J. Cryptol. (2011) 24: 545–587
DOI: 10.1007/s00145-010-9070-1

Journal of
CRYPTOLOGY

Short Undeniable Signatures Based on Group Homomorphisms*

Jean Monnerat
SwissSign AG, Sägereistrasse 25, 8152 Glattbrugg, Switzerland
jean.monnerat@gmail.com

Serge Vaudenay
EPFL, 1015 Lausanne, Switzerland
serge.vaudenay@epfl.ch

Communicated by Matthew Franklin

Received 23 May 2007
Online publication 18 June 2010

Abstract. This paper is devoted to the design and analysis of short undeniable signatures based on a random oracle. Exploiting their online property, we can achieve signatures with a fully scalable size depending on the security level. To this end, we develop a general framework based on the interpolation of group homomorphisms, leading to the design of a generic undeniable signature scheme called MOVA with batch verification and featuring nontransferability. By selecting group homomorphisms with a small group range, we obtain very short signatures. We also minimize the number of moves of the verification protocols by proposing some variants with only two moves in the random oracle model. We provide a formal security analysis of MOVA and assess the security in terms of the signature length. Under reasonable assumptions and with some carefully selected parameters, the MOVA scheme makes it possible to consider signatures of about 50 bits.

Key words. Undeniable signatures, Short signatures, Group homomorphisms, Interpolation, Interactive proofs

1. Introduction

An undeniable signature scheme is similar to a classical digital signature except that the recipient of a message cannot verify its validity alone: he needs to interact with the signer in order to be convinced of the validity or invalidity of the signature. This property, called *invisibility*, opposes to the universal verifiability of classical digital signatures and allows the signer to have a control on how his signature spreads. This concept was put forth by Chaum and van Antwerpen [17] in 1989 and was mainly motivated by the need for privacy of a signer dealing with private or sensitive contract. Later

* This paper builds upon the conference papers [41,43].

on, several additional applications have been proposed such as licensing sensitive software [15], electronic voting [52], and digital cash [10,16,50]. An undeniable signature scheme is composed of a key generation algorithm, a signature generation algorithm, a confirmation protocol to prove the validity of a valid signature, and a denial protocol to prove the invalidity of an invalid signature. It achieves nonrepudiation in the sense that valid signatures can only be forged by the signer and cannot be denied, but invalid signatures can be denied. Privacy is based on invisibility. It can be strengthened further by having proofs *nontransferable*: a malicious verifier cannot take advantage of the interaction with the prover to prove (in)validity of a signature to a third party.

Following the seminal article of Chaum and van Antwerpen, a quite fair amount of work has been dedicated to this field. In particular, their original scheme gave rise to a series of articles [9,15,18,20,33,38,39,45,46] devoted to schemes based on the discrete logarithm problem. As alternative of the discrete logarithm, Gennaro et al. [24] proposed an undeniable signature based on RSA and Biehl et al. [6] devised a scheme based on quadratic orders. The tremendous development of pairing-based cryptography over the last few years also influenced the field of undeniable signatures as illustrated by the identity-based scheme of Libert and Quisquater [36] and schemes proposed by Laguillaumie and Vergnaud [34,35]. Besides, it is worth to mention papers dealing with other issues such as the transferability of the (in)validity proof of a signature [21] or blackmailing against the signer [29].

In traditional digital signature schemes, the security collapses when the signature is too short because of universal verifiability: an attacker can try to guess a signature until it is valid in order to forge it. One advantage of undeniable signatures is that the security smoothly decreases with the signature length. As an example, we can think of 20-bit signatures which cannot be forged but with a probability of success of 2^{-20}. The forger can increase it in an on-line attack, but the number of verification or signing queries can be easily limited. So, undeniable signatures could in principle be arbitrarily small, e.g., as small as a MAC, although no such signatures were proposed prior to this work.

Our Contribution As far as we know, all previous signature schemes did not fully exploit the lack of offline verification capability towards the design of schemes offering very short signatures. One of the main contributions of this work is to remedy to this situation. To this goal, we develop a general framework based on the sole notion of the interpolation of group homomorphisms. Based on it, we define a decisional problem and a computational problem, which generalizes several fundamental problems related to public-key cryptography. Among them, we find the decision and computational Diffie–Hellman problems as well as the quadratic residuosity problem. We use this settings to develop a new scheme called MOVA which is based on a group homomorphism privately known by the signer.

The interest of this technique to undeniable signatures is twofold. First, group homomorphisms allow one to express the well-known Chaum's undeniable signature [15] and the RSA undeniable signature of Gennaro et al. [24] in a unified formalism. Secondly, we obtain very short signatures in a quite natural way, namely by instantiating our MOVA scheme with group homomorphisms with a range group of small size. In addition to this, we introduce nontransferability features in our MOVA scheme.

We also propose some two-move verification protocols for MOVA in the random oracle model. As far as we know, these are the first interactive verification protocols achieving only two moves. We provide some formal security proofs on the different required properties related to the confirmation and denial protocols such as the *soundness*, *zero-knowledge*, and *nontransferability*. We address invisibility and unforgeability in settings where the attacker has access to signing, confirmation, and denial oracles. This provides precise security bounds and explain how to select MOVA parameters.

Finally, we offer a batch verification protocol.

Structure of the Paper The next section gives the definition of cryptographic primitives, and the subsequent one is devoted to the security model of undeniable signatures. In Sect. 4, we develop the concept of interpolation of group homomorphisms and provide some technical results that are necessary for the rest of this article. It includes interactive protocols that are used in some setup and verification protocols of MOVA. Then, we give a description of our new scheme called MOVA and prove its security in Sect. 5. Finally, we provide some possible instantiations and parameters, discuss additional properties of MOVA, and conclude this paper.

2. Preliminaries

Notation A function $f(n)$ is called *polynomial* and we write $f(n) = \mathsf{poly}(n)$ if there exists an integer k such that $f(n) = \mathcal{O}(n^k)$. It is called *negligible* and we write $f(n) = \mathsf{negl}(n)$ if for any integer k, we have $f(n) = \mathcal{O}(n^{-k})$. For a set S, the notation $s \in_U S$ means that we assign to s an element picked uniformly at random in S. For a probabilistic algorithm A, we denote by $\mathsf{A}(x; r)$ the output produced by A on input x with coins r and by $y \leftarrow \mathsf{A}(x)$ the action of assigning y to the output of A with input x and random coins as we often omit the coins from the notation when unnecessary. The *statistical distance* between two random variables X_1 and X_2 with range $\mathcal{X}$ is $\Delta(X_1, X_2) = \frac{1}{2}\sum_{x\in\mathcal{X}} |\Pr[X_1 = x] - \Pr[X_2 = x]|$. Two random variables X and Y are computationally resp. statistically resp. perfectly *indistinguishable* and we write $X \approx_c Y$ resp. $X \approx_s Y$ resp. $X \approx_p Y$ if no polynomially bounded distinguisher can tell X and Y apart with nonnegligible advantage resp. $\Delta(X, Y)$ is negligible resp. X and Y have the same probability distribution.

When dealing with Abelian groups, we will use additive notation for group operations.

Proof of Membership A proof protocol Proto is a pair $\mathsf{Proto} = (\mathcal{P}, \mathcal{V})$ of interactive algorithms called a *prover* and a *verifier*. The verifier $\mathcal{V}$ is assumed to be probabilistic with polynomial-time complexity. The prover $\mathcal{P}$ is unbounded. One of the two algorithms is called the *initiator* in the protocol.

We recall that an interactive machine is defined by a deterministic algorithm $\mathcal{A}$ mapping an input x, some coins r, and a list (possibly empty in the case of the initiator) of input messages $m_1, \ldots, m_n$ to a next-message $\mathcal{A}(x, m_1, \ldots, m_n; r) = m'$. A message ending by a special termination symbol is called a final message. The $(x, r, m_1, \ldots, m_n)$ tuple is called the *partial view* of the machine. It is complete if either m_n or $\mathcal{A}(x, m_1, \ldots, m_n; r)$ is a final message.

An instance of a protocol execution denoted $\mathcal{P}(w;r_P) \overset{x}{\leftrightarrow} \mathcal{V}(z;r_V)$ refers to using x as a *common input*, w as a private input for $\mathcal{P}$ (this may be called a *witness*), z as a private input for $\mathcal{V}$ (this may be called an *auxiliary input*), and r_P resp. r_V as random coins for $\mathcal{P}$ resp. $\mathcal{V}$.[1] For instance, if the verifier is the initiator and if $m_0^V, m_1^V, \ldots$ resp. $m_1^P, m_2^P, \ldots$ are the messages sent by the verifier resp. the prover, we have $m_i^V = \mathcal{V}(x, z, r_V, m_1^P, \ldots, m_{i-1}^P)$ and $m_i^P = \mathcal{P}(x, w, r_P, m_0^V, \ldots, m_i^V)$ for all i. The complete view of $\mathcal{V}$ is denoted

$$\mathsf{View}_{\mathcal{V}}\big(\mathcal{P}(w;r_P) \overset{x}{\leftrightarrow} \mathcal{V}(z;r_V)\big) = \big(x, z, r_V, m_1^P, \ldots, m_n^P\big).$$

The output from $\mathcal{V}$, denoted $\mathsf{Output}_{\mathcal{V}}(\mathcal{P}(w;r_P) \overset{x}{\leftrightarrow} \mathcal{V}(z;r_V))$, is the last message from $\mathcal{V}$.

In the *random oracle model*, algorithms may access to an oracle Gen which implements a random function with uniform distribution mapping elements of a specified domain to elements of a specified range. For simplicity, we restrict to cases where the domain and the range are finite sets. Note that an empty domain corresponds to the standard model where random oracles are not used.

A *proof of membership* for language L is defined by a set $\mathcal{K}$ of pairs $(K_{\mathsf{p}}^{\mathcal{V}}, K_{\mathsf{s}}^{\mathcal{V}})$, a mapping $L(x)$ defining the set of witnesses for a given $x \in L$, and a proof protocol $(\mathcal{P}^{\mathsf{Gen}}, \mathcal{V}^{\mathsf{Gen}})$. The proof must verify the following properties.

Completeness. For any $x \in L$, $w \in L(x)$, $(K_{\mathsf{p}}^{\mathcal{V}}, K_{\mathsf{s}}^{\mathcal{V}}) \in \mathcal{K}$, r_P, r_V, and any instance of Gen, we have

$$\mathsf{Output}_{\mathcal{V}}\Big(\mathcal{P}^{\mathsf{Gen}}(w;r_P) \overset{x, K_{\mathsf{p}}^{\mathcal{V}}}{\longleftrightarrow} \mathcal{V}^{\mathsf{Gen}}\big(K_{\mathsf{s}}^{\mathcal{V}};r_V\big)\Big) = \mathsf{accept}.$$

ε-Soundness. For any $x \notin L$, any algorithm $\mathcal{P}^*$, any $(K_{\mathsf{p}}^{\mathcal{V}}, K_{\mathsf{s}}^{\mathcal{V}}) \in \mathcal{K}$, given a random r_V and a random instance Gen, we have

$$\Pr\Big[\mathsf{Output}_{\mathcal{V}}\Big(\mathcal{P}^{*\mathsf{Gen}} \overset{x, K_{\mathsf{p}}^{\mathcal{V}}}{\longleftrightarrow} \mathcal{V}^{\mathsf{Gen}}\big(K_{\mathsf{s}}^{\mathcal{V}};r_V\big)\Big) = \mathsf{accept}\Big] \leq \varepsilon.$$

The definition was adapted here to accommodate cases where the verifier has a public/private key pair. The classical definition with auxiliary input is when $K_{\mathsf{p}}^{\mathcal{V}}$ is void. Later, the secret of the verifier may allow a malicious prover to cheat. For this, we need to replace the notion of *proof* by the notion of *argument* where soundness is replaced by the following property.

ε-Computational Soundness. For any $x \notin L$, any polynomial algorithm $\mathcal{P}^*$, given some random $(K_{\mathsf{p}}^{\mathcal{V}}, K_{\mathsf{s}}^{\mathcal{V}}) \in \mathcal{K}$, r_V, and Gen, we have

$$\Pr\Big[\mathsf{Output}_{\mathcal{V}}\Big(\mathcal{P}^{*\mathsf{Gen}} \overset{x, K_{\mathsf{p}}^{\mathcal{V}}}{\longleftrightarrow} \mathcal{V}^{\mathsf{Gen}}\big(K_{\mathsf{s}}^{\mathcal{V}};r_V\big)\Big) = \mathsf{accept}\Big] \leq \varepsilon.$$

[1] Since $\mathcal{P}$ is unbounded, we could assume that the prover uses neither witness nor any random coin without loss of generality. In practice, however, we use a probabilistic polynomially bounded prover with a witness set up as some kind of secret key. So we keep it in the notation to avoid confusion

That is, the malicious prover is now given some limited time to cheat online given the key of the verifier but still has any time to cheat offline given the instance x.

The proof/argument of membership is *zero-knowledge* (ZK) if the following property is satisfied.

Zero-Knowledge. There exists a probabilistic polynomial-time oracle machine $\mathcal{B}$ such that for any $x \in L$, any $w \in L(x)$, and any polynomial-time verifier $\mathcal{V}^*$, any auxiliary input z, and any $K_{\mathsf{p}}^{\mathcal{V}}$, given a random r_P, r_V, r_B, and Gen, we have

$$\mathsf{View}_{\mathcal{V}}\left(\mathcal{P}^{\mathsf{Gen}}(w; r_P) \overset{x, K_{\mathsf{p}}^{\mathcal{V}}}{\longleftrightarrow} \mathcal{V}^{*\mathsf{Gen}}(z; r_V)\right) \approx \mathcal{B}^{\mathcal{V}^*, \mathsf{Gen}}\left(x, K_{\mathsf{p}}^{\mathcal{V}}, z; r_B\right),$$

where $\mathcal{B}^{\mathcal{V}^*, \mathsf{Gen}}(x, K_{\mathsf{p}}^{\mathcal{V}}, z; r_B)$ denotes the output of $\mathcal{B}$ with oracle $\mathcal{V}^*$ and Gen, input $x, K_{\mathsf{p}}^{\mathcal{V}}, z$, and random coins r_B. Depending on the type of indistinguishability, we have computational resp. statistical resp. perfect ZK. We further say that the proof is *straight-line* ZK if we can construct an online fake prover $\mathcal{P}^*$ with input z who cheats by looking at the communication tape between $\mathcal{V}^*$ and Gen, i.e., if $\mathcal{B}$ consists of simulating the interaction

$$\mathcal{P}^{*\mathsf{Gen}, \mathsf{tape}}(z; r_P) \overset{x, K_{\mathsf{p}}^{\mathcal{V}}}{\longleftrightarrow} \mathcal{V}^{*\mathsf{Gen}}(z; r_V).$$

We note that the above definition of zero-knowledge is black-box [26], which means that we require the existence of one "universal" simulator having an oracle access to the verifier.

In the standard model, Barak et al. [4] proved that zero-knowledge proofs of an NP-complete language (possibly non-black-box) requires at least three moves. To overcome this limitation, the notion of zero-knowledge was extended in the random oracle model (for more details, see [5]) in which the queries to the random oracles are controlled by the simulator, i.e., it can simulate the output of the oracles, provided that the output distribution is correct. Pass [49] proposed the notion of *deniable zero-knowledge* in which the simulator is no longer allowed to simulate the output of the random oracles, but is only able to observe the queries made to the random oracles and the corresponding answers. This restriction makes the simulator to produce views that do not yield evidence that the interactive proof occurred. The above definition follows this model. Pass [49] showed that two moves are necessary and sufficient to achieve deniable zero-knowledge for NP. Our interactive zero-knowledge proofs (with two moves) involving random oracles will be deniable. Deniability is indeed crucial to preserve invisibility in undeniable signatures.[2]

The argument of membership is *nontransferable* (NT) if the following property is satisfied.

Nontransferability. There exists a probabilistic polynomial-time algorithm $\mathcal{P}^*$ such that for any $x \in L$, any $(K_{\mathsf{p}}^{\mathcal{V}}, K_{\mathsf{s}}^{\mathcal{V}}) \in \mathcal{K}$, any verifier $\mathcal{V}^*$, given a random r_P, r_V, and

[2] As an artefact of the quite unfortunate standard terminology, we can see that we have an undeniable signature scheme with a deniable protocol which is a deniable proof itself!

Gen, we have

$$\mathsf{View}_{\mathcal{V}}\Big(\mathcal{P}^{\mathsf{Gen}}(w;r_P) \overset{x,K_{\mathsf{p}}^{\mathcal{V}}}{\longleftrightarrow} \mathcal{V}^{*\mathsf{Gen}}\big(K_{\mathsf{s}}^{\mathcal{V}};r_V\big)\Big)$$

$$\approx \mathsf{View}_{\mathcal{V}}\Big(\mathcal{P}^{*\mathsf{Gen}}\big(K_{\mathsf{s}}^{\mathcal{V}};r_P\big) \overset{x,K_{\mathsf{p}}^{\mathcal{V}}}{\longleftrightarrow} \mathcal{V}^{*\mathsf{Gen}}\big(K_{\mathsf{s}}^{\mathcal{V}};r_V\big)\Big).$$

Depending on the type of indistinguishability we have computational resp. statistical resp. perfect NT.

Clearly, the value $K_{\mathsf{s}}^{\mathcal{V}}$ allows a malicious prover to cheat with a verifier with public key $K_{\mathsf{p}}^{\mathcal{V}}$ in polynomial time. Thus, we cannot achieve regular soundness.

We note that the definition of nontransferability allows one to avoid some attacks in which the verifier $\mathcal{V}^*$ identified with $K_{\mathsf{p}}^{\mathcal{V}}$ interact with the honest signer and a hidden malicious verifier $\tilde{\mathcal{V}}$ so that $\tilde{\mathcal{V}}$ gets a proof that $x \in L$. Namely, our definition ensures that $\mathcal{V}^*$ with knowledge of $K_{\mathsf{s}}^{\mathcal{V}}$ could simulate the messages sent by $\mathcal{S}$ (without any help from $\mathcal{S}$) in a straight-line way. Indeed, if we restrict to malicious verifiers knowing $K_{\mathsf{s}}^{\mathcal{V}}$, the protocol is straight-line ZK in a standard-model sense since $\mathcal{P}^*$ does not cheat by looking at the interaction tape between $\mathcal{V}^*$ and the random oracle.

Our definition of nontransferability is similar to the one of Camenisch and Michels [14] with the main difference that our version assumes that $\mathcal{V}^*$ is computationally unbounded. We can thus assume without loss of generality that $\mathcal{V}^*$ makes no queries to the signing and confirmation/denial oracles when considering undeniable signatures. Therefore, the nontransferability of the protocols presented below will also hold with respect to the Camenisch–Michels definition.

The proof of membership is *noninteractive zero-knowledge* (NIZK) if the prover is the initiator and sends a single message and if the protocol satisfies the following property.

Noninteractive Zero-Knowledge. There exists a probabilistic polynomial-time oracle machine $\mathcal{B}$ such that for any $x \in L$, any $w \in L(x)$, and any $K_{\mathsf{p}}^{\mathcal{V}}$, given a random r_P, r_B, and Gen, we have

$$\big(\mathsf{Gen}, \mathcal{P}^{\mathsf{Gen}}\big(x, K_{\mathsf{p}}^{\mathcal{V}}, w; r_P\big)\big) \approx \mathcal{B}\big(x, K_{\mathsf{p}}^{\mathcal{V}}; r_B\big),$$

where $\mathcal{B}(x, K_{\mathsf{p}}^{\mathcal{V}}; r_B)$ generates an algorithm defining a function which simulates Gen.

Trapdoor Commitment Scheme We use trapdoor commitment schemes [11]. These were used by Jakobsson et al. [30] to construct nontransferable proofs. We will follow their methodology as well. Trapdoor commitment schemes are made of three probabilistic algorithms. The first one generates a pair of keys $(K_{\mathsf{p}}, K_{\mathsf{s}})$. The second one is a commitment algorithm Commit, and the third one is a collision algorithm Equivocate. On a given message m and random coins dec (which will be used as the decommitment value), the commitment value is $\mathsf{com} = \mathsf{Commit}(K_{\mathsf{p}}, m; \mathsf{dec})$. To *open* a commitment means to release m and dec and to check that it produces the correct commitment value. The Equivocate algorithm satisfies the following property: for any message m and any commitment value com, running $\mathsf{Equivocate}(K_{\mathsf{s}}, m, \mathsf{com})$ produces a uniformly distributed string dec among all those such that $\mathsf{com} = \mathsf{Commit}(K_{\mathsf{p}}, m; \mathsf{dec})$. Interestingly,

Equivocate is not meant to be used at all. We only need its existence as a security warranty. Somehow, it is a "life jacket algorithm."

