Universally Composable Non-Interactive Zero-Knowledge from Sigma Protocols via a New Straight-line Compiler

Megan Chen[1]*, Pousali Dey[2], Chaya Ganesh[3], Pratyay Mukherjee[4], Pratik Sarkar[4], and Swagata Sasmal[2]

[1] Boston University
megchen@bu.edu
[2] Indian Statistical Institute
deypousali95@gmail.com, swagata.sasmal@gmail.com
[3] Indian Institute of Science
chaya@iisc.ac.in
[4] Supra Research
pratyay85@gmail.com, iampratiksarkar@gmail.com

Abstract. Non-interactive zero-knowledge proofs (NIZK) are essential building blocks in threshold cryptosystems like multiparty signatures, distributed key generation, and verifiable secret sharing, allowing parties to prove correct behavior without revealing secrets. Furthermore, universally composable (UC) NIZKs enable seamless composition in the larger cryptosystems. A popular way to construct NIZKs is to compile interactive protocols using the Fiat-Shamir transform. Unfortunately, Fiat-Shamir transformed NIZK requires rewinding the adversary and is not *straight-line extractable*, making it at odds with UC. Using Fischlin's transform gives straight-line extractability, but at the expense of many repetitions of the underlying protocol leading to poor concrete efficiency and difficulty in setting parameters.

In this work, we propose a simple new transform that compiles a Sigma protocol for an algebraic relation into a UC-NIZK protocol *without any overheads of repetition*.

- Given a Sigma protocol for proving m algebraic statements over n witnesses, we construct a compiler to transform it into a *straight-line extractable* protocol using an additively homomorphic encryption scheme (AHE). Our prover executes the Sigma protocol's prover once and computes $2n$ encryptions. The verification process involves running the Sigma protocol verifier once and then computing n encryptions, which are homomorphically verified against the prover generated encryptions.
- We apply the Fiat-Shamir transform to the above straight-line extractable Sigma protocol to obtain a UC-NIZK. We instantiate AHE using class group based encryption where the public key of the encryption scheme is obliviously sampled using a suitable hash function. This yields a UC-NIZK protocol in the random oracle model.

Keywords: Zero Knowledge Proofs, Universal Composability, Sigma Protocols

* The author is supported by DARPA under Agreement No. HR00112020023.

Table of Contents

1 Introduction

Non-interactive zero-knowledge proofs (NIZK) [BFM90, BSMP91, FLS99] are used to enforce honest behavior and are an important building block in the design of cryptographic protocols like anonymous credentials [RWGM23], threshold signatures [BLS01, KG20, Lin22], distributed key generation [GJKR99, CS04, CD24, KMM$^+$23], and multi-party computation in general. Typically, NIZKs are analyzed in the standalone setting, where the security is proven by showing individual properties separately such as completeness, zero-knowledge, and (knowledge) soundness under a setup assumption (like a common reference string (CRS) or the Random Oracle Model (ROM)). This standalone security guarantee often does not suffice in applications where composability guarantees are expected: NIZKs run concurrently in arbitrarily many sessions. The Universal Composability (UC) framework [Can01] allows for the modular analysis of cryptographic protocols guaranteeing security in the presence of arbitrarily many sessions running concurrently thereby facilitates easy composability .

NIZKs in the ROM. A common design methodology for constructing NIZKs is to construct a public-coin interactive argument, prove zero-knowledge and knowledge-soundness, and then compile this interactive argument into a NIZK in the ROM. A large class of protocols that render themselves to such compilation into NIZKs are *Sigma protocols*. A Sigma protocol is a three-round interactive proof between a prover and a verifier, both possessing a statement x, and additionally, the prover has a secret witness w. A Sigma protocol proceeds in three rounds, where the prover sends a first message (commitment) a to the verifier, the verifier sends a random string c (the challenge), and finally the prover responds with a last message z (the response). The verifier accepts or rejects the claim using the transcript (x, a, c, z). Sigma protocols satisfy two main properties: (i) special soundness, a form of knowledge soundness that guarantees that an extractor can output a valid witness for x given two accepting proofs with the same initial message a but distinct challenges, that is, (x, a, c, z) and $(x, a, c', z'), c \neq c'$ and, (ii) honest-verifier zero knowledge (HVZK), a weak form of the zero-knowledge property that guarantees simulatability of the transcript given randomly sampled challenge c. Additionally, since the verifier's message is a uniformly random string, this is public-coin, making Sigma protocols amenable to compilation into NIZKs. A popular method is the Fiat-Shamir (FS) transform [FS87]: the prover non-interactively computes the challenge $c = \mathcal{H}(x, a)$ by applying a hash function $\mathcal{H}$ (modeled as a random oracle [CJS14]) on (x, a). The NIZK proof sent to the verifier is (x, a, c, z) who checks if $\mathcal{H}(x, a) \overset{?}{=} c$, and then runs the Sigma protocol verifier. Knowledge-soundness of the transformed NIZK relies on the special-soundness of the Sigma protocol and therefore requires an extractor to rewind the malicious prover in order to obtain two transcripts with a shared prefix by programming the RO to $\mathcal{H}(x, a) = c'$ after rewinding. Zero-knowledge of the NIZK follows from HVZK of the Sigma protocol and programming the RO. Applications of Sigma protocols are plenty [FS87, CDS94, DG03, Mau09, SV12, FKMV12, ORV14] as many algebraic languages admit very efficient Sigma protocols, such as Schnorr [Sch91], Chaum-Pederson [CP93] etc. Moreover, compilers are known for expressive languages [Mau09, CDS94]. FS-transformed Sigma protocols are widely used in practice in signature generation [Sch91], signature aggregation [Ks22, CGKN21], proof of correct decryption in threshold cryptosystems, and distributed key generation [GJKR99, CS04, KMM$^+$23] – many of these achieve UC security assuming that the underlying NIZK is UC secure. A natural and pertinent question, then, is whether this large and useful class of Sigma protocols can be compiled into NIZKs that can be shown to be UC-secure?

Proving Sigma-compiled NIZK UC-secure. We now discuss the technical challenges in compiling Sigma protocols to UC-secure NIZKs.

Challenge 1: Straight-line Extraction. In the UC framework, the environment $\mathcal{Z}$, representing all that is external to the execution of the concerned protocol, interacts with the protocol, and outputs a decision bit in the end, indicating its guess of whether it interacted with a "real" adversary $\mathcal{A}$ and parties in the protocol, or with an "ideal" adversary (or simulator) Sim and parties accessing the ideal functionality $\mathcal{F}$ that specifies the ideal outcome of the protocol. NIZKs that rely on rewinding or non-black-box access to the adversary in the proof (either for ZK or for extraction) are at odds with UC. This is because in the UC definition, the environment $\mathcal{Z}$ is an interactive distinguisher between the real protocol and the ideal process, and therefore a simulator Sim in the security proof cannot rewind $\mathcal{Z}$, and does not have the concrete code of $\mathcal{Z}$. Thus, a crucial property that NIZKs must have in order to be compatible with a proof of UC is *black-box straight-line simulation and extraction.* At a high level, a NIZK in the ROM is straight-line extractable if an extractor succeeds given only the transcript that includes the RO queries made by the prover (and crucially without interacting with any successful prover). This immediately precludes FS-compiled Sigma protocols from being shown UC-secure. There exist alternatives to Fiat-Shamir that provide compilers [Pas03, Fis05] in the ROM whose resulting NIZK is straight-line extractable. However, this comes at a cost in efficiency and complexity in design:[5] Pass's compiler [Pas03] requires repeating the underlying Sigma protocol for security parameter (κ) number of times where κ is the computational security parameter and is as high as $\kappa = 128$, leading to a $128\times$ overhead over the FS-compiled NIZK (that relies on rewinding). Fischlin's compiler [Fis05] partially addressed this issue. Here, a *proof-of-work* paradigm is used where the prover is forced to compute several valid proofs which forces the prover to query many "good" values to the random oracle in order to find a pre-image that hashes to a zero string. For knowledge-soundness, the extractor can succeed by observing the RO queries, even though not all of the query/responses make it to the proof (this does away with the overhead in proof size). Fischlin's transform is known to improve over the Pass transform as shown in [CL24, Ks22]. However, choosing the optimal proof-of-work parameters is challenging in practice since it depends on the prover's computation power. Additionally, Fischlin transform also requires repetition of the underlying Sigma protocol and when applied to the Schnorr protocol incurs a $15\times$ overhead as shown by [CL24]. Thus, FS is the most efficient transform incurring essentially no overhead in computation or communication, however, the extractor is rewinding. This motivates the question:

Can we construct a generic compiler that transforms a Sigma protocol into a NIZK with straight-line extraction (without incurring communication overhead of repetition or prover overhead of proof-of-work)?

We answer the above question in the affirmative and show a transform that compiles Sigma protocols for algebraic relations into straight-line-extractable NIZKs without incurring a repetition overhead or prover overhead.

Challenge 2: Simulation Extractability. Another important property crucial to realize UC security is *non-malleability (NM)* [DDN91]. In a malleability attack, an adversary can maul existing proofs observed during the protocol execution, and forge a proof on some statement for which they do not know the corresponding witness. Since $\mathcal{Z}$ may ask uncorrupted provers to produce proofs on arbitrary statement-witness pairs, the ability to maul proofs make Sim fail in extracting a witness, leading to $\mathcal{Z}$ successfully distinguishing between real execution and ideal process. Non-malleability is captured by *simulation-extractability* in the context of UC-NIZK [Sah99, DDO$^+$01, PR05, GMY06, JP14, FKMV12], and [Gro06] proved that simulation-extractability is necessary for UC.

FS-transformation of Sigma protocols are proven to be simulation-extractable [FKMV12], but are not straight-line and hence not UC-compatible. Among the transformations that yield straight-line extractable NIZKs, a randomized version of Fischlin's transform [Ks22] has been shown to be simulation-extractable, and thus UC-secure [LR22b]. In sum, among existing transformations of Sigma protocols to NIZKs: FS transform gives simulation-extractability but not straight-line extraction; Fischlin and Pass transforms give straight-line simulation extraction, but incur the overhead of repetition penalizing proof sizes in practical applications.

Can we transform a Sigma protocol into a UC-NIZK that incurs no overhead compared to FS-transformed NIZK?

We answer the above questions by showing that applying the FS transform on our *straight-line compiled* protocol yields a NIZK that is simulation-extractable, which we then prove is UC-secure.

[5] For example, since the soundness error is related to the number of repetition, one has to be careful in choosing the parameters, such as the number of repetitions.

1.1 Our Contributions

– Primarily we construct a compiler that compiles a Sigma protocol for an algebraic relation into a protocol with *straight-line extraction* using an additively homomorphic encryption (AHE) scheme in the CRS model. For proof of m algebraic statements with n witnesses, the overhead incurred by our compiler is $2n$ encryptions for the prover, and an overhead of $2n$ ciphertexts in the proof size. For proving instances where $m > n$, as it is indeed the case in practical applications of Chaum-Pedersen [CP93], this overhead is amortized away. We discuss such applications at the end of this section.
Our transformation to a straight-line extractable protocol is independently interesting since the extraction avoids "forking" the adversary [PS96], which in practice leads to a slack in tightness of the security reduction [JT20].
– Applying the Fiat-Shamir transform on this compiled Sigma protocol, we obtain a NIZK that is *straight-line extractable*, showing the first property needed for UC. We show that the Fiat-Shamir compiled NIZK of the transformed Sigma protocol satisfies *simulation-extractability.* Towards this, we show that our compiled NIZK satisfies a property called weak-unique response, that says that no adversary can generate two distinct accepting transcripts that share a common prefix. This notion has been used to prove simulation-extractability of Sigma protocols [FKMV12] and other multi-round protocols [GOP+22], and is also necessary for Fischlin's transform. While this is a natural notion towards non-malleability, not all Sigma protocols satisfy this.[6] Nevertheless, we prove that our straight-line transformed Sigma protocol does satisfy unique response, *even when* the underlying Sigma protocol does not enjoy this property. This is a distinct feature of our transform.
– Finally, we show that the resulting NIZK is UC-secure. Our analysis is in the local ROM. As elaborated next, this has a substantial application in several cryptographic protocols, that rely on random oracle based NIZKs.

Instantiation. We provide a concrete instantiation of the compiler by instantiating the encryption scheme using the additive homomorphic encryption scheme based on class groups [CL15, CLR24]. En route, we show two new properties of the class-group based encryption scheme, namely (i) homomorphic well-formedness, which ensures that if a random linear homomorphic computation of two strings in the ciphertext space is well-formed (that is decryptable), then the strings themselves are well-formed; (ii) oblivious sampleability of the public key, which allows sampling of the public key obliviously, without knowing the corresponding secret-key. This enables us to remove the uniform CRS from our straightline-extractable protocol by obliviously sampling the public key as part of the proof, yielding a NIZK only in the ROM (without CRS). The public key consists of two class-group elements. We obliviously sample them by hashing into class groups of unknown order using recently proposed hashing algorithms [CLR24, SBK24], compatible with random oracle.

Applications. All applications of Schnorr and Chaum-Pedersen [CP93] benefit from our compiler and achieve UC security without repetition of the underlying Sigma protocol. We outline some applications of Sigma protocols for algebraic relations where UC security is desired and our UC-NIZK can be used as a drop-in replacement.

– *Signature Protocols:* The works of [Lin22, KG20] consider multiparty threshold signatures based on Schnorr. These protocols are UC-secure to permit composition in larger systems. As building blocks they need proofs of knowledge for discrete logarithms which are instantiated using the Fischlin transform. Our UC-NIZK can replace the Fishclin-transformed Schnorr and result in a simpler and potentially more efficient protocol.
– *Public Key Infrastructure (PKI):* Many protocols, defined in the UC framework (e.g. DKG from class-group [KMM+23]), rely on a public key setup (or PKI) for encryptions with keys of the form $\mathsf{pk} = g^{\mathsf{sk}}$ – establishing this requires a NIZK proof of knowledge of sk in the exponent (a.k.a. Schnorr's proof) of g, a cyclic group element. Proving knowledge soundness of the standard Schnorr would require rewinding,

[6] Examples are Okamoto's protocol [Oka93], Sigma protocol for OR composition [CDS94] – our straight-line extractable transform convert these to satisfy unique response. A closer look reveals that when $m < n$, this may not be satisfied, as witnesses are hidden information theoretically by simple algebraic argument.

thus leaving a gap between the security argument and the desired UC security of the protocol. Using a UC-compatible NIZK, like ours, one instead get the UC security of the full protocol.

– *Distributed Verifiable Random Functions:* The work of [GLOW21] introduced distributed verifiable random functions (DVRF) based on BLS signatures and it required the proof of knowledge variant of Chaum-Pedersen's proof of equal discrete log for proving that the partial evaluations of the DVRF is correct. The recent work of [KMMM23] introduced output private DVRF, formalized in UC, and they also require the proof of knowledge variant of Chaum-Pedersen's proof for the same purpose. In addition, they need Schnorr's proof of discrete log as part of their input blinding. Currently, these protocols rely on rewinding the adversary to extract the witness. Our compiler can be used to make those proofs UC-secure.
– *Secure Content Moderation and Traceability:* The work of [TGL+19] constructs a secure content moderation protocol over encrypted messaging platforms like Signal. Under the hood, they require a Chaum-Pedersen proof and security relies on knowledge-of-exponent assumption. This is not composable due to the non-blackbox nature of the reduction. Using our UC-NIZK will result in a UC-secure content moderation protocol. Our UC-NIZK is also a candidate to be used in the end-to-end secure messaging protocol of [BGJP23] that only traces illegal content.

Our compiler is general enough to work with AND/OR compositions [CDS94, FHJ20] which yields better signature schemes [FHJ20]. We discuss this in Appendix E.

1.2 Related work

In this section, we describe the existing approaches to obtain straight line extraction in Sigma protocols and UC-NIZKs.

Fischlin Transform. The work of [Ks22] improved upon the original Fischlin transform by rerandomizing the prover's transcript in the transform. The recent works of [LR22b, LR22a] study necessary properties for UC-NIZK in the global ROM and show that randomized Fischlin transforms Sigma protocols into UC-NIZKs. The work of [GKO+23] constructs a compiler to lift any witness-succinct simulation-extractable NIZK into a witness-succinct UC-secure one in the global random oracle model using a Fischlin-like transform. However, all these protocols inherit the downside of Fischlin: they inherently require repetition of the underlying Sigma protocol and the number of repetitions increases to achieve stronger soundness. In contrast, our UC-NIZK protocol is statistically soundness by the correctness guarantee of the decryption procedure in the AHE scheme.

NIZKs in the CRS model. The work of [FLS99] constructed a NIZK protocol in the CRS model assuming trapdoor permutations. The initial work of [CGH98] proposed to instantiate the hash function in the Fiat-Shamir transform using Correlation intractable (CI) hash functions. Subsequent works [CCH+19, PS19] construct such CI hash functions for sparse relations from LWE in the CRS model. The work of [BKM20] construct CI hash for approximable relations from LPN+DDH assumptions and then construct NIZKs from it, which was improved to using only sub-exponential DDH in [JJ21]. The work of [CSW22] explored NIZKs that satisfy adaptive zero-knowledge, adaptive soundness and security against adaptive corruptions. Our NIZK protocol can also work in this paradigm by replacing the random oracle in the Fiat-Shamir transform with a CI-hash function and embedding the secret key of the encryption inside the CI-hash. However, this would heavily affect the performance and it would be of theoretical interest as computing the CI-hash function to generate the challenge string is an expensive task.

Non-black-box extractable NIZKs. Another line of work [GOS12, AF07, KNYY19, KNYY20] uses pairing-based techniques in bilinear groups to construct NIZKs, and NIZKs in [DDO+01, GOS06, Gro06] are UC-secure. These constructions either use specific assumptions over bilinear groups, where DDH is easy and popular protocols like Chaum-Pedersen cannot be instantiated; or are in idealized models (like AGM/GGM) [Sho97, FKL18], or use knowledge-type assumptions [Dam92]. These are incompatible with UC security, since knowledge assumptions are non-black-box and hence the extractor depends on the code of the adversary; or are limited in the class of adversaries considered (generic/algebraic). The UC-AGM framework [ABK+21] models composability in the AGM for algebraic adversaries, but incompatible with standard UC; and [KKK21] enable knowledge assumptions in larger protocols, but in a composition framework [Mau11] different from UC.

NIZK in the CRS+ROM. [Lin15] proposed a transform using a dual-mode commitment and a non-programmable random oracle to obtain zero-knowledge via using the secret trapdoor of the setup string, which was subsequently improved upon in [CPSV16]. However, both protocols do not consider proof of knowledge and hence fail to provide a UC-NIZK.

Other compilers for straightline-extractability. [GMY03] uses a technique similar to ours – of using PKE and having the corresponding decryption key as a trapdoor – to achieve straightline-extractability. However, the key difference is our usage of the homomorphic property (over both witness and randomness) of the encryption scheme, which enables checking the linear relation of the Sigma protocol homomorphically. In contrast, [GMY03] use a generic Sigma protocol proof to attest that the encrypted witness satisfies the given relation. Therefore, our protocol would be concretely more efficient. Moreover, [GMY03] augments the relation with a signature scheme (associated PoK of signature augmented in the proof) in order to get simulation-soundness, whereas our Fiat-Shamir-compiled NIZK already achieves simulation extractability, without any additional augmentation.

The work of [Kat21] constructs a straightline extractable NIZK for proving the possession of a short vector $e \in \mathbf{R}_q^m$ such that $Ae = u$ for a given random matrix $A \in \mathbf{R}_q^{n \times m}$ and vector $u \in \mathbf{R}_q^n$ for appropriate parameters n, m and q. Similar to our compiler, they also utilize an (extractable) lattice commitment scheme that is linearly homomorphic over polynomial ring $\mathbf{R}_q$ to transform Lyubashevsky's [Lyu12] Sigma protocol for the above lattice relation to a straightline extractable one. Then they apply the Fiat Shamir transform in the quantum random oracle model to make it a quantum-secure NIZK. They construct candidates for the commitment scheme based on the hardness of lattice problems: one based on the module learning with errors (MLWE) problem, and the other based on the MLWE and the decisional small matrix ratio (DSMR) problem. However, their compiler is specific to proving lattice relations (of the form $Ae = u$), and their commitment scheme only works over polynomial rings. It is unclear how to make it work for group-based NP statements without reducing the statement to one that is compatible with lattice relations. Converting the group-based NP statement to a corresponding lattice relation may not always be possible, or incur additional overheads. Our NIZK works "directly" for group-based statements, that is, it is designed for group-based statements and works with the native operations of the computation for the statement.

1.3 Paper Organization

We present the technical overview in Sec. 2. The necessary preliminaries are presented in Sec. 3. We present our straight-line extractable Sigma protocol for algebraic statements/arbitrary linear relations in Sec. 4. In Sec. 5, we apply the Fiat-Shamir transform to make it a NIZK, argue the UC-security of our NIZK, and provide a concrete instantiation based on class groups. Finally, we discuss the efficiency and applications of our UC-NIZK in Sec. 7.

2 Technical Overview

The goal of this work is to construct UC-NIZKs from Sigma protocols for arbitrary algebraic relations (in the exponent), without repetitions. For well-known Sigma protocols (e.g. the Schnorr, Okamoto, and Chaum-Pedersen protocols), compiling these protocols to be non-interactive is straightforward, via applying the Fiat-Shamir transform. However, a remaining technical difficulty is proving that these protocols are also straight-line extractable. Prior compilers include those of [Pas03, Fis05]. We take a totally new approach to design a straight-line compiler.

We proceed in two main steps, which we detail in the coming sections.

