Triply Adaptive UC NIZK

Ran Canetti[1⋆], Pratik Sarkar[1⋆⋆], and Xiao Wang[2⋆⋆⋆]

[1] Boston University
[2] Northwestern University

Abstract. Non-interactive zero knowledge (NIZK) enables proving the validity of NP statement without leaking anything else. We study multi-instance NIZKs in the common reference string (CRS) model, against an adversary that adaptively corrupts parties and chooses statements to be proven. We construct the first such *triply adaptive* NIZK that provides full adaptive soundness, as well as adaptive zero-knowledge, assuming either LWE or else LPN and DDH (previous constructions rely on non-falsifiable knowledge assumptions). In addition, our NIZKs are universally composable (UC). Along the way, we:

- Formulate an ideal functionality, $\mathcal{F}_{\mathsf{NICOM}}$, which essentially captures *non-interactive* commitments, and show that it is realizable by existing protocols using standard assumptions.
- Define and realize, under standard assumptions, Sigma protocols which satisfy triply adaptive security with access to $\mathcal{F}_{\mathsf{NICOM}}$.
- Use the Fiat-Shamir transform, instantiated with correlation intractable hash functions, to compile a Sigma protocol with triply adaptive security with access to $\mathcal{F}_{\mathsf{NICOM}}$ into a triply adaptive UC-NIZK argument in the CRS model with access to $\mathcal{F}_{\mathsf{NICOM}}$, assuming LWE (or else LPN and DDH).
- Use the UC theorem to obtain UC-NIZK in the CRS model.

1 Introduction

Non-Interactive zero knowledge (NIZK) [BFM90, BSMP91] is a magical primitive: with the help of a trusted reference string, it allows parties to publicly assert knowledge of sensitive data and prove statements regarding the data while keeping the data itself secret. Proofs are written once and for all, to be inspected and verified by anyone at any time.

However, harnessing this magic in a concrete and realizable set of security requirements has turned out to be non trivial. A first thrust provides basic formulations of soundness and zero knowledge in the presence of a reference string, and constructions that satisfy them under standard assumptions

⋆ Supported by NSF Awards 1931714, 1801564, 1414119, and by DARPA under Agreement No. HR00112020023.
⋆⋆ Supported by NSF Awards 1931714, 1414119, and the DARPA SIEVE program.
⋆⋆⋆ Supported by DARPA under Contract No. HR001120C0087, NSF award #2016240, and research awards from Facebook and Google.

[BSMP91, FLS99, GR13]. Indeed, even these basic requirements turn out to be non-trivial to formulate and obtain, especially in the case of multiple proofs that use the same reference string and where the inputs and witnesses are chosen adversarially in an adaptive way.

A second thrust addresses malleability attacks [SCO+01, DDN91], and more generally universally composable (UC) security [CLOS02] in a multi-party setting. In particular, UC NIZK has been used as a mainstay for incorporating NIZK proofs in cryptographic protocols and systems - actively secure MPC [GMW87], CCA secure encryption [NY90, DDN91], signatures [BMW03, BKM06] and cryptocurrencies [BCG+14].

A third thrust is to construct NIZK protocols that are secure in a multi-party setting where the adversary can corrupt parties adaptively [CLOS02, CSW20a, AMPS21, CGPS21] as the computation proceeds. Here the traditional definition (which requires that the attacker does not gain any advantage towards breaking the security of the overall system beyond the ideal case where the NIZK is replaced by a trusted party) is extended to the case where the attacker obtains the hidden internal state of some provers *after* the proof was sent. Indeed, this extended guarantee is essential whenever NIZK is used as a primitive within larger protocols that purport to obtain security against adaptive corruptions[3].

The first protocol that provides security against adaptive corruptions is that of Groth, Ostrovsky, and Sahai [GOS06, GOS12] (GOS). That protocol is also UC secure, even in a multi-proof, multi-party setting. However it only guarantees *culpable soundness*, namely that the sequence of instances proven to be in a language L during an execution of the protocol is indistinguishable (given the reference string) from a sequence of instances that are actually in L. The works of [KNYY19, KNYY20] have similar characteristics: they provide security against adaptive corruptions, but only culpable adaptive soundness.

Abe and Fehr [AF07] show how to prove full adaptive soundness of a variant of the GOS protocol, under a knowledge-of-exponent (KOE) assumption[4]. However, their analysis is incompatible with UC security [KZM+15], since KOE-style assumptions require existence of a knowledge extractor that has full access to, and whose code is larger than the code of the environment, In contrast, in the UC framework a single extractor/simulator would have to handle arbitrary poly-time environments. The recent work of [KKK21] investigated composable security for knowledge assumptions in the generic group model. They rule out general composition but demonstrate that it is possible under restricted settings. We refer to their paper for more details. Proving composable security of [AF07] in their model is still an open question.

[3] In cases where the prover is able to immediately erase all records of its sensitive state - specifically the witness and randomness used in generating the proof - adaptive security is easy to obtain. However such immediate and complete erasure of local state is not always practical.

[4] [AF07] provides adaptive soundness and adaptive zero knowledge and claims security against adaptive corruptions in Remark 11 of their paper.

We are thus left with the following natural question: Can we have triply adaptive NIZK protocols, namely full-fledged UC NIZK protocols in the multi-party, multi-proof setting, in the case of adaptive corruptions without erasures, and with full adaptive soundness? And if so, under what assumptions?

1.1 Our Contributions

We develop a general methodology for obtaining triply adaptive NIZKs, namely UC NIZKs with full adaptive soundness, withstanding adaptive corruptions with no erasures. Using this methodology, we obtain triply adaptive NIZK protocols from statically secure Sigma protocols. The NIZK protocols reuse a single crs for multiple NIZK instances between different pairs of parties. Moreover, one of the NIZK protocols also avoids expensive Karp reductions. Upon concrete instantiation based on either Learning With Errors (LWE), or Decisional Diffie Hellman (DDH) plus Learning Parity with Noise (LPN) assumption, we obtain the following result:

Theorem 1. *(Informal) Assuming either 1) LWE assumption holds or 2) both DDH and LPN assumptions hold, there exists a multi-theorem NIZK protocol that UC-securely implements the NIZK functionality(Fig. 2) against adaptive corruptions in the crs model for multiple instances. Furthermore, it is adaptively sound and adaptively zero knowledge.*

As an independent result we also obtain a compiler that (assuming either LWE or DDH) transforms a given NIZK protocol, where the length of the crs can depend on the NP relation to be asserted, to a NIZK protocol where the length of the crs depends only on the security parameter. Furthermore, we do so while preserving triple adaptive security. Previous such compilers [GGI+15, CsW19] were known only from LWE:

Theorem 2. *(Informal) Assuming either 1) LWE assumption holds or 2) both DDH and LPN assumptions hold, there exists a multi-theorem NIZK protocol that UC-securely implements the NIZK functionality(Fig. 2) against adaptive corruptions with short crs (i.e. $|crs| = poly(\kappa)$ and κ is the computational security parameter) for multiple instances. Furthermore, it is adaptively sound and adaptively zero knowledge.*

Furthermore, by plugging our NIZK protocol in the compiler of [CsW19] we can obtain a triply adaptive NIZK protocol from LWE, where the reference string size depends only on the security parameter and the proof size depends on the witness size and the security parameter.

1.2 Our Techniques

Our approach follows the general paradigm of applying the Fiat-Shamir transform (instantiated via correlation in tractable hash functions) to Sigma protocols, as developed in [CGH98, HL18, CCH+19, PS19, BKM20, HLR21]. However, to preserve triple adaptivity the transform should be applied with some care.

Starting Point. Let us briefly recall the definition of a Sigma protocol: A Sigma protocol is a 3 round protocol for proving validity of an NP statement $x \in \mathcal{L}$ (where $\mathcal{L}$ is the language) using the knowledge of an accepting witness w. The prover sends the first message a, the verifier samples a random challenge c, and based on the challenge $c \in \mathcal{C}$ the prover computes the response z. The verifier accepts an honest proof when $x \in \mathcal{L}$. Soundness ensures that the verifier rejects cheating proofs with $\frac{1}{|\mathcal{C}|}$ probability. Honest verifier zero knowledge (HVZK) ensures that the simulator constructs an honest proof given a random challenge c and the simulated proof is indistinguishable from an honest proof. However, the usual Sigma protocols [FLS99, Blu86] are only secure against static corruption of prover, i.e. upon post-execution corruption of prover the HVZK simulator obtains witness w and is unable to provide randomness such that it is consistent with the proof (a, c, z) constructed by the HVZK simulator.

New UC-Commitment Functionality $\mathcal{F}_{NICOM}$. To solve the above issue in a modular fashion, we first introduce a new non-interactive UC commitment functionality, $\mathcal{F}_{\mathsf{NICOM}}$, that enables modular analyzing of NIZK protocols that use commitments as an underlying primitive. Specifically, $\mathcal{F}_{\mathsf{NICOM}}$ returns a commitment string and a decommitment to the committer as an output of the commit phase, where the committer commits to a message. The open phase allows non-interactive verification of the commitment, decommitment and message tuple by a verifier. Moreover, the functionality is provided with an explicit simulation algorithm $\mathcal{S}_C$ which extracts committed messages from maliciously generated commitments and permits equivocation of simulated commitments. Looking ahead, the CI-hash function would be equipped with the $\mathcal{S}_C$ algorithm to run the bad challenge function and yet we would argue security of the NIZK protocol in the $\mathcal{F}_{\mathsf{NICOM}}$ model. Hence, $\mathcal{F}_{\mathsf{NICOM}}$ provides a cleaner abstraction of non-interactive UC commitments. The formal description of the $\mathcal{F}_{\mathsf{NICOM}}$ functionality can be found in Fig. 1. We also show that the [CF01] protocol satisfies $\mathcal{F}_{\mathsf{NICOM}}$.

Strengthening Sigma protocols in $\mathcal{F}_{NICOM}$ model. Now, we define the notion of an *adaptively secure Sigma protocol* in the $\mathcal{F}_{\mathsf{NICOM}}$ model as a stepping stone towards security against adaptive corruptions. These are Sigma protocols which provide security against adaptive corruption of prover in the $\mathcal{F}_{\mathsf{NICOM}}$ model. To attain constructions of such Sigma protocols, we replace the underlying commitment scheme in the *commit-and-open* protocols of [Blu86, FLS99, HV16] with $\mathcal{F}_{\mathsf{NICOM}}$. Then we prove that these Sigma protocols are adaptively secure in the $\mathcal{F}_{\mathsf{NICOM}}$ model, while preserving special soundness. Furthermore, these protocols satisfies full adaptive soundness and provides adaptive ZK in the $\mathcal{F}_{\mathsf{NICOM}}$ model. If $\mathcal{F}_{\mathsf{NICOM}}$ is concretely instantiated using an adaptively secure non-interactive commitment in non-programmable crs model [5] then the protocol also preserves full adaptive soundness and adaptive ZK.

[5] The crs distribution in the real world is statistically close to the crs distribution in the ideal world

Removing Interaction. It is now tempting to apply the Fiat-Shamir (FS) transform [FS87] using correlation intractable (CI) hash functions [CGH98, HL18, CCH+19, PS19, BKM20, HLR21], and conclude that the resulting protocol is a NIZK. However, it is not clear how the transform would actually work: the bad challenge function for the CI hash function cannot be defined given blackbox access to a and the challenge space can be exponentially large, for example consider Schnorr's protocol [Sch90]. The current CI-based NIZKs [CGH98, HL18, CCH+19, PS19] consider specific Sigma protocols to construct NIZKs. We take a different route to solve this problem by relying on special soundness property. Special soundness property of a Sigma protocol ensures that given two accepting transcripts (a, c_0, z_0) and (a, c_1, z_1) for different challenges $c_0 \neq c_1$ there exists an extractor which extracts a valid witness from the transcripts. If the statement $x \notin \mathcal{L}$ is not in the language then the prover cannot construct two such accepting transcripts for the same a.

