Triply Adaptive UC NIZK

Ran Canetti*
Boston University

Pratik Sarkar
Boston University

Xiao Wang
Northwestern University

Abstract

The only known non-interactive zero-knowledge (NIZK) protocol that is secure against adaptive corruption of the prover is based on that of Groth-Ostrovsky-Sahai (JACM'11) (GOS). However that protocol does not guarantee full adaptive soundness. Abe and Fehr (TCC'07) construct an adaptively sound variant of the GOS protocol under a knowledge-of-exponent assumption, but knowledge assumptions of this type are inherently incompatible with universally composable (UC) security.

We show the first NIZK which is triply adaptive: it is a UC NIZK protocol in a multi-party, multi-instance setting, with adaptive corruptions and no data erasures. Furthermore, the protocol provides full adaptive soundness. Our construction is very different than that of GOS: it is based on the recent NIZK of Canetti et al (STOC'19), and can be based on a variety of assumptions (e.g. LWE, or LPN and DDH). We also show how to get a succinct reference string assuming LWE or DDH from GOS-like techniques.

*Member of the CPIIS. Supported by NSF Awards 1931714, 1801564, 1414119, and the DARPA SIEVE program.

Contents

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. Furthermore, proofs are written once and for all, to be inspected and verified by anyone at any time.

Much work was done on this central concept. A first thrust provides basic formulations of soundness and zero knowledge the presence of a reference string, and constructions that satisfy them under standard assumptions [BSMP91, FLS99, GR13]. Indeed, even basic soundness and zero knowledge 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 considers a multi-party setting and addresses malleability attacks [SCO+01, DDN91], and universally composable (UC) security [CLOS02] 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].

However, one challenge has remained: Can we have NIZK protocols that are secure in a multi-party setting where the adversary can adaptively corrupt parties? Here the traditional definition is extended so as to guarantee that the attacker does not gain any advantage towards breaking the security of the overall system, beyond to the ideal case where the NIZK is replaced by a trusted party, even when corrupting a prover after the proof was sent, and obtaining the hidden internal state of the prover[1].

The first protocol that provides security in such a setting is that of Groth Ostrovsky and Sahai [GOS06] (GOS). That protocol is also UC secure, even in a multi-proof, multi-party setting. This implies non-malleability, as well as a weak form of adaptive soundness (essentially it is guaranteed that the sequence of instances proven to be in L in an execution of the protocol is indistinguishable from a sequence of instances that are actually in L, even given the reference string). However it does not guarantee full soundness, unless all instances to be proven are fixed ahead of time before the reference string is known, to avoid dependency of the instance on the reference string. The works of [KNYY19, KNYY20] have similar characteristics: they provide security against adaptive corruptions, but only weak 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[2]. However, the [AF07] analysis is inherently incompatible [KZM+15] with UC security. Indeed, KOE-style assumptions require existence of a knowledge extractor that has full access to the code of the environment, and furthermore is larger than the same. In contrast, in the UC framework a single extractor/simulator would have to handle arbitrary polytime environments.

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, then under

[1] In cases where the prover can erase 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

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

what assumptions?

1.1 Our Contributions

We obtain the *first* UC-secure NIZK protocol that is triply adaptive in the crs model. The crs can be reused for multiple NIZK instances between different set of parties. Our result can be obtained either from Learning With Errors (LWE) assumption or Decisional Diffie Hellman (DDH) and Learning Parity with Noise (LPN) assumption. Our main result is summarized below.

Theorem 1.1. *(Informal) Assuming LWE assumption holds or DDH and LPN assumptions hold, there exists a multi-theorem NIZK protocol that UC-securely implements the NIZK functionality(Fig. 1) 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 transforms a NIZK protocol in the crs model (where the crs can depend on the NP relation) to the short crs model (where the length of the crs depends only on the security parameter) while preserving triple adaptive security. It can be constructed from DDH or LWE assumptions. Previous such compilers [GGI+15, CsW19] were known only from LWE. Compiling our above protocol we obtain a triply adaptive UC-NIZK in the short crs model.

Theorem 1.2. *(Informal) Assuming LWE assumption holds or DDH and LPN assumptions hold, there exists a multi-theorem NIZK protocol that UC-securely implements the NIZK functionality(Fig. 1) against adaptive corruptions in the short crs model (where $|\mathsf{crs}| = poly(\kappa)$ and κ is the computational security parameter) for multiple instances. Furthermore, it is adaptively sound and adaptively zero knowledge.*

1.2 Our Techniques and Instantiations

Next, we discuss our protocols and their instantiations from various assumptions in more details.

– We demonstrate that the Sigma protocol Π_{ZK} of [FLS99] (FLS) can be modified to obtain a three round protocol Π_{ZK}. The protocol Π_{ZK} runs $\mathcal{O}(\kappa)$ parallel iterations of the modified FLS protocol and UC-realizes the zero knowledge (ZK) functionality (Fig. 3) for a single session in the crs model, given any non-interactive equivocal commitment scheme and a public key encryption (PKE) scheme with oblivious ciphertext sampleability. Moreover, it is triply adaptive in nature. We also show that the crs is multi-proof, i.e. prover can construct multiple proofs for different statements with the same crs as long as the roles of the parties are preserved. The equivocal commitment can be instantiated from DLP [Ped92], Decisional Diffie Hellman [CSW20], LWE [GVW15] and many other assumptions [Oka93, FF02, GVW15]. The PKE can be instantiated from DDH assumption [ElG85] or LWE [GSW13] assumptions. Thus, our triply adaptive ZK protocol can be instantiated solely based on DDH or LWE. Our protocol also satisfies delayed-input property - only the last message of the prover depends on the statement and the witness. This property allows Π_{ZK} to substitute the usage of triply adaptive NIZKs in the online phase of protocols (in the offline-online paradigm). The prover and the verifier can run the first two rounds of Π_{ZK} in the offline phase without the knowledge of the statement. In the online phase the prover receives the statement and the witness. He computes the third message and sends the proof. This serves the purpose of a NIZK in the online phase by relying on DDH assumption; whereas NIZK from DDH is not known.

– In Π_{ZK} the verifier algorithm is public-coin, i.e. the verifier's message consists of publicly sampled random coins. It allows us to apply the Fiat-Shamir heuristic [FS87] using correlation intractable hash [CGH98] functions. This yields the *first* triply adaptive UC-NIZK (for a single session) protocol Π_{NIZK} without relying on knowledge assumptions. Moreover, the setup string is multi-proof and can be used by the prover to prove multiple statements. The hash function can be instantiated from LWE [PS19] or circular LWE [CCH$^+$19]. A recent work by [BKM20] introduced the notion of CI for approximable relations (CI-Approx) and showed that CI-Approx for constant-degree polynomials is sufficient for NIZK, provided the underlying public-coin ZK protocol is implemented using a PKE where decryption can be performed using a constant-degree polynomial. They construct CI-Approx functions from enhanced trapdoor hash functions [DGI$^+$19] based on DDH, Quadratic Residuosity (QR), Decisional composite residuosity (DCR) and LWE, and implement their PKE using Learning Parity with Noise (LPN) assumption (the LPN-based PKE scheme also satisfies oblivious ciphertext sampling). Our results hold if we replace our CI hash function with a CI-Approx hash. This yields our NIZK protocol from LWE or DDH and LPN assumptions.

– 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 NIZK argument Π with a long multi-proof crs, i.e. $|\mathsf{crs}| = \mathrm{poly}(\kappa, |\mathcal{C}|)$ to obtain a NIZK argument Π_{s} with a short multi-proof common reference string scrs, where $|\mathsf{scrs}| = \mathrm{poly}(\kappa)$, $\mathcal{C}$ is the NP verification circuit and κ is the computational security parameter. Moreover, Π_{s} is triply adaptive if Π is triply adaptive. By applying the compiler on our NIZK argument Π_{NIZK} (or $\Pi_{\mathsf{UC\text{-}NIZK}}$) we obtain a NIZK argument Π_{sNIZK} in the short crs model. Such homomorphic commitments can be instantiated from various assumptions like DDH [Ped92, CSW20] and LWE/SIS [GVW15] and other assumptions[Oka93, FF02, GVW15]. The PKE can be instantiated from DDH assumption [ElG85] or LWE [GSW13] assumptions. Our compiler works based on DDH or LWE assumption. Compiling Π_{NIZK} we obtain a triply adaptive NIZK Π_{sNIZK} in the short crs model from LWE or DDH and LPN assumptions.

– We add non-malleability to our NIZK argument Π_{sNIZK} using a tag-based simulation-sound trapdoor commitment scheme and a strong one-time signature scheme to obtain the multi-session UC-secure NIZK $\Pi_{\mathsf{UC\text{-}NIZK}}$ in the short crs model. The tag-based commitment can be instantiated from UC-commitments - DDH [CSW20] and LWE [DPR16] and various other assumptions [MY04, GMY06, MY04, CF01]. Strong one-time signatures can be constructed [Rom90] from one-way functions. We also prove that our protocol also satisfies adaptive zero knowledge and provides adaptive soundness. Our protocol $\Pi_{\mathsf{UC\text{-}NIZK}}$ is triply adaptive under LWE or DDH and LPN assumptions.

1.3 Related Works

The works of [GOS06, KNYY19, KNYY20] construct NIZKs which are secure against adaptive corruptions but they lack adaptive soundness. The works of [CCH$^+$19, BKM20] construct statically secure NIZKs which attain adaptive soundness and adaptive ZK. A concurrent work by [CPV20] compiled delayed input Sigma protocol into a Sigma protocol which satisfies adaptive zero knowledge. Upon applying the result of [CPS$^+$16b] 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 version of FLS in Section 4. Then, we make it non-interactive to obtain our triply adaptively-secure NIZK protocol in Section 5 using CI hash functions. We present our compiler to reduce the crs length in Section 6. Finally, we conclude with our multi-session UC-NIZK protocol in the short crs model in Section 7.

Note. 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 protocol. First, we show that the Sigma protocol of FLS can be modified to obtain a public-coin protocol implementing $\mathcal{F}_{\mathsf{ZK}}$ (Fig. 3) functionality using equivocal commitment and PKE with oblivious ciphertext sampling. Then, we make it non-interactive by applying the Fiat-Shamir transform using correlation intractable hash functions. Assuming homomorphic equivocal commitments we reduce the crs size to $\mathrm{poly}(\kappa)$. All our protocols are triply adaptive and single session UC-secure. Finally, we make it UC-secure for multi-sessions by adding non-malleability.

2.1 ZK Protocol of FLS

We briefly recall the ZK protocol of FLS for the sake of completeness. Let $\mathcal{R}_{\mathsf{Ham}}$ be the set of Hamiltonian graphs. The prover $\mathcal{P}$ 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. $\mathcal{P}$ samples a random n-node cycle H and commits to the adjacency matrix of the cycle. The matrix contains n^2 entries, and $\mathcal{P}$ commits to the edges as $\mathsf{Com}(1)$, and non-edges as $\mathsf{Com}(0)$. The prover sends these commitments to the verifier $\mathcal{V}$. $\mathcal{V}$ samples a random challenge bit e and sends it to the prover. If $e = 0$, then $\mathcal{P}$ 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. $\mathcal{P}$ sends these decommitments as its response z. Upon obtaining z, the verifier performs the following based on e:

- $e = 0$: 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.
- $e = 1$: Verify that z contains decommitments to 0, and the decommitted edges correspond to non-edges in $\pi(G)$.

Soundness and Proof of Knowledge. When $G \notin \mathcal{R}_{\mathsf{Ham}}$, a corrupt prover gets caught with probability $\frac{1}{2}$ unless he breaks the binding property of the commitment scheme. The soundness error can be further reduced to $2^{-\kappa}$ with κ parallel repetitions. When $G \in \mathcal{R}_{\mathsf{Ham}}$ and a corrupt prover computes an accepting transcript (a, e, z) then a witness cycle σ can be extracted. They rely on rewinding the prover as they are in the plain model. Extraction is performed by rewinding the prover to generate a transcript - (a, e', z') on a different challenge $e' = 1 - e$. Given z and z' the extractor obtains (H, π) and computes $\sigma = \pi^{-1}(H)$.

Zero Knowledge. The protocol only achieves zero knowledge against an honest verifier. The ZK simulator samples a random challenge e and based on that he computes (a, z) as follows.

- $e = 0$: $\mathcal{P}$ samples a random n-node cycle H and commits to the adjacency matrix of the cycle as a. He sets z as the decommitment to the cycle.

- $e = 1$: $\mathcal{P}$ sets all the commitments to 0, i.e. commits to an null graph. It computes a random permutation π and decommits to the non-edges in $\pi(G)$.

Let use 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.

Static Security. The protocol provides only security against static corruption of parties. Suppose the ZK simulator constructs a simulated proof where $e = 1$ and the prover gets corrupted post-execution. In such a case, the simulator obtains a valid witness cycle σ and he has to open the unopened commitments (which are commitments to 0) s.t. they contain a cycle. Since the commitments in the original FLS protocol were not equivocal in nature, the simulator fails to equivocate the commitments; thus providing only static security.

Delayed-input property. The first message of the prover is computed based on the parameter n without the knowledge of the graph or the witness. After obtaining e from $\mathcal{V}$, 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.

2.2 Our ZK Protocol

Next, we consider $\ell = \mathcal{O}(8\mu, \kappa)$ (where μ is the statistical security parameter) parallel invocations of the above protocol and we incrementally modify it to obtain our UC-secure ZK protocol $\Pi_{\mathsf{ZK}} = (\mathbf{a}, \mathbf{e}, \mathbf{z}) = (\{a^1, \ldots, a^\ell\}, \{e^1, \ldots, e^\ell\}, \{z^1, \ldots, z^\ell\})$ in the crs model.

Soundness and Proof of Knowledge. We remove dependency on rewinding the prover (for witness extraction) by assuming a PKE where the public key pk is provided in the crs. Recall that the prover's view in the first round consists of an adjacency matrix, i.e. n^2 bits $\{b_j\}_{j\in[n^2]}$, of an n-node cycle where n of the bit entries are 1. For each commitment $c_j = \mathsf{Com}(b_j; r_j)$ (where r_j is the commitment randomness) to bit b_j in the prover's first message a^i (for the ith run of the protocol), the prover also encrypts r_j as $E_j = \mathsf{Enc}(\mathsf{pk}, r_j; s_j)$ using randomness s_j. When the prover decommits to c_j as (b_j, r_j) in the third message z^i he also provides s_j as a valid opening of E_j to r_j. The verifier accepts the proof if both c_j and E_j were correctly opened. Soundness follows similarly to the FLS protocol by relying on the binding property of the commitment scheme. Meanwhile, a corrupt prover's witness can be extracted from the proof by a simulator $\mathcal{S}$. The simulator possesses the secret key sk corresponding to pk. He decrypts E_j to obtain r_j. $\mathcal{S}$ reconstructs c_j by brute forcing over $b_j \in \{0, 1\}$ and using r_j as the commitment randomness. This allows $\mathcal{S}$ to reconstruct the adjacency matrix of the corrupt prover from $(b_1, \ldots, b_{n^2})$ values; hence recovering the witness. The exact witness extraction process is a bit more involved and it requires running the protocol $\ell = \mathcal{O}(8\mu, \kappa)$ times instead of κ times.

Zero Knowledge. The FLS protocol guarantees ZK when the challenge is randomly sampled by an honest verifier. However, in our protocol the verifier can be statically corrupt. In such a case, the ZK simulator of FLS fails. To tackle this issue, we assume the commitments are equivocal in nature and the public parameter of the commitment is provided in the crs. The ZK simulator of Π_{ZK} possesses the trapdoor of the commitment. It allows him to construct the commitments in the equivocal mode for the first message a^i. After obtaining the adversarially generated challenge e^i the simulator can equivocate the commitments s.t. they are consistent with the challenge and the statement graph. However, we also need to equivocate the encryptions of the commitment randomness, i.e. E_j. The encryptions cannot be equivocated since they are perfectly binding due to the correctness of the PKE scheme. We solve this problem by using a PKE that allows obliviously sampling a ciphertext from the ciphertext space. The obliviously sampled ciphertext is indistinguishable from a real ciphertext. Using such a PKE, the prover sends two encryptions $E_{j,0}$ and $E_{j,1}$ corresponding to each $c_j = \mathsf{Com}(b_j; r_j)$ where $E_{i,b_j} = \mathsf{Enc}(\mathsf{pk}, r_j; s_j)$ and $E_{i,\bar{b}_j}$ ($\bar{b}_j = 1 - b_j$) is sampled obliviously. When the prover opens c_j to b_j he also provides (r_j, s_j) and claims that $E_{i,\bar{b}_j}$ was sampled obliviously. The verifier checks that the commitment c_j and the corresponding encryption E_{j,b_j} was computed correctly using the commitment randomness r_j and the encryption randomness s_j respectively. This is the complete description of Π_{ZK} and we summarize it later in this section. Now, we can prove that Π_{ZK} provides ZK. A ZK simulator can compute two openings of c_j as $(0, r_j)$ and $(1, r'_j)$. He encrypts both openings $E_{j,0} = \mathsf{Enc}(\mathsf{pk}, r_j; s_j)$ and $E_{j,1} = \mathsf{Enc}(\mathsf{pk}, r'_j; s'_j)$. For opening c_j to a bit, suppose 0, the simulator sends the opening as $(0, r_j, s_j)$ and claims that $E_{j,1}$ was obliviously sampled. The encryption of 1 (in the simulated proof) is indistinguishable from obliviously sampled ciphertext (in the real ZK proof) due to the oblivious sampling property of PKE. Thus, zero knowledge follows from the equivocal property of the commitment and the oblivious ciphertext sampling property of the PKE.