Step 1: Construct a non-interactive straight-line extractable NIZK protocol Π_{GenLin} for arbitrary linear relations, in the ROM, using additively-homomorphic encryption (AHE). Notably, this construction *does not* require access to a common reference/random string.
Step 2: Prove that Π_{GenLin} UC-realizes the non-interactive zero knowledge functionality $\mathcal{F}_{\mathsf{NIZK}}$.

For simplicity of exposition, we focus the technical overview on Schnorr's protocol, which is an example of an algebraic Sigma protocol, i.e. 3-move interactive argument between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, that checks a single linear relation. We remark that all discussion presented here generalizes to protocols with n witnesses and for checking m linear relations, but we refer the reader to Section 4 for full details.

2.1 Schnorr's Proof of Discrete Log

Schnorr's protocol. We begin by recalling Schnorr's protocol for the discrete log relation $\mathcal{R}_{\mathsf{DLog}}$. Specifically the relation is defined as:

$$\mathcal{R}_{\mathsf{DLog}} := \{\ (g^w, w) \mid g \in \mathbb{G} \ \wedge\ w \in \mathbb{Z}_q\ \} \quad ,$$

where $\mathbb{x} = (g, g^w)$, $\mathbb{w} = w$, $\mathbb{G}$ is a group of prime order q and g is a (fixed, public) generator for $\mathbb{G}$. Schnorr's protocol is the following:

– **Move 1 (commit):** $\mathcal{P}_1(\mathbb{x}, \mathbb{w}) \to a$:
  1. Sample $s \leftarrow \mathbb{Z}_q$ and compute $S := g^s$ and send $a := S$ to $\mathcal{V}$.
– **Move 2 (challenge):** $\mathcal{V}_1(\mathbb{x}, a) \to c$:
  1. Sample a random challenge $c \in \mathbb{Z}_q$, and send c to $\mathcal{V}$.
– **Move 3 (response):** $\mathcal{P}_2(\mathbb{x}, \mathbb{w}, (a, c)) \to z$:
  1. Compute $z := s + c \cdot w \in \mathbb{Z}_q$, and send z to $\mathcal{V}$.
– **Verification:** $\mathcal{V}_2(\mathbb{x}, (a, c, z)) \to b$:
  1. Parse $\mathbb{x}$ as a group element $W \in \mathbb{G}$ and $\mathbb{w}$ as $w \in \mathbb{Z}_q$.
  2. If $g^z = S \cdot W^c \in \mathbb{G}$, output 1. Otherwise, output 0.

In other words, the verifier accepts if $z = s + c \cdot w$ in the exponent. (This check is linear in w.)

Proving knowledge soundness of Schnorr's protocol. The standard proof of knowledge soundness for Schnorr's protocol relies on *rewinding* the prover and rerunning it on a different challenge c', generating two protocol transcripts that share the same first message: (a, c, z) and (a, c', z'). Then, the (knowledge soundness) extractor recovers the witness w, via computing $\mathbb{w} := \frac{z'-z}{c'-c} \in \mathbb{Z}_q$. Furthermore, knowledge extraction fails only when $c = c'$, which occurs with probability $\frac{1}{q}$ since c, c' are sampled uniformly from $\mathbb{Z}_q$.

Unfortunately, this extractor does not satisfy *straight-line* knowledge soundness, i.e. the ability to extract the witness $\mathbb{w}$ without rewinding the adversary. This is problematic for proving that Schnorr's protocol is UC-secure: the environment $\mathcal{Z}$ can distinguish that in the ideal world, the adversarial prover algorithm was rewound whereas in the real-world execution, there was no rewinding. Without extracting the correct witness in the ideal world, the simulator cannot complete the simulation.

Given this conundrum, other techniques like the Pass and Fischlin transforms [Pas03, Fis05] were proposed in the random oracle model. However, these approaches require repetitions of the base Sigma protocol (for soundness) and incur at least a 15× overhead [CL24]. Alternatively, one can consider using a knowledge assumption, i.e. given an accepting transcript knowledge of the witness is assumed. However, this approach also violates [KZM+15] UC-security as the simulator needs non-blackbox access to the adversary.

2.2 A New Straight-line Extractable Schnorr's Proof of Discrete Log from additively homomorphic encryptions (AHE)

We avoid the need for repetition and introduce a new simple compiler for making Schnorr's protocol straight-line extractable in the ROM. The key ingredient is using an additively-homomorphic encryption scheme, denoted $\mathsf{AHE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ to encrypt the witness. Note that it is not sufficient to use a commitment scheme, unless it is an extractable commitment; this is equivalent to using encryption.

Compiled protocol. We briefly describe the compiled scheme. The common reference string includes an encryption key ek for AHE and a (secret) trapdoor, which is the decryption key dk associated with ek. Then, the prover, given AHE, encrypts the NP witness $\mathbb{w} = w$ as ciphertext C_w and the randomness s as C_s. The prover sends the first message $S = g^s$ of the Schnorr's protocol and the two encryptions. The verifier runs the usual Schnorr's verification protocol and in addition, it runs the same check over the encryptions. The additively homomorphic property of AHE allows the verifier to check the linear relation over the encryptions. The protocol is as follows (with differences from baseline Schnorr notated in blue):

- **Setup:** A key pair $(\mathsf{ek}, \mathsf{dk})$ for AHE.
- **Move 1 (commit):** $\mathcal{P}(\mathsf{ek}, \mathbb{x}, \mathbb{w}) \to a$:
  1. Sample $s \leftarrow \mathbb{Z}_q$ and compute $S := g^s$.
  2. Sample encryption randomnesses r_s, r_w.
  3. Compute $C_s := \mathsf{Enc}(\mathsf{ek}, s; r_s)$ and $C_w := \mathsf{Enc}(\mathsf{ek}, w; r_w)$.
  4. Send $a := (S, C_s, C_w)$ to $\mathcal{V}$.
- **Move 2 (challenge):** $\mathcal{V}(\mathsf{ek}, \mathbb{x}, a) \to c$:
  1. Sample a random challenge $c \in \mathbb{Z}_q$, and send c to $\mathcal{P}$.
- **Move 3 (response):** $\mathcal{P}(\mathbb{x}, \mathbb{w}, (a, c)) \to z$:
  1. Compute $z := s + c \cdot w \in \mathbb{Z}_q$, and send z to $\mathcal{V}$.
  2. Compute $r_z := r_s + c \cdot r_w \in \mathbb{Z}_q$, and send (z, r_z) to $\mathcal{V}$.
- **Verification:** $\mathcal{V}(\mathsf{ek}, \mathbb{x}, (a, c, (z, r_z))) \to b$:
  1. Parse $\mathbb{x}$ as a group element $W \in \mathbb{G}$ and $\mathbb{w}$ as $w \in \mathbb{Z}_q$.
  2. Check that:
     - $g^z = S \cdot W^c \in \mathbb{G}$; and
     - C_s, C_w are valid AHE ciphertexts; and
     - $\mathsf{Enc}(\mathsf{ek}, z; r_z) = C_s + c \cdot C_w$.
  3. If all checks pass, output 1. Otherwise, output 0.

The above protocol satisfies straight-line extraction as follows: the extractor is given access to the secret decryption key dk and simply decrypts C_w. And this holds even for a statistical prover. Furthermore, the honest verifier zero-knowledge (HVZK) property follows from the honest verifier zero-knowledge property of Schnorr and the semantic security of AHE. The ZK simulator samples a random challenge and simulates the Schnorr proof (a, c, z) by running the HVZK simulator of the original Schnorr proof. To simulate the encryptions, the HVZK simulator computes $C_s = \mathsf{Enc}(\mathsf{ek}, 0; r')$ for a random r' and sets $C_w = (\mathsf{Enc}(\mathsf{ek}, z; r) - C_s) \cdot c^{-1}$ for a random r. The simulated encryptions are indistinguishable from the encryptions in a real proof due to the semantic security of AHE.

While the above protocol achieves straight-line knowledge soundness, it has two drawbacks: it (1) is interactive and (2) requires a trusted structured setup with secret values. Fortunately, both of these can be mitigated in the ROM.

2.3 NIZK in ROM using Fiat-Shamir

As mentioned in Section 2.2, we use the random oracle to achieve two goals, achieving non-interactivity and removing the common reference string.

Non-interactivity. Achieving non-interactivity for Schnorr's protocol is straightforward via the Fiat-Shamir transform [FS87].[7] As a result of applying Fiat-Shamir, the security proofs change as follows:

- Straight-line extraction: assuming Q is the number of random oracle queries made by the malicious prover, applying Fiat-Shamir incurs a Q-factor security loss, since the malicious prover may sample at most Q possible first message a values, which generates at most Q possible values of c.
- Zero knowledge: the zero-knowledge simulator must program the random oracle so that $c := \mathsf{RO}(\mathbb{x}, a)$. However, there is no change in distinguishing advantage between the real and simulated proofs.

[7] $\mathcal{P}$ and $\mathcal{V}$ have query access to a random oracle $\mathsf{RO}\colon \{0,1\}^* \to \mathbb{Z}_q$. $\mathcal{P}$ samples the random challenge by itself as $c := \mathsf{RO}(\mathbb{x}, a)$ and sends c to $\mathcal{V}$. $\mathcal{V}$ additionally checks that the received transcript $(a, c, \cdot)$ satisfies $c = \mathsf{RO}(\mathbb{x}, a)$.

Minimizing setup assumptions. The above protocol relies on a trusted party to run the AHE key generation algorithm and output the (public) encryption key ek and the (secret) decryption key dk. One way to side-step having a common reference string is via instantiating AHE with a scheme in which public keys are *obliviously sampleable using a hash function*, i.e. the following distributions are indistinguishable

$$\{\, \mathsf{ek} : (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\kappa) \,\} \text{ and } \{\, \mathsf{ek} \leftarrow \mathcal{H}_{\mathsf{ek}}(1^\kappa, \cdot) \,\} \quad .$$

Then, both $\mathcal{P}$ and $\mathcal{V}$ can derive an instance-specific encryption key: $\mathsf{ek} := \mathcal{H}_{\mathsf{ek}}(1^\kappa, \mathbb{x})$. Then, in the analysis of knowledge soundness, an additional hybrid is required, so that the knowledge extractor can recover the decryption key dk. Moreover, zero-knowledge is preserved, assuming that it is inefficient to recover a valid decryption key with respect to $\mathsf{ek} \leftarrow \mathcal{H}_{\mathsf{ek}}(1^\kappa, \cdot)$.

We note that the random oracles for the two tasks are distinct. First, they must be domain-separated, so that soundness is preserved (we need the outputs of both invocations to be independently sampled). Second, the output spaces of the hashes are different too. For non-interactivity, the random oracle simulates the verifier's random challenge, i.e. outputs a value in $\mathbb{Z}_q$. For non-removing the common reference string, the random oracle samples an AHE encryption key, which simulates the distribution of ek output by the AHE key generation algorithm. For the rest of the technical overview, we denote our straightline-extractable protocol in the ROM as Π_{GenLin}.

2.4 Extending our straight-line extractable NIZK to the UC setting

In general, stand-alone NIZK constructions are not universally composable because standard security definitions do not consider adversarial behavior in the presence of concurrent protocol executions. In particular, an adversary, after observing polynomially-many proof strings, should not be able to forge a proof for an instance $\mathbb{x}$, for which it doesn't know a corresponding witness $\mathbb{w}$, i.e. if the adversary produces a valid proof, the (UC) simulator should be able to extract a valid witness $\mathbb{w}$ for $\mathbb{x}$. This notion is called *non-malleability* or *simulation-extractability* (more common in the UC ZK literature) [FKMV12, Gro06, KZM+15].

Toward proving UC security, we show that Π_{GenLin} satisfies simulation-extractability. This is done by following the paradigm [FKMV12] of reducing simulation-extractability to *weak unique response* (WUR) knowledge soundness, and zero-knowledge. We first show that Π_{GenLin} satisfies WUR, i.e. the probability that an adversary can find two accepting proofs $(a, c, z), (a, c, z')$ for instance $\mathbb{x}$, such that $z \neq z'$, is negligible. Simulation-extractability follows via combining our proof of WUR and the knowledge extractor for Π_{GenLin}. While this is standard, we note that, interestingly, our proof of WUR *does not* rely on WUR of the underlying Sigma protocol. Thus, our compiled protocol is WUR even when the underlying protocol is not. Consider a Sigma protocol resulting from OR composition that does not satisfy WUR. This is because the third message can be computed from one of the many witnesses. In our straight-line compiled protocol, however, the first message consists of a ciphertext encrypting the witness which forces the prover to use the same witness in the response, thus recovering the WUR property. Finally, we conclude UC-security by arguing that the following is a (UC) simulator for Π_{GenLin}: (1) the zero-knowledge simulator of Π_{GenLin} simulates proofs output by Π_{GenLin}; and (2) the simulation extractor (constructed above) extracts witnesses from adversarialy generated proof strings.

Extensions While we focus on linear relations in the exponent, we note that our transform can also work for arbitrary algebraic relation with proper representation[8]. Furthermore, it also works for the OR composition of $\mathcal{R}_{\mathsf{DLog}}$. At a high level, this works by (1) running our UC-NIZK for the NP statement, for which the witness is known; then (2) using the HVZK simulator to generate proofs for all other statements. The straightforward construction is provided in Appendix E.

[8] Note that a polynomial $p(x)$ of degree d is linear in powers of x, $1, x, x^2, \ldots, x^d$. So our protocols work for proving relations $p(x) = y$ over a field as well.

3 Preliminaries

We denote by $a \leftarrow \mathcal{D}$ a uniform sampling of an element a from a distribution $\mathcal{D}$. The set of elements $\{1, \ldots, n\}$ is represented by $[n]$. We denote the computational security parameter by κ and statistical security parameter by λ_{st}, respectively.

Multiplicative cyclic groups. We consider *multiplicative cyclic groups* $\mathbb{G}$ of prime order q, i.e. $|\mathbb{G}| = q$. We write $\langle g \rangle := \{g^k : k \in \mathbb{N}\}$ to denote the cyclic group generated by g; the element g is called a generator. (Since $|\mathbb{G}| = q$, every element in $\mathbb{G}$ is a generator.) The corresponding field (for computations in the exponent) is denoted as $\mathbb{Z}_q$.

Vectors and matrices. We use boldface to denote matrices and vectors. Sometimes we use notations $\boldsymbol{A}_{m \times n}$ to a matrix of dimension $m \times n$. The element in the i-th row and j-th column is denoted by $\boldsymbol{A}_{ij}$, and the j-th column vector (of dimension m) of $\boldsymbol{A}$ is denoted $\boldsymbol{A}_j$. Analogously, for a vector $\boldsymbol{v}$, $\boldsymbol{v}_i$ denotes the i-th element.

Random oracle. A *random oracle* is an oracle distribution $\mathcal{U}(m, n)$ given by $\mathsf{RO} \leftarrow (\{0,1\}^m \to \{0,1\}^n)$ for some $m, n \in \mathbb{N}$. In security analyses, we also require the random oracle to be *programmable*, i.e. a simulator can set the oracle's output value at a small number of query points.

Universal Composability. We follow the Universal Composability Framework [Can01], in that a real-world multi-party protocol realizes an ideal functionality in the presence of an adversary. We refer to Appendix. B for a more detailed description. We also assume the existence of a *default authenticated channel* in the real world between any two parties.

3.1 Definition: Additively Homomorphic Encryption

An *additively-homomorphic encryption scheme* is a tuple of algorithms $\mathsf{AHE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ that works as follows.

– $\mathsf{Gen}(1^\kappa) \to (\mathsf{ek}, \mathsf{dk})$. On input a security parameter κ (in unary), outputs a (public) encryption key $\mathsf{ek} \in \mathcal{K}_{\mathsf{ek}}$ and a (secret) decryption key $\mathsf{dk} \in \mathcal{K}_{\mathsf{dk}}$ in the respective key spaces.
– $\mathsf{Enc}(\mathsf{ek}, m, r) \to c$. On input an encryption key $\mathsf{ek} \in \mathcal{K}_{\mathsf{ek}}$, a message m in message space $\mathcal{M}$ and encryption randomness r in randomness space $\mathcal{R}$, outputs a ciphertext c in ciphertext space $\mathcal{C}$.
– $\mathsf{Dec}(\mathsf{dk}, c) \to m$. On input a decryption key $\mathsf{dk} \in \mathcal{K}_{\mathsf{dk}}$ and a ciphertext c, deterministically outputs a message $m \in \mathcal{M}$.

We require AHE to satisfy the following completeness and security properties:

– **Perfect correctness.** For any message $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Dec}(\mathsf{dk}, \mathsf{Enc}(\mathsf{ek}, m, r)) = m \;\middle|\; \begin{array}{r} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\kappa) \\ r \leftarrow \mathcal{R} \end{array}\right] = 1 \quad .$$

– **Semantic security.** AHE is semantically-secure if any PPT (stateful) adversary $\mathcal{A}$ cannot distinguish between the following distributions:

$$\mathcal{D}_0(\kappa) := \left\{ (c_1, \ldots, c_n) \;\middle|\; \begin{array}{r} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\kappa) \\ \{(m_{0,i}, m_{1,i})\}_{i \in [n]} \leftarrow \mathcal{A}(\mathsf{ek}) \\ r_1, \ldots, r_n \leftarrow \mathcal{R} \\ c_i \leftarrow \mathsf{Enc}(\mathsf{ek}, m_{0,i}, r_i) \end{array} \right\}$$
$$\text{and } \mathcal{D}_1(\kappa) := \left\{ (c_1, \ldots, c_n) \;\middle|\; \begin{array}{r} (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\kappa) \\ \{(m_{0,i}, m_{1,i})\}_{i \in [n]} \leftarrow \mathcal{A}(\mathsf{ek}) \\ r_1, \ldots, r_n \leftarrow \mathcal{R} \\ c_i \leftarrow \mathsf{Enc}(\mathsf{ek}, m_{1,i}, r_i) \end{array} \right\} \quad .$$

We also require the scheme to satisfy additive homomorphism and a related well-formedness property.

- **Additive homomorphism.** Let $\kappa \in \mathbb{N}$ be a security parameter. Let $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\kappa)$, then there are polynomial time deterministic algorithms Add and ScMult such that:
  - For any $c_1, c_2 \in \mathcal{C}$ define homomorphic addition $c_{(+)} := \mathsf{Add}(c_1, c_2)$ such that if $c_1 := \mathsf{Enc}(\mathsf{ek}, m_1; r_1)$ and $c_2 := \mathsf{Enc}(\mathsf{ek}, m_2; r_2)$, then $c_{(+)} = \mathsf{Enc}(\mathsf{ek}, m_0 + m_1; r_0 + r_1)$. Similarly, we define homomorphic subtraction as $c_{(-)} := \mathsf{Add}(c_1, -c_2)$ such that $c_{(-)} = \mathsf{Enc}(\mathsf{ek}, m_0 - m_1; r_0 - r_1)$. Here we assume the addition/subtraction operations $+/-$ are defined in both $\mathcal{M}$ and $\mathcal{R}$.
  - For any $c \in \mathcal{C}$, and any scalar s which is in $\mathcal{M}$ and $\mathcal{R}$, define scalar multiplication $c_{(\cdot)} := \mathsf{ScMult}(s, c)$ such that if $c := \mathsf{Enc}(\mathsf{ek}, m; r)$, then $c_{(\cdot)} = \mathsf{Enc}(\mathsf{ek}, sm; sr)$.
- **Homomorphic well-formedness.** Let $c_1, c_2 \in \mathcal{C}$ be two arbitrary strings in the ciphertext space. Suppose, for any uniformly random scalar $s \in \mathcal{M}$, $c^* := \mathsf{Add}(c_1, \mathsf{ScMult}(s, c_2))$. Also, let $m^* \leftarrow \mathsf{Dec}(\mathsf{dk}, c^*)$, then we have that $m_1 \leftarrow \mathsf{Dec}(\mathsf{dk}, c_1)$ and $m_2 \leftarrow \mathsf{Dec}(\mathsf{dk}, c_2)$ such that $m^* = m_1 + s \cdot m_2$.

Finally we need a crucial oblivious sampleability property of the public-key.

- **Oblivious sampleability of public key.** There exists an efficiently computable hash function $\mathcal{H}_{\mathsf{ek}} : \{0,1\}^* \to \mathcal{K}_{\mathsf{ek}}$ such that the public-key can be sampled obliviously as $\mathsf{ek} := \mathcal{H}_{\mathsf{ek}}(x)$ on an uniform random input x and the following distributions are statistically close:

$$\{\, \mathsf{ek} : (\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\kappa) \,\} \text{ and } \{\, \mathsf{ek} \leftarrow \mathcal{H}_{\mathsf{ek}}(1^\kappa, \cdot) \,\} \quad .$$

  Looking ahead, $\mathcal{H}_{\mathsf{ek}}$ is to be modeled as a programmable random oracle in security proof. Obliviousness implies that, this can be done without explicit knowledge of the corresponding secret key, and therefore, given x anyone can check whether ek is generated correctly.

Matrix Encryption. We can extend the above notation to compactly capture encrypting a matrix $\boldsymbol{m} \in \mathcal{M}_{k\times n}$ using $\mathsf{MatEnc}(\mathsf{ek}, \boldsymbol{m})$ which returns a ciphertext matrix $\boldsymbol{c} \in \mathcal{C}_{k\times n}$, in that each element $c_{ij} = \mathsf{Enc}(\mathsf{ek}, m_{ij})$ for $i \in [k], j \in [n]$. Matrix decryption is similarly denoted by $\mathsf{MatDec}(\mathsf{dk}, \boldsymbol{c})$. The addition and scalar multiplication defined above naturally extends for matrices.