We generalize the framework of [CD00] to construct our compiler. In our compiler, the prover computes a, samples two challenges c_0 and c_1, computes responses z_0 and z_1 and commits to (c_0, z_0) and (c_1, z_1) in $\mathcal{F}_{\mathsf{NICOM}}$ model. This step is repeated for $\tau = \mathcal{O}(\kappa)$ times, where κ is the security parameter. Let $\mathbf{Y}$ denote the commitments to (c_0, c_1, z_0, z_1) for the τ iterations. The CI-hash function is defined in the statistical mode equipped with the extraction algorithm $\mathcal{S}_C$ for $\mathcal{F}_{\mathsf{NICOM}}$. The hash function is CI for the bad challenge function - for each iteration (a, c_0, c_1, z_0, z_1) it outputs 0 if (a, c_0, z_0) is accepting. The prover invokes the CI hash function on $(a, \mathbf{Y})$ to obtain a challenge bit e for each iteration. For each iteration, the prover computes the response as the decommitment to (c_0, c_1, z_e). Special soundness of the Sigma protocol ensures that a malicious prover is unable to compute two such valid transcripts (a, c_0, z_0) and (a, c_1, z_1) for a false statement $x \notin \mathcal{L}$.

CI-based NIZK Transformations for Arguments. Now we would like to apply the analysis of [CCH+19] to argue soundness of the NIZK protocol, which says that if the malicious prover is able to construct an accepting proof for $x \notin \mathcal{L}$ then it breaks correlation intractability. However, now we are faced with another barrier: The [CCH+19] analysis for CI crucially needs the underlying Sigma protocol to be statistically sound. In contrast, our Sigma protocols are only computationally sound since it relies on the special soundness property (which can be computational) of the Sigma protocol and the computational binding property of the commitment scheme. Furthermore, this is inherent: Statistically sound ZK protocols cannot possibly be secure against adaptive corruptions. In particular, this means that we cannot "switch the crs in the hybrids to make the sigma protocol statistically sound": As soon as we do so, the protocol (in that hybrid) stops being secure against adaptive corruptions.

We get around this barrier[6] as follows: with each commitment made during the interaction we can associate an event B, determined at the time of com-

[6] The recent work of [CJJ21] also applied the Fiat-Shamir paradigm on an interactive protocol which is not statistically sound using CI hash functions. However, their

mitment, such that: (a) event B can be shown to occur only with negligible probability, and (b) conditioned on event B not occurring, the commitment is statistically binding. Event B is the event where the adversary successfully evades the extraction algorithm $\mathcal{S}_C$ of $\mathcal{F}_{\mathsf{NICOM}}$ and yet the corresponding decommitment is accepted. Given that event B does not occur, we then associate an event D with each of the τ adaptively secure Sigma protocol executions, such that: (a) event D can be shown to occur only with negligible probability, and (b) conditioned on event D not occurring, the Sigma protocol is statistically sound. The event D is the event where the adversary breaks special soundness property of the Sigma protocol. The [CCH+19] analysis can now be resurrected, conditioned on event B not occurring for any of the commitments made, and event D not occurring for the Sigma protocols. Initializing the hash function in the statistical mode ensures that soundness of the protocol is reduced to breaking statistical correlation intractability of the hash function, provided event B and event D does not occur.

Adaptive soundness of our protocol follows in a straightforward way from the fact that the entire proof is performed without changing the distribution of the crs in the $\mathcal{F}_{\mathsf{NICOM}}$ model. (Indeed, this important feature allows us to avoid the main obstacle that prevents the [GOS12] protocol from being adaptively sound.) Adaptive zero knowledge follows from the adaptive security of the Sigma protocol in the $\mathcal{F}_{\mathsf{NICOM}}$ model. If $\mathcal{F}_{\mathsf{NICOM}}$ is concretely instantiated using an adaptively secure non-interactive commitment in non-programmable crs model [7] then the protocol also preserves full adaptive soundness and adaptive ZK.

Instantiations of Adaptively Secure Sigma protocols in $\mathcal{F}_{\mathsf{NICOM}}$ model. We demonstrate that a wide variety of Sigma protocols satisfy (in $\mathcal{F}_{\mathsf{NICOM}}$ model) adaptive security with special soundness and adaptive soundness - Schnorr's protocol, Sigma protocol of [FLS99] (FLS), Blum's Hamiltonicity protocol and garbled circuit (GC) based protocol of [HV16]. Furthermore, the GC based protocol avoids expensive Karp reduction.

Instantiating the CI-hash and $\mathcal{F}_{\mathsf{NICOM}}$. The CI function can be instantiated from LWE [PS19], or it can be replaced by a CI-Approx [BKM20] function based on LPN+DDH. $\mathcal{F}_{\mathsf{NICOM}}$ is instantiated using the protocol of [CF01] from equivocal commitments and CCA-2 secure public key encryption with oblivious ciphertext sampling property in the non-programmable crs model.

Reducing crs size. By applying techniques from GOS, we obtain a compiler which *reduces the crs size* of a NIZK argument. Assuming reusable non-interactive equivocal commitments with additive homomorphism and PKE (with oblivious ciphertext sampleability) we compile any triply adaptive NIZK argument with

protocol is not adaptively sound. Meanwhile, the plain-model sigma protocol that [CCH+19] start from is statistically sound.)

[7] The crs distribution in the real world is statistically close to the crs distribution in the ideal world

a long multi-proof crs, i.e. $|\mathsf{crs}| = \text{poly}(\kappa, |\mathcal{C}|)$ to obtain a triply adaptive NIZK argument with a short multi-proof common reference string scrs, where $|\mathsf{scrs}| = \text{poly}(\kappa)$, $\mathcal{C}$ is the NP verification circuit and κ is the computational security parameter. The prover commits to each wire value (of the circuit) and proves that they are bit commitments using the NIZK. In addition, the prover applies some homomorphic operation on the input wire and output wire commitments for each gate. If the input and output wire values are consistent with the gate evaluation then the homomorphically evaluated commitment will be a bit commitment. The prover proves this using NIZK for every gate in the circuit. Each NIZK statement is short and depends only on the committer's algorithm ($= \text{poly}(\kappa)$) and not on $|\mathcal{C}|$. As a result the crs size of the NIZK can be short. The commitment can be instantiated from DDH (Pedersen commitment or [CSW20b]) or LWE/SIS [GVW15]. The encryption scheme can be instantiated from DDH assumption using Elgamal encryption or LWE [GSW13] assumption.

Obtaining multi-session UC security. We add non-malleability to our NIZK argument using standard techniques from GOS to obtain the multi-session UC-secure NIZK in the short crs model. It relies on a tag-based simulation-sound trapdoor commitment scheme and a strong one-time signature scheme . The tag-based commitment can be instantiated from UC-commitments - DDH [CSW20b] and LWE [CsW19]. Strong one-time signatures can be constructed from one-way functions. This transformation also preserves triply adaptive security.

1.3 Related Work

The works of [GOS06, KNYY19, KNYY20] construct NIZKs which are secure against adaptive corruptions but they lack adaptive soundness. The works of [CCH+19, BKM20] construct statically secure NIZKs which attain adaptive soundness and adaptive ZK. A concurrent work by [CPV20] compiled delayed input Sigma protocol into a Sigma protocol which satisfies adaptive zero knowledge. Upon applying the result of [CPS+16] they obtain adaptive soundness. The Fiat-Shamir transform is applied using CI hash function to obtain NIZKs, but they lack security against adaptive corruptions. The only work which achieves triple adaptive security is [AF07] based on knowledge assumptions; which is incompatible with the UC framework.

The literature consists of work [GGI+15, CsW19] that make the crs size independent of $|\mathcal{C}|$ but those approaches are instantiatable only from LWE. Whereas, our compiler can be instantiated from non-lattice based assumptions like DDH.

Paper Organization. In Section 2, we present the key intuitions behind our protocols. We introduce some notations and important concepts used in this paper in Section 3. This is followed by our triply adaptively-secure NIZK compiler in Section 4. We present our compiler to reduce the crs length in Section 5. Finally, we conclude with our multi-session UC-NIZK protocol in the short crs model in Section 6. Throughout the paper we refer to *security against adaptive corruptions* as *adaptive security.*

2 Technical Overview

In this section we provide an overview of our protocols. As discussed in the Introduction, a key component in our approach is to break the Fiat-Shamir transformation into two steps: A first step that uses an ideal UC commitment fucntionality, and a second step of instantiating this functionality with an adaptively secure protocol. Validity of the approach would follow from the UC theorem and the special soundness of the sigma protocol.

We first overview the new formulation of ideal UC commitments, $\mathcal{F}_{\mathsf{NICOM}}$, that enables our two-step approach, and argue that known protocols, that UC realize the the traditional ([CF01]) formulation of ideal commitment, realize $\mathcal{F}_{\mathsf{NICOM}}$ as well. Next, we overview our notion of fully adaptive Sigma protocols that use $\mathcal{F}_{\mathsf{NICOM}}$, followed by the first step of the Fiat-Shamir transform. We demonstrate that the resulting NIZKs satisfy triply adaptive security in the $\mathcal{F}_{\mathsf{NICOM}}$-hybrid model, and that triple adaptivity is preserved even after replacing $\mathcal{F}_{\mathsf{NICOM}}$ with a protocol that realizes it. Next, we show instantiations of adaptive Sigma protocol. Finally we show how to reduce the crs size of our NIZK protocols to poly(κ) by assuming homomorphic equivocal commitments. Till this point, all our protocols are triply adaptive and single-prover UC-secure. Finally, we make them UC-secure in the general, multi-prover sense by adding non-malleability.

2.1 Formalizing UC non-interactive commitment

Our new UC-commitment functionality $\mathcal{F}_{\mathsf{NICOM}}$ can be found in Fig. 1. The functionality receives an algorithm $\mathcal{S}_C$ algorithm from the adversary $\mathcal{S}$. When an honest committer P wants to commit to a message m for subsession ssid, the functionality invokes $\mathcal{S}_C$ for a commitment string π and an internal state st. π is independent of the message m. The functionality then invokes $\mathcal{S}_C$ with the message m and the state st to obtain a decommitment d and an updated state st. The functionality stores $(\mathsf{ssid}, \mathsf{P}, m, \pi, d, \mathsf{st})$ and returns the commitment string π and the decommitment d to the committer. The committer sends π as the commitment to message m. An honest committer decommits to a commitment string π' by sending (m', d') to the verifier V. The verifier locally verifies the decommitment by invoking $\mathcal{F}_{\mathsf{NICOM}}$ on the tuple (m', π', d'). The functionality returns $\mathsf{verified}$ if the tuple is stored in memory corresponding to the subsession and the same committer P. If the same commitment string π' is stored but with different messages/decommitments/committers/ssid then the functionality rejects the opening by sending $\mathsf{verification\text{-}failed}$. Finally, if the commitment string has never been stored in the memory of $\mathcal{F}_{\mathsf{NICOM}}$ then $\mathcal{F}_{\mathsf{NICOM}}$ invokes $\mathcal{S}_C$ to extract a valid message m'' from the commitment string π'. If $m'' == m'$ then the functionality invokes $\mathcal{S}_C$ with the opening (m', π', d') to verify the decommitment. If the decommitment correctly verifies then the functionality stores the tuple in the memory and returns $\mathsf{verified}$ to V. Else, it rejects the decommitment.

Our model allows a prover to send a commitment that was not computed by invoking the $\mathcal{F}_{\mathsf{NICOM}}$ functionality. Furthermore, access to the $\mathcal{S}_C$ algorithm enables extraction from a maliciously generated commitment and equivocating

Fig. 1. Non-Interactive UC-Commitment Functionality $\mathcal{F}_{\mathsf{NICOM}}$

- At first activation, obtain algorithm $\mathcal{S}_C$ from $\mathcal{S}$.
- **Commit:** On input $(\mathsf{Com}, \mathsf{ssid}, \mathsf{P}, m)$ from committer P:
  - obtain commitment π and internal state st as $(\pi, \mathsf{st}) \leftarrow \mathcal{S}_C(\mathsf{Com}, \mathsf{ssid}, \mathsf{P})$
  - obtain decommitment d and state st as $(d, \mathsf{st}) \leftarrow \mathcal{S}_C(\mathsf{Equiv}, \mathsf{ssid}, \mathsf{P}, \pi, \mathsf{st}, m)$
  - store $(\mathsf{ssid}, \mathsf{P}, m, \pi, d, \mathsf{st})$ and output $(\mathsf{Receipt}, \mathsf{ssid}, \mathsf{P}, \pi, d)$ to P.