We want our commitment scheme to achieve *binding* and *hiding* properties. A commitment scheme is *computationally binding* if no probabilistic polynomial-time algorithm launched with a random $K_{\mathsf{p}}^{\mathcal{V}}$ can output two distinct messages m, m', two decommitment values dec, dec$'$, one commitment value com such that com $=$ Commit$(K_{\mathsf{p}}, m;$ dec$)$ and com $=$ Commit$(K_{\mathsf{p}}, m;$ dec$')$ with nonnegligible probability. We denote the success probability of an algorithm $\mathcal{A}$ in this game by $\mathsf{Succ}_{\mathcal{A}}^{\text{com-bnd}}$. A commitment scheme is *perfectly hiding* if the distribution of com given by com $=$ Commit$(K_{\mathsf{p}}, m;$ dec$)$ is uniformly distributed for any K_{p} and m. For an example of a perfectly-hiding and computationally-binding trapdoor commitment scheme, we refer to Bresson et al. [12].

Trapdoor One-Way Permutations over a set $\mathcal{S}$ of bitstrings of a given fixed length (depending on the security parameter) are made of three algorithms. The first one generates a pair of keys $(K_{\mathsf{p}}, K_{\mathsf{s}})$. The other two are deterministic and map $x \in \mathcal{S}$ to $\mathsf{TPOW}(K_{\mathsf{p}}, x)$ resp. $\mathsf{TPOW}^{-1}(K_{\mathsf{s}}, x)$, both in $\mathcal{S}$. They must be such that $\mathsf{TPOW}^{-1}(K_{\mathsf{s}}, \mathsf{TPOW}(K_{\mathsf{p}}, x)) = x$ for any x. Here again, TPOW^{-1} is a life jacket algorithm which is not meant to ever be used in our constructions. We denote $\mathsf{Succ}_{\mathcal{A}}^{\text{inv-tp}}$ the probability for an adversary $\mathcal{A}$ to compute $\mathsf{TPOW}^{-1}(K_{\mathsf{s}}, y)$ given a random (uniform) $y \in \mathcal{S}$, without knowing K_{s}.

3. Undeniable Signature

We consider two players who are the signer $\mathcal{S}$ and the verifier $\mathcal{V}$. Let $k \in \mathbb{N}$ be a security parameter, $\mathcal{M}$ the message space, and Σ the signature space.[3] An undeniable signature scheme is composed of the four following algorithms.

Setup The setup is composed of two probabilistic polynomial-time (in terms of k) algorithms $\mathsf{Setup}^{\mathcal{S}}$ and $\mathsf{Setup}^{\mathcal{V}}$ producing the signer's key pair $(K_{\mathsf{p}}^{\mathcal{S}}, K_{\mathsf{s}}^{\mathcal{S}}) \leftarrow \mathsf{Setup}^{\mathcal{S}}(1^k)$ and the verifier's key pair $(K_{\mathsf{p}}^{\mathcal{V}}, K_{\mathsf{s}}^{\mathcal{V}}) \leftarrow \mathsf{Setup}^{\mathcal{V}}(1^k)$. Possible public keys are called *well-formed* keys.

Validate A deterministic polynomial-time algorithm $\mathsf{Validate}(K_{\mathsf{p}}^{\mathcal{S}})$ is used to check that $K_{\mathsf{p}}^{\mathcal{S}}$ is a well-formed key.

Sign Let $m \in \mathcal{M}$ be a message to sign. On the input of the signer's secret key $K_{\mathsf{s}}^{\mathcal{S}}$, the polynomial-time algorithm Sign generates a signature $\sigma = \mathsf{Sign}(K_{\mathsf{s}}^{\mathcal{S}}, m)$ which lies in Σ. In this paper we restrict to deterministic Sign algorithms. For all well-formed $K_{\mathsf{p}}^{\mathcal{S}}$, we say that $(K_{\mathsf{p}}^{\mathcal{S}}, m, \sigma)$ is *valid* if for all $K_{\mathsf{s}}^{\mathcal{S}}$ such that $(K_{\mathsf{p}}^{\mathcal{S}}, K_{\mathsf{s}}^{\mathcal{S}})$ could be output by $\mathsf{Setup}^{\mathcal{S}}$, $\mathsf{Sign}(K_{\mathsf{s}}^{\mathcal{S}}, m) = \sigma$. Otherwise, we say that $(K_{\mathsf{p}}^{\mathcal{S}}, m, \sigma)$ is *invalid*. We let Val resp. coVal be the set of all valid resp. invalid $(K_{\mathsf{p}}^{\mathcal{S}}, m, \sigma)$ triplets.

Confirm and Deny Let $(m, \sigma) \in \mathcal{M} \times \Sigma$ be a message-signature pair. Confirm resp. Deny are interactive algorithms that, together with the set of all possible $(K_{\mathsf{p}}^{\mathcal{V}}, K_{\mathsf{s}}^{\mathcal{V}})$,

[3] Sometimes in the literature the signature space depends on the signing key. Since we do not need this, we assume a "fixed" domain (i.e., depending on k) for simplicity reasons.

make a ZK argument of membership between $\mathcal{S}$ and $\mathcal{V}$ for language Val (resp. coVal). The pair of interactive algorithms $(\mathsf{Confirm}_{\mathcal{S}}, \mathsf{Confirm}_{\mathcal{V}})$ resp. $(\mathsf{Deny}_{\mathcal{S}}, \mathsf{Deny}_{\mathcal{V}})$ has a tuple $(K_{\mathsf{p}}^{\mathcal{S}}, m, \sigma, K_{\mathsf{p}}^{\mathcal{V}})$ as common input, witness input $K_{\mathsf{s}}^{\mathcal{S}}$ for $\mathcal{S}$, and auxiliary input $K_{\mathsf{s}}^{\mathcal{V}}$ for $\mathcal{V}$. In these ZK arguments, provers must be polynomially bounded. (Recall that verifiers are bounded by definition of interactive algorithms.)

An execution of the confirmation resp. denial between $\mathcal{S}$ and $\mathcal{V}$ with private input $K_{\mathsf{s}}^{\mathcal{S}}$ and $K_{\mathsf{s}}^{\mathcal{V}}$ will be denoted by $\mathsf{Verify}_{\mathcal{S}}(K_{\mathsf{s}}^{\mathcal{S}}; r_S) \overset{K_{\mathsf{p}}^{\mathcal{S}}, m, \sigma, K_{\mathsf{p}}^{\mathcal{V}}}{\longleftrightarrow} \mathsf{Verify}_{\mathcal{V}}(K_{\mathsf{s}}^{\mathcal{V}}; r_V)$ for $\mathsf{Verify} = \mathsf{Confirm}$ or $\mathsf{Verify} = \mathsf{Deny}$. **Correctness** of the undeniable signature scheme comes from the completeness of the two interactive protocols. This requires that under honest execution of all algorithms, a valid resp. invalid tuple is always proven as such by the signer to the designated verifier.

Remark 3.1. For basic undeniable signature, we do not need keys for the verifier. They will be needed later to address nontransferability. This way, the signer can decide to run an argument for a verifier which is designated by its public key in the tuple instance.

Remark 3.2. Following Kurosawa [32], the undeniable signature $\mathsf{Setup}^{\mathcal{S}}$ should provide a signature simulator together with $K_{\mathsf{p}}^{\mathcal{S}}$ to guarantee invisibility. Here we assume that this simulator always generates uniformly distributed samples in Σ and does not depend on $K_{\mathsf{p}}^{\mathcal{S}}$.

Following [32], the scheme should also provide a $\mathsf{Check}(K_{\mathsf{s}}^{\mathcal{S}}, m, \sigma)$ signature checking algorithm. Here we concentrate on deterministic Sign algorithm, so that $\mathsf{Check}(K_{\mathsf{s}}^{\mathcal{S}}, m, \sigma)$ consists of checking the $\sigma = \mathsf{Sign}(K_{\mathsf{s}}^{\mathcal{S}}, m)$ equation.

Following [32], undeniable signatures come with the following security properties: unforgeability and invisibility, in addition to the ZK properties of Confirm and Deny. Unforgeability ensures *nonrepudiation*, so that we can call the scheme a signature scheme. Nonrepudiation is formalized by resisting adaptive existential forgery attacks. Invisibility with respect to an active attacker who tries to distinguish a valid message–signature pair from a randomly picked one is considered. It thus protects privacy. ZK argument of membership assumes computational *soundness* of the proof and *zero-knowledge*. In addition to this, we will consider *nontransferability*. This last property ensures that a malicious verifier is not able to convince any third party of the validity of the statement (e.g., a given message signature is valid) proven in the protocol. The nontransferability notion may be important in some applications where the validity of the argument itself is valuable (like for licensing software).

We consider the standard security notion of existential forgery under an adaptive chosen-message attack as defined by Goldwasser et al. [28] for classical digital signatures. This notion is similar to Kurosawa–Heng [33] and is adapted as follows.

Existential Unforgeability. An undeniable signature scheme is secure against an existential forgery under adaptive chosen-message attack if there exists no probabilistic polynomial-time algorithm $\mathcal{F}$ which wins the following game with a nonnegligible probability.

Game: $\mathcal{F}$ receives a public key $K_{\mathsf{p}}^{\mathcal{S}}$ from $(K_{\mathsf{p}}^{\mathcal{S}}, K_{\mathsf{s}}^{\mathcal{S}}) \leftarrow \mathsf{Setup}^{\mathcal{S}}(1^k)$ and a verifier's key pair $(K_{\mathsf{p}}^{\mathcal{V}}, K_{\mathsf{s}}^{\mathcal{V}}) \leftarrow \mathsf{Setup}^{\mathcal{V}}(1^k)$. Then, $\mathcal{F}$ can query some chosen messages to a signing oracle, some chosen pairs $(m, \sigma) \in \mathcal{M} \times \Sigma$ to a confirmation (and denial) protocol oracle, and interact with it in a confirmation (denial) protocol where the oracle plays the role of the signer. All these queries must be polynomially bounded in k and can be sent adaptively. $\mathcal{F}$ wins the game if it outputs a valid pair $(m^*, \sigma^*) \in \mathcal{M} \times \Sigma$ such that m^* was not queried to the signing oracle.

The success probability of $\mathcal{F}$ in this game is denoted by $\mathsf{Succ}_{\mathcal{F}}^{\mathsf{ef\text{-}cma}}$.

We use a similar definition as Kurosawa–Heng [33].

Invisibility. Consider first a probabilistic polynomial-time algorithm $\mathcal{D}$ called *invisibility distinguisher* and the two following games with respect to a bit b.

Game$^{\mathsf{inv\text{-}cma\text{-}}b}$. $\mathcal{D}$ receives the public key $K_{\mathsf{p}}^{\mathcal{S}}$ from $(K_{\mathsf{p}}^{\mathcal{S}}, K_{\mathsf{s}}^{\mathcal{S}}) \leftarrow \mathsf{Setup}^{\mathcal{S}}(1^k)$ and a verifier's key pair $(K_{\mathsf{p}}^{\mathcal{V}}, K_{\mathsf{s}}^{\mathcal{V}}) \leftarrow \mathsf{Setup}^{\mathcal{V}}(1^k)$, it can query some chosen messages to a signing oracle and some chosen message–signature pairs $(m, \sigma) \in \mathcal{M} \times \Sigma$ to an oracle telling whether (m, σ) is valid or not. At some point, $\mathcal{D}$ chooses one message $m^* \in \mathcal{M}$ which was not queried to the signing oracle and submits it to the challenger. If $b = 0$, he sets $\sigma^* = \mathsf{Sign}(K_{\mathsf{s}}^{\mathcal{S}}, m^*)$. Otherwise, σ^* is picked uniformly at random in Σ. $\mathcal{D}$ receives σ^*. After that, the distinguisher can query the signing, confirmation, and denial oracles again, provided that m^* is not a query of the signing oracle and (m^*, σ^*) is not a query of the confirmation or denial protocols. Finally, $\mathcal{D}$ outputs a guess bit b'.

We define the advantage of the distinguisher as follows:

$$\mathsf{Adv}_{\mathcal{D}}^{\mathsf{inv\text{-}cma}} = \left|\Pr\left[b' = 1 \text{ in } \mathbf{Game}^{\mathsf{inv\text{-}cma\text{-}1}}\right] - \Pr\left[b' = 1 \text{ in } \mathbf{Game}^{\mathsf{inv\text{-}cma\text{-}0}}\right]\right|,$$

where probabilities are over the random tapes of the involved algorithms. An undeniable signature scheme is said to be *invisible under a chosen-message attack* if there exists no probabilistic polynomial-time algorithm $\mathcal{D}$ with a nonnegligible advantage.

Note that this definition is similar to that of Galbraith et al. [23] except that the distinguisher is not allowed to query m^* to the signing oracle in our definition. The invisibility notion of Galbraith et al. cannot be satisfied when the signature is deterministic (which is the case for MOVA). This will be discussed in Remark 5.3.

4. Interpolation of Group Homomorphisms

4.1. *Problem Definitions*

We define several generic problems related to some arbitrary Abelian groups. Later, these groups will be generated by a specific setup algorithm, and some attached trapdoor will render some of the problems easy.

The concept of group homomorphism interpolation is defined below.

Definition 4.1. Let G, H be two Abelian groups, and S be a subset of $G \times H$ written in the form $S = \{(x_1, y_1), \ldots, (x_s, y_s)\}$.

1. We say that the set of points S *interpolates in a group homomorphism* if there exists a group homomorphism $f : G \longrightarrow H$ such that $f(x_i) = y_i$ for $i = 1, \ldots, s$.
2. We say that a set of points $B \subseteq G \times H$ *interpolates in a group homomorphism with another set of points* $A \subseteq G \times H$ if $A \cup B$ interpolates in a group homomorphism.

4.1.1. *Group Homomorphism Interpolation Problem*

We state here the Group Homomorphism Interpolation problem (GHI problem) and its corresponding decisional problem (GHID problem). All problems assume parameters defining two Abelian groups G and H, a set $S \subseteq G \times H$ of cardinality s, and a positive integer n. We define the language $L_{\mathsf{GHI}}(n, S) \subseteq (G \times H)^n$ of all tuples interpolating with S in a group homomorphism and its complement $L_{\mathsf{coGHI}}(n, S) = (G \times H)^n \setminus L_{\mathsf{GHI}}(n, S)$.

n-S-GHI Problem (n-S-Group Homomorphism Interpolation Problem):
Instance: n elements $x_1, \ldots, x_n$ in G.
Problem: Find $y_1, \ldots, y_n \in H$ such that $((x_1, y_1), \ldots, (x_n, y_n)) \in L_{\mathsf{GHI}}(n, S)$.

The success probability of an n-S-GHI solver $\mathcal{A}$ is denoted by $\mathsf{Succ}^{n\text{-}S\text{-}\mathsf{GHI}}_{\mathcal{A}}$.

n-S-GHID Problem (n-S-GHI Decisional Problem):
Instance Generation: The instance T is generated according to one of the two following ways and is denoted T_0 or T_1, respectively. Following T_0, an n-tuple of points is picked uniformly at random in the language $L_{\mathsf{GHI}}(n, S)$. Following T_1, an n-tuple of points is picked uniformly at random in $(G \times H)^n$.
Problem: Decide whether the instance T is of type T_0 or T_1.

The advantage of an n-S-GHID distinguisher $\mathcal{D}$ is given by

$$\mathsf{Adv}^{n\text{-}S\text{-}\mathsf{GHID}}_{\mathcal{D}} = \left| \Pr_{\mathsf{instance} \in_U L_{\mathsf{GHI}}(n,S)} [\mathcal{D}(\mathsf{instance}) = 0] - \Pr_{\mathsf{instance} \in_U (G \times H)^n} [\mathcal{D}(\mathsf{instance}) = 0] \right|.$$

Remark 4.2. The uniform distribution in $L_{\mathsf{GHI}}(n, S)$ does not seem easy to produce in general. However, when S uniquely determines a homomorphism f, one can generate T_0 by picking the x_i's uniformly at random and setting $y_i = f(x_i)$ for $i = 1, \ldots, n$.

We will only consider n-S-GHI and n-S-GHID problems with a set S which interpolate in a *unique* group homomorphism. The n-S-GHI problem consists in evaluating the uniquely defined homomorphism on n elements. The n-S-GHID problem essentially consists in deciding whether all points of T lie in its graph. Later, in Sect. 4.2, we will provide instances of GHI and GHID problems.

4.1.2. *Related Computational Problems*

We also consider the following problems.

d-G-MGGD Problem in G (Modular Group Generation Decisional Problem):

Parameters: An Abelian group G and a positive integer d.
Instance: A set of values $S_1 \subseteq G$.
Problem: Does S_1 modulo dG span G/dG?

By Lemma 4.3 below, we show that $S = \{(x_1, y_1), \ldots, (x_s, y_s)\}$ interpolates in at most one group homomorphism if and only if $S_1 = \{x_1, \ldots, x_s\}$ spans G/dG modulo dG, where d is the order of H. We let $L_{\mathsf{MGGD}}(d, G)$ be the set of all S_1 that span G/dG.

(d, S_1)-MSR Problem in G (Modular System Representation Problem):
Parameters: An Abelian group G, a set $S_1 \subseteq G$ (of values denoted $x_1, \ldots, x_s$ below), and a positive integer d.
Instance: An element $x \in G$.
Problem: Find $a_1, \ldots, a_s \in \mathbb{Z}$ such that $x \in a_1 x_1 + \cdots + a_s x_s + dG$. If no solution exists, output $\perp$.

d-G-Root Problem in G (dth Root Problem):
Parameters: An Abelian group G and a positive integer d.
Instance: An element $x \in dG$.
Problem: Find $r \in G$ such that $x = dr$.

Group Expert When a participant has a key to run a polynomial-time algorithm to solve the (d, S_1)-MSR and the d-G-Root problems, we say that it is a *group expert* for G relative to (d, S_1).

4.2. *Preliminaries*

This subsection is devoted to technical lemmas related to the interpolation of group homomorphisms. In particular, we provide some criteria for a set to interpolate in at most one group homomorphism, and we show how to sample elements uniformly.

4.2.1. *Uniqueness of the Interpolation*

Lemma 4.3. *Let G, H be two finite Abelian groups, and d be the order of H. Let $x_1, \ldots, x_s \in G$ span a subgroup denoted by G'. The following properties are equivalent. In this case, we say that $x_1, \ldots, x_s$ H-generate G*

1. *For any $y_1, \ldots, y_s \in H$, there exists at most one group homomorphism $f : G \longrightarrow H$ such that $f(x_i) = y_i$ for all $i = 1, \ldots, s$.*
2. *There exists a unique group homomorphism $\varphi : G \longrightarrow H$ such that $\varphi(x_i) = 0$ for $i = 1, \ldots, s$, namely $\varphi = 0$.*
3. *The set $\mathsf{Hom}(G/G', H)$ of all group homomorphisms from G/G' to H is restricted to $\{0\}$.*
4. *$\gcd(\#(G/G'), d) = 1$.*
5. *$G' + dG = G$.*
6. *The cosets $x_1 + dG, \ldots, x_s + dG$ span G/dG.*

Proof. **1** $\Rightarrow$ **2** This directly follows by choosing $y_i = 0$ for all $i = 1, \ldots, s$.

2 $\Rightarrow$ **1** Assume that there exist two group homomorphisms f_1, f_2 from G to H such that $f_1(x_i) = f_2(x_i) = y_i$ for all $i = 1, \ldots, s$. Then, by assertion 2, we deduce that the group homomorphism $f_1 - f_2$ must be equal to the homomorphism 0.