3.2 Definition: Sigma protocols

We define Sigma protocols for an NP relation $\mathcal{R}$ in the common reference string (CRS) model. It works as follows: A 3-move public coin Sigma protocol [CPV20] for a relation $\mathcal{R}$ is a tuple of algorithms $\Sigma = (\mathsf{Setup}, \mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2), \mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2))$. The prover $\mathcal{P}$ receives an instance $\mathbb{x}$ and witness $\mathbb{w}$ as input. The verifier $\mathcal{V}$ receives $\mathbb{x}$ as input. Σ proceeds in the following format:

- $\mathsf{Setup}(1^\kappa) \to (\mathsf{crs}, \mathsf{td})$: The Setup algorithm runs on (unary) security parameter κ and generates a CRS crs and a trapdoor td. All algorithms receive crs as inputs, and td is only used in extraction/simulation.
- $\mathcal{P}_1(\mathsf{crs}, \mathbb{x}, \mathbb{w}; \rho) \to a$: $\mathcal{P}$ runs (randomized) algorithm $\mathcal{P}_1$ on the (public) instance $\mathbb{x}$, (private) witness $\mathbb{w}$ to obtain the first message a – this is also called a **commitment**. $\mathcal{P}$ sends a to $\mathcal{V}$. Here ρ is the prover's randomness, which is stored to be used later in $\mathcal{P}_2$.
- $\mathcal{V}_1(\mathsf{crs}, a) \to c$: $\mathcal{V}$ samples random **challenge** $c \xleftarrow{\$} \mathcal{C}$ and sends c to $\mathcal{P}$.
- $\mathcal{P}_2(\mathsf{crs}, \mathbb{x}, \mathbb{w}, a, c, \rho) \to z$: $\mathcal{P}$ runs algorithm $\mathcal{P}_2$ with $\mathbb{x}, \mathbb{w}, a, c, \rho$ to output z. It sends **response** z to $\mathcal{V}$.
- $\mathcal{V}_2(\mathsf{crs}, \mathbb{x}, (a, c, z)) \to 1/0$: $\mathcal{V}$, on input the instance and the **transcript** (a, c, z), which together constitutes the proof π, outputs 1 if it accepts and 0 if it rejects.

Let us now define the security properties of a Sigma protocol.

- **Perfect completeness.** If $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, then $\mathcal{V}$ accepts all honest 3-move transcripts as long as $\mathcal{P}_1$ and $\mathcal{P}_2$ uses the same ρ.
- **Special soundness.** There exists an efficient extractor Ext that, on input a CRS crs, an instance $\mathbb{x} \in \mathcal{L}$, and two accepting transcripts (a, c, z) and (a, c', z') such that $c \neq c' \in \mathcal{C}$, outputs a witness $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ with probability $1 - \mathsf{negl}(\kappa)$. We call the loss the **special soundness error**.
- **Straight-line knowledge soundness.** There exists an efficient deterministic algorithm called **straight-line Knowledge Extractor** $\mathcal{E}$ that, on input the public information crs, the trapdoor td, instance $\mathbb{x} \in \mathcal{L}$,

and a single accepting transcript (a, c, z) outputs an accepting witness $\mathbb{w}$ for which $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ with probability $1 - \mathsf{negl}(\kappa)$.

- **Honest-verifier zero knowledge (HVZK).** There exists a PPT simulator algorithm Sim that, on input the setup string crs, trapdoor td for crs, instance $\mathbb{x} \in \mathcal{L}$, and a uniform random challenge $c \xleftarrow{\$} \mathcal{C}$, outputs (a, z) such that $\mathcal{V}_2(\mathsf{crs}, \mathbb{x}, (a, c, z)) = 1$. Further, for every PPT adversary $\mathcal{A}$, the following distributions are indistinguishable:

$$\left\{ \mathcal{A}(\mathbb{x}, (a, c, z)) = 1 \;\middle|\; \begin{array}{r} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\kappa) \\ \rho \xleftarrow{\$} \mathcal{R}; a \leftarrow \mathcal{P}_1(\mathsf{crs}, \mathbb{x}, \mathbb{w}, \rho) \\ c \xleftarrow{\$} \mathcal{C}; z \leftarrow \mathcal{P}_2(\mathsf{crs}, \mathbb{x}, \mathbb{w}, c, \rho) \end{array} \right\}$$

$$\text{and} \quad \left\{ \mathcal{A}(\mathbb{x}, (a, c, z)) = 1 \;\middle|\; \begin{array}{r} (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\kappa) \\ c \xleftarrow{\$} \mathcal{C}; (a, z) \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathsf{td}, \mathbb{x}, c) \end{array} \right\} .$$

3.3 Definition: Straight-line Extractable NIZKs

We define straight-line-extractable non-interactive zero-knowledge proofs in the random oracle model (ROM) for an NP relation $\mathcal{R}$. The proof system Π consists of a tuple of algorithms $(\mathsf{Setup}, \mathcal{P}^{\mathrm{RO}}, \mathcal{V}^{\mathrm{RO}})$ defined as follows:

- $\mathsf{Setup}(1^\kappa) \to \mathrm{RO}$. On input a security parameter κ, Setup samples a function RO uniformly from the set of all functions mapping $\{0, 1\}^* \to \mathcal{C}$.
- $\mathcal{P}^{\mathrm{RO}}(\mathbb{x}, \mathbb{w}) \to \pi$. On input an instance $\mathbb{x}$, and a corresponding witness $\mathbb{w}$, the prover $\mathcal{P}$ computes a proof π.
- $\mathcal{V}^{\mathrm{RO}}(\mathbb{x}, \pi) \to 1/0$. On input an instance $\mathbb{x}$, and a corresponding proof π, the verifier $\mathcal{V}$ computes a decision bit.

We require Π to satisfy the following completeness, (computational) zero-knowledge and (statistical) straight-line knowledge soundness properties in the ROM:

- **Perfect Completeness.** For any adversary (possibly unbounded) $\mathcal{A}$

$$\Pr \left[\begin{array}{c} (\mathbb{x}, \mathbb{w}) \notin \mathcal{R} \\ \wedge \\ (\mathcal{V}^{\mathrm{RO}}(\mathbb{x}, \pi) = 1) \end{array} \;\middle|\; \begin{array}{r} \mathrm{RO} \leftarrow \mathsf{Setup}(1^\kappa) \\ (\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}^{\mathrm{RO}} \\ \pi \leftarrow \mathcal{P}^{\mathrm{RO}}(\mathbb{x}, \mathbb{w}) \end{array} \right] = 1$$

  The above formulation of completeness allows $(\mathbb{x}, \mathbb{w})$ to depend on the oracle RO. Here $\mathcal{A}$ can make unbounded many queries to RO.

- **(Computational) Zero Knowledge.** Before defining zero-knowledge we define **NIZK simulator** (in the random oracle model) and associated **wrapper oracles** for an NP relation $\mathcal{R}$. A NIZK simulator $\mathcal{S}$ in the random oracle model is a stateful PPT algorithm that can operate in two modes. The first mode $(h_i, st) \leftarrow \mathcal{S}(1, st, q_i)$ handles RO queries whereas the second mode $(\mathbb{x}, \pi, st) \leftarrow \mathcal{S}(2, st, \mathbb{x})$ returns a simulated proof for $\mathbb{x}$. Let $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_2'$ be wrapper oracles that share state. $\mathcal{S}_1(q_i)$ is a wrapper around $\mathcal{S}(1, st, q_i)$ returning only h_i while internally updating st. Similarly, $\mathcal{S}_2(\mathbb{x}, \mathbb{w})$ and $\mathcal{S}_2'(\mathbb{x})$ be wrappers around $\mathcal{S}(2, st, \mathbb{x})$ returning only $(\mathbb{x}, \pi)$ and internally updating st, except that $\mathcal{S}_2(\mathbb{x}, \mathbb{w})$ aborts if $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}$. We say that Π has *computational zero knowledge* if there exists a simulator $\mathcal{S}$ such that for any PPT adversary $\mathcal{A}$, the following is negligible in κ.

$$\Pr \left[\mathcal{A}^{\mathrm{RO}, \mathcal{P}(\cdot,\cdot)}(1^\kappa) = 1 \;\middle|\; \mathrm{RO} \leftarrow \mathsf{Setup}(1^\kappa) \right] - \Pr \left[\mathcal{A}^{\mathcal{S}_1, \mathcal{S}_2}(1^\kappa) = 1 \;\middle|\; \mathrm{RO} \leftarrow \mathsf{Setup}(1^\kappa) \right]$$

  Above, $\mathcal{P}$ and $\mathcal{S}_2$ both return $\perp$ when queried on $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}$.

- **(Statistical) Straight-line Knowledge Soundness.** We first define a straight-line extractor $\mathcal{E}$ as a stateful PPT algorithm which works in two modes: $(h_i, st) \leftarrow \mathcal{E}(1, q_i, st)$ handles the RO queries using lazy sampling, whereas $(\mathbb{w}, st) \leftarrow \mathcal{E}(2, \mathbb{x}, \pi, st)$ returns a witness. Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be the wrappers around $\mathcal{E}$ such that each outputs the first part of the respective outputs (without the state, which is kept secret). Importantly, $\mathcal{E}$ is straight-line, that is it does not rewind or use forking [JT20]. Π has *straight-line knowledge soundness* if there exists a PPT stateful extractor $\mathcal{E}$ such that for any unbounded adversary $\mathcal{A}$, which makes bounded-many queries to RO we have:

$$\Pr\left[\begin{array}{c} \mathcal{V}^{\mathrm{RO}}(\mathbb{x}, \pi) = 1 \wedge \\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R} \end{array} \middle| \begin{array}{r} \mathrm{RO} \leftarrow \mathsf{Setup}(1^\kappa); (\mathbb{x}, \pi) \leftarrow \mathcal{A}^{\mathcal{E}_1} \\ \mathbb{w} \leftarrow \mathcal{E}_2(\mathbb{x}, \pi) \end{array}\right] \leq \mathsf{negl}(\kappa)$$

We need a few more definitions for showing a stronger *simulation extractability* property for Π.

– **Weak Unique Response [GOP⁺23].** Π is said to satisfy *weak unique response* with respect to the zero-knowledge simulator $\mathcal{S}$ with wrapper oracles $(\mathcal{S}_1, \mathcal{S}_2')$ (as defined above), if given a simulated transcript $(\mathbb{x}, c, z) \leftarrow \mathcal{S}_2'(\mathbb{x})$, for all PPT adversaries $\mathcal{A}$ the following probability is at most $\mathsf{negl}(\kappa)$.

$$\Pr\left[\begin{array}{c} \mathcal{V}^{\mathcal{S}_1}(\mathbb{x}, a, c, z') = 1 \\ \wedge \\ z' \neq z \end{array} \middle| \begin{array}{c} (\mathbb{x}, a, c, z) \leftarrow \mathcal{S}_2'(\mathbb{x}) \\ (\mathbb{x}, a, c, z') \leftarrow \mathcal{A}^{\mathcal{S}_1}(\mathbb{x}, a, c, z) \end{array}\right]$$

– **Simulation Extractability, [FKMV12]** Π is said to satisfy *simulation extractability* with respect to a stateful PPT simulator $\mathcal{S}$ with wrapper oracles $(\mathcal{S}_1, \mathcal{S}_2')$ (as defined above) if there exists a (straight-line) PPT extractor $\hat{\mathcal{E}}$ such that for all PPT adversaries $\mathcal{A}$ the following holds:

$$\Pr\left[\begin{array}{c} \mathcal{V}^{\mathcal{S}_1}(\mathbb{x}^*, \pi^*) = 1 \\ \wedge\ (\mathbb{x}^*, \mathbb{w}^*) \notin \mathcal{R}\ \wedge\ (\mathbb{x}^*, \pi^*) \notin \mathcal{T} \end{array} \middle| \begin{array}{r} (\mathbb{x}^*, \pi^*) \leftarrow \mathcal{A}^{(\mathcal{S}_1, \mathcal{S}_2')} \\ \mathbb{w}^* \leftarrow \hat{\mathcal{E}}(\mathbb{x}^*, \pi^*) \end{array}\right] \leq \mathsf{negl}(\kappa)\ .$$

Here $\mathcal{T}$ is the list of transcripts received by $\mathcal{A}$ on querying $\mathcal{S}_2'$.

4 Straight-line Extractable Proof Systems for Arbitrary Linear Relations

We present our interactive straight-line extractable Sigma protocol. First, we establish our notations for arbitrary linear relations of group elements in Section 4.1 and recall the generic Sigma protocol (Figure 1) for arbitrary relations in Section 4.2.[9] Second, we present the interactive straight-line extractable Sigma protocol (Figure 2) in Section 4.3.

4.1 Notations

We assume a cyclic group $\mathbb{G}$ of prime order q with g as a generator, and a corresponding finite field $\mathbb{Z}_q$. Now we define:

– For each matrix $\boldsymbol{a} \in \mathbb{Z}_q^{m \times n}$, we denote the **matrix exponentiation** $\boldsymbol{A} := g^{\boldsymbol{a}} \in \mathbb{G}^{m \times n}$ where each element $A_{ij} = g^{a_{ij}}$. Below we assume $\boldsymbol{A} = g^{\boldsymbol{a}}$.
– The **scalar power** of $\boldsymbol{A} \in \mathbb{G}^{m \times n}$ with respect to a scalar $s \in \mathbb{Z}_q$ is denoted by $\boldsymbol{A}^s \in \mathbb{G}^{m \times n}$, each entry of which is given by $\boldsymbol{A}_{ij}^s := (\boldsymbol{A}_{ij})^s$. Notice that, $\boldsymbol{A}^s = g^{s\boldsymbol{a}}$, where $s\boldsymbol{a}$ is a standard scalar multiplication.
– Given a vector $\boldsymbol{v} \in \mathbb{Z}_q^m$, the **vector power** of $\boldsymbol{A}$ is denoted by $\boldsymbol{A}^{\boldsymbol{v}} = g^{\boldsymbol{v} \cdot \boldsymbol{a}}$, where $\boldsymbol{v} \cdot \boldsymbol{a}$ is a vector-matrix multiplication resulting into a vector of dimension n. Alternatively, the j-th entry of $\boldsymbol{A}^{\boldsymbol{v}}$ is given by a multi-exponentiation $\prod_{i=1}^m A_{ij}^{b_i}$. For example, let $\boldsymbol{A} = \begin{pmatrix} A_{11} & A_{12} & A_{1,3} \\ A_{21} & A_{22} & A_{23} \end{pmatrix}$ and $\boldsymbol{v} = (v_1\ v_2)$ then $\boldsymbol{A}^{\boldsymbol{v}} = \left(A_{11}^{v_1} \cdot A_{21}^{v_2} \ \middle|\ A_{12}^{v_1} \cdot A_{22}^{v_2} \ \middle|\ A_{13}^{v_1} \cdot A_{23}^{v_2}\right)$. Also note that, if $\boldsymbol{V} = \boldsymbol{A}^{\boldsymbol{v}}$, then for a scalar $s \in \mathbb{Z}_q$ $\boldsymbol{V}^s = \boldsymbol{A}^{s\boldsymbol{v}}$.
– Given a matrix $\boldsymbol{a} \in \mathbb{Z}_q^{m \times n}$, the **element-wise inverse** of $\boldsymbol{a}$ denoted as $(\boldsymbol{a}^{-1}$, each element of which is the inverse (in $\mathbb{Z}_q$) of each element of the vector $\boldsymbol{a}$ in the same position. For a matrix $\boldsymbol{A} \in \mathbb{G}^{m \times n}$, the element-wise inverse $\boldsymbol{A}^{-1}$ is defined as the matrix, in that each element is equal to A_{ij}^{-1} a multiplicative inverse in $\mathbb{G}$ of an element A_{ij} in $\boldsymbol{A}$ in the same position.

[9] The Schnorr, Okamoto and Chaum-Pedersen protocols are all special cases of a generic Sigma protocol for arbitrary linear relations of some group elements (see Chapter 19.5.3 of [BS23]).

– The **Hadamard product** of two arbitrary matrices of same dimensions $\boldsymbol{A}_{m\times n}$ and $\boldsymbol{B}_{m\times n}$, denoted by $\boldsymbol{A} \bullet \boldsymbol{B}$, defines a matrix $\boldsymbol{C}_{m\times n}$ whose entries are element-wise product of the entries of $\boldsymbol{A}$ and $\boldsymbol{B}$. That is, $C_{i,j} = A_{i,j} \cdot B_{i,j}$ for $1 \le i \le m$ and $1 \le j \le n$. When $\boldsymbol{A} = g^{\boldsymbol{a}}$ and $\boldsymbol{B} = g^{\boldsymbol{b}}$, $\boldsymbol{A} \bullet \boldsymbol{B} = g^{\boldsymbol{a}+\boldsymbol{b}}$, where '$+$' denotes the standard matrix addition over $\mathbb{Z}_q$. Furthermore, if $\boldsymbol{V} = \boldsymbol{A}^{\boldsymbol{v}}$ and $\boldsymbol{W} = \boldsymbol{A}^{\boldsymbol{w}}$, then $\boldsymbol{V} \bullet \boldsymbol{W} = \boldsymbol{A}^{\boldsymbol{v}+\boldsymbol{w}}$.

4.2 Standard Sigma Protocol for Arbitrary Linear Relation

Now, for a vector $\boldsymbol{w} \in \mathbb{Z}_q^n$, and a matrix $\boldsymbol{y} \in \mathbb{Z}_q^{n\times m}$ consider the following linear relation: $\boldsymbol{U} = \boldsymbol{Y}^{\boldsymbol{w}} \in \mathbb{G}^m$, where $\boldsymbol{Y} = g^{\boldsymbol{y}} \in \mathbb{G}^{n\times m}$. (Alternatively, we can write $\boldsymbol{U} = g^{\boldsymbol{w}\cdot\boldsymbol{y}}$.) To summarize, define the following relation:

$$\mathcal{R}_{\mathsf{GenLin}} := \left\{ \left((\boldsymbol{Y} \in \mathbb{G}^{n\times m}, \boldsymbol{U} \in \mathbb{G}^m), \boldsymbol{w} \right) : \boldsymbol{w} \in \mathbb{Z}_q^n \text{ and } \boldsymbol{U} = \boldsymbol{Y}^{\boldsymbol{w}} \right\} \quad . \tag{1}$$

Note that $\mathcal{R}_{\mathsf{GenLin}}$ checks the following in the exponent, with respect to a fixed generator $g \in \mathbb{G}$: for $\boldsymbol{y} \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{u} \in \mathbb{Z}_q^m$, there exists $\boldsymbol{w} \in \mathbb{Z}_q^n$ such that $\boldsymbol{u} = \boldsymbol{w}\boldsymbol{y}$.

We describe a (standard) Sigma protocol $\Sigma = (\mathsf{Setup}, \mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2), \mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2))$ for $\mathcal{R}_{\mathsf{GenLin}}$, where the prover $\mathcal{P}$ has the inputs the public instance $\mathbb{x} := (\boldsymbol{Y} \in \mathbb{G}^{n\times m}, \boldsymbol{U} \in \mathbb{G}^m)$ and the witness $\mathbb{w} := \boldsymbol{w} \in \mathbb{Z}_q^n$ and the verifier $\mathcal{V}$ has input $(\boldsymbol{Y}, \boldsymbol{U})$. Here integers m denotes the number of relations (or constraints) and n denotes the number of witnesses.

Inputs: Both prover and verifier know the public instance $\mathbb{x} := (\boldsymbol{Y} \in \mathbb{G}^{n\times m}, \boldsymbol{U} \in \mathbb{G}^m)$, and the prover exclusively has witness $\mathbb{w} := \boldsymbol{w} \in \mathbb{Z}_q^n$.
Round-1 (Commit): The prover $\mathcal{P}$ runs algorithm $\mathcal{P}_1(\mathsf{crs}, \mathbb{x}, \mathbb{w})$, which works as follows:

– Sample $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$ and compute $\boldsymbol{S} := \boldsymbol{Y}^{\boldsymbol{s}} \in \mathbb{G}^m$.
– Set $a := \boldsymbol{S}$ and $\rho := \boldsymbol{s}$.

$\mathcal{P}$ sends a to the verifier.
Round-2 (Challenge): The verifier $\mathcal{V}$ runs algorithm $\mathcal{V}_1(\mathsf{crs}, a)$, which, on receiving $a = \boldsymbol{S}$ samples challenge $c \xleftarrow{\$} \mathbb{Z}_q$ and sends c to $\mathcal{P}$.
Round-3 (Response): The prover $\mathcal{P}$, on receiving the challenge c, runs algorithm $\mathcal{P}_2(\mathsf{crs}, \mathbb{x}, \mathbb{w}, a, c, \rho)$, which works as:
– Parse ρ as $\boldsymbol{s}$.
– Compute $\boldsymbol{z} := \boldsymbol{s} + c\boldsymbol{w} \in \mathbb{Z}_q^n$.
– Set $z := \boldsymbol{z}$.
Send z to the verifier.
Check: The verifier $\mathcal{V}$, on receiving $z = \boldsymbol{z}$ outputs whatever is returned by the algorithm $\mathcal{V}_2(\mathsf{crs}, \mathbb{x}, a, c, z)$ which return 1 if and only if $\boldsymbol{Y}^{\boldsymbol{z}} = \boldsymbol{S} \bullet \boldsymbol{U}^c$.

Fig. 1: Standard Sigma Protocol for $\mathcal{R}_{\mathsf{GenLin}}$.

