Towards a Unified Approach to Black-Box Constructions of Zero-Knowledge Proofs*

Xiao Liang[0000−0003−0858−9289] and Omkant Pandey

Stony Brook University, Stony Brook NY 11790, USA
{liang1,omkant}@cs.stonybrook.edu

Abstract. General-purpose zero-knowledge proofs for all NP languages greatly simplify secure protocol design. However, they inherently require the code of the underlying relation. If the relation contains black-box calls to a cryptographic function, the code of that function must be known to use the ZK proof, even if both the relation and the proof require only black-box access to the function. Rosulek (Crypto'12) shows that non-trivial proofs for even simple statements, such as membership in the range of a one-way function, require non-black-box access.

We propose an alternative approach to bypass Rosulek's impossibility result. Instead of asking for a ZK proof directly for the given one-way function f, we seek to construct a *new* one-way function F given only black-box access to f, *and* an associated ZK protocol for proving non-trivial statements, such as range membership, over its output. We say that F, along with its proof system, is a *proof-based* one-way function. We similarly define proof-based versions of other primitives, specifically pseudo-random generators and collision-resistant hash functions.

We show how to construct proof-based versions of each of the primitives mentioned above from their ordinary counterparts under mild but necessary restrictions over the input. More specifically,

- We first show that if the prover entirely chooses the input, then proof-based pseudo-random generators cannot be constructed from ordinary ones in a black-box manner, thus establishing that some restrictions over the input are necessary.
- We next present black-box constructions handling inputs of the form (x, r) where r is chosen uniformly by the verifier. This is similar to the restrictions in the widely used Goldreich-Levin theorem. The associated ZK proofs support range membership over the output as well as arbitrary predicates over prefixes of the input.

Our results open up the possibility that general-purpose ZK proofs for relations that require black-box access to the primitives above may be possible in the future without violating their black-box nature by instantiating them using proof-based primitives instead of ordinary ones.

* This material is based upon work supported in part by DARPA SIEVE Award HR00112020026, NSF grants 1907908 and 2028920, and a Cisco Research Award. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government, DARPA, NSF, or Cisco.

Keywords: Zero-Knowledge · Black-Box · Separation

1 Introduction

Zero-knowledge proofs (ZKPs) are a method to prove that a statement is true without revealing any additional knowledge [16]. A major achievement in cryptography has been the construction of ZKPs for NP-complete problems [15]. Since every NP relation can be efficiently reduced to any NP-complete relation [3, 29, 34], this yields a ZKP for all languages in NP. Due to this reason, ZKPs for NP-complete problems are often called *general-purpose* proofs. As evidenced by numerous follow up works, general-purpose proofs have been incredibly useful to the theory of cryptography.

Early constructions of general-purpose ZKPs required only *black-box* access to any one-way function (OWF), i.e., they used the given OWF as an *oracle*. A *black-box construction* of this kind thus depends only on the input/output behavior of the given cryptographic primitive. In particular, it is independent of the specific implementation or *code* of the primitive.

A black-box construction is often preferred over a non-black one due to its attractive properties. For example, it remains valid even if the primitive/oracle is based on a *physical* object such as a noisy-channel or tamper-proof hardware [43, 4, 11]. Also, its efficiency does not depend on the implementation details of the primitive, thus establishing that efficiency can be theoretically independent of the primitive's code.

Unfortunately, general-purpose proofs are not suitable when seeking a black-box construction for some desired cryptographic task since they inherently require the full code of the underlying relation to perform the NP reduction. In other words, if the relation requires black-box access to a OWF, the code of the OWF must be known *even though* neither the ZKP nor the relation needs it. In fact, this has been the main reason for the non-black-box nature of many cryptographic constructions that are otherwise optimal. Analogous black-box constructions often require significant effort and technical innovation, as evidenced by the secure computation literature, e.g., [31, 6, 26, 27, 20, 28, 39, 42, 17, 18, 36, 32, 8, 22, 9, 2, 10].

In light of the above situation, it is tempting to imagine a "dream version" of general-purpose proofs where, if the underlying relation R requires black-box access to a cryptographic function f, say from a specified class such as the class of OWFs, then so should the general-purpose ZKP for proving membership in R. We informally refer to such relations as *black-box relations*. Such a result, if possible, would greatly simplify the task of future black-box constructions and potentially unify the diverse set of techniques that exist in this area.

As one might suspect, this dream version is too good to be true. In his beautiful work, Rosulek [41] rules out ZKPs for proving *membership in the range* of a OWF f given as an oracle. More specifically, assuming injective OWFs, Rosulek rules out (even honest-verifier) *witness-hiding* protocols [7] for the relation

$R^f = \{(y, x) \mid y = f(x)\}$ where f is chosen from the class of all OWFs and provided as an oracle to the protocol.

In contrast to the negative result for OWFs, a large body of literature constructs so-called *black-box commit-and-prove* protocols [27, 18, 19, 23, 30, 33]. Informally speaking, a commit-and-prove protocol between a committer and a receiver ensures that at the end of the protocol, the committer is committed to some hidden value satisfying a pre-defined property. This primitive can be constructed with only black-box access to an ordinary commitment scheme which may originally not support any proofs whatsoever. In many situations, commit-and-prove protocols serve as a good substitute for ordinary commitments; moreover, their ability to support proofs over committed values makes them a great tool for constructing larger black-box protocols.

In hindsight, we can view black-box commit-and-prove protocols as an alternative to bypass the aforementioned negative result of [41]. That is, instead of constructing ZKP directly for every OWF, we ask the following indirect question:

> *Given only black-box access to a OWF f, can we construct a new OWF F and a ZKP system Π_F for proving membership in the range of F?*

Of course, we can ask for general properties instead of merely range-membership.

The idea is that F can be used as a substitute for f in any computation $C^{(\cdot)}$ that requires only black-box access to OWFs. More importantly, it gives hope that general-purpose black-box ZKPs for proving the correctness of computation $C^{(\cdot)}$ may be possible since the correctness of responses from F can be ensured using Π_F, all while requiring only black-box access to f. We remark that we do not obtain such a result for general computations in this work and merely point out that the existence of (F, Π_F) may open a path towards it.

We call the pair (F, Π_F) a *proof-based one-way function* (PB-OWF). Analogously, we consider proof-based versions of other primitives, specifically pseudorandom generators (PRGs) and collision-resistant hash functions (CRHFs). Motivated by the aforementioned possibility of a general-purpose proof system for black-box cryptographic computations $C^{(\cdot)}$, this paper initiates a study of black-box constructions of proof-based cryptographic primitives. We obtain a mix of both negative and positive results as outlined below.

1.1 Our Results

Given the existence of black-box commit-and-prove protocols, it is not unreasonable to expect that black-box proof-based versions of OWFs, PRGs, and CRHFs might also exist. Interestingly, the fact that these primitives are *deterministic* functions really separates them from commitments. The existence of their proof-based versions seems to depend on how we view the input, as discussed below.

Negative Results via Black-Box Separation. In common applications of non-interactive primitives such as OWFs and PRGs, the entire input is usually controlled by the evaluator of these functions. We show that *proof-based PRGs* where the input (i.e., the seed) is *entirely* chosen by the evaluator cannot be

constructed in a black-box manner from an (ordinary) OWF chosen from the class of all OWFs. Since PRGs can be constructed in a fully-black-box manner from OWFs [14, 24, 21], this separates proof-based PRGs from ordinary PRGs.

More specifically, black-box construction of a proof-based PRG from (ordinary) OWFs consists of a *deterministic* and efficient oracle algorithm $G^{(\cdot)}$, along with an efficient protocol, $\Pi_G^{(\cdot)} = \langle P^{(\cdot)}, V^{(\cdot)} \rangle$, of two interactive oracle machines.[1] For every OWF f, algorithm G^f should be a PRG, and protocol $\Pi^f = \langle P^f, V^f \rangle$ should be a ZKP system for the relation $R_G^f = \{(y, x) \mid \text{s.t. } y = G^f(x)\}$. Then, we show that a *fully-black-box* reduction [25, 40] from proof-based PRGs to ordinary OWFs does not exist if the prover chooses the entire seed.

The range-membership relation $R^f = \{(y, x) \mid y = f(x)\}$ ruled out in [41] is a special case of the aforementioned relation $R_G^f = \{(y, x) \mid \text{s.t. } y = G^f(x)\}$. In our terminology, Rosulek rules out a special type of proof-based OWF $(F^{(\cdot)}, \Pi_F^{(\cdot)})$ where F is just a "delegate" for the oracle OWF; i.e., it returns the oracle's response when queried on the given input. This is captured in [41] by formally defining the notion of *functionally-black-box* (FBB) protocols. In contrast, the relation we consider can make polynomially many queries to the oracle on arbitrary inputs and compute over the responses to produce the output. We extend the notion of FBB protocols to formally capture these extensions.

In part due to these differences and our overall goals, our negative result is incomparable to that of Rosulek's. While Rosulek rules out black-box proofs for range-membership for OWFs assuming injective OWFs, ours is only a black-box separation, albeit without any additional assumptions. A black-box separation is the best one can hope for in our setting since *non-black-box* constructions of proof-based OWFs that use the code of the oracle trivially exist.

Positive Results. We next investigate whether mild restrictions on the inputs can help bypass the black-box separation result. One option is to consider modifications along the lines of the Goldreich-Levin (GL) hardcore predicate [14], where one considers a OWF F constructed from any given OWF f on inputs of the form (x, r). This makes it possible to show that predicate $\mathsf{hc}(x, r) := \oplus_i (x_i \cdot r_i)$ is hardcore for the modified function $F(x, r) := r \| f(x)$ even though a hardcore predicate for arbitrary OWFs is still unknown. These changes to the function and the input do not seem to significantly affect the applicability of their result.

We adopt a similar approach to construct proof-based primitives. Continuing with OWFs as example, we seek to construct a proof-based OWF $F^{(\cdot)}$ which can be instantiated with only black-box access to any OWF f, and takes inputs of the form (x, r). As in the GL setting, x will act as the "main input" chosen by the evaluator/prover, and r will be publicly accessible from the output of $F^f(x, r)$. However, in a crucial difference, r will be *chosen by the verifier* during the execution of ZKP Π_F^f. There are no other restrictions on any of the objects. Some remarks are in order.

[1] Note that the protocol is allowed to depend on $G^{(\cdot)}$ but not on the oracle which may be arbitrarily chosen later. The same holds for the relation R_G^f introduced next.

1. In light of our black-box separation result, it is essential to let the verifier choose r since no other restrictions are present. This means that the computation of $y = F^f(x, r)$ must be performed during the proof. We formalize this by modeling Π_F^f as *a secure two-party computation* protocol for evaluating the functionality that on inputs x and r from relevant parties, returns y. The ZK property is captured by requiring simulation-based security against malicious receivers; for soundness we only require that the honest verifier, with high probability, does not output a y^* that is not in the range. This is effectively a black-box ZKP for the relation $R_F^f(r) = \{(y, x) \mid \text{s.t. } y = F^f(x, r)\}$.[2]
2. The verifier must choose r from an unpredictable distribution such as the uniform distribution over sufficiently long strings, since otherwise, the soundness would be impossible as a cheating prover can simply guess r, bringing us back to the setting of the separation result.
3. Since r may be maliciously chosen by the verifier to violate the one-way property of F^f, we require that for *every string* r, the function defined by $F^f(\cdot, r)$ is one-way as long as f is one-way.

We follow the same approach for formally defining proof-based versions of PRGs and CRHFs. Having settled on a satisfactory definition, we present black-box constructions of the proof-based versions of OWFs (for range membership), as well as PRGs and CRHFs, directly from their ordinary counterparts.

Theorem 1 (Informal). *There is a fully black-box construction of* proof-based $\mathsf{primitive}$ *as described above for range-membership and two-party inputs of the form* (x, r)*, assuming that* $\mathsf{primitive}$ *exists, where* $\mathsf{primitive} \in \{\mathrm{OWF}, \mathrm{PRG}, \mathrm{CRHF}\}$.

At first glance, one may wonder whether black-box commit-and-prove protocols already yield proof-based OWFs. That is, the commit phase of such protocols can be viewed as a one-way function over the input (x, r) where x is the value to be committed and r is the randomness, the output y is the transcript of the commit-phase, and the proof-phase plays the role of associated ZKP. This approach does not really work since the commit-and-prove protocols merely *bind* the prover to a well-defined value x. They do not guarantee that w.h.p. every accepting transcript has a valid "preimage" (x, r) that maps to it. In contrast, the soundness of range-membership proofs of proof-based OWFs requires that w.h.p. a preimage must exist for the output accepted by the honest verifier. At a technical level, the black-box commit-and-prove protocols are based on cut-and-choose techniques that can only guarantee that the accepted value is *close* to an honestly generated value, which is insufficient to guarantee a preimage.

