An Algebraic Framework for Silent Preprocessing with Trustless Setup and Active Security

Damiano Abram, Ivan Damgård, Claudio Orlandi, and Peter Scholl

Aarhus University

Abstract. Recently, number-theoretic assumptions including DDH, DCR and QR have been used to build powerful tools for secure computation, in the form of *homomorphic secret-sharing* (HSS), which leads to secure two-party computation protocols with succinct communication, and *pseudorandom correlation functions* (PCFs), which allow non-interactive generation of a large quantity of correlated randomness. In this work, we present a group-theoretic framework for these classes of constructions, which unifies their approach to computing distributed discrete logarithms in various groups. We cast existing constructions in our framework, and also present new constructions, including one based on class groups of imaginary quadratic fields. This leads to the first construction of two-party homomorphic secret sharing for branching programs from class group assumptions.

Using our framework, we also obtain pseudorandom correlation functions for generating oblivious transfer and vector-OLE correlations from number-theoretic assumptions. These have a *trustless, public-key setup* when instantiating our framework using class groups. Previously, such constructions either needed a trusted setup in the form of an RSA modulus with unknown factorisation, or relied on multi-key fully homomorphic encryption from the learning with errors assumption.

We also show how to upgrade our constructions to achieve active security using appropriate zero-knowledge proofs. In the random oracle model, this leads to a one-round, actively secure protocol for setting up the PCF, as well as a 3-round, actively secure HSS-based protocol for secure two-party computation of branching programs with succinct communication.

Table of Contents

An Algebraic Framework for Silent Preprocessing with Trustless Setup and Active Security . . . 1
Damiano Abram, Ivan Damgård, Claudio Orlandi, and Peter Scholl
1 Introduction . . . 3
1.1 Our Contributions . . . 4
1.2 An Overview of the Framework . . . 6
2 Notation and Preliminaries . . . 8
2.1 Homomorphic Secret-Sharing . . . 8
2.2 Pseudorandom Correlation Functions . . . 10
3 A Group-Theoretic Framework . . . 12
3.1 Assumptions . . . 14
4 Instantiations of the Framework . . . 15
4.1 Paillier and Damgård-Jurik . . . 15
4.2 Joye-Libert Variants . . . 16
4.3 Class Groups . . . 18
5 HSS Constructions . . . 19
5.1 NIDLS ElGamal . . . 19
5.2 Public-Key HSS . . . 21
5.3 Implementing the Setup Using One Round. . . . 25
6 Public-Key PCFs and One-Round VOLE Protocol without Trusted Setup . . . 26
6.1 Public-Key PCFs without Trusted Setup. . . . 29
7 Actively Secure Public-Key PCFs . . . 31
7.1 An Integer Commitment Scheme in the NIDLS Framework . . . 32
7.2 Zero-Knowledge Proofs in the NIDLS Framework . . . 34
7.3 Actively Secure Public-Key PCF for Vector-OLE . . . 35
7.4 Actively secure public-key PCF for oblivious transfer . . . 44
8 Active HSS Protocol . . . 51
A Sigma protocols in the NIDLS framework . . . 62
A.1 Proof of knowledge of opening . . . 62
A.2 Multiplication proof . . . 64
A.3 Range proof . . . 66
A.4 Proof of commitment to the plaintext . . . 68
A.5 Proof of well-formed inputs . . . 70
A.6 Proof of well-formed MAC keys . . . 73

1 Introduction

Homomorphic secret sharing (HSS) [BGI16] can be seen as a relaxed form of fully-homomorphic encryption (FHE), where two non-colluding servers evaluate a function on private inputs without interaction. At the end of the computation, the servers each obtain a secret share, and these can be combined to obtain the result. At the core of existing HSS constructions is a procedure for *distributed discrete log*, where two parties are given group elements g_0, g_1 such that $g_1 = g_0 \cdot g^x$ for some fixed base g, and want to convert these multiplicative shares into additive shares x_0, x_1, where $x_1 = x_0 + x$ over the integers. The method from [BGI16], which is based on the decisional Diffie-Hellman (DDH) assumption, allows doing this conversion without interaction, however, there is an inherent correctness error. This results in significant extra work to ensure that the magnitude of the error is small. Moreover, the error cannot be made negligible. This limitation carries over to the final HSS construction, which has a non-negligible probability that the result of the computation is incorrect.

Recently, it was shown that the non-negligible correctness error of the DDH construction can be overcome, when switching to the Paillier [Pai99] or Damgård-Jurik [DJ01] cryptosystems based on the decisional composite residuosity (DCR) assumption. With these encryption schemes, which work over $\mathbb{Z}_{N^2}^*$ for an RSA modulus N, discrete logarithms can be computed in a distributed manner with a very simple and perfectly correct algorithm [OSY21, RS21]. This avoids the challenges of the DDH setting, by exploiting the fact that the messages in these schemes lie in a subgroup where solving discrete log is easy.

In [OSY21], the same distributed discrete log technique was used for several other applications in secure computation. In particular, they constructed *pseudorandom correlation functions* based on the Paillier and quadratic residuosity assumptions. A pseudorandom correlation function (PCF) is a way of generating two short, correlated keys, such that when evaluating the function on each of the keys, the two outputs are correlated in some secret manner. This generalizes the notion of a pseudorandom correlation generator [BCG+19], which only supports a bounded number of outputs. Examples of useful correlations for PCFs and PCGs are random oblivious transfer correlations, or secret-shared multiplication triples, which can be used in GMW-style multi-party computation protocols [GMW87] with very lightweight online computation.

An appealing feature of the PCFs from [OSY21] is that the PCF keys can be generated in a *public-key* manner, where after publishing just a single, short message, each party can locally derive their PCF key and compute the correlated randomness. However, a major drawback is that to achieve this public-key setup, the parties first need to have a trusted setup in the form of a public RSA modulus with unknown factorisation.

1.1 Our Contributions

It may seem from the previous work in [OSY21, RS21] that their efficient approach to distributed discrete log depends on very specific properties of Paillier, or more generally Damgård-Jurik encryption.

However, we show that this is not the case: in Section 3 we present a general framework, where we demonstrate that the approach from previous works can be phrased in terms of abstract group-theoretic properties. Naturally, the known methods based on Paillier and Damgård-Jurik become special cases of our framework, but we also show instantiations under different assumptions in Section 4.

Below, we describe the main applications of our framework to secure two-party computation, and the results obtained from our new instantiations.

Homomorphic secret sharing. We show in Section 5 that any instantiation of our framework that supports superpolynomially large plaintexts can be used to build homomorphic secret sharing for the class of polynomial size branching programs. This construction follows the same blueprint as previous works that obtain HSS for branching programs [BGI16, BKS19, OSY21, RS21]. Using this, two new instantiations of our framework imply two new constructions of HSS based on a flavour of the decisional Diffie-Hellman assumption for short exponents.

Firstly, we obtain HSS from a variant of the Joye-Libert cryptosystem [JL13, BHJL17], modified to work over a modulus that is a product of many small, distinct primes; compared with the analogous constructions based on Paillier, this has the advantage that ciphertexts are only a single element of $\mathbb{Z}_N$, and we can be more flexible in our choice of plaintext space, which is limited to $\mathbb{Z}_{N^s}$ otherwise. For a plaintext space modulo Q, we need to choose N such that $p-1, q-1$ are divisible by Q, so when Q is large we should clearly increase p, q to compensate, however, for reasonable sizes of Q the resulting ciphertext size should still be smaller than Paillier, which is an element of $\mathbb{Z}_{N^2}$.

Secondly, we obtain HSS from the DDH assumption in class groups of imaginary quadratic fields, based on the CL cryptosystem [CL15]. Class groups have recently seen many cryptographic applications, since they offer a way to generate a group of unknown order, without relying on any trusted setup to create the group parameters. Using class groups in HSS, we avoid the need for a setup with an RSA modulus where no party knows the factorization, instead only relying on a CRS that can be sampled with public randomness. For security, we rely on the DDH assumption with short exponents, where the short exponents are used to ensure that the secret key fits in the message space of the scheme, which allows us to easily encrypt functions of the secret key without introducing a circular security assumption.

Public-key pseudorandom correlation functions with trustless setup. Our starting point here is the PCFs from Paillier and quadratic residuosity from [OSY21], which give PCFs for generating vector-OLE and OT correlations, respectively.

PCFs, by definition, involve a setup procedure where a trusted dealer distributes a pair of short keys to the two parties. In [OSY21], it was shown that given a 1-round protocol for vector-OLE, where each party sends one parallel message, the PCF setup procedure can be replaced with a simple *public-key setup*, where each party publishes one message, which is then used to derive a PCF key. To realize the 1-round vector-OLE protocol, they give a dedicated construction based on distributed discrete log from Paillier, however, this still relies on a trusted setup in the form of an RSA modulus with unknown factorisation. We show in Section 6 that this construction can be generalised to work under any instantiation of our framework; with our class groups instantiation, we then obtain vector-OLE with a trustless setup. Put together with the PCFs from [OSY21], this leads to a public-key PCF with trustless setup for vector-OLE based on the combination of class group assumptions and DCR, or one for OT by combining class groups and quadratic residuosity.

Active security. Given a public-key PCF, where after exchanging public keys, two parties can compute as much correlated randomness as they need, it is natural to ask, can this type of protocol be made actively secure? Although there are many ways of generically compiling passively secure protocols into active ones [GMW86, IPS08], we want to achieve something reasonably practical, in particular, to avoid using generic zero-knowledge techniques that require expressing group operations as circuits or similar. We show in Section 7 how to upgrade our PCFs to achieve active security, while preserving their public-nature by using Fiat-Shamir based NIZKs in the random oracle model. We do this via a careful combination of sigma protocols, which all make black-box use of the group, so avoid the complications of generic techniques. One challenge is that to build the public-key PCF, we need one party to prove that their input to the vector-OLE protocol corresponds to a secret key for an RSA modulus used in the PCF. As an essential tool, we use an integer commitment scheme which we show can be built from class groups and a trustless set-up. Thus, even our actively secure PCF does not need a trusted dealer. See the next section for details on the assumption required for this.

Finally, we also show how to add active security to our HSS construction in Section 8. In the random oracle model, this gives a 3-round protocol for actively secure two-party computation of branching programs, which makes black-box use of the operations needed by our group-theoretic framework. Here, as well as proving that ciphertexts used to the secret-share HSS inputs are well-formed, we also need range proofs to ensure that the inputs are bounded in size.

A comparison to [OSY21] and [RS21]. As summarised above, previous work focuses its analysis on Paillier and Goldwasser-Micali [OSY21], and Damgård-Jurik [RS21]. Both [OSY21] and [RS21] describe how to solve distributed discrete log in the setting they study and use the techniques to build HSS for branching programs. In [OSY21], the authors also explain how to build public-key PCFs for OT and VOLE using the distributed discrete log techniques. All the construc-

tions presented in [OSY21] and [RS21] rely on a trusted setup for the generation of a public RSA modulo of unknown factorisation.

The main contribution of this work is to generalise the techniques of [OSY21] and [RS21] to an abstract algebraic framework. We characterise the assumptions that the framework needs to satisfy to solve distributed discrete log, build HSS for branching programs and public-key PCFs for OT and VOLE. We present also new instantiations of the framework in addition to Paillier, Goldwasser-Micali and Damgård-Jurik, namely variants of the Joye-Libert cryptosystem and class groups. The latter allows us to build HSS and public-key PCFs that do not need trusted setups. Finally, while [OSY21] and [RS21] limit their study to passive security only, this work explains how to upgrade the constructions to active security obtaining implementable solutions that make black-box use of the underlying group.

1.2 An Overview of the Framework

In a nutshell, our framework consists of a large, finite group G, where $G = F \times H$. In the subgroup F, which is cyclic with generator f, discrete log is easy, and the order of F is public (whereas this is not the case for H). In the distributed discrete logarithm problem, two parties are given group elements $g_0, g_1 \in G$, with the condition that $g_0/g_1 = f^m$ for some message m. The goal is for the parties to convert this into shares m_0, m_1, where $m_0 + m_1 = m$ modulo the order of F. The crucial ingredient we need for distributed discrete log is a function we call a *coset labelling function*, which, for each coset C of F in G, maps all elements in C to a specific element in C. Existence of a coset labelling function turns out to be enough to solve distributed discrete log assuming that the two parties start from elements in the same coset, and it further turns out that this is sufficient to implement all our constructions, as long as some appropriate computational assumptions hold in G.

Instantiations. This framework easily encompasses previous constructions where distributed discrete logs are computed with Paillier, Damgård-Jurik, or Goldwasser-Micali ciphertexts. We also show that a natural variant of the Joye-Libert cryptosystem can be used (although it remains open to find a coset labelling function for the original Joye-Libert scheme, with plaintexts modulo 2^k). Finally, we give an instantiation based on class groups over imaginary quadratic fields. Here, we essentially apply the framework of the CL cryptosystem [CL15] for linearly homomorphic encryption, and combine it with the observation that the coset labelling function can be obtained via a special surjective map, which was previously used in the NICE cryptosystem [PT00] and its cryptanalysis [CJLN09].

Trustless Setup and the DXDH Assumption. For all applications of our framework, we rely on the standard DDH assumption in the group G. In settings where we need a trustless setup, we sometimes use a new assumption we call the *decisional cross-group Diffie-Hellman*, or DXDH, assumption. This states that

for group elements $g, h \leftarrow G$ sampled with random coins ρ_g, ρ_h, and random exponents r, s,

$$(\rho_g, \rho_h, g, h, g^r, h^r) \cong (\rho_g, \rho_h, g, h, g^r, g^s)$$

This assumption arises in settings where we have a CRS with two group elements g, h, and we want the CRS to be public-coin. Having a public-coin CRS implies a trustless setup, since in practice the parties can derive randomness using e.g. a random oracle, and use this to sample the group elements. Note that in a standard cyclic DDH group (such as with elliptic curves), DXDH and DDH are equivalent because g, h always generate the same group, and furthermore, given a group element g it is easy to find some random coins that 'explain' it. With class groups, however, this is not the case, since we are not aware of any invertible sampling algorithm, nor any method for sampling g and h such that they lie in the same subgroup.

Thus, when aiming for a trustless setup, we need DXDH. An additional complication of this setting is that the assumption makes it harder to use a CRS in security proofs: there is no way to introduce a trapdoor in the CRS by picking $h = g^t$ in the simulation, as we do not know how to explain the random coins used to sample h (without leaking t).

We note that recently, [CKLR21] presented zero-knowledge proofs built using integer commitments from class groups, which require a CRS (g, h) and the assumption that (g, h) is indistinguishable from (g, g^s). Note that this assumption is incompatible with a trustless setup: if the CRS contains the random coins used to sample g and h, then the assumption doesn't hold as it is hard for the simulator to come up with the random coins needed to explain sampling $h = g^s$.[1] However, in Section 7 we show that the same commitment scheme *does* permit a trustless setup under the DXDH assumption, and we use this in our zero-knowledge proofs to obtain active security.

Recap of the framework. We now summarise the description of our framework. Our setting is an finite, Abelian group

$$G \cong F \times H \qquad \text{where} \qquad F = \langle f \rangle.$$

The group G needs to satisfy these properties:

1. The discrete log function over F is efficiently computable.
2. There exists an efficiently computable coset labelling function π.
3. There exists an efficiently computable function δ (the lifting function) such that $\pi\big(\delta(x)\big) = x$ for every input x.

In order to build HSS for branching programs and public-key PCFs for OT and VOLE, the group needs to satisfy additional computational assumption which are summarised in Table 1.

[1] The authors of [CKLR21] have acknowledged. They claim to have found a solution and are going to update their work.

Construction	*Assumptions and Model*
HSS for branching programs	DDH, small exponent (DXDH, weak hidden order + RO)
Public-Key PCF for VOLE	DCR, DXDH, DDH (weak hidden order, QR) + RO
Public-Key PCF for OT	QR, DXDH, DDH (weak hidden order) + RO

Table 1. Computational assumptions needed by our constructions. Elements written in between brackets are needed only for active security.

2 Notation and Preliminaries

Let λ denote the security parameter. Our constructions are restricted to the two-party setting and we denote them P_0 and P_1. For any $a, b \in \mathbb{Z}$ with $a < b$, we represent the set of integers $\{a, a+1, \ldots, b\}$ by $[a, b]$. We use $[b]$ to represent $[0, b-1]$. We assume that by reducing an element modulo $t \in \mathbb{N}$, we obtain a value in $[t]$.

Given a deterministic algorithm Alg, we denote its evaluation on an input x and the assignment of the result to a variable y by $y \leftarrow \mathsf{Alg}(x)$. If Alg is instead probabilistic, we write $y \xleftarrow{R} \mathsf{Alg}(x)$. The operation assumes that the random bits used by the algorithm are sampled uniformly. When we want to use a specific random string r, we write instead $y \leftarrow \mathsf{Alg}(x; r)$. Finally, if the element y is uniformly sampled from a set $\mathcal{X}$, we write $y \xleftarrow{R} \mathcal{X}$.

We denote vectorial elements using the bold font, the i-th entry of a vector $\boldsymbol{v}$ is denoted by v_i or by $\boldsymbol{v}[i]$. The cyclic subgroup generated by a group element g is represented by $\langle g \rangle$. Finally, we denote secret-shared elements y using the y-in-a-box notation, i.e. $[y]$. It will be clear from the context if that denotes a secret-sharing or the set $\{0, 1, \ldots, y-1\}$.

2.1 Homomorphic Secret-Sharing

A homomorphic secret-sharing scheme (HSS) [BGI16, BGI17, BKS19] is a construction that allows a set of parties to *non-interactively* apply functions on secret-shared data obtaining secret-shared outputs. The set of supported functions is usually restricted to a class $\mathcal{P}$. Furthermore, the secret-sharing schemes used for inputs and outputs might differ and not be linear. To some extent, HSS can be considered a distributed version of homomorphic encryption, in which, instead of preserving the privacy of the inputs through encryption, we do it by means of secret-sharing schemes. We formalise the definition below.

Definition 1 (Homomorphic Secret-Sharing). *A 2-party, (public key) homomorphic secret-sharing (HSS) scheme for a class of circuits $\mathcal{P}$ over a ring R with input space $\mathcal{I} \subseteq R$ is a triple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Input}, \mathsf{Eval})$ *with the following syntax.*

- $\mathsf{Setup}(1\!\!1^\lambda) \xrightarrow{R} \big(\mathsf{pk}, (\mathsf{ek}_0, \mathsf{ek}_1)\big)$*: On input the security parameter $1\!\!1^\lambda$, the procedure outputs an HSS public key* pk *and a pair of evaluation keys* $(\mathsf{ek}_0, \mathsf{ek}_1)$*, one for each party.*

– $\mathsf{Input}(\mathsf{pk}, x) \xrightarrow{R} (\mathsf{I}_0, \mathsf{I}_1)$*: On input the HSS public key* pk *and a value* $x \in \mathcal{I}$*, the procedure outputs the input information* $(\mathsf{I}_0, \mathsf{I}_1)$*, where* I_i *is addressed to party* P_i*. When* $\mathsf{I}_0 = \mathsf{I}_1 =: \mathsf{I}$*, we simply write* $\mathsf{Input}(\mathsf{pk}, x) \xrightarrow{R} \mathsf{I}$*.*
– $\mathsf{Eval}\big(i, \mathsf{ek}_i, (\mathsf{I}_i^1, \mathsf{I}_i^2, \ldots, \mathsf{I}_i^n), P\big) \to z_i$*: On input an index* $i \in \{0, 1\}$*, the* i*-th evaluation key* ek_i*,* P_i*'s information regarding* n *input values and an* n*-input program* $P \in \mathcal{P}$*, the procedure outputs a value* $z_i \in R$ *corresponding to* P_i*'s additive share of the output.*

Essentially, the setup algorithm is used to generate and distribute the key material necessary to use the construction. Using the HSS public-key pk, the parties are able to encode their inputs to the computation without leaking any information about the values. Finally, by applying the evaluation algorithm on the encoded inputs, the parties are able to obtain an additive secret-sharing of the result. It is possible to reconstruct the output after exchanging the corresponding shares.

We say that the HSS scheme is correct if, by evaluating any program $P \in \mathcal{P}$ on encodings of $x_1, \ldots, x_n \in \mathcal{I}$, the parties obtain an additive secret-sharing of $P(x_1, \ldots, x_n)$. The idea is formally defined below.

Definition 2 (Correctness of HSS Schemes). *Let* $(\mathsf{Setup}, \mathsf{Input}, \mathsf{Eval})$ *be a 2-party public key HSS scheme for the circuit class* $\mathcal{P}$ *over the ring* R*. Let* $\mathcal{I} \subseteq R$ *be the input space. We say that the scheme is correct if, for every* n*-input program* $P \in \mathcal{P}$ *and inputs* $x_1, x_2, \ldots, x_n \in \mathcal{I}$*, the following probability is negligible in the security parameter* λ*.*

$$\mathbb{P}\left[z_0 + z_1 \neq P(x_1, x_2, \ldots, x_n) \;\middle|\; \begin{array}{l} \big(\mathsf{pk}, (\mathsf{ek}_0, \mathsf{ek}_1)\big) \xleftarrow{R} \mathsf{Setup}(1\!\!1^\lambda) \\ \forall j \in [n]: \quad (\mathsf{I}_0^j, \mathsf{I}_1^j) \xleftarrow{R} \mathsf{Input}(\mathsf{pk}, x_j) \\ \forall i \in \{0,1\}: \quad z_i \leftarrow \mathsf{Eval}\big(i, \mathsf{ek}_i, (\mathsf{I}_i^1, \mathsf{I}_i^2, \ldots, \mathsf{I}_i^n), P\big) \end{array} \right]$$

In order for the HSS scheme to be secure, the input encodings should leak no information about their underlying value, even if one of the parties is corrupt. We formally model this by saying that no PPT adversary $\mathcal{A}$ can distinguish an encoding of x_0 from one of x_1, not even if x_0 and x_1 were chosen by $\mathcal{A}$ after seeing the key material of a corrupted party.

Definition 3 (Security of HSS Schemes). *Let* $(\mathsf{Setup}, \mathsf{Input}, \mathsf{Eval})$ *be a 2-party public key HSS scheme for the circuit class* $\mathcal{P}$ *over the ring* R*. Let* $\mathcal{I} \subseteq R$ *be the input space. We say that the scheme is secure if, for every* $i \in \{0, 1\}$*, no PPT adversary* $\mathcal{A}$ *can win the game* $\mathcal{G}_{i,\mathcal{A}}^{\mathsf{HSS\text{-}Sec}}(\lambda)$ *(see Fig. 1) with non-negligible advantage in the security parameter* λ*.*

Note that, since the reconstruction is additive, it is implicit that the shares z_i leak no information about the key ek_i, as they can be simulated given the output of the function and the other key ek_{i-1}.

Observe that a homomorphic secret-sharing scheme naturally induces a semi-honest 2-round MPC protocol with setup for circuits in $\mathcal{P}$. In order to have as little communication complexity as possible, we are interested in designing protocols where the size of the input encodings is small.

The Game $\mathcal{G}^{\mathsf{HSS\text{-}Sec}}_{i,\mathcal{A}}(\lambda)$

1. $b \xleftarrow{R} \{0,1\}$
2. $\big(\mathsf{pk}, (\mathsf{ek}_0, \mathsf{ek}_1)\big) \xleftarrow{R} \mathsf{Setup}(1\!\!1^\lambda)$
3. $(x_0, x_1, \mathsf{state}) \xleftarrow{R} \mathcal{A}(1\!\!1^\lambda, \mathsf{pk}, \mathsf{ek}_i)$
4. $(\mathsf{I}_0, \mathsf{I}_1) \xleftarrow{R} \mathsf{Input}(\mathsf{pk}, x_b)$
5. The adversary wins if $b = \mathcal{A}(\mathsf{state}, \mathsf{I}_i)$ and $x_0, x_1 \in \mathcal{I}$.

Fig. 1. The HSS security game

2.2 Pseudorandom Correlation Functions

A pseudorandom correlation function (PCF) [BCG+20, OSY21] is a construction that allows a set of parties to generate large amounts of distributed correlated material with little communication. Specifically, in the 2-party case, a PCF specifies how to generate 2 small keys, one for each player. After the keys have been dealt, the parties can locally expand them, obtaining large quantities of correlated randomness. The expansion, which is formally called evaluation, takes place in the same fashion as a PRF. In particular, the parties evaluate their keys along with public nonces, obtaining different batches of correlated randomness every time. Notice that the amount of material that can be generated in this way is not necessarily polynomially bounded. Indeed, the nonce space can be exponentially large.

Pseudorandom correlation functions are usually tailored to specific types of correlation. Examples of this kind are OT tuples, in which the receiver obtains a random bit b and a string $\boldsymbol{y} \in \{0,1\}^\lambda$, whereas the sender obtains $\boldsymbol{z} := \boldsymbol{y} \oplus b \cdot \boldsymbol{x}$. Here, the value of $\boldsymbol{x} \in \{0,1\}^\lambda$ is known to the sender and is fixed ahead of time. In particular, all samples from the OT correlation will use the same $\boldsymbol{x}$. In other words, $\boldsymbol{x}$ acts a some kind of secret upon which all the correlation samples depend. In general, each party could own a different correlation secret. In a PCF, all the samples use the same correlation secrets and the latter are distributed to the parties as part of the PCF keys.

On the importance of having small keys. We would like our PCFs to have small keys. Usually, it is not hard to design multiparty computation protocols that generate and distribute the PCF keys with linear communication in the key size. When these protocols are composed with the non-interactive expansion, we obtain secure constructions that generate large quantities of correlated randomness with little communication and no need for trusted dealers.

Unfortunately, having small keys is not possible for any correlation [BCG+19]. In order to have no issues, we have to work with reverse-samplable correlation functions. These are particular types of correlation in which the outputs of the honest parties can be simulated given the outputs of the corrupted players and

their secrets. Moreover, the outputs of the corrupted parties leak no information about the secrets of the honest players. We formalise the definition below.

Definition 4 (Reverse-Samplable 2-party Correlation Function). *A 2-party correlation function is a pair of PPT algorithms* $(\mathsf{Secret}, \mathcal{C})$ *having the following syntax.*

- $\mathsf{Secret}(1\!\!1^\lambda, i) \xrightarrow{R} \mathsf{mk}_i$. *On input the security parameter and the index of a party* $i \in \{0,1\}$*, the algorithm outputs a random correlation secret* mk_i *for* P_i*.*
- $\mathcal{C}(1\!\!1^\lambda, \mathsf{mk}_0, \mathsf{mk}_1) \xrightarrow{R} (R_0, R_1)$. *On input the security parameter and the correlation secret* mk_0 *and* mk_1 *of the parties, the correlation function outputs a pair of correlated values* (R_0, R_1)*, one for each party.*

We say that $(\mathsf{Secret}, \mathcal{C})$ *is reverse-samplable if there exists a PPT algorithm* $\mathsf{RSample}$ *such that, for every* $i \in \{0,1\}$ *and correlation secrets* mk_0*,* mk_1 *and* mk'_i *in the image of* Secret*, no PPT adversary can distinguish between the output of* $\mathcal{C}(1\!\!1^\lambda, \mathsf{mk}_0, \mathsf{mk}_1)$ *and*

$$\left\{ (R_0, R_1) \middle| \begin{array}{l} \mathsf{mk}'_{1-i} \leftarrow \mathsf{mk}_{1-i} \\ (R'_0, R'_1) \xleftarrow{R} \mathcal{C}(1\!\!1^\lambda, \mathsf{mk}'_0, \mathsf{mk}'_1) \\ R_{1-i} \leftarrow R'_{1-i} \\ R_i \xleftarrow{R} \mathsf{RSample}(1\!\!1^\lambda, i, R_{1-i}, \mathsf{mk}_0, \mathsf{mk}_1) \end{array} \right\}$$

The following definition formalises the syntax of PCFs.

Definition 5 (Pseudorandom correlation function). *A 2-party PCF for the reverse-samplable correlation function* $(\mathsf{Secret}, \mathcal{C})$ *is a pair of PPT algorithms* $(\mathsf{Gen}, \mathsf{Eval})$ *with the following syntax.*

- $\mathsf{Gen}(1\!\!1^\lambda) \xrightarrow{R} (\mathsf{k}_0, \mathsf{k}_1)$. *On input the security parameter, the algorithm outputs a pair of PCF keys* $(\mathsf{k}_0, \mathsf{k}_1)$*, one for each party.*
- $\mathsf{Eval}(i, \mathsf{k}_i, x) \to R_i$. *On input the index* $i \in \{0,1\}$ *of a party, the* P_i*'s PCF key* k_i *and a nonce* x *in the nonce space* $\mathcal{X}$*, the algorithm outputs the a value* R_i*, corresponding to the* i*-th output of* $\mathcal{C}$*.*

Informally speaking, we say that a PCF is correct when no PPT adversary can distinguish between samples generated by expanding the keys and values output by the correlation function $\mathcal{C}$. Concerning security, we require that the keys of corrupted parties leak no information about the outputs of the honest parties. This idea is formalised by saying that an adversary provided with the keys of the corrupted players cannot distinguish between the real outputs of the honest parties and reverse-sampled ones [OSY21].

Based on how the nonces are chosen, we can classify PCFs into two main classes [BCG+20]: weak PCFs, in which the nonces are sample at random, for instance using a random oracle, and strong PCFs, in which the parties can adaptively choose them. In this paper, we will work only with weak PCFs.

Public-key PCFs. In general, the MPC protocols used to generate and distribute the PCF keys require multiple rounds of interactions. In some particular cases, however, one round is enough. These particular constructions are called *public-key PCFs* [OSY21]. The name refers to the fact that the only message sent by each party acts as a public key, whereas the randomness used for its generation behaves as its private counterpart.

Formally speaking a public-key PCF can be regarded as a one-round protocol implementing the functionality that generates the correlated material of the honest parties by reverse-sampling the outputs of the corrupted players, which are provided by the adversary. During the initialisation, the adversary is also allowed to choose the correlation secrets of the corrupted parties, whereas those of the honest player are generated by the functionality.

In this paper, we will focus on two types of 2-party correlation, both of which are reverse-samplable. The first one are OT tuples, which were described in the previous paragraph. The second type are vector-OLE tuples, where P_0 is provided with a random pair (z_0, a) in a modular ring $\mathbb{Z}_N$ and P_1 obtains $z_1 = z_0 + a \cdot x \bmod N$. Here, $x \in \mathbb{Z}_N$ denotes the correlation secret of party P_1.

