How to prove knowledge of small secrets

Carsten Baum[⋆], Ivan Damgård[⋆], Kasper Green Larsen[⋆⋆], and Michael Nielsen[⋆]

Department of Computer Science, Aarhus University
{cbaum,ivan,larsen,mik}@cs.au.dk

Abstract. We propose a new zero-knowledge protocol applicable to additively homomorphic functions that map integer vectors to an Abelian group. The protocol demonstrates knowledge of a short preimage and achieves amortised efficiency comparable to the approach of Cramer and Damgård from Crypto 2010, but gives a much tighter bound on what we can extract from a dishonest prover. Towards achieving this result, we develop an analysis for bins-and-balls games that might be of independent interest. We also provide a general analysis of rewinding of a cut-and-choose protocol as well as a method to use Lyubachevsky's rejection sampling technique efficiently in an interactive protocol when many proofs are given simultaneously.
Our new protocol yields improved proofs of plaintext knowledge for (Ring-)LWE-based cryptosystems, where such general techniques were not known before. Moreover, they can be extended to prove preimages of homomorphic hash functions as well.

Keywords: Proofs of Plaintext Knowledge, Lattice-based Encryption, Homomorphic Hashing, Integer Commitments

1 Introduction

Proofs of Knowledge In a zero-knowledge protocol, a prover convinces a sceptical verifier that some claim is true (and in some cases that he knows a proof) while conveying no other knowledge than the fact that the claim is true. Zero-knowledge protocols are one of the most fundamental tools in cryptographic protocol design. In particular, one needs zero-knowledge proofs of knowledge in multiparty computation to have a player demonstrate that he knows the input he is providing. This is necessary to be able to show (UC-)security of a protocol.

[⋆] Supported by The Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which part of this work was performed; by the CFEM research center (supported by the Danish Strategic Research Council) within which part of this work was performed; and by the Advanced ERC grant MPCPRO.

[⋆⋆] Supported by the Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation, grant DNRF84, a Villum Young Investigator Grant and an AUFF Starting Grant.

In this work, we will consider one-way functions $f : \mathbb{Z}^r \mapsto G$ where G is an Abelian group (written additively in the following), and where furthermore the function is additively homormorphic, i.e., $f(\boldsymbol{a})+f(\boldsymbol{b}) = f(\boldsymbol{a}+\boldsymbol{b})$. We will call such functions *ivOWF*'s (for homomorphic One-Way Functions over Integer Vectors). This turns out to be a very general notion: the encryption function of several (Ring-)LWE-based cryptosystems can be seen an ivOWF (such as the one introduced in [BGV12] and used in the so-called SPDZ protocol [DPSZ12]). Even more generally, the encryption function of any semi-homomorphic cryptosystem as defined in [BDOZ11] is an ivOWF. Also, in commitment schemes for committing to integer values, the function one evaluates to commit is typically an ivOWF (see, e.g., [DF02]). Finally, hash functions based on lattice problems such as [GGH96,LMPR08], where it is hard to find a short preimage, are ivOWFs.
We will look at the scenario where a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ are given $y \in G$ and $\mathcal{P}$ holds a short preimage $\boldsymbol{x}$ of y, i.e., such that $||\boldsymbol{x}|| \leq \beta$ for some β. $\mathcal{P}$ wants to prove in zero-knowledge that he knows such an $\boldsymbol{x}$. When f is an encryption function and y is a ciphertext, this can be used to demonstrate that the ciphertext decrypts and $\mathcal{P}$ knows the plaintext. When f is a commitment function this can be used to show that one has committed to a number in a certain interval.
An obvious but inefficient solution is the following 3-message protocol π:

(1) $\mathcal{P}$ chooses $\boldsymbol{r}$ at random such that $||\boldsymbol{r}|| \leq \tau \cdot \beta$ for some sufficiently large τ, the choice of which we return to below.
(2) $\mathcal{P}$ then sends $a = f(\boldsymbol{r})$ to $\mathcal{V}$.
(3) $\mathcal{V}$ sends a random challenge bit b.
(4) $\mathcal{P}$ responds with $\boldsymbol{z} = \boldsymbol{r} + b \cdot \boldsymbol{x}$.
(5) $\mathcal{V}$ checks that $f(\boldsymbol{z}) = a + b \cdot y$ and that $||\boldsymbol{z}|| \leq \tau \cdot \beta$.

If τ is sufficiently large, the distribution of $\boldsymbol{z}$ will be statistically independent of $\boldsymbol{x}$, and the protocol will be honest verifier statistical zero-knowledge[1]. On the other hand, we can extract a preimage of y from a cheating prover who can produce correct answers $\boldsymbol{z}_0, \boldsymbol{z}_1$ to $b = 0, b = 1$, namely $f(\boldsymbol{z}_1 - \boldsymbol{z}_0) = y$. Clearly, we have $||\boldsymbol{z}_1 - \boldsymbol{z}_0|| \leq 2 \cdot \tau \cdot \beta$. We will refer to the factor 2τ as the *soundness slack* of the protocol, because it measures the discrepancy between the interval used by the honest prover and what we can force a dishonest prover to do. The value of the soundness slack is important: if f is, e.g., an encryption function, then a large soundness slack will force us to use larger parameters for the underlying cryptosystem to ensure that the ciphertext decrypts even if the input is in the larger interval, and this will cost us in efficiency.
The naive protocol above requires an exponentially large slack to get zero-knowledge, but using Lyubachevsky's rejection sampling technique, the soundness slack can made polynomial or even constant (at least in the random oracle model).

[1] We will only be interested in honest verifier zero-knowledge here. In applications one would get security for malicious verifiers by generating the challenge in a trusted way, e.g., using a maliciously sure coin-flip protocol.

The obvious problem with the naive solution is that one needs to repeat the protocol k times where k is the statistical security parameter, to get soundness error probability 2^{-k}. This means that one needs to generate $\Omega(k)$ auxiliary f-values. We will refer to this as the *overhead* of the protocol and use it as a measure of efficiency.
One wants, of course as small overhead and soundness slack as possible, but as long as we only want to give a proof for a single f-value, we do not know how to reduce the overhead dramatically in general. But if instead we want to give a proof for k or more f-values, then we know how to reduce the *amortised* overhead: Cramer and Damgård [CD09] show how to get amortised overhead $O(1)$, but unfortunately the soundness slack is $2^{\Omega(k)}$, even if rejection sampling is used. In [DKL+13] two protocols were suggested, where one is only covertly secure, and we will not consider it here as our goal is full malicious security. The other one can achieve polynomial soundness slack with overhead $\Omega(\log(k)^2)$ and works only in the random oracle model[2].

1.1 Contributions & Techniques

In this work, we introduce a new paradigm for zero-knowledge proof of knowledge of preimage under an ivOWF, abbreviated ZKPoKP. For the first time, we are able to optimize both parameters, namely we obtain quasi-polynomial soundness slack (proportional to $(2k+1)^{\log(k)/2}$) and $o(1)$ ciphertext overhead, all results hold in the standard model (no random oracles are needed).
For our zero-knowledge proof, we use the following high-level strategy:

(1) Use a cut-and-choose style protocol for the inputs $y_1, \ldots, y_n$.
(2) Repeat the following experiment several times:
  (2.1) Let the verifier randomly assign each y_i to one of several *buckets*.
  (2.2) For each bucket, add all elements that landed in the bucket and have the prover demonstrate that he knows a preimage of the sum.

The intuition behind the proof then goes as follows: the first step will ensure that we can extract *almost* all of the required n preimages, in fact all but k where k is the security parameter. In the second step, since we only have k elements left that were "bad" in the sense that we could not yet extract a preimage, then if we have more than k buckets, say ck for a constant $c > 1$, there is a significant probability that many of the bad elements will be alone in a bucket. If this happens, we can extract a preimage by linearity of f. Furthermore, the cost of doing such a step is at most n additions, plus work that only depends on the security parameter k and is insignificant if $n \gg k$. We can now repeat the experiment some number of times to extract the remaining bad elements,

[2] The protocol in [DKL+13] is actually stated as a proof of plaintext knowledge for random ciphertexts, but generalizes to a protocol for ivOWFs. It actually offers a tradeoff between soundness slack and overhead in the sense that the overhead is $M \cdot \log(k)$, where M has to be chosen such that $(1/s)^M$ is negligible. Thus one can choose s to be $\mathsf{poly}(k)$ and $M = \log(k)$, or s to be constant and $M = k$.

while adjusting the number of buckets carefully. We are then able to prove that we can extract all preimages quickly, namely after $\log(k)$ repetitions, and this is what give us the small soundness slack. In comparison, in [CD09], the extraction takes place in $\Omega(k)$ stages, which leads to an exponential soundness slack.
Along the way to our main result, we make two technical contributions: first, we show a general result on what you can extract by rewinding from a prover that successfully passes a cut-and-choose test. Second, we show a method for using rejection sampling efficiently in an interactive protocol. In comparison, the protocol from[DKL$^+$13] also used rejection sampling to reduce the soundness slack, but in a more simplistic way that leads to a larger overhead. See Section 3.1 for more information on this.
Our protocol is honest verifier zero-knowlegde and is sound in the sense of a standard proof of knowledge, i.e., we extract the prover's witness by rewinding. Nevertheless, the protocol can be readily used as a tool in a bigger protocol that is intended to be UC secure against malicious adversaries. Such a construction is already known from [DPSZ12]. See more details in Section 4. Here we also explain more concretely how to use our protocol when f is an encryption function.