  Ignore future $(\mathsf{Com}, \mathsf{ssid}, \cdot)$ inputs with the same ssid.
- **Open:** On input $(\mathsf{Open}, \mathsf{ssid}, \mathsf{P}, m', \pi', d')$ from verifier V:
  - If $(\mathsf{ssid}, \mathsf{P}, m', \pi', d', \mathsf{st})$ is stored for some st, then return $(\mathsf{verified}, \mathsf{ssid}, \mathsf{P})$ to V.
  - If $(\mathsf{ssid}, \mathsf{P}, m'', \pi', d'', \mathsf{st})$ is stored, and $m'' \neq m'$ or $d'' \neq d'$ then return $(\mathsf{verification\text{-}failed}, \mathsf{ssid}, \mathsf{P})$ to V.
  - Else (i.e., no record $(\mathsf{ssid}, ...)$ is stored, or there is a stored record of the form $(\mathsf{ssid}, \mathsf{P}, m'', \pi'', d'', \mathsf{st})$ where $\pi'' \neq \pi'$):
    - Obtain $(m'', \mathsf{st}) \leftarrow \mathcal{S}_C(\mathsf{Ext}, \mathsf{ssid}, \mathsf{P}, \pi')$.
    - If $m'' \neq m'$, set $v = \mathsf{verification\text{-}failed}$.
    - If $m'' = m'$, set $v \leftarrow \mathcal{S}_C(\mathsf{Verify}, \mathsf{ssid}, \mathsf{P}, \pi', d', \mathsf{st})$.
    - If $v == \mathsf{verified}$, then store the tuple $(\mathsf{ssid}, \mathsf{P}, m', \pi', d', \mathsf{st})$ and return $(v, \mathsf{ssid}, \mathsf{P})$ to V. Else return $(\mathsf{verification\text{-}failed}, \mathsf{ssid}, \mathsf{P})$ to V.
- **Corruption:** When receiving $(\mathsf{Corrupt}, \mathsf{ssid})$ from $\mathcal{S}$, mark ssid as corrupted. Send all the stored tuples of the form $(\mathsf{ssid}, \ldots)$ to $\mathcal{S}$. If there does not exist any tuple then send $(\mathsf{ssid}, \perp)$ to $\mathcal{S}$.

  On input $(\mathsf{corrupt\text{-}check}, \mathsf{sid}, \mathsf{ssid})$, return whether $(\mathsf{sid}, \mathsf{ssid})$ is marked as corrupted.

a simulated commitment. The $\mathcal{S}_C(\mathsf{Equiv}, \mathsf{ssid}, \mathsf{P}, \pi, \mathsf{st}, m)$ command is used to equivocate a commitment string π such that it opens to m. The $\mathcal{S}_C(\mathsf{Ext}, \mathsf{ssid}, \mathsf{P}, \pi)$ command is used to extract a message from the commitment π. These algorithms come in handy for simulation purposes when $\mathcal{F}_{\mathsf{NICOM}}$ is used in bigger protocols.

Implementing $\mathcal{F}_{NICOM}$. We implement $\mathcal{F}_{\mathsf{NICOM}}$ in the full version [CSW20c] using the non-interactive commitment scheme of [CF01] based on equivocal commitments and CCA-2 secure public key encryption with oblivious ciphertext sampleability. The committer P commits to a bit message m as $c = \mathsf{Com}(m; r)$. The commitment randomness is encrypted via a pair of encryptions. The committer encrypts the corresponding randomness r, subsession id ssid and committer id P using a CCA-2 secure PKE as $E_m = \mathsf{Enc}(\mathsf{pk}, (r, \mathsf{ssid}, \mathsf{P}); s_m)$ with randomness s_m. The other encryption E_{1-m} is obliviously sampled using randomness s_{1-m}. The commitment consists of (c, E_0, E_1) and the opening information is (m, r, s_0, s_1). The verifier performs the canonical verification by reconstructing the commitment. The equivocal commitment can be instantiated from Pedersen Commitment and the obliviously sampleable encryption scheme can be instantiated from Cramer Shoup encryption [CS98], yielding a protocol from DDH. Similarly, we can instantiate the equivocal commitment from LWE [CsW19] and the obliviously sampleable encryption scheme from LWE [MP12].

2.2 Adaptively Secure Sigma Protocols in the $\mathcal{F}_{\mathsf{NICOM}}$ model

We recall the definition of a Sigma protocol and then we introduce the notion of adaptively Sigma protocols in the $\mathcal{F}_{\mathsf{NICOM}}$ model.

Sigma Protocol. A Sigma protocol consists of a prover possessing an NP statement $x \in \mathcal{L}$ (for language $\mathcal{L}$) and witness w which validates the statement. The verifier possesses the statement x. The prover constructs a first message a and the honest verifier challenges the prover with a random challenge $c \leftarrow_R \mathcal{C}$ from the challenge space $\mathcal{C}$. Based on the challenge, the prover computes a response z and sends it to the verifier. Completeness ensures that an honest verifier always accepts the proof (a, c, z). Soundness ensures that the verifier accepts a proof corresponding to an invalid statement $x' \notin \mathcal{L}$ with probability $\frac{1}{|\mathcal{C}|}$. The protocol is repeated κ times to obtain negligible (in κ) soundness error. We also require special soundness which guarantees a witness extractor given two accepting transcripts (a, c, z) and (a, c', z') corresponding to the same first message but different challenges $c \neq c' \in \mathcal{C}$. Finally, we need honest verifier zero knowledge which allows a simulator to simulate an accepting proof given an honestly sampled challenge c. The simulated proof should be indistinguishable from an honestly generated proof.

Limitations of a Sigma protocol. A Sigma protocol does not necessarily guarantee security against adaptive corruptions. The adversary can choose to corrupt the prover after obtaining the simulated proof. In such a case, the simulator obtains the witness and needs to provide prover's randomness such that the simulated proof is consistent with the witness. This problem crops up especially when the first message of the Sigma protocol [FLS99] is statistically binding and doesn't allow equivocation later on. To tackle this issue, we introduce the notion of adaptively secure Sigma protocols in the ideal UC commitment functionality (for multiple subsessions) $\mathcal{F}_{\mathsf{NICOM}}$ model. The traditional UC commitment functionality of [CF01] is not compatible with non-interactive commitments since the functionality is required to interact with the parties during Commit and open phases. So we use our new commitment functionality $\mathcal{F}_{\mathsf{NICOM}}$ which allows non-interactive Commit and Open phases.

Adaptively Secure Sigma Protocols. As seen above, the traditional Sigma protocols does not necessarily guarantee security against adaptive corruptions. In the light of this, we consider Sigma protocols in the $\mathcal{F}_{\mathsf{NICOM}}$ model. The prover sends the first message a to the verifier, the verifier sends a random challenge c to the prover and the prover computes the response z based on c. The prover and verifier has access to the $\mathcal{F}_{\mathsf{NICOM}}$ functionality during the protocol execution. In addition to HVZK and special soundness properties, we also require that the simulator is able to produce consistent randomness for a simulated proof and a valid witness when the prover gets corrupted post-execution. Looking ahead, the first message a will consist of commitments that are obtained by invoking $\mathcal{F}_{\mathsf{NICOM}}$ functionality. This enables the simulator to construct an HVZK proof

during protocol execution - where it opens few of the commitments in a which are required for verification. The other commitments in a remain unopened during the protocol. When the prover gets corrupted post-execution, the simulator obtains the witness w, and it equivocates the unopened commitments in a to produce a simulated prover's randomness such that it is indistinguishable from honestly sampled prover randomness (in the real world execution).

We also require special soundness property from our adaptively secure Sigma protocol to construct a NIZK protocol. We say that the protocol satisfies special soundness if there exists a extractor which extracts the witness given two transcripts (a, c_0, z_0) and (a, c_1, z_1) corresponding to the same a.

2.3 Compiling to an adaptively-secure NIZK

Next, we implement the $\mathcal{F}_{\mathsf{NIZK}}$ functionality for a single session by using the Fiat-Shamir transform on $\tau = \mathcal{O}(\kappa)$ iterations of the adaptively secure Sigma protocol. We instantiate the hash function in the Fiat-Shamir Transform using a correlation intractable hash function H [PS19, CCH+19, BKM20].

Correlation Intractability. A correlation intractable hash function H has the following property: For every efficient function f, given a hash function $H \leftarrow \mathcal{H}$ from the hash family $\mathcal{H}$, it is computationally hard to find an x s.t. $f(x) = H(x)$. Based on the first message $\mathbf{a}$ of a trapdoor-Sigma Protocol, the Fiat-Shamir challenge $\mathbf{e}$ can be generated using the hash function as $\mathbf{e} = H(\mathbf{a})$. The prover computes the third message $\mathbf{z}$ using $\mathbf{e}$. Trapdoor-Sigma protocol ensures that for every statement not in the language there can be only one bad challenge $\mathbf{e} = g(\mathbf{a})$ s.t. $(\mathbf{a}, \mathbf{e}, \mathbf{z})$ is an accepting transcript. By setting the function $f = g$ as the bad challenge function in H it is ensured that a malicious prover who constructs a bad challenge $\mathbf{e} = H(\mathbf{a})$ can be used to break correlation intractability since $\mathbf{e} = g(\mathbf{a}) = f(\mathbf{a})$. This guarantees soundness of the NIZK protocol.

Protocol. We compile our adaptively secure Sigma protocol into an adaptively secure NIZK in the $\mathcal{F}_{\mathsf{NICOM}}$ model (the $\mathcal{F}_{\mathsf{NICOM}}$ functionality is later instantiated using an adaptively secure non-interactive commitment scheme [CF01]). The prover computes the first message a^j of the adaptively secure Sigma protocol for the jth iteration where $j \in [\tau]$. It samples two challenges c_0^j and c_1^j from the challenge space such that $c_0^j \neq c_1^j$. The prover computes the responses z_0^j and z_1^j corresponding to both challenges c_0^j and c_1^j respectively. The prover commits to the challenges c_0^j and c_1^j, and the responses z_0^j and z_1^j. Let us denote the set of commitments as Y^j. The prover repeats the above protocol for τ iterations. Let $\mathbf{Y} = \{Y^j\}_{j\in[\tau]}$ denote the complete set of commitments and let $\mathbf{a} = \{a^j\}_{j\in[\tau]}$ denote the complete set of first messages. The prover computes the challenge bit-vector $\mathbf{e} = H(\mathsf{k}, (\mathbf{a}, \mathbf{Y}))$ (where k is the hash key) by invoking the hash function on the commitments $\mathbf{Y}$. The hash function is initialized in the statistical mode and the hash key contains the algorithm $\mathcal{S}_C$ obtained from $\mathcal{F}_{\mathsf{NICOM}}$. The hash function internally runs $\mathcal{S}_C$ to extract from the commitments. The hash key k

is provided as part of the crs and it is computed as follows.

$$\mathsf{k} = \mathcal{H}.\mathsf{StatGen}(\mathcal{C}_{\mathsf{sk}}).$$

$\mathcal{C}_{\mathsf{sk}}$ is a poly-size circuit that takes $\mathbf{Y}$ as input and $\mathsf{sk} = \mathcal{S}_C$ is the secret algorithm of $\mathcal{F}_{\mathsf{NICOM}}$. $\mathcal{C}_{\mathsf{sk}}(\mathbf{a}, \mathbf{Y})$ is the circuit computing the function $f_{\mathsf{sk}}(\mathbf{a}, \mathbf{Y}) = \mathbf{e}$ s.t. for $j \in [\tau]$, $e^j = 0$ iff (a^j, c_0^j, z_0^j) is an accepting proof where $\mathcal{C}_{\mathsf{sk}}$ extracts the challenges (c_0^j, c_1^j), and the responses (z_0^j, z_1^j) by running $\mathcal{S}_C$. Setting the hash function in the statistical mode ensures that the hash function H is correlation intractable for all relations of the form:

$$\mathcal{R}_{\mathsf{sk}} = \{(\mathbf{a}, \mathbf{Y}, \mathbf{e}) : \mathbf{e} = f_{\mathsf{sk}}(\mathbf{a}, \mathbf{Y})\}$$

In the jth iteration, upon obtaining e as the challenge bit the prover decommits to (c_0^j, c_1^j, z_e^j). The NIZK proof for the jth iteration is $(a^j, c_0^j, c_1^j, z_e^j)$ and the decommitments corresponding to (c_0^j, c_1^j, z_e^j). The verifier checks that - 1) the decommitments are correct, 2) the challenges are different, i.e. $c_0^j \neq c_1^j$, 3) the proof - (a^j, c_e^j, z_e^j) verifies. The verifier runs the verification protocols for every iteration $j \in [\tau]$. The verifier outputs accept if all the τ proofs verify correctly. Correctness of the protocol follows from the correctness of the commitment scheme and correctness of the sigma protocol.