Extensions. We show that it is possible to construct a slightly more general proof-system than merely range-membership for each of our proof-based primitives. Continuing with the OWF example, we can construct a black-box proof-based OWF F^f such that for any predicate ϕ, the verifier learns a value y with

[2] For now, we only focus on range-membership proofs. The definitional approach is consistent with the commit-and-prove literature, although there are important differences since we are dealing with deterministic primitives.

the guarantee that there exists an input (x, r) such that: (1) $y = F^f(x, r)$ where r is chosen uniformly by the honest receiver, (2) $x = \alpha \| x'$, and (3) $\phi(\alpha) = 1$. That is, we can support any predicate (in fact, computation of any function) over a *prefix* of the preimage of the output. The ZKP system here depends on the code of ϕ but not that of f as before.

This extension is motivated by similar results for commit-and-prove which are quite useful in constructing larger black-box protocols [18, 30]. We achieve this by presenting a new construction which combines our ideas for range-membership with the "MPC-in-the-head" technique [27].

Due to space constraints, these extensions are formally described in the full version [35].

2 Technical Overview

2.1 Black-Box Separation

We first present a very brief overview of our black-box separation. A detailed overview is given in Sec. 4.2 after setting up necessary notation and definitions.

Let us first recall how Rosulek [41] rules out FBB constructions of honest-verifier witness-hiding (HVWH) protocols for the range-membership of OWFs, assuming injective OWFs exist.

The proof starts by assuming that such protocols exist. In particular, when instantiated with an injective OWF f, the protocol (P^f, V^f) is HVWH for $R^f = \{(y, x) \mid \text{s.t. } y = f(x)\}$. Since f is injective, for a pair $(x^*, y^* = f(x^*))$ remapping $f(x^*)$ to a value different from y^* will give us a new OWF f' whose range does not contain y^* anymore. Moreover, the verifier accepts in $\langle P^f(x^*, y^*), V^{f'}(y^*)\rangle$ with roughly the same probability as in $\langle P^f(x^*, y^*), V^f(y^*)\rangle$. This is because the only opportunity to distinguish these two executions is when the verifier queries its oracle on x^*; but this happens with negligible probability because of the HVWH property of the protocol. However, this contradicts the soundness: V's oracle now becomes f', and $\nexists x$ s.t. $(y^*, x) \in R^{f'}$.

It is unclear how to reuse the above technique to rule out PB-OWFs. As mentioned earlier, there are no restrictions on how the $F^{(\cdot)}$ part behaves. In particular, it is not guaranteed that F^f is injective even if f is injective. Thus, "carving out" a value from the range of f may not affect the range of F^f.

To derive the desired contradiction, we take a fundamentally different approach to construct f'. We first define a set Q^{Easy}, which consists of only the queries made by the receiver with "high" probability during the (honest) execution $\langle S^f(x^*, y^*), R^f(y^*)\rangle$. We then define f' by maintaining the same behavior as f on Q^{Easy}, *and* re-sampling all the remaining points uniformly at random. Note that the receiver will still accept with "high" probability even if we change its oracle to f', because f' and f only differ at the points that are queried with "low" probability (i.e., the points outside Q^{Easy}). Now, the only thing left is to show that y^* is not in the range of $F^{f'}$. Unfortunately, due to the generality of $F^{(\cdot)}$, we do not know how to do that.

However, if we switch our focus to a PB-PRG G^f (instead of PB-OWF F^f), we can prove the following lemma which helps separate PB-PRGs from OWFs:

Lemma 1 (Informal). *If we start with a y^* in the range of G^f, y^* is still in the range of $G^{f'}$ with probability $0.5 \pm \mathsf{negl}(\lambda)$, where $G^{(\cdot)}$ is a PRG when instantiated with any OWF f as its oracle.*

Let us show the intuition behind Lem. 1. Assume the lemma is false. In the pseudo-randomness game for G^f, we show how to identify the case $y^* \in \mathsf{Range}(G^f)$ *correctly*, with probability noticeably better than 0.5, thus contradicting pseudo-randomness. To do that, the adversary simply estimates the probability that $y^* \in \mathsf{Range}(G^{f'})$. We will show that, if this probability is noticeably far from 0.5, $\Pr\left[y^* \in \mathsf{Range}(G^f)\right]$ is also noticeably far from 0.5.

Doing this successfully requires the adversary to know Q^{Easy}, which it does not. However, the adversary can run the HVZK simulator many times to get an estimate $\widetilde{Q}^{\mathsf{Easy}}$ for the real Q^{Easy}. We will show that $\widetilde{Q}^{\mathsf{Easy}}$ suffices for our proof. Note also that the adversary needs to perform exponential work when computing the probability that $y^* \in \mathsf{Range}(G^{f'})$, even if it knows the set $\widetilde{Q}^{\mathsf{Easy}}$. However, it only makes *polynomially many* oracle queries (when executing the HVZK simulator), which suffices for proving the *fully*-black-box separation.

2.2 Proof-Based One-Way Functions (and PRGs)

Let us start by considering the following basic construction for PB-OWF (F^f, Π_F^f) over inputs of the form (x, r). The construction is based on "cut-and-choose" techniques where, the sender queries the oracle f on "blocks" of x, and the receiver checks a size-t random subset (defined by r) of the responses. This method is not sound since it can only guarantee that the sender's response is correct on most but not all blocks. We will handle this issue by introducing a new idea.

Basic Construction. PB-OWF (F^f, Π_F^f) handles inputs of the form (x, r). F^f computes as follows:[3]

1. Parse x as $(x_1, \ldots, x_n)$.
2. Interpret r as a size-t $(t < n)$ subset of $[n]$, denoted by $\{b_1, \ldots, b_t\}$.
3. Output $\mathsf{y} = (y_1, \ldots, y_n) \| (x_{b_1}, \ldots, x_{b_t}) \| \mathsf{r}$, where $y_i = f(x_i)$ for all $i \in [n]$.

On input x to S^f and r to R^f, the execution $\langle S^f(\mathsf{x}), R^f(\mathsf{r}) \rangle$ is as follows:

1. S^f parses x as $(x_1, \ldots, x_n)$, and computes $(y_1, \ldots, y_n)$ via its oracle access to f (i.e., $y_i = f(x_i)$). It sends $(y_1, \ldots, y_n)$ to the receiver.
2. R^f sends its input r to S^f. Same as in F^f, the r specifies a size-t subset $\{b_1, \ldots, b_t\}$ of $[n]$. Recall that the honest receiver's input r is random. In this case, $\{b_1, \ldots, b_t\}$ is a *random* subset of $[n]$.
3. S^f sends $(x_{b_1}, \ldots, x_{b_t})$, i.e., the x_i's whose indices are specified by r.
4. R^f checks (via its oracle access to f) if $y_{b_i} = f(x_{b_i})$ for all $i \in [t]$. If all the checks pass, R^f output $\mathsf{y} = (y_1, \ldots, y_n) \| (x_{b_1}, \ldots, x_{b_t}) \| \mathsf{r}$.

[3] We use Prover/Verifier and Sender/Receiver interchangeably since our ZKP is captured by considering a secure computation style definition for two parties.

Completeness is straightforward; furthermore $F^f(\cdot, \mathsf{r})$ is trivially one-way for every r since $t < n$ and f is a OWF. Let us first consider the *honest-verifier zero-knowledge* (HVZK) property of protocol Π_F^f.

Recall that the ZK property is defined via the ideal/real paradigm for secure computation, and requires simulation-security against malicious receivers. Thus, to prove the HVZK property, we need to show an ideal-world simulator Sim for the honest receiver. This is easy as the honest receiver will always use the given input r, which is uniformly distributed. More specifically, Sim works by sampling a uniform r by itself, sending the r to the ideal functionality, and receiving back the output $\mathsf{y} = (y_1, \ldots, y_n) \| (x_{b_1}, \ldots, x_{b_t}) \| \mathsf{r}$. With this y, Sim can easily generate a simulated transcript that is identically distributed to the real one.

ZK Against Malicious Receivers. The above simulation strategy does not work for malicious receivers, because they may not use the given input r. Therefore, the simulator needs to somehow extract the candidate input r^* from the malicious receiver. However, the receiver will not give out its r^* until the sender/simulator sends the $\{y_i\}_{i\in[n]}$ values.

We point out that this issue cannot be fixed using standard methods such as requiring the receiver to commit to r and to open it later. This is because later, we will introduce a pre-image editing condition, and require that the sender's computation of F^f be consistent with this editing.

We therefore use a different idea. We modify the protocol to use a black-box commit-and-prove scheme $\Pi_{\mathsf{ZKCnP}} = (\mathsf{BBCom}, \mathsf{BBProve})$ with ZK property. This scheme has a pair of simulators $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ that can be used to simulate the receiver's view in the commit phase and the prove phase respectively. Our new Π_F^f is the same as before, except for the following changes:

- In Step 1, instead of sending y_i's as before, the sender commits to them using BBCom. Formally, the sender sets $\nu = (y_1, \ldots, y_n)$ and executes $\mathsf{BBCom}(\nu)$ with the receiver.
- In Step 3, the sender sends both $\{x_{b_i}\}_{i\in[n]}$ *and* the value ν. It then proves using $\mathsf{BBProve}$ that this ν is indeed the value committed in BBCom.

As before, the receiver needs $(y_1, \ldots, y_n)$ to execute Step 4. Now, these values are contained in ν, and $\mathsf{BBProve}$ guarantees that the sender cannot change ν.

With these modifications, we can prove the ZK property for malicious receivers as follows. The simulator starts by running Sim_1 (the commit-phase simulator) with the malicious receiver R^{*f}. In this way, the simulator can go through Step 1 smoothly, without knowing the actual $\{y_i\}_{i\in[n]}$ values. Then, it will receive the r^* from R^{*f}. The simulator sends r^* to the ideal functionality and receives back $\mathsf{y} = (y_1, \ldots, y_n) \| (x_{b_1}, \ldots, x_{b_t}) \| \mathsf{r}^*$. It sends $\nu = (y_1, \ldots, y_n)$ and $(x_{b_1}, \ldots, x_{b_t})$ to the receiver. Then, instead of executing $\mathsf{BBProve}$, the simulator invokes Sim_2 to help itself go through the $\mathsf{BBProve}$ stage. It is easy to see that the ν and $\{x_{b_i}\}_{i\in[t]}$ sent by the simulator meet the consistency requirement in Step 4. Relying on the ZK property of Π_{ZKCnP}, one can formally prove that the simulation is done properly.

Soundness and Preimage Editing. As mentioned earlier, the "cut-and-choose" structure is not sufficient to guarantee the existence of a preimage. To see that, consider a malicious sender who picks an $i^* \in [n]$ at random, sets y_{i^*} to some value not in the range of f, or behaves honestly otherwise. This malicious sender can still make the honest receiver accept with non-negligible probability, even if t is as large as $n-1$ (the upper bound for t to achieve any non-trivial ZK property). This is addressed by modifying the construction of F^f.

We start by noting that the "cut-and-choose" trick ensures that most of the y_i's are "good" (i.e., having preimages under f). For example, if t is a constant fraction of n, then the protocol ensures (except for negligible probability) that at most k of the y_i's are "bad", where k is another constant fraction of n. Therefore, our idea is to extend the range of F^f to include all the images y that have $\leq k$ bad y_i's. More specifically, our new F^f works as follows. On input (x, r), it still interprets r as $\{b_1, \ldots, b_t\}$. But it will parse x as

$$\mathsf{x} = (x_1, \ldots, x_n) \| \underbrace{(p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k})}_{\beta},$$

where the $\{p_1, \ldots, p_k\}$ form a size-k subset of $[n]$. The evaluation of $F^f(\mathsf{x}, \mathsf{r})$ consists of two cases:

- **Non-Editing Case:** if $\{b_1, \ldots, b_t\} \cap \{p_1, \ldots, p_k\} \neq \emptyset$, then it computes y as before, ignoring the β part. That is, it outputs $\mathsf{y} = (s_1, \ldots, s_n) \| (x_{b_1}, \ldots, x_{b_t}) \| \mathsf{r}$, where $s_i = y_i$ for all $i \in [n]$.
- **Editing Case:** if $\{b_1, \ldots, b_t\} \cap \{p_1, \ldots, p_k\} = \emptyset$, then at positions specified by p_i's, it replace y_{p_i} with y'_{p_i}. Namely, it outputs $\mathsf{y} = (s_1, \ldots, s_n) \| (x_{b_1}, \ldots, x_{b_t}) \| \mathsf{r}$, where $s_i := \begin{cases} y'_i & i \in \{p_1, \ldots, p_k\} \\ y_i & i \in [n] \setminus \{p_1, \ldots, p_k\} \end{cases}$.