1.2 Related Work

On a high level, our approach is related to Luby Transform (LT) codes [Lub02]: here, a sender encodes a codeword by splitting it into blocks of equal size and then sending random sums of these, until the receiver is able to reconstruct all such blocks (because all sums are formed independently, this yields a so-called *erasure code*). We could actually use the LT code approach to construct a protocol like ours, but it would not be a good solution: LT codes do not have to consider any noise expansion because they handle vectors over $\mathbb{Z}_2$, rather than integer vectors. This is a problem since in the worst case a block is reconstructed after n operations, where n is the number of blocks in total, which yields a noise bound that is exponential.
The same bound can be achieved using the technique due to Cramer & Damgård [CD09]. The main technique is to prove linear combinations of ciphertexts using regular 1 out of 2 zero-knowledge proofs. If enough equations are proven correctly, then one can use gaussian elimination to recompute the plaintexts. Unfortunately (as with LT codes) this leads to a blowup in the preimage size that can be exponential, which is not desireable for practical applications.
A different amortization technique was introduced in [DKL$^+$13] and further improved in the full version of [BDTZ16]. The basic idea here is to produce a large number of *auxiliary* ciphertexts, open a part of them and open sums of the plaintexts to be proven and the plaintexts of the auxiliary ciphertexts. This experiment is repeated multiple times, and a combinatorial argument as in [NO09] can then be used to estimate the error probability. As already mentioned above, this proof technique needs $\Omega(\log(k))^2$ auxiliary ciphertexts per proven plaintext, which can be quite substantial for practical applications.
There has been other work conducted for specialized instances of ivOWFs, such as e.g. the proof of plaintext knowledge from [BCK$^+$14] which only applies to

Ring-LWE schemes[3]. Moreover the protocol of [LNSW13] can be applied to ivOWFs with a lattice structure, but the protocol comes with a large soundness gap per instance.

Notation

Throughout this work we will format vectors such as $\boldsymbol{b}$ in lower-case bold face letters, whereas matrices such as $\boldsymbol{B}$ will be in upper case. We refer to the ith position of vector $\boldsymbol{b}$ as $\boldsymbol{b}[i]$, let $[r] := \{1, ..., r\}$ and define for $\boldsymbol{b} \in \mathbb{Z}^r$ that $||\boldsymbol{b}|| = \max_{i \in [r]}\{|\boldsymbol{b}[i]|\}$. To sample a variable g uniformly at random from a set G we use $g \overset{\$}{\leftarrow} G$. Throughout this work we will let λ be a computational and k be a statistical security parameter. Moreover, we use the standard definition for polynomial and negligible functions and denote those as $\mathsf{poly}(\cdot), \mathsf{negl}(\cdot)$.

2 Homomorphic OWFs and Zero-Knowledge Proofs

In this section we will present an abstraction that covers as a special case proofs of plaintext knowledge for lattice-based cryptosystems, and many other cases as well, as explained in the introduction. We call the abstraction *homomorphic one-way functions over integer vectors*. It follows the standard definition of a OWF which can be found in [KL14].
Let $\lambda \in \mathbb{N}$ be the security parameter, G be an Abelian group, $\beta, r \in \mathbb{N}$,
$f : \mathbb{Z}^r \to G$ be a function and $\mathcal{A}$ be any algorithm. Consider the following game:

$\mathsf{Invert}_{\mathcal{A},f,\beta}(\lambda)$**:**
(1) Choose $\boldsymbol{x} \in \mathbb{Z}^r, ||\boldsymbol{x}|| \leq \beta$ and compute $y = f(\boldsymbol{x})$.
(2) On input $(1^\lambda, y)$ the algorithm $\mathcal{A}$ computes an $\boldsymbol{x}'$.
(3) Output 1 iff $f(\boldsymbol{x}') = y, ||\boldsymbol{x}'|| \leq \beta$, and 0 otherwise.

Definition 1 (Homomorphic OWF over Integer Vectors (ivOWF)). *A function $f : \mathbb{Z}^r \to G$ is called a homomorphic one-way function over the integers if the following conditions hold:*

(1) There exists a polynomial-time algorithm $eval_f$ such that $eval_f(\boldsymbol{x}) = f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{Z}^r$.
(2) For all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{Z}^r$ it holds that $f(\boldsymbol{x}) + f(\boldsymbol{x}') = f(\boldsymbol{x} + \boldsymbol{x}')$.
(3) For every probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\Pr[\mathsf{Invert}_{\mathcal{A},f,\beta}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

Our definition is rather broad and does capture, among other primitives, lattice-based encryption schemes such as [BGV12,GSW13,BV14] where the one-way property is implied by IND-CPA and β is as large as the plaintext space. Moreover it also captures hash functions such as [GGH96,LMPR08], where it is hard to find a preimage for all *sufficiently short* vectors that have norm smaller than β.

[3] Their approach only *almost* yields a proof a plaintext knowledge, due to technical limitations.

2.1 Proving Knowledge of Preimage

Consider a setting with two parties $\mathcal{P}$ and $\mathcal{V}$. $\mathcal{P}$ holds some values $\boldsymbol{x}_1, ..., \boldsymbol{x}_n \in \mathbb{Z}^r$, $\mathcal{V}$ has some $y_1, ..., y_n \in R$ and $\mathcal{P}$ wants to prove towards $\mathcal{V}$ that $y_i = f(\boldsymbol{x}_i)$ and that $\boldsymbol{x}_i$ is *short*, while not giving any knowledge about the $\boldsymbol{x}_i$ away. More formally, the relation that we want to give a zero-knowledge proof of knowledge for is

$$R_{\text{KSP}} = \left\{ (v, w) \;\middle|\; \begin{array}{l} v = (y_1, ..., y_n) \wedge w = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n) \wedge \\ \left[y_i = f(\boldsymbol{x}_i) \wedge ||\boldsymbol{x}_i|| \leq \beta \right]_{i \in [n]} \end{array} \right\}$$

However, like all other protocols for this type of relation, we will have to live with a *soundness slack* τ as explained in the introduction. What this means more precisely is that there must exist a knowledge extractor with properties exactly as in the standard definition of knowledge soundness, but the extracted values only have to satisfy $[y_i = f(\boldsymbol{x}_i) \wedge ||\boldsymbol{x}_i|| \leq \tau \cdot \beta]_{i \in [n]}$.

3 Proofs of Preimage

We start by constructing an *imperfect proof of knowledge*. That is, the protocol will allow to prove the above relation with a certain soundness slack, but the knowledge extractor is only required to extract almost all preimages. Furthermore, we will use this protocol as a subprotocol in our actual proof of knowledge. To show knowledge soundness, Goldreich and Bellare [BG93] have shown that it is sufficient to consider deterministic provers, therefore we only need to consider deterministic provers when proving the subprotocol.

On the use of rejection sampling. Conceptually, the idea is to run the naive 3-message protocol π from the intro once for each of the n instances to prove. However, in order to have a small soundness slack, we want to make use of of Lyubashevsky's rejection sampling technique [Lyu08,Lyu09]. The idea here is that the prover will sometimes abort the protocol after seeing the challenge if he notices that the random choices he made in the first message will lead him to reveal information about his witness if he were to send the final message. This is fine when used with the Fiat-Shamir heuristic because the prover only has to communicate the successful execution(s). But in our interactive situation, one has to allow for enough failed attempts so that the honest prover will succeed. The most straightforward idea is to have the prover start up one execution of π in parallel for each instance, complete those that are successful and try again for the rest (this was essentially the approach taken in [DKL+13]). The expected number of attempts needed is constant, so we get a protocol that is expected constant round, but may sometimes run for a longer time. Alternatively, the prover could start so many attempts in parallel for each instance that he is sure

to finish one of them. This will be exact constant round but wasteful in terms of work needed.
Here, we obtain the best of both worlds. The idea is the following: we can make a large list L of T candidates for the prover's first message, and then do standard cut-and-choose where we open half of them to show that most of the remaining ones are correctly formed. Now, for every instance to prove, the prover will take the first unused one from L that leads to success and complete the protocol for that one. Again, since the *expected* number of attempts for one instance is very close to 1, and we run over many instances, L only needs to be of length $O(n)$, the prover will run out of candidates only with negligible probability. Further, since this can all be done in parallel, we get an exact constant round protocol.