Soundness and Proof of Knowledge. The soundness and proof of knowledge argument follows through a sequence of hybrids. The correlation intractability does not hold in the real world since we start off with an argument and not a proof. The proof starts off with the real world protocol in the first hybrid. In the second hybrid the proof relies on the binding and extractability property of the commitment scheme to ensure that the committed messages can be either correctly extracted or the commitment fails to open correctly. In the next hybrid, we rely on the special soundness property of the Sigma protocol to ensure that if for any jth iteration (for $j \in [\tau]$) if the prover constructs an accepting proof for both $e^j = 0$ and $e^j = 1$ then the witness extractor of the sigma protocol correctly extracts the underlying witness. In the final hybrid, if the prover has evaded the witness extractor and yet succeeded in creating an accepting proof then it has predicted the challenge vector $\mathbf{e}$ correctly by breaking the correlation intractability of the hash function. However, we know that there does not exist $\mathbf{e}$ such that the following holds due to statistical correlation intractability and the underlying Sigma protocol in this hybrid is a proof. This ensures that either the witness extractor extracts an accepting witness from atleast one of the iterations or the proof does not verify. This completes our soundness argument.

Adaptive Soundness. Adaptive Soundness follows along the same lines provided the underlying the sigma protocol satisfies adaptive soundness. The distribution of the crs is identical in the real and ideal world. Hence, we argue that the proof fails to verify for a statement $x \notin \mathcal{L}$ since there does not exist any valid witness.

Security against Adaptive Corruptions and Adaptive ZK. The ZK property crucially relies on the adaptive security of the Sigma protocol and security against adaptive corruptions of the commitment scheme. The ZK simulator of the NIZK protocol invokes the HVZK simulator the sigma protocol to obtain a simulated proof (a^j, c^j, z^j) corresponding to a random ZK challenge c^j for the jth iteration. The simulator constructs the commitments $\mathbf{Y}$ in the equivocal mode and invokes the hash function to obtain the challenge string $\mathbf{e}$. Upon obtaining the challenge bits e^j (for $j \in [\tau]$) the simulator opens the commitments corresponding to e^j to the simulated proof (a^j, z^j, c^j). It also equivocates the commitment for the ZK challenge corresponding to bit $1 - e^j$ to open to a different challenge $c^{j'}$ as part of the protocol. The proof verifies correctly due to the HVZK property of the Sigma protocol and equivocal property of the commitment scheme. Upon post-execution corruption of the prover, the NIZK simulator obtains the correct witness w and it invokes the simulator of the adaptively secure Sigma protocol with w to obtain the internal prover state. Using these information the NIZK simulator constructs the response corresponding to challenge $c^{j'}$ for choice bit $1 - e^j$. The simulator equivocates the commitments in $\mathbf{Y}$ (mainly the commitment to the jth response for challenge bit $1 - e^j$) such that the proofs corresponding to challenge bits $1 - e^j$ verify for every jth proof. Indistinguishability follows from the adaptive security of the Sigma protocol and the adaptive security of the commitment scheme. Adaptive zero-knowledge also follows along the same lines provided the sigma protocol satisfies adaptive zero-knowledge.

2.4 Constructing Adaptively Secure Sigma protocols with Special Soundness

Next, we show various instantiations of our adaptively sigma protocol which also satisfies special soundness. Plugging these protocols in a blackbox manner into our above compiler would yield a triply adaptive NIZK protocol.

Schnorr's [Sch90] Protocol. The classic Schnorr's identification protocol provides HVZK and satisfies special soundness. It also provides security against adaptive corruption. Let us recall the protocol and demonstrate that the Sigma protocol trivially satisfies adaptive security.

In the Schnorr's protocol the prover has a witness $w \in \mathbb{Z}_q$ and statement $x \in \mathbb{G}$ such that $x = g^w$, where $g \in \mathbb{G}$ is a generator of the cyclic group $\mathbb{G}$ where Discrete Log problem holds. The prover samples a random $r \in \mathbb{Z}_q$ and sets $a = g^r$. Upon obtaining a random challenge $c \in \mathbb{Z}_q$ from the verifier the prover sends $z = r + wc$ as the response. The verifier checks that $g^z \stackrel{?}{=} a \cdot x^c$.Given two accepting trancripts (a, c, z) and (a, c', z') the witness w can be extracted as $w = \frac{(z-z')}{c-c'}$. On the other hand, for HVZK the simulator samples a random $c \in \mathbb{Z}_q$ and a random $z \in Zq$ and computes $a = \frac{g^z}{x^c}$. Upon post-execution corruption of prover, the simulator obtains w and sets $r = z - wc$ as the internal state. It is straightforward to see that adaptive security follows.

Adaptive Soundness and Adaptive ZK. Adaptive soundness cannot be defined for Schnorr's protocol since every statement $x' \in \mathbb{G}$ lies in the language corresponding to the witness $w' \in \mathbb{Z}_q$ where $x' = g^{w'}$. Adaptive ZK follows from the HVZK property of the protocol.

Sigma protocol of [FLS99]. We briefly recall the Sigma protocol of [FLS99] (FLS) for the sake of completeness. Let $\mathcal{R}_{\mathsf{Ham}}$ be the set of Hamiltonian graphs. The prover P proves that an n-node graph G is Hamiltonian, i.e. $G \in \mathcal{R}_{\mathsf{Ham}}$, given a Hamiltonian cycle σ as a witness. P samples a random n-node cycle H and commits to the adjacency matrix of the cycle as the first message a. The matrix contains n^2 entries, and P commits to the edges as $\mathsf{Com}(1)$, and non-edges as $\mathsf{Com}(0)$. The prover sends these commitments to the verifier V. V samples a random challenge bit e and sends it to the prover. If $c = 0$, then P decommits to the cycle H. Else, it computes a random permutation π s.t. $H = \pi(\sigma)$ and decommits to the non-edges in $\pi(G)$ and sends π. P sends the decommitments as its response z. Upon obtaining z, the verifier performs the following check:

- $c = 0$: Verify that z contains decommitments to 1, and they form a valid cycle, i.e. the prover must have committed to a valid n-node cycle.
- $c = 1$: Verify that z contains decommitments to 0, and the decommitted edges correspond to non-edges in $\pi(G)$.

Special Soundness. There are only two possible challenges in the boolean challenge space. Given the transcripts $(a, 0, z_0)$ and $(a, 1, z_1)$ where a_c and a'_c are computed as described above, the witness extractor obtains H from z_0 and π from z_1. The extractor computes the witness cycle as $\sigma = \pi^{-1}(H)$. This proves special soundness property of the Sigma protocol.

Honest Verifier Zero Knowledge. The FLS protocol achieves honest verifier zero knowledge. The ZK simulator samples a random challenge $e \in \{0, 1\}$ and based on that he computes (a, z) as follows.

- $c = 0$: The simulator samples a random n-node cycle H and commits to the adjacency matrix of the cycle as a. It sets z as the decommitment to the cycle.
- $c = 1$: The simulator sets all the commitments to 0 in a, i.e. commits to a null graph. It computes a random permutation π and decommits to the non-edges in $\pi(G)$. It sets z as π and the decommitments to the non-edges in $\pi(G)$.

Let us denote a proof as $\gamma = (a, e, z)$. It can be observed that an honest γ is identically distributed to a simulated γ when $e = 0$. When $e = 1$, an honestly γ contains a committed cycle whereas γ contains commitments to 0. The two proofs are indistinguishable due to the hiding of the commitment scheme.

Adaptive Security in $\mathcal{F}_{NICOM}$ model. We observe that the FLS protocol satisfies adaptive security if the commitments in a are computed in the $\mathcal{F}_{\mathsf{NICOM}}$ model. We consider the simulated ZK proof and adaptive corruption of prover as follows:

- $c = 0$: The HVZK simulator samples a random n-node cycle H and commits to the adjacency matrix of the cycle as a by invoking $\mathcal{F}_{\mathsf{NICOM}}$. It sets z as the decommitment to the cycle.
  Upon post execution corruption of prover, the simulator obtains the witness cycle σ and it computes the permutation π such that $H = \pi(\sigma)$. The internal state of the prover is set as a, permutation π and the internal state of the committer returned by $\mathcal{F}_{\mathsf{NICOM}}$ (for computing the commitments in a).
- $c = 1$: The HVZK simulator sets a as the commitments to 0 in the $\mathcal{F}_{\mathsf{NICOM}}$ model, i.e. the simulator commits to a null graph. It computes a random permutation π, and sets z as the random permutation π and the decommitments to the non-edges in $\pi(G)$.
  Upon post execution corruption of prover, the simulator obtains the witness cycle σ and it computes the permutation π such that $H = \pi(\sigma)$. The simulator equivocates the unopened commitments in a by invoking the $\mathcal{F}_{\mathsf{NICOM}}$ simulator, such that the unopened commitments decommit to H. The internal state of the prover is set to the permutation π and the commitment randomness returned by $\mathcal{F}_{\mathsf{NICOM}}$ for all the commitments.

For the case of $c == 0$, it can be observed that the simulated internal state is identical to the honest prover internal state. When $c == 1$, the simulated proof consists of commitments to 0 and the simulated prover internal state consists of equivocation randomness which was returned by $\mathcal{F}_{\mathsf{NICOM}}$. Hence, the real and ideal world views are identically distributed in the $\mathcal{F}_{\mathsf{NICOM}}$ model. This shows that the FLS protocol can be plugged into our NIZK compiler to obtain a NIZK protocol which is secure against adaptive corruptions.

Adaptive Soundness and Adaptive ZK. In FLS, the first message a of the prover is computed based on the parameter n without the knowledge of the graph or the witness. After obtaining c from V, the prover requires the input graph G and the witness cycle σ to construct the response. Thus, only the last message in this protocol depends on the input. This property is called delayed-input property. And hence the FLS protocol trivially satisfies adaptive soundness and adaptive ZK in the $\mathcal{F}_{\mathsf{NICOM}}$ model where the input statement can be adversarially chosen after observing the setup string distribution. This allows our NIZK protocol to be adaptively sound and satisfy adaptive ZK when the FLS Sigma protocol is plugged into the triply adaptive NIZK compiler.

Blum's protocol for Hamiltonicity. Following the above idea, it can be shown that the Blum's protocol [Blu86] for hamiltonicity also satisfies adaptive security and special soundness in the $\mathcal{F}_{\mathsf{NICOM}}$ model. The protocol itself does not satisfy delayed input property since the first message of the prover depends on the statement. However, the protocol does achieve adaptive soundness since a

malicious prover would be unsuccessful in generating an accepting proof for a statement $x \notin \mathcal{L}$ in the $\mathcal{F}_{\mathsf{NICOM}}$ model.

Garbled circuit based protocol of [HV16]. Next, we modify the GC based protocol of [HV16] to obtain an adaptively secure sigma protocol with special soundness in the $\mathcal{F}_{\mathsf{NICOM}}$ model. We recall their protocol and then discuss the bottlenecks involved.