Adaptive Security. We only consider the case where the prover is honest and the verifier is statically corrupt (the other adaptive corruption cases are similar to the static corruption cases). The ZK simulator constructs a simulated proof γ as described in the previous paragraph. Recall that the commitments c_j were constructed in the equivocal mode and the corresponding randomness r_j (resp. r'_j) for bit 0 (resp. 1) are encrypted in $E_{j,0}$ (resp. $E_{j,1}$). Later, the prover gets post-execution corrupted and the simulator obtains a valid cycle σ. The simulator needs to provide prover's randomness s.t. the *unopened commitments are consistent with* σ. The simulator can open any unopened commitment c_j to bit b, suppose 0, by sending the opening as $(0, r_j, s_j)$ and claim that $E_{j,1}$ was obliviously sampled. An adaptive adversary cannot distinguish between a real prover's view and a simulated prover's view due to the equivocal property of the commitment scheme and the oblivious ciphertext sampling property of PKE.

Delayed-input Adaptive Soundness. The FLS protocol supports input-delayed property, i.e. only the last message of the prover depends on the statement and the witness. This allows the prover to adaptively choose his statement in the last round of the protocol based on the first and second round messages. We also allow the prover to adaptively choose his statement based on the crs distribution. This does not hamper soundness since the crs distribution in real and ideal worlds are identical and the simulator is provided with the trapdoor of the crs. If a corrupt prover breaks adaptive soundness of the protocol then he breaks the binding property of the commitment scheme or he correctly guesses the e string, which happens with negligible probability. The adversary for the commitment obtains the $\mathsf{crs}_{\mathsf{Com}}$ for the commitment and he samples a public key pk to set the $\mathsf{crs}_{\mathsf{ZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$. He invokes the adversary for adaptive soundness to obtain a proof. The

adversary for the commitment scheme can check if in any iteration $\exists i \in [\ell], \exists j \in [n^2]$, s.t. in the ith iteration the commitment c_j opens both to 0 and 1 by decrypting r_j and r'_j from $E_{j,0}$ and $E_{j,1}$ given the secret key sk. He returns r_j and r'_j to the challenger of Com to break the binding property.

Delayed-input Adaptive Zero Knowledge. Our protocol also satisfies delayed-input adaptive zero knowledge since the ZK simulator (presented above) works when the statement to be proven is chosen adaptively (by a statically corrupt verifier) in the last round based on the crs distribution and the first two messages.

Final Protocol. For the sake of completeness we describe our final protocol Π_{ZK}. Π_{ZK} consists of $\ell = \mathcal{O}(8\mu, \kappa)$ parallel repetitions of the following protocol. The crs consists of a public key pk and $\mathsf{crs}_{\mathsf{Com}}$, i.e. the crs of the equivocal commitment scheme. $\mathcal{P}$ samples a random n-node cycle H and commits to the adjacency matrix of the cycle. The matrix contains n^2 binary entries, and $\mathcal{P}$ commits to the edges as $\mathsf{Com}(1; r)$, and non-edges as $\mathsf{Com}(0; r)$. For each commitment $c_j = \mathsf{Com}(b_j; r_j)$ ($j \in [n^2]$) the prover also encrypts the randomness as $E_{j,b_j} = \mathsf{Enc}(\mathsf{pk}, r_j; s_j)$ and samples $E_{j,\bar{b}}$ obliviously. $\mathcal{P}$ sends these commitments and encryptions as its first message a. The verifier $\mathcal{V}$ samples a random challenge bit e and sends it to the prover. If $e = 0$, then $\mathcal{P}$ decommits to the cycle H and the randomness of the corresponding commitment and encryption, i.e. (r_j, s_j) (the $E_{j,0}$ encryptions are claimed as obliviously sampled where j is an edge in H). Else, it computes a random permutation π from σ to H. $\mathcal{P}$ sends π and decommits to the non-edges in $\pi(G)$ and the randomness of the corresponding commitment and encryption, i.e. (r_j, s_j) (the $E_{j,1}$ encryptions are claimed as obliviously sampled where j is a non-edge in $\pi(G)$). $\mathcal{P}$ sends these decommitments as its response z. Upon obtaining z, the verifier performs the following based on e:

- $e = 0$: 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.

- $e = 1$: Verify that z contains decommitments to 0, and the decommitted edges correspond to non-edges in $\pi(G)$.

For each opened commitment c_j (to bit b_j) the verifier obtains (b_j, r_j, s_j). He checks that $c_j = \mathsf{Com}(b_j; r_j)$ and $E_{j,b_j} = \mathsf{Enc}(\mathsf{pk}, r_j; s_j)$. $\mathcal{V}$ rejects the proof if any check fails. This completes the description of Π_{ZK}.

Multi-Proof Setting. The $\mathsf{crs} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$ can be used by the same prover to prove multiple statements. If the prover is corrupted then his witness for every accepting proof can be extracted by a simulator given the secret key sk. If the prover is honest then the simulator can always construct simulated proofs that are accepting given the trapdoor of $\mathsf{crs}_{\mathsf{Com}}$. He simulates the proofs by equivocating the openings of the commitments. Adaptive security is also ensured based on the equivocal property of the commitment scheme and oblivious sampling property of PKE.

Usefulness in Offline-Online Paradigm. The input-delayed property allows us to substitute the usage of NIZKs in offline-online paradigm protocols where the NIZK is run in the online phase. The prover and the verifier can run the first two rounds of Π_{ZK} in the offline phase without the knowledge of the statement. In the online phase the prover receives the statement and the witness. He computes the third message and sends the proof. It serves the purpose of a NIZK in the online phase of the protocol by relying on DDH.

2.3 Our NIZK protocol

Next, we make Π_{ZK} non-interactive to obtain Π_{NIZK}, which implements $\mathcal{F}_{\mathsf{NIZK}}$ functionality for a single session, using the Fiat-Shamir transform. 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 $\mathbf{e} = g(\mathbf{e}) = f(\mathbf{e})$. This guarantees soundness of the NIZK protocol.

Applying the Fiat-Shamir transform on Π_{ZK}. We denote $\Pi_{\mathsf{ZK}} = (\mathbf{a}, \mathbf{e}, \mathbf{z}) = (\{a^1, \ldots, a^\ell\}, \{e^1, \ldots, e^\ell\}, \{z^1, \ldots, z^\ell\})$. Π_{ZK} protocol is not a trapdoor sigma protocol since the commitments are equivocal in nature. For any $G \notin \mathcal{R}_{\mathsf{Ham}}$ and any prover's first message $\mathbf{a}$ there can be many different openings, hence many accepting transcripts $(\mathbf{a}, \mathbf{e}, \mathbf{z})$. So we rely on the binding property of the commitment scheme and move to a hybrid where the commitments are always binding. In this hybrid, a corrupt prover can produce an accepting transcript corresponding to $\mathbf{a}$ for one possible value of $\mathbf{e}$. The bad challenge function is well-defined in this hybrid. Thus, we can apply the Fiat-Shamir transform using H and rely on the correlation intractability argument for soundness in this hybrid. More details of our NIZK protocol Π_{NIZK} follows.

The setup algorithm initializes the hash function H in the statistical mode, by setting the crs of the protocol to be crs of Π_{ZK} and hash key k, where k is generated as follows:

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

$\mathcal{C}_{\mathsf{sk}}$ is a poly-size circuit that takes $\mathbf{a}$ (of Π_{ZK}) as input. Recall, that $\mathbf{a}$ should be a commitment to a n-node cycle in Π_{ZK}. $\mathcal{C}_{\mathsf{sk}}(\mathbf{a})$ is the circuit computing the function $f_{\mathsf{sk}}(\mathbf{a}) = \mathbf{e}$ s.t. for every $i \in [\ell]$, $e^i = 0$ if it can extract (using the extraction key sk of the commitment scheme) a n-node cycle from a^i, i.e. $\mathsf{Ext}(\mathsf{sk}, a^i)$ is a cycle (where Ext is the extractor algorithm for the commitment scheme). Setting the hash function in the statistical mode ensures that the hash function H is correlation intractable for all relations of the form:

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

The NIZK prover invokes the ZK prover (of Π_{ZK}) to obtain $\mathbf{a}$. The challenge vector $\mathbf{e} = (e^1, e^2, \ldots, e^\ell)$ is generated non-interactively by hashing $\mathbf{a}$ using the hash key k.

$$\mathbf{e} = H(\mathsf{k}, \mathbf{a}),$$

Upon computing $\mathbf{e}$, the NIZK prover invokes the ZK prover with $\mathbf{e}$ to obtain $\mathbf{z} = (z^1, z^2, \ldots, z^\ell)$ as a response. The NIZK prover sends the final proof as $(\mathbf{a}, \mathbf{e}, \mathbf{z})$. The NIZK verifier computes $\mathbf{e}' = H(\mathbf{a})$ and runs the ZK verifier on γ to verify the proof. He aborts if $\mathbf{e} \neq \mathbf{e}'$ or the ZK verifier rejects the proof.

Soundness and Proof of Knowledge. We argue soundness of the protocol by considering a hybrid where the commitments are binding. This hybrid is indistinguishable from the real world due to the binding property of Com. In this hybrid if $G \notin \mathcal{R}_{\mathsf{Ham}}$, then the bad challenge function is well defined for every $\mathbf{a}$. It is guaranteed that the prover cannot construct an accepting proof due to correlation intractability. Correlation intractability ensures that either of the following two exhaustive cases occur for every $i \in [\ell]$:

- $e^i = 0$ *when* $\mathsf{Ext}(\mathsf{sk}, a^i)$ *is not a cycle:* $\mathcal{P}$ has committed to a graph in a^i which does not contain a cycle. In such a case, it can open to a cycle in z^i only if it breaks the binding property of the underlying commitment scheme used to construct the commitments in a^i.

- $e^i = 1$ *when* $\mathsf{Ext}(\mathsf{sk}, a^i)$ *is a cycle:* Let G' denote the graph committed in a^i. We can argue that there does not exist any graph G' which is a permutation of input graph G and G' contains contains a n-node Hamiltonian cycle, since G is not Hamiltonian. Thus, the prover would fail to produce an accepting z^i in this case which is permutation of the non-edges of G, unless it breaks the binding property of the commitment scheme.

For proof of knowledge, the simulator can extract the witness σ from a maliciously generated proof by running the underlying simulator of Π_{ZK} for a corrupt prover.

Adaptive Soundness. Next, we argue adaptive soundness of Π_{NIZK}. The distribution of crs is identical in the real and ideal world executions of Π_{NIZK}. The crs consists of the hash key k and the crs of Π_{ZK}. The Hash function H is instantiated in the statistical mode. It is statistically infeasible to find $\mathbf{e}$, s.t. $\mathbf{e} = f(\mathbf{a})$ unless the adversary breaks the binding of the commitment scheme. Given an adversary for adaptive soundness one can construct an adversary $\mathcal{A}_{\mathsf{COM}}$ for the commitment scheme. $\mathcal{A}_{\mathsf{COM}}$ obtains $\mathsf{crs}_{\mathsf{Com}}$ for the commitment and he samples a PKE key pair $(\mathsf{pk}, \mathsf{sk})$. $\mathcal{A}_{\mathsf{COM}}$ computes $\mathsf{k} \leftarrow \mathcal{H}.\mathsf{StatGen}(1^\kappa, \mathcal{C}_{\mathsf{sk}})$ given sk. $\mathcal{A}_{\mathsf{COM}}$ invokes $\mathcal{A}$ with $\mathsf{crs} = (\mathsf{k}, \mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$ to obtain a proof. The adversary for the commitment scheme can check if $\exists i \in [\ell], \exists j \in [n^2]$, s.t. commitment c^i_j opens both to 0 and 1 by decrypting r^i_j and $r^{i\prime}_j$ from $E^i_{j,0}$ and $E^i_{j,1}$ given the secret key sk. He returns r^i_j and $r^{i\prime}_j$ to the challenger of Com to break the binding property. The only scenario when $\mathcal{A}$ succeeds and $\mathcal{A}_{\mathsf{COM}}$ fails to break the binding property is when $\mathcal{A}$ breaks the correlation intractability property of hash function H. The hash function is initialized in the statistical mode and it is infeasible to compute an $\mathbf{e}$, s.t. $\mathbf{e} = f(\mathbf{a})$, unless $\mathcal{A}$ breaks the binding property of Com. Thus $\mathcal{A}_{\mathsf{COM}}$ always succeeds in breaking the binding property of Com if $\mathcal{A}$ succeeds in breaking adaptive soundness of Π_{NIZK}.

Adaptive Zero Knowledge. For zero-knowledge, the NIZK simulator invokes the ZK simulator of Π_{ZK} to construct the first message $\mathbf{a}$. The NIZK simulator computes $\mathbf{e} = H(\mathbf{a})$ and invokes the ZK simulator of Π_{ZK} to obtain $\mathbf{z}$ as the response. In the real world, $\mathbf{a}$ would contain cycle graphs. In the ideal world the prover's message $\mathbf{a}$ contains commitments in the equivocal mode (due to ZK simulation in Sec. 2.2). We assume that the hash function H works as follows - given sk, decrypt the commitment randomness r_j from the encryption $E_{j,1}$; if c_j is $\mathsf{Com}(1; r_j)$ then set $b_j = 1$, else $b_j = 0$. By following this approach, the hash function H would interprete that all the commitments (in $\mathbf{a}$) in the ideal world are $\mathsf{Com}(1)$, i.e. containing cycle graphs. Hence, the distribution of $\mathbf{a}$ in both worlds is identically distributed for H. The output of H will also be identically distributed in both worlds. Thus, zero knowledge follows from the ZK of Π_{ZK} (by relying on the equivocal property of Com and oblivious sampling property of PKE).

Adaptive Security. We only consider the case where the prover is honest and the verifier is statically corrupt (the other adaptive corruption cases are similar to the static corruption cases). The NIZK simulator constructs a simulated NIZK proof by invoking the ZK simulator of Π_{ZK} as described in the previous paragraph. When the prover gets corrupted post-execution, the simulator obtains the witness σ and invokes the adaptive simulator of Π_{ZK} with σ to obtain randomness that demonstrates consistency between the simulated proof and the witness. Adaptive security of Π_{NIZK} follows from adaptive security of Π_{ZK}.

Multi-Proof Setting. The setup string can be reused by the same prover to prove multiple statements if the underlying ZK protocol satisfies the same property. If a corrupt prover constructs a malicious proof then his witness can be extracted by invoking the simulator of the underlying multi-proof ZK protocol. Similarly, if the prover is honest then the simulator of the underlying multi-proof ZK protocol can be used to construct simulated proofs that are accepting. If the prover gets corrupted post-execution then the simulator of the multi-proof NIZK protocol obtains the corresponding witnesses. He invokes the multi-proof ZK simulator with the witnesses to obtain the prover randomness for the simulated proofs. Adaptive security for multi-proof NIZK protocol follows from the adaptive security of the multi-proof ZK protocol.

2.4 Reducing the length of the crs

Currently, the $\mathsf{crs}_{\mathsf{NIZK}}$ of the Π_{NIZK} protocol contains the public hash key k which depends on the circuit length. We reduce this to $\mathrm{poly}(\kappa)$ by applying a compiler which compiles any single-prover multi-proof NIZK protocol Π_{NIZK} in the $\mathsf{crs}_{\mathsf{NIZK}}$ model to a NIZK protocol Π_{sNIZK} in the short $\mathsf{crs}_{\mathsf{sNIZK}}$ model, where $|\mathsf{crs}_{\mathsf{sNIZK}}| = \mathrm{poly}(\kappa)$, assuming additively homomorphic equivocal commitment Com and a PKE with oblivious ciphertext sampling algorithm. Our compiler is inspired from the work of GOS and it can be instantiated from DDH.

Given the witness w and the statement x for a language $\mathcal{L}$, the prover computes a circuit $\mathcal{C}$ s.t. $\mathcal{C}(y) = \mathcal{R}(x, w)$ where $\mathcal{R}$ is the NP verification relation. Let $y = \{y_i\}_{i\in[|y|]}$. The prover commits to each bit y_i as $c_i = \mathsf{Com}(y_i; r_i)$ and encrypts the corresponding randomness as $e_{i,y_i} = \mathsf{Enc}(\mathsf{pk}, r_i; s_i)$ while $e_{i,\overline{y_i}}$ is sampled obliviously. The output wire is committed as $\mathsf{Com}(1; 1)$. Using Π_{NIZK} the prover proves that each c_i is a commitment to 0 or 1 and the underlying commitment randomness is also encrypted correctly in $e_{i,0}$ or $e_{i,1}$. For each jth NAND gate with input wires α and β and output wires Γ, it computes $C_j = c_\alpha + c_\beta + 2c_\Gamma - 2\mathsf{Com}(0; 1)$ and proves using Π_{NIZK} that C_j is a commitment to 0 or 1. The GOS protocol showed that if the order of the message domain of Com is at least 4 then C_j will always be a commitment to 0 or 1. The verifier verifies the proofs and checks that the commitment corresponding to the output wire is $\mathsf{Com}(1; 1)$.

Adaptive Soundness and Proof of Knowledge. The distribution of the crs is identical in the real and ideal world. A corrupt prover $\mathcal{P}^*$ can adaptively chose the statement based on the crs distribution. If $\mathcal{P}^*$ constructs a proof for a statement $x \notin \mathcal{L}$, then he must have broken the binding property of the commitment scheme or the soundness of Π_{NIZK}. Else if $\mathcal{P}^*$ constructs a proof for a statement $x \in \mathcal{L}$ then the simulator can extract the witness bits y_i from the individual proofs by invoking the simulator of Π_{NIZK}. Note that we encrypt the randomness r_i for each commitment c_i for reduction in the security proof. If a corrupt prover computes two valid openings of c_i then those openings can be decrypted and used to break the binding property of Com.

Adaptive Zero Knowledge. Zero-knowledge is ensured since a ZK simulator can construct the c_i commitments in the equivocal mode, i.e. $c_i = \mathsf{Com}(0; r_i) = \mathsf{Com}(1; r'_i)$, and set the encryptions as $(e_{i,0}, e_{i,1}) = (\mathsf{Enc}(\mathsf{pk}, r_i; s_i), \mathsf{Enc}(\mathsf{pk}, r'_i; s'_i))$. He sets the output wire commitment as $\mathsf{Com}(1; 1)$.