On extraction by rewinding from cut-and-choose. When we need to extract knowledge from the prover in the imperfect proof, we need to exploit the fact that we do cut-and-choose on the list of candidates L as mentioned above, where each candidate is an image under f. If we just wanted to establish that most of the candidates are well formed in the sense that they are images of short enough inputs, it would be easy: if each candidate is opened with probability $1/2$, then if more than k candidates are not well formed, the prover clearly survives with probability at most 2^{-k}. However, we have to actually extract preimages of almost all candidates. Since we want to avoid using random oracles or other set-up assumptions, we can only resort to rewinding. Now it is not so clear what happens: it may be that all candidates are well formed, but the corrupt prover has some (unknown to us) strategy for which challenges he wants to respond to correctly. All we know is that he will answer a non-negligible fraction of them. We show that nevertheless, there is a rewinding strategy that will do almost as well as in the easy case, and we treat this in a separate general lemma, as we believe the solution to this is of independent interest.
To establish this general point of view, consider any polynomial time computable function $g : X \mapsto Y$ and a generic protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ we call $\mathcal{P}_{\text{CutnChoose}}$ that works as follows:

(1) $\mathcal{P}$ chooses $x_1, ..., x_T \in X$ such that all x_i satisfy some predicate pre, we say x_i is good if it satisfies pre.
(2) $\mathcal{P}$ sets $y_i = g(x_i)$ for all i and sends $y_1, ..., y_T$ to $\mathcal{V}$.
(3) $\mathcal{V}$ chooses $\boldsymbol{s} \in \{0,1\}^T$ uniformly at random and sends it $\mathcal{P}$.
(4) $\mathcal{P}$ returns $\{x_i \mid \boldsymbol{s}[i] = 0\}$ and $\mathcal{V}$ accepts if $y_i = g(x_i)$ whenever $\boldsymbol{s}[i] = 0$ and each such x_i is good.

Lemma 1 (Cut-and-choose rewinding lemma). *There exists an extractor $\mathcal{E}$ such that the following holds: for any (deterministic) prover $\hat{\mathcal{P}}$ that makes the verifier in $\mathcal{P}_{\text{CutnChoose}}$ accept with probability $p > 2^{-k+1}$, where T is polynomial in k, $\mathcal{E}$ can extract from $\hat{\mathcal{P}}$ at least $T - k$ good x_i-values such that $g(x_i) = y_i$. $\mathcal{E}$ runs in expected time proportional to $O(\mathsf{poly}(s) \cdot k^2/p)$, where s is the size of the inputs.*

Proof. Let $\hat{\mathcal{P}}$ be a deterministic prover that makes $\mathcal{V}$ accept in $\mathcal{P}_{\text{CutnChoose}}$ with probability $p > 2^{-k+1}$. Consider the following algorithm $\mathcal{E}$:

(1) Start $\hat{\mathcal{P}}$, who in turn outputs $y_1, ..., y_T$.
(2) Run T instances of $\hat{\mathcal{P}}$ in parallel, which we denote $\hat{\mathcal{P}}_1, ..., \hat{\mathcal{P}}_T$.
(3) Let $A = \emptyset$ and do the following until $|A| \geq T - k$:
  (3.1) For each $\hat{\mathcal{P}}_i$ sample a random challenge $\boldsymbol{s}_i \overset{\$}{\leftarrow} \{0,1\}^T$, subject to $\boldsymbol{s}_i[i] = 0$ and run each $\hat{\mathcal{P}}_i$ on challenge $\boldsymbol{s}_i$.
  (3.2) For each instance $\hat{\mathcal{P}}_i$ that does not abort, check that the prover's response contains x_i such that $f(x_i) = y_i$. If so, then $A = A \cup \{x_i\}$.
(4) Output A.

We will now show that $\mathcal{E}$ runs in the required time. Denote the probability that $\hat{\mathcal{P}}_i$ outputs a good x_i in step (3) as p_i. We will say that p_i is bad if $p_i < p/k$, and good otherwise.
Let X_i be the event that $\hat{\mathcal{P}}_i$ eventually outputs a good x_i, where $X_i = 1$ if the event happened or $X_i = 0$ otherwise. If p_i is good then, after α iterations

$$\Pr[X_i = 0] = (1 - p/k)^{\alpha} \leq e^{-p/k \cdot \alpha}$$

so after at most $\alpha = k^2/p$ iterations we can expect that x_i was extracted except with probability negligible in k. This can then be generalized to the success of all $\hat{\mathcal{P}}_i$ (where p_i is good) by a union bound, and the probability of failing is still negligible because T is polynomial in k. Since the experiment of running k^2/p iterations produces success for all good p_i with probability essentially 1, the expected number of times we would need to repeat it to get success is certainly not more than 2, so the claimed expected run time follows, provided there are less than k bad p_i.
Hence, for the sake of contradiction, assume that there are k bad p_i which, for simplicity, are $p_1, ..., p_k$. In the protocol, the challenge $\boldsymbol{s}$ is chosen uniformly at random. The success probability of $\hat{\mathcal{P}}$ can be conditioned on the value of $\boldsymbol{s}[1]$ as

$$p = \Pr[\hat{\mathcal{P}} \text{ succeeds}] = 1/2 \cdot p_1 + 1/2 \cdot \Pr[\hat{\mathcal{P}} \text{ succeeds} \mid \boldsymbol{s}[1] = 1]$$

since p_1 is only of our concern if $\boldsymbol{s}[1] = 0$. Conditioning additionally on $\boldsymbol{s}[2]$ yields

$$\begin{aligned} p &\leq 1/2 \cdot p_1 + 1/2 \cdot (1/2 \cdot 2 \cdot p_2 + 1/2 \cdot \Pr[\hat{\mathcal{P}} \text{ succeeds} \mid \boldsymbol{s}[1] = 1 \wedge \boldsymbol{s}[2] = 1]) \\ &= 1/2 \cdot (p_1 + p_2) + 1/4 \cdot \Pr[\hat{\mathcal{P}} \text{ succeeds} \mid \boldsymbol{s}[1] = 1 \wedge \boldsymbol{s}[2] = 1] \end{aligned}$$

The reason the inequality holds is as follows: the probability that a random challenge asking to open a_2 will yield a preimage of a_2 is p_2. Now, conditioning on $\boldsymbol{s}[1] = 1$, which occurs with probability $1/2$, will increase that probability from p_2 to at most $2p_2$.
Repeating the above argument generalizes to

$$\begin{aligned} p = \Pr[\hat{\mathcal{P}} \text{ succeeds}] &\leq 1/2 \cdot (p_1 + ... + p_k) + \\ &\qquad 2^{-k} \cdot \Pr[\hat{\mathcal{P}} \text{ succeeds} \mid \boldsymbol{s}[1] = 1 \wedge ... \wedge \boldsymbol{s}[k] = 1] \\ &< 1/2 \cdot p + 2^{-k} \end{aligned}$$

which follows since the first k p_i were bad. But this last inequality implies $p < 2^{-k+1}$, and this contradicts the assumption we started from, that $p > 2^{-k+1}$. □

3.1 The Imperfect Proof of Knowledge

We assume the existence of an auxiliary commitment scheme C_{aux} that is computationally hiding and perfectly binding, and which allows to commit to values from the group G that f maps into. The reason we need it is quite subtle and will show up in the proof of security of $\mathcal{P}_{\text{ImperfectProof}}$. We will denote a commitment using C_{aux} to a value $x \in G$ as $C_{aux}(x)$.

Procedure $\mathcal{P}_{\text{ImperfectProof}}$

Let f be an ivOWF and C_{aux} be a commitment scheme over G. $\mathcal{P}$ inputs w to the procedure and $\mathcal{V}$ inputs v.

innerProof(v, w, T, τ, β) :

(1) Let $v = (y_1, ..., y_n), w = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$. $\mathcal{P}$ samples T values $\boldsymbol{g}_1, ..., \boldsymbol{g}_T \xleftarrow{\$} \mathbb{Z}^r$ such that $||\boldsymbol{g}_i|| \leq \tau \cdot \beta$.

(2) $\mathcal{P}$ computes $a_i = f(\boldsymbol{g}_i)$, $d_i = C_{aux}(a_i)$ and sends $d = (d_1, ..., d_T)$ to $\mathcal{V}$.

(3) $\mathcal{V}$ samples $\boldsymbol{s} \xleftarrow{\$} \{0,1\}^T$ and sends $\boldsymbol{s}$ to $\mathcal{P}$.