Protocol of [HV16]. The protocol of [HV16] constructs an adaptively secure ZK proof from one-way functions in the plain model. Their protocol relies on a special commitment scheme called adaptive-instance dependent commitment (AIDCS) schemes. It depends on the statement being proven. AIDCS is statistically binding when the statement (being proven) is not in the language. AIDCS is equivocal when the statement is in the language. The committer can open a commitment to any message given an accepting witness for the statement. In [HV16], the prover constructs a garbled circuit computing the NP relation on the statement x. The prover commits to the garbled circuit $\mathbf{GC}$ (garbling notations can be found in [HV16, CSW20c]), encoding information $\mathbf{u}$ and the decoding information $\mathbf{v}$ using the AIDCS. These commitments are jointly denoted as the first message a. The verifier sends the challenge bit c. If the bit is $c = 0$ then the prover decommits to $(\mathbf{GC}, \mathbf{u}, \mathbf{v})$. The verifier checks that the garbled circuit was correctly constructed. If the bit is $c = 1$ then the prover computes the input wire labels W corresponding to the witness w and decommits to W, the decoding information $\mathbf{v}$ and the path of the computation as $\mathsf{path} = \Pi_{\mathsf{Ev}}(\mathbf{GC}, \mathsf{W})$ in the $\mathbf{GC}$ which corresponds to evaluation of $\mathbf{GC}$ on W. The verifier accepts if the computation of the garbled circuit on W along the path outputs 1. When x is not in the language the AIDCS is statistically binding and hence the prover has to guess the verifier's bit. For ZK, the ZK simulator will guess the random challenge bit of verifier and it will rewind if the guess is wrong. When the prover gets corrupted post-execution, the simulator can equivocate the commitments given the witness w using the equivocal property of AIDCS.

Bottlenecks in obtaining NIZK. The proof is not binding when $x \in \mathcal{L}$ and a corrupt prover knows the witness since the AIDCS is equivocal given the witness. A malicious prover evades the special soundness property using the following adversarial strategy: The adversary constructs the AIDCS in the equivocal mode as the first message a and it constructs the responses as follows:

- $c_0 == 0$ *:* It samples a garbled circuit as $(\mathbf{GC}, \mathbf{u}, \mathbf{v})$ and sets z_0 as $(\mathbf{GC}, \mathbf{u}, \mathbf{v})$ and the decommitment of a to $(\mathbf{GC}, \mathbf{u}, \mathbf{v})$.
- $c_1 == 1$ *:* It invokes the privacy simulator of the garbled circuit on output 1 to obtain a simulated GC and input wire labels for evaluation. The adversary sets the response z_1 as these wire labels and the path of GC evaluation. The response z_1 also contains the decommitments of a to the wire labels and the evaluation path.

The adversary is able to equivocate the AIDCS to open to different values and this hampers witness extraction from the two accepting transcripts (a, c_0, z_0) and (a, c_1, z_1). This hampers the special soundness property.

Our Solution. We solve this issue by replacing the AIDCS with the $\mathcal{F}_{\mathsf{NICOM}}$ model and demonstrate that the new Sigma protocol in the $\mathcal{F}_{\mathsf{NICOM}}$ model satisfies adaptive security and special soundness property. The prover constructs a garbled circuit computing the NP relation on the statement x. The prover sets a as the commitment to garbled circuit $\mathbf{GC}$, encoding information $\mathbf{u}$ and the decoding information $\mathbf{v}$ in the $\mathcal{F}_{\mathsf{NICOM}}$ model. The prover sends a to the verifier. The verifier sends the challenge bit c. The prover performs the following based on challenge c:

- $c = 0$: The prover decommits to the garbled circuit $\mathbf{GC}$, encoding information $\mathbf{u}$ and decoding information $\mathbf{v}$ as the response z_0.
- $c = 1$: The prover decommits to the input wire labels and the evaluation path in the garbled circuit as the response z_1.

The verifier performs verification using the original verifier algorithm of [HV16]. Completeness is straightforward. We show that the above Sigma protocol satisfies special soundness property and adaptive security in $\mathcal{F}_{\mathsf{NICOM}}$ model.

Special Soundness. There are only two possible challenges in the boolean challenge space. Given two accepting transcripts $(a, 0, z_0)$ and $(a, 1, z_1)$, the witness extractor obtains the encoding information $\mathbf{u}$ and the input wire labels W. Assuming the garbling scheme is projective (for every input wire in the circuit the encoding information consists of two posible wire labels corresponding to bit values 0 and 1), it maps the wire labels with the encoding information to extract the witness w. This proves special soundness property of the Sigma protocol.

Adaptive Security in $\mathcal{F}_{\mathsf{NICOM}}$ model. We describe the HVZK simulator and then extend it to satisfy adaptive security in the $\mathcal{F}_{\mathsf{NICOM}}$ model. We crucially rely on the reconstructability property of the garbling scheme to argue adaptive security. Reconstructability property says that given a path of computation, the input wire labels and the input to a garbled circuit for circuit C it is possible to reconstruct the rest of the garbled circuit as being honestly generated by the garbling algorithm. We define the HVZK simulator as follows based on the challenge c :

- $c = 0$: The HVZK simulator computes a fresh garbled circuit as $(\mathbf{GC}, \mathbf{u}, \mathbf{v})$ and commits to it using $\mathcal{F}_{\mathsf{NICOM}}$ as the first message a. It sets a as the commitment to $(\mathbf{GC}, \mathbf{u}, \mathbf{v})$. The simulator sends z_0 as $(\mathbf{GC}, \mathbf{u}, \mathbf{v})$ and the decommitments to a.
  When the prover gets adaptively corrupted, the simulator obtains the witness w and it sets the randomness used to garble $\mathbf{GC}$ and the commitment randomness as the internal randomness.

– $c = 1$: The HVZK simulator invokes the GC privacy simulator on output 1 and circuit C to obtain a simulated garbled circuit, input wire label, decoding information and internal state - $(\mathbf{GC}', \mathsf{W}', \mathbf{v}', \mathsf{st}')$. The HVZK simulator sets a as the commitment to $(\mathbf{GC}', 0^{|\mathbf{u}|}, \mathbf{v}')$ in the $\mathcal{F}_{\mathsf{NICOM}}$ model. The simulator computes the path of computation as $\mathsf{path} = \Pi_{\mathsf{Ev}}(\mathbf{GC}', \mathsf{W}')$ on wire labels W'. The simulator sends z_1 as $(\mathsf{path}, \mathsf{W}')$ and decommitment to $(\mathsf{path}, \mathsf{W}')$ from the set of commitments in a.
When the prover gets adaptively corrupted, the simulator obtains the witness w. Using input w, simulated input wire labels W' and the computation path path, it uses the reconstructability property of the garbling scheme to reconstruct a fresh garbled $\mathbf{GC}$, encoding information $\mathbf{u}$ and decoding information $\mathbf{v}$ and the corresponding garbling randomness. It sets the garbling randomness as the internal state and invokes the $\mathcal{F}_{\mathsf{NICOM}}$ simulator to equivocate the commitments in a such that they open to $(\mathbf{GC}, \mathbf{u}, \mathbf{v})$.

For the case of $c == 0$, it can be observed that the simulated internal state is identical to the honest prover internal state. When $c == 1$, the proof contains the evaluation path, the input wire labels and their decommitments. Upon post execution corruption the simulator relies on the reconstructability property of the garbling scheme to construct the garbled circuit. The distribution of the simulated a in the ideal world is indistinguishable from the honestly constructed a in the real world in the $\mathcal{F}_{\mathsf{NICOM}}$ model due to the reconstructability property. The garbling scheme of [LP09] based on one-way functions satisfies all the required properties for the Sigma protocol. This was shown in the work of [HV16].

Adaptive Soundness and Adaptive ZK. The protocol achieves adaptive soundness and adaptive ZK even when the functionality $\mathcal{F}_{\mathsf{NICOM}}$ is implemented by an adaptively secure commitment protocol [CF01] in the crs model. The distribution of crs is identical in the real and ideal world. A malicious prover fails to prove a false statement $x \notin \mathcal{L}$ without breaking the binding of the commitment scheme (implementing $\mathcal{F}_{\mathsf{NICOM}}$ functionality). Adaptive ZK follows from the adaptive security of the protocol.

3 Preliminaries

Notations. We denote by $a \leftarrow D$ a uniform sampling of an element a from a distribution D. The set of elements $\{1, 2, \ldots, n\}$ is represented by $[n]$. A function $\mathsf{neg}(\cdot)$ is said to be negligible, if for every polynomial $p(\cdot)$, there exists a constant c, such that for all $n > c$, it holds that $\mathsf{neg}(n) < \frac{1}{p(n)}$. We denote a probabilistic polynomial time algorithm as PPT. We denote the computational and statistical security parameters by κ by μ respectively. We denote computational and statistical indistinguishability by $\overset{c}{\approx}$ and $\overset{s}{\approx}$ respectively. When a party $\mathcal{P}$ gets corrupted we denote it by $\mathcal{P}^*$. Let $\mathcal{R}_{\mathsf{Ham}}$ denote the set of n-node Hamiltonian graphs for $n > 1$. We prove security of our protocol in the Universal Composability (UC) model. We refer to the original paper [Can01] for details.

Our protocols are in the common reference string model where the parties of a session $(\mathsf{sid}, \mathsf{ssid})$ have access to a public reference string crs sampled from a distribution. In the one-time crs model, each crs is local to each $(\mathsf{sid}, \mathsf{ssid})$. In the reusable crs model, the same crs can be reused across different sessions by different parties. The simulator knows the trapdoors of the crs in both cases. We refer to [CLOS02] for more details.

Definition 1. *[DN00]*(**PKE *with oblivious ciphertext sampling***) *A public key encryption scheme* $\mathrm{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *over message space* $\mathcal{M}$*, ciphertext space* $\mathcal{C}$ *and randomness space* $\mathcal{R}$ *satisfies oblivious ciphertext sampling property if there exists PPT algorithms* $(\mathsf{oEnc}, \mathsf{Inv})$ *s.t. for any message* $m \in \mathcal{M}$*, the following two distributions are computationally indistinguishable to a PPT adversary* $\mathcal{A}$*:*

$$\big|\Pr[\mathcal{A}(m,c,r)=1 | m \leftarrow \mathcal{A}(\mathsf{pk}), c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r'), r \leftarrow \mathsf{Inv}(\mathsf{pk}, c)]$$

$$-\Pr[\mathcal{A}(m,\tilde{c},r)=1 | m \leftarrow \mathcal{A}(\mathsf{pk}), \tilde{c} \leftarrow \mathsf{oEnc}(\mathsf{pk}; r)]\big| \leq \mathsf{neg}(\kappa),$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$.

We instantiate CCA-2 secure PKE with oblivious ciphertext sampling from DDH [CS98] and LWE [MP12].

3.1 Non-Interactive Zero Knowledge

We provide the ideal UC-NIZK functionality in Fig. 2 for a single prover and a single proof. It also considers the case for adaptive corruption of parties where the prover gets corrupted after outputting the proof π. In such a case, the adversary receives the internal state of the prover.

Fig. 2. Single-Proof Non-Interactive Zero-Knowledge Functionality $\mathcal{F}_{\mathsf{NIZK}}$

$\mathcal{F}_{\mathsf{NIZK}}$ is parametrized by an NP relation $\mathcal{R}$. (The code treats $\mathcal{R}$ as a binary function.)

- **Proof:** On input $(\mathsf{prove}, \mathsf{sid}, x, w)$ from party P: If $\mathcal{R}(x,w) = 1$ then send $(\mathsf{prove}, P, \mathsf{sid}, x)$ to $\mathcal{S}$. Upon receiving $(\mathsf{proof}, \mathsf{sid}, \pi)$ from $\mathcal{S}$, store $(\mathsf{sid}, x, w, \pi)$ and send $(\mathsf{proof}, \mathsf{sid}, \pi)$ to P.
- **Verification:** On input $(\mathsf{verify}, \mathsf{sid}, x, \pi)$ from a party V: If $(\mathsf{sid}, x, w, \pi)$ is stored, then return $(\mathsf{verification}, \mathsf{sid}, x, \pi, \mathcal{R}(x,w))$ to V. Else, send $(\mathsf{verify}, V, \mathsf{sid}, x, \pi)$ to $\mathcal{S}$. Upon receiving $(\mathsf{witness}, \mathsf{sid}, w)$ from $\mathcal{S}$, store $(\mathsf{sid}, x, w, \pi)$, and return $(\mathsf{verification}, \mathsf{sid}, x, \pi, \mathcal{R}(x,w))$ to V.
- **Corruption:** When receiving $(\mathsf{corrupt}, \mathsf{sid})$ from $\mathcal{S}$, mark sid as corrupted. If there is a stored tuple $(\mathsf{sid}, x, w, \pi)$, then send it to $\mathcal{S}$.