3 A Group-Theoretic Framework

We will assume we have a probabilistic polynomial time algorithm Gen that takes $1\!\!1^\lambda$ as input where λ is a security parameter. When running Gen, we get output

$$\mathsf{par} \xleftarrow{R} \mathsf{Gen}(1\!\!1^\lambda), \text{ where } \mathsf{par} = (G, F, H, f, t, \ell, \mathsf{aux}).$$

Here, G is a finite Abelian group with subgroups F, H such that $G = F \times H$, f is a generator of F and t is the order of F. We assume we can compute the group operation and inverses in time polynomial in λ. The natural number ℓ will be used in the following: when we select a random exponent r and compute g^r where $g \in G$, r will usually be chosen uniformly between 0 and ℓ[2]. Finally, we say that Gen is *public-coin* if the random coins used by Gen appear in the string aux.

We also assume a probabilistic polynomial time algorithm $\mathcal{D}$ for sampling random elements in G. We will use the notation $(g, \rho) \xleftarrow{R} \mathcal{D}(1\!\!1^\lambda, \mathsf{par})$, where $g \in G$ is the sampled element and ρ contains the random coins used in the sampling (i.e., the sampling of g is always *public-coin*). We do not require that g is uniform in G, but we do require f is in the subgroup generated by g, except perhaps with negligible probability.

We assume that discrete log base f is easy, that is, given f^a for any $a \in \mathbb{Z}_t$, a can be computed in polynomial time in λ.

In the following sections, we will specify a number of computational problems that we need to assume are hard to solve, given par and various elements sampled

[2] We will always choose ℓ large enough so that g^r is statistically indistinguishable from uniform in $\langle g \rangle$. This is possible, even if $|H|$ is sometimes not known by anyone, since an upper bound is always known.

by $\mathcal{D}$. Loosely speaking, the most basic one is that the order of the subgroup H is hard to compute, and that the DDH assumption holds in the subgroup generated by g where g is sampled by $\mathcal{D}$. More details will be given in Section 3.1.

The main problem we want to solve in the context of the framework is the following, which we call Non-Interactive Discrete Log Sharing (NIDLS). This is defined as follows:

Definition 6. *The NIDLS problem involves two parties, A and B. A gets as input $\alpha \in G$, while B gets $\beta \in G$. It is promised that $\alpha\beta^{-1} \in F$, so that $\alpha\beta^{-1} = f^m$ for some $m \in \mathbb{Z}_t$. A and B now do only local computation and A outputs a number a, while B outputs b. The goal is that $a + b \equiv m \bmod t$.*

It will be convenient to introduce the following notation: for $g \in G$, we denote by C_g be the coset of F in G that contains g. As we explain in a moment, the NIDLS problem can be solved using the following tool:

Definition 7. *A* coset labelling function for F in G *is an efficiently computable function $\phi : G \mapsto G$ with the following property: for any $g \in G$ we have $\phi(g) \in C_g$ and furthermore, for any $h \in C_g$ we have $\phi(h) = \phi(g)$.*

In other words, for every coset C_g, ϕ defines a fixed element $c \in C_g$ and c can be efficiently computed given any element in C_g.

Given a coset labelling function the NIDLS problem can be solved using the following protocol:

1. A computes $\phi(\alpha)^{-1} \cdot \alpha$ which is in F since α and $\phi(\alpha)$ are in the same coset. Using that discrete log in F is easy, A computes a such that $\phi(\alpha)^{-1} \cdot \alpha = f^a$, and outputs a.
2. B computes $\phi(\beta) \cdot \beta^{-1}$ which is in F since β and $\phi(\beta)$ are in the same coset. Using that discrete log in F is easy, B computes b such that $\phi(\beta) \cdot \beta^{-1} = f^b$, and outputs b.

This works because the property of ϕ guarantees that $\phi(\alpha) = \phi(\beta)$. Therefore

$$f^a \cdot f^b = \phi(\alpha)^{-1} \cdot \alpha \cdot \phi(\beta) \cdot \beta^{-1} = \alpha \cdot \beta^{-1} = f^m,$$

from which it follows immediately that $a + b \equiv m \bmod t$.

It turns out that if F is small, then a coset labelling function always exists:

Lemma 1. *Let $G = F \times H$ be groups as described above, where the order t of F is polynomial. Then a coset labelling function for F in G always exists.*

Proof. We define the desired function ϕ as follows: on input g, compute a list of all elements in C_g by multiplying g by all powers of f. This is feasible since t is polynomial. Sort the elements in lexicographical order and output the first element. As the content of the list is the same no matter which element in the coset we start from, this function has the desired property. □

There is also a different approach to constructing a coset labelling function which, as we shall see, sometimes works for superpolynomial size F.

Namely, assume that for every G that Gen can produce, there exists an efficiently computable and surjective homomorphism $\pi : G \mapsto G'$ (for some group G'), where $ker(\pi) = F$. This implies that for each coset of F in G, π maps all elements of the coset to a single element in G', and that distinct cosets are mapped to distinct elements.

Note that $\pi(g)$ is actually a unique "label" for the coset C_g, the only problem is that it is in G' and not in G.

To get around this, we assume that outputs from π can be "lifted" deterministically to G such that we land in the coset we came from. That is, we assume there exists an efficiently computable function $\delta : G' \mapsto G$ such that for any $x \in G'$ we have that $\delta(x)$ is in the coset of F in G that is mapped to x by π. Put slightly differently, what we want is that $\pi(\delta(x)) = x$ for all $x \in G'$.

Now, observe that $\delta(\pi(g))$ only depends on which coset g belongs to, since $\pi(g)$ already has this property. Therefore, the following lemma is immediate:

Lemma 2. *Let $G = F \times H$, G' be groups as described above and π, δ be functions as described above, with $\pi(\delta(x)) = x$ for all $x \in G'$. Then ϕ defined by $\phi(g) = \delta(\pi(g))$ is a coset labelling function for F in G.*

3.1 Assumptions

In this section we list the computational assumptions we need in order to prove our constructions secure.

Definition 8 (Weak Hidden Order Assumption). *We say that the weak hidden order assumption holds in the NIDLS framework if for any PPT adversary $\mathcal{A}$:*

$$\Pr[\mathcal{A}(\mathsf{par}, g, \rho) = x \text{ and } g^x = 1] = \mathsf{negl}(\lambda)$$

when $\mathsf{par} := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(\mathbb{1}^\lambda)$ *and* $(g, \rho) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$.

Notice that in the standard hidden order assumption [Tuc20], the adversary is let free to choose any $g \neq 1$. We rely instead on a *weaker* assumption in which g is sampled according to $\mathcal{D}$.

Definition 9 (DDH Assumption). *We say that the DDH assumption holds in the NIDLS framework if for any PPT adversary $\mathcal{A}$ the following quantity is negligible:*

$$|\Pr[\mathcal{A}(\mathsf{par}, \rho, g, g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{A}(\mathsf{par}, \rho, g, g^x, g^y, g^z) = 1]| = \mathsf{negl}(\lambda)$$

when $\mathsf{par} := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(\mathbb{1}^\lambda), (g, \rho) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par}), (x, y, z) \xleftarrow{R} [\ell]^3$.

We introduce a new variant of the DDH assumption that allows us to infer the security of our protocols that use two generators g, C which are generated with a trustless setup i.e., the adversary is allowed to see the random coins used for their generation. In some settings, this assumption is equivalent to DDH but this does not cover all our instantiations of the framework[3].

Definition 10 (Decisional Cross-Group DH Assumption (DXDH)). *We say that the DXDH assumption holds in the NIDLS framework if for any PPT adversary $\mathcal{A}$:*

$$|\Pr[\mathcal{A}(\mathsf{par}, g, \rho_0, C, \rho_1, g^r, C^r) = 1] - \Pr[\mathcal{A}(\mathsf{par}, g, \rho_0, C, \rho_1, C^s, C^r) = 1]| = \mathsf{negl}(\lambda)$$

when $\mathsf{par} := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(1\!\!1^\lambda)$, $(g, \rho_0) \xleftarrow{R} \mathcal{D}(1\!\!1^\lambda, \mathsf{par})$, $(C, \rho_1) \xleftarrow{R} \mathcal{D}(1\!\!1^\lambda, \mathsf{par})$, $C \neq g$ *and* $(r, s) \xleftarrow{R} [\ell]^2$.

Finally, in our HSS constructions, we would like to have ElGamal-style secret keys bounded by $\ell_{\mathsf{sk}} < t$, which may be significantly smaller than ℓ. This allows to encrypt the private key under its public counterpart without worrying about wrap-arounds. In order for security to hold in these conditions, we rely on the small exponent assumption defined below.

Definition 11 (Small Exponent Assumption). *We say that the small-exponent assumption with length $\ell_{\mathsf{sk}}(\lambda)$ holds in the NIDLS framework if for any PPT adversary $\mathcal{A}$:*

$$|\Pr[\mathcal{A}(\mathsf{par}, \ell_{\mathsf{sk}}, g, \rho, g^x) = 1] - \Pr[\mathcal{A}(\mathsf{par}, \ell_{\mathsf{sk}}, g, \rho, g^y) = 1]| = \mathsf{negl}(\lambda)$$

when $\mathsf{par} := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(1\!\!1^\lambda)$, $(g, \rho) \xleftarrow{R} \mathcal{D}(1\!\!1^\lambda, \mathsf{par})$, $x \xleftarrow{R} [\ell]$ *and* $y \xleftarrow{R} [\ell_{\mathsf{sk}}]$.

4 Instantiations of the Framework

In this section, we give a number of concrete instantiations of the framework we just discussed. Some were already known, and some are new.

4.1 Paillier and Damgård-Jurik

This example was already known from [OSY21] who presented a NIDLS protocol based on Paillier encryption and independent work from [RS21] who did it from Damgård-Jurik encryption.

These instantiations are closely related and we cover them in one go as follows: we let $\mathsf{Gen}(1\!\!1^\lambda)$ output an RSA modulus $n = pq$ of bit length λ, where

[3] For equivalence, it is needed that g and C are random generators of the same subgroup and that $\mathcal{D}$ is invertible, i.e., that given any group element h in the output domain, one can efficiently compute random coins that would cause $\mathcal{D}$ to output h.

$p' = (p-1)/2$ and $q' = (q-1)/2$ are also prime and where $gcd(n, \phi(n)) = 1$. We set $G = \mathbb{Z}^*_{n^s}$, for some constant natural number $s \geq 2$ and it now holds that $G = F \times H$ where F is the subgroup of order n^{s-1}, and H is the subgroup of order $(p-1)(q-1)$. Discrete log in F is easy in this case (see [DJ01] for details). This generator is not public-coin, as the prime factors of n must remain secret.

To get a coset labelling function for this example, we use Lemma 2: we set $G' = \mathbb{Z}^*_n$ and $\pi(g) = g \bmod n$. Since n divides n^s, it is clear that π is a surjective homomorphism from G to G'. Therefore its kernel has order $|G|/|G'| = n^{s-1}$. Note that all non-trivial elements in F must have orders relatively prime to $\phi(n) = |G'|$ and hence the homomorphism into G' must send all these elements to 1. It follows that F is contained in the kernel and so is in fact equal to the kernel because $|F| = n^{s-1}$. We define the function $\delta : G' \mapsto G$ by $\delta(x) = x$, that is, δ just returns its input, but now understood as a number modulo n^s (instead of n).

With these definitions, it is clear that $\pi(\delta(x)) = x$, so by Lemma 2, $\phi(g) = \delta(\pi(g))$ is a coset labelling function.

The sampling algorithm $\mathcal{D}$ will output a random $g \in \mathbb{Z}^*_{n^s}$, such that the Jacobi symbol of g modulo n is 1. Note that because $|F| = n^{s-1}$ contains only large prime factors, a random g will contain F in the subgroup it generates except with negligible probability. Similarly, reducing modulo n, we see that $g \bmod n$ has order divisible by $p'q'$ except with negligible probability since p', q' are prime.

As for the assumptions, computing the order is trivially equivalent to factoring n. The DDH assumption was introduced in [DJ03] and used there for an "El-Gamal style" variant of Paillier encryption. In this setting, we can claim that if you can break the DXDH assumption, you can also break DDH. This is because g (or C) sampled as above have order $n^{s-1}p'q'$ or $2n^{s-1}p'q'$ except with negligible probability. Whether 2 divides the order cannot be efficiently determined (by the standard quadratic residuosity assumption). Further, the sampling algorithm is clearly invertible. All this means that, given an element g^x from a DDH challenge, we can claim it was instead sampled by $\mathcal{D}$ and let it play the role of C in the DXDH setting.

Finally, the small exponent assumption is reasonable in a setting where discrete log and DDH are hard, as we do assume here, as long as the domain from which the exponent is chosen is exponentially large. Also, this type of assumption has been used several times before, for instance in [BCG+17] to optimize an HSS construction.

4.2 Joye-Libert Variants

Small order F. In this example, the generator outputs an RSA modulus $n = pq$ where 2^ℓ is the maximal 2-power that divides $p-1$, and $q-1$. It also outputs an element $f \in \mathbb{Z}^*_n$ of order 2^ℓ modulo both p and q (and so it also has order 2^ℓ modulo n). Let $p' = (p-1)/2^\ell, q' = (q-1)/2^\ell$, where we assume that p', q' are prime. Then we let $F = \langle f \rangle$, we let $H \leq \mathbb{Z}^*_n$ be the subgroup of order $p'q'$, and we set $G = F \times H$. The group G is actually not all of $\mathbb{Z}^*_n$, but this is of no

consequence in the following. Discrete log in F is easy by the Pohlig-Hellman algorithm.

For this variant, as long as 2^ℓ is polynomial, we can use Lemma 1 to get a coset labelling function. Doing it for larger values of 2^ℓ is an open problem. When $\ell = 1$, we can set $f = -1$ and we get a setting closely related to the Goldwasser-Micali cryptosystem, as observed in [OSY21].

Large order F. We now construct a different variant of the Joye-Libert case where we are able to accommodate an exponentially large order subgroup F. Once again, the generator outputs an RSA modulus $n = pq$. This time, both $p-1$ and $q-1$ are divisible by the product of the first ℓ primes q_ℓ, that is $q_\ell = \prod_{i=1}^{\ell} p_i$ where p_i is the i'th prime.

We let $f \in \mathbb{Z}_n^*$ be an element of order q_ℓ modulo both p and q. Let $p' = (p-1)/q_\ell, q' = (q-1)/q_\ell$. As before, we let $F = \langle f \rangle$, we let $H \leq \mathbb{Z}_n^*$ be the subgroup of order $p'q'$, and we set $G = F \times H$.

It is not hard to see that since the i'th prime is approximately $i \ln i$, we can arrange for q_ℓ to be exponentially large, while each prime in the product is only polynomial.

We now show that if all primes in the product q_ℓ are polynomial size, we can solve the NIDLS problem in this setting, basically by using Lemma 1 for each p_i and then assembling a complete solution using the Chinese remainder theorem (CRT).

Some notation: we have $F = F_1 \times ... \times F_\ell$, where F_i is of order p_i. So it follows from Lemma 1 that we have a coset labelling function ϕ_i for the group $G_i = F_i \times H$. Also, if we let $u_i = q_\ell / p_i$, then $f_i = f^{u_i}$ is a generator of F_i. Now observe that if α, β is an instance of the NIDLS problem in $G = F \times H$, then $\alpha^{u_i}, \beta^{u_i}$ is an instance of the NIDLS problem in $G_i = F_i \times H$. This is simply because $\alpha \cdot \beta^{-1} = f^m$ implies $\alpha^{u_i} \cdot (\beta^{u_i})^{-1} = (f^{u_i})^m = f_i^{m \bmod p_i}$. Using this notation, the protocol works as follows:

1. For each $i = 1...\ell$, A uses ϕ_i to compute a solution a_i to the NIDLS problem in G_i. Finally, using CRT, A computes and outputs $a \in \mathbb{Z}_t$ such that $a \bmod p_i = a_i$ for all i.
2. For each $i = 1...\ell$, B uses ϕ_i to compute a solution b_i to the NIDLS problem in G_i. Finally, using CRT, B computes and outputs $b \in \mathbb{Z}_t$ such that $b \bmod p_i = b_i$ for all i.

This works because $(a+b) \bmod p_i = (a_i + b_i) \bmod p_i$ by definition of a, b, and since a_i, b_i solves the NIDLS problem in G_i we further have

$$(a+b) \bmod p_i = (a_i + b_i) \bmod p_i = m \bmod p_i.$$

Since this holds for all i, CRT implies that $a + b \bmod q_\ell = m$.

For this instantiation, the sampling algorithm $\mathcal{D}$ will choose a random $r \in \mathbb{Z}_n^*$ and output $g = f \cdot r^{q_\ell} \bmod n$. Note that $r^{q_\ell} \bmod n$ has order $p'q'$ except with

negligible probability, in particular, the order is prime to q_ℓ so g has order $q_\ell p' q'$, and hence f is in the group generated by g.

The assumptions for this instantiation can be motivated similarly to what was done for Paillier above, as also here we rely on factoring to hide the order of the group. For this to be reasonable, we need, of course, that q_ℓ is much smaller than n so that enough uncertainty remains about p, q even given q_ℓ. The exception is that in this case, $\mathcal{D}$ is not invertible, so we cannot claim that DDH implies DXDH. The assumptions are also closely related to what Joye and Libert [JL13] assumed for their cryptosystem, but one should note that our assumptions are stronger because we need to make an element of order exactly 2^ℓ (or q_ℓ) public, while they just needed an element of order divisible by 2^ℓ. When 2^ℓ is small, such an element can be guessed with good probability while it is not clear how to efficiently compute an element of order exactly 2^ℓ given only n.

4.3 Class Groups

We explain here how to instantiate our framework on top of the CL framework [CL15] (see also [Tuc20] for an excellent introduction to class groups). Basically, we take the CL framework, and combine this with the observation that a coset-labelling function can be obtained from a surjective homomorphism used previously in the NICE cryptosystem [PT00, CJLN09].

Let $\mathsf{Gen}(1\!\!1^\lambda)$ output two primes p and q such that $pq \equiv 3 \pmod 4$ and $(p/q) = -1$. This generator is public-coin, p and q will be public. We set $\Delta_K = -pq$ and $\Delta_q = -pq^3$. We set $G = Cl(\Delta_q)$, the class group of the quadratic order $\mathcal{O}_{\Delta_q}$ of discriminant Δ_q and $G' = Cl(\Delta_K)$ the class group of the maximal order $\mathcal{O}_{\Delta_K}$. The size of pq is chosen such that computing the class number $|G'|$ is intractable.

Let $f \in G$ be the class of the ideal $q^2\mathbb{Z} + (-q + \sqrt{\Delta_q})/2\mathbb{Z}$ then f has order q and the discrete logarithm problem in F, generated by f, is easy.

If q has λ bits then q is prime to $|G'|$ except with negligible probability by the Cohen-Lenstra heuristics. Then $G \simeq F \times H$ where H is a subgroup of order $|G'|$.

We denote by $I(\mathcal{O}_{\Delta_q}, q)$ (resp. $I(\mathcal{O}_{\Delta_K}, q)$) the subgroup of fractional ideals generated by $\mathcal{O}_{\Delta_q}$-ideals prime to q (resp. of $\mathcal{O}_{\Delta_K}$-ideals prime to q). Then, the map $\varphi_q : I(\mathcal{O}_{\Delta_q}, q) \to I(\mathcal{O}_{\Delta_K}, q)$, $\mathfrak{a} \mapsto \mathfrak{a}\mathcal{O}_{\Delta_K}$ is an isomorphism. The reverse map is $\varphi_q^{-1} : I(\mathcal{O}_{\Delta_K}, q) \to I(\mathcal{O}_{\Delta_q}, q)$, $\mathfrak{a} \mapsto \mathfrak{a} \cap \mathcal{O}_{\Delta_q}$. Both maps are efficiently computable knowing q. The map φ_q induces a surjective homomorphism from G to G'. This will be the surjection π of the framework. The kernel of π is F.

We then define the function $\delta : G' \mapsto G$ by $\delta(x) = [\varphi_q^{-1}(\mathfrak{a})]$ where $\mathfrak{a}$ is an ideal in the class of x prime to q (it can also be found efficiently).

We then have $\pi(\delta(x)) = x$ by construction, so by Lemma 2, $\phi(g) = \delta(\pi(g))$ is a coset labelling function.

As sampling algorithm $\mathcal{D}$ we use the one introduced in [CL15], and also described in [Tuc20], section 3.1.2. It outputs g of large order such that f is guaranteed to be in the subgroup generated by g. Very briefly, it works by selecting a small prime r such that Δ_K is a square modulo r. From this r, we can construct an element in G' by considering the ideal that lies "above r" and

the class of this ideal squared. We then lift this element to G as explained above, to get a group element h. Finally, we output $g = f \cdot h^t$.

With this sampling algorithm, the DDH assumption is the same that has been used before in the CL framework, sometimes known as the DDH-CL assumption. The DXDH assumption in this setting is not implied by DDH, since elements sampled from different randomness do not necessarily generate the same group. Nevertheless, we can argue that the assumption is reasonable: to break it, one needs to decide, for given g, C if a pair of group elements is of form g^r, C^r. The natural approach to this is to use index calculus type methods to find a relation of form $g^a = C^b$ which, for a pair of the form mentioned would imply $(g^r)^a = (C^r)^b$. However, once such an attack succeeds one would also be in a position to find orders of elements and hence break the (much more standard) hidden order assumption.

5 HSS Constructions

In this section, we explain how any instantiation of the framework can be used to build a cryptosystem and a homomorphic secret-sharing scheme (HSS) for restricted multiplication straight-line programs (RMS). Note that given the NIDLS-ElGamal encryption and a distributed DDLOG procedure, constructing an HSS follows in a more or less direct way by following the blueprint of the HSS in [OSY21]. However, since upcoming sections build on top of the HSS we provide the full description of the HSS anyway to make the paper self-contained.

5.1 NIDLS ElGamal

Our HSS construction is based on an ElGamal-style encryption scheme instantiated over our group-theoretic framework. We refer to the construction by *NIDLS ElGamal*, the cryptosystem is formally described in Fig. 2. Correctness of the construction follows immediately as for standard ElGamal.

CPA Security. Similarly to [CL15], the security of NIDLS ElGamal is implied by the DDH assumption, which states that random tuples (g, g^x, g^y, g^{xy}) are indistinguishable from (g, g^x, g^y, g^z). Since $\mathcal{D}$ outputs elements g for which $f \in \langle g \rangle$, we can use g^z to hide f^x.

Generating encryptions of the secret key. Note that in addition to the standard algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, we have included an additional algorithm SkEnc which encrypts the message "in the wrong place". It turns out that this results in a valid encryption of the value $s \cdot x \bmod t$ i.e., an encryption of the secret key s times the input value x. In particular

$$c_1 \cdot c_0^{-s} = h^r \cdot (g^r \cdot f^{-x})^{-s} = (g^{rs} \cdot g^{-rs}) \cdot f^{sx}$$

This will be useful in our HSS construction.

ElGamal Cryptosystem

$\mathsf{EG.Gen}(1^\lambda)$:

1. Sample $\mathsf{par} := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(\mathbb{1}^\lambda)$
2. Sample a random $(g, \rho) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
3. Sample a random $s \xleftarrow{R} [\ell]$, and let $h = g^s$
4. Output $\mathsf{pk} = (\mathsf{par}, g, \rho, h)$ and $\mathsf{sk} = s$.

$\mathsf{EG.Enc}(\mathsf{pk}, x)$:

1. Sample a random $r \xleftarrow{R} [\ell]$
2. Output $\mathsf{ct} = (g^r, h^r \cdot f^x)$

$\mathsf{EG.Dec}(\mathsf{sk}, \mathsf{ct} = (c_0, c_1))$:

1. Output $x = \mathsf{DLog}_f(c_1 \cdot c_0^{-s})$

$\mathsf{EG.SkEnc}(\mathsf{pk}, x)$:

1. Sample a random $r \xleftarrow{R} [\ell]$
2. Output $\mathsf{ct} = (g^r \cdot f^{-x}, h^r)$

Fig. 2. A description of the ElGamal cryptosystem in the NIDLS framework.

We prove that encryptions performed using $\mathsf{EG.SkEnc}(\mathsf{pk}, x)$ preserve the privacy of x. Observe that we do not need to rely on any circular security assumption.

Lemma 3. *If the DDH assumption holds in the NIDLS framework, no PPT adversary $\mathcal{A}$ can win the game $\mathcal{G}_{\mathcal{A}}^{s\text{-}\mathsf{EG}}(\lambda)$ with non-negligible advantage in the security parameter.*

Proof. Let $\mathcal{A}$ be a PPT adversary. Consider the following hybrids.

Hybrid 0. The initial stage corresponds to the game $\mathcal{G}_{\mathcal{A}}^{s\text{-}\mathsf{EG}}(\lambda)$. In other words, after receiving the public key pk and selecting a value $x \in \mathbb{Z}_t$, the adversary is provided with $(g^r \cdot f^{-b \cdot x}, h^r)$ with a random $r \xleftarrow{R} [\ell]$.

Hybrid 1. This hybrid is identical to the previous one except for the fact that we substitute g^r with g^u for a uniformly sampled $u \in [\ell]$. By the DDH assumption, this hybrid is indistinguishable from Hybrid 2. Since $f \in \langle g \rangle$ (we recall that g is sampled using $\mathcal{D}$) and the distribution of $g^r \cdot f^{-b \cdot x}$ is statistically close to the uniform distribution over $\langle g \rangle$, no adversary can distinguish between the case $b = 0$ and the case $b = 1$. As a consequence, the advantage of $\mathcal{A}$ in $\mathcal{G}_{\mathcal{A}}^{s\text{-}\mathsf{EG}}(\lambda)$ must be negligible. □

The Game $\mathcal{G}_{\mathcal{A}}^{s\text{-}\mathsf{EG}}(\lambda)$

1. $b \xleftarrow{R} \{0,1\}$
2. $(\mathsf{pk}, s) \xleftarrow{R} \mathsf{EG.Gen}(1\!\!1^{\lambda})$
3. $(x, \mathsf{state}) \xleftarrow{R} \mathcal{A}(1\!\!1^{\lambda}, \mathsf{pk})$
4. The adversary wins if $b = \mathcal{A}\big(\mathsf{state}, \mathsf{EG.SkEnc}(\mathsf{pk}, b \cdot x)\big)$.

Fig. 3. The s-ElGamal game

5.2 Public-Key HSS

We now present a homomorphic secret-sharing scheme (HSS) for RMS programs based on the NIDLS framework. The main advantage of our NIDLS-based HSS compared to the Paillier-based HSS of [OSY21] is that we remove any need for trusted setups when instantiating the NIDLS over class groups, while previous constructions had to rely on a trusted dealer for the generation of an RSA modulus.

RMS programs. Restricted multiplications straight-line (RMS) programs are arithmetic circuits over $\mathbb{Z}$ that never compute multiplications between two intermediate value of the computation: at least one of the two factors must be an input. Intermediate values of the computation are often referred to as memory values. This class includes also branching programs, which likewise contains NC^1.

Definition 12 (RMS Programs). *An RMS program consists of a bound $B \in \mathbb{N}$, a modulo $n_{\mathsf{out}} \in \mathbb{N}$ and a polynomial-sized circuit in which the only gate types allowed are the following.*

- $\mathsf{ConvertInput}(\mathsf{I}_x) \to \mathsf{M}_x$. *Load the value of the input wire I_x to the memory wire M_x.*
- $\mathsf{Add}(\mathsf{M}_x, \mathsf{M}_y) \to \mathsf{M}_z$. *Add the values of the memory wires M_x and M_y and assign the result to the memory wire M_z.*
- $\mathsf{Mult}(\mathsf{I}_x, \mathsf{M}_y) \to \mathsf{M}_z$. *Multiply the value of the input wire I_x by the value of the memory wire M_y. Assign the result to the memory wire M_z.*
- $\mathsf{Output}(\mathsf{M}_z) \to z$. *Output the value of the memory wire M_z reducing it modulo n_{out}.*

The circuit accepts only integral inputs. Whenever the absolute value $|x|$ of any wire exceeds the bound B, the output of the execution is $\perp$.

The public-key HSS scheme. We are now ready to present our construction, which is formally described in Fig. 4. We discuss the main ideas.

Our HSS scheme allows two parties to non-interactively apply an RMS program C on secret-shared inputs, obtaining additively secret-shared outputs. The scheme relies on a setup procedure[4] that provides the parties with a PRF key

[4] Following the blueprint of [OSY21], it is possible to substitute the setup with a one-round protocol.

k, a NIDLS ElGamal public key pk, and a subtractive secret-sharing over the integers of the private counterpart $s = s_1 - s_0$. We assume that the length $\mathsf{len}_{\mathsf{sk}}$ of the private key is sufficiently small, so that $s < t$. If this condition is not satisfied, we need to proceed as in [OSY21], splitting the private key into small blocks and providing the parties with an encryption of each of them.

Input wires and memory wires. During the evaluation of the circuit C, each input wire I_x is associated with two NIDLS ElGamal ciphertexts: an encryption of the value of the wire x and an encryption of the product between x and the ElGamal secret key s. Such ciphertexts are produced and broadcast by the party providing the input. Remember that one does not need to know s in order to encrypt $x \cdot s$. Indeed, the algorithm SkEnc described in Section 5.1 can be used instead. Each memory wire M_x is instead associated with two subtractive secret-sharings over the integers: a secret-sharing of the value of the wire x and a secret-sharing of $x' := x \cdot s$.

Linear operations. Performing additions between memory values is straightforward due to the linearity of subtractive secret-sharing, i.e. to add M_x and M_y, it is sufficient to compute $[z] \leftarrow [x]+[y]$ and $[z'] \leftarrow [x']+[y'] = [x \cdot s]+[y \cdot s]$. Observe that additions allow us to model also multiplications by public constants in $\mathbb{Z}$.