Theorem 1. *Suppose $\mathbb{G}$ is a group of prime order q. Then, the protocol in Figure 1 satisfies perfect completeness, special soundness with error $\frac{1}{q}$, and perfect honest verifier zero-knowledge.*

Proof. We prove Theorem 1 in Appendix A.1 for the sake of completeness. □

4.3 Straight-line Extractable Protocol for $\mathcal{R}_{\mathsf{GenLin}}$

We present our interactive three-move Sigma protocol for any linear relation $\mathcal{R}_{\mathsf{GenLin}}$ that is straight-line extractable in the crs model. An additional ingredient we use here is an additively homomorphic public-key encryption scheme $\mathsf{AHE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, which has message space $\mathbb{Z}_q$ and the property that the encryption key is obliviously sampleable. The crs consists of the encryption key ek. Further, recall the extended matrix encryption/decryption notations MatEnc and MatDec (see Section 3.1).

In Figure 2, we present a Sigma protocol $\Sigma = (\mathsf{Setup}, \mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2), \mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2))$ for $\mathcal{R}_{\mathsf{GenLin}}$ (Equation (1)) – recall that this relation consists of instances $\boldsymbol{Y} \in \mathbb{G}^{n\times m}, \boldsymbol{U} \in \mathbb{G}^m$ and witnesses $\boldsymbol{w} \in \mathbb{Z}_q^n$ satisfying $\boldsymbol{U} = \boldsymbol{Y}^{\boldsymbol{w}}$. Further, we notate how Figure 2 differs from Figure 1 in blue.

Inputs: Both prover and verifier know the public instance $\mathbb{x} := (\boldsymbol{Y} \in \mathbb{G}^{n\times m}, \boldsymbol{U} \in \mathbb{G}^m)$, and the prover exclusively has witness $\mathbb{w} := \boldsymbol{w} \in \mathbb{Z}_q^n$. The $\mathsf{Setup}(1^\lambda)$ samples key pairs $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda)$; set public $\mathsf{crs} := \mathsf{ek}$ and trapdoor $\mathsf{td} := \mathsf{dk}$.
Round-1 (Commit): The prover $\mathcal{P}$ runs algorithm $\mathcal{P}_1(\mathsf{crs}, \mathbb{x}, \mathbb{w})$, which works as follows:
– Sample $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$ and compute $\boldsymbol{S} := \boldsymbol{Y}^{\boldsymbol{s}} \in \mathbb{G}^m$.
– Sample encryption randomness $\boldsymbol{r_s}, \boldsymbol{r_w} \xleftarrow{\$} \mathcal{R}^n$.
– Compute encryptions $\boldsymbol{C}_s := \mathsf{MatEnc}(\mathsf{ek}, \boldsymbol{s}; \boldsymbol{r_s})$ and $\boldsymbol{C_w} := \mathsf{MatEnc}(\mathsf{ek}, \boldsymbol{w}; \boldsymbol{r_w})$ where $\mathsf{ek} = \mathsf{crs}$.
– Set $a := (\boldsymbol{S}, \boldsymbol{C}_s, \boldsymbol{C_w})$ and $\rho := (\boldsymbol{s}, \boldsymbol{r_w}, \boldsymbol{r_s})$.
Send a to the verifier.
Round-2 (Challenge): The verifier $\mathcal{V}$ runs algorithm $\mathcal{V}_1(\mathsf{crs}, a)$, which, on receiving $a = (\boldsymbol{S}, \boldsymbol{C}_s, \boldsymbol{C_w})$ samples challenge $c \xleftarrow{\$} \mathbb{Z}_q$ and send that to the prover.
Round-3 (Response): The prover $\mathcal{P}$, on receiving the challenge c, runs algorithm $\mathcal{P}_2(\mathsf{crs}, \mathbb{x}, \mathbb{w}, a, c, \rho)$, which works as:
– Parse ρ as $(\boldsymbol{s}, \boldsymbol{r_w}, \boldsymbol{r_s})$.
– Compute $\boldsymbol{z} := \boldsymbol{s} + c\boldsymbol{w} \in \mathbb{Z}_q^n$.
– Compute $\boldsymbol{r_z} := \boldsymbol{r_s} + c\boldsymbol{r_w}$.
– Define $z := (\boldsymbol{z}, \boldsymbol{r_z})$.
Send z to the verifier.
Check: The verifier $\mathcal{V}$, on receiving $z = (\boldsymbol{z}, \boldsymbol{r_z})$ outputs whatever is returned by the algorithm $\mathcal{V}_2(\mathsf{crs}, \mathbb{x}, a, c, z)$ which returns 1 if and only if:
– $\boldsymbol{Y}^{\boldsymbol{z}} = \boldsymbol{S} \bullet \boldsymbol{U}^c$.
– $\boldsymbol{C_s}, \boldsymbol{C_w} \in \mathcal{C}$.
– $\mathsf{MatEnc}(\mathsf{ek}, \boldsymbol{z}; \boldsymbol{r}_z) = \mathsf{Add}(\boldsymbol{C}_s, \mathsf{ScMult}(c, \boldsymbol{C_w}))$ where $\mathsf{ek} = \mathsf{crs}$.

Fig. 2: Our straight-line-extractable Sigma protocol for $\mathcal{R}_{\mathsf{GenLin}}$. We highlight the changes from the standard Sigma protocol of Figure 1 in blue.

Theorem 2. *Suppose $\mathbb{G}$ is a group of prime order q. Suppose* $\mathsf{AHE} = (\mathit{Gen}, \mathit{Enc}, \mathit{Dec})$ *is an additively homomorphic encryption scheme satisfying the perfect correctness, semantic security with distinguishing advantage at most δ_{sem}, and homomorphic well-formedness properties (defined in Section 3.1).*

Then, the Sigma protocol described in Figure 2 satisfies the following properties in the crs *model:*

– **Perfect Completeness**, *due to the additive homomorphism of* AHE*;*
– **(Statistical) Straight-line Knowledge Soundness** *with probability $1 - 1/q$, due to the correctness and homomorphic well-formedness of* AHE*; and*
– **(Computational) Honest Verifier Zero-Knowledge** *with simulation error δ_{sem}, due to the additive homomorphism and the semantic security of* AHE*.*

Proof. **Perfect Completeness.** From the verifier's computation, we have that $\boldsymbol{Y}^{\boldsymbol{z}} = \boldsymbol{Y}^{\boldsymbol{s}+c\boldsymbol{w}} = \boldsymbol{Y}^{\boldsymbol{s}} \bullet \boldsymbol{Y}^{c\boldsymbol{w}} = \boldsymbol{S} \bullet \boldsymbol{U}^c$ and

$$\begin{aligned}&\mathsf{MatEnc}(\mathsf{ek}, \boldsymbol{z} = \boldsymbol{s} + c\boldsymbol{w}, r_{\boldsymbol{z}} = r_{\boldsymbol{s}} + cr_{\boldsymbol{w}})\\ &= \mathsf{Add}(\mathsf{MatEnc}(\mathsf{ek}, \boldsymbol{s}; r_{\boldsymbol{s}}), \mathsf{MatEnc}(\mathsf{ek}, c\boldsymbol{w}; cr_{\boldsymbol{w}})\\ &= \mathsf{Add}(\boldsymbol{C}_s, \mathsf{ScMult}(c, \boldsymbol{C}_{\boldsymbol{w}}))\ .\end{aligned}$$

The second equation holds from the *correctness of the homomorphism* of the encryption scheme. Therefore, the verifier outputs 1 and our protocol is complete.

(Statistical) Straight-line Knowledge Soundness. We construct a straight-line knowledge extractor $\mathcal{E}_\Sigma$ which works as follows:

$\underline{\mathcal{E}_\Sigma(\mathsf{crs}, \mathsf{td}, \mathbb{x}, \pi) :}$

- On input $(\mathsf{crs}, \mathsf{td}, \mathbb{x}, \pi)$ parse $\mathsf{ek} = \mathsf{crs}$, the trapdoor $\mathsf{dk} = \mathsf{td}$, instance $(\boldsymbol{U}, \boldsymbol{Y}) = \mathbb{x}$, and a single accepting transcript $(a, c, z) = \pi$ where:
  - $a = (\boldsymbol{S}, \boldsymbol{C}_{\boldsymbol{s}})$; $c \in \mathbb{Z}_q$; $z = (\boldsymbol{z}, \boldsymbol{r}_{\boldsymbol{z}})$
  - $\boldsymbol{Y}^{\boldsymbol{z}} = \boldsymbol{S} \bullet \boldsymbol{U}^c$
  - $\boldsymbol{C}_{\boldsymbol{s}}, \boldsymbol{C}_{\boldsymbol{w}} \in \mathcal{C}$.
  - $\mathsf{MatEnc}(\mathsf{ek}, \boldsymbol{z}; \boldsymbol{r}_z) = \mathsf{Add}(\boldsymbol{C}_{\boldsymbol{s}}, \mathsf{ScMult}(c, \boldsymbol{C}_{\boldsymbol{w}}))$.
- Use dk to decrypt $\boldsymbol{w} \leftarrow \mathsf{MatDec}(\mathsf{dk}, \boldsymbol{C}_{\boldsymbol{w}})$ and $\boldsymbol{s} \leftarrow \mathsf{MatDec}(\mathsf{dk}, \boldsymbol{C}_{\boldsymbol{s}})$.
- Output $\boldsymbol{w}$ if $\boldsymbol{z} = \boldsymbol{s} + c\boldsymbol{w}$.

Now, we argue why the extractor works. First note that, since (a, c, z) is an accepting transcript, both the verification equations satisfy:

- $\boldsymbol{Y}^{\boldsymbol{z}} = \boldsymbol{S} \bullet \boldsymbol{U}^c$.
- $\mathsf{MatEnc}(\mathsf{ek}, \boldsymbol{z}; \boldsymbol{r}_z) = \mathsf{Add}(\boldsymbol{C}_s, \mathsf{ScMult}(c, \boldsymbol{C}_{\boldsymbol{w}}))$.

Then, combining the *homomorphic well-formedness* property with *correctness of the encryption*, we get that since $\mathsf{Add}(\boldsymbol{C}_s, \mathsf{ScMult}(c, \boldsymbol{C}_{\boldsymbol{w}}))$ equals $\mathsf{MatEnc}(\mathsf{ek}, \boldsymbol{z}; \boldsymbol{r}_z)$ which correctly decrypts to $\boldsymbol{z}$, and c is uniformly at random, each ciphertext $\boldsymbol{C}_{\boldsymbol{w}}$ and $\boldsymbol{C}_{\boldsymbol{s}}$ would always decrypt successfully. So the extractor $\mathcal{E}$ never fails while decrypting these ciphertexts. In fact, we have $\boldsymbol{z} = \boldsymbol{s} + c\boldsymbol{w}$, where $\boldsymbol{s} \leftarrow \mathsf{MatDec}(\mathsf{dk}, \boldsymbol{C}_{\boldsymbol{s}})$ and $\boldsymbol{w} \leftarrow \mathsf{MatDec}(\mathsf{dk}, \boldsymbol{C}_{\boldsymbol{w}})$. Since $\mathbb{G}$ is a cyclic group, we can write $\boldsymbol{S} = \boldsymbol{Y}^{\boldsymbol{s}'}$ and $\boldsymbol{U} = \boldsymbol{Y}^{\boldsymbol{w}'}$. The first verification equation is $\boldsymbol{z} = \boldsymbol{s}' + c\boldsymbol{w}'$. If $\boldsymbol{s} \neq \boldsymbol{s}'$ and $\boldsymbol{w} \neq \boldsymbol{w}'$, then c is uniquely defined as $c = (\boldsymbol{s}' - \boldsymbol{s})(\boldsymbol{w} - \boldsymbol{w}')^{-1}$, where the second term is an element-wise inverse of the vector $(\boldsymbol{w} - \boldsymbol{w}')$ – this fixes c in the commitment phase, which happens with probability $1/q$ as c is randomly chosen by verifier later in the challenge phase once $\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{w}, \boldsymbol{w}'$ are fixed in the commitment phase. So, with probability $1 - 1/q$, $\boldsymbol{s} = \boldsymbol{s}'$ and $\boldsymbol{w} = \boldsymbol{w}'$. This completes the proof.

(Computational) Honest Verifier Zero Knowledge.

We describe zero-knowledge simulator $\mathcal{S}_\Sigma$ as follows:

$\underline{\mathcal{S}_\Sigma(\mathsf{crs}, \mathbb{x}, c) :}$

- On input $(\mathsf{crs}, \mathbb{x}, c)$, where c is uniformly distributed over $\mathbb{Z}_q$, parse $\mathsf{ek} = \mathsf{crs}$ and $(\boldsymbol{Y} \in \mathbb{G}^{n\times m}, \boldsymbol{U} \in \mathbb{G}^m) = \mathbb{x}$.
- Sample $\boldsymbol{z} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$.
- Compute the element wise inverse $\boldsymbol{U}^{\boldsymbol{-1}}$.
- Compute $\boldsymbol{S} = \boldsymbol{Y}^{\boldsymbol{z}} \bullet (\boldsymbol{U}^{\boldsymbol{-1}})^c$.
- Sample $\boldsymbol{r}_{\boldsymbol{s}}, \boldsymbol{r}_{\boldsymbol{w}} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$.
- Compute $\boldsymbol{r}_{\boldsymbol{z}} = \boldsymbol{r}_{\boldsymbol{s}} + c \cdot \boldsymbol{r}_{\boldsymbol{w}}$.
- Compute encryptions $\boldsymbol{C}_z := \mathsf{MatEnc}(\mathsf{ek}, \boldsymbol{z}; r_{\boldsymbol{z}})$ and $\boldsymbol{C}_{\boldsymbol{w}} := \mathsf{MatEnc}(\mathsf{ek}, 0^n; \boldsymbol{r}_{\boldsymbol{w}})$.
- Compute homomorphically $\boldsymbol{C}_s := \mathsf{Add}(\boldsymbol{C}_z, -\mathsf{ScMult}(c, \boldsymbol{C}_{\boldsymbol{w}}))$.
- Set $a := (\boldsymbol{S}, \boldsymbol{C}_s, \boldsymbol{C}_{\boldsymbol{w}})$ and $z := (\boldsymbol{z}, \boldsymbol{r}_{\boldsymbol{z}}))$.
- Output (a, c, z).

We argue that the simulated transcript is computationally indistinguishable from the real transcript of Figure 2, as long as the semantic security and the homomorphic property of the underlying encryption scheme hold.

Observe that all values are distributed identically, except for the ciphertext $\boldsymbol{C}_{\boldsymbol{w}}$. In the real execution $\boldsymbol{C}_{\boldsymbol{w}} := \mathsf{MatEnc}(\mathsf{ek}, \boldsymbol{w}; \boldsymbol{r}_{\boldsymbol{w}})$, whereas Sim sets $\boldsymbol{C}_{\boldsymbol{w}} := \mathsf{MatEnc}(\mathsf{ek}, 0^n; \boldsymbol{r}_{\boldsymbol{w}})$. Hence, the transcripts are indistinguishable, except when the adversary breaks the semantic security of AHE. We give the full reduction to the semantic security of AHE in Appendix A.2.

□

5 Universally Composable NIZK Protocol

We present our UC-NIZK protocol for relation $\mathcal{R}_{\mathsf{GenLin}}$. We perform this by applying the Fiat-Shamir transform over our straight-line extractable Sigma protocol and proving that it is UC-secure. Before presenting our protocol, we present the general UC-NIZK functionality [GOS12, CSW22] in Fig. 3.[10]

$\mathcal{F}_{\mathsf{NIZK}}$ is parameterized by an NP Relation $\mathcal{R}$ and runs with a prover P, a verifier $\mathcal{V}$ and an ideal simulator Sim which stores proof transcripts in a list Q.

- **Proof** On input $(\mathit{prove}, \mathsf{sid}, \mathbb{x}, \mathbb{w})$ from P, if $\mathcal{R}(\mathbb{x}, \mathbb{w}) = 1$, send $(\mathit{prove}, \mathsf{sid}, P, \mathbb{x})$ to Sim. On receiving $(\mathit{proof}, \mathsf{sid}, \pi)$ from Sim, store $(\mathsf{sid}, \mathbb{x}, \pi)$ in Q and respond to P with $(\mathit{proof}, \mathsf{sid}, \pi)$.
- **Verify** On input $(\mathsf{sid}, \mathbb{x}, \pi)$ from $\mathcal{V}$, if $(\mathbb{x}, \pi) \notin Q$ then send $(\mathit{verify}, \mathsf{sid}, \mathbb{x}, \pi)$ to Sim. Upon receiving a witness $(\mathit{witness}, \mathbb{w})$ from Sim, if $\mathcal{R}(\mathbb{x}, \mathbb{w}) = 1$, store $(\mathsf{sid}, \mathbb{x}, \pi)$ in Q. Return $(\mathit{verification}, \mathsf{sid}, \mathbb{x}, \pi, \mathcal{R}(\mathbb{x}, \mathbb{w}))$ to $\mathcal{V}$.

Fig. 3: Ideal functionality $\mathcal{F}_{\mathsf{NIZK}}$

Next, we present our NIZK protocol Π_{GenLin} and show that it UC-securely implements $\mathcal{F}_{\mathsf{NIZK}}$ for relation $\mathcal{R}_{\mathsf{GenLin}}$.

In Fig. 4, we present our UC-NIZK protocol Π_{GenLin} for $\mathcal{R}_{\mathsf{GenLin}}$ (Eq.1). It is obtained by applying the standard Fiat-Shamir transformation [FS87] to our interactive straight-line-extractable Sigma protocol (from Fig.2) using the hash function $\mathcal{H}_{\mathcal{V}}$. We note that our Sigma protocol was in the URS model where the encryption key ek was part of the URS. But in our NIZK we use the public sampleability property of the underlying AHE scheme and generate the ek using a separate hash function $\mathcal{H}_{\mathsf{ek}}$. As a result, we do not require additional URS and prove security of our protocol in the random oracle model by modeling $\mathcal{H}_{\mathsf{ek}}$ and $\mathcal{H}_{\mathcal{V}}$ as random oracles. Before presenting our UC proof we show that our NIZK satisfies the standard property based definitions. We summarize it in Thm.3.

We formalize the security analysis of Π_{GenLin} via the following thoerem.

Theorem 3. *Suppose that:*

- *the underlying Sigma protocol satisfies perfect completeness, computational honest verifier zero-knowledge and statistical straight-line knowledge soundness;*
- *the underlying AHE scheme is obliviously sampleable; and*
- RO $= (\mathcal{H}_{\mathsf{ek}}, \mathcal{H}_{\mathcal{V}})$ *are programmable random oracles.*

[10] We do not require sub-session IDs, denoted ssid's, in $\mathcal{F}_{\mathsf{NIZK}}$; in the UC framework, ssid's are used for modelling multi-instance functionalities that have a local/internal shared resource such as a common reference string. The simulation-extractability and non-malleability properties are required for UC, independently of whether the NIZK functionality has access to a shared local resource.

Π_{GenLin}

Ingredients and Settings:

– **Input:** Both prover and verifier know the public instance $\mathbb{x} := (\boldsymbol{Y} \in \mathbb{G}^{n\times m}, \boldsymbol{U} \in \mathbb{G}^m)$, and the prover exclusively has witness $\mathbb{w} := \boldsymbol{w} \in \mathbb{Z}_q^n$.

– **Primitives:** The interactive Sigma protocol from Fig. 2 $(\mathsf{Setup}, \mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2), \mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2))$ based on an AHE scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with oblivious sampleability enabled by a hash function $\mathcal{H}_{\mathsf{ek}} : \{0,1\}^* \to \mathcal{K}_{\mathsf{ek}}$. Another hash function, $\mathcal{H}_{\mathcal{V}} : \{0,1\}^* \to \mathbb{Z}_q$. Both together are modeled as random oracle $\mathrm{RO} = (\mathcal{H}_{\mathsf{ek}}, \mathcal{H}_{\mathcal{V}})$

Protocol Description:

– $\mathcal{P}^{\mathrm{RO}}(\mathit{prove}, \mathsf{sid}, \mathbb{x}, \mathbb{w}) \to \pi$.
  - Parse RO as $(\mathcal{H}_{\mathsf{ek}}, \mathcal{H}_{\mathcal{V}})$.
  - Compute $\mathsf{ek} := \mathcal{H}_{\mathsf{ek}}(\mathsf{sid}, \mathbb{x})$ and set $\mathsf{crs} := \mathsf{ek}$.
  - Run $(\boldsymbol{S}, \boldsymbol{C}_s, \boldsymbol{C}_{\boldsymbol{w}}) := \mathcal{P}_1(\mathsf{crs}, \mathbb{x}, \mathbb{w}; \rho)$ where $\rho := (\boldsymbol{s}, \boldsymbol{r}_{\boldsymbol{w}}, \boldsymbol{r}_{\boldsymbol{s}})$.
  - Define $a := (\boldsymbol{S}, \boldsymbol{C}_s, \boldsymbol{C}_{\boldsymbol{w}})$.
  - Compute $c := \mathcal{H}_{\mathcal{V}}(\mathsf{sid}, \mathbb{x}, a)$.
  - Run $(\boldsymbol{z}, \boldsymbol{r}_{\boldsymbol{z}}) := \mathcal{P}_2(\mathsf{crs}, \mathbb{x}, \mathbb{w}, a, c, \rho)$.
  - Define $z := (\boldsymbol{z}, \boldsymbol{r}_{\boldsymbol{z}})$.
  - Output $\pi := (a, c, z)$.