(4) Both parties set $C = \{i \mid \boldsymbol{s}[i] = 1\}$, $O = [T] \setminus C$.

(5) $\mathcal{P}$ sends $(\boldsymbol{g}_i)_{i \in O}$ as well as the randomness used to generate d_i to $\mathcal{V}$ who checks that $\forall i \in O$:

  (5.1) $a_i = f(\boldsymbol{g}_i)$

  (5.2) $||\boldsymbol{g}_i|| \leq \tau \cdot \beta$

  (5.3) $d_i = C_{aux}(a_i)$

  If any check fail he aborts.

(6) Let $Z = \emptyset$. For $i \in [n]$ do the following:

  (6.1) Find the first $j \in C \setminus Z$ such that $||\boldsymbol{g}_j + \boldsymbol{x}_i|| \leq (\tau - 1) \cdot \beta$. If no such j exists then continue. Else set $\boldsymbol{z}_i = \boldsymbol{g}_j + \boldsymbol{x}_i$ and $Z = Z \cup \{1, ..., j\}$.

(7) If $|Z| < n$ then $\mathcal{P}$ aborts.

(8) Else $\mathcal{P}$ sends $(Z, \boldsymbol{z}_1, ..., \boldsymbol{z}_n, (a_j)_{j \in Z})$ as well as the randomness used to generate $d_j, j \in Z$ to $\mathcal{V}$.

(9) $\mathcal{V}$ checks that

  (9.1) $||\boldsymbol{z}_i|| \leq (\tau - 1) \cdot \beta$

  (9.2) $a_j + y_i = f(\boldsymbol{z}_i)$

  (9.3) $d_j = C_{aux}(a_j)$

  If yes, then he outputs accept. Else, he outputs reject.

Fig. 1. Imperfect proof for the relation R_{KSP}

Theorem 1. *Let f be an ivOWF, k be a statistical security parameter, C_{aux} be a perfectly binding/computationally hiding commitment scheme over G, $\tau = 100 \cdot r$ and $T = 3 \cdot n, n \geq \max\{10, k\}$. Then $\mathcal{P}_{\text{ImperfectProof}}$ has the following properties:*

Correctness: *If $\mathcal{P}, \mathcal{V}$ are honest and run on an instance of R_{KSP}, then the protocol succeeds with probability at least $1 - \mathsf{negl}(k)$.*

Soundness: *For every deterministic prover $\hat{\mathcal{P}}$ that succeeds to run the protocol with probability $p > 2^{-k+1}$ one can extract at least $n - k$ values $\boldsymbol{x}'_i$ such that*

$f(\boldsymbol{x}'_i) = y_i$ and $||\boldsymbol{x}'_i|| \leq 2 \cdot \tau \cdot \beta$, in expected time $O(\mathsf{poly}(s) \cdot k^2/p)$ where s is the size of the input to the protocol.

Zero-Knowledge: *The protocol is computational honest-verifier zero-knowledge.*

Proof.

Completeness. By the homomorphic property of f, all the checked equations hold. The protocol can only abort if $\mathcal{P}$ aborts, which can only happen in step (7). We first show that $|C| \geq 1.1 \cdot n$ with all but negligible probability for large enough n. Using this, we show that $\Pr[\mathcal{P}_{\text{ImperfectProof}}$ aborts $\mid |C| \geq 1.1 \cdot n]$ is negligible in n.

Let $\#_1(\boldsymbol{s})$ denote the number of ones in $\boldsymbol{s}$, then $\#_1(\boldsymbol{s}) \sim \mathcal{BIN}_{1/2,T}$ where $\mathcal{BIN}$ is the Binomial distribution. Using the Chernoff bound we obtain

$$\begin{aligned}\Pr[\#_1(\boldsymbol{s}) \leq 1.1 \cdot n \mid \boldsymbol{s} \xleftarrow{\$} \{0,1\}^T] &\leq \exp\left(-2\frac{(1/2 \cdot T - 1.1 \cdot n)^2}{T}\right) \\ &= \exp\left(\frac{-32}{300} \cdot n\right)\end{aligned}$$

Since $n \geq k$ this becomes negligible for large enough n and we can assume that $|C| \geq 1.1 \cdot n$.

Consider a single coordinate of a $\boldsymbol{z}_i$. The chance that it fails the bound is $1/\tau$. Each vector has length r, so $\boldsymbol{z}_i$ exceeds the bound with probability $r/\tau = 1/100$. In such a case, $\mathcal{P}$ would take the next (independently chosen) $\boldsymbol{g}_j$ and try again. The ith attempt of $\mathcal{P}$ is denoted as X_i, where $X_i = 1$ if he fails and 0 otherwise. We allow $\mathcal{P}$ to do at most T of these attempts[4]. Then $X_i \sim \mathcal{B}_{1/100}$ and $X \sim \mathcal{BIN}_{1/100,T}, X = \sum X_i$. We set $\overline{X} = \frac{1}{T}X$ where $E[\overline{X}] = 1/100$. Using Hoeffding's inequality, one can show that the probability of failure is

$$\Pr[\overline{X} - E[\overline{X}] \geq 0.09] \leq \exp\left(-2.2 \cdot n \cdot 0.09^2\right)$$

which is negligible in k since we assume $n \geq k$.[5]

Soundness. Let $\hat{\mathcal{P}}$ be a deterministic prover that makes an honest $\mathcal{V}$ accept $\mathcal{P}_{\text{ImperfectProof}}$ with probability $p > 2^{-k+1}$. Consider the following algorithm $\mathcal{E}_{ImperfectProof}$:

(1) Start $\hat{\mathcal{P}}$, who in turn outputs $d = (d_1, ..., d_T)$.

(2) Observe that the first part of the protocol is an instance of $\mathcal{P}_{\text{CutnChoose}}$ with $g = C_{aux} \circ f$ and where a preimage $\boldsymbol{g}_i$ is good if $||\boldsymbol{g}_i|| \leq \tau \cdot \beta$, We therefore run the extractor $\mathcal{E}$ guaranteed by Lemma 1 which gives us a set A with $T - k$ good $\boldsymbol{g}_i$-values.

[4] The probability that $\mathcal{P}$ needs more auxiliary ciphertexts is $\approx 0.63^n$ and therefore negligible in n.

[5] In fact, setting $n = 40$ already makes $\mathcal{P}$ abort with probability 2^{-10}.

(3) Let $X = \emptyset$ and do the following until $|X| \geq n - k$:
  (3.1) Run a regular instance with $\hat{\mathcal{P}}$.
  (3.2) If the instance was accepting, then for each $\boldsymbol{z}_i$ with a corresponding $\boldsymbol{g}_j \in A, j \in C$ add the preimage to X, i.e. $X = X \cup \{\boldsymbol{z}_i - \boldsymbol{g}_j\}$.
(4) Output X.

We will now show that $\mathcal{E}_{ImperfectProof}$ runs in the required time. The run-time of $\mathcal{E}$ was established in Lemma 1. Using the set A it outputs, we can now argue that step (3) also terminates as required: $\mathcal{E}_{ImperfectProof}$ reaches step (3.2) after an expected number of $1/p$ rounds. At most k of the T preimages of a_j are not given in A and therefore step (3.2) is only executed once. From the bound on the a_i, the bound on the extracted $\boldsymbol{x}_i$ immediately follows.

Zero-Knowledge. Consider the following algorithm $\mathcal{S}_{\text{ImperfectProof}}$

(1) On input $(v = (y_1, ..., y_n), T, \tau, \beta)$ sample the string $\boldsymbol{s} \overset{\$}{\leftarrow} \{0,1\}^T$ as in the protocol.
(2) Compute the sets C, O as in $\mathcal{P}_{\text{ImperfectProof}}$. For each $i \in O$ sample $\boldsymbol{g}_i \in \mathbb{Z}^r, ||\boldsymbol{g}_i|| \leq \tau \cdot \beta$ and set $a_i = f(\boldsymbol{g}_i)$ as well as $d_i = C_{aux}(a_i)$.
(3) For $i \in C$ sample $\boldsymbol{z}_i \in \mathbb{Z}^r, ||\boldsymbol{z}_i|| \leq \tau \cdot \beta$ uniformly at random. Let $Z' := \{i \in C \mid ||\boldsymbol{z}_i|| \leq (\tau - 1) \cdot \beta\}$.
(4) If $|Z'| < n$ then for $i \in C$ set $\boldsymbol{g}_i = \boldsymbol{z}_i, a_i = f(\boldsymbol{z}_i), d_i = C_{aux}(a_i)$, output $(\boldsymbol{s}, d_1, ..., d_T, (a_i, \boldsymbol{g}_i)_{i \in O})$ and abort.
(5) If $|Z'| \geq n$ then let Z be the first n elements of Z'. For each $i \in C \setminus Z$ set $a_i = f(\boldsymbol{z}_i), d_i = C_{aux}(a_i)$.
(6) Denote Z as $Z = \{i_1, ..., i_n\}$. For all $i_j \in Z$ set $a_{i_j} = f(\boldsymbol{z}_{i_j}) - y_j, d_{i_j} = C_{aux}(a_{i_j})$.
(7) Output $(\boldsymbol{s}, d_1, ..., d_T, (a_i, \boldsymbol{g}_i)_{i \in O}, Z, (a_i, \boldsymbol{z}_i)_{i \in Z})$.