We also consider $\mathcal{F}^{\mathsf{m}}_{\mathsf{NIZK}}$ (Fig. 3) functionality where a single prover can parallelly prove multiple statements in a single session. The verifier verifies each of them separately. It is a weaker notion than multi-session UC NIZK since $\mathcal{F}^{\mathsf{m}}_{\mathsf{NIZK}}$

Fig. 3. Non-Interactive Zero-Knowledge Functionality $\mathcal{F}_{\mathsf{NIZK}}^{\mathsf{m}}$ for single prover multi-proof setting

$\mathcal{F}_{\mathsf{NIZK}}$ is parametrized by an NP relation $\mathcal{R}$. (The code treats $\mathcal{R}$ as a binary function.)

- **Proof:** On input $(\mathsf{prove}, \mathsf{sid}, \mathsf{ssid}, x, w, P)$ from party P: If there exists $(\mathsf{sid}, P') \in \mathcal{Q}$ and $P \neq P'$ or $\mathcal{R}(x, w) \neq 1$ then ignore the input. Else record $\mathcal{Q} = (\mathsf{sid}, \mathsf{ssid}, P)$. Send $(\mathsf{prove}, P, \mathsf{sid}, \mathsf{ssid}, x)$ to $\mathcal{S}$. Upon receiving $(\mathsf{proof}, \mathsf{sid}, \mathsf{ssid}, \pi)$ from $\mathcal{S}$, store $(\mathsf{sid}, \mathsf{ssid}, x, w, \pi)$ and send $(\mathsf{proof}, \mathsf{sid}, \mathsf{ssid}, \pi)$ to P.
- **Verification:** On input $(\mathsf{verify}, \mathsf{sid}, \mathsf{ssid}, x, \pi)$ from a party V: If $(\mathsf{sid}, \mathsf{ssid}, x, w, \pi)$ is stored, then return $(\mathsf{verification}, \mathsf{sid}, \mathsf{ssid}, x, \pi, \mathcal{R}(x, w))$ to V. Else, send $(\mathsf{verify}, V, \mathsf{sid}, \mathsf{ssid}, x, \pi)$ to $\mathcal{S}$. Upon receiving $(\mathsf{witness}, \mathsf{sid}, \mathsf{ssid}, w)$ from $\mathcal{S}$, store $(\mathsf{sid}, \mathsf{ssid}, x, w, \pi)$, and return $(\mathsf{verification}, \mathsf{sid}, \mathsf{ssid}, x, \pi, \mathcal{R}(x, w))$ to V.
- **Corruption:** When receiving $(\mathsf{corrupt}, \mathsf{sid}, \mathsf{ssid})$ from $\mathcal{S}$, mark $(\mathsf{sid}, \mathsf{ssid})$ as corrupted. If there are stored tuples of the form $(\mathsf{sid}, \mathsf{ssid}, x, w, \pi)$, then send it to $\mathcal{S}$.
  On input $(\mathsf{corrupt\text{-}check}, \mathsf{sid}, \mathsf{ssid})$, return whether $(\mathsf{sid}, \mathsf{ssid})$ is marked as corrupted.

considers only a single session between a pair of parties with roles preserved. Different provers have to use different instances of $\mathcal{F}_{\mathsf{NIZK}}^{\mathsf{m}}$ to prove statements.

Next, we define the notion of triple adaptive security for NIZK protocols and provide the property-based definitions of NIZK for completeness. UC-secure NIZKs in the crs model imply adaptive ZK since an environment can statically corrupt the verifier, obtain the crs of the protocol and then choose the statement x to be proven by the honest prover. The simulator against a corrupt verifier ensures that it constructs an accepting simulated proof which is indistinguishable from an honestly generated proof. Hence, UC-NIZK implies adaptive ZK if the environment is allowed to choose the statement being proven after corrupting the verifier.

Definition 2. *A non-interactive zero-knowledge argument system (NIZK) for an NP-language $\mathcal{L}$ consists of three PPT machines $\Pi_{\mathsf{NIZK}} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$, that have the following properties:*

- ***Completeness**: For all $\kappa \in \mathsf{N}$, and all $(x, w) \in \mathcal{R}$, it holds that:*

$$\Pr[\mathsf{V}(\mathsf{crs}, x, \mathsf{P}(\mathsf{crs}, x, w)) = 1 | (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, 1^{|x|})] = 1.$$

- ***Soundness**: For all PPT provers P^* and $x \notin \mathcal{L}$ the following holds for all $\kappa \in \mathsf{N}$:*

$$\Pr[\mathsf{V}(\mathsf{crs}, x, \pi) = 1 | (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, 1^{|x|}), \pi \leftarrow \mathsf{P}^*(\mathsf{crs})] \leq \mathsf{neg}(\kappa).$$

- ***Zero knowledge**: There exists a PPT simulator $\mathcal{S}$ such that for every $(x, w) \in \mathcal{R}$, the following distribution ensembles are computationally indistinguishable:*

$$\{(\mathsf{crs}, \pi) | (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, 1^{|x|}), \pi \leftarrow \mathsf{P}(\mathsf{crs}, x, w)\}_{\kappa \in \mathsf{N}}$$

$$\approx \{(\mathsf{crs}, \{\mathcal{S}(1^\kappa, x, \mathsf{td})\}) | (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, 1^{|x|}\}_{\kappa \in \mathbb{N}}$$

Definition 3. *(**Full Adaptive Soundness**) Π_{NIZK} is adaptively sound if for every PPT cheating prover P^* the following holds:*

$$\Pr[x \notin \mathcal{L} \wedge \mathsf{V}(\mathsf{crs}, x, \pi) = 1 | (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, 1^{|x|}), (x, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})] < \mathsf{neg}(\kappa).$$

Definition 4. *(**Adaptive Zero-Knowledge**) Π_{NIZK} is adaptively zero-knowledge if for all PPT verifiers V^* there exists a PPT simulator $\mathcal{S}$ such that the following distribution ensembles are computationally indistinguishable:*

$$\{(\mathsf{crs}, \mathsf{P}(\mathsf{crs}, x, w), \mathsf{aux})\} \overset{c}{\approx} \{\mathcal{S}(\mathsf{crs}, \mathsf{td}, 1^\kappa, x)\}_{\kappa \in \mathbb{N}}$$

where $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, 1^{|x|})$ and $(x, w, \mathsf{aux}) \leftarrow \mathsf{V}^*(\mathsf{crs})$.

The Gen algorithm takes the $|x|$ (length of the statement) as input to generate the crs. This shows that the crs size depends on $|x|$. When the crs is independent of $|x|$, the Gen algorithm only takes 1^κ as input.

Definition 5. *(**Triple Adaptive Security for a single instance**)*
Let $\Pi_{\mathsf{NIZK}} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ be a NIZK protocol in the crs model. Then Π_{NIZK} satisfies triple adaptive security for a single instance if it securely implements $\mathcal{F}_{\mathsf{NIZK}}$ functionality for a single instance and provides adaptive soundness and adaptive zero knowledge.

Definition 6. *(**Triple Adaptive Security for multiple instances**)*
Let $\Pi_{\mathsf{NIZK}} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ be a NIZK protocol in the crs model. Then Π_{NIZK} satisfies triple adaptive security for multiple instances if it UC-securely implements $\mathcal{F}_{\mathsf{NIZK}}$ functionality for multiple instances and provides adaptive soundness and adaptive zero knowledge.

3.2 Commitment Schemes

A commitment scheme $\mathsf{Com} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Ver}, \mathsf{Equiv})$ allows a committing party C to compute a commitment c to a message m, using randomness r, towards a party V in the Com phase. Later in the open phase, C can open c to m by sending the decommitment to V who verifies it using Ver. It should be binding, hiding and equivocal using Equiv algorithm given trapdoor td of the crs. Moreover, we require our commitment scheme to be additively homomorphic for message domain of size at least four, i.e. $\mathsf{Com}(m_1; r_1) + \mathsf{Com}(m_2; r_2) = \mathsf{Com}(m_1 + m_2; r_1 + r_2)$. We also need a tag-based simulation sound commitment consists of $\mathsf{Com}_{\mathsf{SST}} = (\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Ver}, \mathsf{TCom}, \mathsf{TOpen})$ for our protocols. Formal definitions can be found in the full version [CSW20c].

$\mathcal{F}_{\mathsf{NICOM}}$-model. We also provide a new non-interactive UC-commitment functionality in Fig. 1. The $\mathcal{F}_{\mathsf{NICOM}}$ functionality (Fig. 1) is implemented against adaptive adversaries using adaptively secure non-interactive UC commitments [CF01] in the crs model. We perform this using equivocal commitments and CCA-2 secure PKE with oblivious ciphertext sampleability in the non-programmable crs model. It can be found in the full version [CSW20c]. We also prove that this new functionality implies the old UC commitment functionality (of [CF01]) but our new functionality is more compatible with non-interactive protocols.

3.3 Correlation Intractability.

As in [CCH+19, PS19, BKM20] we define efficiently searchable relations and recall the definitions of correlation intractability, in their computational and statistical versions.

Definition 7. *We say that a relation $R \subseteq X \times Y$ is searchable in size S if there exists a function $f : X \to Y$ that is implementable as a boolean circuit of size S, such that if $(x, y) \in R$ then $y = f(x)$. (In other words, $f(x)$ is the unique witness for x, if such a witness exists.)*

Definition 8. *Let $R = \{\mathcal{R}_\kappa\}$ be a relation class, i.e., a set of relations for each κ. A hash function family $\mathcal{H} = (\mathsf{Gen}, H)$ is correlation intractable for R if for every non-uniform PPT adversary $\mathcal{A} = \{\mathcal{A}_\kappa\}$ and every $R \in \mathcal{R}_\kappa$ the following holds:*

$$\Pr[(x, H(\mathsf{k}, x)) \in R : \mathsf{k} \leftarrow \mathsf{Gen}(1^\kappa), x = \mathcal{A}_\kappa(\mathsf{k})] \leq \mathsf{neg}(\kappa)$$

Definition 9. *Let $R = \{\mathcal{R}_\kappa\}$ be a relation class. A hash function family $\mathcal{H} = (\mathsf{Gen}, H)$ with a fake-key generation algorithm $\mathsf{StatGen}$ is somewhere statistically correlation intractable for R if for every $R \in R_\kappa$ and circuits $\exists z_R \in Z_\kappa$ s.t:*

$$\Pr[\exists x \text{ s.t. } (x, H(\mathsf{k}, x)) \in R : \mathsf{k} \leftarrow \mathsf{StatGen}(1^\kappa, z_R)] \leq \mathsf{neg}(\kappa).$$

and for every $z_\kappa \in Z_\kappa$ if the following distributions the indistinguishable:

$$\{\mathsf{StatGen}(1^\kappa, z_\kappa)\}_\kappa \overset{c}{\approx} \{\mathsf{Gen}(1^\kappa)\}_\kappa.$$

Definition 10. *A hash family $\mathcal{H} = (\mathsf{Gen}, H)$, with input and output length $n := n(\kappa)$ and, resp., $m := m(\kappa)$, is said to be programmable if the following two conditions hold:*

- *1-Universality: For every $\kappa \in \mathsf{N}, x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, the following holds: $\Pr[H(\mathsf{k}, x) = y : \mathsf{k} \leftarrow \mathsf{Gen}(1^\kappa)] = 2^{-m}$.*
- *Programmability: There exists a PPT algorithm $\mathsf{Gen}'(1^\kappa, x, y)$ that samples from the conditional distribution $Sample(1^\kappa)|H(\mathsf{k}, x) = y$.*

4 Triply Adaptive NIZK Argument in the crs model

In this section, we present our NIZK protocol. First, we recall the definition of Sigma protocol in the crs model and then build upon it to define adaptively Sigma protocol in the $\mathcal{F}_{\mathsf{NICOM}}$ model. Finally, we compile adaptively Sigma protocols into NIZKs using the Fiat-Shamir transform.