– $\mathcal{V}^{\mathrm{RO}}(\mathsf{sid}, \mathbb{x}, \pi) \to 1/0$.
  - Parse $(a, c, z) := \pi$.
  - Parse RO as $(\mathcal{H}_{\mathsf{ek}}, \mathcal{H}_{\mathcal{V}})$.
  - Compute $\mathsf{ek} := \mathcal{H}_{\mathsf{ek}}(\mathsf{sid}, \mathbb{x})$.
  - Output $(c = \mathcal{H}_{\mathcal{V}}(\mathsf{sid}, \mathbb{x}, a) \wedge \mathcal{V}_2(\mathsf{crs}, \mathbb{x}, a, c, z))$.

Fig. 4: Our UC-NIZK protocol for $\mathcal{R}_{\mathsf{GenLin}}$.

Then Π_{GenLin} *of Figure 4 satisfies the following :*

– **Perfect completeness** *based on the perfect completeness of the Sigma protocol;*

– **Computational zero-knowledge** *based on the honest verifier zero-knowledge of the underlying Sigma protocol, assuming* $\mathcal{H}_{\mathcal{V}}$ *to be a programmable random oracle and* AHE *satisfies oblivious sampling;*

– **(Statistical) straight-line knowledge soundness** *based on the oblivious sampleability of the underlying encryption scheme, programmability of the random oracle* $\mathcal{H}_{\mathsf{ek}}$ *and the statistical straight-line knowledge soundness of the underlying Sigma protocol.*

Proof Sketch. The perfect completeness is immediate from the perfect completeness of the underlying Sigma protocol.

Computational zero-knowledge follows in a standard Fiat-Shamir argument by sampling a random challenge c, invoking the HVZK simulator $\mathcal{S}_{\Sigma}$ of Sigma protocol on $(\mathbb{x}, c)$ and then programming $\mathcal{H}_{\mathcal{V}}$ on $(\mathsf{sid}, \mathbb{x}, a)$ s.t. it returns c. We formally demonstrate this by explicitly defining the $\mathcal{S}_1$ and $\mathcal{S}_2$ algorithms below.

$\underline{\mathcal{S}_1(\cdots):}$

– *Answering* $\mathcal{H}_{\mathsf{ek}}(\mathsf{sid}, \mathbb{x})$ *queries:* Return $\mathcal{H}_{\mathsf{ek}}(\mathsf{sid}, \mathbb{x})$.

– *Answering* $\mathcal{H}_{\mathcal{V}}(\mathsf{sid}, \mathbb{x}, a)$ *queries made by* $\mathcal{A}$*:* Return $\mathcal{H}_{\mathcal{V}}(\mathsf{sid}, \mathbb{x}, a)$.

– *Answering* $\mathcal{H}_{\mathcal{V}}(\mathsf{sid}, \mathbb{x}, a)$ *queries made by* $\mathcal{S}_2$*:* Read $(\mathsf{sid}, \mathbb{x}, a, c, z)$ from *st*. Program $\mathcal{H}_{\mathcal{V}}$ s.t. it return $\mathcal{H}_{\mathcal{V}}(\mathsf{sid}, \mathbb{x}, a) = c$. If the query is repeated in the future then return c.

$\underline{\mathcal{S}_2(\mathsf{sid}, \mathbb{x}):}$

– On input $(\mathsf{sid}, \mathbb{x})$ compute $\mathsf{ek} \leftarrow \mathcal{H}_{\mathsf{ek}}(\mathsf{sid}, \mathbb{x})$.

– Sample $c \leftarrow \mathbb{Z}_q$ and obtain simulated transcript $(a, c, z) \leftarrow \mathcal{S}_\Sigma(\mathsf{ek}, \mathbb{x}, c)$ by the invoking the HVZK simulator $\mathcal{S}$ of the Sigma protocol.
– Update st as $st := st \cup (\mathsf{sid}, \mathbb{x}, a, c, z)$. Store simulated transcript as $\mathcal{T} = \mathcal{T} \cup (\mathsf{sid}, \mathbb{x}, a, c, z)$
– Query $\mathcal{H}_\mathcal{V}(\mathsf{sid}, \mathbb{x}, a)$ to obtain c.
– Return $\pi = (a, c, z)$.

The only way an adversarial verifier can prevent zero-knowledge is if it queries the random oracle on $(\mathsf{sid}, \mathbb{x}, a)$ before the simulator programs it to output c. However, this is not possible since the first message a is determined by $\boldsymbol{s} \in \mathbb{Z}_q^n$ in the Sigma protocol. Concretely, the probability that an adversarial verifier prevents the ZK simulator from programming $\mathcal{H}_\mathcal{V}$ on a particular a is $\frac{Q_{\mathcal{H}_\mathcal{V}}}{\min(q^n, |\mathbb{G}|^m)}$, where $Q_{\mathcal{H}_\mathcal{V}}$ is the number of queries made by the adversarial verifier to the hash function $\mathcal{H}_\mathcal{V}$, a is computed by sampling $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and computing $a := \boldsymbol{S} := \boldsymbol{Y}^{\boldsymbol{s}} \in \mathbb{G}^m$.

Next, we focus on straight-line knowledge soundness. According to the definition (Section 3.3) the stateful extractor $\mathcal{E}$ has two modes $\mathcal{E}(1, \cdots)$ which programs and simulates the random oracle and $\mathcal{E}(2, \cdots)$ which extracts the witness. We define them as follows for Π_{GenLin}.

$\underline{\mathcal{E}(1, \cdots):}$

– *Answering* $\mathcal{H}_{\mathsf{ek}}(\mathsf{sid}, \mathbb{x})$ *queries:* Sample $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{AHE.Gen}(1^\kappa)$ and program $\mathcal{H}_{\mathsf{ek}}$ to return ek and store $(\mathsf{ek}, \mathsf{dk})$ in st. If the query is repeated in the future then return ek.
– *Answering* $\mathcal{H}_\mathcal{V}(\mathsf{sid}, \mathbb{x}, a)$ *queries:* Sample $c \leftarrow \mathbb{Z}_q$ and program $\mathcal{H}_\mathcal{V}$ to return c. If the query is repeated in the future then return c.

$\underline{\mathcal{E}(2, \mathsf{sid}, \mathbb{x}, \pi):}$

– On input $(\mathsf{sid}, \mathbb{x}, \pi)$ parse $(a, c, z) := \pi$ and compute $\mathsf{ek} = \mathcal{H}_\mathcal{V}(\mathsf{sid}, \mathbb{x})$.
– Abort if $\mathcal{H}_{\mathsf{ek}}(\mathsf{sid}, \mathbb{x}, a) \neq c$. Otherwise, retrieve dk corresponding to $(\mathsf{sid}, \mathsf{ek})$ from st and set $\mathsf{crs} = \mathsf{ek}$ and $\mathsf{td} = \mathsf{dk}$.
– Output $\mathcal{E}_\Sigma(\mathsf{crs}, \mathsf{td}, \mathbb{x}, \pi)$.

Now we argue why the extraction works. First, due to oblivious sampleability of the underlying encryption scheme, the public key ek is distributed identically with the ek in the actual protocol Π_{GenLin} except with negligible probability. Then, we note that with probability $1/Q_{\mathcal{H}_{\mathsf{ek}}}$, $\mathcal{E}(2, \ldots)$ does not abort in the first step, where $Q_{\mathcal{H}_{\mathsf{ek}}}$ denotes the total number of random oracle queries asked by $\mathcal{A}$ to $\mathcal{H}_{\mathsf{ek}}$. Next, if $\mathcal{A}$ can predict the output of $\mathcal{H}_\mathcal{V}(\mathsf{sid}, \mathbb{x}, a)$ without querying, then only the second abort condition is triggered, but this happens only with $\frac{Q_{\mathcal{H}_\mathcal{V}}}{q}$ probability where a can have q possibilities. Assuming, no abort is triggered, $\mathcal{E}_\Sigma$ returns a correct witness except with negligible probability $\mathsf{negl}(\lambda)$. So, if we bound both $Q_{\mathcal{H}_{\mathsf{ek}}}$ and $Q_{\mathcal{H}_\mathcal{V}}$ to be at most sub-exponential in λ the extractor $\mathcal{E}$ outputs a correct witness except with negligible probability in λ as well. □

Next, we show that Π_{GenLin} UC-securely realizes $\mathcal{F}_{\mathsf{NIZK}}$ for relation $\mathcal{R}_{\mathsf{GenLin}}$. This requires constructing a simulator against a corrupt verifier and a simulator against a corrupt prover. For the former, we simply use the NIZK simulator against a corrupt verifier from the previous subsection. For the latter, we need straight-line blackbox simulation-extractability [KZM+15] where the environment $\mathcal{Z}$ corrupts (via dummy adversary $\mathcal{A}$) the prover in session sid and sees simulated proofs from sessions where the verifier is corrupt. We need to argue that the environment $\mathcal{Z}$ still cannot distinguish the ideal world execution of sid from a real-world execution of the same session. To argue simulation-extractability, we need to show that the protocol satisfies weak-unique response property [FKMV12]. We refer to Definition 3.3 for the formal definitions of simulation-extractability and weak unique response. The formal UC-proof is more involved and we refer to Appendix. C for the full proof.

6 Concrete Instantiation of AHE using Class Groups

We instantiate our additive homomorphic encryption scheme with the class-group based PKE scheme of [CL15]. However, we need to additionally show that it satisfies our newly introduced *oblivious sampleability* and *homomorphic well-formedness* properties. All other required properties were already shown to hold in prior works, and hence we omit the details for them.

Background and Notation. We provide a brief background (which is mostly borrrowed from [KMM+23, CCL+19, BDO23]) on class-groups before recalling the encryption scheme. The class-group setting considers a finite abelian group $\widehat{\mathbb{G}}_{\mathsf{CL}}$ of unknown order $q \cdot \widehat{s}$ where $\widehat{s}$ is unknown and hard to compute and q is known. Consider a cyclic subgroup $\mathbb{F}_{\mathsf{CL}} = \langle f \rangle$ of $\widehat{\mathbb{G}}_{\mathsf{CL}}$ of order q, where q is prime. The set $\widehat{\mathbb{G}}^q_{\mathsf{CL}} = \{g^q : g \in \widehat{\mathbb{G}}_{\mathsf{CL}}\}$ is a subgroup of $\widehat{\mathbb{G}}_{\mathsf{CL}}$. Which is of order $\widehat{s}$. Therefore, $\widehat{\mathbb{G}}_{\mathsf{CL}}$ is factored as $\widehat{\mathbb{G}}_{\mathsf{CL}} \simeq \mathbb{F}_{\mathsf{CL}} \times \widehat{\mathbb{G}}^q_{\mathsf{CL}}$. Suppose $U \in \mathbb{Z}$ be an upper bound of $\widehat{s}$ which is known. Although, $\widehat{\mathbb{G}}_{\mathsf{CL}}$ is the base group, we are focusing on a cyclic subgroup $\mathbb{G}_{\mathsf{CL}}$ of $\widehat{\mathbb{G}}_{\mathsf{CL}}$, such that $\mathbb{G}_{\mathsf{CL}}$ has order $q \cdot s$ and s divides $\widehat{s}$. So, $\mathbb{F}_{\mathsf{CL}}$ is also a cyclic subgroup of $\mathbb{G}_{\mathsf{CL}}$. Consider $\mathbb{G}^q_{\mathsf{CL}} = \{g^q : g \in G\}$ which is a cyclic subgroup of $\mathbb{G}_{\mathsf{CL}}$ of order s. Now, q and s are also co-prime. Therefore, $\mathbb{G}_{\mathsf{CL}}$ can be factored as $\mathbb{G}_{\mathsf{CL}} \simeq \mathbb{F}_{\mathsf{CL}} \times \mathbb{G}^q_{\mathsf{CL}}$. Both $s, \widehat{s}$ are odd and all $s, \widehat{s}, q$ are exponential in λ. For our purpose, we consider two distributions $\mathcal{D}$ and $\mathcal{D}_q$ over $\mathbb{Z}$ such that $\{g^x \mid x \leftarrow \mathcal{D}\}$ and $\{g_q^x \mid x \leftarrow \mathcal{D}_q\}$ produce two distributions over $\mathbb{G}_{\mathsf{CL}}$ and $\mathbb{G}^q_{\mathsf{CL}}$ respectively, which are statistically close (within distance $2^{-\lambda_{\mathsf{st}}}$, for a statistical security parameter λ_{st}, typically set to 40 in practice) to uniform distributions over $\mathbb{G}_{\mathsf{CL}}$ and $\mathbb{G}^q_{\mathsf{CL}}$ respectively. While discrete log is hard in groups $\widehat{\mathbb{G}}_{\mathsf{CL}}, \mathbb{G}_{\mathsf{CL}}, \widehat{\mathbb{G}}^q_{\mathsf{CL}}, \mathbb{G}^q_{\mathsf{CL}}$, it is easy in $\mathbb{F}_{\mathsf{CL}}$. Precisely, there are two efficient algorithms:

– $(U, \widehat{\mathbb{G}}_{\mathsf{CL}}, \mathbb{F}_{\mathsf{CL}}, f, g_q, \mathcal{D}, \mathcal{D}_q, \rho) \leftarrow \mathbf{Gen}(1^\lambda, 1^{\lambda_{\mathsf{st}}}, q)$. This algorithm, on input the computational security parameter λ, the statistical security parameter λ_{st}, and a prime q, outputs the group parameters and the randomness ρ used to generate them. For convenience, we include the descriptions of the distributions $\mathcal{D}$ and $\mathcal{D}_q$ as well.

– $x \leftarrow \mathbf{Solve}(X = f^x, U, q, \widehat{\mathbb{G}}_{\mathsf{CL}}, \mathbb{F}_{\mathsf{CL}}, g_q, f)$. This algorithm solve discrete log problem deterministically and efficiently in the group $\mathbb{F}_{\mathsf{CL}}$.

Construction. Now we are ready to describe the encryption scheme $\mathsf{CG\text{-}AHE} := (\mathsf{CG.Gen}, \mathsf{CG.Enc}, \mathsf{CG.Dec})$ for message space $\mathcal{M} = \mathbb{Z}_q$, and encryption key-space $\mathcal{K}_{\mathsf{ek}} = \mathbb{G}^q_{\mathsf{CL}}$. Let $\mathsf{pp}_{\mathsf{CG}} := (U, q, \widehat{\mathbb{G}}_{\mathsf{CL}}, \mathbb{F}_{\mathsf{CL}}, \widehat{\mathbb{G}}^q_{\mathsf{CL}}, g_q, f, \mathcal{D}, \mathcal{D}_q, \rho) \leftarrow \mathbf{Gen}(1^\lambda, 1^{\lambda_{\mathsf{st}}}, q)$ for some computational security parameter λ and for some statistical security parameter λ_{st} and a prime q. The the scheme $\mathsf{CG\text{-}AHE}$ is described as:

$\mathsf{CG.Gen}(\mathsf{pp}_{\mathsf{CG}}) \to (\mathsf{dk}, \mathsf{ek})$:
- $\mathsf{dk} \xleftarrow{\$} \mathcal{D}_q$
- $\mathsf{ek} \leftarrow g_q^{\mathsf{dk}}$

$\mathsf{CG.Enc}(\mathsf{pp}_{\mathsf{CG}}, \mathsf{ek}, m) \to c$:
- $r \xleftarrow{\$} \mathcal{D}_q$
- $R \leftarrow g_q^r$
- $E \leftarrow f^m \cdot \mathsf{ek}^r$
- Set $c := (R, E)$

$\mathsf{CG.Dec}(\mathsf{pp}_{\mathsf{CG}}, \mathsf{dk}, c) \to m$:
- Parse $c = (R, E)$
- $M \leftarrow \frac{E}{R^{\mathsf{dk}}}$
- $m \leftarrow \mathbf{Solve}(\mathsf{pp}_{\mathsf{CG}}, M)$

We show that $\mathsf{CG\text{-}AHE}$ encryption satisfies all properties required, as described in Section 3.1. First, we note that the perfect correctness, semantic security and additive homomorphism are already shown in [CL15, BDO23, KMM+23, CLT18, CCL+19, CCL+20] based on computational assumptions, such as hard subgroup membership and/or strong root assumption. So, here we only show *homomorphic well-formedness* and *oblivious sampleability of the encryption key*, both of which holds unconditionally except with negligible probability in λ_{st}.

Homomorphic Well-formedness of $\mathsf{CG\text{-}AHE}$. We first recall the homomorphic property of $\mathsf{CG\text{-}AHE}$, where Add and ScMult are specified as:

– $\mathsf{Add}(c_1, c_2)$: Parse $c_1 = (R_1, E_1)$ and $c_2 = (R_2, E_2)$. Then compute $R_{(+)} := R_1 \cdot R_2$ and $E_{(+)} := E_1 \cdot E_2$. Output $c_{(+)} = (R_{(+)}, E_{(+)})$.

– $\mathsf{ScMult}(s, c)$: Parse $c = (R, E)$, $s \in \mathbb{Z}_q$ and compute $R_{(\cdot)} := R^s$ and $E_{(\cdot)} := E^s$. Output $c_{(\cdot)} = (R_{(\cdot)}, E_{(\cdot)})$. Note that, s can just be parsed as an integer for R^s operation, since this is in a cyclic group.

First note that, unlike ElGamal encryption, $\mathsf{CG\text{-}AHE}$ does not have *dense ciphertexts*, which is the property that for any element $c \in \mathcal{C}$, we can get $m \leftarrow \mathsf{CG.Dec}(\mathsf{pp}_{\mathsf{CG}}, \mathsf{dk}, c)$. We call such successfully decryptable ciphertexts, *valid ciphertexts*. However, for $\mathsf{CG\text{-}AHE}$, not all ciphertexts are valid. For example, choose a random ciphertext $(R, E) \overset{\$}{\leftarrow} \mathcal{C}$. Then, $R = g_q^r$ and $E = g^e$. The operation E/R^{dk} yields g^δ which, with overwhelming probability, is not in the easy group $\mathbb{F}_{\mathsf{CL}}$. Hence **Solve** will fail in $\mathsf{CG.Dec}$.

Now, we argue $\mathsf{CG\text{-}AHE}$ has the homomorphic well-formedness property. Fix two arbitrary ciphertexts $c_1, c_2 \in \mathcal{C}$. Then sample a uniform random $s \overset{\$}{\leftarrow} \mathbb{Z}_q$, and compute $c^* := \mathsf{Add}(c_1, \mathsf{ScMult}(s, c_2))$. Now, if $c^* = (R^*, E^*)$ is valid, that implies $E^*/R^{*\mathsf{dk}}$ is in $\mathbb{F}_{\mathsf{CL}}$. This implies that E^* must be of the form $g_q^{r^*\mathsf{dk}} f^{m^*}$ for some $m^* \in \mathbb{Z}_q$ and $r^* \in \mathbb{Z}_s$. We have:

$$g_q^{r^*\mathsf{dk}} f^{m^*} = E_1 \cdot E_2^s \quad \text{and} \quad g_q^{r^*} = R^* = R_1 \cdot R_2^s.$$

We can write $R_1 = g_q^{r_1}$ and $R_2 = g_q^{r_2}$ such that $r^* = r_1 + sr_2$ and $E_1 = g_q^{r'_1} f^{m_1}$ and $E_2 = g_q^{r'_2} f^{m_2}$, such that $r'_1 + sr'_2 = r^* = r_1 + sr_2$. Now, since s is chosen uniformly at random once c_1, c_2 are fixed, this equation must be identically zero expect with probability $1/q = O(\mathsf{negl}(\lambda_{\mathsf{st}}))$. Which implies that, with overwhelming probability we have $r_1 = r'_1$ and $r_2 = r'_2$. This, in turn, implies that for each i E_i/R_i^{dk} is in $\mathbb{F}_{\mathsf{CL}}$, and hence both c_1, c_2 are valid ciphertexts. This concludes the argument that $\mathsf{CG\text{-}AHE}$ has homomorphic well-formedness unconditionally with overwhelming probability over the choices of randomnesses.

Oblivious Sampleability of encryption key. Recall that oblivious sampleability property requires existence of a hash function $\mathcal{H}_{\mathsf{ek}}^{\mathsf{pp}_{\mathsf{CG}}} : \{0,1\}^* \rightarrow \mathbb{G}_{\mathsf{CL}}^q$ such that the public key sampled using $\mathsf{ek} \leftarrow \mathcal{H}_{\mathcal{V}}^{\mathsf{pp}_{\mathsf{CG}}}(\cdots)$ is statistically close to the public key sampled as $\mathsf{ek} \leftarrow \mathsf{CG.Gen}(\mathsf{pp}_{\mathsf{CG}})$. This follows from two facts:

– The public key space $\mathcal{K}_{\mathsf{ek}} = \mathbb{G}_{\mathsf{CL}}^q$ is, with overwhelming probability over λ_{st}, a *dense public key space* – this implies that a uniform random element ek sampled from $\mathbb{G}_{\mathsf{CL}}^q$ is a valid public key, for which there is a secret decryption key (maybe unknown) dk, such that $\mathsf{ek} = g_q^{\mathsf{dk}}$. This follows from the description of the distribution of $\mathcal{D}_q$. Note that, the non-oblivious $\mathsf{CG.Gen}$ samples $\mathsf{dk} \leftarrow \mathcal{D}_q$, and it guarantees that this results into a $\mathsf{ek} = g_q^{\mathsf{dk}}$ which, has a statistically close distribution to the uniform distribution over $\mathbb{G}_{\mathsf{CL}}^q$.
– There exists an efficient hash function, which maps to $\mathbb{G}_{\mathsf{CL}}^q$ such that, on uniform random input, the output distribution is statistically close to uniformly random. Such an instantiation has been recently proposed in a couple of recent works [Wes19, SBK24, CLR24]. We refer to [SBK24], where it is briefly mentioned how such an hash can be used to sample class-group public keys obliviously, also required in a prior work [TCLM21].