Multiplications between input wires and memory wires. Multiplications between input wires and memory wires require more interesting techniques based on DDLOG. Let $\mathsf{ct}_x = (c_0, c_1)$ be the ElGamal encryption of x, the value of the input wire I_x. Moreover, let $[y]$ and $[y' = y \cdot s]$ be the subtractive secret-sharings associated with the memory wire M_y. In particular, the parties P_0 and P_1 own integers y_0, y'_0 and y_1, y'_1 such that $y_1 = y_0 + y$ and $y'_1 = y'_0 + y \cdot s$. Now, observe that $c_1^{y_0} \cdot c_0^{-y'_0}$ and $c_1^{y_1} \cdot c_0^{-y'_1}$ are a divisive secret-sharing of f^{xy}. Indeed,

$$c_1^{y_1} \cdot c_0^{-y'_1} = c_1^{y_0+y} \cdot c_0^{-(y'_0+y \cdot s)} = (c_1 \cdot c_0^{-s})^y \cdot c_1^{y_0} \cdot c_0^{-y'_0} = f^{xy} \cdot c_1^{y_0} \cdot c_0^{-y'_0}.$$

By applying DDLOG on the respective divisive shares, the parties are therefore able to obtain a secret-sharing of the product $x \cdot y$ over $\mathbb{Z}_t$ (we recall that $t := \mathsf{ord}(f)$). By repeating the procedure for the other ciphertext associated with the input wire I_x, namely the encryption of $x \cdot s$, the parties can non-interactively obtain also a secret-sharing of $x \cdot y \cdot s$. Observe that the additive secret-sharings over $\mathbb{Z}_t$ can be easily converted into subtractive ones by simply changing the signs of the shares of P_0. In order to be sure that the shares are random over $\mathbb{Z}_t$, we rerandomise them using the PRF key k. As a consequence, as long as $|x \cdot y \cdot s| \ll t$, with overwhelming probability, the difference of the shares does not wrap around t, so the parties actually obtain a subtractive secret-sharing over $\mathbb{Z}$.

Input conversions and outputs. It remains to explain how to perform the input conversions and how to retrieve the outputs. Both operations are now rather straightforward. In order to convert an input to a memory element, it is indeed

HSS Scheme

$\mathsf{Setup}(\mathbb{1}^\lambda)$:

1. Let $\mathsf{par} := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(\mathbb{1}^\lambda)$ and ℓ_{sk} be the parameter for the small-exponent assumption.
2. $(g, \rho) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
3. $s_0, s_1 \xleftarrow{R} [\ell_{\mathsf{sk}}]$
4. $\mathsf{pk} \leftarrow g^{s_1} \cdot g^{-s_0}$
5. $k \xleftarrow{R} \{0,1\}^\lambda$
6. Output $\big(\mathsf{par}, g, \rho, \ell_{\mathsf{sk}}, \mathsf{pk}, k, (s_0, s_1)\big)$.

$\mathsf{Input}(\mathsf{pk}, x)$:

1. $\mathsf{ct}_x \xleftarrow{R} \mathsf{EG.Enc}(\mathsf{pk}, x)$
2. $\mathsf{ct}_{xs} \xleftarrow{R} \mathsf{EG.SkEnc}(\mathsf{pk}, x)$
3. Output $\mathsf{I}_x \leftarrow (\mathsf{ct}_x, \mathsf{ct}_{xs})$.

$\mathsf{Eval}\big(i, s_i, (\mathsf{I}^1, \mathsf{I}^2, \ldots, \mathsf{I}^n), P\big)$:

Party P_i evaluates the RMS program P gate by gate as follows.

- $\mathsf{M}_x \leftarrow \mathsf{ConvertInput}(\mathsf{I}_x)$:
  Compute $\mathsf{M}_x \leftarrow \mathsf{Mult}\big(\mathsf{I}_x, \mathsf{M}_1 := (i, s_i)\big)$.
- $\mathsf{M}_z \leftarrow \mathsf{Add}(\mathsf{M}_x, \mathsf{M}_y)$:
  Compute $z_i \leftarrow x_i + y_i$ and $z'_i \leftarrow x'_i + y'_i$ and set $\mathsf{M}_z \leftarrow (z_i, z'_i)$.
- $\mathsf{M}_z \leftarrow \mathsf{Mult}(\mathsf{I}_x, \mathsf{M}_y)$:
  Let $\mathsf{ct}_x = (c_0, c_1)$ and $\mathsf{ct}_{xs} = (d_0, d_1)$. Let id be the label of the gate.
  1. $z_i \leftarrow (-1)^{1-i} \cdot \mathsf{DDLog}\big(c_1^{y_i} \cdot c_0^{-y'_i}\big) + \mathsf{F}_k(\mathsf{id}, 0) \bmod t$
  2. $z'_i \leftarrow (-1)^{1-i} \cdot \mathsf{DDLog}\big(d_1^{y_i} \cdot d_0^{-y'_i}\big) + \mathsf{F}_k(\mathsf{id}, 1) \bmod t$
  3. $\mathsf{M}_z \leftarrow (z_i, z'_i)$
- $\mathsf{Output}(\mathsf{M}_z)$:
  1. Output $(-1)^{1-i} \cdot z_i \bmod n_{\mathsf{out}}$

Fig. 4. The HSS scheme for RMS programs based on the NIDLS framework.

sufficient to multiply it by a memory value containing 1. The latter corresponds to a subtractive secret-sharing of 1, e.g. $y_1 = 1$ and $y_0 = 0$ and a subtractive secret-sharing of s, which was provided to the parties by the initial setup. Outputting the value of a memory wire M_z is even simpler, the parties just broadcast their share of z reducing it modulo n_{out}. By subtracting the two messages modulo n_{out}, the players can obtain the final result of the computation.

On the bound on the values of the wires. The correctness of the HSS scheme described above relies on the assumption that $|x \cdot y \cdot s| \ll t$ for every multiplication. If this condition is not satisfied, there is a non-negligible probability that the secret-sharing over $\mathbb{Z}_t$ obtained as result cannot be converted into an integer secret-sharing of the same value. Observe, anyway, that denoting by B the bound of the RMS circuit, $|x \cdot y \cdot s| \leq B \cdot 2^{\mathsf{len}_{\mathsf{sk}}}$, so, in order to circumvent

the problem, we can choose the parameters of the NIDLS framework so that $B \cdot 2^{\mathsf{len_{sk}}} \cdot 2^{\lambda} < t$.

Theorem 1. *If the DDH assumption and the small exponent assumption hold in the NIDLS framework and* F *is a secure PRF outputting values in* $\mathbb{Z}_t$*, the construction in Fig. 4 is a correct and secure HSS scheme for RMS circuits with bound* $B < t/2^{\mathsf{len_{sk}}+\lambda}$*. The ring where the computation takes place is* $R = \mathbb{Z}_{n_{\mathsf{out}}}$*. Assuming* $n_{\mathsf{out}} < B$*, the input space is* $\mathcal{I} = R$*.*

Proof. It is straightforward to see that our construction follows the syntax of HSS schemes. We start by proving that it also satisfied correctness.

Correctness. Let P be the RMS circuit we are evaluating and let $x_1, x_2, \ldots, x_n \in R$ be the inputs. We assume that the bound B of the program is smaller than $t/2^{\mathsf{len_{sk}}+\lambda}$.

We introduce some notation: for every memory wire M in P, we denote by $\mathsf{val}(\mathsf{M})$ the value that M assumes in $P(x_1, x_2, \ldots, x_n)$ where additions and multiplications are performed over the integers (i.e. we perform no modulo n_{out} reductions). We prove that during the HSS evaluation, every memory wire M is associated with a subtractive secret-sharing over the integers of $\mathsf{val}(\mathsf{M})$ and a subtractive secret-sharing over the integers of $\mathsf{val}(\mathsf{M}) \cdot s$. Once we have proven that, correctness is straightforward. Indeed, since modulo reductions commute with integer additions and multiplications, by reducing the shares of an output wire modulo n_{out} and flipping the sign of P_0's value, we obtain an additive secret-sharing of the result over $\mathbb{Z}_{n_{\mathsf{out}}}$.

Initially, the only memory wire with an assigned value is the one used for input conversions. Each party P_i associates it with a pair $(y_i, y_i') = (i, s_i)$, which indeed corresponds to subtractive secret-sharing of 1 and s (the setup outputs s_0 and s_1 such that $s_1 - s_0 = s$).

The correctness of additions is straightforward: if M_x is associated with $[x]$ and $[s \cdot x]$ and M_y is associated with $[y]$ and $[s \cdot y]$, by the linearity of subtractive secret-sharing over the integers, the sum of the two wires is associated with $[x] + [y] = [x + y]$ and $[s \cdot x] + [s \cdot y] = [s \cdot (x + y)]$.

It remains to prove the correctness of multiplications. Let $(c_0, c_1) = (g^r, f^x \cdot \mathsf{pk}^r)$ be an ElGamal ciphertext encrypting x. Let M_y be a memory wire and assume that P_0 and P_1 own pairs (y_0, y_0') and (y_1, y_1') respectively where $y_1 = y_0 + y$ and $y_1' = y_0' + y \cdot s$. As we pointed out above, we have that

$$c_1^{y_1} \cdot c_0^{-y_1'} = c_1^{y_0+y} \cdot c_0^{-(y_0'+y \cdot s)} = (c_1 \cdot c_0^{-s})^y \cdot c_1^{y_0} \cdot c_0^{-y_0'} = f^{xy} \cdot c_1^{y_0} \cdot c_0^{-y_0'}.$$

So, by applying DDLog on $c_1^{y_1} \cdot c_0^{-y_1'}$ and $c_1^{y_0} \cdot c_0^{-y_0'}$, the parties obtain an additive secret-sharing of $x \cdot y$ over $\mathbb{Z}_t$ (see Section 3). Since the sign of P_0's DDLOG output is flipped, it is immediate to see that $z_1 - z_0 \equiv x \cdot y \bmod t$. By repeating the argument with an encryption of $x \cdot s$, we conclude, in a similar way, that $z_1' - z_0' \equiv x \cdot y \cdot s \bmod t$. Now, by the PRF security, substituting $\mathsf{F}_k(\mathsf{id}, 0)$ and $\mathsf{F}_k(\mathsf{id}, 1)$ with uniformly random values in $\mathbb{Z}_t$ does not affect the correctness of

the HSS scheme (if that was the case, we could break the PRF security). In such hybrid, the outputs of the multiplication are truly random 2-party subtractive secret-sharing of $x \cdot y$ and $x \cdot y \cdot s$ over $\mathbb{Z}_t$. The probability that $z_1 - z_0$ and $z_1' - z_0'$ wrap around t is therefore negligible. Indeed, since $|x \cdot y| \leq |x \cdot y \cdot s| \leq B \cdot 2^{\mathsf{len}_{\mathsf{sk}}}$, a wrap-around can occur only if

$$z_0, z_0' \in [-t/2, -t/2 + B \cdot 2^{\mathsf{len}_{\mathsf{sk}}}] \cup [t/2 - B \cdot 2^{\mathsf{len}_{\mathsf{sk}}}, t/2].$$

The probability of these events is smaller than $2B \cdot 2^{\mathsf{len}_{\mathsf{sk}}}/t < 2 \cdot 2^{-\lambda}$, so it is negligible.

Security. Let $\mathcal{A}$ be a PPT adversary. We prove security through a sequence of indistinguishable hybrids.

Hybrid 0. The initial stage corresponds to the HSS security game. So, the adversary is initially provided with the ElGamal public key, a share s_i of the private key and a PRF key. After specifying two inputs x_0 and x_1, the adversary obtains the ElGamal encryptions of x_b and $x_b \cdot s$ for some $b \in \{0, 1\}$.

Hybrid 1. This hybrid is identical to the previous one except for s_{1-i} that is now uniformly sampled in $[\ell]$ instead of $[\ell_{\mathsf{sk}}]$. By the small exponent assumption, this hybrid is indistinguishable from Hybrid 0.

Hybrid 2. In this hybrid, we substitute pk with a g^s for s uniformly sampled in $[\ell]$. This hybrid is indistinguishable from Hybrid 1, due to the statistical closeness between the distribution of pk in the two cases.

Hybrid 3. In this hybrid, we substitute the encryption of x_b with an encryption of 0. By the IND-CPA security of NIDLS ElGamal, Hybrid 2 and Hybrid 3 are indistinguishable.

Hybrid 4. In this hybrid, we substitute the encryption of $x_b \cdot s$ with an encryption of 0. By Lemma 3, this hybrid is indistinguishable from Hybrid 3.

Observe that the view of the adversary is now independent of b, so the advantage of $\mathcal{A}$ is 0. As a consequence, the advantage of $\mathcal{A}$ in the HSS security game must be negligible. □

5.3 Implementing the Setup Using One Round.

The HSS scheme described in Fig. 4 relies on a setup producing a NIDLS ElGamal public key and a subtractive secret-sharing over the integers of the private counterpart. One of the main goals of this work is to improve upon the results of [OSY21] by removing the need for trusted dealers. In this section, we therefore explain how the parties can setup the HSS material in one round. The protocol, which is formally described in Fig. 5, relies on a CRS providing the parties with the parameters of the NIDLS framework and a PRF key k. When the framework is instantiated over class groups, the generation of the CRS does not need any trusted dealer. Indeed, the parties just need to produce public, random coins and input them into the algorithm producing the CRS. In the random oracle model, this procedure can be performed non-interactively. In [OSY21], the HSS scheme was based on Paillier. Since the associated group is described by an RSA

Function $\mathcal{F}_{\mathsf{HSS\text{-}Setup}}$

1. Compute $\left(\mathsf{par}, g, \rho, \ell_{\mathsf{sk}}, \mathsf{pk}, k, (s_0, s_1)\right) \xleftarrow{R} \mathsf{Setup}(1\!\!1^\lambda)$
2. Output $(\mathsf{par}, g, \rho, \ell_{\mathsf{sk}}, \mathsf{pk}, s_i, k)$ to every party P_i.

Protocol $\Pi_{\mathsf{HSS\text{-}Setup}}$

CRS:

1. Let $\mathsf{par} := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(1\!\!1^\lambda)$ and ℓ_{sk} be the parameter for the small-exponent assumption.
2. $(g, \rho) \xleftarrow{R} \mathcal{D}(1\!\!1^\lambda, \mathsf{par})$
3. $k \xleftarrow{R} \{0,1\}^\lambda$
4. Output $(\mathsf{par}, \ell_{\mathsf{sk}}, g, \rho, k)$

Procedure:

1. Every party P_i samples $s_i \xleftarrow{R} [\ell_{\mathsf{sk}}]$
2. Every party P_i sends $\mathsf{pk}_i \leftarrow g^{s_i}$ to P_{1-i}
3. Every party P_i outputs $\mathsf{pk} \leftarrow \mathsf{pk}_1/\mathsf{pk}_0$, s_i and k.

Fig. 5. The HSS setup functionality and a one-round protocol implementing it.

modulo N where $\varphi(N)$ needs to remain secret, designing an efficient setup for the HSS scheme without relying on trusted dealers is a challenging task in that case.

Our setup protocol is very simple. Each party just generates a NIDLS ElGamal key pair, publishing the public counterpart. The parties then output the quotient between the two public keys and their respective secret key.

Theorem 2. *The protocol $\Pi_{\mathsf{HSS\text{-}Setup}}$ implements the functionality $\mathcal{F}_{\mathsf{HSS\text{-}Setup}}$ against a semi-honest adversary with perfect security.*

Proof. Suppose that P_i is corrupted. The simulator receives $(\mathsf{par}, g, \rho, \ell_{\mathsf{sk}}, \mathsf{pk}, s_i, k)$ from the functionality. It can then simulate the CRS by providing the adversary with $(\mathsf{par}, \ell_{\mathsf{sk}}, g, \rho, k)$. The view of P_i is perfectly simulated by sending s_i and $\mathsf{pk} \cdot g^{s_i}$ if $i = 0$ or g^{s_i}/pk if $i = 1$. Observe that the output of P_{1-i} is consistent with the elements sent to the adversary. □

6 Public-Key PCFs and One-Round VOLE Protocol without Trusted Setup

In [OSY21], the authors designed a one-round VOLE protocol based on the Paillier cryptosystem and the NIDLS problem on the underlying group. A VOLE

Function $\mathcal{F}_{\mathsf{VOLE}}$

INITIALISATION: The functionality waits for a value $t \in \mathbb{N}$ from the adversary.
EVALUATION: On input $x \in \mathbb{Z}_t$ from P_0 and $\boldsymbol{a} \in \mathbb{Z}_t^m$ from P_1, the functionality sends m to the adversary.

- If both parties are honest, $\mathcal{F}_{\mathsf{VOLE}}$ samples $\boldsymbol{y}_0 \xleftarrow{R} \mathbb{Z}_t^m$ and sets $\boldsymbol{y}_1 \leftarrow \boldsymbol{a} \cdot x - \boldsymbol{y}_0$. Then, it outputs $\boldsymbol{y}_i$ to P_i for every $i \in \{0, 1\}$.
- If P_i is corrupt, $\mathcal{F}_{\mathsf{VOLE}}$ waits for $\boldsymbol{y}_i \in \mathbb{Z}_t^m$ from the adversary and sets $\boldsymbol{y}_{1-i} \leftarrow \boldsymbol{a} \cdot x - \boldsymbol{y}_i$. Then, it outputs $\boldsymbol{y}_{1-i}$ to P_{1-i}.

Function $\mathcal{F}_{\mathsf{NIKE}}$

If both parties are honest, sample $k \xleftarrow{R} \{0,1\}^\lambda$ and output it to all the parties.
If one party is corrupted, wait for $k \in \{0,1\}^\lambda$ from the adversary and output it to the other party.

Fig. 6. The NIKE and vector-OLE functionalities

protocol involves two parties, the input of the first one is a element in a ring R, the input of the second party is a R-vector $\boldsymbol{a}$. The output of the protocol consists of an additive secret-sharing of the product $x \cdot \boldsymbol{a}$.

We now present a version of such protocol in the NIDLS framework (see Fig. 7). By generalising the techniques to a more abstract setting, we are able to leverage the properties of the various instantiations. In the case of class groups, that allows us to not rely on any trusted setup. In order to achieve this goal, we had to slightly modify the CRS used by the protocol. In [OSY21], the latter consisted of a pair of group elements (g, C) where $C = g^r$ for some unknown r. In order to avoid trusted setups, we now need to provide the parties with the randomness used for the generation of the CRS. Unfortunately, in class groups, such randomness would leak the value of r to the adversary, compromising security. In order to circumvent the problem, in this work, g and C are sampled independently using $\mathcal{D}(1\!\!1^\lambda)$, so with high probability $C \notin \langle g \rangle$. We prove security by relying on the DXDH assumption.

The construction makes use of a non-interactive key exchange functionality $\mathcal{F}_{\mathsf{NIKE}}$ (see Fig. 6). The latter provides the parties with a random PRF key $k \in \{0,1\}^\lambda$. When one of the parties is corrupt, the functionality lets the adversary choose k, forwarding it to the honest party. It is possible to implement $\mathcal{F}_{\mathsf{NIKE}}$ in one round using NIKE constructions such as Diffie-Hellman.

Correctness. To understand why the protocol works, observe that

$$D^{r_1^i} \cdot E^{a_i} = g^{r_0 \cdot r_1^i} \cdot f^{x \cdot a_i} \cdot C^{r_0 \cdot a_i} = f^{x \cdot a_i} \cdot A_i^{r_0}.$$

Protocol Π_{VOLE}

INPUTS: The first party P_0 has input $x \in \mathbb{Z}_t$. The other party P_1 has input $\boldsymbol{a} \in \mathbb{Z}_t^m$ for some $m \in \mathbb{N}$.
SETUP $\mathsf{Setup}(\mathbb{1}^\lambda)$:

1. $\mathsf{par} := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(\mathbb{1}^\lambda)$
2. $(g, \rho_0) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
3. $(C, \rho_1) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
4. If $g = C$, go to step 3.
5. Output $(\mathsf{par}, g, \rho_0, C, \rho_1)$

PROCEDURE:

1. The parties call $\mathcal{F}_{\mathsf{NIKE}}$ to obtain a key $k \in \{0,1\}^\lambda$.
2. $\forall i \in [m]:$ P_1 sends $A_i \leftarrow g^{r_1^i} \cdot C^{a_i}$ where $r_1^i \xleftarrow{R} [\ell]$.
3. P_0 sends $(D, E) \leftarrow (g^{r_0}, f^x \cdot C^{r_0})$ where $r_0 \xleftarrow{R} [\ell]$.
4. P_1 outputs $\boldsymbol{y}_1$ where $\boldsymbol{y}_1[i] \leftarrow \mathsf{DDLog}_{\mathsf{par}}(D^{r_1^i} \cdot E^{a_i}) + \mathsf{F}_k(i)$ for every $i \in [m]$.
5. P_0 outputs $\boldsymbol{y}_0$ where $\boldsymbol{y}_0[i] \leftarrow \mathsf{DDLog}_{\mathsf{par}}(A_i^{r_0}) - \mathsf{F}_k(i)$

Fig. 7. A one-round VOLE protocol based on the NIDLS framework.

In other words, for every index i, the elements $D^{r_1^i} \cdot E^{a_i}$ and $A_i^{-r_0}$ are a multiplicative secret-sharing of $f^{x \cdot a_i}$. Using DDLog for every $i \in [m]$, the parties are therefore able to obtain an additive secret-sharing of $x \cdot \boldsymbol{a}$ without any additional interaction.

Security. At first glance, it might seem that the security of the protocol follows from the fact that the A_i's are Pedersen commitments with respect to (g, C). However, note that the element C is *not guaranteed to be* in the cyclic group generated by g. As a consequence, A_i does not hide the input a_i with information-theoretic security and we need instead to rely on a computational assumption. The same happens also in step 3, where C plays the role of the public key in an NIDLS ElGamal encryption. However, again, C is not guaranteed to belong to $\langle g \rangle$. Therefore, we need to argue for security in a different way. To solve both issues, we use the DXDH assumption (see Definition 10).

Observe that under the DXDH assumption, g^r looks like C^s even when the randomness used for the generation of the CRS is known. As a consequence, no adversary can distinguish $A_i = g^{r_1^i} \cdot C^{a_i}$ from $C^s \cdot C^{a_i}$. The latter contains no information about a_i. The privacy of x is instead preserved as $(D, E) = (g^{r_0}, f^x \cdot C^{r_0})$ is indistinguishable, under our assumption, from $(C^s, f^x \cdot C^r)$. Since the distribution $\mathcal{D}$ outputs an element C such that $f \in \langle C \rangle$, the pair $(C^s, f^x \cdot C^r)$ hides all the information about x. The proof of the next theorem is omitted, as it easily follows from the arguments sketched above.

Theorem 3. *If the DXDH assumption holds and* F *is a secure PRF outputting pseudorandom elements in* $\mathbb{Z}_t$*, the protocol* Π_{VOLE} *UC implements the functionality* $\mathcal{F}_{\mathsf{VOLE}}$ *against a semi-honest adversary in the* $\mathcal{F}_{\mathsf{NIKE}}$*-hybrid model.*

6.1 Public-Key PCFs without Trusted Setup.

In [OSY21], Orlandi *et al.* present PCFs for vector-OLE and OT based on Paillier and the Goldwasser-Micali cryptosystem respectively (see Fig. 8 and Fig. 9). The interesting property of both constructions is that, thanks to the one-round VOLE protocol of [OSY21], the PCF keys can be set up using only one round of interaction and low-communication in the output size. For this reason, the authors introduced the notion of public-key PCF to refer to them.

On the downside, as we mentioned in the previous subsection, the one-round VOLE protocol of [OSY21] needs a trusted setup. The issue is immediately inherited by the public-key PCFs. Now, by plugging our new VOLE protocol, we obtain public-key PCFs with no need for trusted setups. We describe the resulting protocols in the following paragraphs.

Paillier, Goldwasser-Micali and the NIDLS operation. In both Paillier and the Goldwasser-Micali cryptosystem, the decryption is performed by raising the ciphertext to the private key and computing an (easy) discrete logarithm, i.e. $\mathsf{DLog}_f(\mathsf{ct}^d)$ where $f = 1 + N$ for Paillier ($t := \mathsf{ord}(f) = N$) and $f = -1$ for Goldwasser-Micali ($t := \mathsf{ord}(f) = 2$).

Now, suppose the parties have a subtractive secret-sharing of $d \cdot x$ over $\mathbb{Z}$ where x is a random element known to P_1 and the secret key d is known to P_0. We can assume that the players obtain this information as part of their PCF keys. Given a random ciphertext $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m)$, the players can locally obtain a divisive secret-sharing of $f^{m \cdot x}$ by simply raising the ciphertext ct to their respective shares of $d \cdot x$. Specifically, if $y_1 - y_0 = d \cdot x$, we have that

$$\mathsf{ct}^{y_1} = \mathsf{ct}^{y_0 + d \cdot x} = \mathsf{ct}^{y_0} \cdot f^{m \cdot x}.$$

This divisive secret-sharing can be non-interactively converted into additive shares of $m \cdot x$ over $\mathbb{Z}_t$ using the NIDLS techniques (see Section 3). Observe that P_0 also knows m as the knowledge of the private key allows it to decrypt ct.

PCFs for Vector-OLE and OT. Thanks to the technique described above, obtaining a PCF for vector-OLE becomes easy. Indeed, the parties can just sample a new ciphertext for each entry of the vector and locally apply the operations described above. In this way, P_0 will obtain a vector $\boldsymbol{a}$ and an additive share of $\boldsymbol{a} \cdot x$. The other player will obtain the other share. In order to obtain a random-looking secret-sharing, the parties rerandomise the shares using a PRF.

The PCF for OT uses similar techniques, repeating the operations λ times for each PCF sample. Specifically, let $\boldsymbol{x}$ be the random λ-bit string belonging to the OT sender. At the beginning, the parties are provided with a subtractive secret-sharing of $d \cdot x_i$ for every $i \in [\lambda]$ as part of their PCF key. In order to generate an

OT tuple, the players sample a random Goldwasser-Micali ciphertext and apply the techniques described in the previous paragraph using the secret-sharing of $d \cdot x_i$ for every $i \in [\lambda]$. In this way, P_0 obtains a random bit b and a binary share of $b \cdot \boldsymbol{x}$. The other player obtains the other share. Once again, the parties rerandomise the secret-sharing using a PRF.

Setting up the PCF Keys using One Round. The keys of the PCFs described above consist of an RSA modulo N, a PRF key and a subtractive secret-sharing of $d \cdot x$ where d is the private key corresponding to N and is known to P_0, and x is a random element known to P_1. For the vector-OLE PCF, x is uniform over $\mathbb{Z}_N$, in the case of the OT PCF, x is a binary vector in $\{0,1\}^\lambda$. While we can use $\mathcal{F}_{\mathsf{NIKE}}$ to sample the PRF key, we can allow P_0 to generate the RSA modulo N and the corresponding private key, whereas P_1 can sample x. The parties can then obtain the subtractive secret-sharing $y_1 - y_0 = d \cdot x$ by inputting d and x into the one-round vector-OLE protocol Π_{VOLE}. The RSA modulo N is finally sent by P_0 to P_1 in the only round of interaction, in parallel with the execution of Π_{VOLE}.

Observe that in the vector-OLE PCF, P_1 does not know the exact set from which x needs to be sampled. Indeed, the party will learn N only after receiving the message from P_0. Since x must be determined before the reception of this message, P_1 samples x over a set 2^λ times larger than an upper bound on N. We denote such upper bound by 2^{len_N}.

We also notice that the secret-sharings output by Π_{VOLE} are over $\mathbb{Z}_t$, while what we desire is a subtractive secret-sharing over $\mathbb{Z}$. We fix the problem by choosing $t := \mathsf{ord}(f)$ such that $t > 2^\lambda \cdot 2^{2\mathsf{len}_N} \cdot 2^{\lambda+\mathsf{len}_N}$ in the vector-OLE case, and $t > 2^\lambda \cdot 2^{\mathsf{len}_N}$ for the OT PCF. Since in both cases, t is 2^λ times bigger than $|d \cdot x|$, the probability that the difference $y_1 - y_0$ wraps around t is negligible after rerandomisation. Hence, Π_{VOLE} actually provides the parties with a subtractive secret-sharing over the integers.

On the need for the hardness of factoring. The security of both our public-key PCFs still relies on the hardness of factoring. This requirement is inherited from the original PCFs of [OSY21]. At first, it may seem possible to generalise the two constructions to the NIDLS framework, potentially obtaining public-key PCFs based on class groups only. Unfortunately, this turns out to be false.

Indeed, in the public-key PCFs for VOLE and OT, we need to non-interactively sample random ciphertexts without leaking any information about the plaintext to P_1. For Paillier, this is not a problem as any element in $\mathbb{Z}_{N^2}^\times$ is a valid encryption. For Goldwasser-Micali instead, it is sufficient to sample a random element in $\mathbb{Z}_N$ with Jacobi symbol 1. Now, if we try to move the constructions to class groups, we need to use the ElGamal cryptosystem. By modifying the PCF keys and using techniques as in the HSS scheme (see Section 5), it is still possible for the parties to non-interactively obtain an additive secret-sharing of $a \cdot x$ given the encryption of a random a. The issue is that the only known way to sample such encryption is to directly encrypt a (not every pair of elements in the class group is an ElGamal ciphertext). That would leak the value of a to P_1.

PK-PCF for VOLE

Let F and F' be PRFs outputting pseudorandom elements in $\mathbb{Z}_N$ and $\mathbb{Z}_t$ respectively. Let len_N denote the length of the Paillier modulo and let t, the order of the group used by $\mathcal{F}_{\mathsf{VOLE}}$, be greater than $2^\lambda \cdot 2^{2\mathsf{len}_N} \cdot 2^{\lambda+\mathsf{len}_N}$.
CRS: Output $k \xleftarrow{R} \{0,1\}^\lambda$.

GENERATION PROTOCOL:

1. The parties call $\mathcal{F}_{\mathsf{NIKE}}$ to obtain a key $K \in \{0,1\}^\lambda$.
2. P_0 computes $(N, d) \xleftarrow{R} \mathsf{Paillier.Gen}(1\!\!1^\lambda)$[a].
3. P_1 samples $x \xleftarrow{R} [2^{\lambda+\mathsf{len}_N}]$
4. The parties call $\mathcal{F}_{\mathsf{VOLE}}$ with inputs d (from P_0) and x (from P_1). As a result, they receive v_0 and v_1 such that $v_1 + v_0 = x \cdot d \bmod t$.
5. P_0 sends N to P_1.
6. P_0 outputs $\mathsf{k}_0 \leftarrow (N, K, y_0 := -v_0 + \mathsf{F}'_k(0), d)$.
7. P_1 outputs $\mathsf{k}_1 \leftarrow (N, K, y_1 := v_1 + \mathsf{F}'_k(0), x)$.

EVALUATION:
For a random nonce $\mathsf{ct} \in \mathbb{Z}^\times_{N^2}$:

1. P_0 computes $a \leftarrow \mathsf{Paillier.Dec}(d, \mathsf{ct})$.
2. Each P_i computes $z_i \leftarrow (-1)^{1-i} \cdot \mathsf{DDLog}_{\mathsf{Paillier}}(\mathsf{ct}^{y_i}) + \mathsf{F}_K(\mathsf{ct})$.
3. P_0 outputs (a, z_0), P_1 outputs (x, z_1).