In the simulation, we can assume that there exists a witness w for v according to relation R_{KSP}. We first observe that if an a_i and its randomness when generating a d_i are ever revealed, then it holds that $d_i = C_{aux}(a_i)$. For those commitments that are not opened the computational hiding property implies that their distribution in the simulated case is indistinguishable from the real protocol.
What remains to study is the abort probability of the protocol, the sets C, O, Z and the $a_i, z_i, \boldsymbol{g}_i$. The choice of C, O is identical in $\mathcal{P}_{\text{ImperfectProof}}, \mathcal{S}_{\text{ImperfectProof}}$ for an honest verifier since they are computed the same way.

Abort probability and Z. The probability of abort of $\mathcal{S}_{\text{ImperfectProof}}$ in step (4) is the same as in (7) in $\mathcal{P}_{\text{ImperfectProof}}$. This indeed is true if $\#_1(\boldsymbol{s}) < n$ and also if $\#_1(\boldsymbol{s}) \geq n, |Z'| < n$. The second is a little more subtle and can be seen by arguing what the chance is that a certain $\boldsymbol{z}_i$ ends up in Z': for the sake of simplicity, assume $n = r = 1$ since all these vectors and their entries are chosen i.i.d. In the above simulator, $z \in [-\tau \cdot \beta, \tau \cdot \beta]$ was chosen uniformly at random. Hence $z \notin Z'$ with probability $1/100$. In the case of $\mathcal{P}_{\text{ImperfectProof}}$ we have $z = x + g$ where $g \in [-\tau \cdot \beta, \tau \cdot \beta]$ was chosen uniformly at random and $x \in [-\beta, \beta]$, i.e. $z \in [-\tau \cdot \beta + x, \tau \cdot \beta + x]$ chosen uniformly at random from a *shifted* interval of equal length. But $[-(\tau-1)\cdot\beta, (\tau-1)\cdot\beta] \subset [-\tau\cdot\beta+x, \tau\cdot\beta+x]$ always holds due to the upper bound of x, hence the probability of abort is also $1/100$. By the same reasoning, Z has the same distribution in $\mathcal{S}_{\textsc{ImperfectProof}}$ and $\mathcal{P}_{\textsc{ImperfectProof}}$.

Distribution of $\boldsymbol{g}_j, a_j$ *for* $j \in O$. Due to the homomorphism of f, the checks from step (9) do also hold on the simulated output. For all $j \in O$ the distribution of the $\boldsymbol{g}_j, a_j$ is the same in both the protocol and $\mathcal{S}_{\textsc{ImperfectProof}}$ as the values are chosen exactly the same way.

Distribution of $\boldsymbol{z}_i, a_i$ *for* $i \in Z$. Consider the distribution of the $\boldsymbol{z}_{i_j}, a_{i_j}$ for $i_j \in Z$ when $\mathcal{S}_{\textsc{ImperfectProof}}$ runs successfully. By the above argument, the distribution of the $\boldsymbol{z}_{i_j}$ is independent of the $\boldsymbol{x}_j$ in $\mathcal{P}_{\textsc{ImperfectProof}}$. In $\mathcal{P}_{\textsc{ImperfectProof}}$ exactly those $\boldsymbol{z}_{i_j}$ will be sent to $\mathcal{V}$ where $\boldsymbol{z}_{i_j} = \boldsymbol{g}_{i_j} + \boldsymbol{x}_j$ is in the correct interval. Since by our assumption there exists a witness $w = (\boldsymbol{x}'_1, ..., \boldsymbol{x}'_n)$ then due to the linearity of f there must exist a $\boldsymbol{g}'_{i_j}$ of the same bound as the $\boldsymbol{g}_i$ in the protocol, where $a_{i_j} = f(\boldsymbol{g}'_{i_j})$ by linearity.

Why using C_{aux}*?* It may not be directly obvious from the above proof why the commitment C_{aux} is necessary. But a problem can occur in step (7) of $\mathcal{P}_{\textsc{ImperfectProof}}$ with the elements with indices from $C \setminus Z$: although the simulator can simulate perfectly the choice of this set, we would have a problem if we had to reveal the corresponding $f(\boldsymbol{g}_i)$-values. The issue is that the $\boldsymbol{g}_i$'s should be values that cause an abort and we cannot choose such values unless we know the corresponding secrets. One solution is to apply f to a random input and make a non-standard assumption that this cannot be distinguished from the real thing, but this is undesirable. Instead, sending $C_{aux}(a_i)$ allows to both hide the distribution, while soundness is still guaranteed because $C_{aux} \circ f$ is hard to invert due to the binding property of C_{aux}.

□

3.2 The Full Proof of Knowledge

We use the above imperfect protocol as a building block of the actual proof. After executing it with the $(\boldsymbol{x}_i, y_i)$ as input, we can assume that a preimage of most of the y_i's (in fact, all but k) can be extracted from the prover.
Our strategy for the last part of the protocol is to repeat the following procedure several times: we let the verifier randomly assign each y_i to one of several *buckets*. Then, for each bucket, we add all elements that landed in the bucket and have the prover demonstrate that he knows a preimage of the sum. The observation is that since we only have k elements left that were "bad" in the sense that we could not yet extract a preimage, then if we have more than k buckets, say ck for a constant $c > 1$, there is a significant probability that many of the bad elements will be alone in a bucket. If this happens, we can extract a preimage by

linearity of f. Furthermore, the cost of doing such a step is at most n additions, plus work that only depends on the security parameter k and is insignificant if $n \gg k$. Now, by repeating this game some number of times with the right number of buckets, we shall see that we can extract all preimages quite quickly. In the following, the experiment where we throw n values randomly into b buckets will be denoted $\mathrm{Exp}(b, n)$. As is apparent from the above discussion, we will need to analyse the probability that the bad values will be "killed" by being alone in a bucket. That is, we need to consider $\mathrm{Exp}(b, v)$, where $v \leq k$ can be thought of as the number of bad elements. We will say that an element *survives* if it is not alone in a bucket. We will write t independent repetitions of the experiment as $\mathrm{Exp}^t(b, v)$ and we say that an element survives this if it survives in every repetition. The following lemma will be helpful:

Lemma 2. *Notation as above. Consider* $\mathrm{Exp}^t(b, v)$ *and assume* $b \geq 4v$ *and* $t \geq 8$*. Then the probability* p *that at least* $v/4$ *elements survive satisfies* $p \leq \frac{3v}{4}\epsilon^{tv}$*, where* $\epsilon = \frac{e^{5/32}}{2^{5/16}} \approx 0.94$.

Proof. Consider the event of exactly s bad elements surviving $\mathrm{Exp}^t(b, v)$ where $s \geq v/4$. The s surviving elements could be any of the v values, but must cover less than $s/2$ buckets in each repeated experiment, since surviving elements are not alone in a bucket. From this we get the bound

$$\Pr[s \text{ survive}] \leq \binom{v}{s}\left(\binom{b}{s/2}\left(\frac{s/2}{b}\right)^{s}\right)^{t}$$

on which we apply upper bounds on the binomial coefficients:

$$\leq \left(\frac{ve}{s}\right)^{s}\left(\left(\frac{be}{s/2}\right)^{s/2}\left(\frac{s/2}{b}\right)^{s}\right)^{t}$$
$$= \left(\frac{ve}{s}\right)^{s}\left(\frac{se}{2b}\right)^{ts/2}$$
$$= \left(\left(\frac{ve}{s}\right)^{1/t}\left(\frac{se}{2b}\right)^{1/2}\right)^{ts}$$

and finally maximize using $b \geq 4v, t \geq 8$ and $s \in [v/4, v]$:

$$\leq \left(\left(\frac{ve}{v/4}\right)^{1/8}\left(\frac{ve}{2 \cdot 4v}\right)^{1/2}\right)^{tv/4}$$
$$= \left((4e)^{1/8}\left(\frac{e}{8}\right)^{1/2}\right)^{tv/4}$$
$$= \left(\frac{e^{5/32}}{2^{5/16}}\right)^{tv}$$