Let us explain how this editing technique resolves the soundness issue. Consider a y^* learned by the honest receiver with input r. As mentioned before, there are at most k y_i values (among those contained in y^*) that do not have preimages under f. These values can be expressed as $\{y^*_{p_1}, \ldots, y^*_{p_k}\}$, i.e., their indices are $\{p_1, \ldots, p_k\}$. Moreover, this set of bad indices does not overlap with the $\{b_1, \ldots, b_t\}$ specified by r; otherwise, the receiver would abort when performing the checks in Step 4. Therefore, by setting the β part to $(p_1, y^*_{p_1}), \ldots, (p_k, y^*_{p_k})$, we will obtain a valid preimage for y^* under our new $F^f(\cdot, \mathsf{r})$.

One may wonder whether a malicious sender can cheat by taking advantage of the **editing** case. However, since the honest receiver will use a random r, the set $\{b_1, \ldots, b_t\}$ will always overlap with $\{p_1, \ldots, p_k\}$ (except for negligible probability). That is, although we prove soundness by relying on the **editing** case, it almost never happens in a real execution. So, this will not give malicious senders any extra power.

We remark that the above preimage-editing idea is compatible with our technique for achieving (full) ZK. Now, the sender will append $(y_1, \ldots, y_n)$ and β to the committed value ν. Upon receiving R^f's challenge r, the sender computes $\mathsf{s} = (s_1, \ldots, s_n)$ according to the above definition of F^f. It sends both s and

$\{x_{b_1}, \ldots, x_{b_t}\}$ to the receivers. Then, it runs $\mathsf{BBProve}$ to prove that it does the editing (or non-editing) honestly. Note that this statement can be expressed as a predicate on the values s, r, β, and $\{y_i\}_{i\in[n]}$, where the last two are committed in $\mathsf{BBCom}(\nu)$. Since it does not involve the code of f, the protocol remains black-box in f. We provide more details in Sec. 5.2.

Proof-Based PRGs. Following the above paradigm, we also obtain a proof-based PRG by simply replacing the oracle OWF f with a PRG in the above PB-OWF construction. We provide a formal treatment in the full version [35].

2.3 Proof-Based Collision-Resistant Hash Functions

Recall that a PB-CRHF consists of a function H^h and a protocol Π_H^h such that for any CRHF h:

- For all r, $H^h(\cdot, \mathsf{r})$ is a CRHF; **and**
- $\Pi_H^h = (S^h, R^h)$ is protocol satisfying similar completeness, soundness and ZK properties as for our PB-OWFs, but w.r.t. H^h.

Let us first try to reuse the idea from our PB-OWFs. On input (x, r), the H^h first parses x as $(x_1, \ldots, x_n)\|\beta$, where the β has the same structure as before, for the purpose of preimage editing. It then generates $\{y_i\}_{i\in[n]}$ where $y_i = h(x_i)$, and outputs $\mathsf{y} = \mathsf{s}\|(x_{b_1}, \ldots, x_{b_t})\|\mathsf{r}$, where the value $\mathsf{s} = (s_1, \ldots, s_n)$ is computed by editing $\{y_i\}$ (in the same way as for our PB-OWFs).

Since h is also a OWF, the H^h is surely one-way. However, it is not collision-resistant. To see that, recall that in the **non-editing** case, the β part is not used when computing $H^h(\mathsf{x}, \mathsf{r})$. This implies the following collision-finding attack. For a fix r, the adversary first computes $\mathsf{y}^* = H^h(\mathsf{x}^*, \mathsf{r})$ with an x^* whose β part does *not* trigger the **editing** condition. Then, it can easily find many preimages for y^* by using different β's, as long as they do not trigger the **editing** condition. Therefore, we need to come up with a new editing method that does not compromise collision resistance.

To do that, we modify H^h as follows. We sample a public string z and hard-wire it in H^h. In this way, H_z^h can be viewed as a member of the public-coin collision-resistant hash *family* indexed by z, instead of a single CRHF. Then, we can think of x as containing additionally two strings τ and μ. When evaluating $H_z^h(\mathsf{x}, \mathsf{r})$, we will perform the editing if $\{b_1, \ldots, b_t\} \cap \{p_1, \ldots, p_k\} = \emptyset$ *and* $\alpha \neq z$ *and* $h(\tau) = h(z)$. Moreover, we include the value $\mathsf{t} = h(\beta\|\tau\|\mu)$ in the output y. Intuitively, this hash of β in y prevents the adversary from constructing collisions using a different β.

We now explain how to perform editing in this setting. First, we will include in x an additional value τ such that $\tau \neq z$ and $h(\tau) = h(z)$. This allows us to trigger the **editing** condition. With z sampled randomly, it is not hard to see that such a τ exists with overwhelming probability[4]. We can then set β as

[4] This holds if the size of range of h is exponentially larger than its image space. It is also worth noting that τ does not need to be efficiently computable, because our soundness proof (or the editing technique) is only an existential argument.

before to ensure that the $(x_1, \ldots, x_n)$ part is "edited" properly. However, note that the y^* here contains additionally a t^* value. To handle this, we modify the construction of H_z^f slightly—We require that, when the **editing** condition is triggered, H_z^f sets $\mathsf{t} = \mu$ in its output y. With this change, when performing editing, we can simply let μ equal the t^*. It is not hard to verify that this editing technique will lead to a valid preimage for y^*.

Finally, we remark that our real construction uses a Merkle tree for the prefix $(x_1, \ldots, x_n)$ of x. We only put the Merkle root in y, instead of the element-wise hash values described above. The soundness can be proved following essentially the same idea as above, except that we now "edit" the Merkle tree, which is done by extending the editing ideas to the tree setting. This allows us to compress a prefix of any length to a *fixed-length* string, such as 256 bits if using SHA256 for h. We refer the reader to Sec. 6 for a formal treatment of PB-CRHF.

2.4 Supporting Predicates

We discuss how to extend our constructions using "MPC-in-the-head" to additionally guarantee not only that the output learned by the receiver is in the range of the deterministic primitives, but also that the set of preimages contains one whose prefix satisfies some predicate ϕ.

Let us take a fresh look at the PB-OWF construction. It first parses the input as $\mathsf{x} = \alpha \| \beta$. The β is for preimage editing; and the $\alpha = (x_1, \ldots, x_n)$ can be regarded as a form of **Encoding** the prefix of x, i.e. $\mathsf{Enc}(\alpha) = (x_1, \ldots, x_n)$. Then, it computes $y_i = f(x_i)$ for all $i \in [n]$. Since this is mainly to introduce hardness (or one-wayness) to the final output, we can refer to this step as **Hardness Inducing**.

To support the proof of a predicate ϕ, we update the construction with new **Encoding** and **Hardness Inducing** methods. We first secret-share α to $([\alpha]_1, \ldots, [\alpha]_n)$ using a verifiable secret sharing (VSS) scheme. This can be viewed as a new encoding method: $\mathsf{Enc}(\alpha) = \mathsf{VSS}(\alpha) = ([\alpha]_1, \ldots, [\alpha]_n)$.

Next, we commit to these shares using Naor's commitment [38], which can be built in black-box from the oracle OWF f. This can be thought of as a new **Hardness Inducing** method. Now, the output of F^f is of the following form:

$$F^f(\mathsf{x}, \mathsf{r}) = (\mathsf{Com}([\alpha]_1), \ldots, \mathsf{Com}([\alpha]_n)) \| ([\alpha]_{b_1}, \ldots, [\alpha]_{b_t}) \| \mathsf{r}.$$

In the protocol Π_F^f, we additionally ask the sender to compute the value $\phi(\alpha)$ using the MPC-in-the-head technique. That is, the sender imagines n virtual parties $\{P_i\}_{i \in [n]}$, where P_i has $[\alpha]_i$ as its input. These n parties then execute a MPC protocol w.r.t. to the ideal functionality, which recovers α from the VSS shares, and outputs $\phi(\alpha)$ to each party. Let v_i denote the view of party i from the execution. The sender first commits to these views, and then opens some of them (picked by the receiver) for the receiver to check that the MPC for $\phi(\alpha)$ was performed honestly. In this way, the receiver not only learns $\phi(\alpha)$, but also believes that the sender did not cheat.

Finally, we make a few remarks:

– To achieve soundness, we also need to apply the preimage editing idea to the above construction.
– Both the VSS and Com require randomness, which can come from x. That is, we require that the x is long enough such that it also contains an η part (in addition to α and β). This η will provide the randomness for VSS and Com.
– The above approach applies directly to the PB-PRG and PB-CRHF constructions to make them support predicates on the α part of the preimage.

3 Preliminaries

Familiarity with basic cryptographic concepts such as ensembles, indistinguishability, and interactive Turing machines, etc. are assumed; we refer to [12, 13] for formal treatments of these. We also provide additional preliminaries in the full version [35].

Notations. We use "\" to denote set difference. That is, for any two sets A and B, $A \setminus B \coloneqq \{x : (x \in A) \wedge (x \notin B)\}$. The security parameter is denoted by λ. Symbols $\overset{\text{c}}{\approx}$, $\overset{\text{s}}{\approx}$ and $\overset{\text{i.d.}}{=\!=}$ are used to denote computational, statistical, and perfect indistinguishability respectively; and $\mathsf{negl}(\lambda)$ denotes negligible functions of λ. For a distribution D, $x \leftarrow D$ means that x is sampled according to D. Unless emphasized otherwise, we assume uniform distribution by default. We use $y \in D$ to mean that y is in the support of D. For a set S we overload the notation by using $x \leftarrow S$ to indicate that x is chosen uniformly at random from S. PPT denotes probabilistic polynomial time.

Let p be a predicate and $D_1, D_2, \ldots$ probability distributions, then the notation $\Pr\left[x_1 \leftarrow D_1; x_2 \leftarrow D_2; \ldots : p(x_1, x_2, \ldots)\right]$ denotes the probability that $p(x_1, x_2, \ldots)$ holds after the ordered execution of the probabilistic assignments $x_1 \leftarrow D_1; x_2 \leftarrow D_2; \ldots$. The notation $\{x_1 \leftarrow D_1; x_2 \leftarrow D_2; \ldots : p(x_1, x_2, \ldots)\}$ denotes the new probability distribution over $\{(x_1, x_2, \ldots)\}$.

Black-Box Zero-Knowledge Commit-and-Prove. We need a zero-knowledge commit-and-prove protocol Π_{ZKCnP} with the following additional properties:
– it consists of two *separate* phases: a **Commit** phase BBCom and a **Prove** phase $\mathsf{BBProve}$;
– the **Commit** phase itself constitutes a statistically-binding commitment scheme;
– for a public predicate $\phi(\cdot)$, the **Prove** phase constitutes a zero-knowledge argument for the value $\phi(x)$, where x is the value committed in BBCom;
– Π_{ZKCnP} can be constructed assuming only black-box access to OWFs.
A formal definition can be found in the full version [35]. There exist constructions satisfying the above requirements (e.g., [27, 18, 2]).

The "One-Oracle" Separation Technique. We first recall in Def. 1 the notion of *fully-black-box* reductions. We say that P cannot be obtained from Q in a fully-black-box way if there is no fully-black-box reduction from Q to P.

Definition 1 (Fully-Black-Box reductions [40]). *There exists a fully-black-box reduction from a primitive Q to a primitive P, if there exist PPT oracle machines G and S such that:*

– **Correctness:** *For every (possibly-inefficient) f that implements P, G^f implements Q;*
– **Security:** *For every (possibly-inefficient) f that implements P and every (possibly-inefficient) machine $\mathcal{A}$, if $\mathcal{A}$ breaks G^f (w.r.t. Q-security), then $S^{\mathcal{A},f}$ breaks f (w.r.t. P-security).*

A paradigm to rule out fully-black-box constructions is to design an oracle $\mathcal{O}$, and show that, relative to $\mathcal{O}$, primitive P exists but Q does not. A critical step in this proof is to construct an oracle machine $\mathcal{A}^{\mathcal{O}}$ that breaks the security of Q. We emphasize that $\mathcal{A}$ is allowed to be *computationally unbounded*, as long as it only makes polynomially-many queries to $\mathcal{O}$ (see e.g., [25, 1]). Our fully-black-box separation results in Sec. 4 will follow this paradigm.

4 The Impossibility Results

4.1 Meta-Functionally Black-Box Constructions

Functionally Black-Box Protocols. To capture MPC protocols that "do not know" the code of the target function g, Rosulek [41] proposes the following notion of functionally-black-box protocols.