[a] The secret key d satisfies $d \equiv 0 \bmod \varphi(N)$ and $d \equiv 1 \bmod N$.

Fig. 8. Public-key PCF for vector-OLE

7 Actively Secure Public-Key PCFs

In addition to requiring a trusted setup, the public key PCFs in [OSY21] achieve security in the semi-honest setting only. In this section, we explain how to upgrade the constructions described in Section 6 to active security, while preserving, at the same time, their round-complexity properties, namely that the parties need to speak only once. When the NIDLS framework is instantiated over class groups, the constructions do not need any trusted setup.

The particular interaction pattern limits the techniques we can rely on. For instance, we cannot perform checks that verify the correctness of the outputs, as that would require an additional round of interaction after the outputs are derived. For this reason, we develop NIZKs for our framework which might be of independent interest. We start presenting some building blocks (commitments in Section 7.1 and ZK proofs in Section 7.2). Then, we describe actively secure public-key PCFs for vector-OLE (Section 7.3) and OT (Section 7.4).

PK-PCF for OT

Let F and F' be PRFs outputting pseudorandom elements in $\{0,1\}$ and $\mathbb{Z}_t^\lambda$ respectively. Let GM be the Goldwasser-Micali cryptosystem. Assume that t, the order of the group used by $\mathcal{F}_{\mathsf{VOLE}}$, is greater than $2^\lambda \cdot 2^{\mathsf{len}_N}$.
Crs: Output $k \xleftarrow{R} \{0,1\}^\lambda$.

Generation Protocol:

1. The parties call $\mathcal{F}_{\mathsf{NIKE}}$ to obtain a key $K \in \{0,1\}^\lambda$.
2. P_0 computes $(N, d) \xleftarrow{R} \mathsf{GM.Gen}(1\!\!1^\lambda)$.
3. P_1 samples $\boldsymbol{x} \xleftarrow{R} \{0,1\}^\lambda$
4. The parties call $\mathcal{F}_{\mathsf{VOLE}}$ with inputs d (from P_0) and $\boldsymbol{x}$ (from P_1). As a result, they receive $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ such that $\boldsymbol{v}_1 + \boldsymbol{v}_0 = \boldsymbol{x} \cdot d \bmod t$.
5. P_1 sends N to P_0.
6. P_0 outputs $\mathsf{k}_0 \leftarrow (N, K, \boldsymbol{y}_0 := -\boldsymbol{v}_0 + \mathsf{F}'_k(0), d)$.
7. P_1 outputs $\mathsf{k}_1 \leftarrow (N, K, \boldsymbol{y}_1 := \boldsymbol{v}_1 + \mathsf{F}'_k(0), \boldsymbol{x})$.

Evaluation:
For a random nonce $\mathsf{ct} \in \mathbb{J}_N$[a]:

1. P_0 computes $b \leftarrow \mathsf{GM.Dec}(d, \mathsf{ct})$.
2. For each $i \in [\lambda]$, each P_j computes $z_{j,i} \leftarrow \mathsf{DDLog}_{\mathsf{GM}}(\mathsf{ct}^{y_{j,i}}) \oplus \mathsf{F}_K(\mathsf{ct}, i)$.
3. P_0 outputs $(b, \boldsymbol{z}_0)$, P_1 outputs $(\boldsymbol{x}, \boldsymbol{z}_1)$.

[a] $\mathbb{J}_N$ denotes the set of elements in $\mathbb{Z}_N$ having Jacobi symbol equal to 1.

Fig. 9. Public-key PCF for oblivious transfer

7.1 An Integer Commitment Scheme in the NIDLS Framework

Our NIZKs follow a commit-and-prove approach. Notice that in order to achieve active security, party P_0 has to prove that its input to the one-round vector-OLE protocol is the private key associated with the RSA modulo N. For this reason, we need to prove particular number-theoretic relations for which commitment schemes based on modular rings such as $\mathbb{Z}_t$ are not really suited.

Recall that, in the NIDLS framework, determining the order of the group from its parameters is assumed to be hard. This property crucially allows us to design integer commitment schemes. This fact was already noticed for class groups by Couteau *et al.* [CKLR21]. In this work, we adopt a generalisation of their construction to the NIDLS framework (see Fig. 10), basing however its security on the DXDH assumption. As we discuss at the end of this section, despite the claims in [CKLR21], their construction is not compatible with trustless setup.

Integer commitment scheme

CRS:

1. $\mathsf{par}' := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(1\!\!1^\lambda)$
2. $(g, \rho_0) \xleftarrow{R} \mathcal{D}(1\!\!1^\lambda, \mathsf{par})$
3. $(C, \rho_1) \xleftarrow{R} \mathcal{D}(1\!\!1^\lambda, \mathsf{par})$
4. If $C = g$, go to step 3.
5. Output $\mathsf{par} := (\mathsf{par}', g, \rho_0, C, \rho_1)$

COMMITMENT: $\mathsf{Commit}(\mathsf{par}, x)$

1. $r \xleftarrow{R} [\ell]$
2. Output the commitment $X \leftarrow C^x \cdot g^r$ and the opening information r.

VERIFICATION: $\mathsf{Verify}(\mathsf{par}, X, x, r)$

1. If $X = C^x \cdot g^r$ output 1 otherwise output 0.

Fig. 10. Integer commitment scheme in the NIDLS framework.

Theorem 4. *If the DXDH assumption and the weak hidden order assumption hold, the construction in Fig. 10 is a hiding and binding integer commitment scheme. Moreover, the scheme is linearly homomorphic.*

Proof. It is straightforward to see that the construction is correct and linearly homomorphic.

Binding. The proof for binding is made interesting by the fact that C is not (necessarily) an element in the group generated by g. Suppose that we have an adversary that breaks binding e.g., after being provided with the parameters, the adversary returns (x, r) and (y, s) with $x \neq y$ such that $C^x g^r = C^y g^s$, and therefore $C^{x-y} = g^{s-r}$. Let $\alpha := x - y \neq 0$ and $\beta = s - r$. Since the order of the group is unknown we cannot invert these elements. Instead, we resort to the DXDH assumption, which implies the following claim:

Claim. Assume there exists an adversary $\mathcal{A}$ that, on input (g, C), returns (α, β) with $C^\alpha = g^\beta$ and $\alpha \neq 0$. Then, with overwhelming probability over $u, v \xleftarrow{R} [\ell]$, it holds that:

$$(g^u)^\alpha = (g^v)^\beta \tag{1}$$

Proof (of claim). The reduction is given a DXDH tuple (g, C, g^u, T) where T is either C^u or g^v for random $u, v \xleftarrow{R} [\ell]$, and feeds (g, C) to $\mathcal{A}$. Now the reduction concludes that $T = C^u$ when

$$(T)^\alpha = (g^v)^\beta \tag{2}$$

or $T = g^v$ otherwise. Note that if $T = C^u$ then Equation 2 is trivially true. Thus if $(g^u)^\alpha \neq (g^v)^\beta$ the reduction correctly distinguished between DXDH tuple and non-DXDH tuples. □

We now go back to the proof of the binding property and argue that under the weak hidden order assumption, no adversary can output α, β such that Equation 1 holds for random (u, v). We first rewrite $(g^u)^\alpha = (g^v)^\beta$ as $u \cdot \alpha \equiv v \cdot \beta \bmod \mathsf{ord}(g)$. We argue the following:

Claim. Let α, β be such that

$$u \cdot \alpha \equiv v \cdot \beta \bmod \mathsf{ord}(g)$$

with overwhelming probability for uniform $u, v \xleftarrow{R} [\ell]$. Then $\mathsf{ord}(g) | \alpha$.

Proof (of claim). For the sake of contradiction assume that this is not the case, e.g., $\mathsf{ord}(g) \nmid \alpha$. Then there is a non-negligible probability that $u \cdot \alpha \not\equiv v \cdot \beta \bmod \mathsf{ord}(g)$. Indeed, let p be a prime that divides $\mathsf{ord}(g)$ but not α, it must hold that $u \equiv v \cdot \beta \cdot \alpha^{-1} \bmod p$. This happens with probability $1/p < 1/2$, so it must be that α is a multiple of $\mathsf{ord}(g)$. □

We have reached a contradiction. Indeed, under the weak hidden order assumption, no adversary can output α such that $g^\alpha = 1$.

Hiding. We show that no adversary can distinguish a commitment to x_0 from a commitment to x_1. Indeed by the DXDH assumption, (g, C, g^r, C^r) with r uniform in $[\ell]$, is indistinguishable from (g, C, C^s, C^r) with s again uniform in $[\ell]$. Thus $C^{x_b} \cdot g^r$ is indistinguishable from $C^{x_b} \cdot C^s$. From the way ℓ is chosen, C^s is statistically close to the uniform distribution over $\langle C \rangle$. So, as commitments to both x_0, x_1 are indistinguishable from random elements in $\langle C \rangle$, no adversary can distinguish between a commitment to x_0 and a commitment to x_1. □

In [CKLR21], the authors proved the security of this commitment scheme in the class group setting by relying on the *subgroup indistinguishability* assumption. The latter states that no PPT adversary can distinguish between a pair of random elements (g, C) both sampled according to $\mathcal{D}$ and a pair (g, g^s) where s is uniform over $[\ell]$. Despite what the authors claim, this assumption is not sufficient to prove security when we do not rely on a trusted dealer for the generation of the CRS. Indeed, in order to remove trusted setups, we need to provide the parties with the random coins used for the generation of the CRS. That prevents us from substituting C with g^s in the security proofs. The reason is that the distribution $\mathcal{D}$ is, surprisingly, not invertible over class groups. Specifically, given $C \in \mathsf{Supp}(\mathcal{D})$, it is hard to find a bit string r such that $\mathcal{D}(1\!\!1^\lambda; r) = C$.

7.2 Zero-Knowledge Proofs in the NIDLS Framework

We describe how to build useful ZK-proofs in the NIDLS framework such as: range proofs, multiplication proofs, proofs of knowledge of openings and proofs of commitment to the plaintext. In particular, we build sigma protocols that use

the NIDLS framework in a black-box way, independently of its instantiation. Thus, our proofs do not need to express operations in the NIDLS framework as circuits. Since these tools are all based on fairly standard techniques, we will only give a brief overview and direct the reader to Appendix A for more details.

Proof of knowledge of openings. Π_{com} allows to convince a verifier holding a commitment X that the prover knows integers x and r such that $X = C^x \cdot g^r$.

Compared to standard Σ-protocols for proving knowledge of a (Pedersen) commitment in a prime order group, we need two major changes: First, all the computation between scalars is done over the integers (since the order of the group is unknown) and therefore, the random strings chosen in the first round must be larger than an upper bound on the witness (x, r). Second, we can only use binary challenges: this is due to the fact that, again, the order of the group is unknown and therefore, we cannot invert the challenge when extracting the witness in the special soundness property. Thus, we need to repeat the proofs λ times. Note that for most of our instantiations there usually are ways around this issue, mostly relying on instantiation-dependent assumptions (such as the strong root problem and the low order assumption for class groups). However, those do not carry over to our general framework.

Multiplication proofs. Π_{mult} allows to convince a verifier with commitments X, Y and Z, that the prover knows $x, y, z \in \mathbb{Z}$ and $r_1, r_2, r_3 \in \mathbb{Z}$ such that $X = C^x \cdot g^{r_1}$, $Y = C^y \cdot g^{r_2}$, $Z = C^z \cdot g^{r_3}$ and $z = x \cdot y$. We construct Π_{mult} by adapting the protocol of [DF02] to our framework, similarly to what we did for Π_{com}.

Range proofs. Π_{range} allows to convince a verifier holding a commitment X and a bound $B \in \mathbb{N}$ that the prover knows $x, r \in \mathbb{Z}$ such that $x \in [0, B]$ and $X = C^x \cdot g^r$. Our protocol is based on a technique by Groth [Gro05], who observed that

$$x \in [0, B] \quad \Longleftrightarrow \quad \exists x_1, x_2, x_3 \in \mathbb{Z} \quad \text{s.t. } 1 + 4x \cdot (B - x) = x_1^2 + x_2^2 + x_3^2.$$

The protocol can be therefore constructed exploiting multiplication proofs just introduced and the linearity of the commitment.

We remark that in [CKLR21], the authors designed a range proof for our commitment scheme in the class group setting. Their solution never relies on binary challenges, so its efficiency is better by a factor of λ. However, their construction is only proven secure when the CRS is generated by a trusted dealer. This is due to the issue described at the end of Section 7.1.

Proof of commitment to the plaintext. Π_{plain} can be used to convince a verifier holding group elements D, E, X that the prover knows $x, r, s \in \mathbb{Z}$ such that $X = C^x \cdot g^s$, $D = g^r$ and $E = f^x \cdot C^r$. The protocol uses standard techniques adapted to our framework as sketched for Π_{com}.

7.3 Actively Secure Public-Key PCF for Vector-OLE

In the semi-honest public-key PCF for vector-OLE (Fig. 8 and Fig. 7), the only message sent by party P_0 consists of an RSA modulo N and a pair of groups

elements D, E where $D = g^{r_0}$ and $E = f^d \cdot C^{r_0}$. Here, the exponent d represents the Paillier private key associated with the RSA modulo N, whereas g and C are groups elements described in the CRS. We recall that d is the only element in $[0, N \cdot \varphi(N) - 1]$ satisfying $d \equiv 0 \bmod \varphi(N)$ and $d \equiv 1 \bmod N$.

The only message sent by party P_1 is instead $A := C^x \cdot g^{r_1}$. In order for the construction to be correct, the value of x needs to be smaller than $2^\lambda \cdot 2^{\mathsf{len}_N}$.

An active adversary can always deviate from the protocol and send malformed material. For this reason, it is fundamental that our NIZKs prove the well-formedness of the messages of the parties. In the case of P_1, the task is rather simple. Using the Fiat-Shamir heuristic, we can indeed convert Π_{range} into the NIZK we are looking for. Proving the well-formedness of P_0's message is however more challenging.

Proving the well-formedness of P_0's message. As usual we first design a public coin honest-verifier zero-knowledge proof and then convert it into a NIZK by applying the Fiat-Shamir heuristic. Our protocol makes use of a public-coin HVZK $\Pi_{\mathsf{semiprime}}$ for proving that the RSA modulo N is the product of two distinct primes p and q. Moreover, $\Pi_{\mathsf{semiprime}}$ proves that $\gcd\big(N, \varphi(N)\big) = 1$. Such protocol can be found e.g., in [GRSB19]. The main idea of our protocol is as follows: the prover commits to d, the primes p and q and integers k_1 and k_2 satisfying $d = k_1 \cdot \varphi(N)$ and $d = k_2 \cdot N + 1$. We denote the five commitments by Z, X_1, X_2, Y_1 and Y_2 respectively. The parties run $\Pi_{\mathsf{semiprime}}$ to verify that N is semiprime. By relying on Π_{mult}, the prover also shows that X_1 and X_2 are commitments to a factorisation of N. Furthermore, using Π_{range}, the verifier checks that the value committed in X_1 belongs to $[2, N-1]$ (this is done by showing that $C^{-2} \cdot X_1$ is a commitment to a value in $[0, N-3]$). In this way, it is sure that the prover committed to a proper factorisation and not just $N \cdot 1$. Now, the verifier is also certain that $W := C^N \cdot X_1^{-1} \cdot X_2^{-1} \cdot C$ is a commitment to $N - p - q + 1 = \varphi(N)$. Next, using Π_{range}, the prover shows that the value committed in Y_1 belongs to $[0, N-1]$. Using Π_{com}, it also proves the knowledge of opening for Y_2. The verifier also checks that Y_1 is a commitment to $d/\varphi(N)$. This is done by running Π_{mult} on Y_1, W and Z. If the check passes, the verifier is also sure that the value committed in Z belongs to $[0, N \cdot \varphi(N) - 1]$. In the end, the prover shows that Y_2 is a commitment to $(d-1)/N$ by proving that $Z \cdot C^{-1} \cdot Y_2^{-N}$ opens to 0. Finally, the prover uses Π_{plain} to convince the verifier that the values hidden in Z and in (D, E) coincide. The formal description of the protocol can be found in Fig. 11.

Theorem 5. *Let $\Pi_{\mathsf{semiprime}}$ be a honest-verifier zero-knowledge public-coin proof proving that N is the product of two distinct primes and $\gcd\big(N, \varphi(N)\big) = 1$. If the commitment scheme in Fig. 10 is hiding and binding, the construction*

Proof of encryption of Paillier private key Π_{Paillier}

CRS:

1. $\mathsf{par}' := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(\mathbb{1}^\lambda)$
2. $(g, \rho_0) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
3. $(C, \rho_1) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
4. If $C = g$, go to step 3.
5. Output $\mathsf{par} := (\mathsf{par}', g, \rho_0, C, \rho_1)$

PROTOCOL:

1. The prover computes $k_1 \leftarrow d/\varphi(N)$ and $k_2 \leftarrow (d-1)/N$.
2. The prover samples $s_1, s_2, s_3, s_4 \xleftarrow{R} [\ell]$ and $s \xleftarrow{R} [2^\lambda \cdot N \cdot \ell]$.
3. The prover sends $Z \leftarrow C^d \cdot g^s$, $X_1 \leftarrow C^p \cdot g^{s_1}$, $X_2 \leftarrow C^q \cdot g^{s_2}$, $Y_1 \leftarrow C^{k_1} \cdot g^{s_3}$ and $Y_2 \leftarrow C^{k_2} \cdot g^{s_4}$.
4. The parties run $\Pi_{\mathsf{semiprime}}$ with input N and witness (p, q).
5. The parties set $B \leftarrow N - 1$.
6. The parties run Π_{mult} with input X_1, X_2, C^N. The witness is $(p, q, N, s_1, s_2, 0)$.
7. The parties run Π_{range} with input $C^{-2} \cdot X_1$ and bound $B - 2$. The witness is $(p - 2, s_1)$.
8. The parties run Π_{range} with input Y_1 and bound B. The witness is (k_1, s_3).
9. The parties run Π_{com} with input Y_2. The witness is (k_2, s_4).
10. The parties set $W \leftarrow C^N \cdot X_1^{-1} \cdot X_2^{-1} \cdot C$.
11. The parties run Π_{mult} with input Y_1, W, Z. The witness is $(k_1, \varphi(N), d, s_3, -s_1 - s_2, s)$.
12. The prover sends $v \leftarrow s - s_4 \cdot N$.
13. The parties run Π_{plain} with input Z, D, E and witness (d, r, s).
14. The verifier accepts if all the above proofs succeed and $Z \cdot C^{-1} \cdot Y_2^{-N} = g^v$.

Fig. 11. Honest-verifier zero-knowledge proof of encryption of the Paillier private key

Π_{Paillier} in Fig. 11 is a complete, special-sound public-coin proof for the relation

$$\mathcal{R}_{\mathsf{Paillier}} := \left\{ (D, E, N), (d, p, q, r) \middle| \begin{array}{l} N = p \cdot q, \text{ where } p, q \text{ are positive primes} \\ \gcd\big(N, \varphi(N)\big) = 1 \\ D = g^r, E = f^d \cdot C^r \\ d \equiv 0 \bmod \varphi(N) \\ d \equiv 1 \bmod N \\ 0 \le d < N \cdot \varphi(N) \end{array} \right\}$$

Moreover, when $r \in [\ell]$, the proof is honest-verifier zero-knowledge.

Proof. Completeness. Completeness of the protocol follows by inspection. Note in particular that all subprotocols ($\Pi_{\mathsf{semiprime}}$, Π_{range}, Π_{com}, Π_{plain}) are complete

and that the input to Π_{mult} is a valid instance e.g.,

$$W = C^N \cdot X_1^{-1} \cdot X_2^{-1} \cdot C = C^{N-p-q+1} \cdot g^{-s_1-s_2} = C^{\varphi(N)} \cdot g^{-s_1-s_2}.$$

Thus the verifier can reject only if $Z \cdot C^{-1} \cdot Y_2^{-N} \neq g^v$ but

$$Z \cdot C^{-1} \cdot Y_2^{-N} = C^d \cdot g^s \cdot C^{-1} \cdot C^{-k_2 \cdot N} \cdot g^{-N \cdot s_4} = C^{d-1-k_2 \cdot N} \cdot g^{s-N \cdot s_4} = g^v.$$

Special Soundness. By the special soundness of Π_{range}, the extractor is able to retrieve $p' \in [0, N-3]$, $k_1 \in [0, N-1]$ and $s_1', s_3 \in \mathbb{Z}$ such that $C^{-2} \cdot X_1 = C^{p'} \cdot g^{s_1'}$ and $Y_1 = C^{k_1} \cdot g^{s_3}$. Moreover, by the special soundness of Π_{com}, the extractor is also able to extract $k_2, s_4 \in \mathbb{Z}$ such that $Y_2 = C^{k_2} \cdot g^{s_4}$.

By the special soundness of Π_{mult}, the extractor is also able to obtain values $p, q, N' \in \mathbb{Z}$ and $s_0', s_1, s_2 \in \mathbb{Z}$ such that $N' = p \cdot q$, $X_1 = C^p \cdot g^{s_1}$, $X_2 = C^q \cdot g^{s_2}$ and $C^N = C^{N'} \cdot g^{s_0'}$. By the binding properties of the commitment scheme in Fig. 10, it must be $N = N'$. We conclude that $N = p \cdot q$. Since $X_1 = C^{p'+2} \cdot g^{s_1'}$, by the binding properties of the commitment scheme, we have that $p = p' + 2$ and so $p \in [2, N-1]$. In this way, we are certain that $p \cdot q$ is a proper factorisation. By the soundness of $\Pi_{\mathsf{semiprime}}$, we know that N is a product of two distinct primes. Therefore, p and q must be prime.

Again, by the special soundness of Π_{mult}, the extractor is also able to obtain values $k_1', \varphi', d' \in \mathbb{Z}$ and $s_3', s_5', s' \in \mathbb{Z}$ such that $d' = k_1 \cdot \varphi'$, $Y_1 = C^{k_1'} \cdot g^{s_3'}$, $W = C^{\varphi'} \cdot g^{s_5'}$ and $Z = C^{d'} \cdot g^{s'}$. By the binding properties of the commitment scheme in Fig. 10, it must be that $k_1 = k_1'$. Moreover, we know that

$$W = C^N \cdot X_1^{-1} \cdot X_2^{-1} \cdot C = C^N \cdot C^{-p} \cdot g^{-s_1} \cdot C^{-q} \cdot g^{-s_2} \cdot C = C^{N-p-q+1} \cdot g^{-s_1-s_2}.$$

So, again, by the binding properties of the commitment scheme, we have that $\varphi(N) = N - p - q + 1 = \varphi'$. We understand also that $d' \equiv 0 \bmod \varphi(N)$ and $0 \leq d' < N \cdot \varphi(N)$.

Observe that

$$g^v = Z \cdot C^{-1} \cdot Y_2^{-N} = C^{d'} \cdot g^{s'} \cdot C^{-1} \cdot C^{-N \cdot k_2} \cdot g^{-N \cdot s_4} = C^{d'-N \cdot k_2 - 1} \cdot g^{s-N \cdot s_4}.$$

By the binding properties of the commitment scheme, it must be that $d' - N \cdot k_2 - 1 = 0$, so $d' \equiv 1 \bmod N$.

Finally, by the special soundness of Π_{plain}, the extractor is able to retrieve $d, r, s \in \mathbb{Z}$ such that $Z = C^d \cdot g^s$, $D = g^r$, $E = f^d \cdot C^r$. Once again, by the binding properties of the commitment scheme in Fig. 10, it must be that $d = d'$.

Honest-Verifier Zero-Knowledge. Suppose that $r \in [\ell]$. We proceed by a series of hybrids.

Hybrid 1. In this hybrid, the simulator generates the transcripts in $\Pi_{\mathsf{semiprime}}$, Π_{range}, Π_{com}, Π_{mult} and Π_{plain} using the corresponding honest verifier simulators. Since $p, q \in [0, B]$, $d \in [0, N \cdot \varphi(N) - 1]$ and $r \in [\ell]$, by the honest-verifier zero-knowledge of $\Pi_{\mathsf{semiprime}}$, Π_{range}, Π_{com}, Π_{mult} and Π_{plain}, no adversary can distinguish Hybrid 1 from the real protocol.

Hybrid 2. In this hybrid, the simulator samples v uniformly over $[2^\lambda \cdot N \cdot \ell]$ and sets $Z \leftarrow C \cdot Y_2^N \cdot g^v$. The only difference between this hybrid and the previous one is the distribution of v. The statistical distance is however dominated by $2^{-\lambda}$, which is negligible. So, no adversary is able to distinguish between Hybrid 1 and Hybrid 2.

Hybrid 3. In this hybrid, the simulator sends $X_1 \leftarrow g^{s_1}$, $X_2 \leftarrow g^{s_2}$, $Y_1 \leftarrow g^{s_3}$ and $Y_4 \leftarrow g^{s_4}$ instead of $C^p \cdot g^{s_1}$, $C^q \cdot g^{s_2}$, $C^{k_1} \cdot g^{s_3}$ and $C^{k_2} \cdot g^{s_4}$. The rest of the transcript is generated as in Hybrid 2. By the hiding properties of the commitment scheme in Fig. 10, no adversary can distinguish between Hybrid 2 and Hybrid 3.

Observe that now the transcript can be generated without knowing the witness. □

Deploying the NIZKs to obtain active security. We can finally present our active public key PCF for vector-OLE. The construction, called $\Pi^{\mathsf{Active}}_{\mathsf{VOLE}}$, is described in Fig. 12.

We prove that the pk-PCF protocol implements the random vector-OLE functionality $\mathcal{F}_{\mathsf{r\text{-}VOLE}}$ (see Fig. 13) in the UC model. $\mathcal{F}_{\mathsf{r\text{-}VOLE}}$ is a functionality that, during the initialisation, samples a random RSA modulo N and a value $x \in \mathbb{Z}_N$, which outputs to P_1. Upon any request for a vector-OLE tuple, the functionality samples a random $a \in \mathbb{Z}_N$ and computes a subtractive secret-sharing of $z_1 - z_0 = a \cdot x$ over $\mathbb{Z}_N$. Then, $\mathcal{F}_{\mathsf{r\text{-}VOLE}}$ outputs (a, z_0) to P_0 and z_1 to P_1. If one of the parties is corrupted, the functionality let the adversary choose the outputs of the corrupt player, then it samples the outputs of the honest party at random conditioned on $z_1 = z_0 + a \cdot x$. Moreover, if P_0 is corrupted, the functionality lets the adversary select the RSA modulo N. When P_1 is corrupt, instead, $\mathcal{F}_{\mathsf{r\text{-}VOLE}}$ lets the adversary choose x after providing it with N.

The resources. The protocol $\Pi^{\mathsf{Active}}_{\mathsf{VOLE}}$ relies on the non-interactive key-exchange functionality $\mathcal{F}_{\mathsf{NIKE}}$ (see Fig. 6) and a ZK functionality $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ (see Fig. 14). The former provides the parties with a random PRF key $k \in \{0,1\}^\lambda$. When one of the parties is corrupt, the functionality lets the adversary choose k, forwarding it to the honest party. It is possible to implement $\mathcal{F}_{\mathsf{NIKE}}$ in one round using NIKE constructions such as Diffie-Hellman, augmenting them with NIZKs to achieve security against an active adversary.

The functionality $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ is instead used to prove statements for a fixed set of NP relations. We can assume that this set includes range proofs and $\mathcal{R}_{\mathsf{Paillier}}$. Upon initialisation, $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ outputs the parameters of the NIDLS framework, including the random coins used for their generation. When $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ is provided with a statement x for one of the supported NP relations, the functionality waits for the prover to provide the corresponding witness w. If the verification fails, $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ outputs 0 to both parties, otherwise, it outputs 1. The functionality $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ is also equipped with a different predicate for each supported NP-relation. Such predicate makes sure that the witness satisfies the properties for zero-knowledge. If that is not the case, the w is leaked to the adversary.

Active PK-PCF for VOLE $\Pi^{\mathsf{Active}}_{\mathsf{VOLE}}$

Let F be a PRF. Let len_N denote the length of the Paillier modulo and let t, the order of the NIDLS group, be greater than $2^{\lambda} \cdot 2^{2\mathsf{len}_N} \cdot 2^{\lambda+\mathsf{len}_N}$.
INITIALISATION:

1. The parties initialise $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ obtaining $\mathsf{par} := (\mathsf{par}', g, \rho_0, C, \rho_1)$.
2. The parties call $\mathcal{F}_{\mathsf{NIKE}}$ to obtain a PRF key k.
3. P_0 computes $(N, d) \xleftarrow{R} \mathsf{Paillier.Gen}(1\!\!1^{\lambda})$ where $N = p \cdot q$.
4. P_1 samples $x \xleftarrow{R} [B]$ where $B := 2^{\lambda+\mathsf{len}_N}$.
5. P_0 samples $r_0 \xleftarrow{R} [\ell]$ and sets $D \leftarrow g^{r_0}$, $E \leftarrow f^{d} \cdot C^{r_0}$.
6. P_0 sends N, D, E.
7. P_1 samples $r_1 \xleftarrow{R} [\ell]$ and computes $A \leftarrow C^{x} \cdot g^{r_1}$
8. P_1 sends A.
9. The parties call $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ with input $(\mathsf{Paillier}, D, E, N)$. P_0 inputs also (p, q, r_0). The parties abort if the functionality outputs 0 or if $N > 2^{\mathsf{len}_N}$.
10. The parties call $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ with input (range, A, B). P_1 inputs also (x, r_1). The parties abort if the functionality outputs 0.
11. The parties query (A, D, E, N) to the random oracle and obtain a random $u \in \mathbb{Z}_t$ as a reply.
12. P_0 computes $v_0 \leftarrow \mathsf{DDLog}_{\mathsf{par}}(A^{r_0}) - u \bmod t$
13. P_1 computes $v_1 \leftarrow \mathsf{DDLog}_{\mathsf{par}}(D^{r_1} \cdot E^{x}) + u \bmod t$
14. P_0 stores $\mathsf{k}_0 \leftarrow (N, k, y_0 := -v_0, d)$.
15. P_1 stores $\mathsf{k}_1 \leftarrow (N, k, y_1 := v_1, x \bmod N)$.

