# Amortization with Fewer Equations for Proving Knowledge of Small Secrets

Rafael del Pino[1,2,3,4], and Vadim Lyubashevsky[4]

[1] INRIA, Paris
[2] École Normale Supérieure, Paris
[3] CNRS
[4] IBM Research Zurich

**Abstract.** For a linear function $f$, a vector $\mathbf{x}$ with small coefficients, and a vector $y = f(\mathbf{x})$, we would like to be able to give a zero-knowledge proof for the knowledge of an $\mathbf{x}'$ with small coefficients that satisfies $f(\mathbf{x}') = y$. This is a common scenario in lattice-based cryptography, and there is currently no satisfactory solution for this problem. All known protocols are built via the repetition a basic protocol that only has constant (1/2 or 2/3) soundness error. This implies that the communication complexity of the final protocol will be at least a factor of $k$ larger than that of the basic one, where $k$ is the security parameter.

One can do better if one considers simultaneously proving the knowledge of many instances of the above linear equation. The protocol that has the smallest amortized communication complexity while achieving close-to-optimal slack (i.e. the ratio between the coefficients in the secret and those that can be extracted from the proof) is due to Cramer et al. (Eurocrypt '17) which builds on an earlier work of Baum et al. (Crypto '16). The main downside of this protocol is that the amortization only kicks in when the number of equations is rather large – $4k^2$. This means that for $k = 128$, it is only truly optimal when one has more than $2^{16}$ equations to prove. The aforementioned work of Cramer et al. also shows how to achieve a protocol requiring $o(k^2)$ samples, but it is only applicable for much larger values of $k$ and the number of required samples ends up being larger than $2^{16}$.

The main result of our work is reducing the concrete minimal number of equations required for the amortization, while keeping the communication complexity almost unchanged. The cost of this is an increase in the running time of the zero-knowledge proof. More specifically, we show that one can decrease the required number of equations by a factor of $\Omega(\log^2 \alpha)$ at the cost of increasing the running time by a factor of $\Omega(\alpha)$. For example, increasing the running time by a factor of 8 allows us to decrease the required number of samples from 66000 to 4500 – a factor of 14. As a side benefit, the slack of our protocol decreases by a factor of $\log \alpha$ as well.

We also show that in the case that $f$ is a function over the polynomial ring $\mathbb{Z}[X]/(X^d + 1)$ and we would like to give a proof of knowledge of an $\mathbf{x}'$ with small coefficients such that $f(\mathbf{x}') = 2y$, then the number of samples needed for amortization is even lower. Without any trade-offs in the running time, our algorithm requires around 2000 samples, and for the same factor 8 increase in the running time, the requirement goes down to 850.

## 1 Introduction

Every lattice-based cryptographic construction relies on the fact that when given a matrix $\mathbf{A}$ and a vector $y$ over some ring $R$ (such as $\mathbb{Z}_q$ or $\mathbb{Z}_q[X]/(X^d + 1)$ with the usual addition and multiplication operations), it is hard to recover a vector $\mathbf{x}$ with small coefficients such that

$$\mathbf{A}\mathbf{x} = y. \tag{1}$$

In many instances, one would also like to construct a zero-knowledge protocol where the prover, who knows $\mathbf{x}$, is able to convince a verifier (who only has $\mathbf{A}$ and $y$) that he possesses this knowledge.

There are several known approaches for constructing such protocols. The first method is to adapt the classic Stern protocol [Ste93], which was used for a similar code-based problem, to working over larger rings [KTX08,LNSW13]. The main issue with this protocol is that each round has soundness error

2/3 and therefore needs to be repeated 192 times (to achieve 128 bits of security). For most practical applications, this technique is therefore unsuitable.

A second approach is to use the "Fiat-Shamir with Aborts" idea of Lyubashevsky [Lyu08, Lyu09, Lyu12] whose original application was to digital signatures. If one uses a ring $R$ that contains a lot of elements with small coefficients (e.g. $R = \mathbb{Z}_q[X]/(X^d + 1)$), then one can prove the knowledge of a short $\mathbf{x}'$ and $c \in R$ such that $\mathbf{A}\mathbf{x}' = cy$. This is not exactly equivalent to proving (1), but it suffices for the purposes of digital signatures, commitments [?], and to some applications of verifiable encryption [LN17].

The most natural and useful scenario, however, is proving the knowledge of some $\mathbf{s}'$ that exactly satisfies (1). One could directly apply the "Fiat-Shamir with Aborts" technique with 0/1 challenges, but this leads to protocols with soundness error 1/2, which is essentially as inefficient as those using the Stern technique. When working over the ring $R = \mathbb{Z}_q[X]/(X^d + 1)$, it was shown that one can decrease the soundness error to $1/(2d+1)$ [BCK$^+$14] and prove the knowledge of an $\mathbf{x}'$ such that $\mathbf{A}\mathbf{x}' = 2y$. The main observation in that paper was that rather than using challenges from the set 0/1, one could use them from the set $\{0, X^i\}$ for $0 \leq i < 2d$. Even though this latter proof does not exactly prove (1), the fact that one can prove the knowledge for a constant multiple of $y$ (rather than some arbitrary, unknown $c$) makes this type of proof suitable for a variety of applications. But still, the soundness error of $1/(2d+1)$ would require the proof to be repeated around a dozen times for typical values of $d = 1024$.

**Amortized Proofs.** A very interesting line of work, which built upon ideas from [CD09], considered the *amortized* complexity of the [Lyu08, Lyu09] protocol. In [DPSZ12], it was shown that one could prove the knowledge of a linear (in the security parameter) number of equations with essentially optimal communication per equation. The main downside was that, while the prover may have known $\mathbf{x}_i$ with small coefficients that satisfied $\mathbf{A}\mathbf{x}_i = y_i$, he would only be able to prove knowledge of $\mathbf{x}_i'$ whose coefficients were on the order of $2^{\Omega(k)}$ larger. In practice, this *slack* is quite bad as it would require setting all the parameters to be very large so as to make the proofs non-vacuous (i.e. so that there isn't an efficient algorithm that can simply compute such $\mathbf{x}'$ from $\mathbf{A}$ and $y$).

More recently, using different and novel ideas, Baum et al. [BDLN16] showed how to reduce the slack to super-polynomial in the security parameter, and the most recent work of Cramer and Damgard [CD16] reduced this slack to being only a factor $k$ larger than what one would get by running the basic protocol from [Lyu08, Lyu09] with 0/1 challenges. The main downside of this latter algorithm is that it requires doing at least $4k^2$ proofs at the same time. So for $k = 128$, this implies that one needs to have at least $2^{16}$ equations that one wishes to prove simultaneously. When wanting to prove fewer than that, one could include some "dummy" values, but this will have the effect of increasing the per-proof communication complexity and running time. The main open direction in this line of work is therefore to reduce the necessary number of equations while keeping the slack and communication to be as low as in [CD16]. This is the main result of the current paper.

## 1.1 Prior Work

**High-level overview of [BDLN16, CD16].** We will use the notation from [CD16]. The setup is that the prover has a linear function $f$ and ordered pairs $(y_1, \mathbf{x}_1), \ldots, (y_n, \mathbf{x}_n)$ such that $f(\mathbf{x}_i) = y_i$ (in (1), the function $f$ is defined by the matrix $\mathbf{A}$). He wishes to prove the knowledge of $\mathbf{x}_i'$ with small coefficients such that $f(\mathbf{x}_i') = y_i$. The algorithm from [CD16] works in two stages. In the first stage, it runs the "imperfect prover" from [BDLN16] which proves the knowledge of all-but-$k$ $\mathbf{x}_i'$. The main issue is that after the first stage, we do not know which $k$ secrets the extractor cannot extract.

In the second stage, the prover creates $4k^2$ additive combinations of $y_i$, for which the pre-image is the corresponding additive combination of the $\mathbf{x}_i$ due to the linearity of the function $f$.[1] The main result of the paper is showing a strategy for producing these combinations such that for any set $S$ of $\mathbf{x}_i$ of size $k$, each $\mathbf{x}_i$ from $S$ appears in at least $k + 1$ combinations without any other $\mathbf{x}_i$ from $S$. One can then run the imperfect proof on the $4k^2$ linear combinations and again get the guarantee that all but $k$ secrets can be extracted. Each element in $S$ therefore appears in some extracted combination in which all other elements were already extracted in the first stage. And due to the linearity of $f$, we can now extract the sole element from the set $S$ appearing in the combination.

[1] To be more precise, the number of combinations is the first prime greater than $4k^2$.

| | [CD16] | 0/1 Challenges | | | $\mathbf{x}^i$ Challenges | | |
|---|---|---|---|---|---|---|---|
| variable parameter $\alpha$ | 2 | 16 | 64 | 1024 | 16 | 64 | 1024 |
| minimum equations $n$ | 67103 | 4493 | 2213 | 853 | 853 | 443 | 293 |
| communication per equation ($\approx$ kB) | 8.5 | 9.3 | 9.7 | 10.9 | 8.9 | 9.5 | 10.7 |
| run-time per equation (function evaluations) | 16 | 128 | 512 | 8092 | 128 | 512 | 8092 |

**Table 1.** Trade-offs between the running time and the minimum number of samples. We are considering proofs for (Ring)-LWE instances of dimension 1024 where the secrets and errors have coefficients drawn from $\{-1, 0, 1\}$.

An asymptotically more efficient construction is also given in [CD16]. This construction uses two different additive combinations of the $y_i$, the first one is a relaxed version in which for any set $S$ of $\mathbf{x}_i$ of size $k$, all but $k - 5k^{0.75}$ of the $\mathbf{x}_i$ from $S$ appears in at least $k+1$ combinations without any other $\mathbf{x}_i$ from $S$. By running the imperfect proof on these sums all but $5k^{0.75}$ secrets can now be extracted. The second additive combination is identical to the one of the previous proof but is now used on sets of size $5k^{0.75}$, ensuring that after another execution of the imperfect proof all secrets can be extracted. This improved version requires at least $4(5k^{0.75})^2 = 100k^{1.5} = O(k^{1.5})$ secrets. However it is clear that this construction only makes sense if $k > 5k^{0.75}$, i.e. $k > 625$. So while this construction is more efficient asymptotically we only consider the previous one which is better for all reasonable security parameters.

**More concrete description of the "imperfect proof" from [BDLN16].** The original protocol from [BDLN16] is a $\Sigma$-protocol that can be seen as a very particular type of parallel composition of the protocol from [Lyu08]. The basic protocol from [Lyu08] for proving the knowledge of $\mathbf{x}'$ such that $f(\mathbf{x}') = y$ is as follows: The prover starts by choosing a mask $\mathbf{g}$ from some distribution and sends $h = f(\mathbf{g})$. The verifier then chooses a random bit $b \in \{0, 1\}$ as a challenge and sends it to the prover. The prover computes $c\mathbf{x} + \mathbf{g}$ and performs a rejection sampling step (the rejection sampling step is necessary for zero-knowledge). If it passes, then the prover sends $c\mathbf{x} + \mathbf{g}$ to the verifier. The verifier checks that $f(c\mathbf{x} + \mathbf{g}) = cy + h$.

The idea in [BDLN16] for giving "imperfect proofs" for $n$ equations was to choose $T = 2\kappa n$ masking parameters $\mathbf{g}_j$ (for some small constant $\kappa$) and send $h_j = f(\mathbf{g}_j)$ to the verifier. The verifier then sends a $T$-bit challenge string $c_1, \ldots, c_T$, and the prover sends the $\mathbf{g}_j$ for which $c_j = 0$. For every $1 \le i \le n$, the prover also tries to send $\mathbf{x}_i + \mathbf{g}_j$ for the first non-used $\mathbf{g}_j$ (a $\mathbf{g}_j$ is considered used if it was revealed in the clear or was previously tried to be used for masking another $\mathbf{x}_{i'}$ with $i' < i$ – there should initially be approximately $\kappa n$ unused $\mathbf{g}_j$). If the rejection sampling step passes, then the prover indeed sends the $\mathbf{x}_i + \mathbf{g}_j$. Otherwise, he tries to send $\mathbf{x}_i + \mathbf{g}_{j'}$ where $\mathbf{g}_{j'}$ is the next unused $\mathbf{g}$. The verifier checks that all the revealed $\mathbf{g}_j$ satisfy $f(\mathbf{g}_j) = h_j$, and then checks that $y_i + h_j = f(\mathbf{x}_i + \mathbf{g}_j)$ for all $i$. It is then shown that if a prover succeeds with probability $2^{-k+1}$, then an extractor can extract $n - k$ $\mathbf{x}'_i$ that satisfy $f(\mathbf{x}'_i) = y_i$. Thus the protocol is a proof of knowledge of all-but-$k$ pre-images.

## 1.2 Our Results

Our main result builds upon the works of [BDLN16, CD16] and allows us to reduce the required minimum number of proofs at the expense of a higher running time. Most importantly, the communication complexity per equation does not increase too much. As an example, if we increase the running-time by a factor of 8, we can decrease the required number of equations from 67000 to around 4400 (see Table 1). We also construct a protocol for proving knowledge of $\mathbf{s}_i$ with small coefficients over the ring $R = \mathbb{Z}_q[X]/(X^d + 1)$ such that $\mathbf{A}\mathbf{s}_i = 2\mathbf{t}_i$. This protocol gets an even better trade-off between running time and the minimum number of samples. For the same factor of 8 increase in running time, we only now need to have 853 equations.