Definition 2 (Functionally-Black-Box Protocols [41]). *Let $\mathcal{C}$ be a class of functions, and let $\mathcal{F}^{(\cdot)}$ be an ideal functionality that is an (uninstantiated) oracle machine. Let $A^{(\cdot)}$ and $B^{(\cdot)}$ be PPT interactive oracle machines. Then, we say that $(A^{(\cdot)}, B^{(\cdot)})$ is a functionally-black-box (FBB) protocol for $\mathcal{F}^{\mathcal{C}}$ in a certain security model if, for all $g \in \mathcal{C}$, the protocol (A^g, B^g) is a secure protocol (in the model in question) for the ideal functionality $\mathcal{F}^g$.*

By instantiating $\mathcal{C}$ and $\mathcal{F}^{(\cdot)}$ properly, Def. 2 could capture black-box constructions of many useful cryptographic protocols. For example, let $\mathcal{C}_{\text{OWF}}$ be the collection of OWFs. For any $g \in \mathcal{C}_{\text{OWF}}$, let $\mathcal{F}_{\text{ZK}}^g$ be the functionality that takes input x from party A, queries its oracle g to obtain $y = g(x)$, and outputs y to party B. Such an $\mathcal{F}_{\text{ZK}}^g$ is essentially a zero-knowledge argument (of knowledge) functionality for statements of the form "$\exists x$ s.t. $g(x) = y$". However, Rosulek showed that if injective OWFs exist, then it is impossible to have FBB protocols that implement $\mathcal{F}_{\text{ZK}}^{\mathcal{C}_{\text{OWF}}}$ with semi-honest security (in the standard MPC setting), even in the presence of an arbitrary trusted setup. Given the broad application of ZK proofs, this result is quite discouraging.

Meta-FBB Functionalities. Observe that the above $\mathcal{F}_{\text{ZK}}^g$ functionality simply collects input x from A, queries its oracle g, and sends $g(x)$ to B. It only plays the role of a delegate for A and B to interact with the OWF g. Therefore, it is temping to investigate whether we can circumvent Rosulek's lower bound by allowing the "delegate" $\mathcal{F}_{\text{ZK}}$ to perform extra computations, such as preprocessing x, post-processing $g(x)$, or making multiple queries to the oracle g, etc.

More formally, we want a non-cryptographic and deterministic computation F (used to capture the aforementioned extra computations), such that $\mathcal{C}'_{\text{OWF}} =$

$\{F^g \mid g \in \mathcal{C}_{\text{OWF}}\}$ is a collection of OWFs. And we hope that there exists a FBB protocol (A^g, B^g) implementing $\mathcal{F}_{\text{ZK}}^{F^g}$ for all $F^g \in \mathcal{C}'_{\text{OWF}}$ (we can also denote it as $\mathcal{F}_{\text{ZK}}^{\mathcal{C}'_{\text{OWF}}}$). Note that we require $(A^{(\cdot)}, B^{(\cdot)})$ to access g in a black-box way only; they can make use the code of F. Since $\mathcal{C}'_{\text{OWF}}$ is also a collection of one-way families, $\mathcal{F}_{\text{ZK}}^{\mathcal{C}'_{\text{OWF}}}$ can be used as a substitute for $\mathcal{F}_{\text{ZK}}^{\mathcal{C}_{\text{OWF}}}$, with the only overhead coming from the computations represented by $F^{(\cdot)}$. Because $F^{(\cdot)}$ is supposed to contain only simple non-cryptographic operations, the implementation of $\mathcal{F}_{\text{ZK}}^{\mathcal{C}'_{\text{OWF}}}$ should be as efficient as that of $\mathcal{F}_{\text{ZK}}^{\mathcal{C}_{\text{OWF}}}$. Therefore, if this approach is possible, it will alleviate the negative implications of Rosulek's lower bound.

We can also interpret $\mathcal{F}_{\text{ZK}}^{\mathcal{C}'_{\text{OWF}}}$ as a new FBB functionality $\mathcal{F}_{\text{ZK}}^{\mathcal{C}_{\text{OWF}}}[F]$, i.e., a new oracle machine $\mathcal{F}_{\text{ZK}}^{(\cdot)}[F]$ to be instantiated with oracle OWFs from the original collection $\mathcal{C}_{\text{OWF}}$. For any $g \in \mathcal{C}_{\text{OWF}}$, $\mathcal{F}_{\text{ZK}}^{g}[F]$ collects the input X from Party A, evaluates $F^g(X)$, and sends $y = F^g(X)$ to Party B.

With this interpretation, $\mathcal{F}_{\text{ZK}}^{\mathcal{C}'_{\text{OWF}}}$ is just an instantiation of Def. 2 with $\mathcal{F}^{(\cdot)} = \mathcal{F}_{\text{ZK}}^{(\cdot)}[F]$ and $\mathcal{C} = \mathcal{C}_{\text{OWF}}$. To distinguish with Rosulek's $\mathcal{F}_{\text{ZK}}^{(\cdot)}$ functionality. We call $\mathcal{F}_{\text{ZK}}^{(\cdot)}[F]$ the *Meta-FBB* ZK Functionality. Similarly, one can also extend other FBB functionalities in [41] (e.g., 2-party secure function evaluation $\mathcal{F}_{\text{SFE}}^{(\cdot)}$, pseudo-random generator $\mathcal{F}_{\text{PRG}}^{(\cdot)}$, where sender A holds the seed and receiver B holds the key) to the corresponding Meta-FBB version.

4.2 The Main Theorem

In this part, we show that although we relax Rosulek's FBB notion to the Meta-FBB one, there still exists strong impossibility result. More specifically, we prove that, given only black-box access to OWFs, it is impossible to build a PRG that admits Meta-FBB honest-verifier zero-knowledge protocols.

Definition 3 (Fully-Black-Box PRGs from OWFs). *Let $\mathcal{C}$ be the collection of OWFs. A (deterministic) polynomial-time oracle machines $G^{(\cdot)}$ is a* fully-black-box *construction of PRG from OWF if there exists a PPT oracle machines $\mathcal{A}^{(\cdot,\cdot)}$ such that:*

- **Correctness:** *$\forall f \in \mathcal{C}$, G^f is a PRG;*
- **Security:** *$\forall f \in \mathcal{C}$ and every (possibly inefficient) machine M, if M breaks the pseudo-randomness of G^f, then $\mathcal{A}^{M,f}$ breaks the one-wayness of f.*

Theorem 2 (Main Theorem). *Let $\mathcal{C} = \{f \mid f \text{ is a OWF}\}$. There does not exist a (deterministic) oracle machine $G^{(\cdot)}$ such that*

1. *$G^{(\cdot)}$ is a fully-black-box construction of PRG from OWF;* **and**
2. *for all $f \in \mathcal{C}$, there exists a stand-alone, Meta-FBB, honest-verifier zero-knowledge argument system $\Pi^f = \langle P^f, V^f \rangle$ for the functionality $\mathcal{F}_{\text{ZK}}^{f}[G]$.*

Before showing the full proof in Sec. 4.3, let us provide the high-level idea.

Proof Sketch. We start by assuming (for contradiction) that the $G^{(\cdot)}$ and $\Pi^{(\cdot)}$ specified in the theorem exist. We will construct a special oracle denoted as $\mathsf{O} \diamond Q^{\mathsf{Easy}}$ (explained later) such that:

1. The oracle $\mathsf{O}\diamond Q^{\mathsf{Easy}}$ is one-way. Thus, $G^{\mathsf{O}\diamond Q^{\mathsf{Easy}}}$ will be a PRG and $\Pi^{\mathsf{O}\diamond Q^{\mathsf{Easy}}}$ will be the HVZK system for the language $\mathcal{L} = \{Y : \exists X \ s.t. \ Y = G^{\mathsf{O}\diamond Q^{\mathsf{Easy}}}(X)\}$.
2. There exist a $\ddot{Y} \notin \mathcal{L}$ (the false statement) and a $\mathbb{P}^{\mathsf{O}\diamond Q^{\mathsf{Easy}}}$ (the cheating prover $\mathbb{P}$ with the oracle $\mathsf{O} \diamond Q^{\mathsf{Easy}}$) that is able to make $V^{\mathsf{O}\diamond Q^{\mathsf{Easy}}}(\ddot{Y})$ accept.

This will give us the desired contradiction as it breaks the soundness of the protocol $\Pi^{\mathsf{O}\diamond Q^{\mathsf{Easy}}}$.

Toward the above goal, we first sample two random oracles O, O', a random string X, and compute $Y = G^{\mathsf{O}}(X)$. Let $Q = \{(q_1, \mathsf{O}(q_1)), \ldots, (q_t, \mathsf{O}(q_t))\}$ denote the query-answer pairs exchanged between G and its oracle O during computation $Y = G^{\mathsf{O}}(X)$. We now define the oracle $\mathsf{O}'\diamond Q(q) \coloneqq \begin{cases} \mathsf{O}(q) & \text{if } (q, \mathsf{O}(q)) \in Q \\ \mathsf{O}'(q) & \text{otherwise} \end{cases}$. It is not hard to verify that $Y = G^{\mathsf{O}'\diamond Q}(X)$. By completeness, V will accept with probability $1 - \delta_c$ (where δ_c is the completeness error) in the execution $\mathsf{Exec}^{\mathsf{O}'\diamond Q}_{X,Y} = \langle P^{\mathsf{O}'\diamond Q}(X, Y), V^{\mathsf{O}'\diamond Q}(Y)\rangle$.

Note that during $\mathsf{Exec}^{\mathsf{O}'\diamond Q}_{X,Y}$, the verifier may make queries to its oracle $\mathsf{O}'\diamond Q$. We define a set of "easy" queries:

$$Q^{\mathsf{Easy}} \coloneqq \{(q, \mathsf{O}(q)) \,\big|\, V \text{ queries } q \text{ with "high" probability during } \mathsf{Exec}^{\mathsf{O}'\diamond Q}_{X,Y}\}.$$

Let Q^{Hard} be the set difference $Q \setminus Q^{\mathsf{Easy}}$. It is not hard to see that $Y = G^{\mathsf{O}'\diamond(Q^{\mathsf{Easy}}\cup Q^{\mathsf{Hard}})}(X)$. By completeness, V will accepts with probability $1 - \delta_c$ in the execution $\mathsf{Exec}^{\mathsf{O}'\diamond(Q^{\mathsf{Easy}}\cup Q^{\mathsf{Hard}})}_{X,Y}$.

Now, consider the execution $\langle P^{\mathsf{O}'\diamond(Q^{\mathsf{Easy}}\cup Q^{\mathsf{Hard}})}(X, Y), V^{\mathsf{O}'\diamond Q^{\mathsf{Easy}}}(Y)\rangle$, which is identical to $\mathsf{Exec}^{\mathsf{O}'\diamond(Q^{\mathsf{Easy}}\cup Q^{\mathsf{Hard}})}_{X,Y}$ except that we remove the Q^{Hard} from the verifier's oracle. In this execution, the probability that V accepts will not differ too much from that in $\mathsf{Exec}^{\mathsf{O}'\diamond(Q^{\mathsf{Easy}}\cup Q^{\mathsf{Hard}})}_{X,Y}$, because the queries in Q^{Hard} are asked by V with only "low" probability.

We then prove that Y is in the range of $G^{\mathsf{O}'\diamond Q^{\mathsf{Easy}}}(\cdot)$ with probability at most 0.5 (up to negligible error). But the previous argument says that $V^{\mathsf{O}'\diamond Q^{\mathsf{Easy}}}(Y)$ accepts with probability close to 1. It then follows from an averaging argument that there exists "bad" $\ddot{\mathsf{O}}$, $\ddot{\mathsf{O}}'$ and $\ddot{X}$[5] such that $\ddot{Y} = G^{\ddot{\mathsf{O}}}(\ddot{X})$ is *not* in the range of $G^{\ddot{\mathsf{O}}'\diamond \ddot{Q}^{\mathsf{Easy}}}(\cdot)$, but $V^{\ddot{\mathsf{O}}'\diamond \ddot{Q}^{\mathsf{Easy}}}(\ddot{Y})$ can be convinced with probability close to 1, by the malicious prover $P^{\ddot{\mathsf{O}}'\diamond(\ddot{Q}^{\mathsf{Easy}}\cup \ddot{Q}^{\mathsf{Hard}})}(\ddot{X}, \ddot{Y})$ (which can be viewed as an oracle machine $\mathbb{P}^{\ddot{\mathsf{O}}'\diamond \ddot{Q}^{\mathsf{Easy}}}$ with non-uniform advice $\ddot{X}$, $\ddot{Y}$, and $\ddot{Q}^{\mathsf{Hard}}$). This breaks the soundness of $\Pi^{\ddot{\mathsf{O}}'\diamond \ddot{Q}^{\mathsf{Easy}}}$, thus completing the proof.