EVALUATION: Query the label id to the oracle. Let $\mathsf{ct} \in \mathbb{Z}^{\times}_{N^2}$ be the response:

1. P_0 computes $a \leftarrow \mathsf{Paillier.Dec}(d, \mathsf{ct}) \bmod N$.
2. Each P_i computes $z_i \leftarrow (-1)^{1-i} \cdot \mathsf{DDLog}_{\mathsf{Paillier}}(\mathsf{ct}^{y_i}) + \mathsf{F}_k(\mathsf{ct}) \bmod N$.
3. P_0 outputs (a, z_0), P_1 outputs $(x \bmod N, z_1)$.

Fig. 12. Active public-key PCF for vector-OLE

Note that Fiat-Shamir NIZKs, including the ones we designed, do not implement the functionality $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ in the UC model. Indeed, in order to extract the witness w, we need to rewind the adversary and this operation is incompatible with UC. Using Fiat-Shamir NIZKs to implement $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ is, however, a common practice, which is considered secure. Moreover, the resulting protocols can be proven secure in weaker models that allow sequential composability only. Finally, using standard techniques [DP92], it is still possible to adapt our NIZKs so that they implement $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ in the UC model.

Theorem 6. *Let $\mathsf{len}_N(\lambda)$ be the length of the RSA modulo and assume that $t > 2^{2\lambda+3\mathsf{len}_N}$. Let F be a secure PRF outputting pseudorandom elements in $[2^{\lambda+\mathsf{len}_N}]$. If the DXDH assumption holds, the protocol $\Pi^{\mathsf{Active}}_{\mathsf{VOLE}}$ UC-implements*

Function $\mathcal{F}_{\text{r-VOLE}}$

INITIALISATION:

- If both parties are honest, generate $(N, p, q) \xleftarrow{R} \mathsf{Paillier.Gen}(1\!\!1^\lambda)$ and sample $x \xleftarrow{R} \mathbb{Z}_N$.
- If P_0 is corrupt, wait for N from the adversary and sample $x \xleftarrow{R} \mathbb{Z}_N$. If the adversary sends $\perp$, abort.
- If P_1 is corrupt, generate $(N, p, q) \xleftarrow{R} \mathsf{Paillier.Gen}(1\!\!1^\lambda)$, send N to the adversary to the adversary and wait for $x \in \mathbb{Z}_N$ as a reply. If the adversary sends $\perp$, abort.

EVALUATION: On input a fresh label id from an honest party P_i.

- If both parties are honest, the functionality samples $a, z_0 \xleftarrow{R} \mathbb{Z}_N$ and sets $z_1 \leftarrow a \cdot x - z_0$. Then, it sets $R_0 \leftarrow (a, z_0)$ and $R_1 \leftarrow (x, z_1)$. $\mathcal{F}_{\text{r-VOLE}}$ outputs R_i to P_i and stores $(\mathsf{id}, 1-i, R_{1-i})$.
- If $i = 1$ and P_0 is corrupted, the functionality waits for $a, z_0 \in \mathbb{Z}_N$ from the adversary and sets $z_1 \leftarrow a \cdot x - z_0$. Then, it outputs (x, z_1) to P_i.
- If $i = 0$ and P_1 is corrupted, the functionality waits for $z_1 \in \mathbb{Z}_N$ from the adversary, samples $a \xleftarrow{R} \mathbb{Z}_N$ and computes $z_0 \leftarrow a \cdot x - z_1$. Then, it outputs (a, z_0) to P_i.

If id is not fresh, retrieve the triple (id, i, R_i) and output R_i to P_i.

Fig. 13. The random vector-OLE functionality

the functionality $\mathcal{F}_{\text{r-VOLE}}$ against an active adversary in the $(\mathcal{F}_{\text{NIDLS-ZK}}, \mathcal{F}_{\text{NIKE}})$-hybrid model with random oracle.

Proof. We start by proving the correctness of the protocol.

Claim. If the protocol does not abort, we have that $y_1 = y_0 + d \cdot x$. Here, the operations are computed over the integers and the terms y_0, y_1, d and x are computed as in the protocol using the witnesses input into $\mathcal{F}_{\text{NIDLS-ZK}}$.

Proof of the claim. Notice that

$$D^{r_1} \cdot E^x = (g^{r_0})^{r_1} \cdot (f^d \cdot C^{r_0})^x = f^{x \cdot d} \cdot (C^x \cdot g^{r_1})^{r_0} = f^{x \cdot d} \cdot A^{r_0}.$$

We conclude that $v_0 + v_1 = x \cdot d \bmod t$ and so $y_1 = y_0 + d \cdot x \bmod t$. Now, observe that, due to the rerandomisation with the random oracle, y_0 is uniformly distributed over $\mathbb{Z}_t$. Since $t > 2^\lambda \cdot 2^{2\mathsf{len}_N} \cdot 2^{\lambda + \mathsf{len}_N} \geq 2^\lambda \cdot d \cdot x$, the probability that $y_0 \in [t - d \cdot x, t]$ is smaller than $2^{-\lambda}$, which is negligible. So, with overwhelming probability, there are no wrap-arounds and $y_1 = y_0 + d \cdot x$ even if the operations are performed over the integers. Notice that it is fundamental that the query to

Function $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$

Let $\mathcal{U}$ be a finite set of NP relations. Let P_{L} be a predicate corresponding to the relation $\mathcal{R}_{\mathsf{L}} \in \mathcal{U}$.

INITIALISATION:

1. $\mathsf{par}' := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(\mathbb{1}^\lambda)$
2. $(g, \rho_0) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
3. $(C, \rho_1) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
4. If $C = g$, go to step 3.
5. Output $\mathsf{par} := (\mathsf{par}', g, \rho_0, C, \rho_1)$ to all the parties.

VERIFY:

On input an NP relation $\mathcal{R}_{\mathsf{L}} \in \mathcal{U}$ and a statement st from both parties and a witness w from only one of the parties, $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ checks whether $(\mathsf{st}, w) \in \mathcal{R}_{\mathsf{L}}$. If that is the case, $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ outputs 1 to all the parties, otherwise it outputs 0. If $P_{\mathsf{L}}(\mathsf{par}, w) = 0$, the functionality leaks w to the adversary.

Fig. 14. The NIDLS ZK functionality

the oracle includes A, D, E and N, so the adversary cannot pick the messages of the corrupted parties as a function of the random oracle output, as this could allow them to cause wrap-arounds and compromise the correctness of the result. □

Claim. If the protocol does not abort, we have that $z_1 = z_0 + a \cdot x \bmod N$ for every execution of Evaluation.

Proof of the claim. Even if N is not the product of two safe-primes, the NIDLS techniques are still working. Specifically, since $N = p \cdot q$ and $p \neq q$ (indeed, $\gcd(N, \varphi(N)) = 1$), the order of $\mathbb{Z}_{N^2}^\times$ is still $N \cdot \varphi(N)$. Moreover, $1 + N$ has still order N and we can efficiently compute y from $c := (1+N)^y = 1 + yN$ as $y = (c-1)/N$. It is also still possible to compute canonical representatives of $\langle 1+N \rangle$-cosets. Indeed, take $r \in \mathbb{Z}_{N^2}^\times$ and rewrite it as $r_0 + r_1 \cdot N$ where $r_0, r_1 \in \mathbb{Z}_N$. Notice that $r \in \mathbb{Z}_{N^2}^\times$, implies that $r_0 \in \mathbb{Z}_N^\times$. We have that $r_0 = (1+N)^y \cdot r \bmod N$ for every $y \in \mathbb{Z}_N$. Moreover, $r = r_0 \cdot (1+N)^y$ for $y = r_1/r_0 \bmod N$.

Now, let $\mathsf{ct} = (1+N)^a \cdot r^N$ for some $r \in \mathbb{Z}_{N^2}^\times$. We have that

$$\begin{aligned} \mathsf{ct}^{y_1} &= \mathsf{ct}^{y_0 + x \cdot d} = \mathsf{ct}^{y_0} \cdot (1+N)^{a \cdot x \cdot d} \cdot r^{N \cdot d \cdot x} = \\ &= \mathsf{ct}^{y_0} \cdot (1+N)^{a \cdot x \cdot (1 + k_2 \cdot N)} \cdot r^{N \cdot k_1 \cdot \varphi(N) \cdot x} = \mathsf{ct}^{y_0} \cdot (1+N)^{a \cdot x}. \end{aligned}$$

So, after applying DDLog, the parties obtain z_0 and z_1 that satisfy $z_1 - z_0 = a \cdot x \bmod N$. □

All the parties are honest. Consider the following series of indistinguishable hybrids.

Hybrid 1. In this hybrid, the simulator sets $z_1 \leftarrow z_0 + a \cdot x \bmod N$ for every execution of $\mathsf{Evaluation}$. By the previous claim, this hybrid is indistinguishable from the real protocol.

Hybrid 2. Consider the hybrid in which the simulator sets $A \leftarrow g^{r_1}$ instead of $A \leftarrow C^x \cdot g^{r_1}$. This hybrid is indistinguishable from the previous one by the hiding properties of the commitment scheme in Fig. 10.

Hybrid 3. In this hybrid, instead of generating E as in the protocol, the simulator sets $E \leftarrow f^d \cdot g^{r'_0}$ where r'_0 is uniform over $[\ell]$ and independent of r_0. Observe that this stage is indistinguishable from the previous one by the DXDH assumption.

Hybrid 4. In this hybrid, the simulator sets $E \leftarrow f^s \cdot g^{r'_0}$ where s is uniformly sampled over $[t]$. Observe that this stage is indistinguishable from the previous one as $g^{r'_0}$ is close to uniform over g and $f \in \langle g \rangle$.

Hybrid 5. In this hybrid, the simulator samples z_0 uniformly over $\mathbb{Z}_N$. Since $2^{\lambda + \mathsf{len}_N} \geq 2^\lambda \cdot N$ and by the security of the PRF F, the adversary cannot distinguish between this hybrid and the previous one.

Hybrid 6. In this hybrid, instead of sampling x uniformly over $[2^\lambda \cdot 2^{\mathsf{len}_N}]$, the simulator samples it uniformly over $\mathbb{Z}_N$. Observe that this stage is unconditionally indistinguishable from the previous one. Indeed, the statistical distance is dominated by $2^{-\lambda}$, which is negligible.

Hybrid 7. In this hybrid, the simulator lets the functionality $\mathcal{F}_{\mathsf{r\text{-}VOLE}}$ generate the outputs and the RSA modulo. Specifically, the simulator receives N from the functionality. Moreover, it replies to all random oracle queries by sending random Paillier encryptions of 0. This hybrid is indistinguishable from the previous one by the IND-CPA security of Paillier. Notice that in both situations, a is uniformly distributed over $\mathbb{Z}_N$.

P_0 is corrupted. Consider the following series of indistinguishable hybrids.

Hybrid 1. In this hybrid, the simulator simulates the resources $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}, \mathcal{F}_{\mathsf{NIKE}}$, the random oracle and the messages sent by P_1 as in the protocol. If the procedure does not abort, the simulator computes y_0 using the witnesses provided by P_0. Then, it sets $y_1 \leftarrow y_0 + d \cdot x$ (the addition is performed over $\mathbb{Z}$). The rest of the operations is performed as in the protocol. This hybrid is indistinguishable from the real protocol by the first claim showed in this proof.

Hybrid 2. In this hybrid, the simulator performs the same operations as in Hybrid 1. However, instead of generating A as in the protocol, the simulator sets $A \leftarrow g^{r_1}$. Observe that this stage is indistinguishable from the previous one by the hiding properties of the commitment scheme in Fig. 10.

Hybrid 3. In this hybrid, the simulator performs the same operations as in Hybrid 2. However, instead of sampling x uniformly over $[2^\lambda \cdot 2^{\mathsf{len}_N}]$, it samples it uniformly over $\mathbb{Z}_N$. Observe that this stage is unconditionally indistinguishable from the previous one. Indeed, the statistical distance is dominated by $2^{-\lambda}$, which is negligible.

Hybrid 4. In this hybrid, the simulator behaves as in the previous stage, but lets the functionality $\mathcal{F}_{\mathsf{r\text{-}VOLE}}$ generate the outputs. Specifically, the simulator forwards the RSA modulo N received from P_0 to the functionality. Then, it retrieves a and z_0 and forwards them to the functionality. This hybrid is perfectly indistinguishable from the previous one.

P_1 is corrupted. Consider the following series of indistinguishable hybrids.

Hybrid 1. In this hybrid, the simulator simulates the resources $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$, $\mathcal{F}_{\mathsf{NIKE}}$, the random oracle and the messages sent by P_1 as in the protocol. If the procedure does not abort, the simulator computes y_1 using the witnesses provided by P_1. Then, it sets $y_0 \leftarrow y_1 - d \cdot x$ (the subtraction is performed over $\mathbb{Z}$). The rest of the operations is performed as in the protocol. This hybrid is indistinguishable from the real protocol by the first claim showed in this proof.

Hybrid 2. In this hybrid, the simulator performs the same operations as in Hybrid 1. However, instead of generating E as in the protocol, the simulator sets $E \leftarrow f^d \cdot g^{r'_0}$ where r'_0 is uniform over $[\ell]$ and independent of r_0. Observe that this stage is indistinguishable from the previous one by the DXDH assumption.

Hybrid 3. In this hybrid, the simulator performs the same operations as in Hybrid 2. However, the simulator sets $E \leftarrow f^s \cdot g^{r'_0}$ where s is uniformly sampled over $[t]$. Observe that this stage is indistinguishable from the previous one as $g^{r'_0}$ is close to uniform over g and $f \in \langle g \rangle$.

Hybrid 4. In this hybrid, the simulator computes z_1 using the witnesses provided by P_1 to $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$. Then, it sets $z_0 \leftarrow z_1 - a \cdot x$. This hybrid is indistinguishable from the previous one, by the second claim showed in this proof.

Hybrid 5. In this hybrid, the simulator behaves as in the previous stage, but lets the functionality $\mathcal{F}_{\mathsf{r\text{-}VOLE}}$ generate the outputs and the RSA modulo. Specifically, the simulator obtains N from the functionality, receives x from the adversary and sends $x \bmod N$ to the functionality. Then, it computes the output z_1 and forwards it to the $\mathcal{F}_{\mathsf{r\text{-}VOLE}}$. Moreover, the simulator replies to all oracle queries with random encryptions of 0. This hybrid is indistinguishable from the previous one by the IND-CPA security security of Paillier. Notice that in both situations, a is uniformly distributed over $\mathbb{Z}_N$. □

7.4 Actively secure public-key PCF for oblivious transfer

We build this protocol in a way similar to the actively secure public-key PCF for vector-OLE in Section 7.3. Similarly to the vector-OLE case, in the semi-honest public-key PCF for OT, party P_0 sends an RSA modulo $N = p \cdot q$ and a pair of group elements (D, E) where $D = g^{r_0}$ and $E = f^d \cdot C^{r_0}$. Here, d denotes the Goldwasser-Micali private key associated with N. Specifically, $d = \varphi(N)/4 = (p-1)/2 \cdot (q-1)/2$, since p and q are odd, d is always an integer. The Goldwasser-Micali cryptosystem requires also that -1 is a quadratic non-residue of $\mathbb{Z}_N^\times$ with Jacobi symbol equal to 1. That happens if and only if p and q are both congruent to 3 modulo 4.

The message sent by P_1 consists instead of λ group elements $A_1, \ldots, A_\lambda$ where $A_i = C^{x_i} \cdot g^{r_1^i}$ for a bit $x_i \in \{0, 1\}$.

In order to upgrade the public-key PCF for OT to active security, we need to design NIZKs that prove the well-formedness of the messages of the parties. As in the vector-OLE case, the task is simple for the messages of party P_1. Indeed, in order to prove that A_i is a commitment to a bit x_i, it is sufficient to prove that A_i is also a commitment to x_i^2 ($x = 0$ and $x = 1$ are the only integers for which $x^2 = x$). We can achieve this by applying the Fiat-Shamir heuristic on Π_{mult}. Proving the well-formedness of P_0's message is a bit more challenging.

Proving the well-formedness of P_0's message. We follow the same approach as in the vector-OLE case: first, we design a public-coin honest-verifier zero-knowledge proof and then we convert it into a NIZK using the Fiat-Shamir heuristic. Our solution is described in Fig. 15.

The prover starts by committing to $p' = (p-3)/4$ and $q' = (q-3)/4$. Let the resulting commitments be X_1 and X_2. Using $\Pi_{\mathsf{semiprime}}$ as in the vector-OLE case, the verifier check that N is the product of two primes and $\gcd\big(N, \varphi(N)\big) = 1$.

The parties define W_1 and W_2 to be $X_1^4 \cdot C^3$ and $X_2^4 \cdot C^3$. Notice that if the prover is honest, these are commitments to the primes p and q. Using Π_{mult}, the prover shows that W_1 and W_2 are indeed commitments to a factorisation of N. Moreover, using Π_{range} and Π_{com}, the verifier checks that p' belongs to $[0, N/4 - 1]$ and that the prover knows an opening for X_2. As a consequence, it is certain that $3 \leq 4p' + 3 < N - 1$, hence, W_1 and W_2 are a commitment to a proper factorisation of N, which must be $p \cdot q$. This also proves that p and q are congruent to 3 modulo 4.

In the end, the parties define W to be $C^{(N-5)/4} \cdot X_1^{-1} \cdot X_2^{-1}$. Notice that N is congruent to 1 modulo 4, so $(N-5)/4$ is an integer. Moreover, by the linear properties of the scheme, W is commitment to

$$\frac{N-5}{4} - p' - q' = \frac{N - 4p' - 4q' - 5}{4} = \frac{N - p + 3 - q + 3 - 5}{4} = \frac{\varphi(N)}{4} = d.$$

Using Π_{plain} on W, the verifier checks that the value hidden in (D, E) is d.

Theorem 7. *Let $\Pi_{\mathsf{semiprime}}$ be a honest-verifier zero-knowledge public-coin proof proving that N is the product of two primes and $\gcd\big(N, \varphi(N)\big) = 1$. If the commitment scheme in Fig. 10 is hiding and binding, the construction Π_{GM} in Fig. 15 is a complete, special-sound public-coin proof for the relation*

$$\mathcal{R}_{\mathsf{GM}} := \left\{ (D, E, N), (d, p, q, r) \middle| \begin{array}{l} N = p \cdot q, \text{ where } p, q \text{ are positive primes} \\ \gcd\big(N, \varphi(N)\big) = 1 \\ D = g^r, E = f^d \cdot C^r \\ d = \varphi(N)/4 \\ p, q \equiv 3 \bmod 4 \end{array} \right\}$$

Moreover, if $r \in [\ell]$, the proof is honest-verifier zero-knowledge.

Proof of encryption of the GM private key Π_{GM}

CRS:

1. $\mathsf{par}' := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(\mathbb{1}^\lambda)$
2. $(g, \rho_0) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
3. $(C, \rho_1) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
4. If $C = g$, go to step 3.
5. Output $\mathsf{par} := (\mathsf{par}', g, \rho_0, C, \rho_1)$

PROTOCOL:

1. The prover computes $p' \leftarrow (p-3)/4$ and $q' \leftarrow (q-3)/4$.
2. The prover samples $s_1, s_2 \xleftarrow{R} [\ell]$.
3. The prover sends $X_1 \leftarrow C^{p'} \cdot g^{s_1}$ and $X_2 \leftarrow C^{q'} \cdot g^{s_2}$.
4. The parties run $\Pi_{\mathsf{semiprime}}$ with input N and witness (p, q).
5. The parties set $W \leftarrow C^{(N-5)/4} \cdot X_1^{-1} \cdot X_2^{-1}$, $W_1 \leftarrow X_1^4 \cdot C^3$ and $W_2 \leftarrow X_2^4 \cdot C^3$.
6. The parties run Π_{mult} with input W_1, W_2, C^N. The witness is $(p, q, N, 4s_1, 4s_2, 0)$.
7. The parties run Π_{range} with input X_1 and bound $B = N/4 - 1$. The witness is (p', s_1).
8. The parties run Π_{com} with input X_2. The witness is (q', s_2).
9. The parties run Π_{plain} with input W, D, E and witness $(\varphi(N)/4, r, -2s_1 - 2s_2)$.
10. The verifier accepts if all the above proofs succeed.

Fig. 15. Honest-verifier zero-knowledge proof of encryption of the GM private key

Proof. Completeness. By the completeness of Π_{range}, Π_{com}, Π_{plain} and Π_{mult}, the verifier always accepts when dealing with a honest prover. Indeed,

$$C^{\frac{N-5}{4}} \cdot X_1^{-1} \cdot X_2^{-1} = C^{\frac{N-5}{4}} \cdot C^{-p'} \cdot g^{-s_1} \cdot C^{-q'} \cdot g^{-s_2} = C^{\frac{N-5}{4} - p' - q'} \cdot g^{-s_1 - s_2}$$

and

$$(N-5)/4 - p' - q' = (N - 4p' - 4q' - 5)/4 = (N - p - q + 1)/4 = \varphi(N)/4.$$

Special Soundness. By the special soundness of Π_{mult}, the extractor is able to retrieve $p, q \in \mathbb{Z}$ and $s_1', s_2', s_3' \in \mathbb{Z}$ such that $W_1 = C^3 \cdot X_1^4 = C^p \cdot g^{s_1'}$ and $W_2 = C^3 \cdot X_2^4 = C^q \cdot g^{s_2'}$.

By the special soundness of Π_{range} and Π_{com}, the extractor is also able to obtain values $p' \in [0, N/4 - 1]$ and $q', s_1, s_2 \in \mathbb{Z}$ such that $X_1 = C^{p'} \cdot g^{s_1}$ and $X_2 = C^{q'} \cdot g^{s_2}$. Observe that

$$C^p \cdot g^{s_1'} = C^3 \cdot X_1^4 = C^{4p'+3} \cdot g^{4s_1}, \qquad C^q \cdot g^{s_2'} = C^3 \cdot X_2^4 = C^{4q'+3} \cdot g^{4s_2}.$$

By the binding properties of the commitment scheme in Fig. 10, it must be that $p = 4p' + 3$ and $q = 4q' + 3$. Moreover, since $p' \in [0, N/4 - 1]$, we know that

$p \in [3, N-1]$ and so $N = p \cdot q$ is a proper factorisation. By the soundness of $\Pi_{\mathsf{semiprime}}$, we know that N is a product of two distinct primes. Therefore, p and q must be prime. Moreover, p and q are both congruent to 3 modulo 4.

Finally, by the special soundness of Π_{plain}, the extractor is able to retrieve $d, s \in \mathbb{Z}$ such that $W = C^d \cdot g^s$, $D = g^r$, $E = f^d \cdot C^r$. As we observed when we analysed completeness,

$$C^{\frac{N-5}{4}} \cdot X_1^{-1} \cdot X_2^{-1} = C^{\frac{\varphi(N)}{4}} \cdot g^{-s_1 - s_2}.$$

Once again, by the binding properties of the commitment scheme in Fig. 10, it must be that $d = \varphi(N)/4$.

Honest-Verifier Zero-Knowledge. Suppose that $r \in [\ell]$. We proceed by a series of hybrids.

Hybrid 1. In this hybrid, the simulator generates the transcripts in $\Pi_{\mathsf{semiprime}}$, Π_{range}, Π_{com}, Π_{mult} and Π_{plain} using the corresponding honest verifier simulators. Since $p' \in [0, B]$, $d \in [0, N]$ and $r \in [\ell]$, by the honest-verifier zero-knowledge of $\Pi_{\mathsf{semiprime}}$, Π_{range}, Π_{com}, Π_{mult} and Π_{plain}, no adversary can distinguish Hybrid 1 from the real protocol.

Hybrid 2. In this hybrid, the simulator sends $X_1 \leftarrow g^{s_1}$ and $X_2 \leftarrow g^{s_2}$, instead of $C^{p'} \cdot g^{s_1}$ and $C^{q'} \cdot g^{s_2}$. The rest of the transcript is generated as in Hybrid 1. By the hiding properties of the commitment scheme in Fig. 10, no adversary can distinguish between Hybrid 1 and Hybrid 2.

Observe that now the transcript can be generated without knowing the witness. □

Deploying the NIZKs to obtain active security. We are now ready to present our actively secure public-key PCF for oblivious transfer. The construction, called $\Pi_{\mathsf{OT}}^{\mathsf{Active}}$ is described in Fig. 17.

We prove that our pk-PCF implements the OT functionality $\mathcal{F}_{\mathsf{OT}}$ (see Fig. 16) in the UC model. $\mathcal{F}_{\mathsf{OT}}$ is a functionality that, during the initialisation samples a random $\boldsymbol{x} \in \{0,1\}^\lambda$ and outputs it to P_1. Upon any request for an OT tuple, $\mathcal{F}_{\mathsf{OT}}$ samples a random bit $b \in \{0,1\}$ and generates a random secret-sharing $\boldsymbol{z}_0 \oplus \boldsymbol{z}_1 = b \cdot \boldsymbol{x}$. Then, $\mathcal{F}_{\mathsf{OT}}$ outputs $(b, \boldsymbol{z}_0)$ to P_0 and $\boldsymbol{z}_1$ to P_1. If one of the parties is corrupt, the functionality lets the adversary choose the outputs for the corrupted player, then, it samples the outputs of the honest party at random conditioned on $\boldsymbol{z}_0 \oplus \boldsymbol{z}_1 = b \cdot \boldsymbol{x}$. Moreover, if P_1 is corrupt, $\mathcal{F}_{\mathsf{OT}}$ lets the adversary choose $\boldsymbol{x}$ during the initialisation.

As in the vector-OLE case, we use $\mathcal{F}_{\mathsf{NIKE}}$ and $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ as resources. Again, we notice that our NIZKs do not implement $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ in the UC model. However, using them is still considered secure.

Theorem 8. *Let* $\mathsf{len}_N(\lambda)$ *be the length of the RSA modulo and assume that* $t > 2^{\lambda + \mathsf{len}_N}$. *Let* F *be a secure PRF outputting pseudorandom elements in* $\mathbb{Z}_2$. *If the DXDH assumption holds in the NIDLS framework, the protocol* $\Pi_{\mathsf{OT}}^{\mathsf{Active}}$ *UC implements the functionality* $\mathcal{F}_{\mathsf{OT}}$ *against an active adversary in the* ($\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$, $\mathcal{F}_{\mathsf{NIKE}}$)*-hybrid model with random oracle.*

Function $\mathcal{F}_{\mathsf{OT}}$

INITIALISATION:

- If P_1 is honest, sample $\boldsymbol{x} \xleftarrow{R} \{0,1\}^\lambda$.
- If P_1 is corrupt, wait for $\boldsymbol{x} \in \{0,1\}^\lambda$ from the adversary.
- If the adversary sends $\perp$, abort.

EVALUATION: On input a fresh label id from an honest party P_i.

- If both parties are honest, the functionality samples $b \xleftarrow{R} \{0,1\}$ and $\boldsymbol{z}_1 \xleftarrow{R} \{0,1\}^\lambda$ and sets $\boldsymbol{z}_0 \leftarrow \boldsymbol{z}_1 \oplus b \cdot \boldsymbol{x}$. Then, it sets $R_0 \leftarrow (b, \boldsymbol{z}_0)$ and $R_1 \leftarrow (\boldsymbol{x}, \boldsymbol{z}_1)$. $\mathcal{F}_{\mathsf{OT}}$ outputs R_i to P_i and stores $(\mathsf{id}, 1-i, R_{1-i})$.
- If $i = 1$ and P_0 is corrupted, the functionality waits for $b \in \{0,1\}$ and $\boldsymbol{z}_0 \in \{0,1\}^\lambda$ from the adversary and sets $\boldsymbol{z}_1 \leftarrow \boldsymbol{z}_0 \oplus b \cdot \boldsymbol{x}$. Then, it outputs $(\boldsymbol{x}, \boldsymbol{z}_1)$ to P_i.
- If $i = 0$ and P_1 is corrupted, the functionality waits for $\boldsymbol{z}_1 \in \{0,1\}^\lambda$ from the adversary, samples $b \in \{0,1\}$ and computes $\boldsymbol{z}_0 \leftarrow \boldsymbol{z}_1 \oplus b \cdot \boldsymbol{x}$. Then, it outputs $(b, \boldsymbol{z}_0)$ to P_i.

If id is not fresh, retrieve the triple (id, i, R_i) and output R_i to P_i.

Fig. 16. The OT functionality

Proof. We start by proving the correctness of the protocol.

Claim. If the protocol does not abort, we have that $\boldsymbol{y}_1 = \boldsymbol{y}_0 + d \cdot \boldsymbol{x}$. Here, the operations are computed over the integers and the terms $\boldsymbol{y}_0, \boldsymbol{y}_1, d$ and $\boldsymbol{x}$ are computes as in the protocol using the witnesses input into $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$.

Proof of the claim. Notice that

$$D^{r_1^i} \cdot E^{x_i} = (g^{r_0})^{r_1^i} \cdot (f^d \cdot C^{r_0})^{x_i} = f^{x_i \cdot d} \cdot (C^{x_i} \cdot g^{r_1^i})^{r_0} = f^{x_i \cdot d} \cdot A_i^{r_0}.$$

We conclude that $\boldsymbol{v}_0 + \boldsymbol{v}_1 = \boldsymbol{x} \cdot d \bmod t$ and so $\boldsymbol{y}_1 = \boldsymbol{y}_0 + d \cdot \boldsymbol{x} \bmod t$. Now, observe that $\boldsymbol{y}_0$ is uniformly distributed over $\mathbb{Z}_t^\lambda$ due to the random oracle rerandomisation. Since $t > 2^\lambda \cdot 2^{\mathsf{len}_N} \geq 2^\lambda \cdot d \cdot x_i$, the probability that $y_{0,i} \in [t-d, t]$ is smaller than $2^{-\lambda}$, which is negligible. So, with overwhelming probability, there are no wrap-arounds and $\boldsymbol{y}_1 = \boldsymbol{y}_0 + d \cdot \boldsymbol{x}$ even if the operations are performed over the integers. Notice that it is fundamental that the query to the oracle includes $(A_i)_{i \in [\lambda]}, D, E$ and N, so the adversary cannot pick the messages of the corrupted parties as a function of the random oracle output, as this could allow them to cause wrap-arounds and compromise the correctness of the result. □

Claim. If the protocol does not abort, we have that $\boldsymbol{z}_0 = \boldsymbol{z}_1 \oplus b \cdot \boldsymbol{x}$ for every execution of Evaluation.