Figure 1 shows a graph that illustrates how increasing the running time by a factor $\alpha$ reduces the minimum number of required equations. The implication is that for larger values of $\alpha$, the added reduction in the minimum number of equations is not worth the increase in the running time. For practical purposes, the best trade-offs are achieved for small $\alpha$'s. Figure 2 illustrates the small effect that increasing $\alpha$ has on the communication complexity of the protocol. Even increasing $\alpha$ by $2^{20}$, which is not advisable as we just mentioned, would result in the communication complexity growing by less than a factor of 2.

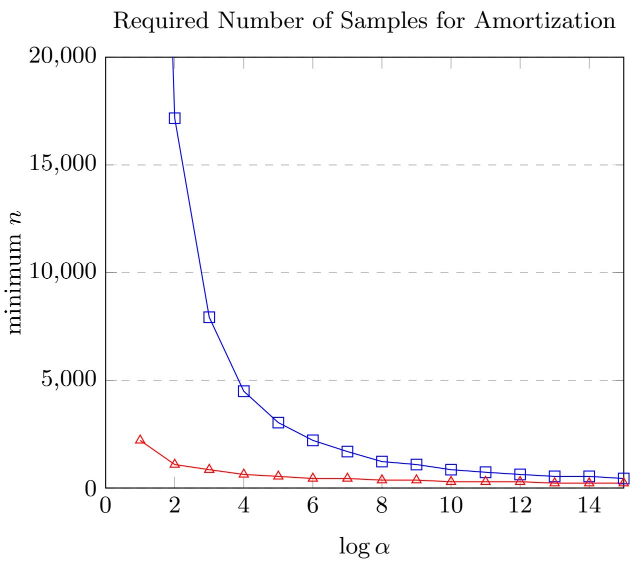


**Fig. 1.** The minimum number of samples required for amortization as a function of $\log \alpha$. The squares represent our first protocol (with 0/1 challenges) and the triangles represent the second (with challenges of the form $X^i$) when working over the ring $\mathbb{Z}[X]/(X^d + 1)$ for $d = 1024$.

**Techniques.** We achieve this improvement by modifying the first stage of the protocol – that is, the "imperfect proof" from [BDLN16]. Improving this protocol to make it a proof of knowledge of all-but-$\tau$ pre-images for some $\tau < k$, allows us to only do the amortized second stage of [CD16] with only $4\tau^2 < 4k^2$ equations. A way to reduce $\tau$ is for the prover to produce a larger number of $h_j$ in the first step of the $\Sigma$-protocol and then for the verifier to demand that the prover reveal the pre-images of a larger fraction of the $h_j$. The protocol of [BDLN16] can be thought of as a cut-and-choose protocol, thus more reveals intuitively implies a higher probability of the correctness of the non-revealed parts. If we introduce a parameter $\alpha$, then the prover produces $T = \alpha\kappa n$ elements $h_j$ in the first part, sends them to the verifier, and receives a challenge $c_1, \ldots, c_T$ where a $1 - 1/\alpha$ fraction of the $c_j$ are 0. The prover reveals the pre-images of the corresponding $h_j$ and then uses the non-revealed $\mathbf{g}_j$ (of which there are $\kappa n$) to send $\mathbf{x}_i + \mathbf{g}_j$ in the same manner as in [BDLN16] described in Section 1.1. We prove that this results in a protocol that proves the knowledge of all-but-$\tau$ pre-images for $\tau = k/\log \alpha$. Therefore, now only $4(k/\log \alpha)^2$ equations are needed for amortization to kick in.

One issue that still needs to be resolved is the communication complexity. Naively, it seems that one would need to send $T = \alpha\kappa n$ elements $h_j$ which would increase the communication complexity by a factor $\alpha$. We instead give an approach in which the communication is only logarithmically dependent on $\alpha$ – furthermore it will only be small additive factors that have a dependence on $\log \alpha$. Rather than sending $h_1, \ldots, h_T$, the prover can instead send a hash $h = H(h_1, \ldots, h_T)$ where $H$ is a collision-resistant hash function. This does not completely solve the problem because at some point the prover will need to send the $h_j$ so that the verifier can check the validity of $h$. But here we use the fact that all except $\kappa n$ of the $h_j$ will have their pre-images simply revealed. Our strategy is therefore as follows: we create the $\mathbf{g}_j$ from 256-bit seeds $s_j$ which are leaves on a tree generated by a pseudorandom function (modeled as a random oracle). That is, from the root of the tree, one can generate the entire tree. When required to reveal pre-images of a set of $h_j$, the prover does not need to send the $\mathbf{g}_j$ (or their seeds) individually. He can instead send roots of sub-trees which only include the seeds that will be revealed. We prove that with this strategy, rather than sending $\alpha\kappa n$ seeds, one only needs to send a maximum of $\kappa n \log \alpha$ many elements from the tree (which are themselves 256 bits each).

Putting everything together, we show that at the expense of increasing the running time by a factor of $\alpha$, one can reduce the minimum number of samples required for amortization by a factor of $\log^2 \alpha$. Our second contribution is showing that when working over the ring $\mathbb{Z}[X]/(X^d + 1)$, proving the knowledge of $\mathbf{x}_i$ such that $f(\mathbf{x}_i) = 2y_i$ has an even better trade-off between running-time and the minimum number

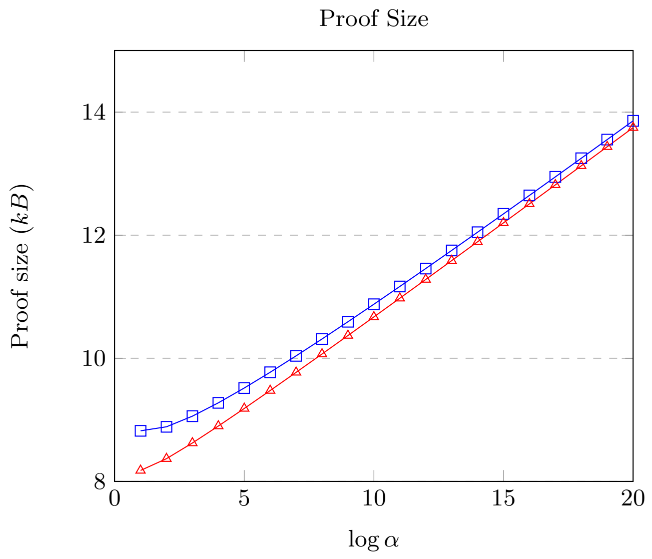


**Fig. 2.** Proof sizes as a function of $\log \alpha$. We are considering proofs for the same types of instances as in Table 1. The squares represent our first protocol (with 0/1 challenges) and the triangles represent the second (with $X^i$ challenges) when working over the ring $\mathbb{Z}[X]/(X^d + 1)$ for $d = 1024$.

of samples. In particular, we show that at the expense of an $\alpha$-fold increase in running time, one can reduce the minimum number of vectors by a factor of $\left(\frac{\log \alpha + \log 2d}{1 + 1/\log \alpha}\right)^2$.

To obtain such an improvement we adapt the proof of [BCK+14] to the framework of [BDLN16]. Though merging the two protocols is rather straightforward, the knowledge extractors of both of these schemes don't combine as nicely. The knowledge extractor of [BDLN16] first recovers a set of all but $k$ of the masking parameters $\mathbf{g}_j$ and then simply extracts $\mathbf{x}_i$ from $\mathbf{x}_i + \mathbf{g}_j$. This method falls apart when used with the protocol of [BCK+14] as the latter scheme uses rewinding to obtain two equations $X^a\mathbf{x} + \mathbf{g}$ and $X^b\mathbf{x} + \mathbf{g}$ and recovers a pre-image from their difference. The same rewinding is still possible in our scheme but will yield two equations of the form $X^a\mathbf{x}_i + \mathbf{g}_j$ and $X^b\mathbf{x}_i + \mathbf{g}_{j'}$ and extraction will be only possible if $j = j'$, which cannot be guaranteed. We resolve this issue by conditioning our extractor on the fact that $j = j'$ which results in a slightly sub-optimal number of extracted preimages: $n - \frac{k\cdot(1+1/\log \alpha)}{\log \alpha + \log 2d}$ instead of simply $n - \frac{k}{\log \alpha + \log 2d}$. It is not clear to us whether this small loss is necessary or simply an artifact of our proof.

### 1.3 Paper Organization

In Section 2, we introduce the notation and definitions that we will be using throughout the paper. In Section 3 we present a modification of the "imperfect proof" protocol of [BDLN16], which is a proof of knowledge of all-but-$\tau$ pre-images for $\tau = k/\log \alpha$. This protocol only serves as intuition, and we do not formally prove its correctness or security because the communication complexity (i.e. the proof size) grows linearly in $\alpha$. In Section 4, we show how to reduce the communication complexity of the interactive protocol from Section 3 and prove its correctness, zero-knowledge, and soundness. We only show honest-verifier zero-knowledge because this is enough to convert the protocol to a non-interactive one using the Fiat-Shamir transform, which is the manner in which one would use these schemes in practice. Analyzing the size of the communication is delayed until in Section 6 because this analysis also applies to the protocol in Section 5. In Section 5, we show that if the proof is done over the ring $\mathbb{Z}[X]/(X^d + 1)$, then the number of required equations can be made even smaller if one wants to prove $f(\mathbf{x}') = 2y$.

## 2 Preliminaries

### 2.1 Notation

We will write vectors such as $\mathbf{b}$ or $\mathbf{B}$ in bold face. We refer to the i$^{th}$ position of a vector $\mathbf{b}$ as $\mathbf{b}[i]$. Define $[r] = \{1, \ldots, r\}$. The euclidean norm of a vector , $\mathbf{b} \in \mathbb{Z}^r$ is $\|\mathbf{b}\| = \sqrt{\sum_{i \in [r]} \mathbf{b}[i]^2}$. For a set $\mathcal{S}$, we write $s \overset{\$}{\leftarrow} \mathcal{S}$ to denote that $s$ was drawn uniformly at random from $\mathcal{S}$. For a distribution $D$, we write $s \leftarrow D$ to denote that $s$ is drawn from $D$.

### 2.2 Homomorphic OWF

In this section we follow the framework of [BDLN16] in defining homomorphic one-way functions over integer vectors (which includes polynomial rings) as well as amortized zero-knowledge proofs of preimage for these functions. Let $\lambda \in \mathbb{N}$ be a 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:

$\mathbf{Invert}_{\mathcal{A},f,\beta}(\lambda)$ :

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

- *There exist a polynomial time algorithm $eval_f$ such that $eval_f(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{Z}^r$.*
- *for all $\mathbf{x}, \mathbf{x}' \in \mathbb{Z}^r$ it holds that $f(\mathbf{x}) + f(\mathbf{x}') = f(\mathbf{x} + \mathbf{x}')$.*
- *for every PPT algorithm $\mathcal{A}$ there exists a negligible function $\mathrm{negl}(\lambda)$ such that:*

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

### 2.3 Rejection Sampling and the Normal Distribution

For a protocol to be zero-knowledge, the output of the prover needs to be independent of his secret. In certain situations achieving this independence requires rejection sampling. While [BDLN16] used rejection sampling in the infinity norm (as in [Lyu08, Lyu09]) we use the euclidean norm and thus rejection sampling over the $\ell_2$ norm using normal distributions (as in [Lyu12]), which allows for tighter parameters. But all our techniques easily work for the $\ell_\infty$ norm as well.