2 $\Rightarrow$ **3** Suppose that there exists a homomorphism $\bar{\varphi} : G/G' \to H$ which is not equal to 0. Let $\pi_{G'} : G \to G/G'$ denote the canonical projection. We define the homomorphism $\varphi = \bar{\varphi} \circ \pi_{G'}$ from G to H. By definition, $\pi_{G'}(x_i) = 0$, and therefore $\varphi(x_i) = 0$ for any $i = 1, \ldots, s$. Moreover, since $\pi_{G'}$ is onto and $\bar{\varphi}$ is not trivial, φ must be different from 0, which contradicts assertion 2.

3 $\Rightarrow$ **4** Suppose the existence of a common prime factor p of $\#(G/G')$ and d. Then, from the structure of Abelian groups, G/G' and H must both possess one cyclic subgroup U and V, respectively, of order p. Let λ' denote the exponent of the group G/G'. By the structure of Abelian groups, we can choose U of the form $\lambda'/p \cdot (G/G')$. Hence, we have a group homomorphism

$$\begin{aligned} \varphi : G/G' &\longrightarrow U \\ x &\longmapsto \tfrac{\lambda'}{p} x \end{aligned}$$

which is onto. So, we can define a nontrivial homomorphism which is the composition of φ and the isomorphism between U and V. This contradicts 3.

4 $\Rightarrow$ **5** Let $x \in G$, and let $k = \mathsf{ord}(x \bmod G')$ be the order of $x \bmod G'$ in the quotient group G/G'. By assertion 4, d must be invertible modulo k. Let $m \in \mathbb{Z}$ be such that $m \cdot d \equiv 1 \pmod{k}$. We have $m \cdot d \cdot x \equiv x \pmod{G'}$. Hence, $x - d(m \cdot x) \in G'$, and therefore $x \in G' + dG$.

5 $\Rightarrow$ **2** Let $\varphi \in \mathsf{Hom}(G, H)$ such that $\varphi|_{G'} = 0$ and $x \in G$. By assertion 5, we can write $x = a_1 x_1 + \cdots + a_s x_s + dr$ for some integers $a_1, \ldots, a_s$ and an element $r \in G$. Thus, $\varphi(x) = d\varphi(r) = 0$. This holds for any $x \in G$, i.e., $\varphi = 0$.

5 $\Leftrightarrow$ **6** This follows from $G' + dG = G \Leftrightarrow \{x' + dG \mid x' \in G'\} = G/dG$. □

Remark 4.4. Replacing d by the exponent λ of H in assertions 5 and 6 leads to some equivalent assertions.

Note that the criteria 4–6 suggest that H is only involved by the prime factors in its order. Later, the smallest prime factor p will play an important role. Note also that if $G = H$, these criteria mean that $x_1, \ldots, x_s$ generate G. Furthermore, from assertion 6 we see that $S_1 \in L_{\mathsf{MGGD}}(d, G)$ is equivalent to, say, that S_1 H-generate G.

4.2.2. *Existence of the Interpolation*

Here is a condition allowing one to determine whether a set of points interpolates in a group homomorphism. The following result assumes that the G-coordinates of this set of points H-generate G so that the group homomorphism is unique when it exists.

Lemma 4.5. *Let G, H, and d be as in Lemma 4.3. Let $x_1, \ldots, x_s \in G$ H-generate G. The set $S = \{(x_1, y_1), \ldots, (x_s, y_s)\} \subseteq G \times H$ interpolates in a group homomorphism if and only if for any $a_1, \ldots, a_s \in \mathbb{Z}$ such that $a_1 x_1 + \cdots + a_s x_s \in dG$, we have $a_1 y_1 + \cdots + a_s y_s = 0$.*

Proof. "⇒" By assumption, there exists a homomorphism $f : G \to H$ such that $f(x_i) = y_i$ for $i = 1, \ldots, s$. Since dG lies in the kernel of f, we have $f(a_1x_1 + \cdots + a_sx_s) = a_1y_1 + \cdots + a_sy_s = 0$, whenever $a_1x_1 + \cdots + a_sx_s \in dG$.

"⇐" By assertion 5 of Lemma 4.3, we know that any element $x \in G$ can be written in the form $x = dr + a_1x_1 + \cdots + a_sx_s$ for some integers $a_1, \ldots, a_s$ and an element $r \in G$. We now define a function $f : G \to H$ such that $f(dr + a_1x_1 + \cdots + a_sx_s) = a_1y_1 + \cdots + a_sy_s$ for any $a_1, \ldots, a_s \in \mathbb{Z}$ and $r \in G$. It remains to prove that f is well defined on G and that it is homomorphic. Assume that an element $x \in G$ admits two different representations, i.e.,

$$x = dr + a_1x_1 + \cdots + a_sx_s = dr' + a'_1x_1 + \cdots + a'_sx_s.$$

By assumption, we must have $(a_1 - a'_1)y_1 + \cdots + (a_s - a'_s)y'_s = 0$, and therefore

$$f(dr + a_1x_1 + \cdots + a_sx_s) = f(dr' + a'_1x_1 + \cdots + a'_sx_s).$$

Finally, the homomorphic property of f follows from the linearity in the a_i's. □

Remark 4.6. Note that we can replace d by the exponent λ of the group H in Lemma 4.5.

Remark 4.7. Lemma 4.5 does not hold anymore if we relax the assumption stating that the elements $x_1, \ldots, x_s$ H-generate G. Choosing $G = \mathbb{Z}_{27}$, $H = \mathbb{Z}_9 \oplus \mathbb{Z}_3$, $s = 1$, $x_1 = 3$, and $y_1 = (0, 1)$ illustrates this fact.

4.2.3. *Examples of GHI and GHID Problems*

We can often meet the GHI and GHID problems in cryptography as the following examples suggest. Here, we exclusively consider 1-GHI and 1-GHID variants.

Example 4.8 (DL parameters). We take a cyclic group G of order q, $H = \mathbb{Z}_q$, and a generator g of G. The set $S = \{(g, 1)\}$ interpolates in a unique group homomorphism, and the GHI problem is exactly the discrete logarithm problem. The GHID problem is easy.

Example 4.9 (DH parameters). We take a cyclic group $G = H$ of order q and a generator g of G. For any $a \in \mathbb{Z}_q$, $S = \{(g, ag)\}$ interpolates in a unique group homomorphism: the exponentiation to the power a. The GHI and GHID problems correspond to the Diffie–Hellman problem [22] and the decisional Diffie–Hellman problem with static key a.

Example 4.10 (QR parameters). Let $n = pq$ be such that p, q are different odd primes and $H = \{-1, +1\}$. We let $x_1, x_2 \in \mathbb{Z}_n^*$ be such that x_1 is a quadratic residue modulo p and not modulo q and that x_2 is a quadratic residue modulo q, and not modulo p. We notice that $S = \{(x_1, 1), (x_2, -1)\}$ interpolates in a unique group homomorphism which is the Legendre symbol $(\cdot/p)$. Since it is easy to compute $(\cdot/n)$, the quadratic residuosity problem [27] with the information x_1 and x_2 is equivalent to the GHI and GHID problems.

Example 4.11 (RSA parameters). Here, we consider the well-known RSA cryptosystem [51]. Let $n = pq$ be an RSA modulus, and $G = H = \mathbb{Z}_n^*$. Let $f : \mathbb{Z}_n^* \to \mathbb{Z}_n^*$ be defined by $f(x) = x^e \bmod n$ for an exponent e such that $\gcd(e, \varphi(n)) = 1$. Given enough many pairs $(x_i^e \bmod n, x_i) \in \mathbb{Z}_n^* \times \mathbb{Z}_n^*$, $i = 1, \ldots, s$, such that the first coordinates generate $\mathbb{Z}_n^*$, the RSA problem corresponds to the GHI problem with S composed of the above pairs and e known. If e is known, the GHID problem is easy in this case.

Example 4.12 (BDH parameters). We show here how we can apply the GHI problem to the Bilinear Diffie–Hellman Problem (BDHP). This problem was used in the seminal paper of Boneh and Franklin [7,8] to propose an identity-based encryption scheme based on it. Let $\hat{e} : G_1 \times G_1 \to G_2$ be a bilinear, nondegenerate and computable mapping, where G_1 and G_2 are cyclic groups of a large prime order p. Let P be a generator of G_1; we can state the BDHP as follows: given three random elements aP, bP and $cP \in G_1$, compute $\hat{e}(P, P)^{abc}$. (G_1 resp. G_2 is written additively resp. multiplicatively.) BDHP is equivalent to the GHI problem with the set $S = \{(P, \hat{e}(aP, bP))\}$ and $x_1 = cP$ when S is refreshed for each instance with some a and b picked uniformly at random in $\mathbb{Z}_p$.

Example 4.13 (Paillier parameters). Let us consider the Paillier trapdoor function [48] that maps an element $(x, y) \in \mathbb{Z}_n \times \mathbb{Z}_n^*$ to the element $g^x \cdot y^n \bmod n^2$ of $\mathbb{Z}_{n^2}^*$, with g an element of $\mathbb{Z}_{n^2}^*$ of order n. For such a g, the Paillier trapdoor function is an isomorphism. Thus, assuming that we have s pairs of plaintext/ciphertext that generate $\mathbb{Z}_n \times \mathbb{Z}_n^*$ resp. $\mathbb{Z}_{n^2}^*$, the decryption problem of a challenged ciphertext corresponds to the GHI problem with $G = \mathbb{Z}_{n^2}^*$ and $H = \mathbb{Z}_n \times \mathbb{Z}_n^*$. This application of GHI problem to the decryption problem can be adapted to every homomorphic trapdoor function. Again, if the public key is known, the GHID problem is easy.

Note that Examples 4.9, 4.10, 4.11, 4.12, and 4.13 include trapdoors in order to interpolate the group homomorphism. Furthermore, Example 4.10 includes a trapdoor in order to solve the MSR and 2-G-Root problems, thus to make a group expert relative to any $(2, S_1)$. Also note that the order d of H is publicly known in Examples 4.8, 4.9, 4.10, 4.12. It is further quite small in Example 4.10. We will also consider the following example inspired by [1].

Example 4.14 (Newton parameters). Let $n = pq$ be such that $p = rd + 1$ and q are prime, $\gcd(r, d) = 1$, $\gcd(q - 1, d) = 1$, with d small prime. We take $G = \mathbb{Z}_n^*$ and $H = \mathbb{Z}_d$. We can easily compute a group homomorphism by first raising to the power $r(q - 1)$, then computing a discrete logarithm in a small cyclic subgroup of order d.

Our construction for a signature scheme requires the hardness of the GHI problem (for unforgeability), the hardness of the GHID problem (for invisibility), a trapdoor for interpolation (for computability), d publicly known, and sometimes a group expertise. So, in what follows, we focus on DH, QR, BDH, and Newton parameters in Examples 4.9, 4.10, 4.12, 4.14, respectively.

Parameters	GHI	GHID	Trapdoor	d known	Expert
DL	Hard	Easy	No	Yes	No
DH	**Hard**	**Hard**	**Yes**	**Yes**	**No**
QR	**Hard**	**Hard**	**Yes**	**Yes**	**Yes**
RSA	Hard	Easy	Yes	No	No
BDH	**Hard**	**Hard**	**Yes**	**Yes**	**No**
Paillier	Hard	Easy	Yes	No	No
Newton	**Hard**	**Hard**	**Yes**	**Yes**	**Yes**

4.2.4. *Sampling G uniformly*

Lemma 4.15. *Let $f : G \to H$ be a surjective group homomorphism from the group G to the group H. Then, f is balanced, i.e., $\#f^{-1}(y) = \#\mathsf{Ker}(f)$ for any $y \in H$.*

Proof. Let $x, x' \in G$ and $y = f(x)$. The lemma follows by noticing that $f(x') = y$ if and only if $x' \in x + \mathsf{Ker}(f)$. □

We provide a useful lemma to sample group elements.

Lemma 4.16. *Let G, H, d be defined as in Lemma 4.5. Let $x_1, \ldots, x_s \in G$ H-generate G. The following mapping from $G \times \mathbb{Z}_d^s$ to G is balanced:*

$$g : (r, a_1, \ldots, a_s) \longmapsto dr + a_1 x_1 + \cdots + a_s x_s. \tag{4.1}$$

By abuse of notation, $\mathbb{Z}_d$ denotes the set $\{0, 1, \ldots, d-1\}$ here. Note that g is not necessarily a group homomorphism because some x_i may have orders larger than d.

Proof. Let n be the order of G. Let $h : G \times \mathbb{Z}_{nd}^s \to G$ be the function defined by $h(r, a_1, \ldots, a_s) = dr + a_1 x_1 + \cdots + a_s x_s$. Obviously, h is a homomorphism. It is onto due to assertion 5 of Lemma 4.3. Hence, it is balanced by Lemma 4.15. Let $\varphi : G \times \mathbb{Z}_{nd}^s \to G \times \mathbb{Z}_d^s$ be a function defined by $\varphi(r, a_1, \ldots, a_s) = (r + q_1 x_1 + \cdots + q_s x_s, a_1 \bmod d, \ldots, a_s \bmod d)$, where $a_i - (a_i \bmod d) = dq_i$ for $i = 1, \ldots, s$. We have $g \circ \varphi = h$. We note that φ is balanced onto $G \times \mathbb{Z}_d^s$ since

$$\begin{aligned}&\varphi^{-1}(r, a_1, \ldots, a_s)\\ &\quad = \left\{(r - q_1 x_1 - \cdots - q_s x_s, a_1 + dq_1, \ldots, a_s + dq_s) \mid (q_1, \ldots, q_s) \in \mathbb{Z}_n^s\right\}.\end{aligned}$$

If $\#g^{-1}(x) = m$, we have $mn^s = \#\varphi^{-1}(g^{-1}(x)) = \#h^{-1}(x) = (dn)^s$. Hence, $m = d^s$ does not depend on x, so g is balanced. □

Remark 4.17. Lemma 4.16 also holds if d is replaced by the exponent λ of the group H.

Note that a group expert relative to $(d, \{x_1, \ldots, x_s\})$ knows how to invert g in Lemma 4.16. We show here that this ability even allows us to pick an element in $g^{-1}(x)$ uniformly at random for any $x \in G$. Since $g \circ \varphi = h$, we have $g^{-1}(x) = \varphi \circ h^{-1}(x)$.

Because φ is balanced, it is enough to sample an element of $h^{-1}(x)$. For that, we only need one element, and we can randomize it by adding a random element from $h^{-1}(0)$. For that, we pick a tuple $\tau \in_U G \times \mathbb{Z}_{nd}^s$, apply our group expertise, and retrieve a tuple $t' \in g^{-1}(h(\tau))$. We can then compute $\tau - \psi(t')$ where $\psi(t')$ is any function such that $\varphi \circ \psi$ is the identity mapping. Finally, we obtain

$$t = \varphi\big(\psi(t_0) + \tau - \psi(t')\big) \quad \text{where } t_0 \in g^{-1}(x), t' \in g^{-1}\big(h(\tau)\big).$$

The representation t is uniformly distributed in $g^{-1}(x)$.

4.3. *Approximations of the Homomorphism*

In this subsection we present a hardness result of approximation related to the existence of a homomorphism which interpolates a set of points. This is inspired by the theory of checkable proofs [2,3].

Lemma 4.18. *Given two finite Abelian groups G and H, and a set of s points $S = \{(x_i, y_i) \mid i = 1, \ldots, s\} \subseteq G \times H$ such that $x_1, \ldots, x_s$ H-generate G. We let d be the order of H and p be its smallest prime factor. We assume that S does not interpolate in any group homomorphism and define for any $x \in G$ the set $U_x = \{(r, a_1, \ldots, a_s) \in G \times \mathbb{Z}_d^s \mid dr + a_1x_1 + \cdots + a_sx_s = x\}$. Then, for any $x \in G$ and any $y \in H$, we have*

$$\Pr_{(r,a_1,\ldots,a_s)\in_U U_x}[a_1y_1 + \cdots + a_sy_s = y] = 0 \text{ or } \delta$$

for a constant $\delta \leq 1/p$. Therefore, for any $x \in G$ and any function $f : G \to H$, we have

$$\Pr_{(r,a_1,\ldots,a_s)\in_U U_x}\big[f(x) = a_1y_1 + \cdots + a_sy_s\big] \leq \frac{1}{p}.$$

Proof. Let K be the subgroup of $\mathbb{Z}_d^s$ defined by $K = \{(a_1, \ldots, a_s) \in \mathbb{Z}_d^s \mid a_1x_1 + \cdots + a_sx_s \in dG\}$. By Lemma 4.5, the image of $g : (b_1, \ldots, b_s) \mapsto b_1y_1 + \cdots + b_sy_s$ defined on K is a subgroup of order greater or equal to p. Moreover, by Lemma 4.15, g is balanced on its image, which shows that

$$\Pr_{(b_1,\ldots,b_s)\in_U K}[b_1y_1 + \cdots + b_sy_s = y] = 0 \text{ or } \delta$$

for any $y \in H$, where $\delta = 1/|\mathsf{Im}(g)|$. Let x be an arbitrary element of G. We can deduce that for any fixed tuple $(r', b'_1, \ldots, b'_s) \in U_x$, we also have

$$\Pr_{(b_1,\ldots,b_s)\in_U K}\big[(b_1 + b'_1)y_1 + \cdots + (b_s + b'_s)y_s = y\big] = 0 \text{ or } \delta$$

for any $y \in H$. This is equivalent to $\Pr_{(a_1,\ldots,a_s)\in_U V_x}[a_1y_1 + \cdots + a_sy_s = y] = 0$ or δ, for any $y \in H$, where $V_x = \{(a_1, \ldots, a_s) \mid \exists r \in G \text{ s.t. } (r, a_1, \ldots, a_s) \in U_x\}$. Here, we remark that for any tuple $(a_1, \ldots, a_s) \in V_x$, there exists the same number of elements $r \in G$ such that $(r, a_1, \ldots, a_s) \in U_x$. Namely, this number is equal to the cardinality of

the kernel of the homomorphism $r \mapsto dr$ defined on G, which is equal to $\#G/\#dG$. From this we finally deduce that

$$\Pr_{(r,a_1,\ldots,a_s)\in_U U_x}[a_1y_1+\cdots+a_sy_s=y]=\Pr_{(a_1,\ldots,a_s)\in_U V_x}[a_1y_1+\cdots+a_sy_s=y]=0 \text{ or } \delta$$

for any $y \in H$. □

Corollary 4.19. *Let G, H, S, d, and p be as in Lemma 4.18. We assume that there exists a function $f : G \longrightarrow H$ such that*

$$\rho=\Pr_{(r,a_1,\ldots,a_s)\in_U G\times\mathbb{Z}_d^s}\left[f(dr+a_1x_1+\cdots+a_sx_s)=a_1y_1+\cdots+a_sy_s\right]>\frac{1}{p}.$$

The set of points S interpolates in a group homomorphism. Furthermore, given $x \in_U G$, the value $y = f(x)$ matches the unique interpolation with probability ρ.

The above result can be extended to the computability of the homomorphism. Similarly as above and using techniques of linear cryptanalysis [31], one can efficiently amplify such a function f to compute the homomorphism with a negligible error probability. More details can be found in [40,41].

4.4. *A 4-Move ZK Protocol for GHI and coGHI*

In this section, we develop some interactive proof protocols for GHI and coGHI under the assumption that group expert algorithm exists.

Let G, H be parameters of a GHID problem. Let $\ell \in \mathbb{N}$ be a security parameter. We note that "T interpolates with S" is equivalent to "$S \cup T$ interpolates," and so we use a single set N as input without loss of generality. We present here an interactive proof of membership in which a prover wants to convince a verifier that N interpolates in a group homomorphism $f : G \longrightarrow H$ used as a witness. This protocol is denoted $\mathbf{GHIproof}_\ell(N)$ and is depicted below.

$\mathbf{GHIproof}_\ell(N)$
Parameters: G, H, d, ℓ
Common input: $N = \{(g_1, e_1), \ldots, (g_n, e_n)\} \subseteq G \times H$, the public key $K_{\mathsf{p}}^{\mathcal{V}}$ of the verifier
Witness input: f such that $f(g_i) = e_i$, $i = 1, \ldots, n$.