Active PK-PCF for OT $\Pi_{\mathsf{OT}}^{\mathsf{Active}}$

Let F be a PRF. Let len_N denote the length of the GM modulo and let t, the order of the NIDLS group, be greater than $2^\lambda \cdot 2^{\mathsf{len}_N}$.
INITIALISATION:

1. The parties initialise $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ obtaining $\mathsf{par} := (\mathsf{par}', g, \rho_0, C, \rho_1)$.
2. The parties call $\mathcal{F}_{\mathsf{NIKE}}$ to obtain a PRF key k.
3. P_0 computes $(N, d) \xleftarrow{R} \mathsf{GM.Gen}(1\!\!1^\lambda)$ where $N = p \cdot q$.
4. P_1 samples $\boldsymbol{x} \xleftarrow{R} \{0,1\}^\lambda$.
5. P_0 samples $r_0 \xleftarrow{R} [\ell]$ and sets $D \leftarrow g^{r_0}$, $E \leftarrow f^d \cdot C^{r_0}$.
6. P_0 sends N, D, E.
7. P_1 samples $r_1^i \xleftarrow{R} [\ell]$ and computes $A_i \leftarrow C^{x_i} \cdot g^{r_1^i}$ for every $i \in [\lambda]$.
8. P_1 sends $(A_i)_{i\in[\lambda]}$.
9. The parties call $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ with input (GM, D, E, N). P_0 inputs also (d, p, q, r_0). The parties abort if the functionality outputs 0.
10. For every $i \in [\lambda]$, the parties call $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ with input $(\mathsf{mult}, A_i, A_i, A_i)$ to prove that $x_i^2 = x_i$. P_1 also inputs (x_i, r_1^i). If the functionality outputs 0, the parties abort.
11. The parties query $(N, D, E, A_1, \ldots, A_\lambda)$ to the random oracle and they obtain λ random elements $u_1, \ldots, u_\lambda \in \mathbb{Z}_t$.
12. P_0 computes $v_{0,i} \leftarrow \mathsf{DDLog}_{\mathsf{par}}(A_i^{r_0}) - u_i \bmod t$ for every $i \in [\lambda]$
13. P_1 computes $v_{1,i} \leftarrow \mathsf{DDLog}_{\mathsf{par}}(D^{r_1^i} \cdot E^x) + u_i \bmod t$ for every $i \in [\lambda]$.
14. P_0 outputs $\mathsf{k}_0 \leftarrow (N, k, \boldsymbol{y}_0 := -\boldsymbol{v}_0, d/4)$.
15. P_1 outputs $\mathsf{k}_1 \leftarrow (N, k, \boldsymbol{y}_1 := \boldsymbol{v}_1, \boldsymbol{x})$.

EVALUATION: Query the label id to the oracle. Let $\mathsf{ct} \in \mathbb{J}_N$ be the response:

1. P_0 computes $b \leftarrow \mathsf{GM.Dec}(d, \mathsf{ct})$.
2. For every $i \in [\lambda]$, each P_j computes $z_{j,i} \leftarrow \mathsf{DDLog}_{\mathsf{GM}}(\mathsf{ct}^{y_{j,i}}) \oplus \mathsf{F}_k(\mathsf{ct}, i) \bmod 2$.
3. P_0 outputs $(b, \boldsymbol{z}_0)$, P_1 outputs $(\boldsymbol{x}, \boldsymbol{z}_1)$.

Fig. 17. Active public-key PCF for OT

Proof of the claim. Even if N is not the product of two safe-primes, the NIDLS techniques are still working. Specifically, since $N = p \cdot q$, $p \neq q$ and $(p-1)/2$ and $(q-1)/2$ are odd, -1 is still a non square element with Jacobi symbol 1. Moreover, $\mathbb{J}_N$ is still isomorphic to $\langle -1 \rangle \times SQ$ where SQ is the subgroup of squares of $\mathbb{Z}_N^\times$.

Now, let $\mathsf{ct} = (-1)^b \cdot r^2$ for some $r \in \mathbb{Z}_N^\times$. We recall that $d = \varphi(N)/4$ is odd and $r^{2d} = 1$ for every element in $\mathbb{Z}_N^\times$, so we have that

$$\mathsf{ct}^{y_1^i} = \mathsf{ct}^{y_0^i + x_i \cdot d} = \mathsf{ct}^{y_0^i} \cdot (-1)^{b \cdot x_i \cdot d} \cdot r^{2 \cdot d \cdot x_i} = \mathsf{ct}^{y_0^i} \cdot (-1)^{b \cdot x_i}.$$

So, after applying DDLog, the parties obtain $z_{0,i}$ and $z_{1,i}$ that satisfy $z_{0,i} \oplus z_{1,i} = b \cdot x_i$. □

All the parties are honest. Consider the following series of indistinguishable hybrids.

Hybrid 1. In this hybrid, the simulator sets $\boldsymbol{z}_0 \leftarrow \boldsymbol{z}_1 \oplus b \cdot \boldsymbol{x}$ for every execution of $\mathsf{Evaluation}$. By the previous claim, this hybrid is indistinguishable from the real protocol.

Hybrid 2. Consider the hybrid in which the simulator sets $A_i \leftarrow g^{r_1^i}$ instead of $A_i \leftarrow C^{x_i} \cdot g^{r_1^i}$ for every $i \in [\lambda]$. This hybrid is indistinguishable from the previous one by the hiding properties of the commitment scheme in Fig. 10.

Hybrid 3. In this hybrid, instead of generating E as in the protocol, the simulator sets $E \leftarrow f^d \cdot g^{r'_0}$ where r'_0 is uniform over $[\ell]$ and independent of r_0. Observe that this stage is indistinguishable from the previous one by the DXDH assumption.

Hybrid 4. In this hybrid, the simulator sets $E \leftarrow f^{r'} \cdot g^{r'_0}$ where r' is uniformly sampled over $[t]$. Observe that this stage is indistinguishable from the previous one as $g^{r'_0}$ is close to uniform over g and $f \in \langle g \rangle$.

Hybrid 5. In this hybrid, the simulator samples $\boldsymbol{z}_1$ uniformly over $\{0,1\}^\lambda$. By the security of the PRF F, the adversary cannot distinguish between this hybrid and the previous one.

Hybrid 6. In this hybrid, the simulator lets the functionality $\mathcal{F}_{\mathsf{OT}}$ generate the outputs. Moreover, it replies to all random oracle queries by sending random GM encryptions of 0. This hybrid is indistinguishable from the previous one by the IND-CPA security of GM. Notice that in both situations, b is uniformly distributed over $\{0,1\}$.

P_0 is corrupted. Consider the following series of indistinguishable hybrids.

Hybrid 1. In this hybrid, the simulator simulates the resources $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$, $\mathcal{F}_{\mathsf{NIKE}}$, the random oracle and the messages sent by P_1 as in the protocol. If the procedure does not abort, the simulator computes $\boldsymbol{y}_0$ and d using the witnesses provided by P_0. Then, it sets $\boldsymbol{y}_1 \leftarrow \boldsymbol{y}_0 + d \cdot \boldsymbol{x}$ (the addition is performed over $\mathbb{Z}$). The rest of the operations are performed as in the protocol. This hybrid is indistinguishable from the real protocol by the first claim showed in this proof.

Hybrid 2. In this hybrid, the simulator performs the same operations as in Hybrid 1. However, instead of generating A_i as in the protocol, the simulator

sets $A_i \leftarrow g^{r_1^i}$ for every $i \in [\lambda]$. Observe that this stage is indistinguishable from the previous one by the hiding properties of the commitment scheme in Fig. 10.

Hybrid 3. In this hybrid, the simulator behaves as in the previous stage, but let the functionality $\mathcal{F}_{\mathsf{OT}}$ generate the outputs. Specifically, the simulator retrieves b and $\boldsymbol{z}_0$ using the witnesses provided by P_0. Then, it forwards them to the functionality. This hybrid is perfectly indistinguishable from the previous one by the second claim shown in this proof.

P_1 is corrupted. Consider the following series of indistinguishable hybrids.

Hybrid 1. In this hybrid, the simulator simulates the resources $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$, $\mathcal{F}_{\mathsf{NIKE}}$, the random oracle and the messages sent by P_1 as in the protocol. If the procedure does not abort, the simulator computes $\boldsymbol{y}_1$ using the witnesses provided by P_1. Then, it sets $\boldsymbol{y}_0 \leftarrow \boldsymbol{y}_1 - d \cdot \boldsymbol{x}$ (the subtraction is performed over $\mathbb{Z}$). The rest of the operations are performed as in the protocol. This hybrid is indistinguishable from the real protocol by the first claim showed in this proof.

Hybrid 2. In this hybrid, the simulator performs the same operations as in Hybrid 1. However, instead of generating E as in the protocol, the simulator sets $E \leftarrow f^d \cdot g^{r'_0}$ where r'_0 is uniform over $[\ell]$ and independent of r_0. Observe that this stage is indistinguishable from the previous one by the DXDH assumption.

Hybrid 3. In this hybrid, the simulator performs the same operations as in Hybrid 2. However, the simulator sets $E \leftarrow f^{r'} \cdot g^{r'_0}$ where r' is uniformly sampled over $[t]$. Observe that this stage is indistinguishable from the previous one as $g^{r'_0}$ is close to uniform over g and $f \in \langle g \rangle$.

Hybrid 4. In this hybrid, the simulator behaves as in the previous stage, but lets the functionality $\mathcal{F}_{\mathsf{OT}}$ generate the outputs. Specifically, the simulator receives $\boldsymbol{x}$ from the adversary and forwards it to the functionality. Then, it computes the output $\boldsymbol{z}_1$ using the witnesses provided by P_1 and sends it to the functionality. The simulator also replies to all random oracle queries with a random GM encryption of 0. This hybrid is indistinguishable from the previous one by the IND-CPA security of GM. Notice that in both cases, b is uniformly distributed over $\{0, 1\}$. □

8 Active HSS Protocol

In this section, we explain how to design efficient 2-round MPC protocols for RMS programs with security against active adversaries. The functionality corresponding to our construction is described in Fig. 18. Our solution is based on the semi-honest HSS scheme we described in Section 5 and relies on a one-round setup procedure. When the NIDLS framework is instantiated over class groups, the construction does not need any trusted setup.

Limited to the context of RMS programs for 2 parties, our solution achieves higher efficiency compared to other 2-round MPC protocols. We point out, for instance, that assuming $B \geq 2^\lambda$, the communication complexity of the scheme is independent of the size of the circuit it is evaluated.

Function $\mathcal{F}_{\mathsf{RMS}}$

INITIALISATION: On input Init from both parties the functionality activates.
INPUT: On input (Input, i, x) with $x \in \mathbb{Z}_{n_{\mathsf{out}}}$ from P_i and (Input, i) from P_{1-i}, the functionality assigns the value x to I_x.
EVALUATION: On input $\big(\mathsf{Eval}, (\mathsf{I}^1, \ldots, \mathsf{I}^n), P\big)$ from each party, where P is a B-bounded RMS program with n inputs, the functionality retrieves the value x_i assigned to I^i for every $i \in [1, n]$. Then, it computes $z \leftarrow P(x_1, \ldots, x_n)$ a provides it to the adversary. If the adversary replies with Abort, the functionality aborts, otherwise it outputs z to the honest parties.
ABORT: On input Abort from the adversary, the functionality aborts.

Fig. 18. The RMS functionality

The challenges posed by an active adversary. The HSS scheme described in Section 5 induces a semi-honest 2-round MPC protocol with setup. When dealing with an active adversary, the scheme can fail in two points: the input phase and the output reconstruction. A corrupted party can indeed send malformed input messages or add additive errors to the shares of the outputs. We now explain how we solved these problems.

Verifying the well-formedness of the input messages. We address the first issue using NIZKs. In particular, we need to prove that every input message is of the form $D = g^r$, $E = f^x \cdot h^r$, $\overline{D} = g^{\overline{r}} \cdot f^{-x}$, $\overline{E} = h^{\overline{r}}$. Since the HSS scheme is correct only when the inputs are bounded, we also need to prove that $x \in [-B, B]$ for some bound $B \in \mathbb{N}$. It is possible to design efficient solutions for this problem using standard techniques. Specifically, we consider the public-coin honest-verifier zero-knowledge protocol Π_{input} in which the prover commits to x using the scheme in Fig. 10 and proves that $x \in [-B, B]$ (this is done by running Π_{range} on $X \cdot C^B$ with bound $2B$, where X denotes the commitment). Moreover, the prover and the verifier run also a procedure similar to Π_{plain} to prove that $D = g^r$, $E = f^x \cdot h^r$, $\overline{D} = g^{\overline{r}} \cdot f^{-x}$, $\overline{E} = h^{\overline{r}}$ where x is the value committed beforehand. For more details, we refer to Appendix A.5.

Authenticating the output shares. To detect additive errors in the output shares, we introduce information theoretical MACs. The latter are used to authenticate the secret-sharings associated with the memory wires. Since we want to perform the output reconstruction with only one round of interaction, we adopt BeDOZa-style MACs [BDOZ11]. Specifically, each party P_i holds a random MAC key $\alpha_i \in [n_{\mathsf{out}}]$. For each memory wire of value z, in addition to its integer share z_i, each party P_i holds values $M_{0,i}(z), M_{1,i}(z) \in \mathbb{Z}$ such that

$$M_{i,i}(z) = M_{i,1-i}(z) + \alpha_i \cdot z.$$

The operations in the above formula are all intended over $\mathbb{Z}$.

When P_i sends its share of the output $z_i \bmod n_{\mathsf{out}}$, the player attaches also $M_{1-i,i} \bmod n_{\mathsf{out}}$. The other party checks that

$$M_{1-i,1-i}(z) = M_{1-i,i}(z) + \alpha_{1-i} \cdot (z_1 - z_0) \bmod n_{\mathsf{out}}.$$

Since P_i will know no information about the MAC key α_{1-i}, the probability that P_i successfully fools P_{1-i} by sending a tampered share is limited. Specifically, when n_{out} is prime, the probability is smaller than $1/n_{\mathsf{out}}$. In order to make the latter negligible, we assume that n_{out} is a prime larger than 2^λ. If n_{out} does not satisfy these properties, it is still possible to adapt our MACs so that we can verify the authenticity of the shares in one round (If n_{out} is small, we just apply λ of these MACs, as in [NNOB12]. If n_{out} is not prime, as in [CDE+18], we choose the MAC keys at random in $\mathbb{Z}_{2^\lambda \cdot n_{\mathsf{out}}}$ and we embed z in $\mathbb{Z}_{2^\lambda \cdot n_{\mathsf{out}}}$ for the verification).

Authenticating the computations. The MACs we applied on the HSS scheme have useful linear properties. In particular, by locally adding the MACs authenticating two memory wires M_x and M_y, we obtain MACs over their addition without any interaction. Indeed,

$$M_{i,i}(x) + M_{i,i}(y) = \big(M_{i,1-i}(x) + M_{i,1-i}(y)\big) + \alpha_i \cdot (x + y).$$

Authenticating the outputs of multiplication gates without interaction is a more challenging task, but the fact that we are dealing with RMS programs simplifies the problem. Indeed, we know that one of the factors is an input, so we have ElGamal ciphertexts ct_x and ct_{xs} encrypting its value x and $x \cdot s$. The main observation in Section 5 was that, given subtractive secret-sharings over the integers of y and $y \cdot s$, the NIDLS techniques permit obtaining subtractive secret-sharings of $x \cdot y$ and $x \cdot y \cdot s$ over $\mathbb{Z}$ without any interaction.

We observe that in an authenticated memory wire M_y, the MACs $M_{i,i}(y)$ and $M_{i,1-i}(y)$ are exactly a subtractive secret-sharing of $\alpha_i \cdot y$ over $\mathbb{Z}$. Suppose now that the parties have also a subtractive secret-sharing of $\alpha_i \cdot y \cdot s$ over $\mathbb{Z}$ for every $i \in \{0, 1\}$. In other words, each party P_i holds $M'_{i,i}(y), M'_{1-i,i}(y) \in \mathbb{Z}$ such that

$$M'_{i,i}(y) = M'_{i,1-i}(y) + \alpha_i \cdot y \cdot s.$$

By applying the NIDLS techniques on these secret-sharings and on ct_x and ct_{xs}, the parties obtain a subtractive secret-sharing of $\alpha_i \cdot (x \cdot y)$ and $\alpha_i \cdot (x \cdot y) \cdot s$ for every $i \in \{0, 1\}$ without any interaction. This is exactly what we want.

Notice also that the values $M'_{i,i}(y), M'_{1-i,i}(y)$ enjoy the same linear properties as the BeDOZa-style MACs. Specifically, if we have two memory wires M_x and M_y that are equipped with secret-sharings of $\alpha_i \cdot x \cdot s$ and $\alpha_i \cdot y \cdot s$ respectively, we can equip their addition with analogous secret-sharings. Indeed,

$$M'_{i,i}(x) + M'_{i,i}(y) = \big(M'_{i,1-i}(x) + M'_{i,1-i}(y)\big) + \alpha_i \cdot (x + y) \cdot s.$$

To summarise, during the evaluation of an RMS program, each memory wire M_y will be associated with subtractive secret-sharings of its value y, of $y \cdot s$, of

$\alpha_i \cdot y$ and of $\alpha_i \cdot y \cdot s$. So, each party P_i will hold a tuple

$$\big(y_i, y'_i, M_{0,i}(y), M_{1,i}(y), M'_{0,i}(y), M'_{1,i}(y)\big)$$

such that

$$\begin{aligned} y_1 - y_0 &= y, & y'_1 - y'_0 &= y \cdot s, \\ M_{i,i}(y) &= M_{i,1-i}(y) + \alpha_i \cdot y, & M'_{i,i}(y) &= M'_{i,1-i}(y) + \alpha_i \cdot y \cdot s. \end{aligned}$$

In order for the construction to be correct, we require t to be 2^λ times larger than $\alpha_i \cdot y \cdot s$. If this condition is not satisfied, we might not be able to convert the secret-sharings over $\mathbb{Z}_t$ output by the multiplication gates into subtractive secret-sharings over $\mathbb{Z}$. So, if we denote the bound of the RMS programs by B, we need t to be greater than $2^\lambda \cdot n_{\mathsf{out}} \cdot B \cdot \ell_{\mathsf{sk}}$.

Designing a one-round setup procedure. The semi-honest HSS scheme described in Section 5 already needed a setup phase that generated the ElGamal public key and provided the parties with a subtractive secret-sharing of the private counterpart. We also showed that it is possible for the parties to set up the parameters using only one-round of interaction. In the protocol, each party P_i sampled s_i at random in $[\ell_{\mathsf{sk}}]$ and sent $h_i := g^{s_i}$ to the other player. The public key was set to be $h := h_1/h_0$.

In the actively secure HSS protocol, the setup needs to perform some additional work. Indeed, we need to provide the parties with the material associated with the initial memory wire M_1, the one used to load the inputs into memory cells. The value associated with M_1 is 1, the material associated with it is subtractive secret-sharings of 1, s, α_i and $\alpha_i \cdot s$. Dealing the first three terms is rather straightforward: the parties already know a subtractive secret-sharing of $s = s_1 - s_0$, whereas subtractive secret-sharings of 1 and α_i are trivially 1 and 0 and α_i and 0. Deriving the last term is slightly more challenging.

In the solution we designed, each party P_i samples $s_i \in [\ell_{\mathsf{sk}}]$ and sends $h_i := g^{s_i}$ as in the semi-honest case. However, P_i also sends $A_i := f^{\alpha_i} \cdot g^{r_i}$ for some r_i uniformly sampled over $[\ell]$. After proving the well-formedness of these messages using NIZKs, the parties derive the ElGamal public key $h := h_1/h_0$. The parties are now able to compute a subtractive-secret-sharing of $\alpha_i \cdot s$ for every $i \in \{0, 1\}$. Indeed, we have that

$$A_i^{s_{1-i}} = f^{\alpha_i \cdot s_{1-i}} \cdot g^{r_i \cdot s_{1-i}} = h_{1-i}^{r_i}.$$

Notice that the term on the left-hand side is known to P_{1-i}, whereas $h_{1-i}^{r_i}$ is known to P_i. Using the NIDLS techniques, the parties are able to convert this divisive secret-sharing of $f^{\alpha_i \cdot s_{1-i}}$ into a subtractive secret-sharing of $\alpha_i \cdot s_{1-i}$. At this point, it is sufficient that party P_i add $\alpha_i \cdot s_i$ to its share.

On the NIZKs used in the setup procedure. The correctness of the HSS scheme requires the MAC key α_i and the share of the secret s_i to be bounded in size. In

particular, the NIZK used in the setup procedure need to prove that P_i knows $s_i \in [\ell_{\mathsf{sk}}]$, $\alpha_i \in [n_{\mathsf{out}}]$ and $r_i \in \mathbb{Z}$ such that $h_i = g^{s_i}$ and $A_i = f^{\alpha_i} \cdot g^{r_i}$.

We observe that g^{s_i} can be regarded as a commitment to s_i using the scheme in Fig. 10. The peculiarities are that the role of C and g has been switched and the randomness is set to 0. Party P_i can therefore use Π_{range} to prove that $s_i \in [\ell_{\mathsf{sk}}]$.

Proving the well-formedness of A_i is slightly more complex, but we can still design efficient solutions. For instance, P_i can commit to α_i and prove that its value belongs to $[0, n_{\mathsf{out}} - 1]$ using Π_{range}. Then, using procedures analogous to those used in Π_{plain}, it can show that the value hidden in A_i coincides with the committed one. For more details on this protocol, we refer to Appendix A.6.

Active HSS Scheme Π_{RMS} – One-Round Setup

SETUP:

1. The parties initialise $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ to obtain $\mathsf{par} := (\mathsf{par}', g, \rho_0, C, \rho_1)$.
2. Each party P_i samples $s_i \leftarrow [\ell_{\mathsf{sk}}]$ and $\alpha_i \xleftarrow{R} [n_{\mathsf{out}}]$.
3. Each party P_i sends $h_i \leftarrow g^{s_i}$ and $A_i \leftarrow f^{\alpha_i} \cdot g^{r_i}$ where $r_i \xleftarrow{R} [\ell]$.
4. For every $i \in \{0,1\}$, the parties call $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ with public input $(\mathsf{range}, A_i, B = n_{\mathsf{out}} - 1)$. The witness, provided by P_i, is (α_i, r_i). If $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ outputs 0, the parties abort.
5. For every $i \in \{0,1\}$, the parties call $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ with public input $(\mathsf{range}, h_i, B = \ell_{\mathsf{sk}} - 1)$ inverting the role of C and g. The witness, provided by P_i, is $(s_i, 0)$. If $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ outputs 0, the parties abort.
6. Each party P_i computes $h \leftarrow h_1/h_0$ and sets $\mathsf{pk} \leftarrow (\mathsf{par}', g, \rho, h)$.
7. The parties call the random oracle with input (h_0, h_1, A_0, A_1) to obtain elements $v_0, v_1 \in \mathbb{Z}_t$. Each party P_i computes

$$\alpha'_{i,i} \leftarrow (-1)^i \cdot \left(\mathsf{DDLog}_{\mathsf{par}}(h_{1-i}^{r_i}) - \alpha_i \cdot s_i\right) + v_i \bmod t,$$
$$\alpha'_{1-i,i} \leftarrow (-1)^i \cdot \mathsf{DDLog}_{\mathsf{par}}(A_{1-i}^{s_i}) + v_{1-i} \bmod t.$$

Fig. 19. The active HSS scheme for RMS programs – one-round setup

Theorem 9. *Assume that the parties are connected by an authenticated and private channel. Let n_{out} be a prime greater than 2^λ and assume that $t > 2^\lambda \cdot B \cdot \ell_{\mathsf{sk}} \cdot n_{\mathsf{out}}$. If the DDH assumption and the small exponent assumption hold, the protocol Π_{RMS} in Fig. 19 and Fig. 20 UC-implements the RMS functionality $\mathcal{F}_{\mathsf{RMS}}$ in the $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$-hybrid model with random oracle. Moreover, if $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ is implemented using only one round, the initialisation, the input procedure and the evaluations in Π_{RMS} require only one round.*

Proof. Observe that the simulator knows the view of the corrupted parties thanks to $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$. Indeed, if the protocol does not abort, the adversary pro-

Active HSS Scheme Π_{RMS} – Two-Round MPC

INPUT:

The i-th party can input a value $x \in [-n_{\mathsf{out}}/2, n_{\mathsf{out}}/2]$ as follows:

1. P_i sends $\mathsf{ct}_x \leftarrow \mathsf{EG.Enc}(\mathsf{pk}, x; r)$ where $r \xleftarrow{R} [\ell]$.
2. P_i sends $\mathsf{ct}_{xs} \leftarrow \mathsf{EG.SkEnc}(\mathsf{pk}, x; \overline{r})$ where $\overline{r} \xleftarrow{R} [\ell]$.
3. The parties call $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ with public input $(\mathsf{input}, \mathsf{ct}_x, \mathsf{ct}_{xs}, B = n_{\mathsf{out}}/2)$. The witness is $(x, r, \overline{r})$. If $\mathcal{F}_{\mathsf{Setup}}$ outputs 0, the parties abort.
4. The parties set $\mathsf{I}_x \leftarrow (\mathsf{ct}_x, \mathsf{ct}_{xs})$.

EVALUATION: The parties evaluate a RMS program P gate by gate as follows.

- $\mathsf{M}_x \leftarrow \mathsf{ConvertInput}(\mathsf{I}_x)$:
  The parties define a memory wire M_1. P_0 assigns $(0, s_0, \alpha_0, 0, \alpha'_{0,0}, \alpha'_{1,0})$ to M_1. P_1 assigns $(1, s_1, 0, \alpha_1, \alpha'_{1,0}, \alpha'_{1,1})$. The parties compute $\mathsf{M}_x \leftarrow \mathsf{Mult}(\mathsf{I}_x, \mathsf{M}_1)$.
- $\mathsf{M}_z \leftarrow \mathsf{Add}(\mathsf{M}_x, \mathsf{M}_y)$:
  Each party P_i computes $z_i \leftarrow x_i + y_i$, $z'_i \leftarrow x'_i + y'_i$ and, $\forall j \in \{0,1\}$,
  $$M_{j,i}(z) \leftarrow M_{j,i}(x) + M_{j,i}(y), \qquad M'_{j,i}(z) \leftarrow M'_{j,i}(x) + M'_{j,i}(y).$$
  Then, P_i assigns $\big(z_i, z'_i, M_{0,i}(z), M_{1,i}(z), M'_{0,i}(z), M'_{1,i}(z)\big)$ to M_z.
- $\mathsf{M}_z \leftarrow \mathsf{Mult}(\mathsf{I}_x, \mathsf{M}_y)$:
  Let $\mathsf{ct}_x = (c_0, c_1)$ and $\mathsf{ct}_{xs} = (d_0, d_1)$. Let id be the label of the gate. Each party P_i computes the following operations
  1. The parties queries the random oracle with $(h_0, h_1, A_0, A_1, P, \mathsf{I}^1, \ldots, \mathsf{I}^n, \mathsf{id})$ obtaining random $u_0, u_1, u_2, u_3 \in \mathbb{Z}_t$.
  2. $z_i \leftarrow (-1)^{1-i} \cdot \mathsf{DDLog}\big(c_1^{y_i} \cdot c_0^{-y'_i}\big) + u_0 \bmod t$
  3. $z'_i \leftarrow (-1)^{1-i} \cdot \mathsf{DDLog}\big(d_1^{y_i} \cdot d_0^{-y'_i}\big) + u_1 \bmod t$
  4. $\forall j \in \{0,1\}: \quad M_{j,i}(z) \leftarrow (-1)^{j-i} \cdot \mathsf{DDLog}\big(c_1^{M_{j,i}(y)} \cdot c_0^{-M'_{j,i}(y)}\big) + u_2 \bmod t$
  5. $\forall j \in \{0,1\}: \quad M'_{j,i}(z) \leftarrow (-1)^{j-i} \cdot \mathsf{DDLog}\big(d_1^{M_{j,i}(y)} \cdot d_0^{-M'_{j,i}(y)}\big) + u_3 \bmod t$

  Then, P_i assigns $\big(z_i, z'_i, M_{0,i}(z), M_{1,i}(z), M'_{0,i}(z), M'_{1,i}(z)\big)$ to M_z.
- $\mathsf{Output}(\mathsf{M}_z)$: Each party P_i performs the following operations:
  1. It sends $z_i \bmod n_{\mathsf{out}}$ and $M_{1-i,i}(z) \bmod n_{\mathsf{out}}$.
  2. It receives $z_{1-i} \bmod n_{\mathsf{out}}$ and $M_{i,1-i}(z) \bmod n_{\mathsf{out}}$.
  3. It sets $z \leftarrow z_1 - z_0 \bmod n_{\mathsf{out}}$
  4. If $M_{i,i}(z) = M_{i,1-i}(z) + \alpha_i \cdot z \bmod n_{\mathsf{out}}$, it outputs z, otherwise it aborts.

Fig. 20. The active HSS scheme for RMS programs – two-round MPC operations

vides all the witnesses needed for the simulation. Consider the following sequence of hybrids.

Hybrid 1. Suppose that P_i is honest. In this hybrid, the simulator sets $\alpha'_{i,i} \leftarrow \alpha'_{i,1-i} + \alpha_i \cdot (s_1 - s_0)$ and $\alpha'_{1-i,i} \leftarrow \alpha'_{1-i,1-i} - \alpha_{1-i} \cdot (s_1 - s_0)$. This hybrid is unconditionally indistinguishable from the real protocol. Indeed, for every $j \in \{0, 1\}$

$$A_j^{s_{1-j}} = f^{\alpha_j \cdot s_{1-j}} \cdot g^{r_j \cdot s_{1-j}} = f^{\alpha_j \cdot s_{1-j}} \cdot h_{1-j}^{r_j}.$$

So, after applying DDLog on $A_j^{s_{1-j}}$ and $h_{1-j}^{r_j}$, the parties obtain an additive secret-sharing over $\mathbb{Z}_t$ of $\alpha_j \cdot s_{1-j}$. After adjusting the signs, P_j adds $\alpha_j \cdot s_j$ to its value, obtaining a subtractive secret-sharing of $\alpha_j \cdot (s_1 - s_0)$.

The property is satisfied over $\mathbb{Z}_t$, however, we would like to obtain a subtractive secret-sharing over $\mathbb{Z}$. Observe that the shares are uniform over $\mathbb{Z}_t$ due to the rerandomisation with the random oracle. Since $t > 2^\lambda \cdot B \cdot \ell \cdot n_{\mathsf{out}} > 2^\lambda \cdot |\alpha_j \cdot (s_1 - s_0)|$, the probability that the difference between the shares wraps around t is smaller than $2^{-\lambda}$, so, it is negligible. Therefore, with overwhelming probability, $\alpha'_{j,j}$ is equal to $\alpha'_{j,1-j} + \alpha_j \cdot (s_1 - s_0)$ even over $\mathbb{Z}$. Notice that it is fundamental that the query to the random oracle includes the messages sent in the setup, so the adversary cannot pick its setup message as a function of the random oracle output, as this could allow them to cause wrap-arounds and compromise the correctness of the result.