**Definition 2 (Continuous Normal Distribution).** *The continuous Normal distribution over $\mathbb{R}^r$ centered at $\mathbf{v}$ with standard deviation $\sigma$ is defined by the probability density function $\rho^r_{\mathbf{v},\sigma}(\mathbf{x}) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^r e^{-\frac{\|\mathbf{x}-\mathbf{v}\|^2}{2\sigma^2}}$*

**Definition 3 (Discrete Normal Distribution).** *The discrete Normal distribution over $\mathbb{Z}^r$ centered at $\mathbf{v}$ with standard deviation $\sigma$ is defined by the probability mass function $\mathcal{D}^r_{\mathbf{v},\sigma}(\mathbf{x}) = \rho^r_{\mathbf{v},\sigma}(\mathbf{x}) / \rho^r_\sigma(\mathbb{Z}^r)$*

**Lemma 4 (Tail-Cut Bound [Ban93]).** $\Pr\left[\|\mathbf{z}\| \geq 2\sigma\sqrt{r}; \mathbf{z} \leftarrow \mathcal{D}^r_\sigma\right] < 2^{-r}$

**Theorem 5 (Rejection sampling [Lyu12] Theorem 4.6).** *Let $V$ be a subset of $\mathbb{Z}^r$ with elements of norm less than $T$, let $h$ be a distribution over $V$. Let $\sigma = 11T$, for $\mathbf{v}, \mathbf{z} \in \mathbb{Z}^r$ let $\mathbf{Rej}(\mathbf{v}, \mathbf{z})$ be the algorithm that outputs 1 with probability $\min\left(\mathcal{D}^r_\sigma(\mathbf{z}) / (3\mathcal{D}^r_{\mathbf{v},\sigma}(\mathbf{z})), 1\right)$ and 0 otherwise. Then we have:*

$$(\mathbf{v}, \mathbf{z} \mid \mathbf{Rej}(\mathbf{v}, \mathbf{z}) = 1) \sim_s (\mathbf{v}, \mathbf{z}')$$

*Where $\mathbf{v} \leftarrow h$, $\mathbf{z} \leftarrow \mathcal{D}^r_{\mathbf{v},\sigma}$, and $\mathbf{z}' \leftarrow \mathcal{D}^r_\sigma$, i.e. the output of $\mathbf{Rej}$ is a discrete Normal distribution centered on 0. Moreover the probability that $\mathbf{Rej}$ outputs 1 is exponentially close to $1/3$.*

### 2.4 Zero-Knowledge Proofs of Knowledge

We will consider amortized proofs of knowledge for preimages of an ivOWF. Formally, given an ivOWF $f$ the relation we want to give a zero-knowledge proof of knowledge for is:

$$\mathcal{R}_{\mathrm{KSP}}(n, f, \beta) = \left\{ (Y, \mathbf{X}) \in (G \times \mathbb{Z}^r)^n \;\middle|\; Y = (y_1, \ldots, y_n) \wedge \mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \right. \\ \left. \wedge\, [y_i = f(\mathbf{x}_i) \wedge \|\mathbf{x}_i\| \leq \beta]_{i \in [n]} \right\}$$

We define a second binary relation $\mathcal{R}'$, such that $\mathcal{R} \subset \mathcal{R}'$, which characterizes the soundness slack of the protocol, i.e. while the input to the protocol is a pair $(Y, \mathbf{X}) \in \mathcal{R}$ the knowledge extractor can only extract values in $\mathcal{R}'$. Typically the relation $\mathcal{R}'$ is identical to $\mathcal{R}$ except for the fact that the components of $\mathbf{X}$ are bounded in norm by a constant $\beta' = \tau\beta$ with $\tau > 1$. We will however see in section 5 a ZKPOK for a different relation $\mathcal{R}'$.

**Definition 6 (Zero-Knowledge Proof of Knowledge).** *Let $\mathcal{P}_{ZK}$ be a two-party protocol, let $\mathcal{R}, \mathcal{R}'$ be binary relations such that $\mathcal{R} \subseteq \mathcal{R}'$, let $k$ be a statistical security parameter. $\mathcal{P}_{ZK}$ is a zero-knowledge proof of knowledge if the following properties hold:*

**Correctness:** *If $\mathcal{P}, \mathcal{V}$ are honest and run $\mathcal{P}_{ZK}$ on an instance of $\mathcal{R}$, then the protocol terminates with probability greater than $1 - 2^{O(k)}$*

**Soundness:** *For any pair $(a, b) \in \mathcal{R}$, for any deterministic prover $\hat{\mathcal{P}}$ that succeeds with probability $p > 2^{-k}$ one can extract $b'$ such that $(a, b') \in \mathcal{R}'$ in expected time poly$(s, k) \cdot 1/p$, where $s$ is the size of the input to the protocol.*

**Computational Honest-Verifier Zero-Knowledge:** *There exists an expected PPT simulator $\mathcal{S}$ such that for any $(a, b) \in \mathcal{R}$, and for any PPT algorithm $\mathcal{A}$. $\mathcal{A}$ has advantage* $\mathrm{negl}(k)$ *in distinguishing between the two following distributions:*

- *$View_{\mathcal{V}}[\mathcal{P}(a, b) \leftrightarrow \mathcal{V}(a)]$ the view of $\mathcal{V}$ consisting in the transcript of the protocol as well as the random coins of $\mathcal{V}$.*
- *$\mathcal{S}(a)$*

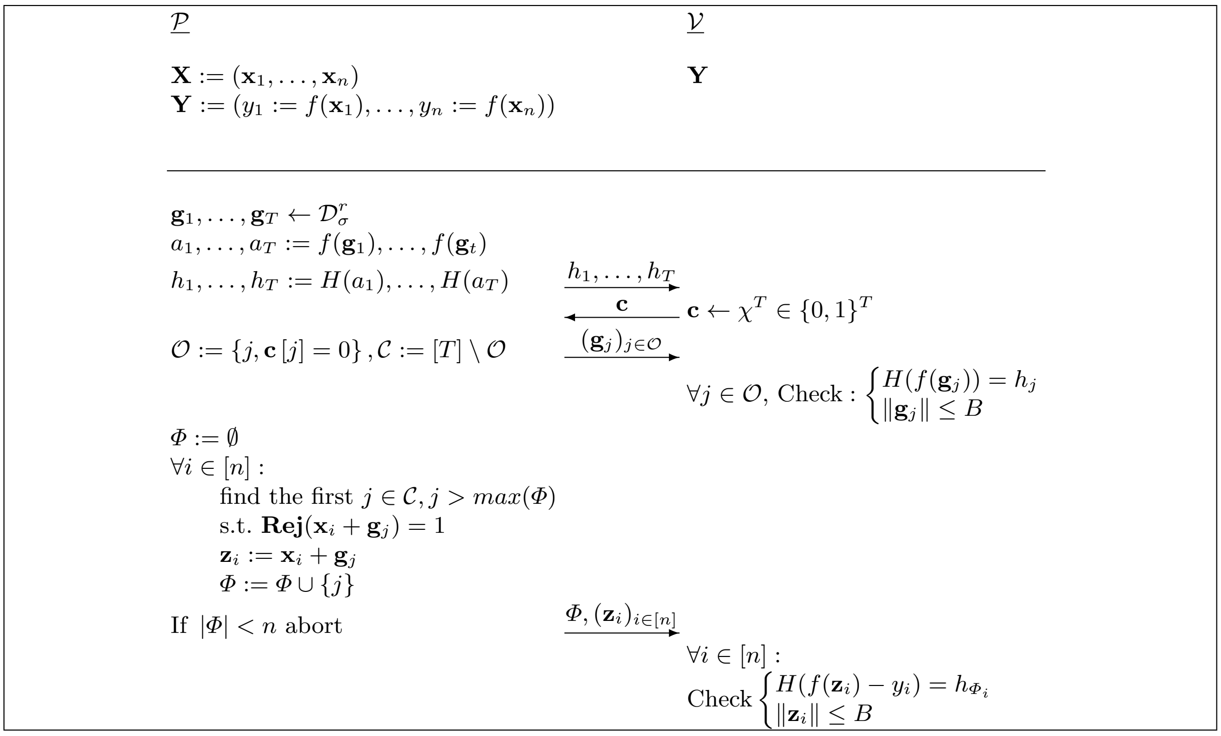
### 2.5 Imperfect Proof of Knowledge and a Compiler

In [BDLN16], the authors introduce the concept of an imperfect proof of knowledge. An imperfect proof of knowledge is a protocol that proves knowledge of pre-images in the relation $\mathcal{R}_{\mathrm{KSP}}$, however the knowledge extractor is not required to be able to extract all the pre-images.

**Definition 7 (Imperfect Proof of knowledge).** *Let $\mathcal{P}_{IProof}$ be a two-party protocol, let $f$ be an ivOWF, let $\mathcal{R}_{KSP}(n, f, \beta)$ and $\mathcal{R}_{KSP}(n, f, \beta')$ be two binary relations on $f$, $k$ be the security parameter. The protocol $\mathcal{P}_{IProof}$ is an imperfect proof of knowledge with imperfection $\tau(k)$ if the following properties hold:*

**Correctness:** *If $\mathcal{P}, \mathcal{V}$ are honest and run on an instance of $\mathcal{R}$, then the protocol terminates with probability greater than $1 - negl(k)$*

**Soundness:** *For any pair $(Y = (y_1, \ldots, y_n), \mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)) \in \mathcal{R}_{KSP}(n, f, \beta)$, for any deterministic prover $\hat{\mathcal{P}}$ that succeeds with probability $p > 2^{-k}$ one can extract at least $n - \tau(k)$ values $\mathbf{x}'_i$ such that $f(\mathbf{x}'_i) = y_i$ and $\|\mathbf{x}'_i\| \leq \beta'$ in expected time poly$(s, k) \cdot 1/p$, where $s$ is the size of the input to the protocol.*



$\underline{\mathcal{P}}$ $\qquad$ $\underline{\mathcal{V}}$

$\mathbf{X} := (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ $\qquad$ $\mathbf{Y}$

$\mathbf{Y} := (y_1 := f(\mathbf{x}_1), \ldots, y_n := f(\mathbf{x}_n))$

$\mathbf{g}_1, \ldots, \mathbf{g}_T \leftarrow \mathcal{D}^r_\sigma$

$a_1, \ldots, a_T := f(\mathbf{g}_1), \ldots, f(\mathbf{g}_t)$

$h_1, \ldots, h_T := H(a_1), \ldots, H(a_T)$ $\xrightarrow{h_1, \ldots, h_T}$

$\xleftarrow{\mathbf{c}}$ $\mathbf{c} \leftarrow \chi^T \in \{0,1\}^T$

$\mathcal{O} := \{j, \mathbf{c}[j] = 0\}, \mathcal{C} := [T] \setminus \mathcal{O}$ $\xrightarrow{(\mathbf{g}_j)_{j \in \mathcal{O}}}$

$\forall j \in \mathcal{O}$, Check : $\begin{cases} H(f(\mathbf{g}_j)) = h_j \\ \|\mathbf{g}_j\| \leq B \end{cases}$

```
$\Phi := \emptyset$
$\forall i \in [n]$ :
    find the first $j \in \mathcal{C}, j > max(\Phi)$
    s.t. $\mathbf{Rej}(\mathbf{x}_i + \mathbf{g}_j) = 1$
    $\mathbf{z}_i := \mathbf{x}_i + \mathbf{g}_j$
    $\Phi := \Phi \cup \{j\}$
If $|\Phi| < n$ abort
```

$\xrightarrow{\Phi, (\mathbf{z}_i)_{i \in [n]}}$

$\forall i \in [n]$ :

Check $\begin{cases} H(f(\mathbf{z}_i) - y_i) = h_{\Phi_i} \\ \|\mathbf{z}_i\| \leq B \end{cases}$

**Fig. 3.** Warm-up construction. An imperfect proof of knowledge with imperfection $k/\log \alpha + 1$ and communication that grows linearly with $\alpha$

**Computational Honest-Verifier Zero-Knowledge:** *There exists an expected PPT simulator $\mathcal{S}$ such that for any $(Y, \mathbf{X}) \in \mathcal{R}_{KSP}(n, f, \beta)$, and for any PPT algorithm $\mathcal{A}$. $\mathcal{A}$ has advantage* negl$(k)$ *in distinguishing between the two following distributions:*

- $View_{\mathcal{V}}[\mathcal{P}(Y, \mathbf{X}) \leftrightarrow \mathcal{V}(Y)]$ *the view of $\mathcal{V}$ consisting in the transcript of the protocol has well as the random coins of $\mathcal{V}$.*
- $\mathcal{S}(Y)$

[BDLN16] introduced a ZKPOK that uses an imperfect proof as a building block. The construction was later improved in [CD16] allowing for very efficient proofs that only require two executions of the imperfect proof system, while only introducing an additional soundness slack of $k$. The protocol, however, requires the amortization to be done on at least $4k^2$ secrets, which can be impractical. We give a somewhat refined statement of this construction as the proof of [CD16] can be straightforwardly adapted to using the imperfection $\tau(k)$ instead of k.

**Theorem 8 (Compiler [CD16] Theorem 2).** *Let $f$ be an ivOWF, let $k$ be a statistical security parameter, let $\mathcal{R}_{KSP}(n, f, \beta)$ and $\mathcal{R}_{KSP}(n, f, \beta')$ be two binary relations on $f$. Let $\mathcal{P}_{IProof}$ be an imperfect proof with imperfection $\tau(k)$. If $n \geq 4\tau(k)^2 + O(\log k)$ then there exists an efficient construction for a zero-knowledge proof of knowledge $\mathcal{P}_{CProof}$ with soundness slack $\tau(k)\beta'$.*

In this paper, we give constructions that can reduce the imperfection $\tau(k)$ of the imperfect proof to values less than $k$, thus allowing for more efficient zero-knowledge protocols in cases where the number of available equations is less than $\tau(k)$.

## 3 Warmup Construction

We present a first construction that achieves imperfection $\tau(k) = k/log(\alpha) + 1$ for any parameter $\alpha$, but has proof size that grows linearly in $\alpha$. This first construction is similar to the one of [BDLN16]. Their protocol works in two phases: first the prover samples masking parameters $\mathbf{g}_j, j \in [T]$ and a cut-and-choose protocol reveals each one with probability one half. After this step, the verifier is convinced that with probability $1 - 2^{-k}$ all but $k$ of them are well formed. In the second phase the masking parameters that were not revealed are used to hide the secrets of the prover. We modify the first phase of this protocol so that the prover reveals each masking parameter with probability $1 - 1/\alpha$, for $\alpha \geq 2$, this reduces the percentage of $\mathbf{g}_j$ on which the prover can cheat and, in turn, reduces the imperfection of the proof. However, the number of masking parameters necessary for the second phase is on the order of $n$, meaning that, since the prover will reveal a fraction $1 - 1/\alpha$ of them, the protocol then requires $T = \Theta(\alpha n)$ masking parameters.