4.1 Sigma Protocol

We consider Sigma protocol [CPV20] $\Sigma = (\mathsf{Setup}, \mathsf{P}_1, \mathsf{V}_1, \mathsf{P}_2, \mathsf{V}_2)$ for relation $\mathcal{R}$ between a prover P and a verifier V that receive a common input statement x. P receives an additional private input a witness w for x. The protocol has the following form:

- $\mathsf{Setup}(1^\kappa)$: The Setup algorithm runs on security parameter κ and generates a common reference string crs and a trapdoor td. The crs is published as the public setup string.
- $\mathsf{P}_1(\mathsf{crs}, x, w, 1^\kappa; \mathsf{st})$: The prover runs algorithm P_1 on common input x, crs, private input w, security parameter κ and randomness st obtaining $a = \mathsf{P}_1(x, w, 1^\kappa; \mathsf{st})$ and sends a to verifier.
- $\mathsf{V}_1(\mathsf{crs}, a)$: Verifier samples random challenge $c \leftarrow_R \mathcal{C}$ and sends c to prover.
- $\mathsf{P}_2(\mathsf{crs}, x, w, \mathsf{st}, c)$: The prover runs algorithm P_2 on input $x, w, \mathsf{crs}, \mathsf{st}, c$ and obtain z. It sends z to verifier.
- $\mathsf{V}_2(\mathsf{crs}, x, a, c, z)$: The verifier outputs 1 if it accepts the proof else it outputs 0 to reject the proof.

The above protocol should satisfy completeness, honest verifier zero knowledge and special soundness. We refer to the full version [CSW20c] for the property definitions of Sigma protocol.

4.2 Fully Adaptive Sigma Protocol in $\mathcal{F}_{\mathsf{NICOM}}$ model

The traditional Sigma protocols are not secure against adaptive corruption of parties. Hence, we introduce the notion of fully adaptive Sigma protocols in the UC-commitment functionality $\mathcal{F}_{\mathsf{NICOM}}$ model. Consider the above Sigma protocol transcript (a, c, z). In the fully adaptive Sigma protocol, the prover has access to the $\mathcal{F}_{\mathsf{NICOM}}$ functionality while computing the first message a. The prover sends a to the verifier. Upon obtaining the challenge c, the prover computes the response z and sends it to the verifier.

Definition 11. *Let $\Sigma = (\mathsf{Setup}, \mathsf{P}_1, \mathsf{V}_1, \mathsf{P}_2, \mathsf{V}_2)$ be a Sigma protocol for relation $\mathcal{R}$ over corresponding domains $(\mathcal{A}, \mathcal{C}, \mathcal{Z})$, where parties make use of an instance of $\mathcal{F}_{\mathsf{NICOM}}$ where the prover is the commiter, and where the first message consists exclusively of a sequence of commitment strings that the prover obtains from $\mathcal{F}_{\mathsf{NICOM}}$. Then Σ is* fully adaptive *in the $\mathcal{F}_{\mathsf{NICOM}}$ model if the following requirements hold:*

1. ***Completeness.*** *If $(x, w) \in \mathcal{R}$, then honest transcripts of the form (x, a, c, z) obtained by the verifier for (x, w) are accepting.*
2. ***Computational Special soundness.*** *There exists a PPT algorithm Ext such that for any polytime adversarial prover P^* and two transcripts (a, c, z) and (a, c', z'), such that $P^*(\kappa) \to (S_C, x, a)$ where S_C is the adversarial code used by $\mathcal{F}_{\mathsf{NICOM}}$, $P^*(\kappa, c) \to z$, $P^*(\kappa, c') \to z'$, $c' \neq c$, and such that the verifier accepts both transcripts when given access to $\mathcal{F}_{\mathsf{NICOM}}^{S_C}$, it holds that:*

$$\Pr[\mathsf{Ext}(\mathsf{crs}, S_C, x, a, c, z, c', z') = w \ \&\ (x, w) \notin \mathcal{R}] < \mathsf{neg}(\kappa)$$

3. ***Adaptive Honest-verifier zero knowledge.*** *There exists PPT algorithm* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *such that, for any* $(x, w) \in \mathcal{R}$*, any PPT distinguisher* $\mathcal{A}$*, and any PPT adversarial code* S_C *for* $\mathcal{F}_{\mathsf{NICOM}}$*:*

$$\Big|\Pr\left[(a, c, z, st) \leftarrow \mathcal{S}_1(\mathsf{crs}, S_C, x; \mathsf{td}), r \leftarrow \mathcal{S}_2(st, w) : \mathcal{A}^{\mathcal{F}_{\mathsf{NICOM}}^{S_C}}(a, c, z, r) = 1\right] -$$
$$\Pr\left[r \leftarrow \{0,1\}^\kappa, (a, st) \leftarrow P_1^{\mathcal{F}_{\mathsf{NICOM}}^{S_C}}(x, w, r), c \leftarrow \mathcal{C}, z \leftarrow P_2^{\mathcal{F}_{\mathsf{NICOM}}^{S_C}}(x, w, st, c) : \mathcal{A}^{\mathcal{F}_{\mathsf{NICOM}}^{S_C}}(a, c, z, r) = 1\right]\Big| \leq \mathsf{neg}(\kappa)$$

where $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\kappa)$.

4.3 Our NIZK Compiler in the $\mathcal{F}_{\mathsf{NICOM}}$ model

We apply the Fiat-Shamir transform on the Sigma protocol using correlation intractable hash functions H to remove interaction and obtain our NIZK protocol. The CI hash function is provided with the description of the $\mathcal{S}_C$ algorithm to extract the prover's view and compute the bad challenge function. Our compiler can be found in Figure. 4.

A corrupt prover breaks soundness of the protocol if it breaks the special soundness of the adaptively secure Sigma protocol or it breaks the binding property of the commitment scheme. In the former case, the witness can be extracted by invoking the witness extractor algorithm Ext (according Def. 11) of the Sigma protocol on the proof. We show that our NIZK protocol Π_{NIZK} implements $\mathcal{F}_{\mathsf{NIZK}}$ functionality against adaptive corruption of parties by proving Thm. 3 in the full version [CSW20c]. It can be further shown that the same protocol implements the single prover multi-proof NIZK functionality $\mathcal{F}_{\mathsf{NIZK}}^{\mathsf{m}}$.

Theorem 3. *If* $\mathcal{H}$ *is a somewhere statistically correlation intractable hash function family with programmability,* $\Sigma = (\mathsf{Setup}, P_1, V_1, P_2, V_2)$ *is an adaptively secure Sigma protocol (in the* $\mathcal{F}_{\mathsf{NICOM}}$ *model) with computational special soundness then* Π_{NIZK} *implements* $\mathcal{F}_{\mathsf{NIZK}}$ *functionality in* $(\mathsf{crs}_{\mathsf{NIZK}}, \mathcal{F}_{\mathsf{NICOM}})$ *model against adaptive corruption of parties.*

Adaptive Soundness and Adaptive Zero knowledge. The NIZK protocol can be made triply adaptive secure by adding adaptive soundness and adaptive zero-knowledge. The NIZK protocol satisfies adaptive soundness if the underlying Sigma protocol satisfies adaptive soundness and $\mathcal{F}_{\mathsf{NICOM}}$ is implemented using a non-interactive UC-commitment Com in the non-programmable $\mathsf{crs}_{\mathsf{Com}}$ model Com, whose real and ideal world $\mathsf{crs}_{\mathsf{Com}}$ distribution are identical. Moreover, the NIZK protocol satisfies adaptive zero knowledge if the underlying Sigma protocol satisfies adaptive zero knowledge and Com is a non-interactive adaptively secure commitment in the non-programmable crs model. This is summarized in Thm. 4 and proven in the full version [CSW20c].

Fig. 4. Adaptively Secure NIZK Protocol Π_{NIZK}

- **Primitives:** Adaptively-secure Sigma Protocol $\Sigma = (\mathsf{Setup}, \mathsf{P}_1, \mathsf{V}_1, \mathsf{P}_2, \mathsf{V}_2)$, that uses functionality $\mathcal{F}_{\mathsf{NICOM}}$ (with algorithm $\mathcal{S}_C$). Correlation Intractable hash function family $\mathcal{H} = (\mathsf{Gen}, \mathsf{StatGen}, H)$.
- **Public Inputs:** Setup string $\mathsf{crs}_{\mathsf{NIZK}} = (\mathsf{k}, \mathsf{crs}_{\Sigma})$ where $(\mathsf{crs}_{\Sigma}, \mathsf{td}_{\Sigma}) \leftarrow \Sigma.\mathsf{Setup}(1^{\kappa})$ and $\mathsf{k} \leftarrow \mathcal{H}.\mathsf{StatGen}(1^{\kappa}, C_{\mathsf{sk}})$ where $\mathsf{sk} = (\mathsf{td}_{\Sigma}, \mathcal{S}_C)$. [a] The Sigma protocol is repeated for $\tau = \mathcal{O}(\kappa)$ times.
- **Private Inputs:** V has input statement x. P has the same input statement x and secret witness w such that $\mathcal{R}(x, w) = 1$.

$\mathsf{P}(\mathsf{crs}_{\mathsf{NIZK}}, x, w, 1^{\kappa})$:
Upon invoked with command $(\mathsf{prove}, \mathsf{sid}, x, w)$ the prover performs the following for $j \in [\tau]$:

- $(a^j, \mathsf{st}^j_{\Sigma}) \leftarrow \Sigma.\mathsf{P}_1(\mathsf{crs}_{\Sigma}, x, w, 1^{\kappa})$.
- Sample $c^j_0, c^j_1 \leftarrow_R \mathcal{C}$ such that $c^j_0 \neq c^j_1$. Commit to challenges as $(\mathsf{Receipt}, C^j, \mathsf{st}^j_C) \leftarrow \mathcal{F}_{\mathsf{NICOM}}(\mathsf{Com}, 3j+2, \mathsf{P}, (c^j_0, c^j_1))$.
- For $\delta \in \{0, 1\}$, the prover performs the following:
  - Compute $z^j_{\delta} \leftarrow \Sigma.\mathsf{P}_2(\mathsf{crs}_{\Sigma}, x, w, \mathsf{st}^j_{\Sigma}, c^j_{\delta})$.
  - Commit to the responses as follows: $(\mathsf{Receipt}, 3j+\delta, \mathsf{P}, Z^j_{\delta}, \mathsf{st}^j_{Z,\delta}) \leftarrow \mathcal{F}_{\mathsf{NICOM}}(\mathsf{Com}, 3j+\delta, \mathsf{P}, z^j_{\delta})$.
- The commitments for the jth run are denoted as $Y^j = (C^j, Z^j_0, Z^j_1)$.

Assemble the commitments as $\mathbf{Y} = \{Y^j\}_{j\in[\tau]}$ and and the first messages as $\mathbf{a} = \{a^j\}_{j\in[\tau]}$. Compute the challenge as $\mathbf{e} = \{e^j\}_{j\in[\tau]} = H(\mathsf{k}, (\mathbf{a}, \mathbf{Y}))$. The prover performs the following for $j \in [\tau]$:

- Set the challenge as $\delta = e^j \in \{0, 1\}$.
- Construct the response as $U^j = (c^j_0, c^j_1, z^j_{\delta}, \mathsf{st}^j_C, \mathsf{st}^j_{Z,\delta})$ by decommitting to the challenges and the response z^j_{δ}.