He invokes the ZK simulator of Π_{NIZK} to produce the proofs for each wire and each NAND gate. In the real world one of the encryptions corresponding to a commitment is honestly generated while the other is obliviously sampled. In the ideal world both encryptions are honestly generated. The two cases are indistinguishable due to oblivious ciphertext sampling property of PKE. Thus, ZK follows from the ZK of Π_{NIZK}, hiding of Com and oblivious ciphertext sampling property of PKE. A statically corrupt verifier can also choose the statement adaptively based on the crs distribution. Adaptive ZK of this protocol is ensured since Π_{NIZK} supports adaptive ZK, the Com is hiding and PKE provides oblivious ciphertext sampling property.

Adaptive Security. When the prover gets corrupted and it obtains the witness w it can compute y. Suppose $y_i = 0$, then he opens c_i and $e_{i,0}$ as $(0, r_i, s_i)$ and claims that $e_{i,1}$ was obliviously sampled. It invokes the Π_{NIZK} simulator of wire i with input witness (y_i, r_i) to obtain randomness for the NIZK proof corresponding to wire i. Similar steps are repeated to obtain randomness for each NAND gate j and the corresponding NIZK proof. The encryption of r'_i (in ideal world) is indistinguishable from an obliviously sampled ciphertext (in real world) due to the oblivious sampling property. Thus, security against adaptive corruption is ensured due to adaptive security of Π_{NIZK}, equivocal property of Com and oblivious sampleability of PKE.

Multi-proof. The protocol Π_{sNIZK} also allows the prover to prove multiple statements using the same $\mathsf{crs}_{\mathsf{sNIZK}}$. If a corrupt party breaks the security of the protocol in one of the proof then that party can be used to either break the multi-proof security of Π_{NIZK} or the security of Com.

2.5 Obtaining UC Security for multiple subsessions

We add non-malleability to our Π_{sNIZK} protocol to attain UC-security for multiple statements in different subsessions. This is performed in the same way as GOS using tag based simulation-sound trapdoor commitment $\mathsf{Com}_{\mathsf{SST}}$ and strong one-time signature Sig. The prover generates a pair of signature keys $(\mathsf{vk}, \mathsf{sk}) \leftarrow \text{Sig.}\mathsf{KeyGen}$. It commits to the witness bits w using $\mathsf{Com}_{\mathsf{SST}}$ with the tag being $(\mathsf{vk}, \mathsf{sid}, \mathsf{ssid}, x)$ (where sid is the session ID of the multi-instance NIZK functionality and ssid is the sub-session ID for the particular proof) and encrypts the randomness for the commitments. It proves using Π_{sNIZK} that $\mathcal{R}(x, w) = 1$ and the witness bits are correctly committed to compute a proof π. It signs the proof π using sk and sends the proof π and the signature as the final proof. The signature enables that an adversary cannot forge a signature on a different proof with the same vk. Whereas, $\mathsf{Com}_{\mathsf{SST}}$ and Π_{sNIZK} ensures that an adversary cannot reuse the same proof π in a different session ssid since it is bound to the vk and ssid.

Security against Statically Corrupt Prover. Soundness follows from the binding property of $\mathsf{Com}_{\mathsf{SST}}$, unforgeability of Sig and adaptive soundness of Π_{sNIZK}. The witness can be extracted from the commitments by decrypting the randomness of the commitments from the encryptions using sk. Next, we briefly discuss the different cases for triple adaptive security.

ZK and security against Adaptive Corruption of Prover. The ZK simulator commits to all 0s as witness and invokes the ZK simulator of Π_{sNIZK} to construct the simulated proof. Upon obtaining the witness it can equivocate the commitments using the trapdoor and equivocate the proof by invoking the adaptive simulator of Π_{sNIZK}.

Adaptive Soundness and Adaptive Zero Knowledge. The crs distribution is identical in the real and ideal world for our multi-session UC protocol. Adaptive soundness follows from the unforgeability of signature, binding property of the tag-based commitment scheme and adaptive

soundness of underlying single instance NIZK protocol. For adaptive ZK our ZK simulator (mentioned in previous paragraph) suffices.

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.

Definition 3.1. *([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 a PPT algorithm* oEnc *s.t. for any message* $m \in \mathcal{M}$*, the following two distributions are computationally indistinguishable to a PPT adversary* $\mathcal{A}$*:*

$$|\Pr[\mathcal{A}(m, c) = 1 | (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa), m \leftarrow \mathcal{A}(\mathsf{pk}), c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)]$$

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

3.1 Non-Interactive Zero Knowledge

We also provide the ideal UC-NIZK functionality in Fig. 1. 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.

Figure 1: 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 $(\mathsf{prove}, \mathsf{sid}, \mathsf{ssid}, x, w)$ from party P: If $\mathcal{R}(x, w) = 1$ then 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 ssid as corrupted. If there is a stored tuple $(\mathsf{sid}, \mathsf{ssid}, x, w, \pi)$, then send it to $\mathcal{S}$.

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

We also consider $\mathcal{F}^{\mathsf{m}}_{\mathsf{NIZK}}$ (Fig. 2) 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 $\mathcal{F}^{\mathsf{m}}_{\mathsf{NIZK}}$ considers only a single session between a pair of parties with roles preserved. Different provers have to use different instances of $\mathcal{F}^{\mathsf{m}}_{\mathsf{NIZK}}$ 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.

Figure 2: 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 $(\mathsf{prove}, \mathsf{sid}, x, w, P)$ 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 $\mathcal{Q} = (\mathsf{sid}, P)$. 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 are stored tuples of the form $(\mathsf{sid}, x, w, \pi)$, then send it to $\mathcal{S}$.

Definition 3.2. *A non-interactive zero-knowledge argument system (NIZK) for an NP-language $\mathcal{L}$ consists of three PPT machines $\Pi_{\mathsf{NIZK}} = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$, that have the following properties:*

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

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

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

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

- ***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:*

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

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

Definition 3.3. ***(Adaptive Soundness)*** *Π_{NIZK} is adaptively sound if for every PPT cheating prover $\mathcal{P}^*$ the following holds:*

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

Definition 3.4. ***(Adaptive Zero-Knowledge)*** *Π_{NIZK} is adaptively zero-knowledge if for all PPT verifiers $\mathcal{V}^*$ there exists a PPT simulator $\mathcal{S}$ such that the following distribution ensembles are computationally indistinguishable:*

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

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

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

Definition 3.6. *(**Triple Adaptive Security for multiple instances**)*
Let $\Pi_{NIZK} = (Gen, \mathcal{P}, \mathcal{V})$ be a NIZK protocol in the crs model. Then Π_{NIZK} satisfies triple adaptive security for multiple instances if it UC-securely implements $\mathcal{F}_{NIZK}$ functionality for multiple instances and provides adaptive soundness and adaptive zero knowledge.

3.2 Interactive Zero Knowledge

We present the ZK functionality from [JKO13] (in Fig. 3) in the UC model of [Can01]. Similar to $\mathcal{F}_{\mathsf{NIZK}}^{\mathsf{m}}$ we also consider a $\mathcal{F}_{\mathsf{ZK}}^{\mathsf{m}}$ functionality (Fig. 4) where a single prover proves multiple statements to a verifier in a single session. Different provers have to use different instances of $\mathcal{F}_{\mathsf{ZK}}^{\mathsf{m}}$ to prove statements.

Figure 3: Zero-Knowledge Functionality $\mathcal{F}_{\mathsf{ZK}}$

$\mathcal{F}_{\mathsf{ZK}}$ is parametrized by an NP relation $\mathcal{R}$.

- On input $(\mathsf{prove}, \mathsf{sid}, x, w)$ from $\mathcal{P}$ and $(\mathsf{verify}, \mathsf{sid}, x)$ from $\mathcal{V}$, output $(\mathsf{verification}, \mathsf{sid}, x, \mathcal{R}(x, w))$ to $\mathcal{V}$.

Figure 4: Zero-Knowledge Functionality $\mathcal{F}_{\mathsf{ZK}}^{\mathsf{m}}$ for multiple statements in a single session

$\mathcal{F}_{\mathsf{ZK}}^{\mathsf{m}}$ is parametrized by an NP relation $\mathcal{R}$.

- On input $(\mathsf{prove}, \mathsf{sid}, x, w)$ from $\mathcal{P}$ and $(\mathsf{verify}, \mathsf{sid}, x)$ from $\mathcal{V}$: 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 $\mathcal{Q} = (\mathsf{sid}, P)$ and output $(\mathsf{verification}, \mathsf{sid}, x, \mathcal{R}(x, w))$ to $\mathcal{V}$.

We also define triple adaptive security for interactive ZK. An interactive ZK protocol consists of $\Pi_{\mathsf{ZK}} = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ where Gen generates the crs and $\mathcal{P}$ and $\mathcal{V}$ are interactive algorithms. We denote by $\langle \mathcal{P}(w), \mathcal{V} \rangle(x, \mathsf{crs})$ the distribution of $\mathcal{V}$'s output after running Π_{ZK} with $\mathcal{P}$ on public input (x, crs) and $\mathcal{P}$ possesses the private witness w.

Definition 3.7. *A pair of PPT interactive algorithms $\Pi_{ZK} = (Gen, \mathcal{P}, \mathcal{V})$ constitute an argument system for an NP language $\mathcal{L}$, if the following conditions hold:*

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

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

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

$$\Pr\left[\langle \mathcal{P}^*, \mathcal{V} \rangle(x, crs) = 1 | (crs, td) \leftarrow Gen(1^\kappa, 1^{|x|})\right] \leq neg(\kappa).$$

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

$$\{\langle \mathcal{P}(w), \mathcal{V}^* \rangle (x, \mathsf{crs}) | (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, 1^{|x|})\}_{\kappa \in \mathbb{N}}$$

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

We can also define adaptive soundness and adaptive zero knowledge for Π_{ZK}.

Definition 3.8. *(**Adaptive Soundness**) Π_{ZK} is adaptively sound if for every PPT cheating prover $\mathcal{P}^*$ the following holds:*

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

Definition 3.9. *(**Adaptive Zero-Knowledge**) Π_{ZK} is adaptively zero-knowledge if for all PPT verifiers $\mathcal{V}^*$ there exists a PPT simulator $\mathcal{S}$ such that the following distribution ensembles are computationally indistinguishable:*

$$\{\langle \mathcal{P}(w), \mathcal{V}^*(\mathsf{aux}) \rangle (x, \mathsf{crs}) | (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, 1^{|x|}), (x, w, \mathsf{aux}) \leftarrow \mathcal{V}^*(\mathsf{crs})\}_{\kappa \in \mathbb{N}}$$

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

Π_{ZK} enjoys delayed-input completeness if $\mathcal{P}$ needs x and w only to compute the last round and $\mathcal{V}$ needs x only to compute the output. The previous round messages of Π_{ZK} can be computed by $\mathcal{P}$ and $\mathcal{V}$ having as input only the size of x, i.e. $|x|$. The notion of delayed-input completeness was defined in [CPS+16a] for Sigma protocols. It can be generalized for interactive ZK protocols as follows.

Definition 3.10. *(**Input-Delayed ZK protocols**) Let $\Pi_{ZK} = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ is a $2r + 1$-round interactive ZK protocol where $\mathcal{P}$ sends the first and last round message. We denote $\mathcal{P} = \{\mathcal{P}_i\}_{i \in [r+1]}$ where $\mathcal{P}_i$ denotes the prover algorithm for computing $(2(i-1)+1)$th round message of Π_{ZK}. Similarly, $\mathcal{V} = (\{\mathcal{V}_i\}_{i \in [r]}, \mathcal{V}^{\mathsf{out}})$ where $\mathcal{V}_i$ denotes the verifier algorithm for computing $2i$th round message of Π_{ZK} and $\mathcal{V}^{\mathsf{out}}$ either accepts or rejects the proof. Π_{ZK} is input-delayed ZK protocol if the following holds:*

- *$\{\mathcal{P}_i\}_{i \in [r]}$ takes as input the private state of $\mathcal{P}_{i-1}$ and the public values - length of the statement, i.e. $|x|$, crs and previous round messages.*
- *$\{\mathcal{V}_i\}_{i \in [r]}$ takes as input the private state of $\mathcal{V}_{i-1}$ and the public values - length of the statement, i.e. $|x|$, crs and previous round messages.*
- *$\mathcal{P}_{r+1}$ takes as input (x, w) and private state of $\mathcal{P}_r$.*
- *$\mathcal{V}^{\mathsf{out}}$ takes as input x and private state of $\mathcal{V}_r$.*

We define triple adaptive security for Π_{ZK} for a single instance as follows.

Definition 3.11. *(**Triple Adaptive Security for a single instance**) Let $\Pi_{ZK} = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ be an interactive ZK protocol in the crs model. Then Π_{ZK} satisfies triple adaptive security for a single instance if it securely implements $\mathcal{F}_{ZK}$ functionality for a single instance and provides adaptive soundness and adaptive zero knowledge.*

3.3 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 $\mathcal{V}$ in the Com phase. Later in the open phase, C can open c to m by sending the decommitment to $\mathcal{V}$ 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:

$$\mathsf{Com}(m_1; r_1) + \mathsf{Com}(m_2; r_2) = \mathsf{Com}(m_1 + m_2; r_1 + r_2)$$

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.

We define an equivocal commitment scheme $\mathsf{Com} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Ver}, \mathsf{Equiv})$ as follows:

Definition 3.12. ***(Correctness)*** *Com is a correct commitment scheme if the following holds true*

$$\Pr\left[\mathsf{Ver}(m, c, \mathsf{crs}, r) = 1 | (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa), c \leftarrow \mathsf{Com}(m, \mathsf{crs}; r)\right] = 1$$

Definition 3.13. ***(Binding)*** *Com is computationally binding scheme if the following holds true for all PPT adversary $\mathcal{A}$*

$$\Pr\left[(m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(\mathsf{crs}) | (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa),\right.$$

$$\left.\mathsf{Com}(m_0; r_0) = \mathsf{Com}(m_1; r_1)\right] \leq \mathsf{neg}(\kappa)$$

Definition 3.14. ***(Hiding)*** *Com is computationally hiding scheme if the following holds true for all PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.*

$$\Pr\left[b == b' | (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa), (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{crs}), b \leftarrow_r \{0, 1\},\right.$$

$$\left.(c, d) \leftarrow \mathsf{Com}(m_b), b' \leftarrow \mathcal{A}_2(c; \mathsf{st})\right] \leq \frac{1}{2} + \mathsf{neg}(\kappa)$$

Definition 3.15. ***(Equivocal)*** *Com is equivocal if it has a PPT algorithm Equiv s.t. the following holds true for all PPT adversary $\mathcal{A}$ and all message pairs (m_0, m_1).*

$$\Big|\Pr\left[\mathcal{A}(c, r) = 1 | (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa), m \leftarrow \mathcal{A}(\mathsf{crs}), c = \mathsf{Com}(\mathsf{crs}, m; r)\right]$$

$$-\Pr\left[\mathcal{A}(c, r) = 1 | (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa), m \leftarrow \mathcal{A}(\mathsf{crs}), c = \mathsf{Com}(\mathsf{crs}, m'; r'),\right.$$

$$\left.r = \mathsf{Equiv}(m, r, c', \mathsf{td})\right]\Big| \leq \mathsf{neg}(\kappa),\ \textit{for } m \neq m'$$

Definition 3.16. ***(Extractable)*** *Com is extractable if it has a PPT algorithm Ext if the following holds true for any $m \in \{0, 1\}$ and $r \in \{0, 1\}^*$.*

$$\Pr\left[\mathsf{Ver}(m, c, \mathsf{crs}, r) = 1 \wedge \mathsf{Ext}(\mathsf{td}, c) \neq m : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa), c = \mathsf{Com}(m; r)\right] \leq \mathsf{neg}(\kappa)$$

We denote $\mathsf{Com}(m, \mathsf{crs}; r)$ as $\mathsf{Com}(m; r)$ to avoid notation overloading.

Tag-based simulation soundness. A tag-based simulation sound commitment scheme is denoted as $\mathsf{Com}_{\mathsf{SST}} = (\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Dec}, \mathsf{Ver}, \mathsf{TCom}, \mathsf{TOpen})$. We define it following [GOS12]. The key generation algorithm KeyGen produces a crs as well as a trapdoor key td. There is a Com algorithm that takes as input crs, a message m and any tag t and outputs a commitment $c = \mathsf{Com}(\mathsf{crs}, m, t; r)$. To open a commitment c with tag t we reveal m and the randomness r. Verification is performed by verifying the commitment with r on (m, t). $\mathsf{Com}_{\mathsf{SST}}$ is a correct, binding and hiding commitment scheme. $\mathsf{Com}_{\mathsf{SST}}$ has trapdoor opening if the following holds for all PPT adversary $\mathcal{A}$.

$$\Pr\left[\mathsf{Ver}(\mathsf{crs}, m, t, r) = 1 | (c, \alpha) \leftarrow \mathsf{TCom}(\mathsf{td}, t), m \leftarrow \mathcal{A}(c, \mathsf{crs}), \right.$$
$$\left. r \leftarrow \mathsf{TOpen}(\mathsf{crs}, \alpha, c, m, t)\right] = 1$$

The tag-based simulation-soundness property means that a commitment using tag t remains binding even if we have made equivocations for commitments using different tags. For all non-uniform PPT adversaries $\mathcal{A}$ we have

$$\Pr\left[(\mathsf{crs}, td) \leftarrow \mathsf{KeyGen}(1^\kappa), (c, t, m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}^{O(\cdot)}(\mathsf{crs}) | t \notin Q, \right.$$
$$\left. c = \mathsf{Com}(m_0, t; r_0) = \mathsf{Com}(m_1, t; r_1), m_0 \neq m_1)\right] \leq \mathsf{neg}(\kappa).$$

where $O(commit, t)$ computes $(c, \alpha) \leftarrow \mathsf{TCom}(\mathsf{td}, t)$ returns c and stores (c, t, α), and $O(open, c, m, t)$ returns $r \leftarrow \mathsf{TOpen}(\mathsf{crs}, \alpha, c, m, t)$ if (c, t, α) has been stored, and where Q is the list of tags for which equivocal commitments have been made by oracle $O(\cdot)$.