We describe this protocol in Figure 3. We do not give a formal proof that it is an imperfect proof of knowledge with imperfection $k/\log\alpha + 1$ as the protocol presented in the next section is a strict improvement upon this one. While this first protocol achieves better imperfection than the one of [BDLN16], it has a major downside in that the communication cost grows linearly with $\alpha$, since we need $T \geq \alpha n$. This voids any improvement over the previous protocol. To remedy this problem we will modify this protocol as follows:

- Rather than sending the hash of every $a_i$ in the first round the prover will only send $h = H(h_1, \ldots, h_T)$, thus making the first flow of the protocol constant size.
- In his second move, the prover sends $\mathbf{g}_j, j \in \mathcal{O}$. This is an issue because $|\mathcal{O}| \simeq (\alpha - 1)4n$, but also because the $\mathbf{g}_i$ can be rather large. We solve these problems by sending a set of seeds from which a PRG will be used to derive the $\mathbf{g}_i$. This way only 256 bits need to be sent for each seed. Most crucially, by using a tree data-structure, we show that the prover only needs to send $4n\log\alpha$ seeds in his second move.

## 4 Amortized Proof for $f(\mathrm{x}_i) = y_i$ with Fewer Equations

In this section we describe our first concrete imperfect proof of knowledge and prove that it has imperfection $\tau(k) = k/\log\alpha + 1$. We show that the proof is only slightly dependent on $\alpha$ in Section 6.

We will need the following two functions, which can both be efficiently implemented using an extendable output function (e.g. SHAKE128 [BDPA16]) which will be modeled as a random oracle:

- **PRF** : $\{0,1\}^{256} \to \{0,1\}^{512}$ a size doubling pseudo-random function
- **PRG** : $\{0,1\}^{256} \to \{0,1\}^{*}$ a pseudo-random generator

For a randomized algorithm $h$ and a seed $s \in \{0,1\}^{256}$ we will write $h\,[\mathbf{PRG}(s)]$ to denote an execution of $h$ using as randomness the bits output by $\mathbf{PRG}(s)$.
We first describe the tree structure that we will use. From now on we will only consider $T = 2^t$ a power of two, which simplifies the description of the protocols and does not affect efficiency – all the results we obtain can be adapted to general T. A tree $\Gamma$ is a binary tree with nodes labeled in $\{0,1\}^*$ (the root will have the label $\emptyset$, its left child will have label 0, its right child will have label 1, etc...). We consider complete binary trees of depth t, which implies that the leaves will be labeled in $\{0,1\}^t$. We map the range $[T]$ to the labels of the leaves through the mapping where the image of $t \in [T]$ is the leaf labeled by the binary decomposition of $t - 1$. Each node will have two extra attributes, one will be the seed associated to the node (which can be bottom for the verifier since he will not know all the seeds), the other will be a bit indicating whether the associated seed must be sent to the verifier in the first flow.
The purpose of this seed tree is twofold. We will use the leaves as seeds for the **PRG** when generating the $\mathbf{g}_j, j \in [T]$. This way sending the seeds to the verifier in the first flow will be sufficient as he can then reconstruct the $\mathbf{g}_j, j \in \mathcal{O}$ using the **PRG**. More importantly, rather than directly sending the leaves of the seed tree, it will be more efficient to send the smallest set of nodes needed to recover the leaves for indices that lie in $\mathcal{O}$. We define the tree structure as follows:

**Tree T:**

- Label $\subset \{0,1\}^*$
- Left $\in$ **Tree** $\cup \perp$
- Right $\in$ **Tree** $\cup \perp$
- Leaf $\in \{0,1\}$
- Sel $\in \{0,1\}$
- Seed $\in \{0,1\}^{256} \cup \perp$

```
$\underline{\mathcal{P}}$                                                                $\underline{\mathcal{V}}$

$\mathbf{X} := (\mathbf{x}_1, \dots, \mathbf{x}_n)$                                    $\mathbf{Y}$
$\mathbf{Y} := (y_1 := f(\mathbf{x}_1), \dots, y_n := f(\mathbf{x}_n))$
--------------------------------------------------------------------------------------------------
$\textbf{Tree } \mathit{\Gamma}_{\mathcal{P}} := \textbf{new Tree}$                     $\textbf{Tree } \mathit{\Gamma}_{\mathcal{V}} := \textbf{new Tree}$
$\textbf{Initialize}(\mathit{\Gamma}_{\mathcal{P}}, \emptyset, t)$                      $\textbf{Initialize}(\mathit{\Gamma}_{\mathcal{V}}, \emptyset, t)$
$s \xleftarrow{\$} \{0,1\}^{256}$
$\textbf{SeedTree}(\mathit{\Gamma}_{\mathcal{P}}, s)$
$\forall i \in [T] : \ \mathbf{g}_i \leftarrow \mathcal{D}_\sigma^r\left[\textbf{PRG}(s_i)\right]$
$a_1, \dots, a_T := f(\mathbf{g}_1), \dots, f(\mathbf{g}_t)$
$h_1, \dots, h_T := H(a_1), \dots, H(a_T)$
$h := H(h_1, \dots, h_T)$                     $\xrightarrow{h}$
                                              $\xleftarrow{\mathbf{c}}$       $\mathbf{c} \leftarrow \chi^T \in \{0,1\}^T$
$\mathcal{O} := \{j, \mathbf{c}[j] = 0\}, \mathcal{C} := [T] \setminus \mathcal{O}$
$\textbf{Prefix}(\mathit{\Gamma}_{\mathcal{P}}, \mathcal{O})$
$\mathcal{S} := \left\{j \subset \{0,1\}^t,\ \mathit{\Gamma}_{\mathcal{P}}[j].Sel = 1\right\}$    $\xrightarrow[(h_j)_{j \in \mathcal{C}}]{(s_j)_{j \in \mathcal{S}}}$
                                                              $\mathcal{O} := \{j, \mathbf{c}[j] = 0\}, \mathcal{C} := [T] \setminus \mathcal{O}$
                                                              $\textbf{Prefix}(\mathit{\Gamma}_{\mathcal{V}}, \mathcal{O})$
                                                              $\textbf{Reconstruct}(\mathit{\Gamma}_{\mathcal{V}}, (s_j)_{j \in \mathcal{S}}, \mathcal{O})$
                                                              $(s_j)_{j \in \mathcal{O}} := (\mathit{\Gamma}_{\mathcal{V}}[j].Seed)_{j \in \mathcal{O}}$
                                                              $\forall j \in \mathcal{O} \begin{cases} \mathbf{g}_j := f(\mathcal{D}_\sigma^r\left[\textbf{PRG}(s_j)\right]) \\ h_j := H(\mathbf{g}_j) \\ \text{Check: } \|\mathbf{g}_j\| \leq B \end{cases}$
                                                              $Check : H(h_1, \dots, h_T) = h$
$\mathit{\Phi} := \emptyset$
$\forall i \in [n] :$
    find the first $j \in \mathcal{C}, j > max(\mathit{\Phi})$
    s.t. $\textbf{Rej}(\mathbf{x}_i + \mathbf{g}_j) = 1$
    $\mathbf{z}_i := \mathbf{x}_i + \mathbf{g}_j$
    $\mathit{\Phi} := \mathit{\Phi} \cup \{j\}$
If $|\mathit{\Phi}| < n$ abort                $\xrightarrow{\mathit{\Phi}, (\mathbf{z}_i)_{i \in [n]}}$
                                                              $\forall i \in [n] :$
                                                              Check $\begin{cases} H(f(\mathbf{z}_i) - y_i) = h_{\mathit{\Phi}_i} \\ \|\mathbf{z}_i\| \leq B \end{cases}$
```

**Fig. 4.** Our first construction: an imperfect proof of knowledge with imperfection $\frac{k}{\log \alpha} + 1$. The communication complexity only has a small dependence on $\log \alpha$.

For $j \in \{0,1\}^*$ we denote by $\Gamma[j]$ the node with label $j$. We will describe four algorithms: the first to initialize the tree will be performed by both parties, the second to initialize the seeds will only be used by the prover, the third to compute the indexes of the seeds that will be sent in the first flow of the protocol will be used by both parties, and the fourth to recover the seeds needed to compute the $\mathbf{g}_j, j \in \mathcal{O}$ will only be used by the verifier.

**Algorithm 1 Initialize**$(\Gamma, l, d)$

```
Require: A tree $\Gamma$, a label $l \subset \{0,1\}^t$, a depth $d$
 1: $\Gamma.Label := l$
 2: $\Gamma.Sel := 0$
 3: $\Gamma.Seed := \perp$
 4: if $d = 0$ then
 5:     $Self.Leaf := 1$
 6:     $Self.Left := \perp$
 7:     $Self.Right := \perp$
 8: else
 9:     $Self.Leaf := 0$
10:     Initialise$(\Gamma.Left, (l,0), d-1)$
11:     Initialise$(\Gamma.Right, (l,1), d-1)$
12: end if
```

The second algorithm will use a seed fixed by the prover and compute the seed associated with the children of each node as the first and second half of **PRF** applied on the seed of the parent node.

**Algorithm 2 SeedTree**$(\Gamma, v)$

```
Require: A tree $\Gamma$, $v \in \{0,1\}^{256}$
 1: $\Gamma.Seed := v$
 2: if $\Gamma.Leaf = 0$ then
 3:     $(v_1, v_2) := \mathbf{PRF}(v)$
 4:     SeedTree$(\Gamma.left, v_1)$
 5:     SeedTree$(\Gamma.right, v_2)$
 6: end if
```

The **Prefix** algorithm will compute the prefix of a set of nodes and set their attribute *Sel* to 1. A node $n$ will be in the prefix of a set $\mathcal{O}$ if all the leaves that descend from $n$ are in $\mathcal{O}$ and none of the ancestors of $n$ are in the prefix of $\mathcal{O}$. The algorithm ensues directly from this definition.

**Algorithm 3 Prefix**$(\Gamma, \mathcal{O})$

```
Require: A tree $\Gamma$, a set of indices $\mathcal{O} \subset [T]$
 1: if $\Gamma.Leaf = 1 \wedge \Gamma.label \in \mathcal{O}$ then
 2:     $\Gamma.Sel := 1$
 3:     return 1
 4: else if $\Gamma.Leaf = 0 \wedge$ Prefix$(\Gamma.Left, \mathcal{O}) = 1 \wedge$ Prefix$(\Gamma.Right, \mathcal{O}) = 1$ then
 5:     $\Gamma.Sel := 1$
 6:     $\Gamma.Left.Sel := 0$
 7:     $\Gamma.Right.Sel := 0$
 8:     return 1
 9: end if
10: return 0
```

The **Reconstruct** algorithm will use a tree in which the prefix $\mathcal{S}$ of $\mathcal{O}$ has been computed as well as a set of seeds $s_j, j \in \mathcal{S}$ and will reconstruct the seeds $s_j, j \in \mathcal{O}$ by using **SeedTree** for each node in $\mathcal{S}$.

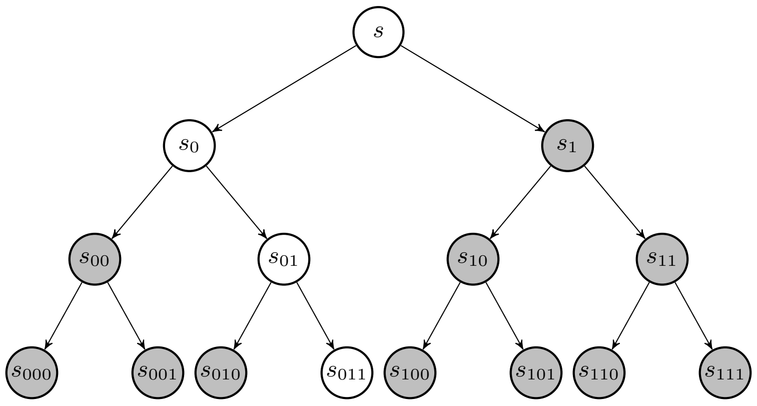


**Fig. 5.** Seed tree for $t = 3$, and $\mathcal{O} = \{1, 2, 3, 5, 6, 7, 8\}$, the nodes needed to reconstruct $(s_j)_{j\in\mathcal{O}}$ are those in $prefix(\mathcal{O}) = \{00, 010, 1\}$.

**Algorithm 4 Reconstruct**$(\Gamma, S, \mathcal{O})$

```
Require: A tree $\Gamma$, a list of seeds $S = [s_j]$, a set $\mathcal{O} \subset [T]$. We assume that Prefix$(\Gamma, \mathcal{O})$ was applied.
1: if $\Gamma.Sel = 1$ then
2:     SeedTree$(\Gamma, S[0])$
3:     $S := S[1:]$
4: else
5:     Reconstruct$(\Gamma.Left, S)$
6:     Reconstruct$(\Gamma.Right, S)$
7: end if
```

We give in Figure 5 an example of a seed tree as well as a set $\mathcal{O}$ and its prefix.
We describe our improved protocol in Figure 4. We will first prove its correctness.