1. The verifier picks $r_i \in_U G$ and $a_{i,j} \in_U \mathbb{Z}_d$ for $i = 1, \ldots, \ell$ and $j = 1, \ldots, n$. He computes $u_i = dr_i + a_{i,1}g_1 + \cdots + a_{i,n}g_n$ and $w_i = a_{i,1}e_1 + \cdots + a_{i,n}e_n$ for $i = 1, \ldots, \ell$. He sends $u_1, \ldots, u_\ell$ to the prover.
2. The prover computes the values $v_i = f(u_i)$ for $i = 1, \ldots, \ell$, picks dec, and the commitment $\mathsf{com} = \mathsf{Commit}(K_{\mathsf{p}}^{\mathcal{V}}, v_1, \ldots, v_\ell; \mathsf{dec})$. He sends com to the verifier.
3. The verifier sends all r_i's and $a_{i,j}$'s to the prover.
4. The prover checks that the u_i's were computed correctly by verifying that $u_i = dr_i + a_{i,1}g_1 + \cdots + a_{i,n}g_n$ for $i = 1, \ldots, \ell$. If not, he aborts the protocol. He then opens his commitment by sending dec.

5. The verifier checks that the commitment is opened correctly, i.e.,

$$\mathsf{com} = \mathsf{Commit}\big(K_{\mathsf{p}}^{\mathcal{V}}, w_1, \ldots, w_\ell; \mathsf{dec}\big).$$

If this is the case, the verifier accepts the proof. Otherwise, he rejects it.

The commitment scheme is crucial to achieve zero-knowledge, since it allows the prover to disclose the answers v_i's after having checked that the verifier generated the challenges correctly. Otherwise, a malicious verifier could use the prover as an oracle to evaluate the function f on any element of G. Note also that the parameter ℓ corresponds to the number of challenges sent by the verifier and is thus directly related to the security level of **GHIproof**. For asymptotic security, this one can be seen as a function of a global security parameter k.

Theorem 4.20. *We consider the above protocol* **GHIproof** *with parameters G, H, d, ℓ and a trapdoor commitment scheme which is computationally binding and perfectly hiding. We let S be a subset of $G \times H$ and S_1 be the set of all G-coordinates of its elements. Let t be an integer. If for any N_1 such that $S_1 \subseteq N_1 \subseteq G$, there exists a group expert algorithm relative to (d, N_1), then* **GHIproof**$(S \cup T)$ *is a ZK argument of membership for $T \in L_{\mathsf{GHI}}(t, S)$. More precisely, we have the following properties.*

1. *Let p be the smallest prime factor of d. For any $\varepsilon > p^{-\ell}$,* **GHIproof** *is ε-sound. From any cheating signer $\mathcal{S}^*$ who passes the protocol on an invalid N with a probability ε and a group expert we can construct an algorithm $\mathcal{B}$ which finds a collision on the commitment scheme with probability*

$$\mathsf{Succ}_{\mathcal{B}}^{\mathsf{com\text{-}bnd}} \geq \varepsilon(\varepsilon - p^{-\ell})$$

   by rewinding $\mathcal{S}^*$ once.
2. **GHIproof** *is perfect black-box zero-knowledge.*
3. **GHIproof** *is perfect nontransferable.*

Proof. Completeness is trivial.

Soundness By Corollary 4.19, if the set N does not admit any interpolating homomorphism, it is impossible to find any procedure to deduce $w_i = \sum_{j=1}^{n} a_{i,j} e_j$ from $u_i = dr_i + \sum_{j=1}^{n} a_{i,j} g_j$ with probability greater than $1/p$, for any $i = 1, \ldots, \ell$. Since the challenges u_i's are generated independently, no prover is able to find the correct $w = (w_1, \ldots, w_\ell)$ from the challenge $u = (u_1, \ldots, u_\ell)$ with probability greater than $p^{-\ell}$. Below, we show that a (cheating) prover $\mathcal{P}^*$ must break the binding property of Commit with nonzero probability in order to succeed in the protocol with a probability $\varepsilon > p^{-\ell}$.

We construct below a simulator $\mathcal{B}$ who interacts with $\mathcal{P}^*$ and plays the role of an honest verifier. The simulator will launch two protocol runs sequentially with $\mathcal{P}^*$ in such a way that a collision on Commit can be found with a certain probability. For this, he rewinds the prover to play twice with the same inputs, coins, and challenge. We stress that this is allowed as long as the prover receives messages which are correctly

distributed in both runs of the protocol. So, if we look at both runs separately, the prover cannot see any difference from an interaction with an honest verifier.

The simulator $\mathcal{B}$ picks some coefficients $a_{i,j}$'s and r_i's uniformly at random and computes the challenges u_i's. He then sends u to $\mathcal{P}^*$. This one answers a committed value com. The simulator releases the coefficients $a_{i,j}$'s and r_i's to $\mathcal{P}^*$. At the end, $\mathcal{P}^*$ succeeds if he opens the commitment correctly on the values w_i's. If he does not succeed, the simulator aborts. Otherwise, using his group expertise, the simulator finds some coefficients $a^*_{i,j}$'s and r^*_i's that satisfy

$$u_i = dr^*_i + \sum_{j=1}^{n} a^*_{i,j} g_j \quad \text{for } i = 1, \ldots, \ell$$

and that are picked uniformly at random among all possible representations of the u_i's. Now, $\mathcal{B}$ rewinds the prover with the same random tape. The simulator sends the same challenges u_i's as before. Therefore, the prover answers the same commitment com. At this time, $\mathcal{B}$ releases the coefficients $a^*_{i,j}$'s and r^*_i's to $\mathcal{P}^*$. This one succeeds if he is able to open com on the values $w^*_i = \sum_{j=1}^{n} a^*_{i,j} e_j$ for $i = 1, \ldots, \ell$. In case of success, the simulator directly finds a collision with respect to Commit if $w_i \neq w^*_i$ holds for at least one i.

It remains to compute the probability that this event occurs. At first, we note that the simulations are perfect in both runs of the protocol. Namely, if we look at both protocol runs independently, we remark that all coefficients $r_i, a_{i,j}$ are chosen uniformly at random. We now have to take into account that rewinding the prover with the same random tape and the same challenges is a restriction in the space of all possible protocol runs. So, we decompose the probability of success according to the random tape ϖ of $\mathcal{P}$ and challenges u. Note that once the random tape and the challenge are fixed, both protocol runs are independent.

Let A be the probability event that $\mathcal{P}^*$ succeeds in the first protocol run, and $E_{\varpi,u}$ be the event that the random tape is ϖ and the challenge is u. Similarly, we define the same event A^* for the second protocol run. We set $\Pr[A|E_{\varpi,u}] = \Pr[A^*|E_{\varpi,u}] = \varepsilon_{\varpi,u}$. We also denote by B the event that $w \neq w^*$, where $w^* = (w^*_1, \ldots, w^*_\ell)$. Since A and A^* conditioned to $E_{\varpi,u}$ are independent, we have

$$\Pr[A \wedge A^*|E_{\varpi,u}] = \varepsilon^2_{\varpi,u}.$$

From this we deduce that

$$\Pr[A \wedge A^* \wedge B|E_{\varpi,u}] \geq \varepsilon^2_{\varpi,u} - \Pr[A \wedge A^* \wedge \neg B|E_{\varpi,u}] \geq \varepsilon^2_{\varpi,u} - \Pr[A \wedge \neg B|E_{\varpi,u}].$$

Note that $\Pr[\neg B|A, E_{\varpi,u}] \leq p^{-\ell}$ by Lemma 4.18: no matter the values of u and w, picking the a^* and r^* leading to the same u at random will give the same $w^* = w$ with probability bounded by $p^{-\ell}$. Hence, $\Pr[A \wedge \neg B|E_{\varpi,u}] \leq \varepsilon_{\varpi,u} p^{-\ell}$. We obtain that

$$\Pr[A \wedge A^* \wedge B|E_{\varpi,u}] \geq \varepsilon_{\varpi,u}\left(\varepsilon_{\varpi,u} - p^{-\ell}\right).$$

Applying Jensen's inequality, we obtain

$$\Pr[A \wedge A^* \wedge B] \geq \varepsilon\left(\varepsilon - p^{-\ell}\right),$$

where ε is the probability that $\mathcal{P}^*$ passes the proof with a honest verifier. This leads us to the desired result.

Zero-Knowledge We construct a black-box simulator $\mathcal{B}$ which is given an instance $T \in L_{\mathsf{GHI}}(n, S)$, the public key $K_{\mathsf{p}}^{\mathcal{V}}$ of the verifier, and an auxiliary input z for the malicious verifier $\mathcal{V}^*$. As usual, we let $N = S \cup T$.

1. The simulator first runs $\mathcal{V}^*(T, K_{\mathsf{p}}^{\mathcal{V}}, z; r_V)$ and gets the list u of the u_i's. (If these are not well formed, the simulator stops with $(T, K_{\mathsf{p}}^{\mathcal{V}}, z, r_V, \mathsf{abort})$ which perfectly simulates the view after interacting with the prover.)
2. Then, the simulator commits to some dummy v_i's and sends the commit value com to $\mathcal{V}^*$. Since the commitment is perfectly hiding, the commit value has the perfect distribution as if it was from the honest prover.
3. If $\mathcal{V}^*$ reveals incorrect $a_{i,j}$'s and r_i's, the simulator stops with $(T, K_{\mathsf{p}}^{\mathcal{V}}, z, r_V, \mathsf{com}, \mathsf{abort})$ which perfectly simulates the view. Otherwise, the simulator can now compute the correct v_i's, rewind the verifier, and go back to the commit phase on the correct v_i's.
4. After rewinding and committing again, if the verifier does not return any correct $a_{i,j}$'s and r_i's, rewind again until it works. Even though the $a_{i,j}$'s and r_i's may have changed, the v_i's are uniquely defined by the u_i's. So, any correct $a_{i,j}$'s and r_i's may allow the simulator to produce an accepting view with perfect distribution.

One caveat though: the loop in the last step may be nonpolynomial. However, the expected complexity of the simulator remains polynomial. Indeed, given some fixed $(T, K_{\mathsf{p}}^{\mathcal{V}}, z, r_V)$, let p be the probability (over the distribution induced by com) that $\mathcal{V}^*(T, K_{\mathsf{p}}^{\mathcal{V}}, z, \mathsf{com}; r_V)$ reveals some correct $a_{i,j}$'s and r_i's. The expected number of $\mathcal{V}^*(T, K_{\mathsf{p}}^{\mathcal{V}}, z, \mathsf{com}; r_V)$ calls in the simulator is

$$(1-p) + p\left(1 + \frac{1}{p}\right),$$

which is 2. So, we rewind only once on average and the simulator has a polynomial average complexity.

Nontransferability Following the definition of nontransferability, the malicious prover $\mathcal{P}^*$ is given the trapdoor $K_{\mathsf{s}}^{\mathcal{V}}$ of the commitment. So, after receiving the challenge $u = (u_1, \ldots, u_\ell)$ from $\mathcal{V}^*$, he can make a junk commitment by picking $w' = (w'_1, \ldots, w'_\ell) \in_U H^\ell$ uniformly at random and computing $\mathsf{com}' = \mathsf{Commit}(K_{\mathsf{p}}^{\mathcal{V}}, w'; \mathsf{dec}')$. Then he can send com' to $\mathcal{V}^*$. Then, the verifier sends the values r_i's and $a_{i,j}$'s to $\mathcal{P}^*$. He can check whether these values satisfy $u_i = dr_i + \sum_{j=1}^{n} a_{i,j} g_j$ for $i = 1, \ldots, \ell$. If it is not the case, he answers abort. If it is the case, he deduces the right tuple w and computes $\mathsf{dec} \leftarrow \mathsf{Equivocate}(K_{\mathsf{s}}^{\mathcal{V}}, w, \mathsf{com}')$. He then sends w, dec to open the commitment com'. Note that in this case, the transcript of this interaction is $(u, \mathsf{com}', r_i\text{'n}, a_{i,j}\text{'n}, w, \mathsf{dec})$. Since the commitment is perfectly hiding, the transcript has exactly the same distribution as a transcript produced between an honest prover and the verifier $\mathcal{V}^*$. □

Let G, H, and $S = \{(g_1, e_1), \ldots, (g_s, e_s)\} \subseteq G \times H$ be parameters of a GHID problem, and let d be the order of H with smallest prime factor p. Let $T = \{(x_1, z_1), \ldots, (x_t, z_t)\} \subseteq G \times H$ be a set of t points. We assume that S interpolates in a unique group homomorphism $f : G \to H$. A prover wants to convince a verifier that $T \notin L_{\mathsf{GHI}}(t, S)$. For this, the prover makes use of the knowledge of a group homomorphism f uniquely interpolating S. Let $\ell \in \mathbb{N}$ be a security parameter. He performs the following interaction with a verifier.

coGHIproof$_\ell(S, T)$

Parameters: G, H, d, p, ℓ, B such that $1 < B \leq p$ and B polynomially bounded.

Common input: $S = \{(g_1, e_1), \ldots, (g_s, e_s)\}$, $T = \{(x_1, z_1), \ldots, (x_t, z_t)\} \subseteq G \times H$, the public key $K_{\mathsf{p}}^{\mathcal{V}}$ of the verifier.

Witness input: f such that $f(g_i) = e_i$, $i = 1, \ldots, s$. We let $y_k = f(x_k)$, $k = 1, \ldots, t$.

1. The verifier picks $r_{i,k} \in_U G$, $a_{i,j,k} \in_U \mathbb{Z}_d$, and $\lambda_i \in_U \{0, 1, \ldots, B-1\}$ for $i = 1, \ldots, \ell$, $j = 1, \ldots, s$, $k = 1, \ldots, t$. He computes $u_{i,k} = dr_{i,k} + \sum_{j=1}^{s} a_{i,j,k} g_j + \lambda_i x_k$ and $w_{i,k} = \sum_{j=1}^{s} a_{i,j,k} e_j + \lambda_i z_k$ for all i and k. Set $u = (u_{1,1}, \ldots, u_{\ell,t})$ and $w = (w_{1,1}, \ldots, w_{\ell,t})$. He sends u and w to the prover.
2. The prover computes $v_{i,k} = f(u_{i,k})$ for $i = 1, \ldots, \ell$, $k = 1, \ldots, t$. By the equation $w_{i,k} - v_{i,k} = \lambda_i(z_k - y_k)$, he should be able to find each λ_i by exhaustive search if the verifier is honest, since $y_k \neq z_k$ for at least one k. Otherwise, he picks $\lambda_i \in_U \{0, 1, \ldots, B-1\}$ uniformly at random for $i = 1, \ldots, \ell$. He computes $\mathsf{com} = \mathsf{Commit}(K_{\mathsf{p}}^{\mathcal{V}}, \lambda; \mathsf{dec})$, where $\lambda = (\lambda_1, \ldots, \lambda_\ell)$. The prover sends the committed value com to the verifier.
3. The verifier sends all $r_{i,k}$'s and $a_{i,j,k}$'s to the prover.
4. The prover checks that u and w were correctly computed by verifying that $u_{i,k} = dr_{i,k} + \sum_{j=1}^{s} a_{i,j,k} g_j + \lambda_i x_k$ and $w_{i,k} = \sum_{j=1}^{s} a_{i,j,k} e_j + \lambda_i z_k$ for all i and k. If not, he aborts the protocol. He then opens the commitment by sending λ and dec.
5. The verifier checks that the prover has found the right λ and that the commitment is correctly opened by checking $\mathsf{com} = \mathsf{Commit}(K_{\mathsf{p}}^{\mathcal{V}}, \lambda; \mathsf{dec})$. If this is the case, the verifier accepts the proof. Otherwise, he rejects it.

Remark 4.21. Provided that the discrete logarithm in H is easy, we can take $B = p$ and replace exhaustive search on λ_i by a discrete logarithm algorithm.

This protocol was inspired from the denial protocol of Gennaro et al. [24]. This one can actually be seen as a special case of ours with the RSA encryption function as homomorphism.

We also notice that λ was chosen such that it can uniquely be retrieved from every nonzero values of H that can be taken by the elements $z_k - y_k$'s. This is shown by the following result.

Lemma 4.22. *Let H, d, p as above, and $a, b \in H$ such that $b \neq 0$. If the equation $a = \lambda b$ has a solution λ in $\{0, 1, \ldots, p-1\}$, then this one is unique.*

Proof. Let us first consider the subgroup $\langle b \rangle$ generated by b. If there exists a solution to the above equation, we must have $a \in \langle b \rangle$. Moreover, the coefficient λ is uniquely defined modulo $\mathsf{ord}(b)$. By definition of p, we have $\mathsf{ord}(b) \geq p$. Therefore, λ is uniquely defined in $\{0, 1, \ldots, p-1\}$. □

Theorem 4.23. *Let S be a set which interpolates in exactly one group homomorphism. We consider the above* **coGHIproof** *protocol with parameters G, H, d, p, ℓ, B. For any trapdoor commitment scheme which is computationally binding and perfectly hiding, if there exists a group expert relative to (d, S_1) for any $S_1 \subseteq G$, then* **coGHIproof** *is a ZK argument of membership for the language $L_{\mathsf{coGHI}}(t, S)$ for any integer t. More precisely, we have the following properties.*

1. *Let p be the smallest prime factor of d. For any $\varepsilon > B^{-\ell}$,* **coGHIproof** *is ε-sound. From any cheating prover $\mathcal{P}^*$ who passes the protocol on $T \notin L_{\mathsf{coGHI}}(t, S)$ with probability ε and a group expert relative to (d, S_1), we can construct an algorithm $\mathcal{B}$ which finds a collision on the commitment scheme with probability at least $\varepsilon(\varepsilon - B^{-\ell})$ by rewinding $\mathcal{P}^*$ once.*
2. **coGHIproof** *is perfect black-box zero-knowledge.*
3. **coGHIproof** *is perfect nontransferable.*

Proof. Completeness is trivial.

Soundness We first remark that $u_{i,k}$ is uniformly distributed for any fixed λ_i for $i = 1, \ldots, \ell$ and $k = 1, \ldots, t$ by Lemma 4.16. Moreover, if the set of points T interpolates in f, we have that $f(x_k) = z_k$ for all $k = 1, \ldots, t$. By the homomorphic property of f, we have $f(u_{i,k}) = w_{i,k}$ for any i and k. Putting all together implies that the distribution of the challenge $(u_{i,k}, w_{i,k})$ is completely independent of the value λ_i for any i and k. Thus, a prover cannot deduce the right λ_i's with a probability greater than $B^{-\ell}$ from the challenges. Below, we show how we can break the binding property of Commit using a prover succeeding with a probability $\varepsilon > B^{-\ell}$. To this, we proceed similarly as for proving the soundness of **GHIproof**. We run the protocol once with the prover, rewind this one, and run the protocol with carefully chosen coefficients.

The simulator $\mathcal{B}$ first picks the values $a_{i,j,k}$'s, $r_{i,k}$'s, and λ_i's uniformly at random and computes the tuples $u = (u_{1,1}, \ldots, u_{\ell,t})$ and $w = (w_{1,1}, \ldots, w_{\ell,t})$ according to the protocol. Then, $\mathcal{B}$ sends u, w to the prover $\mathcal{P}^*$. This one answers com. The simulator releases the coefficients $a_{i,j,k}$'s, $r_{i,k}$'s, λ_i's, and the prover succeeds if he opens com on $\lambda = (\lambda_1, \ldots, \lambda_\ell)$. Now, $\mathcal{B}$ picks $\lambda^* = (\lambda_1^*, \ldots, \lambda_\ell^*)$ uniformly at random. By using his group expertise, he is able to find some uniformly random coefficients $a_{i,j,k}^*$'s, $r_{i,k}^*$'s satisfying

$$u_{i,k} - \lambda_i^* x_k = d r_{i,k}^* + \sum_{j=1}^{s} a_{i,j,k}^* g_j \quad \text{for } i = 1, \ldots, \ell \text{ and } k = 1, \ldots, t.$$

The simulator rewinds $\mathcal{P}^*$ with the same random tape and the same challenges. He answers the same commitment com. This time, $\mathcal{B}$ sends $a^*_{i,j,k}$'s, $r^*_{i,k}$'s. The prover wins if he is able to open com on the value λ^*. If $\lambda^* \neq \lambda$, $\mathcal{B}$ breaks the computationally binding property of Commit.