Combining these two facts we can use the hash function mentioned above to enable oblivious sampling. In fact, the hash function can be used as a programmable random oracle in the proof as required – this is shown by Chalkias et al. [CLR24] (see Theorem 1 in the paper [CLR24]).

Discussion on our choice. We elaborate on our choice of class-group based AHE. First we note that, among existing AHE candidates, exponentiated ElGamal does not support large message space $\mathbb{Z}_q$ efficiently. The Paillier encryption scheme falls short as it does not have a dense public-key space, and therefore does not satisfy the crucial oblivious sampleability property. Other prominent AHE candidates come form lattice-based cryptography, such as Regev's [Reg04] encryption, GPV [GPV08] etc. While their basic versions only support bit-encryption, there are optimization techniques to pack large plaintext. However, a major issue with these schemes are the existence of noise. This is because, in our protocols (Figs. 2 and 4), we need to use the linear relation between the encryption randomness. For lattice based encryption, the encryption randomness additionally contains noise terms, that come from Gaussian distributions. So it is not at all clear, how the linear relation check via re-encrypting would work, and how the security would hold up. We leave exploring lattice-based (and other AHE) schemes in our framework as future works. In contrast, the class-group based encryption satisfies all our requirements in fairly straightforward manner.

7 Application of our UC-NIZK

We demonstrate concrete applications of our compiler by applying to the well-known Chaum-Pedersen Protocol [CP93]. We make it UC-secure at the cost of two additional encryptions without performing any repetition of the original Chaum-Pedersen protocol. It can be used in the works of [TGL+19, BGJP23]. Our UC-NIZK for Chaum-Pedersen is as follows:

– **Input:** Prover and verifier have input statement $\mathbb{x} = (g, h, W_1, W_2)$. Prover has secret witness $\mathbb{w} = w$ s.t. $W_1 = g^w$ and $W_2 = h^w$.
– **Primitives:** $\mathcal{H}_{\mathcal{V}}$ and $\mathcal{H}_{\mathsf{ek}}$ are random oracles. AHE is the additively homomorphic encryption scheme.
– **Setup:** A key pair $(\mathsf{ek}, \mathsf{dk})$ for AHE.
– **Prover Algorithm:** $\mathcal{P}(\mathsf{ek}, \mathbb{x}, \mathbb{w}) \to a$:
  1. Compute $\mathsf{ek} := \mathcal{H}_{\mathsf{ek}}(\mathsf{sid}, \mathbb{x})$.
  2. Sample $s \leftarrow \mathbb{Z}_q$. Compute $S_1 := g^s$ and $S_2 := h^s$.
  3. Sample encryption randomnesses r_s, r_w.
  4. Compute $C_s := \mathsf{Enc}(\mathsf{ek}, s, r_s)$ and $C_w := \mathsf{Enc}(\mathsf{ek}, w, r_w)$.
  5. Set $a := (S, C_s, C_w)$ to $\mathcal{V}$.
  6. Compute $c := \mathcal{H}_{\mathcal{V}}(\mathsf{sid}, \mathbb{x}, a)$.
  7. Compute $z := s + c \cdot w \in \mathbb{Z}_q$, and send z to $\mathcal{V}$.
  8. Compute $r_z := r_s + c \cdot r_w \in \mathbb{Z}_q$, and send (a, c, z, r_z) to $\mathcal{V}$.
– **Verifier Algorithm:** $\mathcal{V}(\mathsf{ek}, \mathbb{x}, (a, c, (z, r_z))) \to b$:
  1. Compute $\mathsf{ek} := \mathcal{H}_{\mathsf{ek}}(\mathsf{sid}, \mathbb{x})$.
  2. Check that:
     • $g^z = S_1 \cdot W_1^c \in \mathbb{G}$; and
     • $h^z = S_2 \cdot W_2^c \in \mathbb{G}$; and
     • C_s, C_w are valid AHE ciphertexts; and
     • $\mathsf{Enc}(\mathsf{ek}, z, r_z) = C_s + c \cdot C_z$.
  3. If all checks pass, output 1. Otherwise, output 0.

It can be observed that the above protocol is a specific instantiation of Π_{GenLin} for $n = 1$ and $m = 2$ and we incurred the cost of two additional encryptions. A generalized version of Chaum-Pedersen where the same witness is used to prove m statements can be similarly considered as $\mathbb{x} = (g_i, g_i^w)$ for secret witness $\mathbb{w} = w$. In that case, our compiler still incurs two encryptions as overhead for UC security for those m statements. That would amortize the cost of those two encryptions over m statements since the entire Chaum-Pedersen Proof would be dominated by the cost of $2m$ exponentiations over group $\mathbb{G}$. This also captures AND composition using our compiler. The Schnorr's protocol can be found in Sec. 2 where $n = 1$ and $m = 1$. We refer to the *Applications* paragraph in Sec. 1.1 for the concrete applications of Schnorr's and Chaum-Pedersen's proof of knowledge. We provide the OR composition using our compiler in Appendix. E. This will improve existing works like [TGL+19] where both OR composition and Chaum-Pedersen is used.

References

ABK$^+$21. Michel Abdalla, Manuel Barbosa, Jonathan Katz, Julian Loss, and Jiayu Xu. Algebraic adversaries in the universal composability framework. LNCS, pages 311–341. Springer, Heidelberg, 2021.

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.

BBF19. Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 561–586. Springer, Heidelberg, August 2019.

BDO23. Lennart Braun, Ivan Damgård, and Claudio Orlandi. Secure multiparty computation from threshold encryption based on class groups. LNCS, pages 613–645. Springer, Heidelberg, 2023.

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.

BFS20. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 677–706. Springer, Heidelberg, May 2020.

BGJP23. James Bartusek, Sanjam Garg, Abhishek Jain, and Guru-Vamsi Policharla. End-to-end secure messaging with traceability only for illegal content. LNCS, pages 35–66. Springer, Heidelberg, 2023.

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.

BLS01. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

BS23. Dan Boneh and Victor Shoup. A graduate course in applied cryptography. https://toc.cryptobook.us/book.pdf, 2023. (Version 0.6).

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.

CCL$^+$19. Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Two-party ECDSA from hash proof systems and efficient instantiations. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 191–221. Springer, Heidelberg, August 2019.

CCL$^+$20. Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA. In *PKC 2020, Part II*, LNCS, pages 266–296. Springer, Heidelberg, 2020.

CD24. Ignacio Cascudo and Bernardo David. Publicly verifiable secret sharing over class groups and applications to DKG and YOSO. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part V*, volume 14655 of *Lecture Notes in Computer Science*, pages 216–248. Springer, 2024.

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994.

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.

CGKN21. Konstantinos Chalkias, François Garillot, Yashvanth Kondi, and Valeria Nikolaenko. Non-interactive half-aggregation of EdDSA and variants of Schnorr signatures. LNCS, pages 577–608. Springer, Heidelberg, 2021.

CJS14. Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 597–608. ACM Press, November 2014.

CL15. Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from DDH. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 487–505. Springer, Heidelberg, April 2015.

CL24. Yi-Hsiu Chen and Yehuda Lindell. Optimizing and implementing fischlin's transform for uc-secure zero-knowledge. *IACR Cryptol. ePrint Arch.*, page 526, 2024.

CLR24. Kostas Kryptos Chalkias, Jonas Lindstrøm, and Arnab Roy. An efficient hash function for imaginary class groups. Cryptology ePrint Archive, Paper 2024/295, 2024. https://eprint.iacr.org/2024/295.

CLT18. Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 733–764. Springer, Heidelberg, December 2018.

Coh93. Henry Cohen. A course in computational algebraic number theory. *Springer Berlin, Heidelberg*, GTM, volume 138:XXI, 536, 1993.

CP93. David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Heidelberg, August 1993.

CPSV16. Michele Ciampi, Giuseppe Persiano, Luisa Siniscalchi, and Ivan Visconti. A transform for NIZK almost as efficient and general as the Fiat-Shamir transform without programmable random oracles. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 83–111. Springer, Heidelberg, January 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,*, pages 670–690, 2020.

CS04. John F. Canny and Stephen Sorkin. Practical large-scale distributed key generation. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 138–152. Springer, Heidelberg, May 2004.

CSW22. Ran Canetti, Pratik Sarkar, and Xiao Wang. Triply adaptive UC NIZK. LNCS, pages 466–495. Springer, Heidelberg, 2022.

Dam92. Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, Heidelberg, August 1992.

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

DDO$^+$01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001.

DG03. Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *35th ACM STOC*, pages 426–437. ACM Press, June 2003.

FHJ20. Marc Fischlin, Patrick Harasser, and Christian Janson. Signatures from sequential-OR proofs. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, LNCS, pages 212–244. Springer, Heidelberg, May 2020.

Fis05. Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, August 2005.

FKL18. Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

FKMV12. Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, volume 7668 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2012.

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.

GJKR99. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 295–310. Springer, Heidelberg, May 1999.

GKO$^+$23. Chaya Ganesh, Yashvanth Kondi, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Witness-succinct universally-composable SNARKs. LNCS, pages 315–346. Springer, Heidelberg, 2023.

GLOW21. David Galindo, Jia Liu, Mihai Ordean, and Jin-Mann Wong. Fully distributed verifiable random functions and their application to decentralised random beacons. In *IEEE European Symposium on Security and Privacy, EuroS&P 2021, Vienna, Austria, September 6-10, 2021*, pages 88–102. IEEE, 2021.

GMY03. Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 177–194. Springer, Heidelberg, May 2003.

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.

GOP$^+$22. Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). LNCS, pages 397–426. Springer, Heidelberg, 2022.

GOP$^+$23. Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat-shamir bulletproofs are non-malleable (in the random oracle model). Cryptology ePrint Archive, Paper 2023/147, 2023. https://eprint.iacr.org/2023/147.

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.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, page 197–206, New York, NY, USA, 2008. Association for Computing Machinery.

Gro06. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006.

JJ21. Abhishek Jain and Zhengzhong Jin. Non-interactive zero knowledge from sub-exponential DDH. LNCS, pages 3–32. Springer, Heidelberg, 2021.

JP14. Abhishek Jain and Omkant Pandey. Non-malleable zero knowledge: Black-box constructions and definitional relationships. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 435–454. Springer, Heidelberg, September 2014.

JT20. Joseph Jaeger and Stefano Tessaro. Expected-time cryptography: Generic techniques and applications to concrete soundness. LNCS, pages 414–443. Springer, Heidelberg, March 2020.

Kat21. Shuichi Katsumata. A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure NIZKs. LNCS, pages 580–610. Springer, Heidelberg, 2021.

KG20. Chelsea Komlo and Ian Goldberg. FROST: flexible round-optimized schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, volume 12804 of *Lecture Notes in Computer Science*, pages 34–65. Springer, 2020.

KKK21. Thomas Kerber, Aggelos Kiayias, and Markulf Kohlweiss. Composition with knowledge assumptions. LNCS, pages 364–393. Springer, Heidelberg, 2021.

KMM$^+$23. Aniket Kate, Easwar Vivek Mangipudi, Pratyay Mukherjee, Hamza Saleem, and Sri Aravinda Krishnan Thyagarajan. Non-interactive vss using class groups and application to dkg, 2023. https://eprint.iacr.org/2023/451.

KMMM23. Aniket Kate, Easwar Vivek Mangipudi, Siva Maradana, and Pratyay Mukherjee. FlexiRand: Output private (distributed) VRFs and application to blockchains. pages 1776–1790. ACM Press, 2023.

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.

Ks22. Yashvanth Kondi and abhi shelat. Improved straight-line extraction in the random oracle model with applications to signature aggregation. LNCS, pages 279–309. Springer, Heidelberg, 2022.

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.*, page 1093, 2015.

Lin15. Yehuda Lindell. An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 93–109. Springer, Heidelberg, March 2015.

Lin22. Yehuda Lindell. Simple three-round multiparty schnorr signing with full simulatability. Cryptology ePrint Archive, Report 2022/374, 2022. https://eprint.iacr.org/2022/374.

LR22a. Anna Lysyanskaya and Leah Namisa Rosenbloom. Efficient and universally composable non-interactive zero-knowledge proofs of knowledge with security against adaptive corruptions. *IACR Cryptol. ePrint Arch.*, page 1484, 2022.

LR22b. Anna Lysyanskaya and Leah Namisa Rosenbloom. Universally composable Σ-protocols in the global random-oracle model. LNCS, pages 203–233. Springer, Heidelberg, 2022.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755. Springer, Heidelberg, April 2012.

Mau09. Ueli M. Maurer. Unifying zero-knowledge proofs of knowledge. In Bart Preneel, editor, *AFRICACRYPT 09*, volume 5580 of *LNCS*, pages 272–286. Springer, Heidelberg, June 2009.

Mau11. Ueli Maurer. Constructive cryptography - A new paradigm for security definitions and proofs. In Sebastian Mödersheim and Catuscia Palamidessi, editors, *Theory of Security and Applications - Joint Workshop, TOSCA 2011, Saarbrücken, Germany, March 31 - April 1, 2011, Revised Selected Papers*, volume 6993 of *Lecture Notes in Computer Science*, pages 33–56. Springer, 2011.

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.

ORV14. Rafail Ostrovsky, Vanishree Rao, and Ivan Visconti. On selective-opening attacks against encryption schemes. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 578–597. Springer, Heidelberg, September 2014.

Pas03. Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Heidelberg, August 2003.

PR05. Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 533–542. ACM Press, May 2005.

PS96. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Heidelberg, May 1996.

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.

Reg04. Oded Regev. A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space. arXiv:quant-ph/0406151, June 2004.

RWGM23. Michael Rosenberg, Jacob D. White, Christina Garman, and Ian Miers. zk-creds: Flexible anonymous credentials from zkSNARKs and existing identity infrastructure. pages 790–808. IEEE Computer Society Press, 2023.

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

SBK24. István András Seres, Péter Burcsi, and Péter Kutas. How (not) to hash into class groups of imaginary quadratic fields? Cryptology ePrint Archive, Paper 2024/034, 2024. https://eprint.iacr.org/2024/034.

Sch91. Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

SV12. Alessandra Scafuro and Ivan Visconti. On round-optimal zero knowledge in the bare public-key model. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 153–171. Springer, Heidelberg, April 2012.

TCLM21. Sri AravindaKrishnan Thyagarajan, Guilhem Castagnos, Fabien Laguillaumie, and Giulio Malavolta. Efficient cca timed commitments in class groups. Cryptology ePrint Archive, Paper 2021/1272, 2021. https://eprint.iacr.org/2021/1272.

TGL$^+$19. Nirvan Tyagi, Paul Grubbs, Julia Len, Ian Miers, and Thomas Ristenpart. Asymmetric message franking: Content moderation for metadata-private end-to-end encryption. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 222–250. Springer, Heidelberg, August 2019.

Wes19. Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 379–407. Springer, Heidelberg, May 2019.

Appendix

A Additional Proofs

A.1 Proof of Theorem 1

We show that the protocol in Figure 1 satisfies completeness, special soundness, and honest-verifier zero knowledge.

Completeness. From the notations and definitions presented in Section 3, we obtain correctness of the verification check:

$$\boldsymbol{Y}^{\boldsymbol{z}} = \boldsymbol{Y}^{\boldsymbol{s}+c\boldsymbol{w}} = \boldsymbol{Y}^{\boldsymbol{s}} \bullet \boldsymbol{Y}^{c\boldsymbol{w}} = \boldsymbol{S} \bullet \boldsymbol{U}^{c} \ .$$

The correctness of the verification equation implies that our protocol is complete.

Special Soundness. The deterministic efficient witness extractor $\mathcal{E}$ takes as input the statement $\mathbb{x} := (\boldsymbol{Y} \in \mathbb{G}^{n\times m}, \boldsymbol{U} \in \mathbb{G}^{m})$ and two accepting transcripts $(\boldsymbol{S}, c, \boldsymbol{z})$ and $(\boldsymbol{S}, c', \boldsymbol{z'})$ where $c \neq c'$. (The extractor obtain two such transcripts with the same first message since it has rewinding access to the prover.) As both the transcripts are accepting, the witness extractor $\mathcal{E}$ has two equations:

$$\boldsymbol{Y}^{\boldsymbol{z}} = \boldsymbol{S} \bullet \boldsymbol{U}^{c} \text{ and } \boldsymbol{Y}^{\boldsymbol{z'}} = \boldsymbol{S} \bullet \boldsymbol{U}^{c'} \ .$$

Now the $\mathcal{E}$ computes the **vector inverse** $\boldsymbol{Y}^{-\boldsymbol{z'}}$ of $\boldsymbol{Y}^{\boldsymbol{z'}}$ and computes $\boldsymbol{Y}^{\boldsymbol{z}-\boldsymbol{z'}} := \boldsymbol{Y}^{\boldsymbol{z}} \bullet (\boldsymbol{Y}^{-\boldsymbol{z'}})$. Then, $\mathcal{E}$ gets

$$\boldsymbol{Y}^{\boldsymbol{z}-\boldsymbol{z'}} = \boldsymbol{S} \bullet \boldsymbol{U}^{c} \bullet (\boldsymbol{S} \bullet \boldsymbol{U}^{c'})^{-1} = \boldsymbol{U}^{c} \bullet \boldsymbol{S} \bullet \boldsymbol{S}^{-1} \bullet \boldsymbol{U}^{-c'} = \boldsymbol{U}^{c-c'} \ ,$$

in which the second equality follows since $\mathbb{G}$ is cyclic and therefore commutative. As $c \neq c'$, $\mathcal{E}$ computes $\frac{1}{c-c'} \in \mathbb{Z}_q$, then gets

$$\boldsymbol{U} = (\boldsymbol{Y}^{\boldsymbol{z}-\boldsymbol{z'}})^{\frac{1}{c-c'}} = \boldsymbol{Y}^{\frac{\boldsymbol{z}-\boldsymbol{z'}}{c-c'}} \ .$$

The witness extractor $\mathcal{E}$ gets the witness $\boldsymbol{w} := \frac{\boldsymbol{z}-\boldsymbol{z'}}{c-c'}$ as long as $c \neq c'$; this bad event occurs with $\frac{1}{|\mathbb{G}|} = \frac{1}{q}$ probability. $\mathcal{E}$ outputs $\boldsymbol{w}$ satsifying $\mathcal{R}_{\mathsf{GenLin}}$ with probabilty $1 - \frac{1}{q}$. Hence for sufficiently large q, the protocol satisfies special soundness.

Honest Verifier Zero Knowledge. There exists an efficient PPT algorithm called the **simulator** which takes as input the statement $\boldsymbol{U} \in \mathbb{G}^{m}$ and a random challenge $c \overset{\$}{\leftarrow} \mathbb{Z}_q$. Now it computes $\boldsymbol{z} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n}$ and $\boldsymbol{S} = \boldsymbol{Y}^{\boldsymbol{z}} \bullet (\boldsymbol{U}^{-1})^{c}$, where $\boldsymbol{U}^{-1} = (u_1^{-1}, \cdots, u_m^{-1})$ and u_i^{-1} is inverse of u_i in group $\mathbb{G}$. Then the **simulator** outputs the triplet $(\boldsymbol{S}, c, \boldsymbol{z})$. The protocol is HVZK as $(\boldsymbol{S}, c, \boldsymbol{z})$ is distributed identically to a real transcript of the conversation between the prover and the verifier.

A.2 Reduction for computational ZK for Theorem 2

Suppose that there exists a PPT distinguisher $\mathcal{D}$ between the transcripts. Then, we can construct a reduction to the semantic security of AHE, via sending challenge messages replacing the ciphertext $\boldsymbol{C}_{\boldsymbol{w}}$ in the transcript. Specifially, construct an adversary $\mathcal{A}_{\mathsf{AHE}}$ breaking the semantic security of AHE as follows:

$\underline{\mathcal{A}_{\mathsf{AHE}}(\mathsf{ek}):}$

– Generate a Σ-protocol transcript as follows:

- Simulate a $\mathcal{R}_{\mathsf{GenLin}}$ instance-witness pair $(\mathbb{x} = (\boldsymbol{Y}, \boldsymbol{U}), \mathbb{w} = \boldsymbol{w})$ by computing $\boldsymbol{Y} \xleftarrow{\$} \mathbb{G}^{n \times m}$ and $\boldsymbol{w} \xleftarrow{\$} \mathbb{Z}^n$ and $\boldsymbol{U} := \boldsymbol{Y}^{\boldsymbol{w}} \in \mathbb{G}^m$.
- Compute $\boldsymbol{S} := \boldsymbol{Y}^{\boldsymbol{z}} \bullet (\boldsymbol{U}^{-1})^c$.
- Define messages $m_0 = 0^n$, $m_1 = \boldsymbol{w}$ and send them to the AHE semantic security challenger.
- Receive the ciphertext $\boldsymbol{C}_{\boldsymbol{w}} \in \mathcal{C}^n$ (for either m_0 or m_1) from the AHE semantic security challenger.
- Sample $c \xleftarrow{\$} \mathbb{Z}_q$ and $\boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_q^n$.
- Compute $\boldsymbol{r}_{\boldsymbol{z}} := \boldsymbol{r}_{\boldsymbol{s}} + c \cdot \boldsymbol{r}_{\boldsymbol{w}} \in \mathcal{R}^n$, in which $\boldsymbol{r}_{\boldsymbol{s}}, \boldsymbol{r}_{\boldsymbol{w}} \xleftarrow{\$} \mathcal{R}^n$.
- Compute $\boldsymbol{C}_{\boldsymbol{z}} := \mathsf{MatEnc}(\mathsf{ek}, \boldsymbol{z}; \boldsymbol{r}_{\boldsymbol{z}})$.
- Compute homomorphically $\boldsymbol{C}_s := \mathsf{Add}(\boldsymbol{C}_z, -\mathsf{ScMult}(c, \boldsymbol{C}_{\boldsymbol{w}}))$.