**Theorem 9 (Correctness).** *Let $f$ be an ivOWF, $k$ be a statistical security parameter, $H$ a collision resistant hash function, $r \geq 128$ be an integer, $\chi$ the bernouilli distribution of parameter $1 - 1/\alpha$ (i.e. $P[\chi = 0] = 1 - P[\chi = 1] = 1/\alpha$). Let $T = 5\alpha n$, $\sigma = 11\beta$, $B = 2\sigma\sqrt{r}$.*
*For honest $\mathcal{P}$, $\mathcal{V}$, and for $(Y, \mathbf{X}) \in \mathcal{R}_{KSP}(n, f, \beta)$ , the protocol $\mathcal{P}_{IProof}$ given in Figure 4 completes with probability greater than $1 - 2^{-100}$.*

*Proof.* By the homomorphic property of $f$ and by construction of **Initialize**, **SeedTree**, **Prefix**, and **Reconstruct** all the checked equalities hold. We fist consider the probability that $\mathcal{P}$ aborts. $\mathcal{P}$ will abort if he runs out of samples during the rejection sampling. For each $\mathbf{g}_j, j \in [T]$ the probability that $\mathbf{g}_j$ will not be revealed is $1/\alpha$, and by Theorem 5 the probability that the rejection sampling will succeed is $1/3$, in which case the vector obtained will be of norm less than $B$ with overwhelming probability (4). We can model the probability that each $g_j$ will not be revealed and will pass both checks of the rejection step by a Bernoulli variable $X_j$ s.t $\Pr[X_j = 1] = 1/(3\alpha) - 2^{-O(n)}$. $\mathcal{P}$ will abort if $\sum_{j\in[T]} X_j < n$. Using the Chernoff bound we obtain:

$$\Pr\left[\sum_{j\in[T]} X_j < n\right] \leq \exp\left(-\frac{(T - 3\alpha n)^2}{3\alpha T} + 2^{-O(n)}\right)$$
$$= \exp\left(-\frac{4n}{15} + 2^{-O(n)}\right),$$

which is negligible asymptotically (and in practice less than $2^{-100}$ whenever we amortize over $n \geq 260$ secrets.)
We now consider the probability that $\mathcal{V}$ aborts. $\mathcal{V}$ will abort if there exists either $j \in \mathcal{O}$ such that $\|\mathbf{g}_j\| > B$ or $i \in [n]$ such that $\|\mathbf{z}_i\| > B$. Since the $\mathbf{g}_j$ and the $\mathbf{z}_i$ are drawn independently from the

distribution $\mathcal{D}_\sigma^r$ by using a union bound we have that the probability that the norm of one of them exceeds $B$ is less than $(T+n)2^{-r}$. □

We now show that this protocol is honest-verifier zero-knowledge.

**Theorem 10 (HVZK).** *With parameters set as in Theorem 9, the protocol $\mathcal{P}_{IProof}$ described in Figure 4 is computationally honest-verifier zero-knowledge.*

*Proof.* The honest-verifier zero-knowledge proof is very close to that of [BDLN16], but we still include it here for completeness as there are slight differences. Consider the following algorithm $\mathcal{S}_{IProof}$:

– On input $(\mathbf{Y} = (y_1, \ldots, y_n), \beta)$ sample $s \xleftarrow{\$} \{0,1\}^{256}$ and $(s_j)_{j \subset \{0,1\}^t}$ using **SeedTree**.
– Sample $\mathbf{c} \leftarrow \chi^T$, compute the sets $\mathcal{O}$ and $\mathcal{C}$.
– Set $\Phi' = \emptyset$, for $j \in \mathcal{C}$ sample $\mathbf{z}_j \leftarrow \mathcal{D}_\sigma^r$ and do the following:
  - Sample $b \xleftarrow{\$} \{0,1,2\}$
  - If $b = 0 \wedge \|\mathbf{z}_j\| \leq B$ then $\Phi' = \Phi' \cup j$
– For $j \in \mathcal{O}$ set $h_j = H(f(\mathcal{D}_\sigma^r[\mathbf{PRG}(s_j)]))$
– If $|\Phi'| < n$ then for $j \in \mathcal{C}$ set $h_j \xleftarrow{\$} \{0,1\}^{256}$, $h = H(h_1, \ldots, h_T)$, output $(h, \mathbf{c}, (s_j)_{j \in prefix(\mathcal{O})}, (h_j)_{j \in \mathcal{C}})$ and abort.
– If $|\Phi'| \geq n$ set $\Phi$ to be the first n elements of $\Phi'$ and for $i \in [n]$ rename $\mathbf{z}_{\Phi[i]}$ as $\mathbf{z}_i$. For $j \in \mathcal{C} \setminus \Phi$ set $h_j \xleftarrow{\$} \{0,1\}^{256}$.
– For $i \in [n]$ set $a_{\Phi[i]} = f(\mathbf{z}_i) - y_i$, $h_{\Phi[i]} = H(a_{\Phi[i]})$.
– Set $h = H(h_1, \ldots, h_T)$, output

$$(h, \mathbf{c}, (s_j)_{j \in prefix(\mathcal{O})}, (h_j)_{j \in \mathcal{C}}, \Phi, (\mathbf{z}_{\Phi[i]})_{i \in [n]})$$

We first consider the abort probability of the simulator: $\mathcal{S}$ will abort if $|\Phi'| < n$. For each $j \in [T]$ the simulator adds a $\mathbf{z}_j$ to $\Phi'$ iff $\mathbf{c}[j] = 1 \wedge b = 0 \wedge \|\mathbf{z}_j\| \leq B$, the probability of this event is $1/(3\alpha) - 2^{-O(n)}$, thus the probability of abort will be exponentially close to the one of $\mathcal{P}_{IProof}$. Regardless of whether the simulator aborts or not, all the checks performed by the verifier will accept: $h$ is set to be $h = H(h_1, \ldots, h_T)$, and when $\mathcal{S}$ does not abort he sets $h_{\Phi[i]}$ so that $H(f(\mathbf{z}_i) - y_i) = h_{\Phi[i]}$. The sets $\mathcal{O}$ and $\mathcal{C}$ are defined in the same way as in $\mathcal{P}_{IProof}$ and the $s_j, j \in prefix(\mathcal{O})$ are also sampled according to the protocol. Note that in $\mathcal{P}_{IProof}$ for $j \in \mathcal{C} \setminus \Phi$ the $h_j$ are distributed uniformly since $H$ is modeled as a random oracle and no preimages of the $h_j$ are given (note that for some leaves of the tree the verifier knows half of the output of **PRF** on the parent node, even conditioning on this knowledge the second half of the output is uniform as **PRF** is modeled as a random oracle). It remains to analyze the distribution of $\mathbf{z}_i$ for $i \in [n]$. We have by Theorem 5 that the distribution of $\mathbf{z}_i, i \in [n]$ in $\mathcal{P}_{IProof}$ is that of a discrete gaussian centered in 0 with standard deviation $\sigma$ and thus identical to the distribution of $\mathbf{z}_i$ in $\mathcal{S}$. □

We finally show the soundness of the protocol, i.e. that one can extract all but $\tau(k) = k/\log\alpha + 1$ preimages from a prover that succeeds with probability greater than $2^{-k}$

**Theorem 11 (Soundness).** *With parameters set as per Theorem 9, the protocol $\mathcal{P}_{IProof}$ has imperfection $\tau(k) = k/\log\alpha + 1$ and slack $2B$.*

*Proof.* Let $k' = k/\log\alpha + 1$, let $\hat{\mathcal{P}}$ be a deterministic prover that makes an honest verifier accept with probability $p > 2^{-k}$. We will construct an extractor $\mathcal{E}$ that extracts $n - k'$ values $\mathbf{x}'_i, i \in I \subset [n]$ such that $f(\mathbf{x}'_i) = y_i$ and $\|\mathbf{x}'_i\| \leq 2B$. $\mathcal{E}$ will run in expected time $poly(s,k) \cdot 1/p$ where s is the size of the input to $\mathcal{P}_{IProof}$.
$\mathcal{E}$ starts $\hat{\mathcal{P}}$ who outputs $h$ and runs the protocol on random challenges until he outputs $(s_j)_{j \in prefix(\mathcal{O})}$ and $(h_j)_{j \in \mathcal{C}}$, from this $\mathcal{E}$ can recover hashes $(h_j)_{j \in [T]}$ such that $H(h_1, \ldots, h_T) = h$, fix $\overline{h} := h$ and $\overline{h}_j := h_j$. Set $A := \emptyset$ and run $T$ instances of $\hat{\mathcal{P}}$ in parallel, which we denote $\hat{\mathcal{P}}_1, \ldots, \hat{\mathcal{P}}_T$. Do the following until $|A| \geq T - k'$:

– For each $\hat{\mathcal{P}}_j$ sample a random challenge $\mathbf{c}_j \leftarrow \chi^T$ subject to $\mathbf{c}_j[j] = 0$ and run $\hat{\mathcal{P}}_j$ on challenge $\mathbf{c}_j$.

– For each instance $\hat{\mathcal{P}}_j$ that does not abort, reconstruct $s_j$ from the prover's response and set $\mathbf{g}_j = \mathcal{D}_\sigma^r\left[\mathbf{PRG}(s_j)\right]$. Verify the proof output by $\hat{\mathcal{P}}_j$ and set $A = A \cup \mathbf{g}_j$. Note that if the proof is valid then the verifier can reconstruct $h_1, \ldots, h_T$ s.t

$$H(h_1, \ldots, h_T) = \overline{h} = H(\overline{h}_1, \ldots, \overline{h}_T)$$

since $H$ is collision resistant we have in particular that $h_j = \overline{h}_j$ which implies $H(f(\mathbf{g}_j)) = \overline{h}_j$. We also have $\|\mathbf{g}_j\| \leq B$.

Observe that if this algorithm terminates we obtain a set $A$ of at least $T - k'$ preimages of the $\overline{h}_j$ by the function $H \circ f$. We will now show that this extractor finishes in expected polynomial time. This proof is very similar to the one of [BDLN16] but we choose to present it anyway as it will be reused in the next section.

Let $p_j$ be the probability that $\hat{\mathcal{P}}_j$ outputs a good $\mathbf{g}_j$ (i.e such that $H(f(\mathbf{g}_j)) = \overline{h}_j \wedge \|\mathbf{g}_j\| \leq B$). We say that $p_j$ is bad if $p_j < p/k'$ and good otherwise. Let $X_j$ be the event that $\hat{\mathcal{P}}_j$ eventually outputs a good $\mathbf{g}_j$, where $X_j = 1$ if the event happens and $X_j = 0$ otherwise. If $p_j$ is good then after $l$ iterations:

$$\Pr\left[X_j = 0\right] \leq \left(1 - p/k'\right)^l \leq e^{-lp/k'}$$

so after at most $l = k \cdot k'/p$ iterations we can expect that $\mathbf{g}_j$ was extracted except with probability negligible in k. This can be generalized to the success of all $\hat{\mathcal{P}}_j$ (where $p_j$ is good) by a union bound, and the probability of failing is still negligible because $T$ is polynomial in k. The resulting extractor thus runs in time $O(Tk^2/p \log \alpha)$ provided there are less than $k'$ bad $p_j$.
Assume there are $k'$ bad $p_j$ which, for simplicity, are $p_1, \ldots, p_{k'}$. In the protocol the challenge is taken according to the distribution $\chi^T$. The success probability of $\hat{\mathcal{P}}$ can be conditioned on the value of $\mathbf{c}[1]$ as

$$\begin{aligned} p &= \Pr\left[\hat{\mathcal{P}} \text{ succeeds}\right] \\ &= \Pr\left[\hat{\mathcal{P}} \text{ succeeds} \middle| \mathbf{c}[1] = 0\right] \cdot \Pr\left[\mathbf{c}[1] = 0\right] \\ &\quad + \Pr\left[\hat{\mathcal{P}} \text{ succeeds} \middle| \mathbf{c}[1] = 1\right] \cdot \Pr\left[\mathbf{c}[1] = 1\right] \\ &= p_1\left(1 - \frac{1}{\alpha}\right) + \frac{1}{\alpha}\Pr\left[\hat{\mathcal{P}} \text{ succeeds} \middle| \mathbf{c}[1] = 1\right] \end{aligned}$$

Conditioning additionally on $\mathbf{c}[2]$ yields

$$\begin{aligned} p &\leq p_1\left(1 - \frac{1}{\alpha}\right) + \frac{1}{\alpha}\left(\left(1 - \frac{1}{\alpha}\right)\alpha p_2 + 1/\alpha \Pr\left[\hat{\mathcal{P}} \text{ succeeds} \middle| \mathbf{c}[1] = 1 \wedge \mathbf{c}[2] = 1\right]\right) \\ &= \left(1 - \frac{1}{\alpha}\right)(p_1 + p_2) + \frac{1}{\alpha^2}\Pr\left[\hat{\mathcal{P}} \text{ succeeds} \middle| \mathbf{c}[1] = 1 \wedge \mathbf{c}[2] = 1\right] \end{aligned}$$

The reason the inequality holds is as follows: the probability that a random challenge s.t $\mathbf{c}[2]$ will yield a preimage of $\overline{h}_2$ is $p_2$. Now conditioning on $\mathbf{c}[1] = 1$, which occurs with probability $1/\alpha$, will increase that probability from $p_2$ to at most $\alpha p_2$.