Using this we can bound the probability p by union bound:

$$p = \Pr\left[\geq \frac{v}{4} \text{ survive}\right] \leq \sum_{s=v/4}^{v} \Pr\left[s \text{ survive}\right] \leq \frac{3v}{4}\left(\frac{e^{5/32}}{2^{5/16}}\right)^{tv}$$

□

Before we continue, let us discuss the implications of the above Lemma. First, due to the first equation of the proof we yield that, except with probability p, in an instance of $\mathrm{Exp}^t(b, v)$ *at least one* of the t iterations contains at least $3v/4$ buckets with single elements. A second, somewhat surprising fact is that for fixed values of t, b the probability p is not monotone in v. This will be particularly difficult, as we only know upper bounds on the value v in the proof of the main protocol.
Our soundness argument implicitly defines an extraction algorithm that runs in $\log_2(k)$ rounds, where in each round the same total number of buckets is used (the number of buckets per iteration drops in half, but the total number of iterations per round doubles). What we then show (using the above Lemma) is that the upper bound on the number of unextracted preimages is reduced by a factor of 2 between each two successive rounds, while the error probability stays somewhat constant. This is due to the following thought experiment: assume as an invariant that, for $O(k)$ buckets and k balls, at least $k/2$ of these balls land in their own bucket except with probability $2^{-O(k)}$. By running the experiment again, we see that the error probability now increases because we now only use $k/2$ balls. But by independently running the experiment twice, one obtains that half of the $k/2$ balls are alone (in one of the two experiments) except with essentially the same probability as before. This now allows for a recursive extraction strategy.

Theorem 2. *Let f be an ivOWF, k be a statistical security parameter, β be a given upper bound and $n > k \cdot \log_2(k)$. Then $\mathcal{P}_{\text{CompleteProof}}$ is an interactive honest-verifier zero-knowledge proof of the relation R_{KSP} with knowledge error 2^{-k+1}. More specifically, it has the following properties:*

Correctness: *If $\mathcal{P}, \mathcal{V}$ are honest then the protocol succeeds with probability at least $1 - 2^{-O(k)}$.*

Soundness: *For every deterministic prover $\hat{\mathcal{P}}$ that succeeds to run the protocol with probability $p > 2^{-k+1}$ one can extract n values $\boldsymbol{x}'_i$ such that $f(\boldsymbol{x}'_i) = y_i$ and $||\boldsymbol{x}'_i|| \leq O((2k+1)^{\log_2(k)/2} \cdot n \cdot r \cdot \beta)$ except with negligible probability, in expected time $\mathsf{poly}(s, k)/p$, where s is the size of the input to the protocol.*

Zero-Knowledge: *The protocol is computational honest-verifier zero-knowledge.*

Proof.

Correctness. The first call to $\mathcal{P}_{\text{ImperfectProof}}$ in step (1) will succeed with all but negligible probability due to Theorem 1. For each i in step (2) the experiment is repeated 2^{i+4} times using $4k \cdot 2^{-i}$ buckets, hence the total number of sums for

Procedure $\mathcal{P}_{\text{CompleteProof}}$

Let f be an ivOWF. $\mathcal{P}$ inputs w to the procedure and $\mathcal{V}$ inputs v. We assume for simplicity that the security parameter k is a 2-power.

proof(v, w, β) :

(1) Let $v = (y_1, ..., y_n), w = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$. Run innerProof$(v, w, 3n, 100r, \beta)$. If $\mathcal{V}$ in $\mathcal{P}_{\text{ImperfectProof}}$ aborts then abort, otherwise continue.
(2) For $i = 0, 1, \ldots, \log_2(k)$, execute (in parallel) $\text{Exp}^{t_i}(b_i, n)$, where $t_i = 2^{i+4}, b_i = 4k \cdot 2^{-i}$ and where $\mathcal{V}$ chooses the randomness for all the experiments, i.e., chooses how to distribute elements in buckets.
(3) Let $h = 64k \cdot (\log_2(k) + 1)$ be the total number of buckets used in all the experiments in the previous step and order all the buckets in some arbitrary order. Bucket j contains some subset of the n input values, let these be designated by index set I_j. Now, for $j = 1, \ldots, h$, both players compute $\gamma_j = \sum_{i \in I_j} v_i$ and $\mathcal{P}$ also computes $\boldsymbol{\delta}_j = \sum_{i \in I_j} \boldsymbol{x}_i$.
(4) Run innerProof$(\gamma, \boldsymbol{\delta}, 3h, 100r, n\beta)$. If $\mathcal{V}$ in $\mathcal{P}_{\text{ImperfectProof}}$ aborts then abort, otherwise accept.

Fig. 2. A protocol to prove the relation R_{KSP}

each such round is $64k$ which determines h. A set I_j as chosen in step (3) can have size at most n by definition, therefore $||\boldsymbol{\delta}_j|| \leq \beta \cdot n$. The call to $\mathcal{P}_{\text{ImperfectProof}}$ in step (4) will then be successful according to Theorem 1 with overwhelming probability.

Soundness. We will first prove the existence of an efficient extractor, then give a bound on the extraction probability and only establish the bound on the norm of the preimage afterwards.

An efficient extractor. From the subprotocol used in step (1) and Theorem 1 all but k of the n ciphertexts can be extracted. The same holds for step (4) from which we can argue that at most k of the h sums are proven incorrectly. For each i, observe that of the $t_i = 2^{i+4}$ iterations, there must be at least 2^{i+3} of them that each contain at most $2^{-i} \cdot k/4$ bad buckets. For otherwise, we would have at least 2^{i+3} iterations that each have at least $2^{-i} \cdot k/4$ buckets which adds up to $2k$ bad buckets.
For $i = 0, 1, ..$ the number of bad values entering into the experiment $\text{Exp}^{t_i}(b_i, n)$ is $v_i \leq k \cdot 2^{-i}$, except with negligible probability (we will consider the error probability later). This can be seen as follows: for $i = 0$, we have $v_0 \leq k$ due to step (1). So let $v_i \leq 2^{-i}k$, then by the proof of Lemma 2 at least one of the 2^{i+3} iterations, $3/4 \cdot v_i$ or more buckets contain only one of the not-yet extracted elements and can hence be extracted now. For this instance, we established that at most $2^{-i} \cdot k/4$ of the sums can be bad, hence

$$v_{i+1} \leq v_i/4 + k/4 \cdot 2^{-i} \leq k/4 \cdot 2^{-i} + k/4 \cdot 2^{-i} = k \cdot 2^{-i-1}$$

Hence after $\text{Exp}^{t_i}(b_i, v_i)$ we can extract at least $v_i/2$ of the ciphertexts. In the last round we have $v_{\log_2(k)} \leq 2$ and must prove that after $\text{Exp}^{t_{\log_2(k)}}(b_{\log_2(k)}, v_{\log_2(k)})$, no unextracted preimages are left. Therefore consider the following two cases:

$v_{\log_2(k)} = 1$ In this case, for the experiment to fail the remaining unextracted preimage must be in the bad sum for all $8k$ instances. For each such instance, there are 4 buckets out of which at most 1 can be bad. The extraction will hence only fail with probability 2^{-16k}.

$v_{\log_2(k)} = 2$ To be able to extract, we want both unextracted preimages to be in different buckets and none of them in a bad bucket. The chance that the first preimage ends up in a bad bucket is $1/4$, and the second preimage can either fall into the bucket with the first preimage (with probability $1/4$) or in the bad bucket, so in total with probability at most $3/4$ one of the $8k$ iterations will fail, and all will be bad with probability at most $(3/4)^{8k} < 2^{-2k}$.

By a union bound, the last experiment will fail with probability at most $p_{log_2(k)} = 2^{-k}$.

For rounds $i = 0, ..., \log_2(k) - 1$, the extractor will only extract from the experiment i if $k \cdot 2^{-i-1} \leq v_i \leq k \cdot 2^{-i}$ and otherwise safely continue with round $i + 1$. By Lemma 2, extraction will fail in this round with probability at most

$$\begin{aligned} p_i &\leq \max_{\tilde{v}_i \in [k \cdot 2^{-i-1}, k \cdot 2^{-i}]} \{3/4\tilde{v}_i \cdot \epsilon^{t_i \cdot \tilde{v}_i}\} \\ &< 3/4k \cdot 2^{-i} \cdot \epsilon^{2^{i+3} \cdot k \cdot 2^{-i-1}} \\ &= 3k \cdot 2^{2-i} \cdot \epsilon^{4k} \end{aligned}$$

because the actual value of v_i is unknown. The extraction process can fail if it fails in one of the experiments. By a union bound, we obtain

$$\begin{aligned} p_0 + ... + p_{\log_2(n)} &< 3k \cdot 2^2 \cdot \epsilon^{4k} + ... + 3k \cdot 2^{2-\log_2(k)+1} \cdot \epsilon^{4k} + 2^{-k} \\ &= 3k \cdot \epsilon^{4k} \cdot \sum_{j=0}^{\log_2(k)-1} 2^{2-j} + 2^{-k} \\ &< 24k \cdot \epsilon^{4k} + 2^{-k} \end{aligned}$$

which is in $2^{-O(k)}$ because $\epsilon < 1$ and constant. Since soundness for Theorem 1 fails with probability $2^{-O(k)}$ as well, this proves the claim.