3.4 Common Reference String Model

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.

3.5 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 3.17. *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 3.18. *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 3.19. *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 3.20. *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 Delayed Input Three Round ZK Argument

The classical result of [FLS99] present an honest verifier zero knowledge protocol for Hamiltonian graph with adaptive soundness (i.e. the graph can be chosen after $\mathcal{V}$ challenges $\mathcal{P}$) and adaptive zero knowledge. We remove the honest verifier assumption and prove that it can be made zero knowledge and secure against adaptive corruption using a non-interactive equivocal commitment scheme and a public key encryption scheme with an oblivious ciphertext sampling algorithm. Our ZK protocol Π_{ZK} is presented in Fig. 7 and the high-level overview can be found in Sec. 2.2. We prove triple adaptive security of Π_{ZK} by proving Thm. 4.1.

Theorem 4.1. *If Com is a non-interactive equivocal commitment scheme and* PKE *is an IND-CPA public key encryption scheme with oblivious ciphertext sampleability, then Π_{ZK} UC-realizes $\mathcal{F}_{ZK}$ for a* single instance *of the Hamiltonian Relation $\mathcal{R}_{Ham}$ against adaptive corruptions in the common reference string model. Furthermore, Π_{ZK} is adaptively sound and adaptive zero-knowledge.*

Proof. We demonstrate that Π_{ZK} is triply adaptive as follows. We consider $\ell = max(8\mu, \kappa)$ for statistical error $2^{-\mu}$. $\mathcal{V}$ does not possess an input and hence the simulator can trivially simulate its view for ℓ iterations by sampling ℓ random bits. Based on the $\mathcal{P}$'s view, we have two possible corruption cases:

$\mathcal{P}$ is statically corrupt and $\mathcal{V}$ is honest. In this case, the adversary corrupts the prover from the start and tries to construct a correct proof without knowing the witness σ. The simulator $\mathcal{S}_{\mathcal{P}}$ plays the role of the verifier. This demonstrates the soundness property of the protocol. If $\mathcal{P}^*$ constructs an accepting proof then the simulator tries to extract a valid witness from it. The simulation algorithm is presented in Fig. 5. We present the formal hybrids and prove indistinguishability as follows:

- $\mathtt{Hyb}_0$**:** Real world.
- $\mathtt{Hyb}_1$**:** Same as $\mathtt{Hyb}_0$, except the reduction aborts if any commitment c^i_j opens to both 0 and 1 using randomness $r^i_{j,0}$ and $r^i_{j,1}$.

  The hybrids are distinguishable when $\mathcal{P}^*$ has broken the binding property of the commitment scheme c^i_j. In such a case, the adversary for the binding property of the commitment scheme can decrypt $r^i_{j,0}$ and $r^i_{j,1}$ from $E^i_{j,0}$ and $E^i_{j,1}$ respectively using sk. It returns $(0; r^i_{j,0})$ and $(1; r^i_{j,1})$ as the response to the challenger of the binding property of Com. If $\mathcal{P}^*$ distinguishes between the two hybrids then there exists an adversary who breaks the binding property of Com.

Figure 5: Simulation against a statically corrupt $\mathcal{P}^*$ by $\mathcal{S}_{\mathcal{P}}$

- **Public Inputs:** Common reference string $\mathsf{crs}_{\mathsf{ZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$.
- **Simulator Inputs:** Trapdoor of $\mathsf{crs} = (\mathsf{td}, \mathsf{sk})$ where td is the equivocating trapdoor of Com and sk is the secret key for pk.

Prove :
$\mathcal{P}^*$ sends $\mathbf{a} = (a^1, a^2, \ldots, a^\ell)$.

Challenge : $\mathcal{S}_{\mathcal{P}}(1^\kappa, 1^n)$
$\mathcal{S}_{\mathcal{P}}$ samples challenge bits $\mathbf{e} = (e^1, e^2, \ldots, e^\ell)$ and sends it to $\mathcal{P}^*$.

Response :
$\mathcal{P}^*$ sends $\gamma = (\mathbf{a}, \mathbf{e}, \mathbf{z})$ where, $\mathbf{z} = (z^1, z^2, \ldots, z^\ell)$, as response.

Verify : $\mathcal{S}_{\mathcal{P}}(G, \gamma)$

- $\mathcal{S}_{\mathcal{P}}$ runs the honest verifier algorithm. If it outputs Reject, then $\mathcal{S}_{\mathcal{P}}$ outputs Reject.
- For $i \in [\ell], j \in [n^2]$, $\mathcal{S}$ decrypts $r^i_{j,0}$ and $r^i_{j,1}$ from $E^i_{j,0}$ and $E^i_{j,1}$ using sk. $\mathcal{S}$ aborts if $r^i_{j,0}$ and $r^i_{j,1}$ are valid decommitments of c^i_j to both 0 and 1.
- $\mathcal{S}$ aborts if μ instances of a^i are malformed (i.e. does not commit cycle graphs).
- Else, $\mathcal{S}_{\mathcal{P}}$ extracts σ as follows. For $i \in [\ell]$ and $e^i = 1$:
  - Obtain the permutation π^i.
  - Extract the committed n-node permuted cycle H^i from a^i.
  - Obtain the candidate Hamiltonian cycle σ^i by applying the inverse of permutation π^i on H^i.

  $\mathcal{S}$ outputs the majority among the candidate cycles as the witness cycle σ. Invoke $\mathcal{F}_{\mathsf{ZK}}$ with σ and conclude simulation. If extraction fails then $\mathcal{S}_{\mathcal{P}}$ sets $\sigma = \perp$ and aborts.

– $\texttt{Hyb}_2$**:** Same as $\texttt{Hyb}_1$, except $\mathcal{S}_{\mathcal{P}}$ aborts if it fails to extract the correct witness.

The simulator fails to extract the correct witness when the majority of the candidate cycles are incorrect. A corrupt $\mathcal{P}^*$ can compute μ of the a^i values, except with negligible probability $2^{-\mu}$. We can assume that $e_i = 1$ for those bad μ instances. In such a case, the simulator needs to have $\mu+1$ good instances where $e_i = 1$. So there should be at least $2\mu+1$ instances where $e_i = 1$. That means that the challenge vector $\mathbf{e}$ should contain $2\mu+1$ 1s. Such an event occurs with overwhelming probability $(1 - 2^{-2\mu})$ when $\ell = 8\mu$. The details of the calculation is provided in Appendix. A. The two hybrids are indistinguishable statistically except with negligible probability $2^{-\mu}$.

If the verifier gets corrupted post-execution then the simulator can trivially simulate him by returning $\mathbf{e}$.

$\mathcal{P}$ is honest and $\mathcal{V}$ is statically corrupt. In this case, the simulator $\mathcal{S}_{\mathcal{V}}$ constructs an accepting proof without knowing the witness. This demonstrates the zero knowledge property of the protocol. Later, when the prover gets corrupted post-execution the simulator has to provide randomness s.t. the simulated proof is consistent with the randomness. The original FLS prove ZK by sampling the challenge $\mathbf{e}$ first and then constructing $\mathbf{a}$ based on that. However, we cannot perform that in the UC setting as we lack rewinding property. Instead, we rely on the equivocal property of the commitment scheme. We provide high-level overview for simulating one iteration and the same can be repeated for ℓ instances. $\mathcal{S}_{\mathcal{V}}$ computes the commitments of a^i in the equivocal mode. Upon obtaining the challenge bit e^i, he performs either of the following. If $e^i = 0$, then he samples a random n-node cycle H^i and opens the commitments corresponding to H^i as 1. He sets z^i as the decommitments. If $e^i = 1$, then he samples a random permutation π^i and then decommits to the non-edges in $\pi^i(G)$ as z^i and equivocates the corresponding commitments s.t. they open to 0. In both cases, an adversarial $\mathcal{V}^*$ cannot distinguish a real world execution from ideal world execution due to the hiding property of the commitment scheme. The detailed simulation algorithm is provided in Fig. 6. We present the formal hybrids and prove indistinguishability as follows:

– $\texttt{Hyb}_0$**:** Real world.

– $\texttt{Hyb}_1$**:** Same as $\texttt{Hyb}_0$, except $\mathcal{S}_{\mathcal{V}}$ follows the simulation algorithm. The two hybrids differ in the distribution of c^i_j where j is an edge in H. In $\texttt{Hyb}_0$, the value of c^i_j is $\mathsf{Com}(1)$, whereas in $\texttt{Hyb}_1$ c^i_j is in the equivocal mode. The two hybrids are indistinguishable due to the hiding property of the commitment and the real encryptions of the commitment randomness (in ideal world corresponding to $\overline{b^i_j}$) are indistinguishable from oblivious ciphertexts (in real world) due to the oblivious ciphertext sampling property of PKE. The simulator can successfully equivocate these commitments in $\texttt{Hyb}_1$ due to the equivocal property of the commitment scheme.

$\mathcal{P}$ gets corrupted post-execution. In this case, the prover gets corrupted post-execution and $\mathcal{S}_{\mathcal{V}}$ has to provide randomness s.t. the transcript looks consistent with the witness. We consider two cases of simulation based on the value of e^i_j :

– $e^i = 0$ *:* In this case, $\mathcal{S}_{\mathcal{V}}$ has previously opened the commitments corresponding to a random n-node cycle to 1 as z^i. Upon post-execution corruption, the simulator provides opening as $(0, r^i_{j,0}, s^i_{j,0})$ for the unopened commitments.

Indistinguishability proceeds due to equivocal proeprty of the commitment scheme and oblivious ciphertext sampling property of PKE.

Figure 6: Simulation against a statically corrupt $\mathcal{V}^*$ by $\mathcal{S}_{\mathcal{V}}$

– **Public Inputs:** Common reference string $\mathsf{crs}_{\mathsf{ZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$.
– **Simulator Inputs:** Trapdoor of $\mathsf{crs} = (\mathsf{td}, \mathsf{sk})$ where td is the equivocating trapdoor of Com and sk is the secret key for pk.

Prove : $\mathcal{S}_{\mathcal{V}}(1^\kappa, 1^n)$
For each $i \in [\ell]$ and $j \in [n^2]$, $\mathcal{S}_{\mathcal{V}}$ constructs a^i as follows:
– Construct $c^i_j = \mathsf{Com}(0; r^i_{j,0}) = \mathsf{Com}(1; r^i_{j,1})$.
– Set $E^i_{j,0} = \mathsf{Enc}(\mathsf{pk}, r^i_{j,0}; s^i_{j,0})$ and $E^i_{j,1} = \mathsf{Enc}(\mathsf{pk}, r^i_{j,1}; s^i_{j,1})$.
– Construct $a^i = \{c^i_j, E^i_{j,0}, E^i_{j,1}\}$.

$\mathcal{S}$ sends $\mathbf{a} = (a^1, a^2, \ldots, a^\ell)$ to $\mathcal{V}^*$.

Challenge :
$\mathcal{V}^*$ sends challenge bits $\mathbf{e} = (e^1, e^2, \ldots, e^\ell)$ and sets internal state $\mathsf{st}_{\mathcal{V}} = \mathbf{e}$.

Response : $\mathcal{S}_{\mathcal{V}}(G, \mathbf{a}, \mathbf{e})$
– For $i \in [\ell]$ and $e^i = 0$, $\mathcal{S}_{\mathcal{V}}$ samples a random n-node cycle H^i and computes $z^i = \{j, 1, r^i_{j,1}, s^i_{j,1}\}_{j \in H^i}$.
– For $i \in [\ell]$ and $e^i = 1$, $\mathcal{S}_{\mathcal{V}}$ performs the following:
  • Sample a random permutation π^i.
  • Apply $\pi^i(G)$ and decommit to non-edges in $\pi^i(G)$ s.t. they open to 0 by providing $r^i_{j,0}$ and $s^i_{j,0}$ as opening randomness.
  • Construct $z^i = (\pi^i, \{j, 0, r^i_{j,0}, s^i_{j,0}\}_{j \in \overline{\pi^i(G)}})$.
– Construct the response $z = (z^1, z^2, \ldots, z^\ell)$.
– Sends $\gamma = (\mathbf{a}, \mathbf{e}, \mathbf{z})$.

Verify :
Performs its own adversarial algorithm.

– $e^i = 1$ *:* In this case, the simulator has opened the commitments corresponding to the non-edges of $\pi^i(G)$ during the proof phase. Upon post-execution corruption, the simulator equivocates the commitments corresponding to $\pi^i(\sigma)$ s.t. it opens to 1 and the other unopened commitments open to 0.

  Indistinguishability proceeds due to equivocal property of the commitment scheme and oblivious ciphertext sampling property of PKE.

There is another adaptive corruption case, where the prover can get corrupted after sending a^i and before receiving e^i. The simulator opens the commitments s.t. a^i contains a random n-node cycle graph. Indistinguishability proceeds due to equivocal property of the commitment scheme and oblivious ciphertext sampling property of PKE.

Proof of Adaptive Soundness. Our crs distribution is identical in real and ideal world execution. It consists of $\mathsf{crs}_{\mathsf{ZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$. If a corrupt prover breaks adaptive soundness of the protocol then he breaks the binding property of the commitment scheme or he correctly guesses the $\mathbf{e}$ string, which happens with negligible probability. We build an adversary $\mathcal{A}_{\mathsf{COM}}$ for the commitment scheme as follows. $\mathcal{A}_{\mathsf{COM}}$ obtains the $\mathsf{crs}_{\mathsf{Com}}$ for the commitment and he samples a PKE keypair $(\mathsf{pk}, \mathsf{sk})$ to set $\mathsf{crs}_{\mathsf{ZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$. $\mathcal{A}_{\mathsf{COM}}$ invokes the adversary for adaptive soundness to obtain a proof. The adversary for the commitment scheme can check if $\exists i \in [\ell], \exists j \in [n^2]$, s.t. commitment c^i_j opens both to 0 and 1 by decrypting r^i_j and $r^{i\prime}_j$ from $E^i_{j,0}$ and $E^i_{j,1}$ given the secret key sk. He returns r^i_j and $r^{i\prime}_j$ to the challenger of Com to break the binding property. The only scenario when the adaptive soundness adversary succeeds and $\mathcal{A}_{\mathsf{COM}}$ fails to break the binding property is when the adaptive soundness adversary correctly guesses $\mathbf{e}$ which occurs with $2^{-\ell} \leq 2^{-\mu}$ probability.

Proof of Adaptive ZK. Our ZK simulator sends the first message $\mathbf{a}$ without the knowledge of the statement graph G. It requires the knowledge of G for responding to the verifier's challenge $\mathbf{e}$. The input statement can be adaptively chosen by the adversary after observing $\mathbf{a}$ and still our ZK property would hold. Thus, Π_{ZK} satisfies the notion of adaptive zero-knowledge. □

Protocol Π_{ZK} also implements $\mathcal{F}^{\mathsf{m}}_{\mathsf{ZK}}$ (Fig. 4) functionality, i.e. a single prover can prove multiple statements using the same $\mathsf{crs}_{\mathsf{ZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$, if the $\mathsf{crs}_{\mathsf{Com}}$ of the commitment scheme can be reused multiple times. The protocol also satisfies triple adaptive security for each proof. Our result is summarized in Thm. 4.2.

Theorem 4.2. *If* Com *is a non-interactive equivocal commitment scheme in the reusable* $\mathsf{crs}_{\mathsf{Com}}$ *model and* PKE *is an IND-CPA public key encryption scheme with oblivious ciphertext sampleability, then* Π_{ZK} *UC-realizes* $\mathcal{F}^{\mathsf{m}}_{\mathsf{ZK}}$ *for the Hamiltonian relation* $\mathcal{R}_{\mathsf{Ham}}$ *against adaptive corruptions in the common reference string model. Furthermore,* Π_{ZK} *is adaptively sound and adaptive zero-knowledge.*

Proof. The $\mathsf{crs} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$ can be used by the same prover to prove multiple statements if $\mathsf{crs}_{\mathsf{Com}}$ can be reused for multiple commitments. If the prover is corrupted then his witness for every accepting proof can be extracted by a simulator given the secret key sk. If the prover is honest then the simulator can always construct simulated proofs that are accepting given the trapdoor of $\mathsf{crs}_{\mathsf{Com}}$. He simulates the proofs by equivocating the openings of the commitments. Adaptive security is also ensured based on the equivocal property of the commitment scheme and oblivious sampling property of PKE. Hence, adaptive security for multi-proof setting follows from

the adaptive security of single-proof setting. Adaptive soundness and adaptive zero knowledge for this setting also follows from the single proof setting. □

5 Triply Adaptive NIZK Argument

In this section, we present our adaptively-secure NIZK construction Π_{NIZK} based on H function and the ZK protocol Π_{ZK}. The prover $\mathcal{P}$ invokes the prover algorithm $\Pi_{\mathsf{ZK}}.\mathcal{P}_1$ to obtain $\mathbf{a}$. Then, it computes $\mathbf{e}$ by hashing $\mathbf{a}$ using H. Finally, it constructs the response $\mathbf{z}$ by invoking $\Pi_{\mathsf{ZK}}.\mathcal{P}_2$ on $\mathbf{a}, \mathbf{e}$ and internal state $\mathsf{st}_{\mathcal{P}}$ of $\Pi_{\mathsf{ZK}}.\mathcal{P}_1$. Verification is performed by invoking the verifier algorithm of Π_{ZK} on $\gamma = (\mathbf{a}, \mathbf{e}, \mathbf{z})$. The formal protocol can be found in Fig. 10 and the high-level overview can be found in Sec. 2.3. Our protocol UC-securely realizes a single instance of $\mathcal{F}_{\mathsf{NIZK}}$. We prove triple adaptive security of Π_{NIZK} by proving Thm. 5.1.