Repeating the above argument generalizes to

$$\begin{aligned} p &\leq \left(1 - \frac{1}{\alpha}\right)(p_1 + p_2 + \ldots + p_{k'}) \\ &\quad + \frac{1}{\alpha^{k'}}\Pr\left[\hat{\mathcal{P}} \text{ succeeds} \middle| \mathbf{c}[1] = 1 \wedge \ldots \wedge \mathbf{c}[k'] = 1\right] \\ &< \left(1 - \frac{1}{\alpha}\right)p + \frac{1}{\alpha^{k'}} \end{aligned}$$

This entails that

$$p < \frac{1}{\alpha^{k'-1}} = \alpha^{-k/log\alpha} = 2^{-k}$$

From this we conclude that there are less than k' bad $p_j$, and thus that $\mathcal{E}$ has extracted a set $A$ of size at least $T - k'$ of elements $\mathbf{g}_j$ s.t

$$H(f(\mathbf{g}_j)) = \overline{h}_j \wedge \|\mathbf{g_j}\| \leq B.$$

We will now show how to use this set $A$ to extract $n - k'$ secrets $\mathbf{x}'_i$.
$\mathcal{E}$ runs regular instances of $\hat{\mathcal{P}}$ until one of them succeeds call this instance $\widetilde{\mathcal{P}}$, this takes expected time $1/p$. From the output of $\widetilde{\mathcal{P}}$ $\mathcal{E}$ obtains a set $\widetilde{\Phi}$ as well as $(\widetilde{\mathbf{z}}_i)_{i\in[n]}$ s.t $H(f(\widetilde{\mathbf{z}}_i) - y_i) = \overline{h}_{\widetilde{\Phi}[i]}$ (by collision resistance of $H$ and by the fact that $H(\widetilde{h}_1, \ldots, \widetilde{h}_T) = \overline{h}$) and $\|\widetilde{\mathbf{z}}_i\| \leq B$. For each $i \in [n]$ if there exists $\mathbf{g}_{\widetilde{\Phi}[i]} \in A$, then we have $H(f(\widetilde{\mathbf{z}}_i) - y_i) = H(f(\mathbf{g}_{\widetilde{\Phi}[i]}))$, setting $\mathbf{x}'_i = \widetilde{\mathbf{z}}_i - \mathbf{g}_{\widetilde{\Phi}[i]}$ gives $f(\mathbf{x}'_i) = y_i$ and $\|\mathbf{x}'_i\| \leq 2B$. Since $|A| \geq T - k'$ there are at most $k'$ of the $\widetilde{\Phi}[i]$ that are not in this set and $\mathcal{E}$ can extract $n - k'$ preimages $\mathbf{x}'_i$. □

Using this imperfect proof with the compiler of Theorem 8 results in a proof of knowledge with soundness slack $4k\sqrt{r}\beta/\log\alpha$, communication overhead $O(1)$ (we will discuss this in further details in Section 6) and amortization over $4\left(\frac{k}{\log\alpha} + 1\right)^2$ secrets. e.g. for $\alpha = 2^{10}$ one can create amortized proofs for as few as 853 secrets with a security parameter $k = 128$, while the construction of [CD16] needs to amortize over at least 67103 secrets for the same security. However this protocol is not strictly better in the sense that the computation cost, which is essentially the number of evaluations of the function $f$, increases multiplicatively in $\alpha$ for both the prover and the verifier, making this protocol impractical for very large $\alpha$. In the next section we describe a new variant of the scheme inspired by the work of [BCK⁺14] that reduces the soundness errror $\tau(k)$ without necessarily increasing the computational cost of the protocol.

## 5 Proving $f(\mathbf{x}_i) = 2y_i$ with Even Fewer Equations

In this section we use an idea from the zero-knowledge proof of [BCK⁺14] to improve the imperfection of our previous scheme. In [BCK⁺14] the authors prove knowledge of preimages for an ivOWF over a polynomial ring of dimension $d$, they take advantage of this structure by replacing the binary challenge of the classic 3-round ZKPOK with a challenge in $\{0, \pm1, \pm X, \ldots, \pm X^{d-1}\}$ this improves the soundness error of the protocol from 1/2 to $1/(2d+1)$. We adapt this technique to further improve the imperfection of our imperfect proof. The knowledge extractor becomes however substantially more complicated.

Let $\mathcal{R}$ be the polynomial ring $\mathbb{Z}[X]/\langle X^d + 1\rangle$. For $(a_1, \ldots, a_l) \in \mathcal{R}^l$ and for $b \in \mathcal{R}$ let $\star$ be the following product $\star : \mathcal{R} \times \mathcal{R}^l \to \mathcal{R}^l$ such that $b \star (a_1, \ldots, a_l) = (ba_1, \ldots, ba_l)$.
In this section we will consider ivOWFs $f : \mathbb{Z}^r \simeq \mathcal{R}^l \to \mathcal{R}$ such that for $b \in \mathcal{R}$ and $a \in \mathcal{R}^l$ we have $f(b \star a) = bf(a)$. This type of one-way function is often used in ideal-lattice constructions.

**Lemma 12 ( [BCK⁺14] Lemma 3.2).** *Let $d$ be a power of 2, let $a, b \in \{\pm1, \ldots, \pm X^{d-1}\}$. Then $2(a-b)^{-1} \mod X^d+1$ only has coefficients in $\{-1, 0, 1\}$. In particular $\|2(a-b)^{-1}\| \leq \sqrt{d}$.*

We now prove that the construction of Figure 6 is an imperfect proof of knowledge.

**Theorem 13.** *Let $f : \mathcal{R}^l \to \mathcal{R}$ be an ivOWF, $r = ld \geq 128$ be an integer, let $f' = 2f$, let $k$ be a statistical security parameter, $H$ a collision resistant hash function, $\chi$ a distribution over $\{0, \pm1, \pm X, \ldots, \pm X^{d-1}\}$ with $\Pr[\chi = 0] = 1 - 1/\alpha$ and $\forall c \in \{\pm1, \ldots, \pm X^{d-1}\}$, $\Pr[\chi = c] = 1/(2d\alpha)$. Let $T = 5\alpha n$, $\sigma = 11\beta$, $B = 2\sqrt{r}\sigma$. The protocol $\mathcal{P}_{IProof}$ given in Figure 6 is an imperfect proof of knowledge for the relations $\mathcal{R}_{KSP}(f, n, \beta)$ and $\mathcal{R}_{KSP}(f', n, \sqrt{d}B)$ with imperfection $\frac{k(1+1/\log\alpha)}{\log\alpha + \log 2d} + 1$*

*Proof.* The proofs for the correctness and zero-knowledge of the protocol are identical to the proofs in the previous section. On the other hand the soundness proof is more involved.

**Soundness:** Let $k' = \frac{k(1+1/\log\alpha)}{\log\alpha + \log 2d} + 1$, let $\hat{\mathcal{P}}$ be a deterministic prover that makes an honest verifier accept with probability $p > 2^{-k}$. We will construct an extractor $\mathcal{E}$ that extracts $n - k'$ values $\mathbf{x}'_i, i \in I \subset [n]$ such that $f(\mathbf{x}'_i) = 2y_i$ and $\|\mathbf{x}'_i\| \leq \sqrt{d}B$. $\mathcal{E}$ will run in time $poly(s, k) \cdot 1/p^{1+2/\log\alpha}$ where s is the size of the input to $\mathcal{P}_{IProof}$.

```
$\underline{\mathcal{P}}$                                                   $\underline{\mathcal{V}}$

$\mathbf{X} := (\mathbf{x}_1, \ldots, \mathbf{x}_n)$                        $\mathbf{Y}$
$\mathbf{Y} := (y_1 := f(\mathbf{x}_1), \ldots, y_n := f(\mathbf{x}_n))$
------------------------------------------------------------------------------------------
$\textbf{Tree } \Gamma_{\mathcal{P}} := \textbf{new Tree}$                  $\textbf{Tree } \Gamma_{\mathcal{V}} := \textbf{new Tree}$
$\textbf{Initialize}(\Gamma_{\mathcal{P}}, \emptyset, t)$                   $\textbf{Initialize}(\Gamma_{\mathcal{V}}, \emptyset, t)$
$s \xleftarrow{\$} \{0,1\}^{256}$
$\textbf{SeedTree}(\Gamma_{\mathcal{P}}, s)$
$\forall i \in [T]: \ \mathbf{g}_i \leftarrow \mathcal{D}_{\sigma}^{r}[\mathbf{PRG}(s_i)]$
$a_1, \ldots, a_T := f(\mathbf{g}_1), \ldots, f(\mathbf{g}_t)$
$h_1, \ldots, h_T := H(a_1), \ldots, H(a_T)$
$h := H(h_1, \ldots, h_T)$                   $\xrightarrow{h}$
                                             $\xleftarrow{\mathbf{c}}$       $\mathbf{c} \leftarrow \chi^T \in \left\{0, \pm 1, \ldots, \pm X^{d-1}\right\}^T$
$\mathcal{O} := \{j, \mathbf{c}[j] = 0\}, \mathcal{C} := [T] \setminus \mathcal{O}$
$\textbf{Prefix}(\Gamma_{\mathcal{P}}, \mathcal{O})$
$\mathcal{S} := \left\{j \subset \{0,1\}^t, \ \Gamma_{\mathcal{P}}[j].Sel = 1\right\}$   $\xrightarrow[(h_j)_{j \in \mathcal{C}}]{(s_j)_{j \in \mathcal{S}}}$
                                                                            $\mathcal{O} := \{j, \mathbf{c}_j = 0\}, \mathcal{C} := [T] \setminus \mathcal{O}$
                                                                            $\textbf{Prefix}(\Gamma_{\mathcal{V}}, \mathcal{O})$
                                                                            $\textbf{Reconstruct}(\Gamma_{\mathcal{V}}, (s_j)_{j \in \mathcal{S}}, \mathcal{O})$
                                                                            $(s_j)_{j \in \mathcal{O}} := (\Gamma_{\mathcal{V}}[j].Seed)_{j \in \mathcal{O}}$
                                                                            $\forall j \in \mathcal{O} \begin{cases} \mathbf{g}_j := f(\mathcal{D}_{\sigma}^{r}[\mathbf{PRG}(s_j)]) \\ h_j := H(\mathbf{g}_j) \\ \text{Check: } \|\mathbf{g}_j\| \leq B \end{cases}$
                                                                            $Check: H(h_1, \ldots, h_T) = h$
$\Phi := \emptyset$
$\forall i \in [n]:$
    find the first $j \in \mathcal{C}, j > max(\Phi)$
    s.t. $\mathbf{Rej}(\mathbf{c}[j] \star \mathbf{x}_i + \mathbf{g}_j) = 1$
    $\mathbf{z}_i := \mathbf{c}[j] \star \mathbf{x}_i + \mathbf{g}_j$
    $\Phi := \Phi \cup \{j\}$
If $|\Phi| < n$ abort                        $\xrightarrow{\Phi, (\mathbf{z}_i)_{i \in [n]}}$
                                                                            $\forall i < i' \in [n]$, Check : $\Phi_i < \Phi_{i'}$
                                                                            $\forall i \in [n]:$
                                                                            Check : $\begin{cases} H(f(\mathbf{z}_i) - \mathbf{c}[\Phi_i] y_i) = h_{\Phi_i} \\ \|\mathbf{z}_i\| \leq B \end{cases}$
```

**Fig. 6.** Our second construction: an imperfect proof of knowledge for $f(\mathbf{x}_i) = 2y_i$ with imperfection $\frac{k(1+1/\log\alpha)}{\log\alpha + \log 2d} + 1$

We first use the same extractor as in the proof of Theorem 11. We note that even though the scheme is different, the same extractor applies with the only difference being that the equation verified by the extracted $\mathbf{x}'_i$ will be of the form $f(\mathbf{x}'_i) = bX^a y_i$ for some $b \in \{-1, 1\}, a \in [d]$, which directly gives $f(-bX^{d-a}\mathbf{x}'_i) = y_i$, since this new pre-image has the same norm we can rename it and obtain the same result. We thus obtain the following:

- $\overline{h}$ the hash sent by $\hat{\mathcal{P}}$ on his first flow.
- $\overline{h}_1, \ldots, \overline{h}_T$ such that $H(\overline{h}_1, \ldots, \overline{h}_T) = \overline{h}$
- A set $A$ of at least $T - k/\log\alpha - 1$ vectors $\mathbf{g}'_j$ such that $H(f(\mathbf{g}'_j)) = \overline{h}_j$, we define $\Psi \subset [T]$ to be the indices of the $\overline{h}_j$ for which a preimage was not extracted.
- A set $S$ of at least $n - k/\log\alpha - 1$ vectors $\mathbf{x}'_i$ such that $f(\mathbf{x}'_i) = y_i$, we define $\Upsilon \subset [n]$ to be the indices of the $y_i$ for which a preimage was not extracted.