We remark that proving Y is in the range of $G^{\mathsf{O}'\diamond Q^{\mathsf{Easy}}}(\cdot)$ with probability ≤ 0.5 (up to negligible error) is the most involved part. And this is where the HVZK property of $\Pi^{(\cdot)}$ plays an essential role. Roughly, we will show that if this claim does not hold, then there exists an adversary $\mathcal{A}^{\mathsf{O}}_{\mathrm{PRG}}$ that can break the

[5] Note that these values already determine the sets $\ddot{Q}$, $\ddot{Q}^{\mathsf{Easy}}$, and $\ddot{Q}^{\mathsf{Hard}}$ as defined above.

pseudo-randomness of $G^{\mathsf{O}}(\cdot)$ by making *polynomially* many oracle queries. As we will explain later, this reduction requires $\mathcal{A}^{\mathsf{O}}_{\text{PRG}}$ to know the set Q^{Easy} w.r.t. the challenge string Y in the security game of PRG. But note that $\mathcal{A}^{\mathsf{O}}_{\text{PRG}}$ does not know the preimage X (if Y is indeed in the range), which is necessary to figure out Q^{Easy}. This is where the HVZK simulator comes to our rescue. We will run the simulator $\mathsf{Sim}^{\mathsf{O}}_V(Y)$ repeatedly for (polynomially) many times to get an estimate $\widetilde{Q}^{\mathsf{Easy}}$ for the set Q^{Easy}. This $\widetilde{Q}^{\mathsf{Easy}}$ will be good enough to finish our proof. A more detailed overview of this strategy is provided in Sec. 4.4.

4.3 Proof of Thm. 2

Assume for contradiction that there exists an oracle machine $G^{(\cdot)}$ and a protocol $\langle P^{(\cdot)}, V^{(\cdot)}\rangle$ such that given the access to any one-way function $\{f_n\}_{n\in\mathbb{N}}$:

1. $G^{f_n} : \{0,1\}^{\ell} \to \{0,1\}^{\ell+1}$ is a PRG (ℓ and n are polynomially related); **and**
2. $\Pi = \langle P^{f_n}, V^{f_n}\rangle$ is a semi-honest zero-knowledge argument system for the Meta-FBB functionality $\mathcal{F}^{f_n}_{\text{ZK}}[G]$.

We first recall the following lemma, which says that the measure-one of randomly-sampled oracles is one-way.

Lemma 2 (One-Wayness of Random Oracles [25, 44]). *Let $\mathcal{O} = \{\mathsf{O}_n\}_{n\in\mathbb{N}}$ be a collection of oracles where each O_n is chosen uniformly from the space of functions from $\{0,1\}^n$ to $\{0,1\}^n$. With probability 1 over the choice of $\mathcal{O}$, $\mathcal{O}$ is one-way against unbounded adversaries that make only polynomially many oracle queries to $\mathcal{O}$.*

Let both $\mathcal{O} = \{\mathsf{O}_n\}_{n\in\mathbb{N}}$ and $\mathcal{O}' = \{\mathsf{O}'_n\}_{n\in\mathbb{N}}$ be defined (independently) as in Lem. 2. It follows from Lem. 2 that, with probability 1, both $\mathcal{O}$ and $\mathcal{O}'$is one-way.

In the following, we show two hybrids. From the second hybrid, we will construct a malicious prover breaking the soundness of $\Pi^{(\cdot)}$ (with the oracle being instantiated by a special one-way oracle defined later). This will give us the desired contradiction, and thus will finish the proof of Thm. 2.

Notations. We first define some notations. For an oracle H and a set of tuples $\mathsf{S} = \{(q_1, a_1), \ldots, (q_t, a_t)\}$, we define a new oracle $\mathsf{H} \diamond \mathsf{S}$ as follows: if q equals some q_i for which there exists a pair (q_i, a_i) in the set S, the oracle $\mathsf{H} \diamond \mathsf{S}$ returns a_i; otherwise, it returns $\mathsf{H}(q)$. Formally,

$$\mathsf{H} \diamond \mathsf{S}(q) = \begin{cases} \mathsf{H}(q) & \text{if } q \notin \{q_1, \ldots, q_t\} \\ a_i & \text{if } q = q_i \in \{q_1, \ldots, q_t\} \end{cases}$$

Hybrid H_0. This hybrid samples $X_n \leftarrow \{0,1\}^{\ell(n)}$, and computes $Y_n = G^{\mathsf{O}_n}(X_n)$. W.l.o.g., we assume that G on input X_n makes $t(n)$ *distinct* queries to its oracle O_n, where $t(n)$ is a polynomial of n. Let $Q_n = \big\{\big(q_1, \mathsf{O}_n(q_1)\big), \ldots, \big(q_t, \mathsf{O}_n(q_t)\big)\big\}$ be the query-answer pairs during the computation $Y_n = G^{\mathsf{O}_n}(X_n)$.

Let $\mathsf{Exec}^{\mathsf{O}'_n \diamond Q_n}_{X_n,Y_n} = \langle P^{\mathsf{O}'_n \diamond Q_n}(X_n, Y_n), V^{\mathsf{O}'_n \diamond Q_n}(Y_n)\rangle$ denote the execution where P proves to V that there exists an X_n such that $Y_n = G^{\mathsf{O}'_n \diamond Q_n}(X_n)$. Note that during this execution, the verifier may query to its oracle $\mathsf{O}'_n \diamond Q_n$. For each $q_i \in \{0,1\}^n$, let p_i denote the probability that q_i is queried by V during $\mathsf{Exec}^{\mathsf{O}'_n \diamond Q_n}_{X_n,Y_n}$. Let Q_n^{Easy} defines the set of "easy" queries and their corresponding answers:

$$Q_n^{\mathsf{Easy}} := \left\{ \left(q_i, \mathsf{O}'_n \diamond Q_n(q_i)\right) \mid p_i \geq \frac{1}{t(n) \cdot n} \text{ during } \mathsf{Exec}^{\mathsf{O}'_n \diamond Q_n}_{X_n,Y_n} \right\}. \tag{1}$$

Let Q_n^{Hard} be the set difference $Q_n \setminus Q_n^{\mathsf{Easy}}$. We remark that Q_n and $Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}}$ may not be the same, but it must hold that $Q_n \subseteq Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}}$.

Looking ahead, we will instantiated $G^{(\cdot)}$ and $\Pi^{(\cdot)}$ with the oracle $\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})$. Note that $G^{(\cdot)}$ and $\Pi^{(\cdot)}$ will have the desired property only if they are instantiated with *one-way* functions. Therefore, we show in Claim 3 that the composed oracle $\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})$ is one-way. It is worth noting that the one-wayness of this composed oracle is independent of the choice of $\{X_n\}$, though the definition of Q_n, Q_n^{Easy} and Q_n^{Hard} depends on X_n.

Claim 3. *The collection of oracles* $\left\{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})\right\}_{n\in\mathbb{N}}$ *defined above is one-way with probability 1, where the probability is taken over the sampling of* $\mathcal{O} = \{\mathsf{O}_n\}_n$ *and* $\mathcal{O}' = \{\mathsf{O}'_n\}_n$*, and is* independent *of the distribution of* $\{X_n\}_{n\in\mathbb{N}}$.

Proof. The query-answer pairs in Q_n^{Easy} and Q_n^{Hard} are of the form $\big(q, \mathsf{O}_n(q)\big)$ or $\big(q, \mathsf{O}'_n(q)\big)$. Although X_n decides which $(q, *)$[6] will be in Q_n^{Easy} and Q_n^{Hard}, the answer part $\mathsf{O}_n(q)$'s and $\mathsf{O}'_n(q)$'s are uniformly distributed, *independent* of X_n. That is, if O_n and O'_n are sampled randomly, then for any $X_n \in \{0,1\}^{\ell(n)}$, $\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})$ will also be a random oracle. Therefore, for *any* $\{X_n\}_{n\in\mathbb{N}}$ where $X_n \in \{0,1\}^{\ell(n)}$, the following holds

$$\left\{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})\right\}_{n\in\mathbb{N}} \overset{\text{i.d.}}{\equiv} \{\mathsf{O}''_n\}_{n\in\mathbb{N}},$$

where each O''_n is sampled uniformly from the space of functions from $\{0,1\}^n$ to $\{0,1\}^n$. Since it follows from Lem. 2 that $\{\mathsf{O}''_n\}_{n\in\mathbb{N}}$ is one-way with probability 1, so is $\left\{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})\right\}_{n\in\mathbb{N}}$. □

Claim 3 (together with our assumption) implies that, with probability 1 taken over the sampling of $\mathcal{O}$ and $\mathcal{O}'$:

- $G^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})} : \{0,1\}^{\ell(n)} \to \{0,1\}^{\ell(n)+1}$ is pseudo-random against all (unbounded) adversaries that make polynomially many queries to the oracle $\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})$; **and**
- $\Pi^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}$ is a semi-honest zero-knowledge argument system for the Meta-FBB functionality $\mathcal{F}_{\mathrm{ZK}}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}[G]$.

[6] The symbol "$*$" denotes the wildcard that matches any answer to q.

Let $\mathsf{Exec}^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}_{X_n,Y_n}$ denote the execution $\langle P^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}(X_n,Y_n), V^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}(Y_n)\rangle$. Let $\mathsf{Exec}^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}_{X_n,Y_n} = 1$ denote the event that the verifier accepts at the end of this execution. Claim 3 and the completeness of $\Pi^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}$ imply that:

$$\Pr_{\mathcal{O},\mathcal{O}'}\left[\begin{array}{l}\text{For sufficient large } n\in\mathbb{N}, \forall X_n\in\{0,1\}^{\ell(n)}, Y_n = G^{\mathsf{O}_n}(X_n),\\ \Pr\left[\mathsf{Exec}^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}_{X_n,Y_n}=1\right]\geq 1-\delta_c(n)\end{array}\right]=1, \tag{2}$$

where the inner probability is taken over the random coins of the prover and the verifier during $\mathsf{Exec}^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}_{X_n,Y_n}$, and $\delta_c(n)$ is the completeness error.

Hybrid H_1. This hybrid is identical to the previous one, except that H_1 executes the protocol

$$\langle P^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}(X_n,Y_n), V^{\mathsf{O}'_n\diamond Q_n^{\mathsf{Easy}}}(Y_n)\rangle. \tag{3}$$

(Compared with the execution $\mathsf{Exec}^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}_{X_n,Y_n}$ in H_0, the only difference is that H_1 remove Q_n^{Hard} from the *verifier's* oracle.)

As mentioned in the **Proof Sketch** of Thm. 2, we want to show that the verifier accepts in Execution 3 with probability close to that in the execution $\mathsf{Exec}^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}_{X_n,Y_n}$. This is formalized as Claim 4.

Claim 4. *With probability 1 taken over the sampling of $\mathcal{O}$ and $\mathcal{O}'$, for sufficiently large $n\in\mathbb{N}$, it holds that $\forall X_n\in\{0,1\}^{\ell(n)}$ and $Y_n = G^{\mathsf{O}_n}(X_n)$,*

$$\begin{aligned}&\Pr\left[\langle P^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}(X_n,Y_n), V^{\mathsf{O}'_n\diamond Q_n^{\mathsf{Easy}}}(Y_n)\rangle = 1\right]\\ &\qquad\geq \Pr\left[\mathsf{Exec}^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}_{X_n,Y_n}=1\right]-\frac{1}{n},\end{aligned} \tag{4}$$

where the probabilities in the above inequality are taken over the random coins of the prover and the verifier during the corresponding executions.

Proof. First, we remark that the "with probability 1" part in this claim is to ensure that $\{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})\}_n$ is one-way (see Claim 3). In the following, we proceed with $\{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})\}_n$ being one-way (so the probabilities below are not taken over $\mathcal{O}$ and $\mathcal{O}'$).

By definition, any query[7] $q\in Q_n^{\mathsf{Hard}}$ is asked by V during $\mathsf{Exec}^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}_{X_n,Y_n}$ with probability $<\frac{1}{t(n)\cdot n}$. Let us denote the following event:

$\mathsf{Event}_{\mathsf{NoHard}}$: No $q\in Q_n^{\mathsf{Hard}}$ is asked by V in $\mathsf{Exec}^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}_{X_n,Y_n}$

It follows from union bound that

$$\Pr\left[\mathsf{Event}_{\mathsf{NoHard}}\right]\geq 1-\frac{1}{n}, \tag{5}$$

where the probability is taken over the random coins of P and V in the execution $\mathsf{Exec}^{\mathsf{O}'_n\diamond(Q_n^{\mathsf{Easy}}\cup Q_n^{\mathsf{Hard}})}_{X_n,Y_n}$.