Note that $\mathcal{B}$ simulates an honest verifier perfectly in both protocol runs. We can compute the success probability that $\mathcal{B}$ finds a collision for Commit in a very similar way as for **GHIproof**. Namely, we decompose the probability of success for the different random tapes ϖ and challenges u. Let $\varepsilon_{\varpi,u}$ be the probability that the prover wins in one protocol run with the random tape ϖ and the challenge u (and thus $w = f(u)$). Since the probability that $\lambda = \lambda^*$ for any random tape ϖ and challenge u is equal to $B^{-\ell}$, we can show as for **GHIproof** that the success probability of $\mathcal{B}$ is higher than

$$\sum_{\varpi,u} q_{\varpi,u} \cdot \left(\varepsilon^2_{\varpi,u} - \varepsilon_{\varpi,u} \cdot B^{-\ell}\right),$$

where $q_{\varpi,u}$ denotes the probability that the protocol runs with the random tape ϖ and the challenge u. Again, applying Jensen's inequality leads to the desired bound $\varepsilon(\varepsilon - B^{-\ell})$.

Zero-Knowledge The simulation works as in **GHIproof**. One can rewind the verifier and achieve perfect zero-knowledge. We can just check that λ is uniquely defined by u and w. Indeed, if $\mathcal{V}^*$ provides (after rewinding once) two sets (r, a, λ) and (r^*, a^*, λ^*) with same u and w, we obtain that

$$(\lambda_i - \lambda^*_i)\big(f(x_k) - z_k\big) = 0$$

for all i and k. So, if $f(x_k) \neq z_k$, we obtain that $\lambda_i = \lambda^*_i$ for all i.

Nontransferability The simulation works as in **GHIproof**. One can use the trapdoor of the commitment scheme to open the commitment on the desired value (correct answer). □

4.5. *A Two-Move ZK Protocol for GHI and coGHI in the Random Oracle Model*

Below, we propose a two-move variant of **GHIproof**. This variant is achieved by removing the two messages sent in the middle of the protocol which allow us to achieve zero-knowledge through the commitment scheme. In order to maintain zero-knowledge, the verifier sends a kind of commitment on a seed which generates the coefficients producing the challenges sent to the prover. This commitment can only be opened by the prover after this one solved these challenges. For this, we introduce a pseudorandom generator Gen and a cryptographic hash function denoted Gen′ which will be modeled by random oracles. We notably add a trapdoor one-way permutation with associated secret key $K_{\mathsf{s}}^{\mathcal{V}}$ in order to obtain nontransferability. This two-move protocol called **2GHIproof** is described here.

2GHIproof$_\ell(N)$
Parameters: G, H, d, ℓ, k.
Random oracles: $\mathsf{Gen} : \{0,1\}^k \to G^\ell \times \mathbb{Z}_d^{\ell n}$, $\mathsf{Gen}' : H^\ell \to \{0,1\}^k$.

Common input: $N = \{(g_1, e_1), \ldots, (g_n, e_n)\} \subseteq G \times H$, the public key $K_{\mathsf{p}}^{\mathcal{V}}$ of the verifier.

Witness input: f such that $f(g_i) = e_i$, $i = 1, \ldots, n$.

1. The verifier picks $\mathsf{seed} \in_U \{0,1\}^k$ and, by applying a pseudorandom generator Gen on this seed, generates values $r_i \in G$ and $a_{i,j} \in \mathbb{Z}_d$ for $i = 1, \ldots, \ell$ and $j = 1, \ldots, n$. He computes $u_i = dr_i + a_{i,1}g_1 + \cdots + a_{i,n}g_n$, $w_i = a_{i,1}e_1 + \cdots + a_{i,n}e_n$ for $i = 1, \ldots, \ell$, and $\vartheta = \mathsf{TPOW}(K_{\mathsf{p}}^{\mathcal{V}}, \mathsf{seed})$. Using a cryptographic hash function Gen', he computes $h = \mathsf{Gen}'(w_1, \ldots, w_\ell) \oplus \mathsf{seed}$. The verifier sends $u_1, \ldots, u_\ell$, h, and ϑ to the prover.
2. The prover computes $v_i = f(u_i)$ for $i = 1, \ldots, \ell$, $\mathsf{seed}' = \mathsf{Gen}'(v_1, \ldots, v_\ell) \oplus h$. He checks that $\vartheta = \mathsf{TPOW}(K_{\mathsf{p}}^{\mathcal{V}}, \mathsf{seed}')$ and that $\mathsf{Gen}(\mathsf{seed}')$ generates values $a_{i,j}$'s and r_i's such that $u_i = dr_i + a_{i,1}g_1 + \cdots + a_{i,n}g_n$ for $i = 1, \ldots, \ell$. If not, the prover aborts the protocol. He then sends seed' to the verifier.
3. The verifier accepts the proof if $\mathsf{seed}' = \mathsf{seed}$ holds. Otherwise, he rejects it.

Note that the secret key $K_{\mathsf{s}}^{\mathcal{V}}$ of the verifier is unused in the protocol. Its availability will be used to prove nontransferability.

The two-move variant has a very similar complexity as the four-move ones. In particular, the prover needs to perform the same number of homomorphism evaluations. The computational work related to Commit is replaced by the one induced by Gen, Gen', TPOW.

Theorem 4.24. *We consider the above* **2GHIproof** *protocol with parameters* G, H, d, ℓ, k. *We assume that* Gen *and* Gen' *are random oracles and that* TPOW *is a trapdoor one-way permutation defined on domain* $\{0,1\}^k$. **2GHIproof**$(S \cup T)$ *is a ZK argument of membership for* $T \in L_{\mathsf{GHI}}(n, S)$. *More precisely, we have the following properties.*

1. *Let* p *be the smallest prime factor of* d. *The protocol is sound: There is a probabilistic polynomial-time algorithm* $\mathcal{A}^{\mathcal{P}^*}$ *invoking* $\mathcal{P}^*$ *once and such that if* $\mathcal{P}^*$ *is a cheating prover limited to* $q_{\mathsf{Gen}'}$ *queries to* Gen' *and with probability of success* ε, *then* $\mathcal{A}^{\mathcal{P}^*}$ *inverts* TPOW *on domain* $\{0,1\}^k$ *with success probability higher than* $\varepsilon - q_{\mathsf{Gen}'} p^{-\ell}$.
2. **2GHIproof** *is statistical black-box straight-line deniable zero-knowledge in the random oracle model.*
3. **2GHIproof** *is perfect nontransferable.*

Proof. Completeness is trivial.

Soundness Let $\mathcal{P}^*$ be a cheating prover who wants to pass the protocol with common input N which does not interpolate in a group homomorphism. We let $K_{\mathsf{p}}^{\mathcal{V}}$ and $K_{\mathsf{s}}^{\mathcal{V}}$ be the keys for the verifier, and we assume that $\mathcal{P}^* \leftrightarrow \mathcal{V}$ accepts with probability ε.

When interacting with a honest verifier, $\mathcal{P}^*$ wins if and only if it responds by seed such that $\mathsf{TPOW}(K_{\mathsf{p}}^{\mathcal{V}}, \mathsf{seed}) = \vartheta$. Note that it is easy to check whether a value seed' is equal to seed because TPOW is deterministic. Without loss of generality, we can assume

that $\mathcal{P}^*$ never queries seed to Gen (indeed, he can first check if this is the correct answer before querying). Similarly, we can assume that $\mathcal{P}^*$ always responds correctly whenever he queries the right $w = (w_1, \ldots, w_\ell)$ to Gen' because he can check that $\mathsf{Gen}'(w) \oplus h$ is the correct seed. Therefore, $\mathcal{P}^*$ always wins if the correct w is queried to Gen' and never queries seed to Gen.

We transform $\mathcal{P}^*$ into an algorithm $\mathcal{A}$ inverting $\mathsf{TPOW}(K_{\mathsf{p}}^{\mathcal{V}}, \cdot)$ as follows.

1. $\mathcal{A}$ receives a random challenge ϑ whose preimage by TPOW is denoted seed. The goal of $\mathcal{A}$ is to find seed.
2. $\mathcal{A}$ generates some random values r_i's and $a_{i,j}$'s and deduces the corresponding u_i's and w_i's. $\mathcal{A}$ further picks a random h. Then (u, h, ϑ) is a challenge for the prover, and $\mathcal{A}$ can start simulating $\mathcal{P}^*$.

   Clearly, given a random Gen and Gen', the simulation for $\mathcal{P}^*$ is perfect as long as seed is not queried to Gen and w is not queried to Gen'. Since the former case never happens, $\mathcal{A}$ simulates $\mathcal{P}^*$ and stops if w is queried to Gen'. In this case, $\mathcal{A}$ just fails.
3. The simulation of $\mathcal{P}^*$ leads to either a failure or a release of the correct value seed which can be returned by $\mathcal{A}$.

The algorithm succeeds to invert the trapdoor permutation at the condition that (event A) $\mathcal{P}^*$ succeeds without querying w to Gen'. Let B be the event that $\mathcal{P}^*$ queries w to Gen'. Since the simulation is perfect, $\Pr[A] + \Pr[B]$ is the probability that $\mathcal{P}^*$ passes the protocol with an honest verifier. Clearly, $\Pr[A]$ is the probability that $\mathcal{A}$ succeeds in inverting TPOW. Thus $\Pr[A] \leq \mathsf{Succ}^{\mathsf{inv\text{-}tp}}$. Below we show an upper bound for $\Pr[B]$. To this end, we consider a simulator $\mathcal{B}$ which plays with $\mathcal{P}^*$ to win the following game:

> Game: A challenger picks elements r_i's and $a_{i,j}$'s uniformly at random and computes $u_i = dr_i + \sum_{j=1}^{n} a_{i,j} g_j$. The simulator $\mathcal{B}$ receives the u_i's and wins the game if he finds the correct values $w_i = \sum_{j=1}^{n} a_{i,j} e_j$ for $i = 1, \ldots, \ell$.

We transform $\mathcal{P}^*$ into an algorithm $\mathcal{B}$ as follows.

1. $\mathcal{B}$ receives a random challenge u.
2. $\mathcal{B}$ then picks a random seed, computes $\vartheta = \mathsf{TPOW}(K_{\mathsf{p}}^{\mathcal{V}}, \mathsf{seed})$, and picks a random h. Additionally, $\mathcal{B}$ picks an integer $\ell \in \{1, \ldots, q_{\mathsf{Gen}'}\}$ uniformly at random. Then (u, h, ϑ) is a challenge for the prover, and $\mathcal{B}$ can start simulating $\mathcal{P}^*$.

   Clearly, the simulation works just like for $\mathcal{A}$ as long as $\mathcal{P}^*$ does not query Gen' with the response w to the game. So, $\mathcal{B}$ just let $\mathcal{P}^*$ query Gen' for the $\ell - 1$ first queries and stops at the ℓth one. The value w' of this last query is returned as the response to the game.
3. If the simulation $\mathcal{P}^*$ stops, then $\mathcal{B}$ just fails.

Clearly, $\mathcal{B}$ wins if and only if event B occurs and the simulator has guessed which of the queries by $\mathcal{P}^*$ gave the correct answer. Therefore, $\mathcal{B}$ wins with probability $1/q_{\mathsf{Gen}'} \cdot \Pr[B]$. By Corollary 4.19, this probability is at most $p^{-\ell}$, which implies that $\Pr[B] \leq q_{\mathsf{Gen}'} p^{-\ell}$. So, the confirmation cannot succeed with probability larger than $\mathsf{Succ}^{\mathsf{inv\text{-}tp}} + q_{\mathsf{Gen}'} p^{-\ell}$.

Zero-Knowledge We want to construct a simulator $\mathcal{B}^{\mathcal{V}^*,\mathsf{Gen},\mathsf{Gen}'}$ using any verifier $\mathcal{V}^*$ as a subroutine to simulate the view from $\mathcal{V}^*$ in the protocol. One problem is that $\mathcal{V}^*$ may not have $K_{\mathsf{s}}^{\mathcal{V}}$ as an input by definition. So, we cannot assume that the simulator has it. $\mathcal{B}$ runs the verifier $\mathcal{V}^*$ and looks at the queries made by $\mathcal{V}^*$ to the oracle Gen. $\mathcal{B}$ puts these q_{Gen} queries seed_k for $1 \le k \le q_{\mathsf{Gen}}$ and the corresponding answers of Gen in memory. The simulator then receives the first message M of $\mathcal{V}^*$. If this one has not a correct format, the simulator outputs the abort view $(N, K_{\mathsf{p}}^{\mathcal{V}}, z, r_V, \mathsf{abort})$. Otherwise, the simulator checks whether one answer among those queries seed_k's made to Gen generates the challenges u_i's correctly and the image of this query by TPOW is equal to ϑ. If it is not the case, $\mathcal{B}$ outputs the abort view. Otherwise, the simulator is able to compute the right w_i's from this answer (the right r_i's and $a_{i,j}$'s). From the w_i's, $\mathcal{B}$ computes $\mathsf{seed}^* = h \oplus \mathsf{Gen}'(w_1, \ldots, w_\ell)$ and checks whether seed^* generates the right r_i's and $a_{i,j}$'s. In the positive case, $\mathcal{B}$ outputs the view $(N, K_{\mathsf{p}}^{\mathcal{V}}, z, r_V, \mathsf{seed}^*)$. In the negative case, it outputs the abort view.

It remains to show that the two view distributions are statistically indistinguishable. When the first message has not a correct format, the two transcripts are clearly identical. Let consider the case where the verifier did not query any seed_k which produces the challenges u_i's and whose image by TPOW leads to ϑ. In this case, the honest prover will not abort the protocol only if he retrieves a $\mathsf{seed} = \mathsf{Gen}'(w_1, \ldots, w_\ell) \oplus h$ which generates the challenges u_i's and ϑ. This occurs only if the verifier $\mathcal{V}^*$ was able to guess that the output values of the query seed to the oracle Gen generate the right r_i's and a_{ij}'s. Since Gen is a random oracle, no polynomial-time verifier $\mathcal{V}^*$ can succeed to do that with a nonnegligible probability. We still have to consider the case where the verifier queried a seed_k which produces the challenges u_i's and ϑ. We see that the two transcripts are always identical, since the simulator clearly knows the answer of the honest prover by learning the right w_i's. Therefore, we can conclude that the two transcript distributions are statistically indistinguishable.

Nontransferability We now want to construct an online prover simulator $\mathcal{P}^*$ which would be undistinguishable from the real one, provided that it is given the correct $K_{\mathsf{s}}^{\mathcal{V}}$. Based on receiving the (u, h, ϑ) challenge, if the format is incorrect, then $\mathcal{P}^*$ aborts. Otherwise, $\mathcal{P}^*$ can compute $\mathsf{seed} = \mathsf{TPOW}^{-1}(K_{\mathsf{s}}^{\mathcal{V}}, \vartheta)$, then check whether u and h are well computed from seed. If correct, $\mathcal{P}^*$ can eventually respond with seed. Clearly, the simulation is perfect. □

The interactive **coGHIproof** protocol can be transformed in a two-move protocol in a similar way as for **GHIproof**. This variant called **2coGHIproof** is presented below.

2coGHIproof$_\ell(S, T)$
Parameters: G, H, d, p, ℓ, k, B such that $1 < B \le p$ and B polynomially bounded.
Random oracles: $\mathsf{Gen} : \{0,1\}^k \to G^{\ell t} \times \mathbb{Z}_d^{\ell s t} \times \{0, \ldots, B-1\}^\ell$, $\mathsf{Gen}' : \mathbb{Z}_d^\ell \to \{0,1\}^k$.
Common input: $S = \{(g_1, e_1), \ldots, (g_s, e_s)\}$, $T = \{(x_1, z_1), \ldots, (x_t, z_t)\} \subseteq G \times H$, the public key $K_{\mathsf{p}}^{\mathcal{V}}$ of the verifier.
Witness input: f such that $f(g_i) = e_i$, $i = 1, \ldots, s$. We let $y_k = f(x_k)$, $k = 1, \ldots, t$.

1. The verifier picks $\mathsf{seed} \in_U \{0,1\}^k$ and, by applying a pseudorandom generator Gen on this seed, generates values $r_{i,k} \in G$, $a_{i,j,k} \in \mathbb{Z}_d$, $\lambda_i \in \{0,1,\ldots,B-1\}$ for $i = 1,\ldots,\ell$, $j = 1,\ldots,s$, $k = 1,\ldots,t$. He computes $u_{i,k} = dr_{i,k} + \sum_{j=1}^{s} a_{i,j,k} g_j + \lambda_i x_k$, $w_{i,k} = \sum_{j=1}^{s} a_{i,j,k} e_j + \lambda_i z_k$ for all i, k, and $\vartheta = \mathsf{TPOW}(K_{\mathsf{p}}^{\mathcal{V}}, \mathsf{seed})$. Using a cryptographic hash function Gen', the verifier computes $h = \mathsf{Gen}'(\lambda_1,\ldots,\lambda_\ell) \oplus \mathsf{seed}$. Set $u = (u_{1,1},\ldots,u_{\ell,t})$ and $w = (w_{1,1},\ldots,w_{\ell,t})$. He sends u, w, h, and ϑ to the prover.
2. The prover computes $v_{i,k} = f(u_{i,k})$ for $i = 1,\ldots,\ell$, $k = 1,\ldots,t$. By the equations $w_{i,k} - v_{i,k} = \lambda_i(z_k - y_k)$, he should be able to find every λ_i by exhaustive search if the verifier is honest since $y_k \neq z_k$ for at least one k. The prover computes $\mathsf{seed}' = \mathsf{Gen}'(\lambda_1,\ldots,\lambda_\ell) \oplus h$. He checks that $\vartheta = \mathsf{TPOW}(K_{\mathsf{p}}^{\mathcal{V}}, \mathsf{seed}')$ and that seed' generates coefficients $r_{i,k}$'s, $a_{i,j,k}$'s, λ_i's such that $u_{i,k} = dr_{i,k} + \sum_{j=1}^{s} a_{i,j,k} g_j + \lambda_i x_k$, $w_{i,k} = \sum_{j=1}^{s} a_{i,j,k} e_j + \lambda_i z_k$ for all i and k. If not, he aborts the protocol. He then sends seed' to the verifier.
3. The verifier accepts the proof if $\mathsf{seed}' = \mathsf{seed}$ holds. Otherwise, he rejects it.

Theorem 4.25. *We consider the above* **2coGHIproof** *protocol with parameters* G, H, d, p, ℓ, k, B. *Let* S *be a set which interpolates in exactly one group homomorphism. We assume that* Gen *and* Gen' *are random oracles and that* TPOW *is a trapdoor one-way permutation defined on domain* $\{0,1\}^k$. **2coGHIproof** *is a ZK proof of membership for the language* $L_{\mathsf{coGHI}}(t, S)$. *More precisely, we have the following properties.*

1. *Let* p *be the smallest prime factor of* d. **2coGHIproof** *is sound: From any cheating prover* $\mathcal{P}^*$ *limited to* $q_{\mathsf{Gen}'}$ *queries to* Gen' *who passes the protocol on* $T \notin L_{\mathsf{coGHI}}(t, S)$ *with probability* ε *we can construct an algorithm* $\mathcal{A}$ *to invert* TPOW *on domain* $\{0,1\}^k$ *with probability of success higher than* $\varepsilon - q_{\mathsf{Gen}'} B^{-\ell}$.
2. **2coGHIproof** *is statistical black-box straight-line deniable zero-knowledge in the random oracle model.*
3. **2coGHIproof** *is perfect nontransferable.*

Proof. This proof is very similar to the one of Theorem 4.24. □

4.6. *A NIZK for MGGD*

Let G, d be some parameters, and S_1 some input of a d-G-MGGD problem. We propose here a noninteractive proof in which a prover proves that $S_1 = \{g_1,\ldots,g_s\}$ H-generate G, for any group H of order d. In other words, by assertion 5 of Lemma 4.3 this corresponds to show that $\langle S_1 \rangle + dG = G$ or by assertion 6 of the same lemma that the answer to the d-G-MGGD problem is positive. For this, the prover must be a group expert relative to (d, S_1).

We first introduce a technical result.

Lemma 4.26. *Given a finite Abelian group* G, *a subset* $S_1 = \{g_1,\ldots,g_s\} \subseteq G$, *and an integer* d *with smallest prime factor* p. *We assume that there exists a function* $f : G \to$

$G \times \mathbb{Z}_d^s$ *satisfying*

$$\Pr_x\left[x = dr + a_1 g_s + \cdots + a_s g_s \mid x \in_U G;\ f(x) = (r, a_1, \ldots a_s)\right] > \frac{1}{p}.$$

Then, we have $G = \langle S_1 \rangle + dG$, *i.e.,* S_1 *H-generate* G *for any Abelian group* H *of order* d.