Hybrid 2. Suppose that P_i is honest. In this hybrid, the simulator sets $A_i \leftarrow g^{r_i}$ instead of $A_i \leftarrow f^x \cdot g^{r_i}$. It simulates the following execution of $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ by sending 1 to the adversary. This hybrid is indistinguishable from the previous one. Indeed, g^{r_i} is statistically close to uniform over $\langle g \rangle$ and $f \in \langle g \rangle$.

Hybrid 3. Consider any evaluation of a B-bounded RMS program P with n inputs. Let $x_1, \ldots, x_n$ be the values of the input wires. Notice that, for the corrupted parties, these elements are provided to $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ by the adversary, so the simulator knows them. Consider the j-th output gate M_z of P for any j and let z be the j-th entry of $P(x_1, \ldots, x_n)$. In this hybrid, if both parties are honest, the simulator sets $z_1 \leftarrow z_0 + z$ and, for every $i \in \{0, 1\}$, $M_{i,i}(z) \leftarrow M_{i,1-i}(z) + \alpha_{i,i} \cdot z$. If instead P_j is corrupted, the simulator selects $z_{1-j}, M_{1-j,1-j}(z)$ and $M'_{j,1-j}(z)$ so that $z_1 - z_0 = z$ and $M_{i,i}(z) = M_{i,1-i}(z) + \alpha_{i,i} \cdot z$ for every $i \in \{0, 1\}$.

This hybrid is unconditionally indistinguishable from Hybrid 2. In order to understand why, let us denote the value of the memory wire M_y in $P(x_1, \ldots, x_n)$ by y. For every M_y, we have the following $\mathbb{Z}$-linear relations

$$\begin{aligned} &y_1 - y_0 = y, && y'_1 - y'_0 = y \cdot (s_1 - s_0), \\ &M_{i,i}(y) - M_{i,1-i}(y) = \alpha_i \cdot y, && M'_{i,i}(y) - M'_{i,1-i}(y) = \alpha_i \cdot y \cdot (s_1 - s_0). \end{aligned}$$

This condition is clearly true for the memory value M_1, whose value is 1. It is straightforward to see that if the property described above holds for memory wires M_x and M_y, the condition is also satisfied by their addition.

For multiplications, notice that, for every $(c_0, c_1) = (g^r, f^x \cdot h^x), (d_0, d_1) = (g^{\bar{r}} \cdot f^{-x}, h^{\bar{r}})$ and v_0, v_1, v'_0, v'_1 such that $v'_1 - v'_0 = (s_1 - s_0) \cdot v$ where $v := v_1 - v_0$,

we have

$$\begin{aligned}
c_1^{v_1} \cdot c_0^{-v_1'} &= c_1^{v_0+v} \cdot c_0^{-v_0'-(s_1-s_0)\cdot v} = \\
&= c_1^{v_0} \cdot c_0^{-v_0'} \cdot f^{x\cdot v} \cdot h^{r\cdot v} \cdot g^{-r\cdot v\cdot(s_1-s_0)} = \\
&= f^{x\cdot v} \cdot c_1^{v_0} \cdot c_0^{-v_0'}, \\
d_1^{v_1} \cdot d_0^{-v_1'} &= d_1^{v_0+v} \cdot d_0^{-v_0'-(s_1-s_0)\cdot v} = \\
&= d_1^{v_0} \cdot d_0^{-v_0'} \cdot h^{\overline{r}\cdot v} \cdot g^{-\overline{r}\cdot v\cdot(s_1-s_0)} \cdot f^{x\cdot v\cdot(s_1-s_0)} = \\
&= f^{x\cdot v\cdot(s_1-s_0)} \cdot d_1^{v_0} \cdot d_0^{-v_0'}.
\end{aligned}$$

So, by applying DDLog on $c_1^{v_1} \cdot c_0^{-v_1'}$ and $c_1^{v_0} \cdot c_0^{-v_0'}$ and on $d_1^{v_1} \cdot d_0^{-v_1'}$ and $d_1^{v_0} \cdot d_0^{-v_0'}$, the parties obtain an additive secret-sharing over $\mathbb{Z}_t$ of $x \cdot v$ and $x \cdot v \cdot (s_1 - s_0)$, respectively.

So, we conclude that the output wire M_z of the multiplication between I_x and M_y is associated with elements satisfying

$$\begin{aligned}
&z_1 - z_0 = x \cdot y, && z_1' - z_0' = x \cdot y \cdot (s_1 - s_0), \\
&M_{i,i}(z) - M_{i,1-i}(z) = \alpha_i \cdot y \cdot x, && M'_{i,i}(z) - M'_{i,1-i}(z) = \alpha_i \cdot y \cdot x \cdot (s_1 - s_0).
\end{aligned}$$

The operations described above hold over $\mathbb{Z}_t$, so they equalities may not hold over $\mathbb{Z}$ when the shares are particularly close to t (i.e. when the operations wrap around t). Notice however that, due to the rerandomisation using the random oracle, the shares of the subtractive secret-sharings are all uniform over $\mathbb{Z}_t$. Since $t > 2^\lambda \cdot B \cdot \ell \cdot n_{\mathsf{out}} > 2^\lambda \cdot |x \cdot y \cdot \alpha_i \cdot (s_1 - s_0)|$, the probability that a wrap-around occurs is negligible, so, the equalities above hold also over the integers. Notice that it is fundamental that the query to the random oracle includes the RMS program and the input messages, so the adversary cannot pick its message as a function of the random oracle output, as this could allow them to cause wrap-arounds and compromise the correctness of the result.

Hybrid 4. Let P_i be a honest party. In this hybrid, the simulator samples s_i uniformly over $[\ell]$ and sets $h_i \leftarrow g^s$. This hybrid is indistinguishable from the previous one by the small exponent assumption.

Hybrid 5. In this hybrid, the simulator sets $\mathsf{ct}_x \xleftarrow{R} \mathsf{EG.Enc}(\mathsf{pk}, 0)$, for every input provided by honest parties. The subsequent verification of $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ is simulated by simply sending 1 to the adversary. This hybrid is indistinguishable from the previous one by the IND-CPA security of ElGamal. Suppose that P_1 is a honest party. In the reduction, after receiving an ElGamal public key h' and a ciphertext $\mathsf{ct} = (c_0, c_1)$, we set $h_1 \leftarrow h'$ and we wait for the adversary to provide s_0 to $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$. Then, we set $\mathsf{ct}_x \leftarrow (c_0, c_1 \cdot c_0^{-s_0})$. Notice that if $(c_0, c_1) = (g^r, f^x \cdot h'^r)$, we have that $c_0^{-s_0} = h_0^{-r}$ and so, $c_1 \cdot c_0^{-s_0} = f^x \cdot h_1^r \cdot h_0^{-r} = f^x \cdot h^r$. A similar reduction can be done when P_0 is honest (we need to ask for the encryption of $-x$ instead of x).

Hybrid 6. In this hybrid, the simulator sets $\mathsf{ct}_{xs} \leftarrow (g^r, g^s)$ where $r, s \xleftarrow{R} [\ell]$, for every input provided by honest parties. The subsequent verification of

$\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$ is simulated by simply sending 1 to the adversary. This hybrid is indistinguishable from the previous one by the DDH assumption. Suppose that P_1 is a honest party. In the reduction, after receiving a triple (g^a, g^b, g^c) where $c = a \cdot b$ or c is random, we set $h_1 \leftarrow g^a$ and we wait for the adversary to provide s_0 to $\mathcal{F}_{\mathsf{NIDLS\text{-}ZK}}$. Then, we set $\mathsf{ct}_{xs} \leftarrow (g^b \cdot f^{-x}, g^c \cdot (g^b)^{-s_0})$. Notice that if $c = a \cdot b$, we have that $g^c \cdot (g^b)^{-s_0} = (g^a \cdot g^{-s_0})^b = (h_1/h_0)^b$. If c is random, $g^c \cdot (g^b)^{-s_0}$ is statistically close to uniform over $\langle g \rangle$ and independent of b. Since $f \in \langle g \rangle$ and g^b is also statistically close to uniform over $\langle g \rangle$, no adversary can distinguish between $(g^b \cdot f^{-x}, g^s)$ and (g^r, g^s). A similar reduction can be done when P_0 is honest.

Hybrid 7. Let P_i be a corrupted party. In this hybrid, the simulator lets the functionality generate the outputs. Moreover, for every output wire M_z, the simulator computes $z_i \bmod n_{\mathsf{out}}$ and $M_{1-i,i}(z) \bmod n_{\mathsf{out}}$. If, in the only round of the evaluation, the adversary sends different values, the simulator makes the protocol abort. Observe that the view of the adversary is independent of α_{1-i}. Suppose that the adversary sends z_i' instead of $z_i \bmod n_{\mathsf{out}}$, so $\epsilon := z_i' - z_i \neq 0 \bmod n_{\mathsf{out}}$. In order to make the final check of the honest party pass, the adversary needs to send $M_{1-i,i}(z) - \alpha_{1-i} \cdot \epsilon$ instead of $M_{1-i,i}(z)$. Notice however, that $\alpha_{1-i} \cdot \epsilon$ is uniformly distributed over $\mathbb{Z}_{n_{\mathsf{out}}}$ from the adversary's perspective. Therefore, the probability that the check succeeds is $1/n_{\mathsf{out}} < 2^{-\lambda}$, which is negligible.

If the adversary sends $z_i \bmod n_{\mathsf{out}}$ along with a wrong MAC, the protocol aborts with probability 1. We have proven that the interaction between the functionality and the simulator in Hybrid 7 is indistinguishable from the real protocol. □

Acknowledgments. The authors would like to thank Guilhem Castagnos and Fabien Laguillaumie for their crucial clarifications about the CL framework and class groups, Lasse Rønne Møller for important observations on the DXDH assumption and the anonymous reviewers for their feedback.

Research supported by: the Concordium Blockhain Research Center, Aarhus University, Denmark; the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC); the Independent Research Fund Denmark (DFF) under project number 0165-00107B (C3PO); the Aarhus University Research Foundation; and the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0085. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA). Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

References

BCG+17. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: Optimizations and applications. In *ACM CCS*

2017. ACM Press, October / November 2017.

BCG+19. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *CRYPTO 2019, Part III*, LNCS. Springer, Heidelberg, August 2019.

BCG+20. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In *61st FOCS*. IEEE Computer Society Press, November 2020.

BDOZ11. Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT 2011*, LNCS. Springer, Heidelberg, May 2011.

BGI16. Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In *CRYPTO 2016, Part I*, LNCS. Springer, Heidelberg, August 2016.

BGI17. Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In *EUROCRYPT 2017, Part II*, LNCS. Springer, Heidelberg, April / May 2017.

BHJL17. Fabrice Benhamouda, Javier Herranz, Marc Joye, and Benoît Libert. Efficient cryptosystems from 2^k-th power residue symbols. *Journal of Cryptology*, (2), April 2017.

BKS19. Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In *EUROCRYPT 2019, Part II*, LNCS. Springer, Heidelberg, May 2019.

CDE+18. Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPD $\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In *CRYPTO 2018, Part II*, LNCS. Springer, Heidelberg, August 2018.

CJLN09. Guilhem Castagnos, Antoine Joux, Fabien Laguillaumie, and Phong Q. Nguyen. Factoring pq^2 with quadratic forms: Nice cryptanalyses. In *ASIACRYPT 2009*, LNCS. Springer, Heidelberg, December 2009.

CKLR21. Geoffroy Couteau, Michael Klooß, Huang Lin, and Michael Reichle. Efficient range proofs with transparent setup from bounded integer commitments. In *EUROCRYPT 2021, Part III*, LNCS. Springer, Heidelberg, October 2021.

CL15. Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from DDH. In *CT-RSA 2015*, LNCS. Springer, Heidelberg, April 2015.

DF02. Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT 2002*, LNCS. Springer, Heidelberg, December 2002.

DJ01. Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *PKC 2001*, LNCS. Springer, Heidelberg, February 2001.

DJ03. Ivan Damgård and Mads Jurik. A length-flexible threshold cryptosystem with applications. In *ACISP 03*, LNCS. Springer, Heidelberg, July 2003.

DP92. Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). In *33rd FOCS*. IEEE Computer Society Press, October 1992.

GMW86. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th FOCS*. IEEE Computer Society Press, October 1986.

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

Gro05. Jens Groth. Non-interactive zero-knowledge arguments for voting. In *ACNS 05*, LNCS. Springer, Heidelberg, June 2005.

GRSB19. Sharon Goldberg, Leonid Reyzin, Omar Sagga, and Foteini Baldimtsi. Efficient noninteractive certification of RSA moduli and beyond. In *ASIACRYPT 2019, Part III*, LNCS. Springer, Heidelberg, December 2019.

IPS08. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO 2008*, LNCS. Springer, Heidelberg, August 2008.

JL13. Marc Joye and Benoît Libert. Efficient cryptosystems from 2^k-th power residue symbols. In *EUROCRYPT 2013*, LNCS. Springer, Heidelberg, May 2013.

NNOB12. Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *CRYPTO 2012*, LNCS. Springer, Heidelberg, August 2012.

OSY21. Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of paillier: Homomorphic secret sharing and public-key silent OT. In *EUROCRYPT 2021, Part I*, LNCS. Springer, Heidelberg, October 2021.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, LNCS. Springer, Heidelberg, May 1999.

PT00. Sachar Paulus and Tsuyoshi Takagi. A new public-key cryptosystem over a quadratic order with quadratic decryption time. *Journal of Cryptology*, (2), March 2000.

RS21. Lawrence Roy and Jaspal Singh. Large message homomorphic secret sharing from DCR and applications. In *CRYPTO 2021, Part III*, LNCS. Springer, Heidelberg, August 2021.

Tuc20. Ida Tucker. *Functional encryption and distributed signatures based on projective hash functions, the benefit of class groups. (Chiffrement fonctionnel et signatures distribuées fondés sur des fonctions de hachage à projection, l'apport des groupes de classe)*. PhD thesis, University of Lyon, France, 2020.

A Sigma protocols in the NIDLS framework

In this appendix, we present public-coin honest-verifier zero-knowledge protocols that can be used in our actively secure constructions. All the solutions we describe work in the NIDLS framework without introducing instantiation-dependent assumptions. For this reason, we had to rely on binary challenges, which have the disadvantage of increasing the complexity by a factor of λ. We highlight anyway, that, when tackling specific NIDLS instantiations, the protocols described below can usually be substituted with more efficient constructions, for instance using integral challenges.

A.1 Proof of knowledge of opening

In Fig. 21, we present a sigma protocol that, given a commitment X, proves the knowledge of $x, r \in \mathbb{Z}$ such that $X = C^x \cdot g^r$. Sigma protocols for this kind of relations are well known in cryptography: the prover starts by sending a random commitment $Y := C^y \cdot g^\mu$, the verifier replies with a random challenge $e \in \mathbb{Z}$, which the prover replies with $z := y + e \cdot x$ and $u := \mu + e \cdot r$. The verifier accepts if $Y \cdot X^e = C^z \cdot g^u$.

To adapt this to the NIDLS framework some modifications are necessary. First of all, since the order of C and g are unknown, the values z and u sent by the prover in the last round are integers. So, we obtain honest-verifier zero-knowledge (HVZK) only if y and μ have sufficiently high entropy to hide the witness (x, r). For this reason, given bounds B and ℓ on $|x|$ and $|r|$, we sample y uniformly over $[2^\lambda \cdot B]$ and μ uniformly over $[2^\lambda \cdot \ell]$. Notice that we achieve HVZK only if $|x| < B$ and $|r| < \ell$. When we use this protocol in our constructions, it is fundamental to choose sufficiently large B and ℓ_r, so that the protocol leaks no information about the statements of honest parties.

The other difference with respect to the sigma protocol described above is that we use binary challenges, i.e. $e \in \{0, 1\}$. In order to achieve proper soundness, we therefore need to repeat the protocol λ times in parallel[5]. That blows up the complexity by a factor of λ. The reason why we need binary challenges is the fact that we want constructions that are secure independently on the instantiation of the NIDLS framework. Specifically, when we deal with integral challenges, it is possible to prove that the adversary knows $e', x', u' \in \mathbb{Z}$ such that $X^{e'} = C^{x'} \cdot g^{u'}$. In NIDLS instantiations, the order of C and g is unknown, so, if $e' \neq 1$, it is impossible for us to divide x' and u' by e' and obtain x and u such that $X = C^x \cdot g^u$. Although there usually are ways around this issue, these often rely on instantiation-dependent assumptions (such as the strong root problem and the low order assumption for class groups).

Theorem 10. *The construction Π_{com} in Fig. 21 is a complete and 2-special sound proof for the relation*

$$\mathcal{R}_{\mathsf{com}} := \{X, (x, r) | X = C^x \cdot g^r\}.$$

[5] We can do this because we care only about HVZK.

Commitment ZK Proof Π_{com}

CRS:

1. $\mathsf{par}' := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(\mathbb{1}^\lambda)$
2. $(g, \rho_0) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
3. $(C, \rho_1) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
4. If $C = g$, go to step 3.
5. Output $\mathsf{par} := (\mathsf{par}', g, \rho_0, C, \rho_1)$

STATEMENT: $X \in G$
WITNESS: $x, r \in \mathbb{Z}$ such that $X = C^x \cdot g^r$.
PROTOCOL:

1. The prover samples $y_i \xleftarrow{R} [2^\lambda \cdot B]$, $\mu_i \xleftarrow{R} [2^\lambda \cdot \ell_r]$ for every $i \in [\lambda]$.
2. The prover sends $Y_i := C^{y_i} \cdot g^{\mu_i}$ for every $i \in [\lambda]$.
3. The verifier sends $\boldsymbol{e} \xleftarrow{R} \{0,1\}^\lambda$.
4. The prover sends $z_i \leftarrow y_i + e_i \cdot x$ and $u_i \leftarrow \mu_i + e_i \cdot r$ for every $i \in [\lambda]$.
5. The verifier accepts if $Y_i \cdot X^{e_i} = C^{z_i} \cdot g^{u_i}$ for every $i \in [\lambda]$.

Fig. 21. Proof of knowledge of opening

If the witness (x, r) satisfies $x \in [-B, B]$ and $r \in [-\ell_r, \ell_r]$, the proof is also honest-verifier zero-knowledge.

Proof. Completeness. It is straightforward to see that if $X = C^x \cdot g^r$,

$$Y_i \cdot X^{e_i} = C^{y_i} \cdot g^{\mu_i} \cdot C^{e_i \cdot x} \cdot g^{e_i \cdot r} = C^{y_i + e_i \cdot x} \cdot g^{\mu_i + e_i \cdot r} = C^{z_i} \cdot g^{u_i}.$$

Special Soundness. Suppose that we have two accepting transcripts with coinciding first round messages and different challenges $\boldsymbol{e}$ and $\boldsymbol{e}'$. Let $(\boldsymbol{z}, \boldsymbol{u})$ and $(\boldsymbol{z}', \boldsymbol{u}')$ be the corresponding last round messages. We know that there exists an $i \in [\lambda]$ such that $e_i = 1$ and $e'_i = 0$, or $e_i = 0$ and $e'_i = 1$. Without loss of generality, we assume the first one (in the other case, we just switch the role of two transcripts).

We know that

$$Y_i \cdot X = C^{z_i} \cdot g^{u_i}, \qquad \text{and} \qquad Y_i = C^{z'_i} \cdot g^{u'_i}.$$

We conclude that $X = C^{z_i - z'_i} \cdot g^{u_i - u'_i}$.

Honest-Verifier Zero-Knowledge. Assume that $X = C^x \cdot g^r$ for some values $x \in [-B, B]$ and $r \in [-\ell_r, \ell_r]$. The simulator samples a random challenge $\boldsymbol{e} \in \{0,1\}^\lambda$ and values $z_i \xleftarrow{R} [2^\lambda \cdot B]$ and $u_i \xleftarrow{R} [2^\lambda \cdot \ell_r]$ for every $i \in [\lambda]$. Finally, it computes $Y_i \leftarrow C^{z_i} \cdot g^{u_i} / X^{e_i}$ for every $i \in [\lambda]$.

The only difference between the transcript in the protocol and in the simulation is the distribution of $\boldsymbol{z}$ and $\boldsymbol{u}$. The total statistical distance between the two cases is however upper bounded by $2\lambda/2^{\lambda}$, which is negligible. So the distributions are indistinguishable. Notice that $\boldsymbol{e}$ is distributed as in the protocol, whereas, in both cases, Y_i is the only value for which $Y_i \cdot X^{e_i} = C^{z_i} \cdot g^{u_i}$.

□

A.2 Multiplication proof

The protocol Π_{mult} described in Fig. 22 is used to convince a verifier holding commitments X, Y and Z, that the prover knows $x, y, z \in \mathbb{Z}$ and $r_1, r_2, r_3 \in \mathbb{Z}$ such that $X = C^x \cdot g^{r_1}$, $Y = C^y \cdot g^{r_2}$, $Z = C^z \cdot g^{r_3}$ and $z = x \cdot y$. Our sigma protocol Π_{mult} uses the same techniques as [DF02]. Specifically, after proving the knowledge of (x, r_1) using the protocol from Appendix A.1, the prover sends a commitment $U := C^{\omega} \cdot g^{\mu}$ and $W := X^{\omega} \cdot g^{\nu}$ where ω, μ and ν are random. The verifier sends then a challenge e, which the prover replies with $w := \omega + e \cdot y$, $u := \mu + e \cdot r_2$ and $v := \nu + e \cdot (r_3 - y \cdot r_1)$. The verifier accepts if $U \cdot Y^e = C^w \cdot g^u$ and $W \cdot Z^e = X^w \cdot g^v$. Indeed, notice that, if the prover is honest,

$$W \cdot Z^e = X^{\omega} \cdot g^{\nu} \cdot C^{x \cdot y \cdot e} \cdot g^{r_3 \cdot e} = X^{\omega} \cdot g^{\nu} \cdot (X^{y \cdot e} \cdot g^{-r_1 \cdot y \cdot e}) \cdot g^{r_3 \cdot e} = X^w \cdot g^v.$$

Once again, we rely on binary challenges and repeat the proof λ times in parallel, so that achieve security independently of the NIDLS instantiation. As before, that blows up the complexity by a factor of λ. Moreover, since the order of C and g may be unknown, we sample random ω, μ and ν that are 2^{λ} times bigger than $|y|$, $|r_2|$ and $|r_3 - y \cdot r_1|$ respectively. When the witnesses do not satisfy the expected bounds, the protocol is not guaranteed to satisfy HVZK.

Finally, we notice that if $X = Y$, the prover does not need to use the proof described in the previous paragraph to prove the knowledge of (x, r). So, proving squarings is slightly more efficient than proving multiplications.

Theorem 11. *The construction Π_{mult} described in Fig. 22 is a complete and special-sound proof for the relation*

$$\mathcal{R}_{\mathsf{mult}} := \left\{ \begin{array}{c|l} (X, Y, Z) & z = x \cdot y \\ (x, y, z, r_1, r_2, r_3) & X = C^x \cdot g^{r_1} \\ & Y = C^y \cdot g^{r_2} \\ & Z = C^z \cdot g^{r_3} \end{array} \right\}$$

Moreover, if $x, y \in [-B, B]$, $r_1, r_2 \in [\ell_r, \ell_r]$ and $r_3 \in [-\ell'_r, \ell'_r]$, Π_{mult} is also honest-verifier zero-knowledge.

Proof. Completeness. We know that by the completeness of Π_{com}, the verifier does not reject after the first 3 rounds. Moreover, we have

$$U_i \cdot Y^{e_i} = C^{\omega_i} \cdot g^{\mu_i} \cdot C^{e_i \cdot y} \cdot g^{e_i \cdot r_2} = C^{\omega_i + e_i \cdot y} \cdot g^{\mu_i + e_i \cdot r_2} = C^{w_i} \cdot g^{u_i}$$

Multiplication ZK Proof Π_{mult}

CRS:

1. $\mathsf{par}' := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(1\!\!1^\lambda)$
2. $(g, \rho_0) \xleftarrow{R} \mathcal{D}(1\!\!1^\lambda, \mathsf{par})$
3. $(C, \rho_1) \xleftarrow{R} \mathcal{D}(1\!\!1^\lambda, \mathsf{par})$
4. If $C = g$, go to step 3.
5. Output $\mathsf{par} := (\mathsf{par}', g, \rho_0, C, \rho_1)$

STATEMENT: $X, Y, Z \in G$
WITNESS: $x, y, z \in \mathbb{Z}$ and $r_1, r_2, r_3 \in \mathbb{Z}$ such that $X = C^x \cdot g^{r_1}$, $Y = C^y \cdot g^{r_2}$ and $Z = C^z \cdot g^{r_3}$.
PROTOCOL:

1. If $X \neq Y$, use Π_{com} with input X and witness (x, r_1), to prove the knowledge of x and r_1.
2. The prover samples $\omega_i \xleftarrow{R} [2^\lambda \cdot B]$, $\mu_i \xleftarrow{R} [2^\lambda \cdot \ell_r]$ and $\nu_i \xleftarrow{R} [2^\lambda \cdot \max(\ell'_r, B \cdot \ell_r)]$ for every $i \in [\lambda]$.
3. The prover sends $U_i := C^{\omega_i} \cdot g^{\mu_i}$ and $W_i = X^{\omega_i} \cdot g^{\nu_i}$ for every $i \in [\lambda]$.
4. The verifier sends $\boldsymbol{e} \xleftarrow{R} \{0,1\}^\lambda$.
5. The prover sends $w_i \leftarrow \omega_i + e_i \cdot y$ and $u_i \leftarrow \mu_i + e_i \cdot r_2$ and $v_i \leftarrow \nu_i + e_i \cdot (r_3 - y \cdot r_1)$ for every $i \in [\lambda]$.
6. The verifier accepts if $U_i \cdot Y^{e_i} = C^{w_i} \cdot g^{u_i}$ and $W_i \cdot Z^{e_i} = X^{w_i} \cdot g^{v_i}$ for every $i \in [\lambda]$.

Fig. 22. Honest-verifier zero-knowledge proof for multiplications

and

$$\begin{aligned} W_i \cdot Z^{e_i} &= X^{\omega_i} \cdot g^{\nu_i} \cdot C^{e_i \cdot z} \cdot g^{e_i \cdot r_3} = \\ &= X^{\omega_i} \cdot g^{\nu_i} \cdot X^{e_i \cdot y} \cdot g^{-e_i \cdot y \cdot r_1} \cdot g^{e_i \cdot r_3} = \\ &= X^{\omega_i + e_i \cdot y} \cdot g^{\nu_i + e_i \cdot (r_3 - y \cdot r_1)} = X^{w_i} \cdot g^{v_i}. \end{aligned}$$

So the verifier never rejects when the parties are all honest.

Special Soundness. If $X \neq Y$, by the special soundness of Π_{com}, we can extract x and r_1 such that $X = C^x \cdot g^{r_1}$. In order to do that, we need two accepting transcripts with coinciding first round message and different second-round challenges.

Now, suppose we have two additional accepting transcripts with coinciding messages in the first 3 rounds and different challenges $\boldsymbol{e}$ and $\boldsymbol{e}'$ in the fourth round. Let $(\boldsymbol{w}, \boldsymbol{u}, \boldsymbol{v})$ and $(\boldsymbol{w}', \boldsymbol{u}', \boldsymbol{v}')$ be the corresponding last round messages. We know that there exists an $i \in [\lambda]$ such that $e_i = 1$ and $e'_i = 0$, or $e_i = 0$ and

$e'_i = 1$. Without loss of generality, we assume the first one (in the other case, we just switch the role of two transcripts).

We have that

$$U_i = C^{w'_i} \cdot g^{u'_i} \qquad \text{and} \qquad U_i \cdot Y = C^{w_i} \cdot g^{u_i},$$

so $Y = C^{w_i - w'_i} \cdot g^{u_i - u'_i}$. Moreover, $W_i = X^{w'_i} \cdot g^{v'_i}$ and $W_i \cdot Z = X^{w_i} \cdot g^{v_i}$, so

$$Z = X^{w_i - w'_i} \cdot g^{v_i - v'_i} = C^{x \cdot (w_i - w'_i)} \cdot g^{r_1 \cdot (w_i - w'_i)} \cdot g^{v_i - v'_i}.$$

The extractor can therefore set $y := w_i - w'_i$, $z := x \cdot y$ or $z := y^2$ if $X = Y$, $r_2 := u_i - u'_i$ and $r_3 := r_2 + r_1 \cdot y$ and output the witness (x, y, z, r_1, r_2, r_3).

Honest-Verifier Zero-Knowledge. Suppose that $x, y \in [-B, B]$, $r_1, r_2 \in [-\ell_r, \ell_r]$ and $r_3 \in [-\ell'_r, \ell'_r]$.

Hybrid 1. By the honest-verifier zero-knowledge of Π_{com}, we can consider the hybrid in which we substitute the messages in the first three rounds with simulated ones. Clearly, the adversary cannot distinguish between Hybrid 1 and the real protocol.

Hybrid 2. Now, consider the case in which the simulator takes a random challenge $\boldsymbol{e} \xleftarrow{R} \{0,1\}^\lambda$ and samples $w_i \xleftarrow{R} [2^\lambda \cdot B]$, $u_i \xleftarrow{R} [2^\lambda \cdot \ell_r]$ and $v_i \xleftarrow{R} [2^\lambda \cdot \max(\ell'_r, B \cdot \ell_r)]$ for every $i \in [\lambda]$. The simulator then sets $U_i \leftarrow C^{w_i} \cdot g^{u_i} / Y^{e_i}$ and $W_i \leftarrow X^{w_i} \cdot g^{v_i} / Z^{e_i}$ for every $i \in [\lambda]$.

The only difference between Hybrid 1 and Hybrid 2 is the distribution of w_i, u_i and v_i. In all three cases, the statistical distance is upper bounded by $2 \cdot 2^{-\lambda}$, which is negligible. Hence, no adversary can distinguish between Hybrid 1 and Hybrid 2. Notice that in this stage, the transcript is generated without using the witness. □

A.3 Range proof

The protocol Π_{range} described in Fig. 23, we present a public-coin honest-verifier zero-knowledge range proof. Specifically, given a commitment X and a bound $B \in \mathbb{N}$, the protocol proves the knowledge of $x \in [0, B]$ and $r \in \mathbb{Z}$ such that $X = C^x \cdot g^r$. Our solution is based on the following observation by Groth [Gro05].