Now, we prove Inequality (4). In the following, for succinctness, let

[7] Technically, elements in Q_n^{Hard} are query-answer pairs. From here on, we override the notation "$\in$" such that $q\in Q_n^{\mathsf{Hard}}$ means that there exists a pair $(q,*)$ in Q_n^{Hard}.

– Exec_0 denote the execution $\langle P^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}(X_n, Y_n), V^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}(Y_n)\rangle$;
– Exec_1 denote the execution $\mathsf{Exec}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}_{X_n, Y_n}$.
Then, we have (probabilities below are taken over the random coins over P and V in the corresponding executions):

$$\begin{aligned}
\Pr[\mathsf{Exec}_1 = 1] &\geq \Pr\left[\mathsf{Exec}_1 = 1 \,\middle|\, \mathsf{Event}_{\mathsf{NoHard}}\right] \cdot \Pr\left[\mathsf{Event}_{\mathsf{NoHard}}\right] \\
&= \Pr\left[\mathsf{Exec}_0 = 1 \,\middle|\, \mathsf{Event}_{\mathsf{NoHard}}\right] \cdot \Pr\left[\mathsf{Event}_{\mathsf{NoHard}}\right] && (6) \\
&\geq \Pr[\mathsf{Exec}_0 = 1] - \Pr[\neg \mathsf{Event}_{\mathsf{NoHard}}] && (7) \\
&\geq \Pr[\mathsf{Exec}_0 = 1] - \frac{1}{n} && (8)
\end{aligned}$$

where Step 6 is due to the fact that Exec_1 and Exec_0 are identical assuming V does not make any query $q \in Q_n^{\mathsf{Hard}}$, Step 7 follows from the basic probability inequality that $\Pr\left[A \,\middle|\, B\right] \cdot \Pr[B] \geq \Pr[A] - \Pr[\neg B]$, and Step 8 follows from Inequality (5).

This finishes the proof of Claim 4. □

Claim 4 indicates that the verifier in Execution 3 accepts with "good" probability: at least as large as the accepting probability of $\mathsf{Exec}^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}_{X_n, Y_n}$ minus $1/n$. Thus, we will have the desired contradiction if the Y_n in Execution 3 is a false statement, i.e. Y_n is not in the range of $G^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}$ (i.e. $G^{(\cdot)}$ instantiated by the verifier's oracle in Execution 3). This argument is formalized and proved in Claims 5 and 6, which will eventually finish the proof of Thm. 2.

Claim 5. *Let Q_n^{Easy} be defined as in Expression (1). For sufficiently large $n \in \mathbb{N}$, the following holds:*

$$\Pr_{\mathcal{O}, \mathcal{O}', X_n}\left[G^{\mathsf{O}_n}(X_n) \in G^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right)\right] \leq \frac{1}{2} + \mathsf{negl}(n). \tag{9}$$

Note that the above probability is taken (additionally) over $X_n \leftarrow \{0,1\}^{\ell(n)}$.

Claim 6. *If Claim 5 holds, then Thm. 2 holds.*

The proof of Claim 5 is quite involved. It constitutes the main technical challenge of the current proof (of Thm. 2). Thus, we will deal with it in Sec. 4.4. In the following, we show the proof of Claim 6.

Proof of Claim 6. It follows from Expression (2) and Claim 4 that

$$\Pr_{\mathcal{O}, \mathcal{O}'}\left[\begin{array}{l}\text{For sufficient large } n \in \mathbb{N}, \forall X_n \in \{0,1\}^{\ell(n)}, Y_n = G^{\mathsf{O}_n}(X_n), \\ \Pr\left[\langle P^{\mathsf{O}'_n \diamond (Q_n^{\mathsf{Easy}} \cup Q_n^{\mathsf{Hard}})}(X_n, Y_n), V^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}(Y_n)\rangle = 1\right] \geq 1 - \frac{1}{n} - \delta_c(n)\end{array}\right] = 1. \tag{10}$$

Following the same argument as for Claim 3, we can prove the one-wayness of the oracle $\{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}\}_n$ as follows. For each $(q, \mathsf{O}_n(q)) \in Q_n^{\mathsf{Easy}}$, the $\mathsf{O}_n(q)$ is a randomly sampled string from $\{0,1\}^n$. Therefore, no matter what X_n is,

$\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}$ is always a randomly sampled oracle (though Q_n^{Easy} is determined by X_n). It then follows from Lem. 2 that:

$$\Pr_{\mathcal{O},\mathcal{O}'}\left[\forall X_n \in \{0,1\}^{\ell(n)},\ \{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}\}_{n\in\mathbb{N}} \text{ is one-way}\right] = 1. \tag{11}$$

By an averaging argument over Expressions (9) to (11), it follows that there exists fixed sequences $\{\ddot{\mathsf{O}}_n\}_{n\in\mathbb{N}}$, $\{\ddot{\mathsf{O}}'_n\}_{n\in\mathbb{N}}$ and $\{\ddot{X}_n\}_{n\in\mathbb{N}}$[8] such that for sufficiently large $n \in \mathbb{N}$,

- $\{\ddot{\mathsf{O}}_n \diamond \ddot{Q}_n^{\mathsf{Easy}}\}_n$ is one-way; thus, $G^{\ddot{\mathsf{O}}_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$ is a PRG and $\Pi^{\ddot{\mathsf{O}}_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$ is an HVZK protocol for the membership of $G^{\ddot{\mathsf{O}}_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$; **and**
- $\ddot{Y}_n$ is not in the range of $G^{\ddot{\mathsf{O}}_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$; **and**
- $\Pr\left[\langle P^{\ddot{\mathsf{O}}'_n \diamond (\ddot{Q}_n^{\mathsf{Easy}} \cup \ddot{Q}_n^{\mathsf{Hard}})}(\ddot{X}_n, \ddot{Y}_n), V^{\ddot{\mathsf{O}}'_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}(\ddot{Y}_n)\rangle = 1\right] \geq 1 - \frac{1}{n} - \delta_c(n)$, where the probability is taken over the random coins of P and V.
  Note that we can treat $P^{\ddot{\mathsf{O}}'_n \diamond (\ddot{Q}_n^{\mathsf{Easy}} \cup \ddot{Q}_n^{\mathsf{Hard}})}(\ddot{X}_n, \ddot{Y}_n)$ as an oracle machine $\mathbb{P}^{\ddot{\mathsf{O}}'_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$, which has $(\ddot{Q}_n^{\mathsf{Easy}}, \ddot{X}_n, \ddot{Y}_n)$ as non-uniform advice and makes only polynomially many queries to its oracle $\ddot{\mathsf{O}}'_n \diamond \ddot{Q}_n^{\mathsf{Easy}}$.

Since the completeness error $\delta_c(\cdot)$ is negligible, the above means that $\mathbb{P}^{\ddot{\mathsf{O}}'_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$ (with its non-uniform advice) convinces the verifier with non-negligible probability on the following *false* statement:

$$\ddot{Y}_n \in G^{\ddot{\mathsf{O}}'_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right).$$

This contradicts the soundness of $\Pi^{\ddot{\mathsf{O}}'_n \diamond \ddot{Q}_n^{\mathsf{Easy}}}$. □

4.4 The Proof Sketch for Claim 5

Due to space constraints, we will present the formal proof for Claim 5 in the full version [35]. In this part, we provide an overview of it.

We assume for contradiction that Claim 5 is false and try to break the pseudorandomness of G^{O_n}. First, observe that if $Y_n = G^{\mathsf{O}_n}(X_n)$ where $X_n \leftarrow \{0,1\}^{\ell(n)}$, then our assumption implies that Y_n is in the range of $G^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}(\cdot)$ with probability noticeably larger than $1/2$. Therefore, on an input Y_n, if we can efficiently test if $Y_n \in G^{\mathsf{O}'_n \diamond Q_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right)$, we should have some advantage in the PRG game for $G^{\mathsf{O}_n}(\cdot)$. This strategy has the following potential problems:

1. Without the preimage X_n, we *cannot* compute the set Q_n^{Easy} (see Expression (1)) using only polynomially many queries to O_n;
2. If the input $Y_n \notin G^{\mathsf{O}_n}\left(\{0,1\}^{\ell(n)}\right)$, the set Q_n (thus Q_n^{Easy}) is not even well-defined, as there is no preimage X_n.

To avoid using X_n, we will run the HVZK simulator to obtain an estimate of the set Q_n^{Easy} in the following way. Recall that Q_n^{Easy} contains the "easy" queries made by the verifier during $\mathsf{Exec}^{\mathsf{O}'_n \diamond Q_n}_{X_n, Y_n}$. By the HVZK property of the

[8] Note that these values also fix the corresponding $\{\ddot{Y}_n\}_{n\in\mathbb{N}}$, $\{\ddot{Q}_n^{\mathsf{Easy}}\}_{n\in\mathbb{N}}$ and $\{\ddot{Q}_n^{\mathsf{Hard}}\}_{n\in\mathbb{N}}$ as in the above.

protocol $\Pi^{\mathsf{O}'_n \diamond Q_n}$, each query in Q_n^{Easy} should be made with similar probability in the simulated execution $\mathsf{Sim}_V^{\mathsf{O}'_n \diamond Q_n}(Y_n)$. Therefore, repeating $\mathsf{Sim}_V^{\mathsf{O}'_n \diamond Q_n}(Y_n)$ (polynomially) many times will give us a good estimate to Q_n^{Easy}.

However, without X_n, we cannot figure out the set Q_n, which is necessary if we want to run $\mathsf{Sim}_V^{\mathsf{O}'_n \diamond Q_n}(Y_n)$. Fortunately, by a similar argument as that for Claim 3, we can prove that the oracle $\{\mathsf{O}_n\}_n$ and $\{\mathsf{O}'_n \diamond Q_n\}_n$ are identically distributed, even given X_n and $Y_n = G^{\mathsf{O}_n}(X_n)$. Therefore, running $\mathsf{Sim}_V^{\mathsf{O}_n}(Y_n)$ will be just as good as running $\mathsf{Sim}_V^{\mathsf{O}'_n \diamond Q_n}(Y_n)$. Note that this also solves Problem 2, because the simulator still works when invoked on false statements.

Now, we can construct the PRG distinguisher $\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}(Y_n)$ as follows: on input Y_n, $\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}(Y_n)$ obtains an estimate $\widetilde{Q}_n^{\mathsf{Easy}}$ to Q_n^{Easy} by running $\mathsf{Sim}_V^{\mathsf{O}_n}(Y_n)$ for polynomially many times. It then samples a random function $\mathsf{O}'_n : \{0,1\}^n \to \{0,1\}^n$, and outputs 1 if $Y_n \in G^{\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}}\left(\{0,1\}^{\ell(n)}\right)$; otherwise, it outputs 0. Note that although sampling O' requires exponential time, $\mathcal{A}_{\mathrm{PRG}}^{\mathsf{O}_n}(Y_n)$ only makes *polynomially many* queries to the oracle O_n.

If $Y_n = G^{\mathsf{O}_n}(X_n)$ where $X_n \leftarrow \{0,1\}^{\ell(n)}$, then by our assumption $\mathcal{A}^{\mathsf{O}_n}(Y_n)$ outputs 1 with probability noticeably larger than $1/2$; if $Y_n \leftarrow \{0,1\}^{\ell(n)+1}$, then Y_n is independent of O_n. Moreover, using a similar argument as for Claim 3, we can prove that Y_n is independent of the oracle $\widetilde{Q}_n^{\mathsf{Easy}}$ (thus $\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}$). Since the function $G^{\mathsf{O}'_n \diamond \widetilde{Q}_n^{\mathsf{Easy}}}(\cdot)$ stretch by 1 bit, the random Y_n will be in its range with probability $1/2$. This means $\mathcal{A}^{\mathsf{O}_n}(Y_n)$ outputs 1 with probability exactly $1/2$.

This gives us the desired contradiction.