Proof. First, we notice that if $\gcd(\#G, d) = 1$, we have $dG = G$, which trivially leads to $\langle S_1 \rangle + dG = G$. Now, assuming that $\gcd(\#G, d) \neq 1$, there exists a smallest prime p' such that $p' | \#G$ and $p' | d$. Consider now the unique prime factor decomposition $\#G = \prod_{i=1}^{k} q_i^{a_i}$, where $q_1 < q_2 < \cdots < q_k$. Note that $p' = q_\ell$ for an integer $\ell \leq k$. By the structure of the Abelian groups, we have $G \simeq G(q_1) \oplus \cdots \oplus G(q_k)$, where $G(q_i)$ is the q_i-subgroup of G. For $i < \ell$, since $\gcd(d, q_i) = 1$, we have $dG(q_i) = G(q_i)$. This shows that the structure of dG is of the form

$$dG \simeq G(q_1) \oplus \cdots \oplus G(q_{\ell-1}) \oplus dG(q_\ell) \oplus \cdots \oplus dG(q_k)$$

and that $\#dG = \prod_{i=1}^{\ell-1} q_i^{a_i} \cdot \prod_{i=\ell}^{k} q_i^{b_i}$ for some integers b_i's satisfying $b_i \leq a_i$ for $i = \ell, \ldots, k$. Since dG is a subgroup of $K = \langle S_1 \rangle + dG$, we are ensured that $\#K = \prod_{i=1}^{\ell-1} q_i^{a_i} \cdot \prod_{i=\ell}^{k} q_i^{c_i}$ with $b_i \leq c_i \leq a_i$ for $i \geq \ell$. Thus, $\#G/\#K = \prod_{i=\ell}^{k} q_i^{a_i - c_i}$, which is either 1 or something greater or equal to p'. By the existence of f, we have $\#G/\#K < p \leq p'$, which implies that $G = K$. □

We make use of a pseudorandom generator GenM modeled by a random oracle. Let $\ell \in \mathbb{N}$ be a security parameter. This protocol called NIMGGDproof is depicted below.

NIMGGDproof$_\ell(S_1)$
Parameters: G, d.
Input: ℓ, $S_1 = \{g_1, \ldots, g_s\} \subseteq G$.

1. The prover picks $\mathsf{seedM} \in_U \{0,1\}^{k_m}$ uniformly at random and using the pseudorandom generator GenM produces some challenges $\mathsf{GenM}(G, \mathsf{seedM}) = (x_1, \ldots, x_\ell)$. Then, using his group expertise, he finds $r_i \in G$ and $a_{i,1}, \ldots, a_{i,s} \in \mathbb{Z}_d$ such that $x_i = dr_i + \sum_{j=1}^{s} a_{i,j} g_j$ for $i = 1, \ldots, \ell$. He sends seedM and the coefficients r_i's and $a_{i,j}$'s to the verifier.
2. Using GenM, the verifier generates $x_1, \ldots, x_\ell$ from seedM. He checks whether $x_i = dr_i + \sum_{j=1}^{s} a_{i,j} g_j$ for $i = 1, \ldots, \ell$. If this is the case, the verifier accepts the proof. Otherwise, he rejects it.

Note that the G occurrence in $\mathsf{GenM}(G, \mathsf{seedM})$ means that GenM must be fed with the description and representation of G. This is to make sure that it was chosen *before* generating the elements and avoid problems like in ECDSA [54].

Theorem 4.27. *Let* G *be an Abelian group, and* d *be an integer with smallest prime factor* p. *We assume that* GenM *is a random oracle. We consider the above* **NIMGGDproof** *protocol with parameters* G, d. *For provers who are group experts relative to* (d, S_1), **NIMGGDproof**(S_1) *is a NIZK for the language* $L_{\mathbf{MGGD}}(d, G)$. *More precisely, we have the following properties.*

1. **NIMGGDproof** *is sound*: *for any set* S_1 *such that* $\langle S_1 \rangle + dG \neq G$, *any cheating prover* $\mathcal{P}^*$ *limited to* q_{GenM} *queries to* GenM *has a success probability*

$$\mathsf{Succ}^{\text{sd-NIMGGD}}_{\mathcal{P}^*} \leq q_{\mathsf{GenM}} \cdot p^{-\ell} + (\#G)^{-\ell}.$$

2. **NIMGGDproof** *is perfect noninteractive black-box zero-knowledge in the random oracle model.*

Proof. Completeness is trivial.

Soundness We describe here a simulator $\mathcal{B}$ who uses $\mathcal{P}^*$ in order to win the following game. Without loss of generality, we assume that $\mathcal{P}^*$ does not submit the same query more than once.

> Game: A challenger picks $x_i \in_U G$ uniformly at random for $i = 1, \ldots, \ell$ and sends $x_1, \ldots, x_\ell$ to $\mathcal{B}$. The simulator wins if he is able to find coefficients r_i's and $a_{i,j}$'s such that $x_i = dr_i + \sum_{j=1}^{s} a_{i,j} g_j$ for $i = 1, \ldots, \ell$.

The simulator first receives $x = (x_1, \ldots, x_\ell)$ according to the above game and runs $\mathcal{P}^*$. $\mathcal{B}$ picks an integer $n \in_U \{1, \ldots, q_{\mathsf{GenM}}\}$ uniformly at random. The GenM queries made by $\mathcal{P}^*$ are simulated by maintaining a list of the queries and corresponding answers. Upon queries, the simulator outputs a uniformly random answer and adds the new pair in the list. However, we handle the nth query in a special way. Namely, we answer x to this query. Since x was picked uniformly at random, $\mathcal{B}$ simulates the oracle GenM perfectly. At the end, $\mathcal{P}^*$ outputs a seed seedM and coefficients r_i's and $a_{i,j}$'s. The simulator forwards the same coefficients to his challenger.

Let A be the event "$\mathcal{B}$ wins the game." By Lemma 4.26, $\Pr[A] \leq p^{-\ell}$. We also note that event A occurs only if $\mathcal{P}^*$ sends seedM as nth query made to GenM. Let B be the event "$\mathcal{P}^*$ queried seedM to GenM," and C be the event "$\mathcal{P}^*$ succeeds". We have

$$\Pr[A] = \frac{1}{q_{\mathsf{GenM}}} \cdot \Pr[B \wedge C].$$

Since GenM is a random oracle, $\Pr[\neg B \wedge C] \leq (\#G)^{-\ell}$, because the prover needs to guess x which corresponds to some seedM. Putting all together leads to

$$p^{-\ell} \geq \Pr[A] \geq \frac{1}{q_{\mathsf{GenM}}} \left(\mathsf{Succ}^{\text{sd-NIMGGD}}_{\mathcal{P}^*} - (\#G)^{-\ell}\right),$$

which concludes the proof.

Noninteractive Zero-Knowledge We describe a simulator $\mathcal{B}$ who simulates the message sent by an honest prover. Although the verifier $\mathcal{V}^*$ does not send any message here, the simulator needs to simulate the random oracle GenM which can be queried by $\mathcal{V}^*$. $\mathcal{B}$ picks $r_i \in_U G$, $a_{i,j} \in_U \mathbb{Z}_d$ and computes $x_i = dr_i + \sum_{j=1}^{s} a_{i,j} g_j$ for $i = 1, \ldots, \ell$, $j = 1, \ldots, s$. The simulator picks seedM $\in_U \{0,1\}^{k_m}$ uniformly at random and adds the pair (seedM, x), where $x = (x_1, \ldots, x_\ell)$ in a list maintained to simulate GenM. Then, the simulator can run $\mathcal{V}^*$ and sends him seedM and the coefficients r_i's, $a_{i,j}$'s. $\mathcal{B}$ simulates GenM as usual by maintaining a list of the previous queries and corresponding

answers. For any new query, the simulator simply picks the answer uniformly at random. The simulation is done perfectly since seedM was added in the list before the first query made by $\mathcal{V}^*$. □

4.7. *A 0-Move Proof for MGGD*

We can further reduce the proof by making sure that almost all instances belong to the language. A first idea consists in including the NIZK in the instance so that instances become self-proven. Another idea can be used in order to relax the group expert assumption which is needed in the **NIMGGDproof** protocol. For this, we provide a proof that the instance was randomly generated. The instance is generated by

$$\mathsf{GenK}(G, \mathsf{seedK}) = (g_1, \ldots, g_s).$$

The value seedK is added in the instance as a proof and actually replaces it as the instance is fully defined by seedK. We essentially use two different approaches. The first one consists of having instances long enough so that it is hard to generate a bad instance offline. The second one consists of having a TTP to select the random seed and an instance so that it is hard to generate a bad instance online. The latter approach requires using a certificate for proper generation of the seed.

We start by the following technical lemma.

Lemma 4.28. *Let A be a finite Abelian p-group such that $A \simeq \mathbb{Z}_{p^{e_1}} \oplus \mathbb{Z}_{p^{e_2}} \oplus \cdots \oplus \mathbb{Z}_{p^{e_k}}$, for some integers $0 < e_1 \le e_2 \le \cdots \le e_k$. Set $e = \sum_{i=1}^{k} e_i$. The number of maximal subgroups of A, i.e., of order p^{e-1} is equal to $(p^k - 1)/(p - 1)$.*

Proof. According to some results in combinatorial theory (see [37, p. 87] quoted in Butler [13]), the number of subgroups of A of order p^ℓ is equal to those of order $p^{e-\ell}$, for any integer $\ell < e$. Hence, our problem can be solved by enumerating all subgroups of A of order p. For this, we consider all elements of A of order p. Since any group $\mathbb{Z}_{p^{e_i}}$ contains exactly p elements of order p or which are 0 for $i = 1, \ldots, k$, we have $p^k - 1$ elements in A of order p (we just need to remove the neutral element). To conclude, it suffices to remark that any subgroup generated by an element of order p is in fact generated by $p - 1$ such elements. □

Theorem 4.29. *Let G, H be some Abelian groups, and d the order of H. The probability P_{gen} that some elements $g_1, \ldots, g_s \in_U G$ picked uniformly at random H-generate G satisfies*

$$P_{\mathsf{gen}} \ge \prod_{q \in \mathcal{P}_d} \left(1 - \frac{q^{k_q} - 1}{(q-1) \cdot q^s}\right),$$

where $\mathcal{P}_d$ is the set of all prime factors of $\gcd(\#G, d)$, and k_q is the rank of the maximal q-subgroup of G: Given a prime q, the q-subgroup of G is the subgroup A_q of elements whose orders are powers of q. The rank k_q is the integer such that there exists a unique sequence of integers $a_{q,1} \le \cdots \le a_{q,k_q}$ such that A_q is isomorphic to $\mathbb{Z}_{q^{a_{q,1}}} \oplus \cdots \oplus \mathbb{Z}_{q^{a_{q,k_q}}}$.

Proof. By the assertion 6 of Lemma 4.3, we need to study the probability to generate the quotient group $G/(d \cdot G)$ with some elements picked uniformly at random. Classical results on the structure of Abelian groups states the decomposition $G \simeq A_{p_1} \oplus \cdots \oplus A_{p_n}$. Note that

$$G/(d \cdot G) \simeq A_{p_1}/dA_{p_1} \oplus \cdots \oplus A_{p_n}/dA_{p_n}.$$

We consider $B_q = A_q/dA_q$ and study the probability that elements generate this group. If $\gcd(d, q) = 1$, then $dA_q = A_q$, and B_q is trivial. So, we only focus on the q's that divide d and denote e_q the largest integer such that $q^{e_q} | d$. We deduce that the structure of B_q satisfies

$$B_q \simeq \mathbb{Z}_{q^{a_{q,1}}} \oplus \cdots \oplus \mathbb{Z}_{q^{a_{q,r}}} \oplus \mathbb{Z}_{q^{e_q}} \oplus \cdots \oplus \mathbb{Z}_{q^{e_q}},$$

where r is the largest integer such that $a_{q,r} < e_q$. The probability P_q that s elements do not generate B_q is equal to the probability that these elements stay in one of the maximal subgroups of B_q. By Lemma 4.28, the number of such subgroups is equal to $(q^{k_q} - 1)/(q - 1)$. Therefore,

$$P_q \leq \frac{q^{k_q} - 1}{(q-1) \cdot q^s}.$$

Since these events are independent for the different B_q's, the final probability is obtained by multiplying the terms $1 - P_q$. □

Remark 4.30. As an application, if d is prime and if the d-subgroup of G is a product of k cyclic groups, we have $P_{\mathsf{gen}} \geq 1 - (d^k - 1)/(d-1) \cdot d^{-s}$. In practice, we will rarely have k greater than 2 so that we approximately have a probability of $1 - d^{-s+1}$.

5. MOVA Scheme

5.1. *Description*

We present our scheme called MOVA which is based on a secret group homomorphism. This scheme was first proposed at ASIACRYPT '04 [41] and was inspired by a preliminary version restricted to group characters (with a less efficient denial protocol) presented at PKC '04 [42].

Since MOVA is generic, the precise setup algorithms are not specified. In addition to this, several variants are proposed depending on whether key registration is used and on whether a group expertise can be used. Depending on the situation, one will be preferred on the others. Concrete instances are discussed in Sect. 5.3.

Domain Parameters. To formally comply with asymptotic definitions of security properties, the above parameters can be seen as polynomial functions of one single global security parameter k. We let integers Lkey, Lsig, Lcon, Lden be polynomial security parameters as well as "group ensembles" for Xgroup and Ygroup. The group ensembles should define groups, representation of elements with polynomial length, and polynomial-time addition, inversion, and comparison algorithms. An optional parameter $\mathsf{lval} \in \mathbb{N}$ is used for variant using **NIMGGDproof**.

Primitives. We use two deterministic pseudorandom generators GenK and GenS (modeled by some random oracles) which produce elements of Xgroup. We consider a trapdoor one-way permutation TPOW. The associated key pair $(K_{\mathsf{p}}^{\mathcal{V}}, K_{\mathsf{s}}^{\mathcal{V}})$ is that of the verifier. We also use additional random oracles specified in protocols given in Sect. 4.

Secret Key. $K_{\mathsf{s}}^{\mathcal{S}} = \mathsf{Hom}$ is a group homomorphism from Xgroup to Ygroup.

Public Key. $K_{\mathsf{p}}^{\mathcal{S}} = (\mathsf{Xgroup}, \mathsf{Ygroup}, \mathsf{seedK}, (\mathsf{Ykey}_1, \ldots, \mathsf{Ykey}_{\mathsf{Lkey}}), \mathsf{opt})$, where opt is an optional string which can be used to check the $K_{\mathsf{p}}^{\mathcal{S}}$ validity in the setup variants. We always implicitly assume that the participants check this string before using a public key. We use $(\mathsf{Xkey}_1, \ldots, \mathsf{Xkey}_{\mathsf{Lkey}}) = \mathsf{GenK}(\mathsf{Xgroup}, \mathsf{seedK})$ and $\mathsf{Ykey}_j = \mathsf{Hom}(\mathsf{Xkey}_j)$ for $j = 1, \ldots, \mathsf{Lkey}$.

Validate. The first two options consist of using the 0MGGDproof verification algorithm. Either it is based on a long key (so the public key only includes a seed), or it is based on a TTP-certified seed (in which case the public key also includes a certificate for the seed). A third option consists of verifying the NIZK as specified in the $\mathsf{NIMGGDproof}_{\mathsf{lval}}$ protocol which must be included in the public key.

Signature Generation. Let m be a message to sign. The signer generates

$$\mathsf{GenS}(m) = (\mathsf{Xsig}_1, \ldots, \mathsf{Xsig}_{\mathsf{Lsig}}).$$

He then computes $\mathsf{Ysig}_k = \mathsf{Hom}(\mathsf{Xsig}_k)$ for $k = 1, \ldots, \mathsf{Lsig}$. The signature is

$$\sigma = (\mathsf{Ysig}_1, \ldots, \mathsf{Ysig}_{\mathsf{Lsig}}).$$

It is $\mathsf{Lsig} \cdot \log_2 d$ bits long.

Confirmation Protocol. Let (m, σ) be a supposedly valid message–signature pair. The verifier first checks that the public key is valid using Validate. Both the signer and the verifier (signature's recipient) compute the elements $\mathsf{Xkey}_1, \ldots, \mathsf{Xkey}_{\mathsf{Lkey}}$ from the signer's public key. They also generate $\mathsf{GenS}(m) = (\mathsf{Xsig}_1, \ldots, \mathsf{Xsig}_{\mathsf{Lsig}})$. The signer playing the role of the prover runs $\mathbf{2GHIproof}_{\mathsf{Icon}}$ with the verifier on the set

$$N = \left\{(\mathsf{Xkey}_j, \mathsf{Ykey}_j) \mid j = 1, \ldots, \mathsf{Lkey}\right\} \cup \left\{(\mathsf{Xsig}_k, \mathsf{Ysig}_k) \mid k = 1, \ldots, \mathsf{Lsig}\right\}.$$

Denial Protocol. Let (m, σ') be an alleged invalid message–signature pair. We denote $\sigma' = (\mathsf{Zsig}_1, \ldots, \mathsf{Zsig}_{\mathsf{Lsig}})$. The verifier first checks that the public key is valid. The signer and the verifier compute $\mathsf{Xkey}_1, \ldots, \mathsf{Xkey}_{\mathsf{Lkey}}$ from the public key and $\mathsf{GenS}(m) \to \mathsf{Xsig}_1, \ldots, \mathsf{Xsig}_{\mathsf{Lsig}}$. The signer playing the role of the prover runs $\mathbf{2coGHIproof}_{\mathsf{Iden}}$ with the verifier on the sets

$$S = \left\{(\mathsf{Xkey}_j, \mathsf{Ykey}_j) \mid j = 1, \ldots, \mathsf{Lkey}\right\} \quad \text{and}$$

$$T = \left\{(\mathsf{Xsig}_k, \mathsf{Zsig}_k) \mid k = 1, \ldots, \mathsf{Lsig}\right\}.$$

The protocol options are threefold. First, we can choose the 4-Move or the 2-Move option for the GHI or coGHI proofs. One difference is that the 2-Move option requires random oracles instead of a trapdoor commitment. In addition to this, the 4-Move option requires a group expert algorithm to exist (although it shall not necessarily be known by the signer). Since the signature itself requires random oracles we found no advantage

in using the 4-Move option, and so we concentrate on the 2-Move one. Second, we can choose to have nontransferability (in which case we need keys for the verifier) or not. If nontransferability is not a concern, we can get rid of the key pair for the verifier and no longer use a trapdoor one-way permutation. Finally, there are three options for the validate algorithm. We can use the **0MGGDproof** using either a long key or a TTP-certified seed. Another choice, when group experts exist, consists of expanding the public key with a NIZK (the NIMGGDproof protocol). The selection of the NT versus non-NT and the Validate option span into six possible settings:

Option	**0MGGDproof**	**0MGGDproof**	**NIMGGDproof**	Comment
Non-NT	×	×	×	No key for $\mathcal{V}$
NT	×	×	×	Uses TPOW
Comment	Lkey long	TTP	Group expert	

5.2. *Security Results*

We first prove that the two-move version of the MOVA scheme satisfies the security properties of undeniable signature schemes. The proofs of resistance against forgery attacks and invisibility were inspired from Kurosawa and Heng [33].