Theorem 5.1. *If $\mathcal{H}$ is a somewhere statistically correlation intractable hash function family with programmability, Π_{ZK} is an input-delayed triply adaptive three round public-coin protocol implementing $\mathcal{F}_{\mathsf{ZK}}$, then Π_{NIZK} UC-realizes $\mathcal{F}_{\mathsf{NIZK}}$ for a* single instance *of the Hamiltonian Language $\mathcal{R}_{\mathsf{Ham}}$ against adaptive adversaries. Furthermore, Π_{NIZK} is adaptively sound and adaptively zero knowledge.*

Proof. The hash key k of H function is generated in the statistical mode by running the $\mathsf{StatGen}$ algorithm on $\mathcal{C}_{\mathsf{sk}}$, where sk is the secret key in Π_{ZK} which can be used to extract the committed values in $\mathbf{a}$. $\mathcal{C}$ is a poly-size circuit that takes $\mathbf{a}$ (of Π_{ZK}) as input, computing the function $f_{\mathsf{sk}}(\mathbf{a}) = \mathbf{e}$ s.t. for every $i \in [\ell]$, $e^i = 0$ if it can extract a n-node cycle from a^i, i.e. $\mathsf{Ext}(\mathsf{sk}, a^i)$ is a cycle. The has function is somewhere statistical correlation intractable if the commitments are binding. It ensures that a corrupt prover cannot construct a correct proof when the graph G is not Hamiltonian. For proof of knowledge, the simulator can extract the witness σ from a maliciously generated proof by invoking the simulator $\Pi_{\mathsf{ZK}}.\mathcal{S}_{\mathcal{P}}$ (extractor to be specific) for the Π_{ZK}. For zero-knowledge, the simulator invokes the ZK simulator $\Pi_{\mathsf{ZK}}.\mathcal{S}_{\mathcal{V}}$ to construct a simulated proof using the equivocal property of the commitment scheme. For post-execution corruption of prover, it again invokes $\Pi_{\mathsf{ZK}}.\mathcal{S}_{\mathcal{V}}$ with a correct witness to obtain randomness that demonstrates consistency between the simulated proof and the witness. Next, we present our formal security proof. Our proof contains two corruption cases, similar to the proof of Π_{ZK}.

$\mathcal{P}$ is statically corrupt and $\mathcal{V}$ is honest. First, we demonstrate soundness and proof of knowledge by simulating against a corrupt prover in Fig. 8. The hybrid argument follows:

- $\mathtt{Hyb}_0$**:** Real world.
- $\mathtt{Hyb}_1$**:** Same as $\mathtt{Hyb}_0$, except the reduction decrypts all the encryptions to obtain candidate $r^i_{j,0}$ and $r^i_{j,1}$ values. It aborts if any (both opened and unopened) commitment c^i_j opens to both 0 and 1 using randomness $r^i_{j,0}$ and $r^i_{j,1}$ respectively.

  The hybrids are distinguishable when $\mathcal{P}^*$ has broken the binding property of the commitment scheme c^i_j. In such a case, the adversary for the binding property can return $(0; r^i_{j,0})$ and $(1; r^i_{j,1})$ as the response to the challenger of the binding property. It is ensured that the bad challenge function is defined if the reduction does not abort in $\mathtt{Hyb}_1$ since the commitments are binding and for each $\mathbf{a}$ there is only one possible $\mathbf{e}$ s.t. $(\mathbf{a}, \mathbf{e}, \mathbf{z})$ is valid.

Figure 7: Triply Adaptively Secure Zero Knowledge Protocol

Π_{ZK}

- **Primitives:** Non-interactive equivocal commitment scheme Com=(Gen, Com, Ver, Equiv) and public key encryption scheme PKE = ($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$) with oblivious ciphertext sampling algorithm oEnc.
- **Public Inputs:** Setup string $\mathsf{crs}_{\mathsf{ZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$ where $(\mathsf{crs}_{\mathsf{Com}}, \mathsf{td}) \leftarrow \mathsf{Com.Gen}(1^\kappa)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{PKE.KeyGen}(1^\kappa)$. Let $\ell = \max(8\mu, \kappa)$.
- **Private Inputs:** $\mathcal{V}$ has an n-node input graph G. $\mathcal{P}$ has the same n-node input graph G and a valid Hamiltonian Cycle σ.
- **Notations:** We denote prover algorithm as $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ and verifier algorithm as $\mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2)$. For a graph G we denote the complement graph as $\overline{G}$.

Commit : $\mathcal{P}_1(1^\kappa, 1^n)$
Repeat the following protocol for $i \in [\ell]$:

- Sample a random n-node cycle graph H^i.
- Commit to the adjacency matrix of H^i as follows:
  - For each edge j commit to 1, as $c^i_j = \mathsf{Com}(1; r^i_j)$ with randomness r^i_j.
  - For each non-edge j commit to 0, as $c^i_j = \mathsf{Com}(0; r^i_j)$ with randomness r^i_j.
  - For $j \in [n^2]$, the prover also encrypts the commitment randomness based on the bit b^i_j being committed as follows:
    - If $c^i_j = \mathsf{Com}(0; r^i_j)$, then set $E^i_{j,0} = \mathsf{Enc}(\mathsf{pk}, r^i_j; s^i_j)$ with randomness s^i_j and sample $E^i_{j,1} \leftarrow \mathsf{oEnc}(1^\kappa)$.
    - If $c^i_j = \mathsf{Com}(1; r^i_j)$, then set $E^i_{j,1} = \mathsf{Enc}(\mathsf{pk}, r^i_j; s^i_j)$ with randomness s^i_j and sample $E^i_{j,0} \leftarrow \mathsf{oEnc}(1^\kappa)$.

Construct the first message $\mathbf{a} = \{c^i_j, E^i_{j,0}, E^i_{j,1}\}_{i\in[\ell], j\in[n^2]}$. Send $\mathbf{a}$ to $\mathcal{V}$. Set internal state $\mathsf{st}_{\mathcal{P}} = \{b^i_j, r^i_j, s^i_j\}_{i\in[\ell], j\in[n^2]}$.

Challenge : $\mathcal{V}_1(1^\kappa)$
$\mathcal{V}$ samples a challenge vector $\mathbf{e} \leftarrow \{0,1\}^\ell$ and sends it to $\mathcal{P}$. Sets internal state $\mathsf{st}_{\mathcal{V}} = \mathbf{e} = (e^1, \ldots, e^\ell)$.

Response : $\mathcal{P}_2((G, \sigma), e, \mathsf{st}_{\mathcal{P}})$
Repeat the following protocol for $i \in [\ell]$:

- Upon obtaining the statement graph G and Hamiltonian cycle σ as witness compute a permutation π^i s.t. $H^i = \pi^i(\sigma)$.
- If $e^i = 0$, decommit to edges in H^i. $\mathcal{P}$ also opens to the commitment and encryption randomness, i.e. $z^i = \{j, b^i_j, r^i_j, s^i_j\}_{j\in H}$.
- If $e^i = 1$, reveal π^i and decommit to non-edges in $\pi^i(G)$. $\mathcal{P}$ also opens to the commitment and encryption randomness, i.e. $z^i = (\pi^i, \{j, b^i_j, r^i_j, s^i_j\}_{j\in\overline{\pi^i(G)}})$.

Sends the proof $\gamma = (\mathbf{a}, \mathbf{e}, \mathbf{z}) = (\{a^1, \ldots, a^\ell\}, \mathbf{e}, \{z^1, \ldots, z^\ell\})$.

Verify : $\mathcal{V}_2(G, \gamma, \mathsf{st}_{\mathcal{V}})$
Repeat the following protocol for $i \in [\ell]$:

- If $e^i = 0$, $\mathcal{V}$ outputs Accept if for all $j \in H^i$, the following holds - $b^i_j = 1$, H^i is a cycle and $c^i_j = \mathsf{Com}(1; r^i_j)$ where $E^i_{j,1} = \mathsf{Enc}(\mathsf{pk}, r^i_j; s^i_j)$.
- If $e^i = 1$, $\mathcal{V}$ outputs Accept if for all $j \in \overline{\pi^i(G)}$ the following holds - $b^i_j = 0$, $c^i_j = \mathsf{Com}(0; r^i_j)$ and $e^i_{j,0} = \mathsf{Enc}(\mathsf{pk}, r^i_j; s^i_j)$.
- Else, $\mathcal{V}$ outputs Reject.

Figure 8: Simulation against a statically corrupt $\mathcal{P}^*$ by $\mathcal{S}$

- **Public Inputs:** Common reference string $\mathsf{crs}_{\mathsf{NIZK}} = (\mathsf{k}, \mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$.
- **Simulator Inputs:** Trapdoor of $\mathsf{crs} = (\mathsf{td}, \mathsf{sk})$ where td is the equivocating trapdoor of Com and sk is the secret key for pk.

Prove :
$\mathcal{P}^*$ sends $\gamma = (\mathbf{a}, \mathbf{e}, \mathbf{z})$.

Verify : $\mathcal{S}(G, \gamma)$

- $\mathcal{S}$ runs the honest verifier algorithm. If it outputs REJECT, then $\mathcal{S}$ outputs REJECT.
- For $i \in [\ell], j \in [n^2]$, $\mathcal{S}$ decrypts $r^i_{j,0}$ and $r^i_{j,1}$ from $E^i_{j,0}$ and $E^i_{j,1}$ using sk. $\mathcal{S}$ aborts if $r^i_{j,0}$ and $r^i_{j,1}$ are valid decommitments of c^i_j to 0 and 1 respectively.
- $\mathcal{S}$ extracts σ by invoking $\mathcal{S}_{\mathcal{P}}$ on γ.
- $\mathcal{S}$ aborts if $\sigma = \perp$ else it invokes $\mathcal{F}_{\mathsf{NIZK}}$ with σ to complete simulation.

- $\mathtt{Hyb}_2$**:** Same as $\mathtt{Hyb}_1$, except $\mathcal{S}$ aborts if it fails to extract the correct witness from $\mathbf{a}$.

  The simulator fails to extract the correct witness when the corrupt prover $\mathcal{P}^*$ finds an $\mathbf{a}$ s.t. $f_{\mathsf{sk}}(\mathbf{a}) = \mathbf{e}$ or he breaks the security of Π_{ZK}. H is statistically correlation intractable for $\mathcal{R}_{\mathsf{sk}} = \{(\mathbf{a}, \mathbf{e}) : \mathbf{e} = f_{\mathsf{sk}}(\mathbf{a})\}$ and hence such an $\mathbf{a}$ does not exist. An adversary who can distinguish between the two hybrids can be used to break the security of Π_{ZK} since the verifier accepts the proof and yet the simulator of Π_{ZK} fails to extract a correct witness.

If the verifier gets corrupted post-execution then the simulator can trivially simulate him since he does not possess any input or randomness.

$\mathcal{P}$ is honest and $\mathcal{V}$ is statically corrupt. Next, we demonstrate zero knowledge by simulating against a statically corrupt verifier in Fig. 9. The hybrid argument follows:

Figure 9: Simulation against a statically corrupt $\mathcal{V}^*$

- **Public Inputs:** Common reference string $\mathsf{crs}_{\mathsf{NIZK}} = (\mathsf{k}, \mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$.
- **Simulator Inputs:** Trapdoor of $\mathsf{crs} = (\mathsf{td}, \mathsf{sk})$ where td is the equivocating trapdoor of Com and sk is the secret key for pk.

Prove : $\mathcal{S}(G, 1^\kappa, 1^n)$

- $\mathcal{S}$ invokes $\mathcal{S}_{\mathcal{V}}(1^\kappa, 1^n)$ to obtain $\mathbf{a}$ and computes $\mathbf{e} = H(\mathsf{k}, \mathbf{a})$.
- Constructs $\mathbf{z} = \mathcal{S}_{\mathcal{V}}(G, \mathbf{a}, \mathbf{e})$.
- $\mathcal{S}$ sends the proof $\gamma = (\mathbf{a}, \mathbf{e}, \mathbf{z})$ to $\mathcal{V}$.

Verify :
Performs its own adversarial algorithm.

- $\mathtt{Hyb}_0$**:** Real world.

- $\mathtt{Hyb}_1$**:** Same as $\mathtt{Hyb}_0$, except the hash key k is computed by programming the hash function as follows $k \leftarrow \mathcal{H}.\mathsf{Gen}(1^\kappa) | H(\mathsf{k}, \mathbf{a}) = \mathbf{e}$ for $\mathbf{a} \leftarrow \Pi_{\mathsf{ZK}}.\mathcal{P}_1(1^\kappa, 1^n)$ and a random $\mathbf{e}$. The proof is computed by invoking the honest verifier on the witness.

The adversary cannot distinguish between the two hash keys due to indistinguishability between the two modes of the CI hash function. An adversary for the hash function can simulate either of the two views by running the honest prover algorithm using the witness w. A distinguisher for the hybrids successfully distinguishes between the CI modes.

– $\mathtt{Hyb}_2$**:** Same as $\mathtt{Hyb}_1$, except the simulator invokes the ZK simulator $\mathcal{S}_{\mathcal{V}}$ of Π_{ZK} with graph G instead of $(\Pi_{\mathsf{ZK}}.\mathcal{V}_1, \Pi_{\mathsf{ZK}}.\mathcal{V}_2)$ to compute $(\mathbf{a}, \mathbf{e}, \mathbf{z})$.

In $\mathtt{Hyb}_2$ the first message $\mathbf{a}$ always contains a cycle and in $\mathtt{Hyb}_3$ $\mathbf{a}$ (which contains commitments to all 1) also contains a cycle. The CI output is indistinguishable. Thus, indistinguishability between the hybrids follow from the adaptive zero knowledge property of Π_{ZK} since the distribution of $\mathbf{e}$ is random.

– $\mathtt{Hyb}_3$**:** Same as $\mathtt{Hyb}_2$, except the hash key k is generated in statistical mode $\mathsf{k} \leftarrow \mathcal{H}.\mathsf{StatGen}(1^\kappa, \mathcal{C}_{\mathsf{sk}})$.

The adversary cannot distinguish between the two hash keys due to indistinguishability between the two modes of the CI hash function. An adversary for the hash function can simulate either of the two simulated views by running the ZK simulator (of $pizk$) using the trapdoor td for the commitment scheme. A distinguisher for the hybrids successfully distinguishes between the CI modes.

$\mathcal{P}$ gets corrupted post-execution. When the prover gets corrupted post-execution, the simulator obtains the correct witness σ. It invokes the simulator $\mathcal{S}_{\mathcal{V}}$ of Π_{ZK} with σ to obtain randomness s.t. the proof $(\mathbf{a}, \mathbf{e}, \mathbf{z})$ is consistent with the statement G and the witness σ. An adaptive adversary cannot distinguish the opening of a simulated proof from a real one due to adaptive security of Π_{ZK}. This completes our proof of security for adaptive corruptions.

Proof of Adaptive Soundness. Our crs distribution is identical in real and ideal world execution. It contains the $\mathsf{crs}_{\mathsf{ZK}}$ $(= (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk}))$ and the description of the hash function H. The hash function is defined in the statistical mode and we know that Π_{ZK} is adaptively sound given Com is binding. If an adversary $\mathcal{A}$ breaks adaptive soundness of Π_{NIZK} then we can use $\mathcal{A}$ to construct an adversary $\mathcal{A}_{\mathsf{COM}}$ to break the binding property of Com. $\mathcal{A}_{\mathsf{COM}}$ obtains $\mathsf{crs}_{\mathsf{Com}}$ for the commitment and he samples a PKE key pair $(\mathsf{pk}, \mathsf{sk})$ to set the $\mathsf{crs}_{\mathsf{ZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$. $\mathcal{A}_{\mathsf{COM}}$ computes $\mathsf{k} \leftarrow \mathcal{H}.\mathsf{StatGen}(1^\kappa, \mathcal{C}_{\mathsf{sk}})$ given sk. $\mathcal{A}_{\mathsf{COM}}$ invokes $\mathcal{A}$ with $\mathsf{crs}_{\mathsf{NIZK}} = (\mathsf{k}, \mathsf{crs}_{\mathsf{ZK}})$ to obtain a proof. The adversary for the commitment scheme can check if $\exists i \in [\ell], \exists j \in [n^2]$, s.t. commitment c^i_j opens both to 0 and 1 by decrypting r^i_j and $r^{i\prime}_j$ from $E^i_{j,0}$ and $E^i_{j,1}$ given the secret key sk. He returns r^i_j and $r^{i\prime}_j$ to the challenger of Com to break the binding property. The only scenario when $\mathcal{A}$ succeeds and $\mathcal{A}_{\mathsf{COM}}$ fails to break the binding property is when $\mathcal{A}$ breaks the correlation intractability property of hash function H. The hash function is initialized in the statistical mode and its infeasible to compute an $\mathbf{e}$, s.t. $\mathbf{e} = f(\mathbf{a})$, unless $\mathcal{A}$ breaks the binding property of Com. Thus $\mathcal{A}_{\mathsf{COM}}$ always succeeds in breaking the binding property of Com if $\mathcal{A}$ succeeds in breaking adaptive soundness of Π_{NIZK}.

Proof of Adaptive ZK. The zero knowledge simulator for Π_{NIZK} in Fig. 9 is secure when the statement graph G is adaptively chosen based on the crs distribution. Adaptive zero knowledge of Π_{NIZK} follows from the adaptive zero knowledge property (Theorem. 4.1) of Π_{ZK}. □

Protocol Π_{NIZK} also implements $\mathcal{F}_{\mathsf{NIZK}}^{\mathsf{m}}$ (Fig. 2) functionality, i.e. a single prover can prove multiple statements using the same $\mathsf{crs}_{\mathsf{NIZK}}$, if the Π_{ZK} implements $\mathcal{F}_{\mathsf{ZK}}^{\mathsf{m}}$ (Fig. 4) functionality. The protocol also satisfies triple adaptive security for each proof. Our result is summarized in Thm. 5.2.