5 Proof-Based One-Way Functions

5.1 Definition

Definition 4 (Proof-Based OWFs). *Let $\lambda \in \mathbb{N}$ be the security parameter. Let $a(\cdot)$, $b(\cdot)$ and $c(\cdot)$ be polynomials. A* proof-based one-way function *consists of a function $F_\lambda : \{0,1\}^{a(\lambda)} \times \{0,1\}^{b(\lambda)} \to \{0,1\}^{c(\lambda)}$ and a protocol $\Pi = (S, R)$ of a pair of* PPT *machines. We use $(X, Y) \leftarrow \langle S(1^\lambda, \mathsf{x}), R(1^\lambda, \mathsf{r})\rangle$ to denote the execution of protocol Π where the security parameter is λ, the inputs to S and R are x and r respectively, and the outputs of S and R are X and Y respectively. Let $Y = \perp$ denote that R aborts in the execution. The following conditions hold:*

- **One-Wayness.** *The function $\{F_\lambda\}_\lambda$ is one-way in the following sense:*
  - Easy to compute: *for all $\lambda \in \mathbb{N}$ and all $(\mathsf{x}, \mathsf{r}) \in \{0,1\}^{a(\lambda)} \times \{0,1\}^{b(\lambda)}$, $F_\lambda(\mathsf{x}\|\mathsf{r})$ can be computed in polynomial time on λ.*
  - Hard to invert: *for any non-uniform PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall \mathsf{r} \in \{0,1\}^{b(\lambda)}$,*

$$\Pr\left[\mathsf{x} \leftarrow \{0,1\}^{a(\lambda)}, X^* \leftarrow \mathcal{A}\left(1^\lambda, F_\lambda(\mathsf{x}\|\mathsf{r})\right) : F_\lambda(\mathsf{x}\|\mathsf{r}) = F_\lambda(X^*)\right] \leq \mathsf{negl}(\lambda),$$

- **Completeness.** *The protocol Π computes the ideal functionality $\mathcal{F}_F$ defined in Fig. 1. Namely, $\forall \lambda \in \mathbb{N}$, $\forall \mathsf{x} \in \{0,1\}^{a(\lambda)}$ and $\forall \mathsf{r} \in \{0,1\}^{b(\lambda)}$, if $(X, Y) \leftarrow \langle S(1^\lambda, \mathsf{x}), R(1^\lambda, \mathsf{r})\rangle$, then $X = \mathsf{x}\|\mathsf{r}$ and $Y = F_\lambda(\mathsf{x}\|\mathsf{r})$.*

Figure 1: Functionality $\mathcal{F}_F$ for Proof-Based OWFs

The ideal functionality $\mathcal{F}_F$ interacts with a sender S and a receiver R. Upon receiving the input $\mathsf{x} \in \{0,1\}^{a(\lambda)}$ from S and $\mathsf{r} \in \{0,1\}^{b(\lambda)}$ from R, the functionality $\mathcal{F}_F$ sends $\mathsf{x}\|\mathsf{r}$ to S, and $F(\mathsf{x}\|\mathsf{r})$ to R.

– **Soundness.** *For every* PPT *machine* S^* *and every auxiliary input* $z \in \{0,1\}^*$, *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}\mathsf{r} \leftarrow \{0,1\}^{b(\lambda)};\\ (\cdot, Y) \leftarrow \langle S^*(1^\lambda, z), R(1^\lambda, \mathsf{r})\rangle\end{array} : \begin{array}{l}Y \neq \bot \textbf{ and}\\ \nexists \mathsf{x} \text{ s.t. } F_\lambda(\mathsf{x}\|\mathsf{r}) = Y\end{array}\right] \leq \mathsf{negl}(\lambda),$$

– **Zero-Knowledge.** *This property is defined by requiring* only *security against corrupted* R *in the ideal-real paradigm for 2PC w.r.t. the ideal functionality* $\mathcal{F}_F$ *in Fig. 1. Concretely, denote by* $\mathsf{REAL}_{\Pi,\mathcal{A}(z)}(1^\lambda, \mathsf{x}, \mathsf{r})$ *the random variable consisting of the output of* S *and the output of the adversary* $\mathcal{A}$ *controlling* R *in an execution of* Π, *where* x *is the input to* S *and* r *to* R. *Similarly, denote by* $\mathsf{IDEAL}_{\mathcal{F}_F,\mathsf{Sim}(z)}(1^\lambda, \mathsf{x}, \mathsf{r})$ *the corresponding output of* S *and* Sim *from the ideal execution.Then there exist a PPT simulator* Sim *such that for any PPT adversary* $\mathcal{A}$, $\forall \mathsf{x} \in \{0,1\}^{a(\lambda)}$, $\forall \mathsf{r} \in \{0,1\}^{b(\lambda)}$, *and* $\forall z \in \{0,1\}^*$, $\left\{\mathsf{REAL}_{\Pi,\mathcal{A}(z)}(1^\lambda, \mathsf{x}, \mathsf{r})\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{IDEAL}_{\mathcal{F}_F,\mathsf{Sim}(z)}(1^\lambda, \mathsf{x}, \mathsf{r})\right\}_{\lambda\in\mathbb{N}}$.

If the constructions of both F *and* Π *makes only black-box access to other primitives, we call this a black-box PB-OWF.*

5.2 Our Construction

Following the high-level idea described in Sec. 2.2, we show that PB-OWFs can be built assuming black-box access to OWFs.

Theorem 7 (Black-Box PB-OWFs from OWFs). *There exists a PB-OWF that satisfies Def. 4 and makes only black-box use of OWFs.*

Our construction consists of a one-way function F^f (Constr. 1) together with a protocol Π_F^f (Prot. 1). The construction relies on the following building blocks:
– a one-way function f;
– a zero-knowledge commit-and-prove protocol $\Pi_{\mathsf{ZKCnP}} = (\mathsf{BBCom}, \mathsf{BBProve})$. Such protocols can also be constructed assuming only black-box access to f.

Remark 1 (On the Parameters in Constr. 1). The choice of $t(\lambda) = \log^2(\lambda)$ is somewhat arbitrary. In fact, any $t(\lambda) = \omega(\log \lambda)$ works as long as $(n - k - t)$ is some positive polynomial of λ for sufficiently large λ. This is to ensure that we can prove one-wayness *and* $(1-\delta)^t$ is negligible on λ, which is needed when we prove soundness. We also remark that the role of r is to specify a size-t subset of $[n]$. The canonical way of mapping r to a size-t subset of $[n]$ may consume slightly less randomness than $|\mathsf{r}| = t\log(n)$. For simplicity, we forgo further discussion and assume that there is a deterministic bijection between $\{0,1\}^{t\log(n)}$ and all size-t subsets of $[n]$. Similarly, the $\{p_1, \ldots, p_k\}$ are interpreted as a size-k subset of $[n]$, though we assign each p_i a length of $\log(n)$.

Construction 1: One-Way Function F^f

Let $m(\lambda)$ and $n(\lambda)$ be polynomials on λ. Let $0 < \delta < 1$ be a constant, and $k(\lambda) = \delta n(\lambda)$. Let $t(\lambda) = \log^2(\lambda)$ (see Rmk. 1). Assume that $f : \{0,1\}^\lambda \to \{0,1\}^{m(\lambda)}$ is a one-way function. On input $\mathsf{x} \in \{0,1\}^{n\lambda+(\log(n)+m)k}$ and $\mathsf{r} \in \{0,1\}^{t\log(n)}$, F^f parses them as

$$\mathsf{x} = (x_1, \ldots, x_n) \| (p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k}), \text{ and } \mathsf{r} = (b_1, \ldots, b_t),$$

where $|x_i| = \lambda$, $|y'_{p_i}| = m$, $\{p_i\}_{i\in[k]}$ is a size-k subset of $[n]$, and $\{b_i\}_{i\in[t]}$ is a size-t subset of $[n]$. F^f computes via its oracle access to $f(\cdot)$ the values $(y_1, \ldots, y_n)$, where $y_i = f(x_i)$ for all $i \in [n]$. Then, it computes $\mathsf{s} = (s_1, \ldots, s_n)$ as follows:

1. if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} \neq \emptyset$, then let $s_i \coloneqq y_i$ for all $i \in [n]$.
2. if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$, then let $s_i \coloneqq \begin{cases} y'_i & i \in \{p_1, \ldots, p_k\} \\ y_i & i \in [n] \setminus \{p_1, \ldots, p_k\} \end{cases}$.

It finally outputs $Y = (s_1, \ldots, s_n) \| (x_{b_1}, \ldots, x_{b_t}) \| (b_1, \ldots, b_t)$.

Proof of Security. Due to space constraints, the complete proof for that (F^f, Π^f) satisfies Def. 4 will appear in the full version [35].

Note that Sec. 2.2 already contains the high-level idea for this proof. The one-wayness, completeness and ZK property follow from rather standard techniques. In the following, let us provide more details about the soundness proof.

First, note that the $\mathsf{r} = \{b_1, \ldots, b_t\}$ sent by R in Stage 3 is a size-t random subset of $[n]$. It will overlap with $\{p_1, \ldots, p_k\}$ with negligible probability. Therefore, the **Editing** condition will almost never be triggered during a real execution of Prot. 1, thus can be safely ignored.

Stages 2 to 5 can be though as the following cut-and-choose procedure: the sender computes $\{y_i = f(x_i)\}_{i\in[n]}$; then the receiver checks t of them randomly. This ensures that a malicious S^* cannot cheat on more than $k = \delta n$ of the y_i's. We prove this statement formally in the full version [35], which requires us to handle extra technicalities due to the commit-and-prove structure and **Editing** condition. But this claim implies that a non-aborting Y output by an honest receiver contains at most $k = \delta n$ many s_i's that does *not* have a preimage under f (except for negligible probability). Let us assume w.l.o.g. that there are exactly k such "no-preimage" s_i's, which can be denoted as $\{s_{p_1}, \ldots, s_{p_k}\}$ (i.e. we denote the indices of these no-preimage s_i's by $\{p_1, \ldots, p_k\}$). Then, for each s_i where $i \in [n] \setminus \{p_1, \ldots, p_k\}$, this s_i must have (at least) one preimage under $f(\cdot)$. We denote an arbitrary preimage of such s_i as $f^{-1}(s_i)$. In particular, if i is equal to some $b_j \in \{b_1, \ldots, b_t\}$, the Y already contains the preimage for s_{b_j}, which is x_{b_j}.

We emphasize that, conditioned on $Y \neq \perp$, we have $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$. To see this, recall that R checks at Stage 5 that $y_{b_i} = f(x_{b_i})$ *and* $s_{b_i} = y_{b_i}$ for all $b_i \in \{b_1, \ldots, b_t\}$. If there is a p_i falling in the set $\{b_1, \ldots, b_t\}$, then s_{p_i} $(= y_{p_i})$ does not have a preimage under $f(\cdot)$. Then, R will output $Y = \perp$ at Stage 5.

With these observations, we show in the following how to construct x and r such that $F^f(\mathsf{x} \| \mathsf{r}) = Y$. At a high-level, we take advantage of Case 2. We will use the no-preimage s_{p_i}'s together with their indices as the (p_i, y'_{p_i}) part in x. We will set r to the $\{b_1, \ldots, b_t\}$ contained in Y. Since $\{b_1, \ldots, b_t\} \cap \{p_1, \ldots, p_k\} = \emptyset$,

Protocol 1: Protocol Π_F^f for Our Proof-Based One-Way Function

Let f, m, n, t, and k be as in Constr. 1.

Input: the security parameter 1^λ is the common input. Sender S takes $\mathsf{x} \in \{0,1\}^{n\lambda+(\log(n)+m)k}$ as its private input; receiver R takes $\mathsf{r} \in \{0,1\}^{t\cdot\log(n)}$ as its private input.

1. S parses the input as $\mathsf{x} = (x_1, \ldots, x_n) \| (p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k})$, where $|x_i| = \lambda$ for all $i \in [n]$, $|y'_{p_j}| = m$ for all $j \in [k]$, and $\{p_i\}_{i\in[k]}$ forms a size-k subset of $[n]$. S defines a $2 \times n$ matrix $M = \begin{bmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{bmatrix}$, where $y_i = f(x_i)$ for all $i \in [n]$.
2. S and R execute $\mathsf{BBCom}(\alpha)$, the **Commit** stage of Π_{ZKCnP}, where S commits to the value
$$\alpha \coloneqq M \| (p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k}). \tag{12}$$
3. R sends r to S.
4. S interprets r as a size-t subset $(b_1, \ldots, b_t) \subseteq [n]$. S then defines $M_\mathsf{r} = \begin{bmatrix} x_{b_1} & \cdots & x_{b_t} \\ y_{b_1} & \cdots & y_{b_t} \end{bmatrix}$, i.e. the columns of M specified by r. S also computes $\mathsf{s} = (s_1, \ldots, s_n)$ in the way specified in Constr. 1. S sends to R the values M_r and s.
5. With M_r, R checks (via its oracle access to $f(\cdot)$) if $f(x_{b_i}) = y_{b_i}$ holds for all $i \in [t]$; R also checks if $s_{b_i} = y_{b_i}$ holds for all $i \in [t]$. If all the checks pass, R proceeds to next step; otherwise, R halts and outputs $\perp$.
6. S and R execute $\mathsf{BBProve}$, the **Prove** stage of Π_{ZKCnP}, where S proves that it performs Stage 4 honestly. Namely, S proves that the α committed at Stage 2 satisfies the following conditions:
   (a) the values $\{p_1, \ldots, p_k\}$ contained in α form a size-k subset of $[n]$; **and**
   (b) the M_r does consist of the columns in M specified by r; **and**
   (c) The $\mathsf{s} = (s_1, \ldots, s_n)$ satisfies the following conditions:
   - if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} \neq \emptyset$, then $s_i = y_i$ for all $i \in [n]$.
   - if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$, then $s_i = \begin{cases} y'_i & i \in \{p_1, \ldots, p_k\} \\ y_i & i \in [n] \setminus \{p_1, \ldots, p_k\} \end{cases}$.