Theorem 5.1. *Let e denote the natural logarithm base. The MOVA scheme satisfies the following security properties in the random oracle model.*

1. Confirm *and* Deny *are nontransferable zero-knowledge proof of membership.*
2. *Consider the* Lsig-S-GHI *problem with the same parameters as for the* MOVA *scheme, i.e.,* $G = \mathsf{Xgroup}$, $H = \mathsf{Ygroup}$, *and the set* $S = \{(\mathsf{Xkey}_1, \mathsf{Ykey}_1), \ldots, (\mathsf{Xkey}_{\mathsf{Lkey}}, \mathsf{Ykey}_{\mathsf{Lkey}})\}$. *Assume that for any algorithm* $\mathcal{B}$ *with a given complexity, we have* $\mathsf{Succ}_{\mathcal{B}}^{\text{Lsig-}S\text{-GHI}} \leq \varepsilon$. *Then, any forger* $\mathcal{F}$ *with similar complexity using* q_S *signing queries and* q_V *queries to the confirmation/denial oracle wins the forgery game under a chosen-message attack with probability at most* $\varepsilon e(1+q_S)(1+q_V)$.
3. *Consider the* Lsig-S-GHID *problem with the same parameters. Assume that for any algorithm* $\mathcal{B}$ *and* $\mathcal{B}'$ *with a given complexity, we have*

$$\mathsf{Adv}_{\mathcal{B}}^{\text{Lsig-}S\text{-GHID}} \leq \varepsilon \quad \textit{and} \quad \mathsf{Succ}_{\mathcal{B}'}^{\text{Lsig-}S\text{-GHI}} \leq \varepsilon'.$$

Then, any distinguisher $\mathcal{D}$ *with similar complexity using* q_S *signing queries and* q_V *queries to the confirmation/denial oracle wins the invisibility game under a chosen-message attack with advantage*

$$\mathsf{Adv}_{\mathcal{D}}^{\text{inv-cma}} \leq e(1+q_S)\big(\varepsilon + 2(1+q_V)\varepsilon'\big).$$

Proof. The properties of Confirm and Deny come from Theorems in Sects. 4.4–4.5.

Unforgeability Let $\mathcal{F}$ be a forger who succeeds to existentially forge a signature under an adaptive chosen-message attack with probability ε'. We construct an algorithm $\mathcal{B}$

using the forger $\mathcal{F}$ and $K_s^{\mathcal{V}}$. At the beginning, $\mathcal{B}$ receives the challenges $x_1, \ldots, x_{\mathsf{Lsig}} \in \mathsf{Xgroup}$ of the Lsig-S-GHI problem. Then, $\mathcal{B}$ runs the forger and simulates the queries to the random oracle GenS, q_S queries to the signing oracle Sign and q_V queries to the denial/confirmation oracle Ver. We can assume that all messages sent to Sign resp. Ver was previously queried to GenS (since the oracle Sign resp. Ver has to make such queries anyway). $\mathcal{B}$ simulates the oracles GenS and Sign as follows:

GenS. $\mathcal{B}$ maintains a list of the messages queried to GenS and corresponding answer. If the message was already queried, $\mathcal{B}$ outputs the corresponding answer in the list. Otherwise, he picks $a_{i,j} \in_U \mathbb{Z}_d$ and $r_i \in_U \mathsf{Xgroup}$ uniformly at random for $1 \le i \le \mathsf{Lsig}$, $1 \le j \le \mathsf{Lkey}$. With probability q, he answers $\mathsf{Xsig}_i = dr_i + \sum_{j=1}^{\mathsf{Lkey}} a_{i,j}\mathsf{Xkey}_j$ for $i = 1, \ldots, \mathsf{Lsig}$. We call it type-1 answer. With probability $1-q$, the answer is $\mathsf{Xsig}_i = dr_i + x_i + \sum_{j=1}^{\mathsf{Lkey}} a_{i,j}\mathsf{Xkey}_j$ for $i = 1, \ldots, \mathsf{Lsig}$. We call it type-2 answer. For each message, $\mathcal{B}$ keeps the coefficients $a_{i,j}$'s and r_i's and answer type in memory. Note that the simulation is perfect by Lemma 4.16, since the public key is valid.

Sign. For a message m, if the answer to the GenS query of m was of type-1, then $\mathcal{B}$ answers $\mathsf{Ysig}_i = \sum_{j=1}^{\mathsf{Lkey}} a_{i,j}\mathsf{Ykey}_j$ for $i = 1, \ldots, \mathsf{Lsig}$. Otherwise, it aborts the simulation.

Let (m_i, σ_i) denote the ith query to Ver for $1 \le i \le q_V$, and $(m_{q_V+1}, \sigma_{q_V+1})$ denote the $\mathcal{F}$ output. In order to simulate the answers of the queries made to Ver, $\mathcal{B}$ guesses the smallest i such that (m_i, σ_i) is a valid forged pair (i.e., m was not queried to Sign). For this, $\mathcal{B}$ simply picks ℓ uniformly at random in $\{1, \ldots, q_V + 1\}$. $\mathcal{B}$ deals with the ith query as follows:

$i < \ell$. To any query (m_i, σ_i), $\mathcal{B}$ checks whether m_i was submitted to Sign. If it is the case, $\mathcal{B}$ is able to decide whether (m_i, σ_i) is valid and simulates the appropriate protocol. Otherwise, $\mathcal{B}$ guesses that (m_i, σ_i) is invalid and simulates the appropriate protocol. The simulation is done as the simulator in the proof of nontransferability of the confirmation (resp. denial) protocol.

$i = \ell$. Let $(m_\ell, \sigma_\ell) = (m_\ell, \mathsf{Ysig}_1, \ldots, \mathsf{Ysig}_{\mathsf{Lsig}})$. If the corresponding Xsig_i's were of type-1, $\mathcal{B}$ aborts. Otherwise, when ℓ was correctly guessed, $\mathsf{Ysig}_i = y_i + \sum_{j=1}^{\mathsf{Lkey}} a_{i,j}\mathsf{Ykey}_j$, and $\mathcal{B}$ is able to deduce the y_i's of the Lsig-S-GHI problem.

It remains to compute the probability that $\mathcal{B}$ retrieves the y_i's and did not abort. This event occurs if $\mathcal{B}$ is able to simulate all Sign queries, guess the right ℓ, and use the message m_ℓ to deduce the y_i's. Therefore, $\Pr[\mathcal{B}\text{ succeeds}|\mathcal{F}\text{ succeeds}] = q^{q_S}(1-q)/(q_V+1)$. As for the full-domain hash technique [19] and as in [33], the optimal $q_{\mathsf{opt}} = q_S/(q_S+1)$, and so $\Pr[\mathcal{B}\text{succeeds}|\mathcal{F}\text{ succeeds}] \ge \frac{1}{e(1+q_S)(1+q_V)}$. Thus, $\varepsilon' \le \varepsilon e(1+q_S)(1+q_V)$.

Invisibility Let $\mathcal{D}$ be a distinguisher which breaks the invisibility of the MOVA scheme with an advantage ε. We construct an algorithm $\mathcal{B}$ which solves the Lsig-S-GHID problem by using $\mathcal{D}$ and $K_s^{\mathcal{V}}$. At the beginning, $\mathcal{B}$ is challenged with a tuple $\{(x_1, y_1), \ldots, (x_{\mathsf{Lsig}}, y_{\mathsf{Lsig}})\} \in (\mathsf{Xgroup} \times \mathsf{Ygroup})^{\mathsf{Lsig}}$ for which it has to decide whether $\mathsf{Hom}(x_i) = y_i$ for all $1 \le i \le \mathsf{Lsig}$ or if this tuple was picked at random. Like for the proof of the existential forgery, the simulator $\mathcal{B}$ runs $\mathcal{D}$ and simulates the queries to the

random oracle GenS, q_S queries to the signing oracle Sign, and the queries to the denial/confirmation oracle Ver. We can assume that each message queried to Sign or Ver was previously queried to the random oracle GenS. We assume that no query m to Ver was submitted to Sign beforehand. (Otherwise, we can just simulate them with $K_{\mathsf{s}}^{\mathcal{V}}$.) Let Forge be the event in which $\mathcal{D}$ sends a valid message–signature pair to Ver. We first remove all instances for which the event Forge occurs. So, we can now assume that $\mathcal{D}$ never submits any valid pair (m, σ) to Ver such that m was not previously submitted to Sign. $\mathcal{B}$ simulates the oracles just like in the proof of unforgeability with $\ell = q_V + 1$ (we excluded valid forged pairs).

After a given time, the distinguisher $\mathcal{D}$ sends a message m^* to the challenger of the invisibility game which is simulated by $\mathcal{B}$. We can assume that m^* was queried to GenS (otherwise $\mathcal{B}$ simulates a new query). If the answer of m^* to GenS was of type-1, $\mathcal{B}$ aborts the simulation. Otherwise, it sends the challenge signature $(\mathsf{Ysig}_1^*, \ldots, \mathsf{Ysig}_{\mathsf{Lsig}}^*)$, where $\mathsf{Ysig}_i^* = y_i + \sum_{j=1}^{\mathsf{Lkey}} a_{i,j} \mathsf{Ykey}_j$ for $1 \le i \le \mathsf{Lsig}$. Then, $\mathcal{D}$ continues to query the oracles which are simulated by $\mathcal{B}$ as above.

Finally, $\mathcal{D}$ outputs a guess bit b'. The simulator $\mathcal{B}$ outputs the same bit b' as guess bit to the Lsig-S-GHID challenger or a random bit when $\mathcal{B}$ aborted.

Using the homomorphic property of Hom, we deduce that the set $\{(x_i, y_i)\}_{i=1}^{\mathsf{Lsig}}$ interpolates in a group homomorphism with the set of points S if and only if $(m^*, \mathsf{Ysig}_1^*, \ldots, \mathsf{Ysig}_{\mathsf{Lsig}}^*)$ is a valid message–signature pair. Hence, when the simulator does not abort and the event Forge does not occur, $\mathcal{B}$ perfectly simulates the invisibility games. It remains to compute the advantage of $\mathcal{B}$.

For a bit b, we denote A_b the event that $\mathcal{B}$ does not abort when the challenge to $\mathcal{B}$ was of the form T_b (thus, $\mathcal{B}$ simulates the game $\mathbf{Game}^{\mathsf{inv\text{-}cma\text{-}}b}$ to $\mathcal{D}$). Note that the probability $\Pr[A_1] = \Pr[A_0]$ can be bounded in an optimal way as in the proof of existential forgery attacks, namely, by choosing q adequately we get $\Pr[A_1] \ge (1/e(1 + q_S))$. We now define the events B_b and D_b which occur when $\mathcal{B}$ and $\mathcal{D}$ respectively output the bit 0 when the challenge was of the form T_b. Note that if A_b happens, both events B_b and D_b occur simultaneously. Let us denote ε_0 resp. ε_1 the probability for $\mathcal{D}$ to output 0 in the game $\mathbf{Game}^{\mathsf{inv\text{-}cma\text{-}0}}$ resp. $\mathbf{Game}^{\mathsf{inv\text{-}cma\text{-}1}}$. We now estimate $\Pr[B_0|A_0]$ and $\Pr[B_1|A_1]$ with respect to ε_0 and ε_1. To this end, we notice that the event $B_0|A_0$ resp. $B_1|A_1$ occurs simultaneously with the event where $\mathcal{D}$ outputs 0 in the game $\mathbf{Game}^{\mathsf{inv\text{-}cma\text{-}0}}$ resp. $\mathbf{Game}^{\mathsf{inv\text{-}cma\text{-}1}}$, provided that the event Forge does not occur. Hence, applying the difference lemma of Shoup [53] leads to

$$|\Pr[B_b|A_b] - \varepsilon_b| \le \Pr[\mathsf{Forge}]$$

for $b = 0, 1$. From this we can deduce that $\Pr[B_0|A_0] \ge \varepsilon_0 - \Pr[\mathsf{Forge}]$ and $\Pr[B_1|A_1] \le \varepsilon_1 + \Pr[\mathsf{Forge}]$. Without loss of generality, we can assume that $\Pr[B_0] \ge \Pr[B_1]$. The advantage of $\mathcal{B}$ is then equal to

$$\begin{aligned}\Pr[B_0] - \Pr[B_1] = \Pr[\neg A_0] \cdot \big(\Pr[B_0|\neg A_0] - \Pr[B_1|\neg A_1]\big) \\ + \Pr[A_0] \cdot \big(\Pr[B_0|A_0] - \Pr[B_1|A_1]\big).\end{aligned}$$

Since $\Pr[B_0|\neg A_0] = \Pr[B_1|\neg A_1] = 1/2$ and $\varepsilon_0 - \varepsilon_1 = \mathsf{Adv}_{\mathcal{D}}^{\mathsf{inv-cma}}$, we finally have

$$\mathsf{Adv}_{\mathcal{B}}^{\mathsf{Lsig}\text{-}S\text{-}\mathsf{GHID}} \geq \frac{1}{(1+q_S)e}\left(\mathsf{Adv}_{\mathcal{D}}^{\mathsf{inv-cma}} - 2\Pr[\mathsf{Forge}]\right).$$

We can conclude by noting that Forge occurs with a probability bounded by $e(1+q_S)(1+q_V)\varepsilon'$ by assertion 2. □

Remark 5.2. Similarly to Laguillaumie and Vergnaud [34], the efficiency of the security reduction for the existential forgery can be improved (factor $(1+q_V)^{-1}$ is removed) by replacing the GHI problem by its *gap* variant [46]. This problem consists in solving the GHI problem using an access to an oracle which solves the corresponding GHID problem. This one helps to simulate the confirmation and denial oracles. So, we do not need to guess $\ell \in \{1, \ldots, q_V + 1\}$ to simulate these oracles correctly.

Remark 5.3. MOVA scheme can be made probabilistic so that the invisibility notion defined in Galbraith and Mao [23] is satisfied. To this, it suffices to append some randomness r to the message to sign and to add r in the signature. The drawback is that the signature enlarges.

5.3. *Parameters, Implementation, and Other Properties*

Security results on the unforgeability and the invisibility of MOVA given in Theorem 5.1 allow us to directly derive some bounds on the signature size provided certain assumptions on the GHI and GHID problems. We point out that the hardness of solving GHI and GHID problems can often be scaled by only adjusting the size of Xgroup.

To illustrate this, consider $\mathsf{Xgroup} = \mathbb{Z}_n^*$ with $n = pq$ for two large primes p, q and the Legendre symbol $(\cdot/p)$. Solving the corresponding GHI problem requires to solve the quadratic residuosity assumption for which the best known solver algorithm consists in factoring n.

As shown by the above example, the MOVA scheme can be instantiated so that the signature size is fully scalable depending on the required security. In certain cases, we can also select Xgroup such that the hardness of GHI is adjusted to the computational power of the adversary without any impact on the signature size. Namely, we assume here that Xgroup is adjusted such that

$$\mathsf{Succ}_{\mathcal{B}}^{\mathsf{Lsig}\text{-}S\text{-}\mathsf{GHI}} \approx d^{-\mathsf{Lsig}} \quad \text{and} \quad \mathsf{Adv}_{\mathcal{B}}^{\mathsf{Lsig}\text{-}S\text{-}\mathsf{GHID}} \approx 0$$

for all algorithms $\mathcal{B}$ with similar complexity as the adversary. Using Theorem 5.1, this leads to

$$\mathsf{Succ}_{\mathcal{F}}^{\mathsf{ef\text{-}cma}} \approx eq_Sq_Vd^{-\mathsf{Lsig}} \quad \text{and} \quad \mathsf{Adv}_{\mathcal{D}}^{\mathsf{inv\text{-}cma}} \approx 2eq_Sq_Vd^{-\mathsf{Lsig}}.$$

Since the verification of an undeniable signature must be done online, we can consider some security probabilities of about 2^{-20} instead of the classical offline security of 2^{-80}. In Table 1, we give the required MOVA signature size in order to achieve $\mathsf{Succ}_{\mathcal{F}}^{\mathsf{ef\text{-}cma}} \approx 2^{-20}$ depending on q_S and q_V. We can get the same (except 1 bit due to the factor 2) results for the invisibility.

Table 1. Signature size with unforgeability and invisibility of 2^{-20}.

q_S	q_V	$\mathsf{Lsig} \cdot \log_2(d)$ bits
2^{10}	2^{10}	42
2^{10}	2^{20}	52
2^{20}	2^{10}	52
2^{20}	2^{20}	62

As an example, an application involving $q_S = 2^{10}$ signatures and up to $q_V = 2^{20}$ online verifications per key can tolerate a size of $\mathsf{Lsig} = 52$ bits.

Results for the soundness of the two-move confirmation and denial protocols can be obtained using Theorem 4.24 and 4.25. Under the assumption $\mathsf{Succ}^{\mathsf{inv\text{-}tp}} \approx 0$, we obtain

$$\mathsf{Succ}^{\mathsf{sd\text{-}con}}_{\mathcal{S}^*} \approx q_{\mathsf{Gen}'} p^{-\mathsf{Icon}} \quad \text{and} \quad \mathsf{Succ}^{\mathsf{sd\text{-}den}}_{\mathcal{S}^*} \approx q_{\mathsf{Gen}'} p^{-\mathsf{Iden}}.$$

For instance, we can achieve a soundness probability of 2^{-20} with the parameters $\mathsf{Icon} = \mathsf{Iden} = 60/\log_2(p)$, $q_{\mathsf{Gen}'} = 2^{40}$. Similarly, for the four-move protocols, we assume that Commit satisfies

$$\mathsf{Succ}^{\mathsf{com\text{-}bnd}}_{\mathcal{B}} \approx 0$$

for any algorithm $\mathcal{B}$ with similar complexity as the adversary. This shows that

$$\mathsf{Succ}^{\mathsf{sd\text{-}con}}_{\mathcal{S}^*} \approx p^{-\mathsf{Icon}} \quad \text{and} \quad \mathsf{Succ}^{\mathsf{sd\text{-}den}}_{\mathcal{S}^*} \approx p^{-\mathsf{Iden}},$$

which leads to some smaller parameters Icon and Iden. Namely, for a soundness probability of 2^{-20}, we get $\mathsf{Icon} = \mathsf{Iden} = 20/\log_2(p)$.

We examine here the size of parameters implied by the different setup variants according to their specificity. First, we note that security of Setup Variant using 0-Move proof for MGGD is directly deduced from the results of Theorem 4.29. The main difference is simply due to the number of attempts the adversary can perform until he gets some "bad" elements $\mathsf{Xkey}_1, \ldots, \mathsf{Xkey}_{\mathsf{Lkey}}$. In the first variant, the signer can try as many attempts as he can so that we require an "offline" probability $P_{\mathsf{gen}} \geq 1 - 2^{-80}$, while in the second one he is very limited so that we require an "online" probability $P_{\mathsf{gen}} \geq 1 - 2^{-20}$. Assuming similar assumptions as in Remark 4.30, we get $\mathsf{Lkey} = 81/\log_2(d)$ for 0-Move with long key variant and $\mathsf{Lkey} = 21/\log_2(d)$ for 0-Move with TTP variant. This means that the tuple $(\mathsf{Ykey}_1, \ldots, \mathsf{Ykey}_{\mathsf{Lkey}})$ which is contained in the public key would be 81- and 21-bits long, respectively. As for the soundness of the confirmation and denial protocols, we get a probability that the signer passes the protocol with an invalid public key of 2^{-20} with $\mathsf{Ival} = 20/\log_2(p)$ assuming that no efficient adversary breaks the computationally binding property of Commit. Finally, Theorem 4.27 gives results for Setup variant with NIZK for MGGD. Since Xgroup is usually greater than p, we have

$$\mathsf{Succ}^{\mathsf{sd\text{-}NIMGGD}}_{\mathcal{P}^*} \approx q_{\mathsf{GenM}} \cdot p^{-\mathsf{Ival}},$$

which leads to a probability of 2^{-20} with $q_{\mathsf{GenM}} = 2^{60}$ and $\mathsf{Ival} = 80/\log_2(p)$.

5.4. *Potential Instantiations*

We briefly discuss some potential instantiations of the homomorphism which allow one to consider short signatures. These ones must have a short group range.

Characters Let $n = pq$ be an RSA modulus. A character on $\mathbb{Z}_n^*$ is a homomorphism from $\mathbb{Z}_n^*$ to $\mathbb{C}\backslash\{0\}$. We consider some characters χ of order $d = 2, 3, 4$, i.e., $\chi^d(x) = 1$ for any $x \in \mathbb{Z}_n^*$. For $d = 2$, the nontrivial characters which are assumed hard to compute are both Legendre (Jacobi) symbols $(\cdot/p)$ and $(\cdot/q)$. For $d = 3$ or 4, the characters correspond to some natural generalization of the Legendre symbols arising in the theory of cubic and quartic residuosity, respectively. The cubic characters require the use of Eisenstein integers, i.e., elements of the form $a + b\omega$ for $a, b \in \mathbb{Z}$ and $\omega = (-1 + \sqrt{-3})/2$. We require that $p \equiv q \equiv 1 \pmod 3$ and take an Eisenstein integer π such that $\pi\bar{\pi} = p$. As a candidate homomorphism, we can take χ_π defined by $\chi_\pi(x) = x^{(p-1)/3} \pmod \pi$ for $x \in \mathbb{Z}_n^*$. Similarly, characters of order 4 arises in the set of Gauss integers $\mathbb{Z}[i] = \{a + bi | a, b \in \mathbb{Z}\}$. Choose $p \equiv q \equiv 1 \pmod 4$ and π such that $\pi\bar{\pi} = p$. Then, consider χ_π defined as $\chi_\pi(x) = x^{(p-1)/4} \pmod \pi$ for $x \in \mathbb{Z}_n^*$.