Figure 10: Triply Adaptively Secure Non-Interactive Zero Knowledge Protocol

Π_{NIZK}

- **Primitives:** Correlation Intractable hash family $\mathcal{H} = (\mathsf{Gen}, \mathsf{StatGen}, H)$.
- **Public Inputs:** Common reference string $\mathsf{crs}_{\mathsf{NIZK}} = (\mathsf{k}, \mathsf{crs}_{\mathsf{ZK}}) = (\mathsf{k}, \mathsf{crs}_{\mathsf{Com}}, \mathsf{pk})$. [a] where $\mathsf{k} \leftarrow \mathcal{H}.\mathsf{StatGen}(1^\kappa, C_{\mathsf{sk}})$. [b]
- **Private Inputs:** $\mathcal{V}$ has input graph G. $\mathcal{P}$ has input graph G and a valid Hamiltonian cycle σ.
- **Notations:** In Π_{ZK}, we denote the $\mathcal{P}$ algorithm as $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$, where $\mathcal{P}_1$ constructs $\mathbf{a}$ and $\mathcal{P}_2$ computes $\mathbf{z}$. Similarly, we denote $\mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2)$, where $\mathcal{V}_1$ computes $\mathbf{e}$ and $\mathcal{V}_2$ verifies the proof.

Prove : $\mathcal{P}(G, \sigma, 1^\kappa, 1^n)$

- Compute $\mathbf{a}$ by invoking $\mathcal{P}_1$ of Π_{ZK} as $(\mathbf{a}, \mathsf{st}_{\mathcal{P}}) = \Pi_{\mathsf{ZK}}.\mathcal{P}_1(1^\kappa, 1^n)$. Computes challenge $\mathbf{e} = H(\mathsf{k}, \mathbf{a})$ where $\mathbf{e} \in \{0,1\}^\ell$.
- Computes response by invoking $\mathcal{P}_2$ as $\mathbf{z} = \Pi_{\mathsf{ZK}}.\mathcal{P}_2((G, \sigma), \mathbf{e}, \mathsf{st}_p)$. $\mathcal{P}$ sends proof $\gamma = (\mathbf{a}, \mathbf{e}, \mathbf{z})$ to $\mathcal{V}$.

Verify : $\mathcal{V}(G, \gamma, 1^\kappa, 1^n)$

- $\mathcal{V}$ computes $\mathbf{e}'$ by computing $\mathbf{e}' = H(\mathsf{k}, \mathbf{a})$. $\mathcal{V}$ outputs Reject if $\mathbf{e} \neq \mathbf{e}'$. $\mathcal{V}$ outputs Accept if $\Pi_{\mathsf{ZK}}.\mathcal{V}_2(\gamma, \mathbf{e}) = 1$, else output Reject.

---

[a] $\mathsf{crs}_{\mathsf{Com}}$ of Com in Π_{ZK} and sk is the corresponding decryption key.

[b] $\mathcal{C}_{\mathsf{sk}}$ is a poly-size circuit computing the function $f_{\mathsf{sk}}(\mathbf{a}) = \mathbf{e}$, s.t. for every $i \in [\ell], e^i = 0$ iff $\mathsf{Ext}(\mathsf{sk}, a^i)$ is an n-node cycle.

Theorem 5.2. *If $\mathcal{H}$ is a somewhere statistically correlation intractable hash function family with programmability, Π_{ZK} is an input-delayed triply adaptive three round public-coin protocol implementing $\mathcal{F}_{ZK}^{m}$, then Π_{NIZK} UC-realizes $\mathcal{F}_{NIZK}^{m}$ for the Hamiltonian Language $\mathcal{R}_{Ham}$ against adaptive adversaries. Moreover, Π_{NIZK} is adaptively sound and adaptively zero knowledge.*

Proof. The setup string $\mathsf{crs}_{\mathsf{NIZK}}$ can be reused by the same prover to prove multiple statements if Π_{ZK} satisfies multi-proof in the $\mathsf{crs}_{\mathsf{ZK}}$ model. If a corrupt prover constructs a malicious proof then his witness can be extracted by invoking the simulator of Π_{ZK}. Similarly, if the prover is honest then the simulator of Π_{ZK} can be used to construct simulated proofs that are accepting. If the prover gets corrupted post-execution then the simulator of the multi-proof NIZK protocol obtains the corresponding witnesses. He invokes the simulator of Π_{ZK} with the witnesses to obtain the prover randomness for the simulated proofs. Adaptive security for multi-proof NIZK protocol follows from the adaptive security of Π_{ZK}. Adaptive soundness and adaptive zero knowledge of Π_{NIZK} also follows from the adaptive soundness and adaptive zero knowledge of Π_{ZK}. □

Our ZK protocol Π_{ZK} requires a public key encryption scheme with oblivious ciphertext sampleability and an equivocal commitment scheme. In addition, the Fiat-Shamir transform requires a CI hash function. We can obtain our NIZK from different assumptions based on the instantiation of these primitives.

5.1 Instantiation based on LWE assumption

By instantiating the hash function H [PS19], equivocal commitment [CsW19] and PKE [GSW13] from LWE we can obtain our NIZK protocol solely based on LWE assumption. We summarize our result in Thm. 5.3.

Theorem 5.3. *Assuming LWE assumption holds and the primitives in Π_{NIZK} are instantiated as mentioned above, then Π_{NIZK} UC-realizes $\mathcal{F}_{NIZK}$ for a* single instance *of the Hamiltonian Language $\mathcal{R}_{Ham}$ against adaptive adversaries. Furthermore, Π_{NIZK} is adaptively sound and adaptively zero knowledge.*

5.2 Instantiation based on LPN+DDH assumption

The work of [BKM20] demonstrated that the CI hash function can be replaced by a CI-Apx hash function for constant degree polynomial and the PKE scheme can be replaced by a statistically binding extractable commitment whose extractor can be represented by a constant degree polynomial. The recent work of [BKM20] showed that given a commitment scheme Com any commit-and-open Sigma protocol Π_{Com} can be converted to different commit-and-open Sigma protocol Π'_{Com} with the following properties and their results are summarized in Thm. 5.4.

- If Com is extractable then Π'_{Com} is a trapdoor Sigma protocol.
- If the commitment extraction function $f_{\mathsf{td}}(\mathbf{a}) = \mathsf{Com.Ext}(\mathsf{td}, \mathbf{a})$ has probabilistic constant-degree representation, then so does the trapdoor function $\mathsf{BadChallenge}$, corresponding to Π'_{Com} and, therefore $\mathcal{R}_\Sigma(\Pi_{\mathsf{Com}}')$ is probabilistically searchable by constant-degree polynomials.

Theorem 5.4 ([BKM20])**.** *Let $\Pi_{Com} = (\mathsf{Gen}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ be a trapdoor Sigma protocol for Language $\mathcal{L}$, where the output $\mathbf{a}$ of $\mathcal{P}_1$ is of length $\ell = \ell(\kappa)$. Let Com be a statistically-binding extractable commitment scheme where, for any td, the function $f_{\mathsf{td}}(x) = \mathsf{Com.Ext}(\mathsf{td}, x)$ has an ϵ-probabilistic representation by c-degree polynomials, for a constant $c \in \mathbb{N}$ and $0 < \epsilon(\kappa) < 1/\ell$. Then, for any polynomial $m := m(\kappa)$, there exists a trapdoor Sigma-protocol Π'_{Com} for $\mathcal{L}$ with soundness 2^{-m} such that $\mathcal{R}_\Sigma(\Pi_{Com}')$ is ϵ'-probabilistically searchable by $6cc'$-degree polynomials, where $c' \in \mathbb{N}$ is an arbitrary constant and $\epsilon' = \ell \cdot \epsilon + 2^{-c'}$.*

Next, they showed that the Fiat-Shamir transform can be applied using a correlation approx hash function H_{Apx} to obtain a NIZK protocol. Moreover, if Π_{Com} is input-delayed then the NIZK protocol satisfies adaptive soundness and adaptive zero knowledge.

Theorem 5.5 ([BKM20])**.** *(Sufficient Conditions for NIZK for NP). The following conditions are sufficient to obtain a NIZK argument system for NP with adaptive soundness and adaptive zero-knowledge from Π'_{Com}:*

1. *A trapdoor-Sigma protocol Π_{Com} satisfying delayed-input property.*
2. *A statistically-binding extractable commitment scheme where, for any td, the function $f_{\mathsf{td}}(x) = \mathsf{Com.Ext}(\mathsf{td}, x)$ has an ϵ-probabilistic representation by c-degree polynomials, for a constant $c \in \mathbb{N}$ and $0 < \epsilon(\kappa) < 1/\ell(\kappa)$ for an arbitrarily large polynomial ℓ.*
3. *A programmable correlation intractable hash family $\mathcal{H}_{Apx}$ for relations ϵ probabilistically searchable by c'-degree polynomials, for some constant $\epsilon > 0$ and arbitrarily large constant $c' \in \mathbb{N}$.*

They instantiate the correlation approx hash function H_{Apx} from trapdoor hash [DGI+19] under DDH, QR, DCR and LWE assumptions. The commitment scheme is instantiated under LPN assumption s.t. its extractor algorithm can be probabilistically represented by a constant degree polynomial. We refer to their paper [BKM20] for more details and formal definitions.

The Π_{ZK} (Fig. 7) protocol is a trapdoor Sigma protocol assuming the commitment (in Π_{ZK}) is binding and it also satisfies delayed-input property. The commitment scheme (in Π_{ZK}) is instantiated as Pedersen Commitment under DLP assumption. We replace the PKE (in Π_{ZK}) by the LPN-based commitment scheme of [BKM20]. Their LPN-based PKE scheme also satisfies oblivious ciphertext sampleability which we need for our security proofs. By applying the Fiat-Shamir using H_{Apx} as the hash function and applying the result of Thm. 5.5 we obtain a triply adaptive NIZK protocol from LPN+DDH assumption. We summarize our result in Thm. 5.6.

Theorem 5.6. *Assuming LPN+DDH assumption holds and the primitives in Π_{NIZK} are instantiated as mentioned above, then Π_{NIZK} UC-realizes $\mathcal{F}_{NIZK}$ for a* single proof *of the Hamiltonian Language $\mathcal{R}_{Ham}$ against adaptive adversaries. Furthermore, Π_{NIZK} is adaptively sound and adaptively zero knowledge.*

6 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 the security parameter 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 Fig. 11 and the overview can be found in Sec. 2.4. We prove triple adaptive security of Π_{sNIZK} by proving Thm. 6.1.

Theorem 6.1. *Assuming* PKE *is a public key encryption scheme with oblivious ciphertext sampling, Com is an equivocal additively homomorphic commitment scheme in the reusable crs_{Com} model and Π_{NIZK} implements $\mathcal{F}_{NIZK}$ against adaptive corruption of parties, then Π_{sNIZK} UC-securely implements $\mathcal{F}_{NIZK}$ functionality for NP languages against adaptive adversaries in the crs model where $|\mathsf{crs}| = poly(\kappa)$. In addition, Π_{sNIZK} is adaptively sound and adaptively zero knowledge.*

Proof. The work of [GOS12] showed that if the order of the message domain of Com is atleast 4 and α, β are the input values of a NAND gate and Γ is the output value then,

$$\Gamma = \alpha \odot \beta \text{ if and only if } \alpha + \beta + 2\Gamma - 2 \in \{0, 1\}$$

And if the order is 3 then

$$\Gamma = \alpha \odot \beta \text{ if and only if } \alpha + \beta + 2\Gamma - 2 \in \{0, 1\} \text{ and } \alpha + \beta + \Gamma - 1 \in \{0, 1\}.$$

We assume the order is 4 (for the sake of simplicity) and correctness for the NAND gates follows from this observation. It can be modified when the order is 3. Next, we present two possible corruption cases.

$\mathcal{P}$ is statically corrupt and $\mathcal{V}$ is honest. In this case, the adversary can break the soundness of the protocol if he can break the binding of the commitment or the soundness of Π_{NIZK}. The simulation algorithm is presented in Fig. 12. We present the formal hybrids and prove indistinguishability as follows:

Figure 11: Triply Adaptively Secure NIZK Protocol in the short crs model

Π_{sNIZK}

- **Primitives:** Non-interactive equivocal additively homomorphic commitment scheme $\mathsf{Com} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Ver}, \mathsf{Equiv})$. Public key encryption scheme $\mathrm{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with oblivious ciphertext sampling algorithm oEnc. Adaptively secure NIZK protocol $\Pi_{\mathsf{NIZK}} = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$.
- **Public Inputs:** Common reference string $\mathsf{crs}_{\mathsf{sNIZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk}, \mathsf{crs}_{\mathsf{NIZK}})$ where $(\mathsf{crs}_{\mathsf{Com}}, \mathsf{td}) \leftarrow \mathsf{Com.Gen}(1^\kappa)$, $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \Pi_{\mathsf{NIZK}}.\mathsf{Gen}(1^\kappa)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{PKE.KeyGen}(1^\kappa)$.
- **Notations:** If $x \in \mathcal{L}$ and w is a valid witness then $\mathcal{R}(x, w) = 1$. Circuit $\mathcal{C}$ computes $\mathcal{R}(x)$ s.t. $\mathcal{C}(y_1 y_2 \dots y_n) = 1$ iff $\mathcal{R}(x, w) = 1$ where $y_1 y_2 y_{|w|} = w$ and $\mathcal{C}$ has m NAND gates and n wires, where the nth wire is the output wire of $\mathcal{C}$. $\mathsf{Com.Com}(\alpha; r_\alpha) + \mathsf{Com.Com}(\beta; r_\beta) = \mathsf{Com.Com}(\alpha+\beta; r_\alpha + r_\beta)$.

Prove : $\mathcal{P}(x, w)$

- Assign the wire values $y_1 y_2 \dots y_n$ based on w.
- Commit to each bit y_i as $c_i \leftarrow \mathsf{Com.Com}(y_i; r_i)$ and encrypt the randomness as $E_{i,y_i} = \mathrm{PKE.Enc}(\mathsf{pk}, r_i; s_i)$ and $E_{i,\overline{y_i}} \leftarrow \mathrm{PKE.oEnc}(1^\kappa)$ for $i \in [n-1]$.
- For the output wire let $r_n = 0$ and $c_n = \mathsf{Com.Com}(1; 1)$.
- For all $i \in [n-1]$, prove c_i is a commitment to 0 or 1 by computing proof $\pi_i \leftarrow \Pi_{\mathsf{NIZK}}.\mathcal{P}((c_i, E_{i,0}, E_{i,1}), (w_i, r_i, s_i))$ for the following relation,

$$\mathcal{R}_1((c_i, E_{i,0}, E_{i,1}), (w_i, r_i, s_i)) = (\exists r_i, s_i : (c_i = \mathsf{Com.Com}(0; r_i) \wedge$$

$$E_{i,0} = \mathrm{PKE.Enc}(\mathsf{pk}, r_i; s_i)) \vee (c_i = \mathsf{Com.Com}(1; r_i) \wedge E_{i,1} = \mathrm{PKE.Enc}(\mathsf{pk}, r_i; s_i))).$$

- For the jth NAND gate with input wires α and β and output wire Γ perform the following:
  - Compute $C_j = c_\alpha + c_\beta + 2c_\Gamma - 2$.
  - Compute the randomness for C_j as $R_j = r_\alpha + r_\beta + 2r_\Gamma - 2$.
  - Prove C_j is a commitment to 0 or 1 by computing proof $\pi'_j \leftarrow \Pi_{\mathsf{NIZK}}.\mathcal{P}(C_j, R_j)$ for the following relation,

$$\mathcal{R}_2(C_j, R_j) = (\exists R_j : C_j = \mathsf{Com.Com}(0; R_j) \vee (C_j = \mathsf{Com.Com}(1; R_j))).$$

- $\mathcal{P}$ sends proof $\gamma = \{\{c_i, E_{i,0}, E_{i,1}, \pi_i\}_{i \in [n-1]}, c_n, \{\pi'_j\}_{j \in [m]}\}$ to $\mathcal{V}$.

Verify : $\mathcal{V}(x, \gamma)$

- Verifies $c_n = \mathsf{Com.Com}(1; 0)$.
- Verifies π_i for $i \in [n-1]$ using $\Pi_{\mathsf{NIZK}}.\mathcal{V}((c_i, E_{i,0}, E_{i,1}), \pi_i)$ algorithm.
- For the jth NAND gate with input wires α and β and output wire Γ compute $C_j = c_\alpha + c_\beta + 2c_\Gamma - 2$ and verify π'_j by running $\Pi_{\mathsf{NIZK}}.\mathcal{V}(C_j, R_j)$.
- If verification succeeds then output Accept else output Reject.

Figure 12: Simulation against a statically corrupt $\mathcal{P}^*$ by $\mathcal{S}$

- **Public Inputs:** Common reference string $\mathsf{crs}_{\mathsf{sNIZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk}, \mathsf{crs}_{\mathsf{NIZK}})$.
- **Simulator Inputs:** Trapdoor of $\mathsf{crs}_{\mathsf{sNIZK}} = (\mathsf{td}, \mathsf{sk}, \mathsf{td}_{\mathsf{NIZK}})$ where sk is the secret key for pk, td is the equivocating trapdoor of Com and $\mathsf{td}_{\mathsf{NIZK}}$ is the trapdoor of $\mathsf{crs}_{\mathsf{NIZK}}$.

Prove :
$\mathcal{P}^*$ sends proof $\gamma = \{\{c_i, E_{i,0}, E_{i,1}, \pi_i\}_{i\in[n-1]}, c_n, \{\pi'_j\}_{j\in[m]}\}$ to $\mathcal{V}$.