The prover sends the NIZK proof $\gamma = (\mathbf{a}, \mathbf{Y}, \mathbf{U})$ where $\mathbf{U} = \{U^j\}_{j\in[\tau]}$ to the verifier.
$\mathsf{V}(\mathsf{crs}_{\mathsf{NIZK}}, x, \gamma)$:
Upon invoked with command $(\mathsf{verify}, \mathsf{sid}, x, \gamma)$ the verifier performs the following:

- Parse the proof $\gamma = (\mathbf{a}, \mathbf{Y}, \mathbf{U}) = \{a^j, Y^j, U^j\}_{j\in[\tau]}$.
- Compute the challenge string as $\mathbf{e} = \{e^j\}_{j\in[\tau]} = \mathcal{H}.H(\mathsf{k}, (\mathbf{a}, \mathbf{Y}))$ where $\mathbf{e} \in \{0, 1\}^{\tau}$.
- For $j \in [\tau]$, the verifier performs the following :
  - The verifier sets $\delta = e^j \in \{0, 1\}$.
  - Parse the proof as $Y^j = (C^j, Z^j_0, Z^j_1)$ and $U^j = (c^j_0, c^j_1, z^j, \mathsf{st}^j_Z, \mathsf{st}^j_C)$.
  - Verifies the provided decommitments and proofs. Output $(\mathsf{verification}, \mathsf{sid}, x, \gamma, 0)$ if any of the following occurs:
    1. If $\mathcal{F}_{\mathsf{NICOM}}(\mathsf{Open}, 3j+2, \mathsf{P}, (c^j_0, c^j_1), C^j, \mathsf{st}^j_C)$ returns $\mathsf{verification\text{-}failed}$.
    2. If $\mathcal{F}_{\mathsf{NICOM}}(\mathsf{Open}, 3j+\delta, \mathsf{P}, z^j, Z^j_{\delta}, \mathsf{st}^j_Z)$ returns $\mathsf{verification\text{-}failed}$.
    3. If $\Sigma.\mathsf{V}_2(\mathsf{crs}_{\Sigma}, x, a^j, c^j_{\delta}, z^j) = 0$.

The verifier outputs $(\mathsf{verification}, \mathsf{sid}, x, \gamma, 1)$ if all the above τ proofs verified correctly and the above decommitments were correct.

---

[a] $\mathcal{C}_{\mathsf{sk}}$ is a poly-size circuit computing the function $f_{\mathsf{sk}}(\mathbf{a}, \mathbf{Y}) = \mathbf{e}$, such that for every $j \in [\tau], e^j = 0$ iff $\Sigma.\mathsf{V}_2(\mathsf{crs}_{\Sigma}, x, a^j, c^j_0, z^j) = 1$ where $(c^j_0, c^j_1) \leftarrow \mathcal{F}_{\mathsf{NICOM}}(\mathcal{S}_C, \mathsf{Ext}, 3j+2, \mathsf{P}, C^j), z^j \leftarrow \mathcal{F}_{\mathsf{NICOM}}(\mathcal{S}_C, \mathsf{Ext}, 3j, \mathsf{P}, Z^j_0)$.

Theorem 4. *If $\mathcal{H}$ is a somewhere statistically correlation intractable hash function family, $\Sigma = (\mathsf{Setup}, P_1, V_1, P_2, V_2)$ is a Sigma protocol satisfying adaptive special soundness and adaptive zero knowledge, and $\mathcal{F}_{\mathsf{NICOM}}$ is implemented using an adaptively secure UC commitment in the non-programmable $\mathsf{crs}_{\mathsf{Com}}$ model then Π_{NIZK} satisfies adaptive soundness and adaptive zero knowledge in the $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{crs}_{\mathsf{Com}})$ model.*

Instantiations. The adaptively Sigma protocol can be instantiated using the Schnorr's protocol, Sigma protocol of FLS, Blum's Hamiltonicity protocol or the GC-based protocol of [HV16]. Detailed overview can be found in Sec. 2.4. The CI hash function can be instantiated from LWE [PS19, CCH+19], or from DDH+LPN assumption [BKM20]. This is discussed in the full version [CSW20c]. $\mathcal{F}_{\mathsf{NICOM}}$ is constructed from the UC-commitment scheme of [CF01] in the full version [CSW20c] by relying on equivocal commitments and CCA-2 secure public key encryption with oblivious sampleability. The equivocal commitment can be instantiated from Pedersen Commitment and the obliviously sampleable encryption scheme can be instantiated from Cramer Shoup encryption [CS98], yielding a protocol from DDH. Similarly, we can instantiate the equivocal commitment from LWE [CsW19] and the obliviously sampleable encryption scheme from LWE [MP12].

5 Triply Adaptive NIZK Argument in the short crs model

In this section we present our compiler Π_{sNIZK} which obtains a triply adaptive NIZK protocol where the crs size is independent of the circuit size and depends only on κ assuming a non-interactive equivocal commitment scheme in the reusable crs model which supports additive homomorphism, PKE with oblivious ciphertext sampleability and a triply adaptively secure NIZK protocol Π_{NIZK} in the crs model. Our compiler is presented in the full version [CSW20c]. We prove triple adaptive security of Π_{sNIZK} by proving Thm. 5 in the full version [CSW20c]. By applying this result, we reduce the crs size of Π_{NIZK}. The homomorphic commitment scheme can be instantiated from DDH (Pedersen commitment or [CSW20b]) or LWE [GVW15] asusmptions. The PKE can be instantiated from DDH assumption (Elgamal encryption) or LWE [GSW13] assumptions. This yields our compiler from DDH or LWE assumption.

Theorem 5. *Assuming* PKE *is a public key encryption scheme with oblivious ciphertext sampling, Com is an equivocal additively homomorphic commitment scheme in the reusable $\mathsf{crs}_{\mathsf{Com}}$ model and Π_{NIZK} implements $\mathcal{F}^m_{\mathsf{NIZK}}$ against adaptive corruption of parties, then Π_{sNIZK} UC-securely implements $\mathcal{F}_{\mathsf{NIZK}}$ functionality for NP languages against adaptive adversaries in the crs model where $|\mathsf{crs}| = poly(\kappa)$. Π_{sNIZK} is also adaptively sound and adaptively zero knowledge.*

6 Triply Adaptive, multi-proof UC-NIZK Argument

In this section, we add non-malleability to our Π_{sNIZK} protocol to obtain our multi-proof UC-NIZK protocol $\Pi_{\mathsf{UC\text{-}NIZK}}$ by using simulation sound tag-based

commitments $\mathsf{Com}_{\mathsf{SST}}$ and strong one-time signature scheme SIG. We add non-malleability to our proof by signing the proof using a pair of keys $(\mathsf{vk}, \mathsf{sk})$ from SIG and committing the witness using a $\mathsf{Com}_{\mathsf{SST}}$ where the tag is $(\mathsf{vk}, \mathsf{sid}, \mathsf{ssid}, x)$. The adversary is bound to vk since vk is part of the tag used to encrypt w using $\mathsf{Com}_{\mathsf{SST}}$ in the proof γ. SIG ensures that an adversary cannot forge a signature using vk and this prevents non-malleability. The same crs is used for multiple subsessions and this ensures adaptive soundness and adaptive zero knowledge. The protocol and the proofs can be found in the full version [CSW20c]. Security of $\Pi_{\mathsf{UC\text{-}NIZK}}$ is summarized in Thm. 6.

Theorem 6. *If Π_{sNIZK} UC-realizes $\mathcal{F}_{\mathsf{NIZK}}$ for a single proof,* SIG *is a strong one-time secure signature scheme, $\mathsf{Com}_{\mathsf{SST}}$ is a tag-based simulation-sound trapdoor commitment and* PKE *is a public key encryption scheme with oblivious ciphertext sampling property then $\Pi_{\mathsf{UC\text{-}NIZK}}$ UC-securely implements $\mathcal{F}_{\mathsf{NIZK}}$ for multiple instances against adaptive adversaries. In addition, $\Pi_{\mathsf{UC\text{-}NIZK}}$ is adaptively sound and adaptively zero knowledge.*

References

AF07. Masayuki Abe and Serge Fehr. Perfect NIZK with adaptive soundness. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 118–136. Springer, Heidelberg, February 2007.

AMPS21. Navid Alamati, Hart Montgomery, Sikhar Patranabis, and Pratik Sarkar. Two-round adaptively secure MPC from isogenies, lpn, or CDH. In *ASIACRYPT 2021*, volume 13091 of *LNCS*, pages 305–334. Springer, 2021.

BCG+14. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474, 2014.

BFM90. Manuel Blum, Paul Feldman, and Silvio Micali. Proving security against chosen cyphertext attacks. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 256–268. Springer, Heidelberg, August 1990.

BKM06. Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 60–79. Springer, Heidelberg, March 2006.

BKM20. Zvika Brakerski, Venkata Koppula, and Tamer Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2020, Part III*, LNCS, pages 738–767. Springer, Heidelberg, August 2020.

Blu86. Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, 1986.

BMW03. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003.

BSMP91. Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

CCH+19. Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.

CD00. Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 331–345. Springer, Heidelberg, December 2000.

CF01. Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Heidelberg, August 2001.

CGH98. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

CGPS21. Suvradip Chakraborty, Chaya Ganesh, Mahak Pancholi, and Pratik Sarkar. Reverse firewalls for adaptively secure MPC without setup. In *ASIACRYPT 2021*, volume 13091 of *LNCS*, pages 335–364. Springer, 2021.

CJJ21. Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for NP from standard assumptions. In *CRYPTO 2021*, volume 12828 of *LNCS*, pages 394–423. Springer, 2021.

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

CPS+16. Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Online/offline OR composition of sigma protocols. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 63–92. Springer, Heidelberg, May 2016.

CPV20. Michele Ciampi, Roberto Parisella, and Daniele Venturi. On adaptive security of delayed-input sigma protocols and fiat-shamir nizks. In *SCN 2020*, pages 670–690, 2020.

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Heidelberg, August 1998.

CsW19. Ran Cohen, abhi shelat, and Daniel Wichs. Adaptively secure MPC with sublinear communication complexity. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 30–60. Springer, Heidelberg, August 2019.

CSW20a. Ran Canetti, Pratik Sarkar, and Xiao Wang. Efficient and round-optimal oblivious transfer and commitment with adaptive security. In *ASIACRYPT 2020, Part III*, LNCS, pages 277–308. Springer, Heidelberg, December 2020.

CSW20b. Ran Canetti, Pratik Sarkar, and Xiao Wang. Efficient and round-optimal oblivious transfer and commitment with adaptive security. In *ASIACRYPT 2020*, volume 12493 of *LNCS*, pages 277–308. Springer, 2020.

CSW20c. Ran Canetti, Pratik Sarkar, and Xiao Wang. Triply adaptive UC NIZK. *IACR Cryptology ePrint Archive*, page 1212, 2020. https://eprint.iacr.org/2020/1212.

DDN91. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991.

DN00. Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 432–450. Springer, Heidelberg, August 2000.

FLS99. Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, 1999.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

GGI+15. Craig Gentry, Jens Groth, Yuval Ishai, Chris Peikert, Amit Sahai, and Adam D. Smith. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. *Journal of Cryptology*, 28(4):820–843, October 2015.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

GOS06. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006.

GOS12. Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012.

GR13. Oded Goldreich and Ron D. Rothblum. Enhancements of trapdoor permutations. *Journal of Cryptology*, 26(3):484–512, July 2013.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013.

GVW15. Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 469–477. ACM Press, June 2015.

HL18. Justin Holmgren and Alex Lombardi. Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 850–858. IEEE Computer Society, 2018.

HLR21. Justin Holmgren, Alex Lombardi, and Ron D. Rothblum. Fiat-shamir via list-recoverable codes (or: parallel repetition of gmw is not zero-knowledge). In *STOC 2021*, pages 750–760, 2021.

HV16. Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On the power of secure two-party computation. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 397–429. Springer, Heidelberg, August 2016.

KKK21. Thomas Kerber, Aggelos Kiayias, and Markulf Kohlweiss. Composition with knowledge assumptions. In *CRYPTO 2021*, volume 12828 of *LNCS*, pages 364–393. Springer, 2021.

KNYY19. Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Exploring constructions of compact NIZKs from various assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 639–669. Springer, Heidelberg, August 2019.

KNYY20. Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Compact NIZKs from standard assumptions on bilinear maps. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, LNCS, pages 379–409. Springer, Heidelberg, May 2020.

KZM+15. Ahmed E. Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, T.-H. Hubert Chan, Charalampos Papamanthou, Rafael Pass, Abhi Shelat, and Elaine Shi. How to use snarks in universally composable protocols. *IACR Cryptol. ePrint Arch.*, 2015:1093, 2015.

LP09. Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

PS19. Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114. Springer, Heidelberg, August 2019.

Sch90. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

SCO+01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *CRYPTO 2001*, pages 566–598, 2001.