The cubic and quartic characters can be efficiently (about quadratic time) computed using reciprocity laws as for the quadratic residue. We also note that group expertise for the above characters correspond to the factorization of n. A more detailed treatment on the character instantiations can be found in [40,42].

Example 5.4 (QR parameters). As a concrete example, we propose to adopt the Legendre symbol $\mathsf{Hom}(x) = (x/p)$ and use $\mathsf{Lkey} = 21$, $\mathsf{Lsig} = 52$, $\mathsf{Icon} = \mathsf{Iden} = 60$, $\mathsf{Ival} = 80$, and the 2-Move variant with **NIMGGDproof**. Under the assumption that $q_S < 10^3$, $q_V < 10^6$, and $q_{\mathsf{Gen}'} < 10^{12}$, we have an online security of 2^{20} and an offline security of 2^{80}.

Example 5.5 (Newton parameters). Let n be such that $n = pq$ with $p = rd + 1$, q, d prime, $\gcd(q - 1, d) = 1$, $\gcd(r, d) = 1$, and g generating a subgroup of $\mathbb{Z}_p^*$. We obtain g by choosing a random element $h \in \mathbb{Z}_n^*$ until h satisfies $h^r \bmod p \neq 1$, and we set $g = h^r \bmod p$. Like this, we find a homomorphism by "sending" the input in a hidden cyclic subgroup of order d and then computing its discrete logarithm with respect to the generator g,

$$\begin{aligned} \varphi : \mathbb{Z}_n^* &\longrightarrow \mathbb{Z}_d \\ x &\longmapsto \log_g(x^r \bmod p). \end{aligned}$$

The expert group knowledge is obtained if one knows p and q. Typically, we could use $d = 2^{20} - 3 = 1\,048\,573$, which is prime, $\mathsf{Lkey} = 1$, $\mathsf{Lsig} = 3$, $\mathsf{Icon} = \mathsf{Iden} = 3$, $\mathsf{Ival} = 4$, and the 2-Move variant with **NIMGGDproof**. Under the assumption that $q_S < 10^3$, $q_V < 10^6$, and $q_{\mathsf{Gen}'} < 10^{12}$, we have an online security of 2^{20} and an offline security of 2^{80}.

5.5. *Implementation of the Signature Generation*

Here, we finally compare the time required for generating a MOVA signature with different homomorphisms. We consider a signature size of $\mathsf{Lsig} = 20$ bits (except for RSA).

Table 2. Signature generation.

Homomorphism	$\mathsf{Lsig} = 20$ Time in ms	$\mathsf{Lsig} = 52$ Time in ms
Quartic Residue Symbol (χ_π)	90.32	234.83
Jacobi Symbol (ordinary algorithm)	25.22	65.57
Jacobi Symbol (`mpz_jacobi`)	2.32	6.03
Discrete Logarithm (Precomputed Table)	9.66	n/a
Discrete Logarithm (Baby-Step Giant-Step)	19.47	31.39
Discrete Logarithm (Pollard's rho)	74.93	120.82
RSA	33.87	33.87

We omit the time required by the generation of the values Xsig_i's. Hence, we compare the time required for computing 20 Jacobi symbols $(\cdot/p)_2$, 10 quartic residue symbols χ_π, one homomorphism based on the discrete logarithm in a hidden subgroup and one RSA homomorphism. We recall that for all these homomorphisms, we take a modulus n of size of 1024 bits.

The implementation of all algorithms has been written in C using the GNU Multiple Precision Arithmetic Library (GMP) [25] and was done by Yvonne Anne Oswald in 2005 [47]. The tests have been done on an Intel(R)4 1.4 GHz Desktop Computer with 256 MB RAM. Results are given in Table 2. To scale these numbers with $\mathsf{Lsig} = 52$, the figures for symbols should be multiplied by $\frac{52}{20}$, the ones for baby-step/giant-step and Pollard discrete logarithm should be multiplied by $\sqrt{\frac{52}{20}}$, the one for RSA should be kept as is (the signature is not short), and we should not consider precomputed tables for discrete logarithm any more. Results are also reported.

We have implemented the Jacobi symbol using basic GMP subroutines in order to have a fair comparison with our implementation of the quartic residue symbol. We note that the highly optimized GMP implementation of the Jacobi symbol `mpz_jacobi` provides the fastest signature generation and that the quartic residue symbol χ_π is about four times slower than our implementation of the Jacobi symbol. This is mainly due to the fact that all operations are performed in $\mathbb{Z}[i]$ instead of $\mathbb{Z}$. Due to the nature of Eisenstein integers, a similar result for cubic characters is very likely. The variants of the discrete logarithm offer a very competitive homomorphism. In particular, except for the variant using the Pollard rho method, this homomorphism is more efficient than an RSA ordinary signature. In particular, the variant with the precomputed table is three times faster than an RSA signature.

Note that these results directly apply to the confirmation protocol since the number of homomorphism evaluations the prover needs to perform is proportional (except for RSA which does not provide small signature) to that required for the signature generation.

More details about the optimization of the above homomorphisms are given in [44].

5.6. *Other Properties*

We point out that our scheme allows a *batch verification* of signatures. Indeed, the confirmation protocol can be easily adapted in order to confirm several signatures at the same time by putting all $(\mathsf{Xsig}_k, \mathsf{Ysig}_k)$ in a single set S. The properties of the

2GHIproof protocol are such that the communication complexity remains the same, whereas the computation complexity is linear in the number of the signatures. To isolate a small set of m incorrect signatures in a big set of n signatures, we can use a cut-and-choose algorithm and do it with $\mathcal{O}(m \log n)$ iterations.

Note that the signer with group expertise can selectively convert an undeniable signature into a classical one by finding the coefficients $a_{i,k} \in \mathbb{Z}_d$ and $r_i \in \mathsf{Xgroup}$ such that

$$\mathsf{Xsig}_i = dr_i + \sum_{k=1}^{\mathsf{Lkey}} a_{i,k}\mathsf{Xkey}_k \quad \text{for } i = 1, \ldots, \mathsf{Lsig}.$$

The conversion consists of revealing these coefficients. For verifying a signature, it suffices to check the above equations and verify that $\mathsf{Ysig}_i = \sum_{k=1}^{\mathsf{Lkey}} a_{i,k}\mathsf{Ykey}_k$ for $i = 1, \ldots, \mathsf{Lsig}$.

There are some cases where computing the group homomorphism does not imply group expertise but some group expertise may exist. For instance, take $p \equiv q \equiv 1 \pmod 4$ and $\pi \in \mathbb{Z}[i]$ such that $\pi\bar{\pi} = n$. The quartic residue symbol $\chi_\pi(x)$ can be evaluated using π and x only. It provides a single hard-to-compute bit since $(\chi_\pi(x))^2 = (x/n)$ is the Jacobi symbol, which is easy to compute from x and n only. Here, the value of π does not leak the factors of n so the 4-G-Root problem remains hard. We can thus obtain two levels of secrets: a first level, which is enough to compute signatures and participate to verification protocols, and another level, which can convert signatures and make NIZK proofs for public keys.

6. Conclusion

We have proposed a generic scheme called MOVA based on group homomorphisms which generalizes the Chaum's undeniable signature scheme. For this, we developed a general framework based on the interpolation of group homomorphisms. Several well-known problems such as Diffie–Hellman and discrete logarithm can be easily expressed in this setting. By considering group homomorphisms with a small range group and scaling the domain group with respect to the adversary's complexity, we can naturally achieve very short signatures. As far as we know, this is the first signature scheme for which signatures of less than 80 bits can be considered. As further results, our two-move confirmation and denial protocols reach the minimal number of moves for interactive verification protocols. Possible concrete instantiations of MOVA scheme have been studied, and implementations showed that Legendre symbol offers the most efficient signature generation.

As open issues for further research, we are wondering if the interpolation of group homomorphisms can contribute to the design of other cryptographic primitives or give new theoretical insights in public-key cryptography. We also point out that designing a similar scheme (with short signatures) without using random oracles is still an open problem.

Acknowledgements

Most of this work was done when the first author was at EPFL or UCSD and supported by Swiss National Foundation with grants PBEL2-116915, 200021-101453/1, and 200020-109133.

References

[1] R.J. Anderson, S. Vaudenay, B. Preneel, K. Nyberg, The Newton channel, in *Information Hiding: 1st International Workshop*. Lecture Notes in Computer Science, vol. 1174 (Springer, Berlin, 1996), pp. 151–156

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, Proof verification and hardness of approximation problems, in *33rd Annual IEEE Symposium on Foundations of Computer Science, FOCS '92* (IEEE Computer Society, Los Alamitos, 1992), pp. 14–23

[3] L. Babai, L. Fortnow, L.A. Levin, M. Szegedy, Checking computations in polylogarithmic time, in *23rd Annual ACM Symposium on Theory of Computing, STOC '91* (Assoc. Comput. Mach., New York, 1991), pp. 21–31

[4] B. Barak, Y. Lindell, S.P. Vadhan, Lower bounds for non-black-box zero knowledge, in *44th Annual IEEE Symposium on Foundations of Computer Science, FOCS '03* (IEEE Computer Society, Los Alamitos, 2003), pp. 384–393

[5] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in *1st ACM Conference on Computer and Communications Security* (Assoc. Comput. Mach., New York, 1993), pp. 62–73

[6] I. Biehl, S. Paulus, T. Takagi, Efficient undeniable signature schemes based on ideal arithmetic in quadratic orders. *Des. Codes Cryptogr.* **31**(2), 99–123 (2004)

[7] D. Boneh, M. Franklin, Identity-based encryption from the Weil pairing, in *Advances in Cryptology—CRYPTO '01*. Lecture Notes in Computer Science, vol. 2139 (Springer, Berlin, 2001), pp. 213–229

[8] D. Boneh, M. Franklin, Identity-based encryption from the Weil pairing. *SIAM J. Comput.* **32**(3), 586–615 (2003). Full version of [7]

[9] J. Boyar, D. Chaum, I. Damgård, T.P. Pedersen, Convertible undeniable signatures, in *Advances in Cryptology—CRYPTO '90*. Lecture Notes in Computer Science, vol. 537 (Springer, Berlin, 1991), pp. 189–205

[10] C. Boyd, E. Foo, Off-line fair payment protocols using convertible signatures, in *Advances in Cryptology—ASIACRYPT '98*. Lecture Notes in Computer Science, vol. 1514 (Springer, Berlin, 1998), pp. 271–285

[11] G. Brassard, D. Chaum, C. Crépeau, Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.* **37**(2), 156–189 (1988)

[12] E. Bresson, D. Catalano, D. Pointcheval, A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications, in *Advances in Cryptology—ASIACRYPT '03*. Lecture Notes in Computer Science, vol. 2894 (Springer, Berlin, 2003), pp. 37–54

[13] L.M. Butler, A unimodality result in the enumeration of subgroups of a finite Abelian group. *Proc. Am. Math. Soc.* **101**(4), 771–775 (1987)

[14] J. Camenisch, M. Michels, Confirmer signature schemes secure against adaptive adversaries, in *Advances in Cryptology—EUROCRYPT '00*. Lecture Notes in Computer Science, vol. 1807 (Springer, Berlin, 2000), pp. 243–258

[15] D. Chaum, Zero-knowledge undeniable signatures, in *Advances in Cryptology—EUROCRYPT '90*. Lecture Notes in Computer Science, vol. 473 (Springer, Berlin, 1990), pp. 458–464

[16] D. Chaum, T.P. Pedersen, Wallet databases with observers, in *Advances in Cryptology—CRYPTO '92*. Lecture Notes in Computer Science, vol. 740 (Springer, Berlin, 1993), pp. 89–105

[17] D. Chaum, H. van Antwerpen, Undeniable signatures, in *Advances in Cryptology—CRYPTO '89*. Lecture Notes in Computer Science, vol. 435 (Springer, Berlin, 1990), pp. 212–217

[18] D. Chaum, E. van Heijst, B. Pfitzman, Cryptographically strong undeniable signatures, unconditionally secure for the signer, in *Advances in Cryptology—CRYPTO '91*. Lecture Notes in Computer Science, vol. 576 (Springer, Berlin, 1992), pp. 470–484

[19] J.-S. Coron, On the exact security of full domain hash, in *Advances in Cryptology—CRYPTO '00*. Lecture Notes in Computer Science, vol. 1880 (Springer, Berlin, 2000), pp. 229–235
[20] I. Damgård, T.P. Pedersen, New convertible undeniable signature schemes, in *Advances in Cryptology—EUROCRYPT '96*. Lecture Notes in Computer Science, vol. 1070 (Springer, Berlin, 1996), pp. 372–386
[21] Y. Desmedt, M. Yung, Weaknesses of undeniable signature schemes, in *Advances in Cryptology—EUROCRYPT '91*. Lecture Notes in Computer Science, vol. 547 (Springer, Berlin, 1991), pp. 205–220
[22] W. Diffie, M.E. Hellman, New directions in cryptography. *IEEE Trans. Inf. Theory* **22**(6), 644–654 (1976)
[23] S.D. Galbraith, W. Mao, Invisibility and anonymity of undeniable and confirmer signatures, in *Topics in Cryptology—CT–RSA '03*. Lecture Notes in Computer Science, vol. 2612 (Springer, Berlin, 2003), pp. 80–97
[24] R. Gennaro, H. Krawczyk, T. Rabin, RSA-based undeniable signatures. *J. Cryptol.* **13**(4), 397–416 (2000)
[25] The GNU Multiple Precision Arithmetic Library. http://www.swox.com/gmp/
[26] O. Goldreich, *Foundations of Cryptography, Volume I Basic Tools* (Cambridge University Press, Cambridge, 2001)
[27] S. Goldwasser, S. Micali, Probabilistic encryption. *J. Comput. Syst. Sci.* **28**(2), 270–299 (1984)
[28] S. Goldwasser, S. Micali, R.L. Rivest, A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.* **17**(2), 281–308 (1988)
[29] M. Jakobsson, Blackmailing using undeniable signatures, in *Advances in Cryptology—EUROCRYPT '94*. Lecture Notes in Computer Science, vol. 950 (Springer, Berlin, 1995), pp. 425–427
[30] M. Jakobsson, K. Sako, R. Impagliazzo, Designated verifier proofs and their applications, in *Advances in Cryptology—EUROCRYPT '96*. Lecture Notes in Computer Science, vol. 1070 (Springer, Berlin, 1996), pp. 143–154
[31] P. Junod, On the optimality of linear, differential, and sequential distinguishers, in *Advances in Cryptology—EUROCRYPT '03*. Lecture Notes in Computer Science, vol. 2656 (Springer, Berlin, 2003), pp. 17–32
[32] K. Kurosawa, Universally composable undeniable signature, in *Automata, Languages and Programming: 35th International Colloquium, ICALP '08*. Lecture Notes in Computer Science, vol. 5126 (Springer, Berlin, 2008), pp. 524–535
[33] K. Kurosawa, S.-H. Heng, 3-Move undeniable signature scheme, in *Advances in Cryptology—EUROCRYPT '05*. Lecture Notes in Computer Science, vol. 3494 (Springer, Berlin, 2005), pp. 181–197
[34] F. Laguillaumie, D. Vergnaud, Short undeniable signatures without random oracles: the missing link, in *Progress in Cryptology—INDOCRYPT '05*. Lecture Notes in Computer Science, vol. 3797 (Springer, Berlin, 2005), pp. 283–296
[35] F. Laguillaumie, D. Vergnaud, Time-selective convertible undeniable signatures, in *Topics in Cryptology—CT–RSA '05*. Lecture Notes in Computer Science, vol. 3376 (Springer, Berlin, 2005), pp. 154–171
[36] B. Libert, J.-J. Quisquater, Identity based undeniable signatures, in *Topics in Cryptology—CT–RSA '04*. Lecture Notes in Computer Science, vol. 2964 (Springer, Berlin, 2004), pp. 112–125
[37] I.G. Macdonald, *Symmetric Functions and Hall Polynomials* (Oxford University Press, London, 1979)
[38] M. Michels, M. Stadler, Efficient convertible undeniable signature schemes, in *Selected Areas in Cryptography—SAC '97* (1997), pp. 231–243
[39] M. Michels, H. Petersen, P. Horster, Breaking and repairing a convertible undeniable signature, in *3rd ACM Conference on Computer and Communications Security* (Assoc. Comput. Mach., New York, 1996), pp. 148–152
[40] J. Monnerat, Short undeniable signatures: design, analysis, and applications. PhD thesis, Thèse N° 3691, EPFL, Lausanne, Switzerland (2006)
[41] J. Monnerat, S. Vaudenay, Generic homomorphic undeniable signatures, in *Advances in Cryptology—ASIACRYPT '04*. Lecture Notes in Computer Science, vol. 3329 (Springer, Berlin, 2004), pp. 354–371
[42] J. Monnerat, S. Vaudenay, Undeniable signatures based on characters: how to sign with one bit, in *Public Key Cryptography—PKC '04*. Lecture Notes in Computer Science, vol. 2947 (Springer, Berlin, 2004), pp. 69–85
[43] J. Monnerat, S. Vaudenay, Short 2-move undeniable signatures, in *VIETCRYPT '06*. Lecture Notes in Computer Science, vol. 4341 (Springer, Berlin, 2006), pp. 19–36

[44] J. Monnerat, Y.A. Oswald, S. Vaudenay, Optimization of the MOVA undeniable signature scheme, in *Progress in Cryptology—MYCRYPT '05*. Lecture Notes in Computer Science, vol. 3715 (Springer, Berlin, 2005), pp. 196–209

[45] W. Ogata, K. Kurosawa, S.-H. Heng, The security of the FDH variant of Chaum's undeniable signature scheme, in *Public Key Cryptography—PKC '05*. Lecture Notes in Computer Science, vol. 3386 (Springer, Berlin, 2005), pp. 328–345. Extended version available on: Cryptology ePrint Archive, Report 2004/290, http://eprint.iacr.org/

[46] T. Okamoto, D. Pointcheval, The gap-problems: a new class of problems for the security of cryptographic schemes, in *Public Key Cryptography—PKC '01*. Lecture Notes in Computer Science, vol. 1992 (Springer, Berlin, 2001), pp. 104–118

[47] Y.A. Oswald, Generic homomorphic undeniable signature scheme: optimizations. Semester project, EPFL, LASEC, Lausanne, Switzerland (2005)

[48] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in *Advances in Cryptology—EUROCRYPT '99*. Lecture Notes in Computer Science, vol. 1592 (Springer, Berlin, 1999), pp. 223–238

[49] R. Pass, On deniability in the common reference string and random oracle model, in *Advances in Cryptology—CRYPTO '03*. Lecture Notes in Computer Science, vol. 2729 (Springer, Berlin, 2003), pp. 316–337

[50] D. Pointcheval, Self-scrambling anonymizers, in *Financial Cryptography, FC '00*. Lecture Notes in Computer Science, vol. 1962 (Springer, Berlin, 2001), pp. 259–275

[51] R.L. Rivest, A. Shamir, L.M. Adleman, A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**(2), 120–126 (1978)

[52] K. Sakurai, S. Miyazaki, An anonymous electronic bidding protocol based on a new convertible group signature scheme, in *Information Security and Privacy, ACISP '00*. Lecture Notes in Computer Science, vol. 1841 (Springer, Berlin, 2000), pp. 385–399

[53] V. Shoup, Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004). http://eprint.iacr.org/

[54] S. Vaudenay, Digital signature schemes with domain parameters: yet another parameter issue in ECDSA, in *Information Security and Privacy, ACISP '04*. Lecture Notes in Computer Science, vol. 3108 (Springer, Berlin, 2004), pp. 188–199