By construction of this extractor we have $|\Upsilon| \leq |\Psi| \leq k/\log\alpha + 1$.
Observe that, on a successful run of $\hat{\mathcal{P}}$, the set $\Phi$ is a strictly increasing mapping from $[n]$ to $[T]$ (this is explicitely checked by the verifier) in the previous protocol this was used to show zero-knowledge, as reusing randomness could leak information, but this is now crucial for soundness. We also note that since $\Phi$ is a function from $[n]$ to $[T]$ we have either:

$(A)$ $\Phi(\Upsilon) \subset \Psi$
$(B)$ Or $\exists i \in \Upsilon$ s.t. $\Phi[i] \notin \Psi$

If on a run $\hat{\mathcal{P}}$ is successful and $(B)$ occurs then there exist $i, j \in [n] \times [T]$ such that $H(f(\mathbf{z}_i) - \mathbf{c}[j]\, y_i) = h_j$ and $j = \Phi[i] \notin \Psi$. As we have already extracted $\mathbf{g}'_j$ with $H(f(\mathbf{g}'_j)) = \overline{h}_j = h_j$ we obtain that $\mathbf{x}'_i = \mathbf{c}[j]^{-1}(\mathbf{z}_i - \mathbf{g}'_j)$ is a preimage of $y_i$. We can thus redefine the set $\Upsilon$ to be $\Upsilon := \Upsilon \setminus i$. Suppose that on a successful run of $\hat{\mathcal{P}}$, $(B)$ occurs with probability greater than $1/2$. The extractor can then run $\hat{\mathcal{P}}$ $O(2/p)$ times, successfully extract a new preimage of the $y_i$ and reduce the size of $\Upsilon$ by 1. After repeating this procedure $O(k)$ times we have either that $|\Upsilon| < k'$, in which case the extractor is done, or that $(B)$ occurs with probability strictly lower than $1/2$ on a successful run, for the rest of the proof we assume the latter. Since either $(A)$ or $(B)$ occurs on a successful run this implies that $(A)$ happens with probability strictly greater than $1/2$.
On any run where $(A)$ occurs, $\Phi$ induces a strictly increasing mapping from $\Upsilon$ to $\Psi$, let $\mathcal{G}$ be the set of all such mappings, we have

$$|\mathcal{G}| = \binom{|\Psi|}{|\Upsilon|} \leq 2^{|\Psi|} \leq 2^{k/\log\alpha+1}.$$

The extractor runs $|\mathcal{G}|$ parallel instances of $\hat{\mathcal{P}}$ denoted as $\hat{\mathcal{P}}^g, g \in \mathcal{G}$. and does the following until $|S| \geq n - k'$.

- Run instance $\hat{\mathcal{P}}^g$ with fresh randomness until it succeeds, $(A)$ occurs and $\Phi(\Psi) = g(\Psi)$. Denote the challenge used as $\widetilde{\mathbf{c}}^g$ and the output of the prover as $\widetilde{\mathbf{z}}_i^g, i \in [n]$.
- Run $|\Upsilon|$ parallel instances of $\hat{\mathcal{P}}^g$ denoted as $\hat{\mathcal{P}}_i^g, i \in \Upsilon$, do the following:
  - For each $\hat{\mathcal{P}}_i^g$ sample a random challenge $\mathbf{c}_i^g \leftarrow \chi^T$ subject to $\mathbf{c}_i^g[i] \neq \widetilde{\mathbf{c}}^g[i]$ and run $\hat{\mathcal{P}}_i^g$ on challenge $\mathbf{c}_i^g$.
  - For each instance $\hat{\mathcal{P}}_i^g$ that does not abort. If $(A)$ occurs and $\Phi(\Psi) = g(\Psi)$, then the vector $\mathbf{z}_i$ output by the prover verifies:
  $$H(f(\mathbf{z}_i) - \mathbf{c}_i^g[g(i)]\, y_i) = \overline{h}_i.$$
  From the previous step we had $\widetilde{\mathbf{z}}_i^g$ such that
  $$H(f(\widetilde{\mathbf{z}}_i^g) - \widetilde{\mathbf{c}}^g[g(i)]\, y_i) = \overline{h}_i.$$
  The extractor sets
  $$\mathbf{x}'_i = (\mathbf{z}_i^g - \widetilde{\mathbf{z}}_i^g) \cdot 2\left(\mathbf{c}_i^g[g(i)] - \widetilde{\mathbf{c}}^g[g(i)]\right)^{-1}$$
  Note that $f(\mathbf{x}'_i) = 2y_i$ and by Lemma 12 $\|\mathbf{x}'_i\| \leq \sqrt{d}B$.

We now prove that this extractor terminates in expected time $\mathrm{poly}(s,k) \cdot 1/p^{1+2/\log\alpha}$. Since $|\mathcal{G}| \leq 2/p^{1/\log\alpha}$ it is sufficient to show that there exists $g$ in $\mathcal{G}$ such that $\hat{\mathcal{P}}^g$ runs in time $\mathrm{poly}(s,k) \cdot 1/p^{1+1/\log\alpha}$. On any run where $(A)$ occurs, $\Phi(\Upsilon)$ is a function in $\mathcal{G}$, this implies that

$$\Pr\left[\hat{\mathcal{P}} \textit{ succeeds} \wedge (A)\right] = \sum_{g\in\mathcal{G}} \Pr\left[\hat{\mathcal{P}} \textit{ succeeds} \wedge (A) \wedge \Phi(\Upsilon) = g\right]$$

and thus

$$\begin{aligned}\exists\gamma \in \mathcal{G} \text{ s.t. } \Pr\left[\hat{\mathcal{P}} \textit{ succeeds} \wedge (A) \wedge \Phi(\Upsilon) = \gamma\right] &\geq \frac{\Pr\left[\hat{\mathcal{P}} \textit{ succeeds} \wedge (A)\right]}{|\mathcal{G}|} \\ &\geq \frac{p^{1+1/\log\alpha}}{2}\end{aligned}$$

We will use the shorthand $\hat{\mathcal{P}} \wedge \gamma$ for the event $\hat{\mathcal{P}}$ *succeeds* $\wedge (A) \wedge \Phi(\Upsilon) = \gamma$. Let $p_i$ be the probability that $\hat{\mathcal{P}}_i^\gamma$ succeeds, i.e.

$$p_i = \Pr\left[\hat{\mathcal{P}} \wedge \gamma \middle| \mathbf{c}\left[\gamma(i)\right] \neq \widetilde{\mathbf{c}}^\gamma\left[\gamma(i)\right]\right],$$

we say that $p_i$ is bad if $p_i < \frac{\Pr[\hat{\mathcal{P}}\wedge\gamma]}{k'}$ and good otherwise. If there are less than $k'$ bad $p_i$ then the extractor terminates in expected time

$$\mathrm{poly}(s,k) \cdot \frac{|\mathcal{G}|}{\Pr\left[\hat{\mathcal{P}} \wedge \gamma\right]} = \mathrm{poly}(s,k) \cdot 2^{k(1+2/\log\alpha)}$$

(c.f. the proof of Theorem 11). Assume that there are $k'$ bad $p_i$ which, for simplicity, are $p_1, \ldots, p_{k'}$. Then the event $\hat{\mathcal{P}} \wedge \gamma$ can be conditioned on the value of $\mathbf{c}[\gamma(1)]$ as

$$\begin{aligned}\Pr\left[\hat{\mathcal{P}} \wedge \gamma\right] &= \Pr\left[\hat{\mathcal{P}} \wedge \gamma \middle| \mathbf{c}\left[\gamma(1)\right] \neq \widetilde{\mathbf{c}}^\gamma\left[\gamma(1)\right]\right] \cdot \Pr\left[\mathbf{c}\left[\gamma(1)\right] \neq \widetilde{\mathbf{c}}^\gamma\left[\gamma(1)\right]\right] \\ &\quad + \Pr\left[\hat{\mathcal{P}} \wedge \gamma \middle| \mathbf{c}\left[\gamma(1)\right] = \widetilde{\mathbf{c}}^\gamma\left[\gamma(1)\right]\right] \cdot \Pr\left[\mathbf{c}\left[\gamma(1)\right] = \widetilde{\mathbf{c}}^\gamma\left[\gamma(1)\right]\right] \\ &= \frac{2d\alpha - 1}{2d\alpha} p_1 + \frac{1}{2d\alpha} \Pr\left[\hat{\mathcal{P}} \wedge \gamma \middle| \mathbf{c}\left[\gamma(1)\right] = \widetilde{\mathbf{c}}^\gamma\left[\gamma(1)\right]\right]\end{aligned}$$

Conditioning on $\mathbf{c}\left[\gamma(2)\right], \ldots, \mathbf{c}\left[\gamma(k')\right]$ we have

$$\begin{aligned}\Pr\left[\hat{\mathcal{P}} \wedge \gamma\right] &\leq \frac{2d\alpha - 1}{2d\alpha}\left(p_1 + \ldots + p_{k'}\right) \\ &\quad + \frac{1}{(2d\alpha)^{k'}} \Pr\left[\hat{\mathcal{P}} \wedge \gamma \middle| \mathbf{c}\left[\gamma(1)\right] = \widetilde{\mathbf{c}}^\gamma\left[\gamma(1)\right], \ldots, \mathbf{c}\left[\gamma(k')\right] = \widetilde{\mathbf{c}}^\gamma\left[\gamma(k')\right]\right] \\ &< \frac{2d\alpha - 1}{2d\alpha} \Pr\left[\hat{\mathcal{P}} \wedge \gamma\right] + \frac{1}{(2d\alpha)^{k'}} \\ &\leq \frac{1}{(2d\alpha)^{k'-1}} \\ &= 2^{-k(1+1/\log\alpha) - log(2d\alpha) + 1} \\ &< 2^{-k(1+1/\log\alpha) - 1}\end{aligned}$$

which contradicts the fact that $\Pr\left[\hat{\mathcal{P}} \wedge \gamma\right] \geq \frac{p^{1+1/\log\alpha}}{2}$.
From this we conclude that there are less than $k'$ bad $p_i$, and thus that the extractor has extracted a set $S$ of $n - k'$ $\mathbf{x}'_i$ such that $\|\mathbf{x}'_i\| \leq \sqrt{d}B$ and $f(\mathbf{x}'_i) = 2y_i$ in time $\mathrm{poly}(s,k) \cdot 2^{k(1+2/\log\alpha)}$. □

## 6 Proof Size

In this section we will go more in-depth in the trade-offs offered by the schemes described in Sections 4 and 5. We first give the expected value as well as an upper bound on the size of the prefix $\mathcal{S}$ of the set $\mathcal{O}$ as the second flow of the prover will consist in sending $|\mathcal{S}|$ seeds (and $|\mathcal{C}|$ hashes).

**Lemma 14.** *Let $T = 2^t$, let $\mathbf{c} \leftarrow \chi^T \in C^T$ (the set $C$ from which the values of $\mathbf{c}$ are taken does not matter, all that matters is the probability with which $0$ is sampled) with $\chi$ such that $\Pr[\chi = 0] = 1 - 1/\alpha$, let $\mathcal{O} = \{j \in [T], \mathbf{c}[j] = 0\}$, and let $\mathcal{S}(\mathbf{c}) = prefix(\mathcal{O})$ be as defined in section 4. Then:*

– *The expected size of $\mathcal{S}(\mathbf{c})$ over the choice of $\mathbf{c} \leftarrow \chi^T$ is*

$$\underset{\mathbf{c}\leftarrow\chi^T}{\mathrm{Exp}}\left[|\mathcal{S}(\mathbf{c})|\right] = \left(1 - \frac{1}{\alpha}\right)T - \sum_{i=0}^{t-1} 2^i \left(1 - \frac{1}{\alpha}\right)^{2^{t-i}}$$

– *With overwhelming probability we can bound the size of $\mathcal{S}(\mathbf{c})$ by*

$$|\mathcal{S}(\mathbf{c})| \le \left\lfloor \frac{1.4T}{\alpha} \log \frac{\alpha}{1.4} \right\rfloor$$

