\cdot,\cdot),\mathcal{O}_2(\cdot)}$ is a black-box reduction for AOWFGen, where $\mathcal{O}_1$ and $\mathcal{O}_2$ are defined as in Definition 35 for the adversary here, $\mathcal{R}^{\mathcal{O}_1(\cdot,\cdot),\mathcal{O}_2(\cdot)}$ wins the game $\mathcal{C}$ with probability $c + \varepsilon(\lambda)$ for non-negligible $\varepsilon(\lambda)$.

Next, we give an efficient simulation of this adversary:

- $\mathcal{A}'_1(1^\lambda, \mathsf{pp})$ samples $x \leftarrow \mathcal{C}_{p,0}(\mathsf{pp})$, computes $y := f^{p(t)}(x)$ (where t is the runtime of $\mathcal{A}_1$) and returns $(y, \mathsf{st} = x)$.
- $\mathcal{A}'_2(y, \mathsf{st} = x)$ returns x.

Now let $\mathcal{O}'_1$ and $\mathcal{O}'_2$ be defined as in Definition 35 for the adversary $(\mathcal{A}'_1, \mathcal{A}'_2)$, except that $\mathcal{O}'_1$ still returns the run-time of $\mathcal{A}_1(1^\lambda, \mathsf{pp})$. The behavior of the oracles $(\mathcal{O}'_1, \mathcal{O}'_2)$ and $(\mathcal{O}'_1, \mathcal{O}'_2)$ is computationally indistinguishable, because

- the distribution of y is computationally indistinguishable by assumption
- the distribution of x is in both cases a sample from $\mathcal{C}_{p,0}(\mathsf{pp})$ conditioned on $f^{p(t)}(x) = y$ and thus identical.

Thus, $\mathcal{R}^{\mathcal{O}'_1(\cdot,\cdot),\mathcal{O}'_2(\cdot)}$ wins the game $\mathcal{C}$ with probability $c + \varepsilon'(\lambda)$ for non-negligible $\varepsilon'(\lambda)$. However, since the oracles $\mathcal{O}'_1$ and $\mathcal{O}'_2$ can be efficiently computed, we also have a PPT adversary that wins $\mathcal{C}$ with this advantage. Thus the assumption $(\mathcal{C}, c)$ is false. □

Note that the additional requirement we pose on the adversarial one-way function is very mild. For example, if f_{pp} is a permutation on a set $S \subseteq \mathcal{X}_{\mathsf{pp}}$ and we can efficiently draw uniform samples from S, it is satisfied.