Extraction bound. Let $\tau = 100r$ be the slackness chosen for the instances of innerProof. Consider a value $\boldsymbol{x}'_i$ extracted in round 0, i.e. there exists a *good* $\boldsymbol{\delta}_j, i \in I_j$ such that $\boldsymbol{x}'_i = \boldsymbol{\delta}_j - \sum_{o \in I_j \setminus \{\boldsymbol{x}'_i\}} \boldsymbol{x}'_o$ where all such $\boldsymbol{x}'_o$ were already extracted from $\mathcal{P}_{\textsc{ImperfectProof}}$ in step (1). Then

$$\begin{aligned}
||\boldsymbol{x}'_i|| &\leq ||\boldsymbol{\delta}_j - \sum_{o\in I_j\setminus\{i\}} \boldsymbol{x}'_o|| \\
&\leq ||\boldsymbol{\delta}_j|| + ||\sum_{o\in I_j\setminus\{i\}} \boldsymbol{x}'_o|| \\
&\leq 2\cdot\tau\cdot n\cdot\beta + (n-1)\cdot 2\cdot\tau\cdot\beta \\
&< 4\cdot n\cdot\tau\cdot\beta \\
&:= \beta_0
\end{aligned}$$

In round 1 each preimage that we extract will be a sum of preimage known from the cut-and-choose phase and those from round 0, where from the last round at most $k/2$ can be part of the sum. Calling this upper bound β_1 we obtain

$$\beta_1 = 2\cdot\tau\cdot n\cdot\beta + \frac{k}{2}\beta_0$$

The above argument easily generalizes to an arbitrary round $i > 0$ where it then holds that

$$\beta_i = 2\cdot\tau\cdot n\cdot\beta + \sum_{j=1}^{i-1}\frac{k}{2^j}\beta_{j-1}$$

because in round 0 we extracted at most $k/2$ preimages, in round 1 $k/4$ and so on. In particular, the above can be rewritten as

$$\begin{aligned}
\beta_i &= 2\cdot\tau\cdot n\cdot\beta + \sum_{j=1}^{i}\frac{k}{2^j}\beta_{j-1} \\
&= 2\cdot\tau\cdot n\cdot\beta + \sum_{j=1}^{i-1}\frac{k}{2^j}\beta_{j-1} + \frac{k}{2^i}\beta_{i-1} \\
&\leq \beta_{i-1} + \frac{k}{2^i}\beta_{i-1} \\
&= \left(\frac{k}{2^i}+1\right)\beta_{i-1}
\end{aligned}$$

In particular, for the bound on the last preimages that are extracted in round $\log_2(k)$ one obtains

$$\beta_{\log_2(k)} = \prod_{i=1}^{\log_2(k)-1}\left(\frac{k}{2^i}+1\right)\beta_0$$

To compute a bound on the leading product, we consider the square of the above bound and reorder the terms as

$$\begin{aligned}\prod_{i=1}^{\log_2(k)-1}\left(\frac{k}{2^i}+1\right)^2 &= \prod_{i=1}^{\log_2(k)-1}\left(\frac{k}{2^i}+1\right)\left(\frac{k}{2^{\log_2(k)-i}}+1\right)\\ &= \prod_{i=1}^{\log_2(k)-1}\left(k+\frac{k}{2^i}+\frac{k}{2^{\log_2(k)-i}}+1\right)\\ &< (2k+1)^{\log_2(k)}\end{aligned}$$

and we can conclude that

$$\beta_{\log_2(k)} < (2k+1)^{\log_2(k)/2}\cdot 4\cdot n\cdot\tau\cdot\beta$$

Zero-Knowledge. The simulation algorithm chooses the randomness for all the experiments like an honest $\mathcal{V}$ would do and then uses the simulator from Theorem 1 to simulate the calls to $\mathcal{P}_{\text{ImperfectProof}}$. The computational HVZK property then follows directly from Theorem 1. $\square$

4 Applications

As a first general remark, we note that even though our protocol is only honest verifier zero-knowlegde and proved sound using extraction by rewinding, we can nevertheless use it as a tool in a bigger protocol that is intended to be UC secure against malicious adversaries. Such a construction is already known from [DPSZ12]. The idea is first to generate the verifier's challenge using a secure coin-flip protocol. Then honest verifier zero-knowledge suffices, and the cost of generating the challenge can be amortised over several proofs. Second, if the function f is an encryption function, the UC simulator can be given the secret key and can hence extract straight-line. Rewinding then only takes place in the reduction to show that the UC simulator works.
In the rest of this section we will first show how to rephrase lattice-based encryption as ivOWFs and then show how to generalize the result from the previous section such that it applies in this setting.

4.1 Encryption as ivOWFs

As an example, let us consider a variant of the homomorphic encryption scheme due to Brakerski et al. [BGV12]. Let $n, N, \lambda \in \mathbb{N}^+, p, q \in \mathbb{P}$ and $q \gg p$. Moreover, let χ be a distribution over $\mathbb{Z}$ such that, with overwhelming probability in k, $e \leftarrow \chi \Rightarrow |e| \leq q/2$. We consider λ to be the computational security parameter.

$\mathsf{KG}(1^\lambda)$: Sample $\boldsymbol{t} \overset{\$}{\leftarrow} \chi^n, \boldsymbol{e} \overset{\$}{\leftarrow} \chi^N$ and $\boldsymbol{B} \leftarrow \mathbb{Z}_q^{N \times n}$. Let $\boldsymbol{u}_1 = (1, 0, ..., 0) \in \mathbb{Z}_q^{n+1}$ be the unit vector for the first coordinate. Then compute

$$\boldsymbol{b} \leftarrow \boldsymbol{B}\boldsymbol{t} + p \cdot \boldsymbol{e}$$
$$\boldsymbol{A} \leftarrow \left(\boldsymbol{u}_1^T \,\middle\|\, \begin{pmatrix} \boldsymbol{b}^T \\ -\boldsymbol{B}^T \end{pmatrix} \,\middle\|\, p \cdot \boldsymbol{I}_{n+1} \right)$$

where $\boldsymbol{I}_{n+1}$ is the identity matrix with $n+1$ rows and columns. Output $\mathsf{pk} \leftarrow \boldsymbol{A}, \mathsf{sk} \leftarrow \boldsymbol{t}$.

$\mathsf{Enc}_{\mathsf{pk}}(\begin{pmatrix} m \\ \boldsymbol{r} \end{pmatrix})$: Check that $m \in \mathbb{Z}_p$, $\boldsymbol{r} \in \mathbb{Z}_q^{N+n+1}$ and output

$$\boldsymbol{c} \leftarrow \boldsymbol{A} \times \begin{pmatrix} m \\ \boldsymbol{r} \end{pmatrix}$$

$\mathsf{Dec}_{\mathsf{sk}}(\boldsymbol{s})$: Compute

$$m' \leftarrow \left(\left\langle \boldsymbol{c}, \begin{pmatrix} 1 \\ \boldsymbol{t} \end{pmatrix} \right\rangle \bmod q \right) \bmod p$$

and output $m' \in \mathbb{Z}_p$.

Procedure $\mathcal{P}_{CompleteProof,f,\boldsymbol{\beta}}$

$\mathcal{P}$ inputs w to the procedure and $\mathcal{V}$ inputs v. We assume for simplicity that the security parameter k is a 2-power.

pG$(v, w, \boldsymbol{\beta})$:

(1) Let $v = (y_1, ..., y_n), w = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$. Run iPG$(v, w, 3n, 100r, \boldsymbol{\beta})$. If $\mathcal{V}$ in $\mathcal{P}_{ImperfectProof,f,\boldsymbol{\beta}}$ aborts then abort, otherwise continue.

(2) For $i = 0, 1, \ldots, \log_2(k)$, execute (in parallel) Exp$^{t_i}(b_i, n)$, where $t_i = 2^{i+4}, b_i = 4k \cdot 2^{-i}$ and where $\mathcal{V}$ chooses the randomness for all the experiments, i.e., chooses how to distribute elements in buckets.

(3) Let $h = 64k \cdot (\log_2(k) + 1)$ be the total number of buckets used in all the experiments in the previous step and order all the buckets in some arbitrary order. Bucket j contains some subset of the n input values, let these be designated by index set I_j. Now, for $j = 1, \ldots, h$, both players compute $\gamma_j = \sum_{i \in I_j} v_i$ and $\mathcal{P}$ also computes $\boldsymbol{\delta}_j = \sum_{i \in I_j} \boldsymbol{x}_i$.