*Proof.* Consider the binary tree $\Gamma$ which leaves are numbered according to $[T]$, we will say that a leaf $j \in [T]$ is selected if $\mathbf{c}[j] = 0$. First observe that we can split $\Gamma$ into two trees $\Gamma_L$ and $\Gamma_R$ of size $T/2$, $\Gamma_L$ being the binary tree associated to the first $T/2$ values $\mathbf{c}_L$ of $\mathbf{c}$ and $\Gamma_R$ the tree associated to the last $T/2$ vales $\mathbf{c}_R$ of $\mathbf{c}$. The prefix $\mathcal{S}(\mathbf{c})$ of $\Gamma$ will be the union of the prefixes $\mathcal{S}(\mathbf{c}_L)$ and $\mathcal{S}(\mathbf{c}_R)$, except if all the leaves of $\Gamma$ are selected, in which case its prefix will be its root. i.e. $\forall \mathbf{c} \neq (0, \ldots, 0)$, $\mathcal{S} = \mathcal{S}(\mathbf{c}_L) \cup \mathcal{S}(\mathbf{c}_R)$, which implies $|\mathcal{S}(\mathbf{c})| = |\mathcal{S}(\mathbf{c}_L)| + |\mathcal{S}(\mathbf{c}_R)|$.
We can rewrite the expected value of $|\mathcal{S}(\mathbf{c})|$ as follows:

$$\begin{aligned}
\underset{\mathbf{c}\leftarrow\chi^T}{\mathrm{Exp}}\left[|\mathcal{S}(\mathbf{c})|\right] &= \sum_{\mathbf{c}\in C^T} |\mathcal{S}(\mathbf{c})| \Pr[\mathbf{c}] \\
&= \sum_{(\mathbf{c}_L,\mathbf{c}_R)\in C^T} \left(|\mathcal{S}(\mathbf{c}_L)| + |S(\mathbf{c}_R)|\right) \Pr[\mathbf{c}_L, \mathbf{c}_R] \\
&\quad + 1 \cdot \left(1 - \frac{1}{\alpha}\right)^T - (1+1) \cdot \left(1 - \frac{1}{\alpha}\right)^T \\
&= \sum_{\mathbf{c}_L\in C^{T/2}} \left( \sum_{\mathbf{c}_R\in C^{T/2}} \left(|\mathcal{S}(\mathbf{c}_L)| + |\mathcal{S}(\mathbf{c}_R)|\right) \Pr[\mathbf{c}_R] \right) \Pr[\mathbf{c}_L] - \left(1 - \frac{1}{\alpha}\right)^T \\
&= \sum_{\mathbf{c}_L\in C^{T/2}} \left( \underset{\mathbf{c}_R\leftarrow\chi^{T/2}}{\mathrm{Exp}}\left[|S(\mathbf{c}_R)|\right] + |\mathcal{S}(\mathbf{c}_L)| \right) \Pr[\mathbf{c}_R] - \left(1 - \frac{1}{\alpha}\right)^T \\
&= 2 \underset{\mathbf{c}\leftarrow\chi^{T/2}}{\mathrm{Exp}}\left[|S(\mathbf{c})|\right] - \left(1 - \frac{1}{\alpha}\right)^T
\end{aligned}$$

To obtain the second equality we removed $|\mathcal{S}(\mathbf{c})|$ in the case $\mathbf{c} = (0, \ldots, 0)$ from the sum and artificially added $|\mathcal{S}(\mathbf{c}_L)| + |\mathcal{S}(\mathbf{c}_R)|$. By induction on the size of $\mathbf{c}$ we obtain the expected value.
To obtain the second equation we first use the Chernoff bound to obtain a lower bound on the size of $\mathcal{O}$. Let $\mathcal{C} = [T] \setminus \mathcal{O}$, we have:

$$\Pr[|\mathcal{C}| > 1.4T/\alpha] \le e^{-\frac{T}{15\alpha}} = e^{-\frac{n}{3}}$$

since for all practical parameters we will have $n \ge 250$, we can assume that $|C| \le 1.4T/\alpha$. We consider the worst case for the size of $\mathcal{S}$ for a given $|\mathcal{C}| = a$, i.e. we define

$$W(T, a) = \max_{\#_0(\mathbf{c})=T-a} \left(|\mathcal{S}(\mathbf{c})|\right).$$

We will prove that $\forall a \in [T], W(T, a) \le a \log(T/a)$. Remark that for all $T$, $W(T, 0) = 1$. Since for all $\mathbf{c} \in C^T$ we have

$$|S(\mathbf{c})| \le |S(\mathbf{c}_L)| + |S(\mathbf{c}_R)|,$$

we get

$$W(T, a) \le \max_b \left(W(T/2, b) + W(T/2, a - b)\right)$$

where $\max(0, a - T/2) \le b \le \min(a, T/2)$. We prove that

$$\forall a \in [T], W(T, a) \le a \log(T/a)$$

by induction over $T = 2^t$:

- For $T = 1$, $W(1, 1) = 1$
- For $2T$: Assume that for all $1 \le b \le T$, $W(T, b) \le b \log(T/b)$ (and $W(T, 0) = 1$). Fix $a \in [2T]$. Let $f(b) = W(T, b) + W(T, a - b)$, then

$$W(2T, a) \le \max_b(f(b))$$

for $\max(0, a - T/2) \le b \le \min(a, T/2)$.
  - For $b = a$ or $b = 0$,

$$f(b) = W(T, a) + W(T, 0) \le a \log(T/a) + 1 \le a \log(2T/a)$$

  - For $b \neq a$ and $b \neq 0$,

$$f(b) \le a \log(T/a) + (a - b) \log(T/(a - b)).$$

    Simple analysis shows that this function reaches its maximum for $b = a/2$, and thus $f(b) \le a \log(2T/a)$

We conclude by using the fact that $W(T, a)$ is an integer. Finally, with high probability

$$|\mathcal{S}(\mathbf{c})| \le W(T, 1.4T/\alpha) \le \left\lfloor \frac{1.4T}{\alpha} \log \frac{\alpha}{1.4} \right\rfloor$$

□

We will show that the size of the protocol given in Figure 4 can be made nearly independent of the parameter $\alpha$ by cleverly encoding each flow. We will consider the four flows of the protocol each on its own (though it is clear that the proof really is a three-move protocol since the last two flows can be sent simultaneously).

**First Flow:** The prover sends $h \in \{0, 1\}^{256}$ to the verifier, this is clearly independent of $\alpha$.

$$\text{Flow size} = 256 \text{ bits}$$

**Second Flow:** The verifier sends $\mathbf{c} \in \{0, 1\}^T$ to the prover, this takes $5\alpha n$ bits since $T = 5\alpha n$. However the verifier can compute the sets $\mathcal{O}$ and $\mathcal{C} = [T] \setminus \mathcal{O}$ before sending $\mathbf{c}$ (rather than doing it afterwards) and equivalently send the set $\mathcal{C}$, we have $|\mathcal{C}| \le 7n$ and since the indices of $\mathcal{C}$ are in $[T]$ they can be encoded in $\log(5\alpha n)$ bits. The second flow only depends on $\alpha$ logarithmically.

$$\text{Flow size} \le 7n \log(5\alpha n) \text{ bits}$$

**Third Flow:** The prover sends $(s_j)_{j \in \mathcal{S}}$ and $(h_j)_{j \in \mathcal{C}}$ to the verifier. From Lemma 14 we have that $|\mathcal{S}| \le 7n \log(\alpha/1.4)$ and similarly $|\mathcal{C}| \le 7n$, since the seeds and hash all are in $\{0, 1\}^{256}$ this flow depends logarithmically on $\alpha$.

$$\text{Flow size} \le 7n \log\left(\frac{2\alpha}{1.4}\right) \cdot 256 \text{ bits}$$

**Fourth Flow (i.e. second part of the Third Flow):** The prover sends $\Phi$ and $(\mathbf{z}_i)_{i \in [n]}$ to the verifier. Since $\Phi \in [T]^n$ sending it naively would require $n \log(5\alpha n)$ bits, however all the elements of $\Phi$ correspond to non-zero indices of $\mathbf{c}$, i.e. they are in $\mathcal{C}$. $\Phi$ can thus be encoded using $n \log(|\mathcal{C}|) \le n \log(7n)$ bits. The coefficients of the $\mathbf{z}_i$ come form $\mathcal{D}_\sigma$ by a tail cutting argument they can be represented in $\log(11\sigma) = \log(11^2\beta)$ bits each and there are $nr$ of them. The fourth flow is independent of $\alpha$.

$$\text{Flow size} \le n \log(7n) + nr \log(11^2\beta) \text{ bits}$$

The proof in Figure 6 only differs in size from this proof on the second flow, where the challenge $\mathbf{c}$ is in $\left\{0, \pm 1, \pm X, \ldots, \pm X^{d-1}\right\}^T$. But similarly to the encoding we use for the first protocol, the verifier can simply send the set $\mathcal{C}$ as well as a vector of dimension $|\mathcal{C}|$ containing the challenges in $\left\{\pm 1, \pm X, \ldots, \pm X^{d-1}\right\}$.

| | [CD16] | Protocol I | | Protocol II | | |
|---|---|---|---|---|---|---|
| $\alpha$ | 2 | 64 | 1024 | 2 | 64 | 1024 |
| k | 128 | 128 | 128 | 128 | 128 | 128 |
| $\tau(k)$ | 129 | 23 | 14 | 22 | 10 | 8 |
| n | 67103 | 2213 | 853 | 2027 | 443 | 293 |
| T | $6.7 * 10^6$ | $7.1 * 10^5$ | $4.4 * 10^6$ | $2 * 10^4$ | $1.4 * 10^5$ | $1.5 * 10^6$ |
| Slack | $2.6 * 10^5$ | $4.6 * 10^4$ | $2.8 * 10^4$ | $1.4 * 10^6$ | $6.4 * 10^5$ | $25.1 * 10^5$ |
| Communication | 8.5 kB | 9.7 kB | 10.9 kB | $8.2kB$ | 9.5 kB | 10.7 kB |
| run-time | 16 | 512 | 8192 | 8 | 512 | 8192 |

**Table 2.** Comparison between [CD16] and our protocols for the R-LWE ivOWF with binary secrets. Masking parameters are revealed with probability $1 - 1/\alpha$, $k$ is the security parameter, $\tau(k)$ the imperfection of the protocol, and $n$ the number of secrets. The communication is per secret and the run-time is in number of evaluations of the ivOWF per secret per player.

The size of the second flow now becomes $7n\log(5\alpha n) + 7n\log(2d)$. The total size of the proof is finally upper bounded by:

$$256 + n\left(7\log(5\alpha n) + 1792\log\left(\frac{2\alpha}{1.4}\right) + \log(7n) + \boxed{7\log(2d)} + r\log(11^2\beta)\right) \text{ bits}$$

Where the boxed term only exists in the protocol from Figure 6. Note that this size only has a very slight dependence on $\alpha$. In fact the largest summand will be the one corresponding to the $\mathbf{z}_i$ up to $\alpha \sim 2^{30}$, for which the computation requirements of the proof will already be the bottleneck. The complete proof consists in two iterations of the imperfect proof, one with parameter $\beta$ and the second with parameter $\tau(k)\beta$, the size of the complete proof is thus:

$$\begin{aligned}512 + n\Bigg( & 14\log(5\alpha n) + 3584\log\left(\frac{2\alpha}{1.4}\right) \\ & + 2\log(7n) + \boxed{14\log(2d)} + r\log(\tau(k)11^4\beta^2)\Bigg) \text{ bits}\end{aligned}$$

And if we consider the average case rather than the worst case we can assume that $|\mathcal{S}| \leq 5n\log\alpha$ (which is very close to the expected value given in Lemma 14) and $|\mathcal{C}| = 5n$. Which gives the expected proof size:

$$\begin{aligned}512 + n\Bigg( & 10\log(5\alpha n) + 2560\log(2\alpha) \\ & + 2\log(5n) + \boxed{10\log(2d)} + r\log(\tau(k)11^4\beta^2)\Bigg) \text{ bits}\end{aligned}$$

We compare in Table 2 our scheme with the one of [CD16] for the (Ring)-LWE one-way function with dimension $d = 1024$ (so $r = 2048$), and binary secrets (so $\beta = \sqrt{r}$). For a fair comparison we consider the protocol of [CD16] in the euclidean norm and with our improvements (only one hash in the first flow and seeds instead of $\mathbf{g}_j$ in the third flow). The communication cost per secret and the slack are rather similar in all three protocols. The main difference being that our protocols allows for amortization over very few secrets but at a larger computation cost. In Figure 1 we plot the number $n$ of secrets we can amortize over as a function of $\log\alpha$. It is apparent that increasing $\log\alpha$ past a certain threshold yields very little advantage while drastically increasing the computation cost (which grows linearly in $\alpha$). It is also clear that our second protocol gives better amortization than the first one, though this only proves the knowledge of short pre-images of $2y$.

## References

Ban93. Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296:625635, 1993.

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 *ASIACRYPT*, pages 551–572, 2014.

BDLN16. Carsten Baum, Ivan Damgård, Kasper Green Larsen, and Michael Nielsen. How to prove knowledge of small secrets. In *CRYPTO*, pages 478–498, 2016.

BDPA16. Guido Bertoni, Joan Daemen, Michal Peeters, and Gilles Van Assche1. The keccak sponge function family, 2016.

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

CD16. Ronald Cramer and Ivan Damgård. Amortized complexity of zero-knowledge proofs revisited: Achieving linear soundness slack. *IACR Cryptology ePrint Archive*, 2016:681, 2016.

DPSZ12. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, pages 643–662, 2012.

KTX08. Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT*, pages 372–389, 2008.

LN17. Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption from lattices. In *EUROCRYPT*, 2017.

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

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

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

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755, 2012.

Ste93. Jacques Stern. A new identification scheme based on syndrome decoding. In *CRYPTO*, pages 13–21, 1993.