Verify : $\mathcal{S}(x, \gamma)$

- $\mathcal{S}$ runs the honest verifier algorithm to compute C_j for $j \in [m]$ and verify $c_n = \mathsf{Com}(1; 1)$.
- For $i \in [n]$, $\mathcal{S}$ aborts if $\mathrm{PKE.Dec}(\mathsf{sk}, E_{i,0})$ and $\mathrm{PKE.Dec}(\mathsf{sk}, E_{i,1})$ are valid decommitment randomness for c_i to 0 and 1 respectively.
- $\mathcal{S}$ invokes the simulator of Π_{NIZK} with $\mathsf{td}_{\mathsf{NIZK}}$ to verify $\{\pi_i\}_{i\in[n-1]}, \{\pi'_j\}_{j\in[m]}$ and extract $y_1 \dots y_{|w|}$.
- $\mathcal{S}$ invokes $\mathcal{F}_{\mathsf{NIZK}}$ with $w = y_1 \dots y_{|w|}$ to complete the simulation.

- $\mathtt{Hyb}_0$**:** Real world.

- $\mathtt{Hyb}_1$**:** Same as $\mathtt{Hyb}_0$, except the simulator decrypts $(r_{i,0}, r_{i,1})$ from $(E_{i,0}, E_{i,1})$ for all $i \in [n]$. He aborts if $\exists i \in [n]$ (resp. $\exists j \in [m]$) s.t c_i (resp C_j) opens to both 0 and 1 using randomness $r_{i,0}$ (resp. $R_{j,0}$) and $r_{i,1}$ (resp. $R_{j,1}$).

  The adversary can distinguish between the hybrids if he computes valid decommitments for both 0 and 1. In such a case, the simulator can extract the randomness for both decommitments and he can be used to break the binding of the commitment scheme.

- $\mathtt{Hyb}_2$**:** Same as $\mathtt{Hyb}_1$, except the simulator aborts if the simulator of Π_{NIZK} fails to extract a correct witness from a single proof.

  Indistinguishability follows from the multi-proof security of Π_{NIZK}.

$\mathcal{P}$ is honest and $\mathcal{V}$ is statically corrupt. Next, we demonstrate zero knowledge property of Π_{sNIZK} by relying on the ZK simulator of Π_{NIZK} and the equivocal property of Com. The simulation algorithm is presented in Fig. 13. We present the formal hybrids and prove indistinguishability as follows:

- $\mathtt{Hyb}_0$**:** Real world.

- $\mathtt{Hyb}_1$**:** Same as $\mathtt{Hyb}_0$, except the simulator constructs π_i and π'_j using the ZK simulator of Π_{NIZK}.

  Indistinguishability follows due to the multi-proof security of Π_{NIZK}.

- $\mathtt{Hyb}_2$**:** Same as $\mathtt{Hyb}_1$, except the simulator uses the equivocating trapdoor td (of Com) computes all the commitments c_i as commitment to 0 using randomness r_i and commitment to 1 using randomness r'_i. He encrypts r_i and r'_i to form $E_{i,0}$ and $E_{i,1}$ respectively.

  Indistinguishability follows from the equivocal property of Com. The oblivious ciphertext sampling property of PKE also ensures that for every $i \in [n]$ the following two events are indistinguishable - one of $E_{i,0}$ and $E_{i,1}$ is obliviously sampled (real world view), both $E_{i,0}$ and $E_{i,1}$ are valid encryptions (ideal world view).

Figure 13: Simulation against a statically corrupt $\mathcal{V}^*$ by $\mathcal{S}$

– **Public Inputs:** Common reference string $\mathsf{crs}_{\mathsf{sNIZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk}, \mathsf{crs}_{\mathsf{NIZK}})$.
– **Simulator Inputs:** Trapdoor of $\mathsf{crs}_{\mathsf{sNIZK}} = (\mathsf{td}, \mathsf{sk}, \mathsf{td}_{\mathsf{NIZK}})$ where sk is the secret key for pk, td is the equivocating trapdoor of Com and $\mathsf{td}_{\mathsf{NIZK}}$ is the trapdoor of $\mathsf{crs}_{\mathsf{NIZK}}$.

Prove : $\mathcal{S}(x)$
– For $i \in [n-1]$, compute $c_i = \mathsf{Com.Com}(0; r_i) = \mathsf{Com}(1; r'_i)$. Compute $E_{i,0} = \mathrm{PKE.Enc}(\mathsf{pk}, r_i; s_i)$ and $E_{i,1} = \mathrm{PKE.Enc}(\mathsf{pk}, r'_i; s'_i)$. Compute $c_n = \mathsf{Com.Com}(1; 1)$.
– For $j \in [m], i \in [n-1]$, invoke Π_{NIZK} simulator with $\mathsf{td}_{\mathsf{NIZK}}$ to obtain simulated proofs for π_i and π'_j.
– Send proof $\gamma = \{\{c_i, E_{i,0}, E_{i,1}, \pi_i\}_{i\in[n-1]}, c_n, \{\pi'_j\}_{j\in[m]}\}$ to $\mathcal{V}$.

Verify :
Runs its own adversarial algorithm.

$\mathcal{P}$ gets corrupted post-execution. We discuss simulating the view of an honest prover when it gets adaptively corrupted post-execution. In such a case the adaptive simulator obtains the correct $y_1 \dots y_n$ and he equivocates the c_i s.t. they open to y_i. It claims that $E_{i,\overline{y_i}}$ was randomly sampled owing to the oblivious ciphertext sampling algorithm. π_i and π'_j are simulated by invoking the adaptive simulator for Π_{NIZK} using the respective witnesses. Indistinguishability follows due to the equivocal property of Com which holds in presence of adaptive corruptions, oblivious ciphertext sampling property of PKE and adaptive security for multi-proofs of Π_{NIZK}.

Proof of Adaptive Soundness. In our compiler, the setup string is $\mathsf{crs}_{\mathsf{sNIZK}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pk}, \mathsf{crs}_{\mathsf{Com}})$ where $\mathsf{crs}_{\mathsf{NIZK}}$ is the setup string of an adaptively sound NIZK protocol Π_{NIZK}. If an adversary $\mathcal{A}$ breaks adaptive soundness of Π_{sNIZK} then we can use $\mathcal{A}$ to construct an adversary $\mathcal{A}_{\mathsf{COM}}$, who breaks the binding property of Com, or an adversary $\mathcal{A}_{\mathsf{NIZK}}$, who breaks the adaptive soundness property of Π_{NIZK}. We construct $\mathcal{A}_{\mathsf{COM}}$ and $\mathcal{A}_{\mathsf{NIZK}}$ as follows:

- *Constructing $\mathcal{A}_{COM}$:* $\mathcal{A}_{\mathsf{COM}}$ obtains $\mathsf{crs}_{\mathsf{Com}}$ for the commitment and he samples a PKE key pair $(\mathsf{pk}, \mathsf{sk})$ and $\mathsf{crs}_{\mathsf{NIZK}}$ to set $\mathsf{crs}_{\mathsf{sNIZK}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pk}, \mathsf{crs}_{\mathsf{Com}})$. $\mathcal{A}_{\mathsf{COM}}$ invokes $\mathcal{A}$ to obtain a statement and a proof. $\mathcal{A}$ checks if $\exists i \in [n]$, s.t. commitment c_i opens both to 0 and 1 by decrypting r_i and r'_i from $E_{i,0}$ and $E_{i,1}$ given the secret key sk. If there is one such c_i then $\mathcal{A}_{\mathsf{COM}}$ outputs $(0, r_i)$ and $(1, r'_i)$ to the commitment challenger and breaks the binding property of Com. Else, $\mathcal{A}$ has broken the adaptive soundness of Π_{NIZK}.

- *Constructing $\mathcal{A}_{NIZK}$:* $\mathcal{A}_{\mathsf{COM}}$ obtains $\mathsf{crs}_{\mathsf{NIZK}}$ for Π_{NIZK}. He samples a PKE key pair $(\mathsf{pk}, \mathsf{sk})$ and $\mathsf{crs}_{\mathsf{Com}}$ to set $\mathsf{crs}_{\mathsf{sNIZK}} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pk}, \mathsf{crs}_{\mathsf{Com}})$. $\mathcal{A}_{\mathsf{COM}}$ invokes $\mathcal{A}$ to obtain a statement and a proof. $\mathcal{A}$ checks if $\exists i \in [n]$, s.t. commitment c_i opens both to 0 and 1 by decrypting r_i and r'_i from $E_{i,0}$ and $E_{i,1}$ given the secret key sk. If there is no such c_i then $\mathcal{A}$ has broken adaptive soundness in atleast one of the $\{\pi_i\}_{i\in[n-1]}$ or $\{\pi'_j\}_{j\in[m]}$. $\mathcal{A}_{\mathsf{NIZK}}$ samples one of those proofs randomly and returns it to the challenger of adaptive soundness game of Π_{NIZK}. If $\mathcal{A}$ wins with probability p and $\mathcal{A}_{\mathsf{COM}}$ wins with probability p_c then $\mathcal{A}_{\mathsf{NIZK}}$ wins with probability $\frac{p-p_c}{m+n-1}$.

Proof of Adaptive ZK. Our ZK simulator in Fig. 13 suffices for adaptive zero knowledge as the statement x is adaptively chosen by the adversary after obtaining $\mathsf{crs}_{\mathsf{sNIZK}}$. □

Protocol Π_{sNIZK} also implements $\mathcal{F}_{\mathsf{NIZK}}^{m}$ (Fig. 2) functionality and satisfies triple adaptive security for each proof. Our result is summarized in Thm. 6.2.

Theorem 6.2. *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* $\mathcal{F}_{\mathsf{NIZK}}^{m}$ *against adaptive corruption of parties, then* Π_{sNIZK} *UC-securely implements* $\mathcal{F}_{\mathsf{NIZK}}^{m}$ *functionality for NP languages against adaptive adversaries in the* crs *model where* $|\mathsf{crs}| = poly(\kappa)$*. In addition,* Π_{sNIZK} *is adaptively sound and adaptively zero knowledge.*

Proof. The protocol Π_{sNIZK} also allows the prover to prove multiple statements using the same $\mathsf{crs}_{\mathsf{sNIZK}}$. If a corrupt party breaks the security of the protocol in one of the proof then that party can be used to either break the multi-proof security of Π_{NIZK} or the security of Com. □

The homomorphic commitment scheme can be instantiated from DDH [Ped92, CSW20] or LWE [GVW15] asusmptions. The PKE can be instantiated from DDH assumption [ElG85] or LWE [GSW13] assumptions. This yields our compiler from DDH or LWE assumption.

7 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 binded 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. Our protocol is presented in Fig. 14. The same crs is used for multiple subsessions and this ensures adaptive soundness and adaptive zero knowledge. High-level overview can be found in Sec. 2.5.

We prove security of $\Pi_{\mathsf{UC\text{-}NIZK}}$ by proving Thm. 7.1.

Theorem 7.1. *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.*

Proof. We consider that there are s subsessions and out of that the prover is statically corrupt in s' of those subsessions and he gets adaptively corrupted in $s - s'$ subsessions. In s' of those subsessions the adversary can break the security of the protocol without breaking the security of the single-session NIZK protocol if he can forge a signature or if he can break the binding property of the simulation sound tag-based commitment scheme $\mathsf{Com}_{\mathsf{SST}}$. This is captured by a sequence of hybrids for each such subsession. For the rest $s - s'$ subsessions, the environment $\mathcal{Z}$ can distinguish a real and ideal world view either by distinguishing the views of the single-session NIZK protocol or by breaking the equivocal property of $\mathsf{Com}_{\mathsf{SST}}$(or oblivious ciphertext sampling property of PKE). In Fig. 15 we capture our ideal world simulation against adaptive corruption of parties for multi-subsessions. We consider $5s + 5$ hybrids where in hybrid $5j$ (for $j \in [s]$) the first $j - 1$ subsessions are simulated and the rest are real executions. The jth subsession is modified in hybrids $\mathtt{Hyb}_{5j+1}$- $\mathtt{Hyb}_{5j+3}$ if the prover is statically corrupted. Else, if the prover is honest then the simulator simulates on behalf of the prover and that is captured in hybrids $\mathtt{Hyb}_{5j+4}$ and

Figure 14: Triply Adaptive UC-Secure NIZK Protocol for multiple sessions in the short crs model

$\Pi_{\mathsf{UC\text{-}NIZK}}$

- **Primitives:** Strong one-time signature scheme $\text{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$. Public key encryption scheme $\text{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with oblivious ciphertext sampling algorithm oEnc. Simulation sound tag based commitment $\mathsf{Com}_{\mathsf{SST}} = (\mathsf{KeyGen}, \mathsf{Com}, \mathsf{Dec}, \mathsf{Ver}, \mathsf{TCom}, \mathsf{TOpen})$.
- **Public Inputs:** Common reference string $\mathsf{crs}_{\mathsf{UC\text{-}NIZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk}, \mathsf{crs}_{\mathsf{sNIZK}})$ where $(\mathsf{crs}_{\mathsf{Com}}, \mathsf{td}) \leftarrow \mathsf{Com}_{\mathsf{SST}}.\mathsf{KeyGen}(1^\kappa)$, $(\mathsf{crs}_{\mathsf{sNIZK}}, \mathsf{td}_{\mathsf{sNIZK}}) \leftarrow \Pi_{\mathsf{sNIZK}}.\mathsf{Gen}(1^\kappa)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \text{PKE}.\mathsf{KeyGen}(1^\kappa)$. Session id sid and subsession id ssid.

Prove : $\mathcal{P}(x, w, \mathsf{sid}, \mathsf{ssid})$

- Generates signature key pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \text{SIG}.\mathsf{KeyGen}(1^\kappa)$.
- Commits to the bits of witness w_i using the tag $(\mathsf{vk}, \mathsf{sid}, \mathsf{ssid}, x)$ as $c_i = \mathsf{Com}_{\mathsf{SST}}.\mathsf{Com}(\mathsf{crs}_{\mathsf{Com}}, w_i, (\mathsf{vk}, \mathsf{sid}, \mathsf{ssid}, x); r_i)$ using randomness r_i. Encrypt the randomness as $E_{i,r_i} = \text{PKE}.\mathsf{Enc}(\mathsf{pk}, r_i; s_i)$ and $E_{i,\overline{y_i}} \leftarrow \text{PKE}.\mathsf{oEnc}(1^\kappa)$ for $i \in [n-1]$.
- Computes proof $\pi \leftarrow \Pi_{\mathsf{sNIZK}}.\mathcal{P}((x, \mathsf{vk}, \{c_i, E_{i,0}, E_{i,1}\}_{i\in[|w|]}), (w, \mathsf{sk}, \{r_i, s_i\}_{i\in[|w|]}))$ for the following relation:

$$\mathcal{R}'(x', w') = ((x, \mathsf{vk}, \{c_i, E_{i,0}, E_{i,1}\}_{i\in[|w|]}), (w, \mathsf{sk}, \{r_i, s_i\}_{i\in[|w|]}) : \mathcal{C}(x, w) = 1 \wedge$$

$$(\forall i \in [|w|], c_i = \mathsf{Com}_{\mathsf{SST}}.\mathsf{Com}(\mathsf{crs}_{\mathsf{Com}}, w_i, (\mathsf{vk}, \mathsf{sid}, \mathsf{ssid}, x); r_i) \wedge$$

$$(E_{i,0} = \text{PKE}.\mathsf{Enc}(\mathsf{pk}, r_i; s_i) \vee E_{i,1} = \text{PKE}.\mathsf{Enc}(\mathsf{pk}, r_i; s_i))).$$

- Signs the proof $s \leftarrow \text{SIG}.\mathsf{Sign}(\mathsf{sk}, \pi, \mathsf{sid}, \mathsf{ssid}, x)$. Sends the proof $\gamma = (\pi, \{c_i, E_{i,0}, E_{i,1}\}_{i\in[|w|]}, s, \mathsf{vk})$.

Verify : $\mathcal{V}(x, \gamma, \mathsf{sid}, \mathsf{ssid})$

- The verifier outputs ACCEPT if $\Pi_{\mathsf{sNIZK}}.\mathcal{V}((x, \mathsf{vk}, \{c_i, E_{i,0}, E_{i,1}\}_{i\in[|w|]}, \pi) = 1$ and $\text{SIG}.\mathsf{Ver}(\mathsf{vk}, (\pi, x, \mathsf{sid}, \mathsf{ssid}), s) = 1$. Else, output REJECT.

$\mathtt{Hyb}_{5j+5}$. In $\mathtt{Hyb}_{5j+5}$ the first j subsessions are simulated and the rest $s - j$ subsessions consist of real executions.

- $\mathtt{Hyb}_0$**:** Real world execution of all the subsessions.

  $\vdots$

- $\mathtt{Hyb}_{5j}$**:** The simulator $\mathcal{S}$ simulates the first $j-1$ subsessions by invoking the simulator $\Pi_{\mathsf{sNIZK}}.\mathcal{S}$ and the rest $s - j + 1$ subsessions are simulated by running the real protocol with the honest prover's input witness.

- $\mathtt{Hyb}_{5j+1}$**:** Same as $\mathtt{Hyb}_{5j}$, except if the prover is statically corrupt in subsession j and the reduction aborts if the corrupt prover has successfully generated a signature s using vk in the proof for subsession j, where vk was used to generate a proof γ' by an honest prover in another subsession.

  The adversary $\mathcal{A}$ can distinguish between the two hybrids if he forges a signature s on $m = (\pi, x, \mathsf{sid}, \mathsf{ssid})$ using vk where vk has been used to generate a proof in a different NIZK subsession. One can build an adversary for the strong one-time signature scheme $\mathcal{A}_s$ using $\mathcal{A}$ where $\mathcal{A}_s$ outputs (m, s) as the forgery.

– $\mathtt{Hyb}_{5j+2}$**:** Same as $\mathtt{Hyb}_{5j+1}$, except if the prover is statically corrupt in subsession j and the reduction aborts if c_i (in the maliciously constructed proof for subsession j) opens to both 0 and 1 using randomness $r_{i,0}$ and $r_{i,1}$ from $E_{i,0}$ and $E_{i,1}$ respectively.