Lemma 4. *Let $B \in \mathbb{N}$ be a positive integer, a value $x \in \mathbb{Z}$ belongs to $[0, B]$ if and only if there exist integers $x_1, x_2, x_3 \in \mathbb{Z}$ such that*

$$1 + 4x \cdot (B - x) = \sum_{i=1}^{3} x_i^2.$$

Moreover, if that is the case, $x_i \in [-2B, 2B]$ for every $i \in \{1, 2, 3\}$.

Moreover, the elements x_1, x_2 and x_3 are efficiently computable. So, in order to prove that $x \in [0, B]$, the prover just needs to commit to x_1, x_2, x_3 and x_1^2, x_2^2, x_3^2, x^2. Using multiplication proofs (or better, squaring proofs), it shows

Range Proof Π_{range}

CRS:

1. $\mathsf{par}' := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(\mathbb{1}^\lambda)$
2. $(g, \rho_0) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
3. $(C, \rho_1) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
4. If $C = g$, go to step 3.
5. Output $\mathsf{par} := (\mathsf{par}', g, \rho_0, C, \rho_1)$

STATEMENT: $X \in G$ and $B \in \mathbb{N}$
WITNESS: $x \in [0, B]$ and $r \in \mathbb{Z}$ such that $X = C^x \cdot g^r$.
PROTOCOL:

1. Compute $x_1, x_2, x_3 \in [0, 2B]$ such that $1 + 4x \cdot (B - x) = x_1^2 + x_2^2 + x_3^2$.
2. The prover samples $r_1, r_2, r_3, s, s_2, s_3 \xleftarrow{R} [\ell]$ and $s_1 \xleftarrow{R} [2^\lambda \cdot \max(\ell, B \cdot \ell_r)]$.
3. The prover sends $S \leftarrow C^{x^2} \cdot g^s$ and $X_i \leftarrow C^{x_i} \cdot g^{r_i}$ and $S_i \leftarrow C^{x_i^2} \cdot g^{s_i}$ for $i = 1, 2, 3$.
4. Use Π_{mult} to prove that S is a commitment to x^2 and S_i is a commitment to x_i^2 for $i = 1, 2, 3$.
5. The prover sends $w \leftarrow 4s - 4B \cdot r + s_1 + s_2 + s_3$.
6. The verifier accepts if $S_1 \cdot S_2 \cdot S_3 \cdot S^4 \cdot X^{-4B} = C \cdot g^w$.

Fig. 23. Honest-verifier zero-knowledge range proof

the correctness of these commitments. Finally, using the linear properties of the scheme, it derives a commitment to $x_1^2 + x_2^2 + x_3^2 + 4x^2 - 4B \cdot x$ and proves that it opens to 1.

Our solution is honest-verifier zero-knowledge as long as the witness r belongs to $[-\ell_r, \ell_r]$ for some public bound $\ell_r \in \mathbb{N}$. When we use this protocol in our constructions, it is fundamental to choose a sufficiently large ℓ_r, so that the protocol leaks no information about the statements of honest parties.

Theorem 12. *If the commitment scheme in Fig. 10 is hiding and binding, the construction Π_{range} described in Fig. 23 is a complete and special-sound proof for the relation*

$$\mathcal{R}_{\mathsf{range}} := \left\{ (X, B), (x, r) \middle| x \in [0, B], X = C^x \cdot g^r \right\}.$$

Moreover, if $r \in [-\ell_r, \ell_r]$, Π_{range} is honest-verifier zero-knowledge.

Proof. Completeness. Since $x \in [0, B]$, it is possible for the prover to compute in polynomial time integers $x_1, x_2, x_3 \in [0, 2B]$ such that $1 + 4x \cdot (B - x) = x_1^2 + x_2^2 + x_3^2$. By the completeness of Π_{mult}, the verifier does not reject in Π_{mul}. Moreover, we have that

$$S_1 \cdot S_2 \cdot S_3 \cdot S^4 \cdot X^{-4B} = C^{x_1^2 + x_2^2 + x_3^2 + 4x^2 - 4B \cdot x} \cdot g^{s_1 + s_2 + s_3 + 4s - 4B \cdot r} = C \cdot g^w.$$

Special Soundness. By the special soundness of Π_{mult}, the extractor is able to obtain integers x, x_1, x_2, x_3 and $r, r_1, r_2, r_3, s_1, s_2, s_3$ such that

$$X = C^x \cdot g^r, \quad X_1 = C^{x_1} \cdot g^{r_1}, \quad X_2 = C^{x_2} \cdot g^{r_2}, \quad X_3 = C^{x_3} \cdot g^{r_3},$$
$$S = C^{x^2} \cdot g^s, \quad S_1 = C^{x_1^2} \cdot g^{s_1}, \quad S_2 = C^{x_2^2} \cdot g^{s_2}, \quad S_3 = C^{x_3^2} \cdot g^{s_3}.$$

Now, if we consider any of the accepting transcripts obtained by the extractor, we have that $S_1 \cdot S_2 \cdot S_3 \cdot S^4 \cdot X^{-4B} = C \cdot g^w$, hence

$$C^{x_1^2+x_2^2+x_3^2+4x^2-4B\cdot x} \cdot g^{s_1+s_2+s_3+4s-4B\cdot r} = C \cdot g^w.$$

If $x_1^2 + x_2^2 + x_3^2 + 4x^2 - 4B \cdot x \neq 1$, we would know how to open the commitment $C \cdot g^w$ to two different values. That contradicts the binding properties of the commitment scheme, hence

$$x_1^2 + x_2^2 + x_3^2 = 1 + 4x \cdot (B - x).$$

By lemma 4, we conclude that $x \in [0, B]$.

Honest-Verifier Zero Knowledge. Assume that $r \in [-\ell_r, \ell_r]$. We proceed by a series of indistinguishable hybrids.

Hybrid 1. In this hybrid, we generate the transcript used in Π_{mult} using the correspondent simulator. By the honest-verifier zero-knowledge of Π_{mult}, no adversary can distinguish between the real transcript from the one in Hybrid 1.

Hybrid 2. In this hybrid, the simulator samples $w \xleftarrow{R} [2^\lambda \cdot \max(\ell, B \cdot \ell_r)]$. Then, it sets $S_1 \leftarrow X^{4B} \cdot C \cdot g^w / (S_2 \cdot S_3 \cdot S^4)$. The rest of the transcript is generated as in Hybrid 1. The only difference between this hybrid and the previous one is the distribution of w. The statistical distance between the two cases is however dominated by $10 \cdot 2^{-\lambda}$ which is negligible. So, no adversary can distinguish between Hybrid 1 and Hybrid 2.

Hybrid 3. In this hybrid, the simulator sets $S := g^s$, $S_2 := g^{s_2}$, $S_3 := g^{s_3}$, $X_1 \leftarrow g^{r_1}$, $X_2 \leftarrow g^{r_2}$, $X_3 \leftarrow g^{r_3}$. The rest of the transcript is generated as in Hybrid 2. Observe that no PPT adversary can distinguish between Hybrid 2 and Hybrid 3, by the hiding properties of the commitment scheme.

Observe that the transcript in Hybrid 3 can be generated without using the witness. □

A.4 Proof of commitment to the plaintext

The protocol Π_{plain} described in Fig. 24 allows, given a commitment X and a pair of elements (D, E) in the NIDLS group, to prove that X is a commitment to the plaintext in (D, E). Specifically, the protocol proves the knowledge of $x, r, s \in \mathbb{Z}$ such that $X = C^x \cdot g^s$, $D = g^r$ and $E = f^x \cdot C^r$.

Sigma protocols that proves this kind of relations are rather common: the prover starts by sending $Y := C^y \cdot g^\mu$, $U := g^\nu$ and $V := f^y \cdot C^\nu$ where y, μ and ν are random. The verifier chooses a challenge $e \in \mathbb{Z}$, which the prover answers

Plaintext Commitment Proof Π_{plain}

CRS:

1. $\mathsf{par}' := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(1\!\!1^\lambda)$
2. $(g, \rho_0) \xleftarrow{R} \mathcal{D}(1\!\!1^\lambda, \mathsf{par})$
3. $(C, \rho_1) \xleftarrow{R} \mathcal{D}(1\!\!1^\lambda, \mathsf{par})$
4. If $C = g$, go to step 3.
5. Output $\mathsf{par} := (\mathsf{par}', g, \rho_0, C, \rho_1)$

STATEMENT: $D, E, X \in G$
WITNESS: $x \in \mathbb{Z}$ and $r, s \in \mathbb{Z}$ such that $X = C^x \cdot g^s$, $D = g^r$ and $E = f^x \cdot C^r$.
PROTOCOL:

1. The prover samples $y_i \xleftarrow{R} [2^\lambda \cdot B]$, $\mu_i \xleftarrow{R} [2^\lambda \cdot \ell_r]$and $\nu_i \xleftarrow{R} [2^\lambda \cdot \ell'_r]$ for every $i \in [\lambda]$.
2. The prover sends $Y_i \leftarrow C^{y_i} \cdot g^{\mu_i}$, $U_i \leftarrow g^{\nu_i}$ and $V_i \leftarrow f^{y_i} \cdot C^{\nu_i}$ for every $i \in [\lambda]$.
3. The verifier sends $\boldsymbol{e} \xleftarrow{R} \{0,1\}^\lambda$.
4. The prover sends $z_i \leftarrow y_i + e_i \cdot x$, $u_i \leftarrow \mu_i + e_i \cdot s$ and $v_i \leftarrow \nu_i + e_i \cdot r$ for every $i \in [\lambda]$.
5. The verifier accepts if $Y_i \cdot X^{e_i} = C^{z_i} \cdot g^{u_i}$, $U_i \cdot D^{e_i} = g^{v_i}$ and $V_i \cdot E^{e_i} = f^{z_i} \cdot C^{v_i}$ for every $i \in [\lambda]$.

Fig. 24. Honest-verifier zero-knowledge plaintext commitment proof

with $z := y + e \cdot x$, $u := \mu + e \cdot s$ and $v := \nu + e \cdot r$. The verifier accepts if $Y \cdot X^e = C^z \cdot g^u$, $U \cdot D^e = g^v$ and $V \cdot E^e = f^z \cdot C^v$.

Our solution, which we denote by Π_{plain}, differs from the one described above in the use of binary challenges, as we want to achieve security independently of the NIDLS instantiation. Since usually we do not know the order of C and g, we also need to sample random y, μ and ν that are 2^λ times bigger than $|x|$, $|s|$ and $|r|$ respectively. When the witnesses do not satisfy the expected bounds, the protocol is not guaranteed to satisfy HVZK.

The construction is honest-verifier zero-knowledge as long as $x \in [-B, B]$, $r \in [-\ell_r, \ell_r]$ and $s \in [-\ell'_r, \ell'_r]$ for some public bounds B, ℓ_r and ℓ'_r. When we use this protocol in our constructions, it is fundamental to choose sufficiently large B, ℓ_r and ℓ'_r, so that the protocol leaks no information about the statements of honest parties.

Theorem 13. *The construction Π_{plain} described in Fig. 24 is a complete and 2-special sound proof for the relation*

$$\mathcal{R}_{\mathsf{plain}} := \left\{ (D, E, X), (x, r, s) \middle| X = C^x \cdot g^s, D = g^r, E = f^x \cdot C^r \right\}.$$

Moreover, if $x \in [-B, B]$, $s \in [-\ell_r, \ell_r]$ *and* $r \in [-\ell'_r, \ell'_r]$, Π_{plain} *is honest-verifier zero-knowledge.*

Proof. Completeness. The verifier never rejects when the prover is honest. Indeed,

$$\begin{aligned} Y_i \cdot X^{e_i} &= C^{y_i} \cdot g^{\mu_i} \cdot C^{e_i \cdot x} \cdot g^{e_i \cdot s} = C^{z_i} \cdot g^{u_i}, \\ U_i \cdot D^{e_i} &= g^{\nu_i} \cdot g^{e_i \cdot r} = g^{v_i}, \\ V_i \cdot E^{e_i} &= f^{y_i} \cdot C^{\nu_i} \cdot f^{e_i \cdot x} \cdot C^{e_i \cdot r} = f^{z_i} \cdot C^{v_i}. \end{aligned}$$

Special Soundness. Suppose that the extractor is provided with two accepting transcripts coinciding until the second last round and having different challenges $\boldsymbol{e}$ and $\boldsymbol{e}'$. Let $(\boldsymbol{z}, \boldsymbol{u}, \boldsymbol{v})$ and $(\boldsymbol{z}', \boldsymbol{u}', \boldsymbol{v}')$ be the corresponding last round messages. We know that there exists an $i \in [\lambda]$ such that $e_i = 1$ and $e'_i = 0$, or $e_i = 0$ and $e'_i = 1$. Without loss of generality, we assume the first one (in the other case, we just switch the role of two transcripts).

We know that

$$\begin{aligned} Y_i &= C^{z'_i} \cdot g^{u'_i}, & U_i &= g^{v'_i}, & V_i &= f^{z'_i} \cdot C^{v'_i}, \\ Y_i \cdot X &= C^{z_i} \cdot g^{u_i}, & U_i \cdot D &= g^{v_i}, & V_i \cdot E &= f^{z_i} \cdot C^{v_i}. \end{aligned}$$

So, $X = C^{z_i - z'_i} \cdot g^{u_i - u'_i}$, $D = g^{v_i - v'_i}$ and $E = f^{z_i - z'_i} \cdot C^{v_i - v'_i}$.

Honest-Verifier Zero-Knowledge. Assume that $x \in [-B, B]$, $s \in [-\ell_r, \ell_r]$ and $r \in [-\ell'_r, \ell'_r]$. Consider the simulator that takes a random challenge $\boldsymbol{e} \xleftarrow{R} \{0,1\}^\lambda$ and samples $z_i \xleftarrow{R} [2^\lambda \cdot B]$, $u_i \xleftarrow{R} [2^\lambda \cdot \ell_r]$ and $v_i \xleftarrow{R} [2^\lambda \cdot \ell'_r]$ for every $i \in [\lambda]$. Then, it sets $Y_i \leftarrow C^{z_i} \cdot g^{u_i} \cdot X^{-e_i}$, $U_i \leftarrow g^{v_i} \cdot D^{-e_i}$ and $V_i \leftarrow f^{z_i} \cdot C^{v_i} \cdot E^{-e_i}$ for every $i \in [\lambda]$. Notice that this hybrid is indistinguishable from the real protocol. Indeed, the only difference between the two transcripts is the distributions of z_i, u_i and v_i. In all of the three cases, however, the statistical distance is bounded by $2^{-\lambda}$, which is negligible. So, no adversary can distinguish between protocol and simulation. Observe that the simulated transcript can be generated without knowing the witness. □

A.5 Proof of well-formed inputs

In Fig. 25, we present a sigma protocol proving the well-formedness of the input messages in the HSS scheme of Section 8. Specifically, given a tuple $(h, D, E, \overline{D}, \overline{E})$ of elements in the NIDLS group and $B \in \mathbb{N}$, the construction proves the knowledge of $x \in [-B, B]$ and $r, \overline{r} \in \mathbb{Z}$ such that $D = g^r$, $\overline{D} = g^{\overline{r}} \cdot f^{-x}$, $E = f^x \cdot h^r$ and $\overline{E} = h^{\overline{r}}$. The protocol is honest-verifier zero-knowledge as long as $r, \overline{r} \in [-\ell, \ell]$.

Theorem 14. *The construction* Π_{input} *described in Fig. 25 is a complete and special sound proof for the relation*

$$\mathcal{R}_{\mathsf{input}} := \left\{ (h, D, E, \overline{D}, \overline{E}, B), (x, r, \overline{r}) \middle| \begin{array}{l} x \in [-B, B], \\ D = g^r, E = f^x \cdot h^r, \\ \overline{D} = g^{\overline{r}} \cdot f^{-x}, \overline{E} = h^{\overline{r}} \end{array} \right\}.$$

Proof of well-formed input Π_{input}

CRS:

1. $\mathsf{par}' := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(1\!\!1^\lambda)$
2. $(g, \rho_0) \xleftarrow{R} \mathcal{D}(1\!\!1^\lambda, \mathsf{par})$
3. $(C, \rho_1) \xleftarrow{R} \mathcal{D}(1\!\!1^\lambda, \mathsf{par})$
4. If $C = g$, go to step 3.
5. Output $\mathsf{par} := (\mathsf{par}', g, \rho_0, C, \rho_1)$

STATEMENT: $h, D, E, \overline{D}, \overline{E} \in G$, $B \in \mathbb{N}$
WITNESS: $x \in [-B, B]$ and $r, \overline{r} \in \mathbb{Z}$ such that $D = g^r$, $\overline{D} = g^{\overline{r}} \cdot f^{-x}$, $E = f^x \cdot h^r$ and $\overline{E} = h^{\overline{r}}$.
PROTOCOL:

1. The prover samples $s \xleftarrow{R} [\ell]$ and sends $X \leftarrow C^x \cdot g^s$.
2. The parties run Π_{range} with input $X \cdot C^B$ and bound $2B$. The witness is $(x + B, s)$.
3. The prover samples $y_i \xleftarrow{R} [2^\lambda \cdot B]$, $\mu_i, \nu_i, \overline{\nu}_i \xleftarrow{R} [2^\lambda \cdot \ell]$ for every $i \in [\lambda]$.
4. The prover sends, for every $i \in [\lambda]$,

$$Y_i \leftarrow C^{y_i} \cdot g^{\mu_i}, \qquad U_i \leftarrow g^{\nu_i}, \qquad V_i \leftarrow f^{y_i} \cdot h^{\nu_i},$$
$$\overline{U}_i \leftarrow g^{\overline{\nu}_i} \cdot f^{-x}, \qquad \overline{V}_i \leftarrow h^{\overline{\nu}_i}.$$

5. The verifier sends $\boldsymbol{e} \xleftarrow{R} \{0,1\}^\lambda$.
6. The prover sends $z_i \leftarrow y_i + e_i \cdot x$, $u_i \leftarrow \mu_i + e_i \cdot s$, $v_i \leftarrow \nu_i + e_i \cdot r$ and $\overline{v}_i \leftarrow \overline{\nu}_i + e_i \cdot \overline{r}$ for every $i \in [\lambda]$.
7. The verifier accepts if, for every $i \in [\lambda]$,

$$Y_i \cdot X^{e_i} = C^{z_i} \cdot g^{u_i}, \qquad U_i \cdot D^{e_i} = g^{v_i}, \qquad V_i \cdot E^{e_i} = f^{z_i} \cdot h^{v_i},$$
$$\overline{U}_i \cdot \overline{D}^{e_i} = g^{\overline{v}_i} \cdot f^{-z_i}, \qquad \overline{V}_i \cdot \overline{E}^{e_i} = h^{\overline{v}_i}$$

Fig. 25. Honest-verifier zero-knowledge proof of well-formed inputs

Moreover, if $r, \overline{r} \in [-\ell, \ell]$, Π_{input} is honest-verifier zero-knowledge.

Proof. Completeness. The verifier never rejects when the prover is honest. Indeed, by the completeness of Π_{range}, the verifier can reject only at the end. Moreover,

$$\begin{aligned}
Y_i \cdot X^{e_i} &= C^{y_i} \cdot g^{\mu_i} \cdot C^{e_i \cdot x} \cdot g^{e_i \cdot s} = C^{z_i} \cdot g^{u_i}, \\
U_i \cdot D^{e_i} &= g^{\nu_i} \cdot g^{e_i \cdot r} = g^{v_i}, \\
V_i \cdot E^{e_i} &= f^{y_i} \cdot h^{\nu_i} \cdot f^{e_i \cdot x} \cdot h^{e_i \cdot r} = f^{z_i} \cdot h^{v_i}, \\
\overline{U}_i \cdot \overline{D}^{e_i} &= f^{-y_i} \cdot g^{\overline{\nu}_i} \cdot f^{-e_i \cdot x} \cdot g^{e_i \cdot \overline{r}} = f^{-z_i} \cdot g^{\overline{v}_i}, \\
\overline{V}_i \cdot \overline{E}^{e_i} &= h^{\overline{\nu}_i} \cdot h^{e_i \cdot \overline{r}} = h^{\overline{v}_i}.
\end{aligned}$$

Special Soundness. By the special soundness of Π_{range}, the extractor is able to retrieve $x \in [-B, B]$ and $s \in \mathbb{Z}$ such that $X \cdot C^B = C^{x+B} \cdot g^s$.

Now, suppose that the extractor is provided with two accepting transcripts coinciding until the second last round and having different challenges $\boldsymbol{e}$ and $\boldsymbol{e}'$. Let $(\boldsymbol{z}, \boldsymbol{u}, \boldsymbol{v}, \overline{\boldsymbol{v}})$ and $(\boldsymbol{z}', \boldsymbol{u}', \boldsymbol{v}', \overline{\boldsymbol{v}}')$ be the corresponding last round messages. We know that there exists an $i \in [\lambda]$ such that $e_i = 1$ and $e'_i = 0$, or $e_i = 0$ and $e'_i = 1$. Without loss of generality, we assume the first one (in the other case, we just switch the role of two transcripts).

We know that

$$\begin{array}{llll}
 & Y_i = C^{z'_i} \cdot g^{u'_i}, & Y_i \cdot X = C^{z_i} \cdot g^{u_i}, & \\
U_i = g^{v'_i}, & V_i = f^{z'_i} \cdot h^{v'_i}, & U_i \cdot D = g^{v_i}, & V_i \cdot E = f^{z_i} \cdot h^{v_i} \\
\overline{U}_i = f^{-z'_i} \cdot g^{\overline{v}'_i}, & \overline{V}_i = h^{\overline{v}'_i}, & \overline{U}_i \cdot \overline{D} = f^{-z_i} \cdot g^{\overline{v}_i}, & \overline{V}_i \cdot \overline{E} = h^{\overline{v}_i}.
\end{array}$$

So,

$$\begin{gathered}
X = C^{z_i - z'_i} \cdot g^{u_i - u'_i}, \qquad D = g^{v_i - v'_i}, \qquad E = f^{z_i - z'_i} \cdot h^{v_i - v'_i}, \\
\overline{D} = f^{-(z_i - z'_i)} \cdot g^{\overline{v}_i - \overline{v}'_i}, \qquad \overline{E} = h^{\overline{v}_i - \overline{v}'_i}.
\end{gathered}$$

Since $X = C^x \cdot g^s$, by the binding properties of the commitment scheme in Fig. 10, it must be that $z_i - z'_i = x$.

Honest-Verifier Zero-Knowledge. Assume that $r, \overline{r} \in [-\ell, \ell]$. Consider the following series of hybrids.

Hybrid 1. In this hybrid, we generate the transcript in Π_{range} using the corresponding simulator. By the honest-verifier zero-knowledge of Π_{range}, no adversary can distinguish between this hybrid and the real protocol.

Hybrid 2. In this hybrid, the simulator takes a random challenge $\boldsymbol{e} \xleftarrow{R} \{0,1\}^\lambda$ and samples $z_i \xleftarrow{R} [2^\lambda \cdot B]$ and $u_i, v_i, \overline{v}_i \xleftarrow{R} [2^\lambda \cdot \ell]$ for every $i \in [\lambda]$. Then, it sets, for every $i \in [\lambda]$,

$$\begin{gathered}
Y_i \leftarrow C^{z_i} \cdot g^{u_i} \cdot X^{-e_i}, \qquad U_i \leftarrow g^{v_i} \cdot D^{-e_i}, \qquad V_i \leftarrow f^{z_i} \cdot h^{v_i} \cdot E^{-e_i}, \\
\overline{U}_i \leftarrow f^{-z_i} \cdot g^{\overline{v}_i} \cdot \overline{D}^{-e_i}, \qquad \overline{V}_i \leftarrow h^{\overline{v}_i} \cdot \overline{E}^{-e_i}.
\end{gathered}$$

Proof of small MAC key Π_{MAC}

CRS:

1. $\mathsf{par}' := (G, F, H, f, t, \ell, \mathsf{aux}) \xleftarrow{R} \mathsf{Gen}(\mathbb{1}^\lambda)$
2. $(g, \rho_0) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
3. $(C, \rho_1) \xleftarrow{R} \mathcal{D}(\mathbb{1}^\lambda, \mathsf{par})$
4. If $C = g$, go to step 3.
5. Output $\mathsf{par} := (\mathsf{par}', g, \rho_0, C, \rho_1)$

STATEMENT: $A \in G$, $B \in \mathbb{N}$
WITNESS: $x \in [0, B]$ and $r \in \mathbb{Z}$ such that $A = f^x \cdot g^r$.
PROTOCOL:

1. The prover samples $s \xleftarrow{R} [\ell]$ and sends $X \leftarrow C^x \cdot g^s$.
2. The parties run Π_{range} with input X and bound B. The witness is (x, s).
3. The prover samples $y_i \xleftarrow{R} [2^\lambda \cdot B]$, $\mu_i, \nu_i \xleftarrow{R} [2^\lambda \cdot \ell]$ for every $i \in [\lambda]$.
4. The prover sends $Y_i \leftarrow C^{y_i} \cdot g^{\mu_i}$ and $V_i \leftarrow f^{y_i} \cdot g^{\nu_i}$ for every $i \in [\lambda]$.
5. The verifier sends $\boldsymbol{e} \xleftarrow{R} \{0,1\}^\lambda$.
6. The prover sends $z_i \leftarrow y_i + e_i \cdot x$, $u_i \leftarrow \mu_i + e_i \cdot s$ and $v_i \leftarrow \nu_i + e_i \cdot r$ for every $i \in [\lambda]$.
7. The verifier accepts if $Y_i \cdot X^{e_i} = C^{z_i} \cdot g^{u_i}$ and $V_i \cdot A^{e_i} = f^{z_i} \cdot g^{v_i}$ for every $i \in [\lambda]$.

Fig. 26. Honest-verifier zero-knowledge proof of small MACs

Notice that this hybrid is indistinguishable from Hybrid 1. Indeed, the only difference between the two transcripts is the distributions of z_i, u_i, v_i and $\overline{v}_i$. In all of the three cases, however, the statistical distance is bounded by $2^{-\lambda}$, which is negligible.

Hybrid 3. In this hybrid, the simulator sends $X \leftarrow g^s$ instead of $C^x \cdot g^s$. This hybrid is indistinguishable from the previous one by the hiding properties of the commitment scheme in Fig. 10. Observe that, now, the simulated transcript can be generated without knowing the witness. □

A.6 Proof of well-formed MAC keys

In Fig. 26, we present a sigma protocol proving the well-formedness of the messages encoding the MAC key in the active HSS scheme of Section 8. Specifically, given a group element A and a bound $B \in \mathbb{N}$, the construction proves the knowledge of $x \in [0, B]$ and $r \in \mathbb{Z}$ such that $A = f^x \cdot g^r$. The protocol is honest-verifier zero-knowledge as long as $r \in [-\ell, \ell]$.

Theorem 15. *The construction Π_{MAC} described in Fig. 26 is a complete and special sound proof for the relation*

$$\mathcal{R}_{\mathsf{MAC}} := \{(A, B), (x, r) | x \in [0, B], A = f^x \cdot g^r\}.$$

Moreover, if $r \in [-\ell, \ell]$, Π_{MAC} is honest-verifier zero-knowledge.

Proof. Completeness. The verifier never rejects when the prover is honest. Indeed, by the completeness of Π_{range}, the verifier can reject only at the end. Moreover,

$$Y_i \cdot X^{e_i} = C^{y_i} \cdot g^{\mu_i} \cdot C^{e_i \cdot x} \cdot g^{e_i \cdot s} = C^{z_i} \cdot g^{u_i},$$
$$V_i \cdot A^{e_i} = f^{y_i} \cdot g^{\nu_i} \cdot f^{e_i \cdot x} \cdot g^{e_i \cdot r} = f^{z_i} \cdot g^{v_i}.$$

Special Soundness. By the special soundness of Π_{range}, the extractor is able to retrieve $x \in [0, B]$ and $s \in \mathbb{Z}$ such that $X = C^x \cdot g^s$.

Now, suppose that the extractor is provided with two accepting transcripts coinciding until the second last round and having different challenges $\boldsymbol{e}$ and $\boldsymbol{e}'$. Let $(\boldsymbol{z}, \boldsymbol{u}, \boldsymbol{v})$ and $(\boldsymbol{z}', \boldsymbol{u}', \boldsymbol{v}')$ be the corresponding last round messages. We know that there exists an $i \in [\lambda]$ such that $e_i = 1$ and $e'_i = 0$, or $e_i = 0$ and $e'_i = 1$. Without loss of generality, we assume the first one (in the other case, we just switch the role of two transcripts).

We know that

$$Y_i = C^{z'_i} \cdot g^{u'_i}, \quad Y_i \cdot X = C^{z_i} \cdot g^{u_i}, \quad V_i = f^{z'_i} \cdot g^{v'_i}, \quad V_i \cdot A = f^{z_i} \cdot g^{v_i}.$$

So, $X = C^{z - i - z'_i} \cdot g^{u_i - u'_i}$ and $A = f^{z_i - z'_i} \cdot g^{v_i - v'_i}$. Since $X = C^x \cdot g^s$, by the binding properties of the commitment scheme in Fig. 10, it must be that $z_i - z'_i = x$.

Honest-Verifier Zero-Knowledge. Assume that $r \in [-\ell, \ell]$. Consider the following series of hybrids.

Hybrid 1. In this hybrid, we generate the transcript in Π_{range} using the corresponding simulator. By the honest-verifier zero-knowledge of Π_{range}, no adversary can distinguish between this hybrid and the real protocol.

Hybrid 2. In this hybrid, the simulator takes a random challenge $\boldsymbol{e} \xleftarrow{R} \{0,1\}^\lambda$ and samples $z_i \xleftarrow{R} [2^\lambda \cdot B]$ and $u_i, v_i \xleftarrow{R} [2^\lambda \cdot \ell]$ for every $i \in [\lambda]$. Then, it sets $Y_i \leftarrow C^{z_i} \cdot g^{u_i} \cdot X^{-e_i}$ and $V_i \leftarrow f^{z_i} \cdot g^{v_i}$ for every $i \in [\lambda]$. Notice that this hybrid is indistinguishable from Hybrid 1. Indeed, the only difference between the two transcripts is the distributions of z_i, u_i and v_i. In all of the three cases, however, the statistical distance is bounded by $2^{-\lambda}$, which is negligible.

Hybrid 3. In this hybrid, the simulator sends $X \leftarrow g^s$ instead of $C^x \cdot g^s$. This hybrid is indistinguishable from the previous one by the hiding properties of the commitment scheme in Fig. 10. Observe that, now, the simulated transcript can be generated without knowing the witness. □