– Send the transcript $(a := (\boldsymbol{S}, \boldsymbol{C}_{\boldsymbol{s}}, \boldsymbol{C}_{\boldsymbol{w}}), c, z := (\boldsymbol{z}, \boldsymbol{r}_{\boldsymbol{z}}))$ to $\mathcal{D}$.
– If $\mathcal{D}$ outputs 0, output "simulated". Else, output "real".

Clearly, $\mathcal{A}_{\mathsf{AHE}}(\mathsf{ek})$ generates the simulated transcript when $\boldsymbol{C}_{\boldsymbol{w}}$ encrypts 0^n or the transcript of the real protocol when $\boldsymbol{C}_{\boldsymbol{w}}$ encrypts $\boldsymbol{w}$. Hence, $\mathcal{A}_{\mathsf{AHE}}(\mathsf{ek})$ distinguishes the encryptions of m_0 and m_1 with the same probability as $\mathcal{D}$.

B Universally Composable Security

We recall the standard Universal Composability framework of Canetti [Can01], with static corruptions, for the two-party setting. And we conclude this section with the definition of $\mathcal{F}$-hybrid model, which is instrumental for security proofs in the UC model.

B.1 Static Security in the UC Model

In this model, the real world execution of protocol π is carried out between the honest parties P_1 and P_2 and an adversary $\mathcal{A}$, in the presence of an external entity called the environment $\mathcal{Z}$. All the parties are PPT Turing machines and $\mathcal{Z}$ has an auxiliary information z. At the outset of the protocol the environment initiates the parties with inputs and provides some initial information to $\mathcal{A}$. $\mathcal{Z}$ is allowed to interact with $\mathcal{A}$ throughout the protocol. At the outset of the protocol, $\mathcal{A}$ may or may not corrupt a party. Upon corruption of a party, $\mathcal{A}$ gets access to the internal state and input of that party. From now on the party will behave according to $\mathcal{A}$'s instructions (since we are in the malicious model). At the end of the protocol, the honest parties send their output to $\mathcal{Z}$ while $\mathcal{A}$ outputs $\perp$ on behalf of the corrupted parties and its internal state to $\mathcal{Z}$. We denote the view of $\mathcal{Z}$ as $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(1^\kappa, z)$.

In the ideal world, we consider the honest parties P_1 and P_2, a PPT adversary Sim, $\mathcal{Z}$ and the functionality $\mathcal{F}$. Sim has a random tape r and security parameter κ. He simulates the role of $\mathcal{A}$ in the ideal world and whenever $\mathcal{A}$ corrupts a party in the real world Sim corrupts that party in the ideal world and gets access to its internal state. Sim invokes the algorithm of $\mathcal{A}$, in his head, in another internal protocol execution where Sim simulates the view of the honest parties to $\mathcal{A}$. We will denote this internal copy of $\mathcal{A}$ as $\mathsf{Adv}_{\mathsf{Int}}$. Based on the reply of $\mathsf{Adv}_{\mathsf{Int}}$ in the internal execution, Sim behaves accordingly in the ideal world execution. He extracts the inputs of the corrupted parties in the internal execution and invokes $\mathcal{F}$ in the ideal world with those inputs to obtain the output. In the internal execution he simulates the protocol in such a way that $\mathsf{Adv}_{\mathsf{Int}}$ obtains that output. At the end of the protocol, $\mathsf{Adv}_{\mathsf{Int}}$ forwards his view to Sim who forwards it to $\mathcal{Z}$. We denote the view of $\mathcal{Z}$ as $\text{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}}(1^\kappa, z)$. We say that a protocol π UC-securely implements a functionality $\mathcal{F}$ in the presence of static adversaries if the real world and ideal world views are indistinguishable.

Definition 1. *Let π be a protocol for computing a functionality $\mathcal{F}$. We say that π UC-securely computes the two party protocol functionality $\mathcal{F}$ in the presence of static adversaries if for every PPT adaptive real-world adversary $\mathcal{A}$ and every environment $\mathcal{Z}$, there exists a PPT ideal-world adversary Sim, such that:*

$$\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(1^\kappa, z) \overset{c}{\approx} \text{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{Z}}(1^\kappa, z)$$

B.2 The $\mathcal{F}$-hybrid model.

In order to construct our protocols, we utilize other secure two-party protocols as subprotocols. The standard way of doing this is to work in a "*hybrid model*" where both the parties interact with each other (as in the real model) in the outer protocol and use ideal functionality calls (as in the ideal world) for the subprotocols. The UC composition theorem states that if a protocol ρ UC-securely implements a functionality $\mathcal{F}$, then any execution of ρ in a bigger protocol can be replaced with ideal calls to the functionality $\mathcal{F}$. Specifically, while constructing a protocol π that uses ρ as subprotocol, for securely computing some functionality $\mathcal{F}$, the parties can run π and invoke $\mathcal{F}$. The execution of π that invokes $\mathcal{F}$, for each execution of ρ, is called the *$\mathcal{F}$-hybrid execution of* π and is denoted as $\pi^{\mathcal{F}}$. The hybrid ensemble $\texttt{Hyb}_{\pi^{\mathcal{F}},\mathcal{A},\mathcal{Z}}(1^\kappa, z)$ describes $\mathcal{Z}$'s output after interacting with $\mathcal{A}$ and the parties running protocol $\pi^{\mathcal{F}}$. Whereas, the execution of π that considers execution of ρ is denoted as π^{ρ}. The hybrid ensemble $\texttt{Hyb}_{\pi^{\rho},\mathcal{A},\mathcal{Z}}(1^\kappa, z)$ describes $\mathcal{Z}$'s output after interacting with $\mathcal{A}$ and the parties running protocol π^{ρ}. By UC security, the two hybrids $\texttt{Hyb}_{\pi^{\mathcal{F}},\mathcal{A},\mathcal{Z}}(1^\kappa, z)$ and $\texttt{Hyb}_{\pi^{\rho},\mathcal{A},\mathcal{Z}}(1^\kappa, z)$ are indistinguishable. This permits replacing executions of ρ, in π, with ideal calls to $\mathcal{F}$ functionality; thereby allowing π to execute in the $\mathcal{F}$-hybrid model. It simplifies the security proof of $\pi^{\mathcal{F}}$ as it can be performed in the $\mathcal{F}$-hybrid model, instead of proving security of ρ within the proof of π^{ρ}.

C UC Security Proof of Π_{GenLin}

We show that Π_{GenLin} UC-securely realizes $\mathcal{F}_{\mathsf{NIZK}}$ for relation $\mathcal{R}_{\mathsf{GenLin}}$. We need to show that there is a simulator against a corrupt verifier and a simulator against a corrupt prover. For the former, we simply use the NIZK simulator against a corrupt verifier from the previous subsection. For the latter, we need straight-line blackbox simulation-extractability [KZM+15] where the environment $\mathcal{Z}$ corrupts (via dummy adversary $\mathcal{A}$) the prover in session sid and sees simulated proofs from sessions where the verifier is corrupt. We need to argue that the environment $\mathcal{Z}$ still cannot distinguish the ideal world execution of sid from a real-world execution of the same session. To argue simulation-extractability, we need to show that the protocol satisfies weak-unique response property [FKMV12]. We refer to Definition 3.3 for the formal definitions of simulation-extractability and weak unique response.

Theorem 4. *Assuming* AHE *is an additively homomorphic encryption scheme with perfect correctness, then* Π_{GenLin}*, described in Figure 4, satisfies weak unique response property (Definition 3.3).*

Proof. Let $(\mathcal{S}_1, \mathcal{S}_2)$ be the NIZK simulator for Π_{GenLin} as defined in proof of Theorem 3. Let us assume to the contrary that there is a PPT adversary $\mathcal{A}^{\mathcal{S}_1}$ which given a simulated transcript $(\mathbb{x}, \pi = (\boldsymbol{S}, \boldsymbol{C_s}, \boldsymbol{C_w}, c, \boldsymbol{z}, \boldsymbol{r_z})) \leftarrow \mathcal{S}_2(\mathbb{x})$ outputs another transcript $(\mathbb{x}, \pi' = (\boldsymbol{S}, \boldsymbol{C_s}, \boldsymbol{C_w}, c, \boldsymbol{z'}, r_{\boldsymbol{z'}}))$ such that $(\boldsymbol{z}, \boldsymbol{r_z}) \neq (\boldsymbol{z'}, r_{\boldsymbol{z'}})$, $\mathcal{V}^{\mathrm{RO}}(\mathbb{x}, \pi) = \mathcal{V}^{\mathrm{RO}}(\mathbb{x}, \pi') = 1$.

It follows from the correctness of the encryption scheme that the underlying plaintexts of $C_{\boldsymbol{s}}$ and $C_{\boldsymbol{w}}$ across the two transcripts $(\boldsymbol{S}, \boldsymbol{C_s}, \boldsymbol{C_w}, c, \boldsymbol{z}, \boldsymbol{r_z})$ and $(\boldsymbol{S}, \boldsymbol{C_s}, \boldsymbol{C_w}, c, \boldsymbol{z'}, r_{\boldsymbol{z'}})$ must be equal. This implies that given c, the value of both z and z' is equal to $\boldsymbol{s} + c \cdot \boldsymbol{w}$ and thus is the same. Moreover, the corresponding randomnesses $r_{\boldsymbol{s}}$ and $r_{\boldsymbol{w}}$ across the transcripts must also be equal as the ciphertexts in the first message are the same.That gives $\boldsymbol{r_z} = r_{\boldsymbol{s}} + c r_{\boldsymbol{w}} = r_{\boldsymbol{z'}}$. This contradicts our assumption that $(\boldsymbol{z}, \boldsymbol{r_z}) \neq (\boldsymbol{z'}, r_{\boldsymbol{z'}})$. Therefore, our straight-line extractable Sigma protocol instantiated with perfectly correct encryption satisfies **weak unique response property** the following holds:

$$\Pr\left[\begin{array}{c|c} \mathcal{V}^{\mathrm{RO}}(\mathbb{x}, \boldsymbol{S}, \boldsymbol{C_s}, \boldsymbol{C_w}, c, \boldsymbol{z'}, r_{\boldsymbol{z'}}) = 1 & (\mathbb{x}, c, \boldsymbol{S}, \boldsymbol{C_s}, \boldsymbol{C_w}, \boldsymbol{z}, \boldsymbol{r_z}) \leftarrow \mathcal{S}_2(\mathbb{x}) \\ \wedge & (\mathbb{x}, \boldsymbol{S}, \boldsymbol{C_s}, \boldsymbol{C_w}, \boldsymbol{z'}, r_{\boldsymbol{z'}}) \leftarrow \mathcal{A}^{\mathcal{S}_1} \\ (\boldsymbol{z}, \boldsymbol{r_z}) \neq (\boldsymbol{z'}, r_{\boldsymbol{z'}}) & (\mathbb{x}, c, \boldsymbol{S}, \boldsymbol{C_s}, \boldsymbol{C_w}, \boldsymbol{z}, \boldsymbol{r_z}) \end{array}\right] = 0.$$

□

Next, we show that if Π_{GenLin} satisfies the weak unique response, knowledge soundness and achieves ZK, then it satisfies simulation-extractability. The proof closely follows the techniques in [FKMV12, GOP+23].

Theorem 5. *Assuming protocol Π_{GenLin} satisfies straight-line knowledge soundness, zero-knowledge and weak unique response, Π_{GenLin} is simulation-extractable (Definition 3.3) in the random oracle model.*

Proof. Let $(\mathcal{S}_1, \mathcal{S}_2)$ (Theorem 3) be the ZK simulator for Π_{GenLin}. Let $\mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}$ be a PPT adversary that outputs $(\mathbb{x}^*, \boldsymbol{S}^*, \boldsymbol{C}^*_{\boldsymbol{s}}, \boldsymbol{C}^*_{\boldsymbol{w}}, c^*, \boldsymbol{z}^*, \boldsymbol{r}^*_{\boldsymbol{z}})$ as a valid proof for simulation-extractability having access to the ZK simulators $(\mathcal{S}_1, \mathcal{S}_2)$

Let $\mathcal{T}$ denote the list of transcripts that the adversary obtains on querying $\mathcal{S}_2$.

– Then $(\mathbb{x}^*, \boldsymbol{S}^*, \boldsymbol{C}^*_{\boldsymbol{s}}, \boldsymbol{C}^*_{\boldsymbol{w}}, c^*, \boldsymbol{z}^*, \boldsymbol{r}^*_{\boldsymbol{z}}) \notin \mathcal{T}$ and by weak unique response property, $(\mathbb{x}^*, \boldsymbol{S}^*, \boldsymbol{C}^*_{\boldsymbol{s}}, \boldsymbol{C}^*_{\boldsymbol{w}}, c^*, \boldsymbol{z}^*, \boldsymbol{r}^*_{\boldsymbol{z}}) \notin \mathcal{T}$ for any $(\boldsymbol{z}, \boldsymbol{r}_{\boldsymbol{z}}) \neq (\boldsymbol{z}^*, \boldsymbol{r}^*_{\boldsymbol{z}})$, $\mathcal{T}$ being the list of queried transcripts received from $\mathcal{S}_2$.
– This implies c^* must have been computed by making a fresh query to $\mathcal{S}_1$ on $(\mathbb{x}^*, \boldsymbol{S}^*, \boldsymbol{C}^*_{\boldsymbol{s}}, \boldsymbol{C}^*_{\boldsymbol{w}})$ and not from a query to $\mathcal{S}_2$ on $\mathbb{x}^*$.

Therefore, the proof submitted by $\mathcal{A}^{(\mathcal{S}_1,\mathcal{S}_2)}$ is also a valid proof with respect to an adversary who has access to only $\mathcal{S}_1$. In order to see that an algorithm with access to $\mathcal{S}_1$ can perfectly simulate the view of $\mathcal{A}^{(\mathcal{S}_1,\mathcal{S}_2)}$, we define two games G_0 and G_1 in the following manner:

Game G_0 :

– Run the adversary and receive a instance-proof pair
  $(\mathbb{x}^*, \pi := (\boldsymbol{S}^*, \boldsymbol{C}^*_{\boldsymbol{s}}, \boldsymbol{C}^*_{\boldsymbol{w}}, c^*, \boldsymbol{z}^*, \boldsymbol{r}^*_{\boldsymbol{z}})) \leftarrow \mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}$
– If $(\mathbb{x}^*, \boldsymbol{S}^*, \boldsymbol{C}^*_{\boldsymbol{s}}, \boldsymbol{C}^*_{\boldsymbol{w}}, c^*, \boldsymbol{z}^*, \boldsymbol{r}^*_{\boldsymbol{z}}) \notin \mathcal{T}$ and $\mathcal{V}^{\mathcal{S}_1}(\mathbb{x}^*, \boldsymbol{S}^*, \boldsymbol{C}^*_{\boldsymbol{s}}, \boldsymbol{C}^*_{\boldsymbol{w}}, c^*, \boldsymbol{z}^*, \boldsymbol{r}^*_{\boldsymbol{z}}) = 1$, return 1.
– Otherwise, return $\perp$.

Game G_1 :

– G_1 is same as G_0 except that it has an additional check.
– Run the adversary and receive a instance-proof pair
  $(\mathbb{x}^*, \pi := (\boldsymbol{S}^*, \boldsymbol{C}^*_{\boldsymbol{s}}, \boldsymbol{C}^*_{\boldsymbol{w}}, c^*, \boldsymbol{z}^*, \boldsymbol{r}^*_{\boldsymbol{z}})) \leftarrow \mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}$
– If $(\mathbb{x}^*, \boldsymbol{S}^*, \boldsymbol{C}^*_{\boldsymbol{s}}, \boldsymbol{C}^*_{\boldsymbol{w}}, c^*, \boldsymbol{z}^*, \boldsymbol{r}^*_{\boldsymbol{z}}) \notin \mathcal{T}$ and $\mathcal{V}^{\mathcal{S}_1}(\mathbb{x}^*, \boldsymbol{S}^*, \boldsymbol{C}^*_{\boldsymbol{s}}, \boldsymbol{C}^*_{\boldsymbol{w}}, c^*, \boldsymbol{z}^*, \boldsymbol{r}^*_{\boldsymbol{z}}) = 1$,
  - If $\exists\ (\mathbb{x}^*, \boldsymbol{S}^*, \boldsymbol{C}^*_{\boldsymbol{s}}, \boldsymbol{C}^*_{\boldsymbol{w}}, c^*, \boldsymbol{z}, \boldsymbol{r}_{\boldsymbol{z}}) \in \mathcal{T}$ such that $(\boldsymbol{z}, \boldsymbol{r}_{\boldsymbol{z}}) \neq (\boldsymbol{z}^*, \boldsymbol{r}^*_{\boldsymbol{z}})$, return $\perp$.
  - Else, return 1.
– Otherwise, return $\perp$.

The two games G_0 and G_1 behave identically until there is an adversary $\mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}$ that makes only G_1 return $\perp$. But then we can use that $\mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}$ to build an adversary that breaks weak unique response property in the following manner:

Adversary for Breaking Weak Unique Response:

– Run $\mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}$ and simulate its view.
– Maintain a list of simulated proofs based on queries made by $\mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}$ in $\mathcal{T}$.
– Randomly pick a query, say the k^{th} query, of $\mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}$ to $\mathcal{S}_2$ and use it to get a simulated transcript $(\mathbb{x}^k, \boldsymbol{S}^k, \boldsymbol{C}^k_{\boldsymbol{s}}, \boldsymbol{C}^k_{\boldsymbol{w}}, c^k, \boldsymbol{z}^k, r^k_{\boldsymbol{z}})$ from $\mathcal{S}_2$ on input $\mathbb{x}^k$. Use this simulated transcript to answer the k^{th} query.
– After the query phase, upon receiving a transcript as output, check if it verifies and differs from the k^{th} transcript in $\mathcal{T}$ only in the response. That is, if the outputs makes only G_1 abort. If yes, submit this output along with the k^{th} transcript as forgeries against weak unique-response.

Therefore, assuming weak unique response (Theorem 4), G_0 and G_1 are identical and we can hop from G_0 to G_1. G_1 being of interest to us since the Π_{GenLin} extractor $(\mathcal{E}_1, \mathcal{E}_2)$ can be invoked in G_1. Given an adversary $\mathcal{A}^{(\mathcal{S}_1,\mathcal{S}_2)}$ for simulation extractability, we can now construct an adversary $\mathcal{B}^{\mathcal{S}_1}$ for the knowledge soundness of Π_{GenLin}.

Construction of $\mathcal{B}^{\mathcal{S}_1}$. The knowledge soundness adversary $\mathcal{B}^{\mathcal{S}_1}$ of the non-interactive protocol uses $\mathcal{A}^{(\mathcal{S}_1,\mathcal{S}_2)}$ as a subroutine and perfectly simulates its view, as described in G_1.

Construction of the extractor for simulation extractability. We construct a straight-line extractor $\hat{\mathcal{E}}$ for the simulation extractable case. Let $(\mathcal{E}_1, \mathcal{E}_2)$ be the straight-line extractor for protocol Π_{GenLin}. $\hat{\mathcal{E}}$ on running its adversary $\mathcal{A}^{(\mathcal{S}_1,\mathcal{S}_2)}$ gets a instance-proof pair $(\mathbb{x}^*, \pi^* := (\boldsymbol{S}^*, \boldsymbol{C}^*_{\boldsymbol{s}}, \boldsymbol{C}^*_{\boldsymbol{w}}, c^*, \boldsymbol{z}^*, \boldsymbol{r}^*_{\boldsymbol{z}}))$. As noted above, this proof is also valid w.r.t a prover which only has access to $\mathcal{S}_1$. So $\hat{\mathcal{E}}$ invokes $(\mathcal{E}_1, \mathcal{E}_2)$ on $(\mathbb{x}^*, \pi^*)$ to get a witness $\boldsymbol{w}^*$.

Reduction. Given that $\mathcal{A}^{(\mathcal{S}_1,\mathcal{S}_2)}$ breaks simulation-extractability, we can use $\mathcal{B}^{\mathcal{S}_1}$ to break knowledge-soundness of the underlying NIZK protocol.

Adversary for Breaking Knowledge Soundness:

– $\mathcal{A}^{(\mathcal{S}_1,\mathcal{S}_2)}$ returns a proof $(\mathbb{x}^*, \pi^*)$ such that $\hat{\mathcal{E}}$ fails.
– $\mathcal{B}^{\mathcal{S}_1}$ which runs $\mathcal{A}^{(\mathcal{S}_1,\mathcal{S}_2)}$ as a subroutine, can simply forward this proof to $(\mathcal{E}_1, \mathcal{E}_2)$.
– Given how $\hat{\mathcal{E}}$ is constructed, the underlying extractor $(\mathcal{E}_1, \mathcal{E}_2)$ also fails.

Therefore, from the weak unique-response and knowledge soundness of Π_{GenLin}, we can argue that it also satisfies simulation extractability.

□

Now, we are ready to present the UC proof of Π_{GenLin}. We refer to Appendix. B for an overview of the UC security model. We prove UC-security of Π_{GenLin} by proving Theorem. 6.

Theorem 6. *Assuming Π_{GenLin} satisfies zero-knowledge and straightline-simulation extarctability then it UC-realizes $\mathcal{F}_{\mathsf{NIZK}}$ (Figure 3) for relation $\mathcal{R}_{\mathsf{GenLin}}$ in the random oracle model.*

Proof. First, we consider the case where the verifier is corrupt and then we consider the case where the prover is corrupt in a session sid. For both cases, we provide a simulator Sim.