The adversary can distinguish between the hybrids if he computes valid decommitments for both 0 and 1. In such a case, the simulator can extract the randomness for both decommitments as the adversary can be used to break the the simulation soundness property of $\mathsf{Com}_{\mathsf{SST}}$ for the challenge tag $(\mathsf{vk}, \mathsf{sid}, \mathsf{ssid}, x)$.

– $\mathtt{Hyb}_{5j+3}$**:** Same as $\mathtt{Hyb}_{5j+2}$, except if the prover is statically corrupt in subsession j then the simulator constructs the ideal world view by invoking the single session simulator of Π_{sNIZK}.

Indistinguishability between $\mathtt{Hyb}_{5j+2}$ and $\mathtt{Hyb}_{5j+3}$ follows from the security of Π_{sNIZK}. This completes simulating the jth subsession for static corruption of prover.

– $\mathtt{Hyb}_{5j+4}$**:** Same as $\mathtt{Hyb}_{5j+3}$, except if the prover is honest in the beginning of susbsession j then the simulator simulates an honest prover by computing the commitments using TCom and encrypts the corresponding randomness values for 0 and 1. The proof is generated honestly using the knowledge of the witness and adaptive corruption of prover is simulated by opening the honest prover randomness.

Indistinguishability follows from the security of $\mathsf{Com}_{\mathsf{SST}}$ and oblivious ciphertext sampling property of PKE.

– $\mathtt{Hyb}_{5j+5}$**:** Same as $\mathtt{Hyb}_{5j+4}$, except if the prover is honest in the beginning of susbsession j then the simulator invokes $\Pi_{\mathsf{sNIZK}}.\mathcal{S}(x')$ to simulate the proof. If the prover gets corrupted post-execution with witness w then invoke the adaptive simulator of Π_{sNIZK} with w to obtain randomness consistent with the proof.

Indistinguishability between $\mathtt{Hyb}_{5j+4}$ and $\mathtt{Hyb}_{5j+5}$ follows from the adaptive security of Π_{sNIZK}. This completes simulating the jth subsession for adaptive corruption of prover.

$\vdots$

– $\mathtt{Hyb}_{5s+5}$**:** This is the ideal world execution of the protocol where all the subsessions are simulated. The ideal world adversary view consists of the combined ideal world views of all the subsessions and this is forwarded to the environment $\mathcal{Z}$.

Proof of Adaptive Soundness. If adversary $\mathcal{A}$ breaks adaptive soundness of $\Pi_{\mathsf{UC\text{-}NIZK}}$ for a subsession j then one can either build an adversary $\mathcal{A}_{\mathsf{S}}$, forging signature, or an adversary $\mathcal{A}_{\mathsf{COM}}$, breaking soundness of $\mathsf{Com}_{\mathsf{SST}}$, or an adversary $\mathcal{A}_{\mathsf{sNIZK}}$ breaking adaptive soundness of Π_{sNIZK} as follows:

- *Constructing $\mathcal{A}_{\mathsf{S}}$:* $\mathcal{A}_{\mathsf{S}}$ initiates a multi-session UC protocol with $\mathcal{A}$. $\mathcal{A}_{\mathsf{S}}$ samples $(\mathsf{pk}, \mathsf{sk})$, $\mathsf{crs}_{\mathsf{Com}}$ and $\mathsf{crs}_{\mathsf{sNIZK}}$ s.t. he knows the trapdoors of $\mathsf{crs}_{\mathsf{UC\text{-}NIZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk}, \mathsf{crs}_{\mathsf{sNIZK}})$. $\mathcal{A}$ obtains $\mathsf{crs}_{\mathsf{UC\text{-}NIZK}}$ as the setup string. $\mathcal{A}_{\mathsf{S}}$ has oracle access to obtain signatures using challenge verification key vk. He incorporates the vk as part of the tag in a subsession $j' \neq j$, where the prover is honest. $\mathcal{A}_{\mathsf{S}}$ computes the proof for subsession j' using trapdoors of $\mathsf{crs}_{\mathsf{Com}}$ and $\mathsf{crs}_{\mathsf{sNIZK}}$. $\mathcal{A}_{\mathsf{S}}$ signs the proof using oracle access to the signing algorithm. $\mathcal{A}_{\mathsf{S}}$ sends this proof

to $\mathcal{A}$ as the proof for subsession j'. $\mathcal{A}_{\mathsf{S}}$ simulates other subsessions where the prover is honest, by invoking the ZK simulator using the trapdoors of $\Pi_{\mathsf{UC\text{-}NIZK}}$. $\mathcal{A}$ returns a statement and a proof for subsession j where he breaks adaptive soundness. If $\mathcal{A}$ has used the same vk as part of the proof in subsession j then $\mathcal{A}_{\mathsf{S}}$ forwards this proof as a forgery using vk to the signing algorithm. The proofs in subsession j and subsession j' are ensured to be different because the subsession ids are different and hence the tags are different for each subsession. Thus, the proof for subsession j with same vk will act as a forgery.

- *Constructing $\mathcal{A}_{COM}$:* $\mathcal{A}_{\mathsf{COM}}$ initiates a multi-session UC protocol with $\mathcal{A}$. $\mathcal{A}_{\mathsf{COM}}$ receives $\mathsf{crs}_{\mathsf{Com}}$ from the challenger of the binding game. $\mathcal{A}_{\mathsf{COM}}$ samples $(\mathsf{pk}, \mathsf{sk})$ and $\mathsf{crs}_{\mathsf{sNIZK}}$ and sets $\mathsf{crs}_{\mathsf{UC\text{-}NIZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk}, \mathsf{crs}_{\mathsf{sNIZK}})$. $\mathcal{A}$ obtains $\mathsf{crs}_{\mathsf{UC\text{-}NIZK}}$ as the setup string. If $\mathcal{A}$ breaks adaptive soundness of subsession j then $\mathcal{A}_{\mathsf{COM}}$ checks if $\exists i \in [|w|]$, s.t. commitment c_i opens both to 0 and 1 by decrypting r_i and $r_i{}'$ from $E_{i,0}$ and $E_{i,1}$ given the secret key sk. He returns r_i and $r_i{}'$ to the challenger of Com to break the binding property.

- *Constructing $\mathcal{A}_{sNIZK}$:* $\mathcal{A}_{\mathsf{sNIZK}}$ obtains $\mathsf{crs}_{\mathsf{sNIZK}}$ from the challenger of adaptive soundness for Π_{sNIZK}. $\mathcal{A}_{\mathsf{sNIZK}}$ initiates a multi-session UC protocol with $\mathcal{A}$. $\mathcal{A}_{\mathsf{sNIZK}}$ samples $(\mathsf{pk}, \mathsf{sk})$ and $\mathsf{crs}_{\mathsf{Com}}$ and sets $\mathsf{crs}_{\mathsf{UC\text{-}NIZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk}, \mathsf{crs}_{\mathsf{sNIZK}})$. $\mathcal{A}$ obtains $\mathsf{crs}_{\mathsf{UC\text{-}NIZK}}$ as the setup string. $\mathcal{A}_{\mathsf{sNIZK}}$ simulates subsessions where the prover is honest by invoking the ZK simulator using the trapdoors of $\mathsf{crs}_{\mathsf{Com}}$. If $\mathcal{A}$ returns a statement and a proof for subsession j then $\mathcal{A}_{\mathsf{sNIZK}}$ checks that $\nexists i \in [|w|]$, s.t. commitment c_i opens both to 0 and 1 by decrypting r_i and $r_i{}'$ from $E_{i,0}$ and $E_{i,1}$ given the secret key sk. Also, $\mathcal{A}_{\mathsf{sNIZK}}$ verifies that the vk contained in the tag for session j is not used in any previous subsessions where the prover was honest. These two checks rule out the possibility of $\mathcal{A}$ breaking the binding property of Com and forging a signature. After these checks, it is ensured that $\mathcal{A}$ succeeds in breaking adaptive soundness in subsession j by breaking adaptive soundness of Π_{sNIZK}. $\mathcal{A}_{\mathsf{sNIZK}}$ forwards the statement and the proof (without the signature) forwarded by $\mathcal{A}$ in subsession j.

Proof of Adaptive Zero Knowledge. Our simulator for an honest prover provides adaptive zero knowledge even when a statically corrupt verifier adaptively chooses the statement to be proven based on the $\mathsf{crs}_{\mathsf{UC\text{-}NIZK}}$ distribution. □

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.

[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.

Figure 15: Simulation against adaptive corruption of parties in $\Pi_{\mathsf{UC\text{-}NIZK}}$

- **Public Inputs:** Common reference string $\mathsf{crs}_{\mathsf{UC\text{-}NIZK}} = (\mathsf{crs}_{\mathsf{Com}}, \mathsf{pk}, \mathsf{crs}_{\mathsf{sNIZK}})$.
- **Simulator Inputs:** Trapdoor of $\mathsf{crs}_{\mathsf{UC\text{-}NIZK}} = (\mathsf{td}, \mathsf{sk}, \mathsf{td}_{\mathsf{sNIZK}})$ where td is the trapdoor for $\mathsf{Com}_{\mathsf{SST}}$, sk is the secret key of pk and $\mathsf{td}_{\mathsf{sNIZK}}$ is trapdoor of $\mathsf{crs}_{\mathsf{sNIZK}}$.

The simulator $\mathcal{S}$ simulates multiple NIZK subsessions concurrently using the trapdoor td of crs. $\mathcal{S}$ maintains a list $\mathcal{L}$ of subsession ids, corresponding statements x'' and proofs with entries of the form - $(\mathsf{ssid}'', x'', \gamma'')$. Upon obtaining a request for ideal world adversary view in subsession ssid from the environment $\mathcal{Z}$ with statement x, the simulator $\mathcal{S}$ returns γ to $\mathcal{Z}$ if $(\mathsf{ssid}, x, \gamma) \in \mathcal{L}$. If $(\mathsf{ssid}, x'', \_) \in \mathcal{L}$ and $x \neq x'' : \mathcal{S}$ returns $\perp$ to $\mathcal{Z}$. Else, $\mathcal{S}$ generates the ideal world view as described below for ssid based on the corruption of prover.

Case 1: If the prover is statically corrupt in subsession ssid :

Prove :
$\mathcal{S}$ invokes the dummy adversary for subsession ssid with statement x to obtain $\gamma = (\pi, \{c_i, E_{i,0}, E_{i,1}\}_{i\in[|w|]}, s, \mathsf{vk})$.

Verify : $\mathcal{S}(x, \gamma, \mathsf{sid}, \mathsf{ssid})$

- Abort if vk was the verification key of an honestly generated proof in a different subsession $\mathsf{ssid}' \neq \mathsf{ssid}$ and $\textsc{Sig}.\mathsf{Ver}(\mathsf{vk}, (\pi, x, \mathsf{sid}, \mathsf{ssid}), s) = 1$.
- For $i \in [|w|]$, $\mathcal{S}$ aborts if $\mathrm{PKE}.\mathsf{Dec}(\mathsf{sk}, E_{i,0})$ and $\mathrm{PKE}.\mathsf{Dec}(\mathsf{sk}, E_{i,1})$ are valid decommitment randomness for c_i to 0 and 1 respectively.
- Extracts witness w by invoking the simulator algorithm for Π_{sNIZK} with trapdoor $\mathsf{td}_{\mathsf{sNIZK}}$ of $\mathsf{crs}_{\mathsf{sNIZK}}$ for statement $x' = (x, \mathsf{vk}, \{c_i, E_{i,0}, E_{i,1}\}_{i\in[|w|]})$, i.e. $w = \Pi_{\mathsf{sNIZK}}.\mathcal{S}(x', \pi)$.
- If $w = \perp$ then output $\textsc{Reject}$. Else, invoke $\mathcal{F}_{\mathsf{NIZK}}$ with w and output $\textsc{Accept}$.

Case 2: If the prover is honest at the beginning of subsession ssid :

Prove : $\mathcal{S}(x, \mathsf{sid}, \mathsf{ssid})$

- Generates signature key pair $(\mathsf{sk}, \mathsf{vk})$.
- For $i \in [|w|]$, $\mathcal{S}$ computes $(c_i, r_i') \leftarrow \mathsf{Com}_{\mathsf{SST}}.\mathsf{TCom}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{td}, (\mathsf{vk}, x, \mathsf{sid}, \mathsf{ssid}))$.
- Compute randomness $r_{i,0} \leftarrow \mathsf{Com}_{\mathsf{SST}}.\mathsf{TOpen}(\mathsf{crs}_{\mathsf{Com}}, c, 0, r_i')$ and $r_{i,1} \leftarrow \mathsf{Com}_{\mathsf{SST}}.\mathsf{TOpen}(\mathsf{crs}_{\mathsf{Com}}, c, 1, r_i')$. Encrypt $E_{i,0} = \mathrm{PKE}.\mathsf{Enc}(\mathsf{pk}, r_{i,0})$ and $E_{i,1} = \mathrm{PKE}.\mathsf{Enc}(\mathsf{pk}, r_{i,1})$.
- $\mathcal{S}$ invokes $\pi \leftarrow \Pi_{\mathsf{sNIZK}}.\mathcal{S}(x')$ with trapdoors $\mathsf{td}_{\mathsf{sNIZK}}$ for statement $x' = (x, \mathsf{vk}, \{c_i, E_{i,0}, E_{i,1}\}_{i\in[|w|]})$ to obtain π.
- Signs the proof $s \leftarrow \textsc{Sig}.\mathsf{Sign}(\mathsf{sk}, \pi, \mathsf{sid}, \mathsf{ssid}, x)$.
- $\mathcal{S}$ sends the proof $\gamma = (\pi, \{c_i, E_{i,0}, E_{i,1}\}_{i\in[|w|]}, s, \mathsf{vk})$ to verifier.

Verify :
Performs its own algorithm.

Post-Execution Corruption of Prover :
$\mathcal{S}$ computes the correct decommitment randomness r_{i,w_i} and its encryption randomness s_i. It invokes the adaptive simulator $\Pi_{\mathsf{sNIZK}}.\mathcal{S}$ on $(w, \mathsf{sk}, r_{i,w_i}, s_i)$ to obtain randomness for π. It opens c_i and E_{i,w_i} using randomness r_{i,w_i} and s_i respectively and claims that $E_{i,\overline{w_i}}$ is randomly sampled.

[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.

[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.

[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.

[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+16a] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Improved OR-composition of sigma-protocols. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 112–141. Springer, Heidelberg, January 2016.

[CPS+16b] 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 Clemente Galdi and Vladimir Kolesnikov, editors, *Security and Cryptography for Networks - 12th International Conference, SCN 2020, Amalfi, Italy, September 14-16, 2020, Proceedings*, volume 12238 of *Lecture Notes in Computer Science*, pages 670–690. Springer, 2020.

[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.

[CSW20] Ran Canetti, Pratik Sarkar, and Xiao Wang. Efficient and round-optimal oblivious transfer and commitment with adaptive security. Cryptology ePrint Archive, Report 2020/545, 2020. https://eprint.iacr.org/2020/545 (To appear in Asiacrypt'20).

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

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 3–32. Springer, Heidelberg, August 2019.

[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.

[DPR16] Ivan Damgård, Antigoni Polychroniadou, and Vanishree Rao. Adaptively secure multi-party computation from LWE (via equivocal FHE). In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part II*, volume 9615 of *LNCS*, pages 208–233. Springer, Heidelberg, March 2016.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[FF02] Marc Fischlin and Roger Fischlin. The representation problem based on factoring. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 96–113. Springer, Heidelberg, February 2002.

[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.

[GMY06] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, 19(2):169–209, April 2006.

[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.

[JKO13] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 955–966. ACM Press, November 2013.

[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.

[MY04] Philip D. MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 382–400. Springer, Heidelberg, May 2004.

[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.

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Heidelberg, August 1993.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

[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.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990.

[SCO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 566–598, 2001.

A Calculation for computing occurrence of bad event

The simulator fails to extract the correct witness when the majority of the candidate cycles are incorrect. A corrupt $\mathcal{P}^*$ can compute μ of the a^i values, except with negligible probability $2^{-\mu}$. We can assume that $e_i = 1$ for those bad μ instances. In such a case, the simulator needs to have $\mu+1$ good instances where $e_i = 1$. So there should be at least $2\mu+1$ instances where $e_i = 1$. That means that the challenge vector $\mathbf{e}$ should contain $2\mu+1$ 1s. Such an event occurs with overwhelming probability $(1 - 2^{-2\mu})$ when $\ell = 8\mu$. The probability that $\mathbf{e}$ does not contain $2\mu + 1$ 1s is given as follows. We denote the hamming weight of $\mathbf{e}$ as $\mathsf{HW}(\mathbf{e})$:

$$
\begin{aligned}
\Pr[\mathsf{HW}(\mathbf{e}) < (2\mu+1)] &= \frac{\binom{\ell}{1} + \ldots + \binom{\ell}{2\mu}}{2^{\ell}} \\
&= \frac{\binom{8\mu}{1} + \ldots + \binom{8\mu}{2\mu}}{2^{8\mu}} \\
&< 2\mu \times \frac{\binom{8\mu}{2\mu}}{2^{8\mu}} \\
&= \frac{2\mu}{2^{8\mu}} \times \binom{8\mu}{2\mu} \\
&= \frac{2\mu}{2^{8\mu}} \times \frac{8\mu!}{2\mu! \times 6\mu!} \\
&\approx \frac{2\mu}{2^{8\mu}} \times \frac{\sqrt{2\pi 8\mu}(\frac{8\mu}{e})^{8\mu}}{\sqrt{2\pi 2\mu}(\frac{2\mu}{e})^{2\mu}\sqrt{2\pi 6\mu}(\frac{6\mu}{e})^{6\mu}} \\
&\quad \text{(Due to Stirling's approximation)} \\
&= \frac{1}{2^{8\mu}} \times \sqrt{\frac{4\mu}{3\pi}} \times \frac{8^{8\mu}}{2^{2\mu} \times 6^{6\mu}} \\
&= \sqrt{\frac{4\mu}{3\pi}} \times (0.35)^{\mu} \\
&< 2^{-\mu}
\end{aligned}
$$