(4) Run iPG$(\gamma, \boldsymbol{\delta}, 3h, 100r, n\boldsymbol{\beta})$. If $\mathcal{V}$ in $\mathcal{P}_{ImperfectProof,f,\boldsymbol{\beta}}$ aborts then abort, otherwise accept.

Fig. 3. A protocol to prove the relation $R_{\mathrm{KSP},f,\boldsymbol{\beta}}$

For appropriately chosen parameters, the function $\mathsf{Enc}_{\mathsf{pk}}$ is an ivOWF (by the natural embedding of $\mathbb{Z}_q$ into the integers) assuming that the LWE problem is hard. It therefore seems natural to apply our proof framework in the above setting.
Unfortunately we have to show different bounds for different indices of the preimage, which is impossible for the existing proof.

4.2 Refining the Proof Technique

To gain more flexibility, we start out by defining a predicate $\mathsf{InfNorm}_{\boldsymbol{\beta}}$, which we define as follows

$$\mathsf{InfNorm}_{\boldsymbol{\beta}}(\boldsymbol{x}) = \begin{cases} \top & \text{if } \boldsymbol{\beta} \in \mathbb{N}^+ \wedge \forall i \in [r] : |\boldsymbol{x}[i]| \leq \boldsymbol{\beta}[i] \\ \bot & \text{else} \end{cases}$$

where $\boldsymbol{\beta}$ is supposed to be a *coordinatewise upper bound* on $\boldsymbol{x}$.

Procedure $\mathcal{P}_{ImperfectProof,f,\boldsymbol{\beta}}$

Let f be an ivOWF and C_{aux} be a commitment scheme over G. $\mathcal{P}$ inputs w to the procedure and $\mathcal{V}$ inputs v.

```
iPG($v, w, T, \tau, \boldsymbol{\beta}$) :
    (1) Let $v = (y_1, ..., y_n), w = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$. $\mathcal{P}$ samples $T$ values $\boldsymbol{g}_1, ..., \boldsymbol{g}_T \overset{\$}{\leftarrow} \mathbb{Z}^r$
        such that $\mathsf{InfNorm}_{\tau\cdot\boldsymbol{\beta}}(\boldsymbol{g}_i)$.
    (2) $\mathcal{P}$ computes $a_i = f(\boldsymbol{g}_i)$, $d_i = C_{aux}(a_i)$ and sends $d = (d_1, ..., d_T)$ to $\mathcal{V}$.
    (3) $\mathcal{V}$ samples $\boldsymbol{s} \overset{\$}{\leftarrow} \{0,1\}^T$ and sends $\boldsymbol{s}$ to $\mathcal{P}$.
    (4) Both parties set $C = \{i \mid \boldsymbol{s}[i] = 1\}$, $O = [T] \setminus C$.
    (5) $\mathcal{P}$ sends $(\boldsymbol{g}_i)_{i \in O}$ as well as the randomness used to generate $d_i$ to $\mathcal{V}$ who
        checks that $\forall i \in O$ :
      (5.1) $a_i = f(\boldsymbol{g}_i)$
      (5.2) $\mathsf{InfNorm}_{\tau\cdot\boldsymbol{\beta}}(\boldsymbol{g}_i)$
      (5.3) $d_i = C_{aux}(a_i)$
        If any check fail he aborts.
    (6) Let $Z = \emptyset$. For $i \in [n]$ do the following:
      (6.1) Find the first $j \in C \setminus Z$ such that $||\boldsymbol{g}_j + \boldsymbol{x}_i|| \leq (\tau - 1) \cdot \beta$. If no such $j$
            exists then continue. Else set $\boldsymbol{z}_i = \boldsymbol{g}_j + \boldsymbol{x}_i$ and $Z = Z \cup \{1, ..., j\}$.
    (7) If $|Z| < n$ then $\mathcal{P}$ aborts.
    (8) Else $\mathcal{P}$ sends $(Z, \boldsymbol{z}_1, ..., \boldsymbol{z}_n, (a_j)_{j \in Z})$ as well as the randomness used to gen-
        erate $d_j, j \in Z$ to $\mathcal{V}$.
    (9) $\mathcal{V}$ checks that
      (9.1) $\mathsf{InfNorm}_{(\tau-1)\cdot\boldsymbol{\beta}}(\boldsymbol{z}_i)$
      (9.2) $a_j + y_i = f(\boldsymbol{z}_i)$
      (9.3) $d_j = C_{aux}(a_j)$
        If yes, then he outputs accept. Else, he outputs reject.
```

Fig. 4. Imperfect proof for the relation $R_{\text{KSP},f,\boldsymbol{\beta}}$

We call a vector $\boldsymbol{x} \in \mathbb{Z}^r$ to be *$\boldsymbol{\beta}$-bounded* iff $\mathsf{InfNorm}_{\boldsymbol{\beta}}(\boldsymbol{x}) = \top$. For a function $f : \mathbb{Z}^r \to G$ and the set $\{\boldsymbol{c}_1, ..., \boldsymbol{c}_t\}$ one then tries to prove the following relation

$$R_{\text{KSP},f,\boldsymbol{\beta}} = \Big\{ (\boldsymbol{v}, \boldsymbol{w}) \mid \quad \boldsymbol{v} = (\boldsymbol{c}_1, ..., \boldsymbol{c}_t) \wedge \boldsymbol{w} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_t) \wedge \\ \left[\boldsymbol{c}_i = f(\boldsymbol{x}_i) \wedge \mathsf{InfNorm}_{\boldsymbol{\beta}}(\boldsymbol{x}_i)\right]_{i \in [t]} \Big\}$$

That is, a proof of plaintext knowledge for our defined cryptosystem would then set $\boldsymbol{\beta} \in \mathbb{N}^{N+n+2}, f = \mathsf{Enc}_{\mathsf{pk}}$ with $\boldsymbol{\beta}[1] = \beta_P, \boldsymbol{\beta}[2] = ... = \boldsymbol{\beta}[N+n+2] = \beta_R$ where β_P is the bound on the plaintext and β_R on the randomness. One then uses a modified version of the proof $\mathcal{P}_{\text{ImperfectProof}}$, namely the protocol from Figure 4 and moreover replaces $\mathcal{P}_{\text{CompleteProof}}$ with Figure 3.
Theorem 1 and 2 directly generalize to the above setting due to the linearity of all operations (if the simulators for the rejection sampling just sample from the appropriate bound for each coordinate). This is possible because none of the success probabilities changes since these are independent of the bound β in the first place. The above could be generalized to other predicates which e.g. enforce ℓ_2-norms. We leave this as future work.

References

BCK⁺14. Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *Advances in Cryptology–ASIACRYPT 2014*, pages 551–572. Springer, 2014.

BDOZ11. Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 169–188. Springer Berlin Heidelberg, 2011.

BDTZ16. Carsten Baum, Ivan Damgård, Tomas Toft, and Rasmus Zakarias. Better preprocessing for secure multiparty computation. In *ACNS 2016*. Springer, 2016.

BG93. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology—CRYPTO'92*, pages 390–420. Springer, 1993.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

BV14. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871, 2014.

CD09. Ronald Cramer and Ivan Damgård. On the amortized complexity of zero-knowledge protocols. In *Advances in Cryptology-CRYPTO 2009*, pages 177–191. Springer, 2009.

DF02. Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Advances in Cryptology—ASIACRYPT 2002*, pages 125–142. Springer, 2002.

DKL⁺13. Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure mpc for dishonest majority - or: Breaking the spdz limits. In *ESORICS*, pages 1–18, 2013.

DPSZ12. Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology - CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, Berlin, Germany, 2012.

GGH96. Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 3, pages 236–241, 1996.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013*, pages 75–92. Springer, 2013.

KL14. Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC Press, 2014.

LMPR08. Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. Swifft: A modest proposal for fft hashing. In *Fast Software Encryption*, pages 54–72. Springer, 2008.

LNSW13. San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. Improved zero-knowledge proofs of knowledge for the isis problem, and applications. In *Public-Key Cryptography–PKC 2013*, pages 107–124. Springer, 2013.

Lub02. Michael Luby. Lt codes. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, page 271. IEEE Computer Society, 2002.

Lyu08. Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *Public Key Cryptography–PKC 2008*, pages 162–179. Springer, 2008.

Lyu09. Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *Advances in Cryptology–ASIACRYPT 2009*, pages 598–616. Springer, 2009.

NO09. Jesper Buus Nielsen and Claudio Orlandi. Lego for two-party secure computation. In *Theory of Cryptography*, pages 368–386. Springer, 2009.