Corrupt Verifier. The simulator Sim replaces the honest prover P in session sid and has to simulate the proof given the statement $\mathbb{x}$ without knowing the witness. This can be achieved by invoking the simulator $\mathcal{S}$ of the protocol. The detailed simulation algorithm is as follows:

Simulator against a Corrupt Verifier:

– **Input:** Statement $\mathbb{x}$ and NIZK simulators $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ from Proof Sketch of Theorem 3.
– Output $\mathcal{S}_2(\mathsf{sid}, \mathbb{x})$ to the corrupt verifier.

The real and ideal world executions are computationally indistinguishable by the zero-knowledge property of Π_{GenLin}.

Corrupt Prover. The simulator Sim simulates the honest verifier $\mathcal{V}$ in session sid. The prover is corrupt in session sid. Note that the adversarial prover has access to other simulated proofs from different sessions $\mathsf{sid}' \neq \mathsf{sid}$ where the prover is honest. Given such simulated proofs, the malicious prover generates a proof π^* in session sid for statement $\mathbb{x}^*$. The goal here is to be able to extract a witness from the proof π^*. This is done by invoking the simulation extractability extractor $\hat{\mathcal{E}}$. However, this extractor fails when $(\mathbb{x}^*, \pi^*)$ is a simulated proof in a different session sid' as $(\mathbb{x}^*, \pi^*) \in \mathcal{T}$ where $\mathcal{T}$ is list of simulated proofs obtained from different sessions. We show that the probability of this bad event occurring is negligible due to the random oracle assumption. The detailed simulation algorithm is as follows:

Simulator against a Corrupt Prover:

– **Input:** Adversarial Prover's response $(\mathsf{sid}, \mathbb{x}^*, \pi^*)$, Simulation Extractability Extractor $\hat{\mathcal{E}}$, List of simulated proofs $\mathcal{T}$ in different sessions.
– If there exists $(\mathbb{x}^*, \pi^*) \in \mathcal{T}$ for session sid' then abort.
– Otherwise, extract $\mathbb{w}^* = \hat{\mathcal{E}}(\mathbb{x}^*, \pi^*)$ and invoke $\mathcal{F}_{\mathsf{NIZK}}$ with $(\mathit{prove}, \mathsf{sid}, \mathbb{x}^*, \mathbb{w}^*)$ to complete simulation.

We provide the indistinguishability argument as follows:

- $\texttt{Hyb}_0$: Real world execution of the protocol Π_{GenLin}.
- $\texttt{Hyb}_1$: Same as $\texttt{Hyb}_0$, except the simulator aborts if there exists $(\mathbb{x}^*, \pi^*) \in \mathcal{T}$ for session sid'.
  An adversary distinguishes the two hybrids if $\mathcal{H}_{\mathcal{V}}(\mathsf{sid}, \mathbb{x}^*, a^*) = \mathcal{H}_{\mathcal{V}}(\mathsf{sid}', \mathbb{x}^*, a^*)$ where $\pi^* = (a^*, c^*, z^*)$. However, this occurs with negligible probability as $\mathsf{sid} \neq \mathsf{sid}'$ and random oracle $\mathcal{H}_{\mathcal{V}}$ is collision-resistant.
- $\texttt{Hyb}_2$: This is the simulation algorithm provided above, where the verification algorithm is replaced by the simulation extractability extractor.
  An adversary distinguishes between $\texttt{Hyb}_1$ and $\texttt{Hyb}_2$ if it breaks simulation-extractability, prevents the extractor from extracting a correct witness, and hampers simulating $\mathcal{F}_{\mathsf{NIZK}}$. However, such an adversary can be used to break simulation-extractability by forwarding $(\mathbb{x}^*, \pi^*)$ as the adversarial response to the simulation-extractability challenger. Moreover, it is ensured in hybrids $\texttt{Hyb}_1$ and $\texttt{Hyb}_2$ that $(\mathbb{x}^*, \pi^*) \notin \mathcal{T}$, resulting in $(\mathbb{x}^*, \pi^*)$ qualifying as a valid response.

This completes our UC proof of Π_{GenLin}. $\square$

D Basics of Class Groups

We borrowed the definitions and properties of class groups mostly from [CL15] and [Coh93]. An algebraic number field K is a quadratic number field if it is a degree two field extension of the field of rational number $\mathbb{Q}$. If K is a quadratic number field, then there exists a unique square free integer d such that $K = \mathbb{Q}(\sqrt{d})$. If $d < 0$, $K = \mathbb{Q}(\sqrt{d})$ is called imaginary quadratic field. Otherwise, it's called real quadratic field. Throughout the following, we assume that K is imaginary quadratic field.

The **fundamental discriminant** Δ_K of K is defined as $\Delta_K = 4d$ if $d \equiv 2, 3 \bmod 4$ and $\Delta_K = d$ if $d \equiv 1 \bmod 4$. Suppose $\mathcal{O}$ is a subring of K such that $\mathcal{O}$ contains 1 and $\mathcal{O}$ is a free $\mathbb{Z}$- module of rank 2. Then $\mathcal{O}$ is called an **order** of K.

$\alpha \in \mathbb{C}$ is called an algebraic integer if it is a root of a monic polynomial with integer coefficients. The algebraic integers in K form a ring called ring of integers, denoted as $\mathcal{O}_K$. This is the maximal order of K. The discriminant of $\mathcal{O}_K$ is Δ_K. Every order $\mathcal{O}$ of K has **finite index** $[\mathcal{O}_K : \mathcal{O}]$ in $\mathcal{O}_K$. we write $f := [\mathcal{O}_K : \mathcal{O}]$. Then the discriminant of order $\mathcal{O}$ is defined as $\Delta_f = f^2 \Delta_K$.

Let $\mathcal{O}_\Delta$ be an order of discriminant Δ. A **fractional ideal** in K is a set of the form $\frac{1}{d}\mathfrak{a} = \{\frac{a}{d} : a \in \mathfrak{a}\}$ where $d \in \mathbb{N}$ and $\mathfrak{a}$ is an integral ideal in $\mathcal{O}_\Delta$. A fractional ideal $\mathfrak{a} \subset K$ is called **invertible fractional ideal**, if there exists a unique fractional ideal $\mathfrak{b} \subset K$ of $\mathcal{O}_\Delta$ such that $\mathfrak{a}\mathfrak{b} = (1) = \mathcal{O}_\Delta$. The invertible fractional ideals of $\mathcal{O}_\Delta$ form a multiplicative group I_Δ with identity element $\mathcal{O}_\Delta = (1)$. The fractional ideals of the form $(\alpha) = \alpha\mathcal{O}_\Delta = \{\alpha a : a \in \mathcal{O}_\Delta\}$ for some $\alpha \in K$ are called **principal fractional ideals**. They form a subgroup P_Δ of I_Δ. The **ideal class group** of the order $\mathcal{O}_\Delta$ is the quotient $C_\Delta = I_\Delta / P_\Delta$. The ideal class group is finite. The cardinality of the class group is called **class number**, which is denoted as h_Δ.

A non zero ideal $\mathfrak{a}$ of $\mathcal{O}_\Delta$ is **prime** to f if $\mathfrak{a} + f\mathcal{O}_\Delta = \mathcal{O}_\Delta$. The ideals prime to f form a subgroup in I_Δ. We denote this subgroup as $I_{\Delta,f}$.

D.1 Representation of classes using quadratic forms

Every integral ideal $\mathfrak{a}$ of $\mathcal{O}_\Delta$ can be represented as $\mathfrak{a} = a\mathbb{Z} + \frac{-b+\sqrt{\Delta}}{2}\mathbb{Z}$ with $a \in \mathbb{N}$ and $b \in \mathbb{Z}$ with $b^2 \equiv \Delta \bmod 4a$ and denoted by (a, b).

A **binary quadratic form** is a quadratic homogeneous polynomial in two variables $f(x, y) = ax^2 + bxy + cy^2$. The ideal $\mathfrak{a}$ corresponds to a binary quadratic form $f(x, y) = ax^2 + bxy + cy^2$ such that $b^2 - 4ac = \Delta$. Δ is called the discriminant of $f(x, y)$. A binary quadratic form $f(x, y) = ax^2 + bxy + cy^2$ with discriminant Δ can be transformed into another binary quadratic form $g(x, y) = a'x^2 + b'xy + c'y^2$ with same discriminant by a linear transformation $U = \begin{pmatrix} s & t \\ u & v \end{pmatrix} \in SL_2(\mathbb{Z})$. These two binary quadratic forms f and g are congruent modulo $SL_2(\mathbb{Z})$. We say that they are equivalent quadratic forms and write $f \sim g$. This defines an equivalence relation on the set of all quadratic forms of discriminant Δ, which partitions this set into equivalence classes.

It turns out that these equivalence classes are in one-to-one correspondence to the ideal classes of $\mathcal{O}_\Delta$(Section 5.2 of [Coh93]).

The above representation of ideal $\mathfrak{a}$ is not unique. If $\mathfrak{a} = (a, b)$, $\mathfrak{a}$ can also be represented as $\mathfrak{a} = (a\mathbb{Z} + (ma + \frac{-b+\sqrt{\Delta}}{2})\mathbb{Z}) = (a\mathbb{Z} + \frac{-(b-2ma)+\sqrt{\Delta}}{2}) = (a, b - 2ma)$ for $m \in \mathbb{Z}$. We note that, there is a unique integer m such that, $-a < b - 2ma \leq a$ holds. We say that (a, b) is in **normal** form if $-a < b \leq a$. An ideal (a, b) is in **reduced** form if it is in normal form and $a \leq c$ and $b \geq 0$ for $a = c$. In each class $[\mathfrak{a}]$ of ideals in $\mathcal{O}_\Delta$, there exists a unique reduced ideal denoted as $Red(\mathfrak{a})$. Due to an algorithm by Gauss, which is described in (Algorithm 5.4.7, [Coh93]), we can efficiently compute $Red(\mathfrak{a})$ from $\mathfrak{a}$.

Lemma 1. *(Lemma 5.3.4, [Coh93]) If* $a < \frac{\sqrt{|\Delta|}}{2}$ *and* $-a < b \leq a$ *then* (a, b) *is reduced.*

In general, instead of a class, we will work with the reduced ideal (a, b) in that class. This amounts to the same as working with the binary quadratic form $f(x, y) = ax^2 + bxy + cy^2$ which corresponds to that particular reduced class. We will denote this quadratic form as (a, b, c).

D.2 Hashing into Class Groups

The security of many cryptographic primitives, like verifiable delay functions [BBF19, Wes19], polynomial commitment schemes for zkSnarks [BFS20] rely on groups of unknown order. Moreover, many applications require a trustless setup where the order must remain unknown even to the generation algorithm. A good candidate for such applications is an ideal class group. The class group generation algorithm only requires carefully choosing a public parameter called the discriminant and remains oblivious of the order of the class group. The order of the class group, also known as the class number, is hard to compute for a large enough discriminant. Our straightline-extractable Sigma protocol uses an additively homomorphic encryption scheme based on class groups [CL15]. The usual Sigma protocol doesn't have any trust assumptions. To be able to carry this desirable trait on to its straightline version, the public key of the AHE has to be sampled obliviously. That is, the potentially malicious setup algorithm should remain oblivious of the corresponding secret key which is DLog(pk). One way to achieve that is to construct a hash function that maps a random string to a class group element. Wesolowski proposed one such hash construction in [Wes19] and invited improvements to it. Recents works by Seres et al [SBK24] and Chalkias et al [CLR24] propose improved constructions of Wesolowski's hash. The construction by Chalkias, Lindstrom and Roy is 500 times faster than the original for a 3072 bit discriminant.

Overview of Wesolowski [Wes19] and Chalkias et al. [CLR24] construction Wesolowski's original construction of hashing into a reduced binary quadratic form(each reduced binary quadratic form represents an element in the class group) in [Wes19], first samples an odd prime a uniformly at random from the range $\{0, 1, \ldots, \frac{\sqrt{|\Delta|}}{2}\}$ such that Δ is quadratic residue mod a. Therefore, we can find an integer b such that $b^2 \equiv \Delta$ mod a. If b is odd, $b^2 \equiv 1$ mod 4, which gives $b^2 \equiv \Delta$ mod $4a$, therefore, $(b^2 - \Delta)/4a$ is an integer. We denote it as c and return the quadratic form (a, b, c) with discriminant Δ. If b is even, consider the other square root $a - b$ in place of b and note that $a - b$ is odd. Next we again compute $c = (b^2 - \Delta)/4a$ and return the quadratic form $(a, -b, c)$ with same discriminant Δ. Due to lemma 1, the specific range of a gurantees that the returned binary quadratic forms are reduced.

If Δ is large, Wesolowski's contruction is not efficient for expensive primality checking in the range $\{0, 1, \ldots, \frac{\sqrt{|\Delta|}}{2}\}$ and computation of modular square-root function modulo the large prime which is sampled. Chalkias et al. [CLR24] improved upon that construction by proposing a hash function $\mathcal{H}_{\Delta,k}$, which first samples k many individual primes $a_1, a_2, \ldots, a_k$ for some integer k from the range $\{0, 1, \ldots, (\frac{\sqrt{|\Delta|}}{2})^{1/k}\}$ such that Δ is a quadratic residue modulo $a_i's$. Then construct a as the product of those k many $a_i's$. As, Δ is a quadratic residue modulo $a_i's$, there are integers $b_i's$ such that $b_i^2 \equiv \Delta$ mod a_i. We can construct b such that b

is in the range $\{0,1,\ldots,a-1\}$ with $b \equiv b_i \bmod a_i$ for all i using the Chinese Remainder Theorem. If b is odd, compute $c = \frac{b^2-\Delta}{4a}$, else consider $a-b$ as b and compute c as above. Finally, $\mathcal{H}_{\Delta,k}$ returns binary quadratic forms (a,b,c) or $(a,-b,c)$. As a is product of k many primes in the range $\{0,1,\ldots,(\frac{\sqrt{|\Delta|}}{2})^{1/k}\}$, the returned binary quadratic forms are reduced due to lemma 1. Each reduced binary quadratic forms represent a class in the group C_Δ.

Theorem 7. *[CLR24]. The hash function $\mathcal{H}_{\Delta,k} : \{0,1\}^* \rightarrow \{(a,b,c) \in C_\Delta : a = \prod_{i=1}^{k} a_i, 2 < a_i < (\sqrt{|\Delta|}/2)^{1/k}, -a < b \leq a\}$ is a random oracle. i.e, it samples a uniform random output from it's range and return it for any input.*

In trustless setup, The AHE scheme with a class group based PKE scheme requires sampling the public key from a cyclic group of unknown order. We give an concrete mathematical instantiation of the class groups mentioned in section 6 by following [CCL+19]:

- For a prime q, pick another random prime $\tilde{q}$ such that $q \cdot \tilde{q} \equiv -1 \bmod 4$ and q is a quadratic non residue modulo $\tilde{q}$.
- Compute a fundamental disciminant $\Delta_K := -q \cdot \tilde{q}$. Then, compute the discriminant $\Delta_q = q^2 \Delta_K$ of the order $\mathcal{O}_{\Delta_q}$.
- Set the class group $\widehat{\mathbb{G}}_{\mathsf{CL}} := C_{\Delta_q}$.
- Set $f := [(q^2, q)]$ which is a class in C_{Δ_q} of order q. Consider, $\mathbb{F}_{\mathsf{CL}} = \langle f \rangle$ is a cyclic subgroup of $\widehat{\mathbb{G}}_{\mathsf{CL}}$.
- Set $\widehat{\mathbb{G}}^q_{\mathsf{CL}} := C_{\Delta_K}$. where, $\widehat{\mathbb{G}}^q_{\mathsf{CL}} = \{g^q : g \in \widehat{\mathbb{G}}_{\mathsf{CL}}\}$. The cardinality of the class group C_{Δ_K} is h_{Δ_K} which is unknown. We know that h_{Δ_K}has an upper bound $\frac{1}{\pi} \log |\Delta_K| \sqrt{|\Delta_K|}$([Coh93]). We set h_{Δ_K} as $\widehat{s}$.
- Set the upper bound $U := \lceil \frac{1}{\pi} log|\Delta_K|\sqrt{|\Delta_K|} \rceil$.
- The cardinality of C_{Δ_q} is $h_{\Delta_q} = q \cdot h_{\Delta_K}$.
- [CLR24] describes a hash function $\mathcal{H}_{\Delta,k}$ which samples a reduced ideal $\mathfrak{a} = (a,b) \in \mathcal{O}_K$ by sampling a in the range $\{0,1,\cdots,\sqrt{\Delta_K}/2\}$ and computing b accordingly using Chinese Remainder Theorem. The reduced ideal $\mathfrak{a}$ represents a class $[\mathfrak{a}] \in C_{\Delta_K}$.
- The order of class $[\mathfrak{a}]$ divides the unknown cardinality of C_{Δ_K} which is h_{Δ_K}. This order of $[\mathfrak{a}]$ is also unknown and we set it as s. Now we set $g_q := [\mathfrak{a}]$ and set the cyclic subgroup $\mathbb{G}^q_{\mathsf{CL}} = \{g_q^x : x \xleftarrow{\$} \mathcal{D}_q\}$.
- Set $g = g_q \cdot f$ and $\mathbb{G}_{\mathsf{CL}} = \langle g \rangle$ is a cyclic subgroup of $\widehat{\mathbb{G}}_{\mathsf{CL}}$.
- Set $\mathcal{D}$ and $\mathcal{D}_q$ are the distributions over the set of integers such that they are statistically close to the uniform distribution over the interval $\{0,1,\cdots,qU-1\}$ and $\{0,1,\cdots,U-1\}$ respectively.
- The public key used in the AHE scheme is sampled randomly from the group $\mathbb{G}^q_{\mathsf{CL}}$.

E OR Composition using our Compiler

In this section, we show that our compiler can be used to obtain a UC-NIZK for OR composition in Fig. 5. ZK and soundness follows from the standard OR composition and the security of the compiler. To extract a witness, the extractor decrypts the encryptions from both a and a'. Only one of them will yield a valid witness in the random oracle model.

Ingredients and Settings:

– **Input:** Both prover and verifier know the public instance $\mathbb{x}$ and $\mathbb{x}'$, and the prover exclusively has witness $\mathbb{w}$ for $\mathbb{x}$.

– **Primitives:** The interactive Sigma protocol from Fig. 2 $(\mathsf{Setup}, \mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2), \mathcal{V} = (\mathcal{V}_1, \mathcal{V}_2))$ based on an AHE scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with oblivious sampleability enabled by a hash function $\mathcal{H}_{\mathsf{ek}} : \{0,1\}^* \to \mathcal{K}_{\mathsf{ek}}$. Another hash function, $\mathcal{H}_{\mathcal{V}} : \{0,1\}^* \to \mathbb{Z}_q$. Both together are modeled as random oracle $\mathrm{RO} = (\mathcal{H}_{\mathsf{ek}}, \mathcal{H}_{\mathcal{V}})$. The prover has access to the HVZK simulator $\mathcal{S}_{\Sigma}$.

Protocol Description:

– $\mathcal{P}^{\mathrm{RO}}(\mathit{prove}, \mathsf{sid}, \mathbb{x}, \mathbb{x}', \mathbb{w}) \to \pi$.
  - Parse RO as $(\mathcal{H}_{\mathsf{ek}}, \mathcal{H}_{\mathcal{V}})$.
  - Compute $\mathsf{ek} := \mathcal{H}_{\mathsf{ek}}(\mathsf{sid}, \mathbb{x}, \mathbb{x}')$ and set $\mathsf{crs} := \mathsf{ek}$.
  - Sample $c' \leftarrow \mathbb{Z}_q$ and obtain $(a', c', z') = \mathcal{S}_{\Sigma}(\mathsf{ek}, \mathbb{x}', c')$.
  - Run $a := \mathcal{P}_1(\mathsf{crs}, \mathbb{x}, \mathbb{w}; \rho)$.
  - Compute $\widetilde{c} := \mathcal{H}_{\mathcal{V}}(\mathsf{sid}, \mathbb{x}, \mathbb{x}', a, a')$.
  - Run $z := \mathcal{P}_2(\mathsf{crs}, \mathbb{x}, \mathbb{w}, a, \widetilde{c} - c', \rho)$.
  - Set $c = \widetilde{c} - c'$.
  - Output $\pi := (a, c, z, a', c', z')$.

– $\mathcal{V}^{\mathrm{RO}}(\mathsf{sid}, \mathbb{x}, \mathbb{x}', \pi) \to 1/0$.
  - Parse $(a, c, z, a', c', z') := \pi$.
  - Parse RO as $(\mathcal{H}_{\mathsf{ek}}, \mathcal{H}_{\mathcal{V}})$.
  - Compute $\mathsf{ek} := \mathcal{H}_{\mathsf{ek}}(\mathsf{sid}, \mathbb{x}, \mathbb{x}')$.
  - Compute $\widetilde{c} := \mathcal{H}_{\mathcal{V}}(\mathsf{sid}, \mathbb{x}, \mathbb{x}', a, a')$.
  - Output $\mathcal{V}_2(\mathsf{crs}, \mathbb{x}, a, c, z) \wedge \mathcal{V}_2(\mathsf{crs}, \mathbb{x}', a', c', z') \wedge (\widetilde{c} = c + c')$.

Fig. 5: Our UC-NIZK protocol for OR composition over $\mathcal{R}_{\mathsf{GenLin}}$.