   We remark that these conditions can indeed be expressed a predicate ϕ on the α committed at Stage 2. For completeness, we show the formal definition of ϕ in Fig. 2. It is also worth noting that predicate ϕ needs to have the values r and s hard-wired, which are defined at Stages 3 and 4 respectively. This is why we need a Π_{ZKCnP} that allows us to defer the definition of the predicate until the Prove Stage.
7. **(Receiver's Output).** R outputs $Y = (s_1, \ldots, s_n) \| (x_{b_1}, \ldots x_{b_t}) \| \{b_i\}_{i\in[t]}$.
8. **(Sender's Output).** S outputs $X = (x_1, \ldots, x_n) \| (p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k}) \| \{b_i\}_{i\in[t]}$.

the function F^f will put the no-preimage s_{p_i}'s at the positions specified by p_i's (according to Case 2), which will give us Y. Concretely, we set:

$$\mathsf{x} = (x'_1, \ldots, x'_n) \| (p_1, s_{p_1}), \ldots, (p_k, s_{p_k}) \text{ and } \mathsf{r} = (b_1, \ldots, b_t),$$

where x'_i's are defined as follows: $\forall i \in [n],\ x'_i = \begin{cases} x_i & i \in \{b_1, \ldots, b_t\} \\ 0^\lambda & i \in \{p_1, \ldots, p_k\} \\ f^{-1}(s_i) & \text{otherwise} \end{cases}$.

Figure 2: Predicate $\phi_{\lambda,m,t,n,k,\mathsf{r},\mathsf{s}}(\cdot)$

Predicate ϕ has the values $(\lambda, m, t, n, k, \mathsf{r}, \mathsf{s})$ (defined in Prot. 1) hard-wired. On the input α, $\phi_{\lambda,m,t,n,k,r,s}(\alpha) = 1$ if and only if all of the following hold:

- the α can be parsed as $M \| (p_1, y'_{p_1}), \ldots, (p_k, y'_{p_k})$, where $M = \begin{bmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{bmatrix}$ such that $|x_j| = \lambda$ and $|y_j| = m$ $\forall j \in [n]$, $|p_i| = \log(n)$ and $|y'_{p_i}| = m$ $\forall i \in [k]$; **and**
- the values $\{p_1, \ldots, p_k\}$ form a size-k subset of $[n]$; **and**
- the M_{r} consists of the columns in M specified by r; **and**
- the $\mathsf{s} = (s_1, \ldots, s_n)$ satisfy the following requirement (recall that the $\{b_1, \ldots, b_t\}$ are from r):
  - if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} \neq \emptyset$, then $s_i = y_i$ for all $i \in [n]$.
  - if $\{p_1, \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$, then $s_i = \begin{cases} y'_i & i \in \{p_1, \ldots, p_k\} \\ y_i & i \in [n] \setminus \{p_1, \ldots, p_k\} \end{cases}$.

We remark that $f^{-1}(s_i)$ may not be efficiently computable (indeed, f is a one-way function). But the above proof only relies on the *existence* of $f^{-1}(s_i)$. Also, we have $\{p_1 \ldots, p_k\} \cap \{b_1, \ldots, b_t\} = \emptyset$. It then follows from the description in Constr. 1 (in particular, Case 2) that $F^f(\mathsf{x} \| \mathsf{r}) = Y$.

5.3 Proof-Based Pseudo-Random Generators

We can also define proof-based pseudo-random generators (PB-PRGs) in a similar way as for PB-OWFs. It consists of a two-input function $G^g(\cdot, \cdot)$ and a protocol $\Pi^g_G = (S^g, R^g)$ such that for any PRG g, $G^g(\cdot, \mathsf{r})$ is a PRG for any choice of r, and Π^g_G satisfies the same completeness, soundness and ZK requirements as in Def. 4 but w.r.t. G^g.

Our PB-PRG can be constructed by simply replacing the oracle OWF f with a PRG g in both Constr. 1 and Prot. 1 (our PB-OWF construction). There is one caveat: the output Y of Constr. 1 contains the preimage x_{b_i} for y_{b_i} (or s_{b_i}). While this is fine for one-wayness, such a Y will not be pseudo-random, because an adversary can always learn if Y is in the range of $G^g(\cdot, \mathsf{r})$ by testing whether $y_{b_i} = g(x_{b_i})$. To fix this, in the output Y, we will place x_{b_i} in the position where we originally put y_{b_i} (and we can drop the $(x_{b_1}, \ldots, x_{b_t})$ part from Y). We will show that this modification lead to a valid PB-PRG.

We present the definition, construction, and the security proof for PB-PRGs in the full version [35].

6 Proof-Based Collision-Resistant Hash Families

We now discuss proof-based collision-resistant hash families (PB-CRHFs). As mentioned in Sec. 2.3, the definition and construction of PB-CRHF follow the same template as our PB-OWFs, except that we need to handle the **Editing**

[9] Note that the input to h_i should have length $2m$. But $|\Lambda| > 2m$. This can be handled using domain-extension techniques, e.g., the Merkle-Damgård transformation [37, 5].

Construction 2: Collision-Resistant Hash Family $H_z^{h_i}$

Let $m(\lambda)$ and $n(\lambda)$ be polynomials of λ. Assume w.l.o.g. that is n a power of 2 (i.e., $n = 2^\ell$ for some ℓ). Let $0 < \delta < 1$ be a constant, and $k(\lambda) = \delta n(\lambda)$. Let $t(\lambda) = \log^2(\lambda)$ (see also Rmk. 1). Let $\mathcal{H}' = \{h_i\}_{i\in I}$ be a collision-resistant hash family where $h_i : \{0,1\}^{2m(\lambda)} \to \{0,1\}^{m(\lambda)}$. Denote its key generation as KGen'.

- **Key Generation.** On input 1^λ, sample a function from $\mathcal{H}'$ by running $h_i \leftarrow \mathsf{KGen}'(1^\lambda)$; sample a random string $z \leftarrow \{0,1\}^{m(\lambda)}$; outputs (i, z) as the hash key.
- **Evaluation.** On input $\mathsf{x} \in \{0,1\}^{nm+k(\log(2n)+m)+3m}$ and $\mathsf{r} \in \{0,1\}^{t\log(n)}$, the evaluation algorithm parses them as:

$$\mathsf{x} = (x_1, \ldots, x_n) \| (p_1, v_{p_1}), \ldots, (p_k, v_{p_k}) \| \tau \| \mu, \quad \mathsf{r} = (b_1, \ldots, b_t), \tag{13}$$

where $|x_i| = |v_{p_i}| = m$, $\{p_i\}_{i\in[k]}$ is a size-k subset of $[2n-1]$, and $\{b_i\}_{i\in[t]}$ is a size-t subset of $[n]$. The set $\{b_i\}_{i\in[t]}$ specifies t leaves out of all the n leaves.

The algorithm builds a perfect binary tree T that has n leaves, where all the nodes are dummies. Note that the indices of the nodes in T are well-defined, even though T now contains only dummy nodes. The evaluation procedure outputs $Y = \mathsf{t}_1 \| \mathsf{t}_2 \| (\mathsf{P}_{b_1}, \ldots, \mathsf{P}_{b_t}) \| (b_1, \ldots, b_t)$, which is computed as follows:

1. **Non-Editing:** If $\tau = z$ **or** $h_i(\tau) \neq h_i(z)$ **or** $\mathsf{Ind}(b_1, \ldots, b_t) \cap \{p_1, \ldots, p_k\} \neq \emptyset$:
   (a) It fills the tree T as follows. It places $(x_1, \ldots, x_n)$ at the n leaves. For any other node in T, its content is the hash value under h_i on the concatenation of its left child and right child. Denote the root value as t_1.
   (b) For $i \in [t]$, P_{b_i} is the sibling path of leaf x_{b_i} in the above tree T;
   (c) Use h_i to hash[9] the following Λ value to a length-m string denoted as t_2:
   $$\Lambda = (p_1, v_{p_1}), \ldots, (p_k, v_{p_k}) \| \tau \| \mu.$$
2. **Editing:** if $\tau \neq z$ **and** $h_i(\tau) = h_i(z)$ **and** $\mathsf{Ind}(b_1, \ldots, b_t) \cap \{p_1, \ldots, p_k\} = \emptyset$:
   (a) It fills the tree T as follows. It places $(x_1, \ldots, x_n)$ at the n leaf positions in T. Then, fill the tree bottom up, following the rule for Merkle tree (i.e. the hashing of two children nodes' contents is the parent node's content), with the following exception: for node $p_i \in \{p_1, \ldots, p_k\}$, it fills node p_i with the v_{p_i} contained in x, instead of the hash of the children of node p_i. Denote the root value as t_1.
   (b) For $i \in [t]$, P_{b_i} is defined as the sibling path of leaf x_{b_i} in the tree T;
   (c) Set $\mathsf{t}_2 = \mu$ (recall that μ is contained in x);

condition differently. Due to space constraints, we only show an overview of our PB-CRHF here. See the full version [35] for the details.

On the Definition. Our PB-CRHF consists of an oracle machine $H^{(\cdot)}$ and an oracle protocol $\Pi_H^{(\cdot)}$. As mentioned in Sec. 2.3, the $H^{(\cdot)}$ will be instantiated as a hash *family*. That is, given a collision-resistant hash family $\mathcal{H}'$, we first run its KGen' to sample a function $h_i \in \mathcal{H}'$, and then instantiate $H^{(\cdot)}$'s oracle as h_i. Therefore, H^{h_i} is also a hash family whose KGen simple runs the KGen' for $\mathcal{H}'$ (and samples a random string z that we will discuss later).

Once $H^{(\cdot)}$ and $\Pi_H^{(\cdot)}$ are instantiated with an $h_i \leftarrow \mathsf{KGen}'(1^\lambda)$, we can start talking about the security. Same as in Def. 4, H^{h_i} takes two inputs x and r. We require that, for all r, $H^{h_i}(\cdot, \mathsf{r})$ is collision-resistant on its first input. The

Protocol 2: Protocol $\Pi_z^{h_i}$ for Our PB-CHRF

Let $\mathcal{H}'$, m, n, δ, t and k be as in Constr. 2. Let $\Pi_{\mathsf{ZKCnP}} = (\mathsf{BBCom}, \mathsf{BBProve})$ be a black-box commit-and-prove protocol. For a function defined by (i, z) from the PB-CRHF in Constr. 2, this protocol proceeds as follows. Both parties take the security parameter 1^λ as the common input. Sender S takes a string $\mathsf{x} \in \{0,1\}^{nm+k(\log(n)+m)+3m}$ as private input; receiver R takes a string $\mathsf{r} \in \{0,1\}^{t\log(n)}$ as private input.

1. S parses x as $(x_1, \ldots, x_n) \| (p_1, v_{p_1}), \ldots, (p_k, v_{p_k}) \| \tau \| \mu$ (in the same manner as in Expression (13)). S build a Merkle tree $MT'_{h,m}(x)$ using $(x_1, \ldots, x_n)$ as the leaves (this is identical to Step 1a). Denote the root of this tree as t_x.
2. S and R execute $\mathsf{BBCom}(\nu)$, the **Commit** stage of Π_{ZKCnP}, where S commits to the following value
$$\nu \coloneqq \mathsf{t}_\mathsf{x} \| (p_1, \ldots, p_k). \tag{14}$$
3. R sends the value r.
4. S parses r as $(b_1, \ldots, b_t)$ where each b_i is of length $\log(n)$. With the values x and r, S evaluates the function $H_z^{h_i}$ as per Constr. 2 to compute the following Y, which it sends to R:
$$Y = \mathsf{t}_1 \| \mathsf{t}_2 \| (\mathsf{P}_{b_1}, \ldots, \mathsf{P}_{b_t}) \| (b_1, \ldots, b_t).$$
5. R checks if P_{b_i} is Merkle-consistent for all $i \in [t]$. R aborts if any of the check fails.
6. S and R execute $\mathsf{BBProve}$, the **Prove** stage of Π_{ZKCnP}, where S proves that the ν committed in Stage 2 satisfies the following conditions:
   (a) the $\{p_1, \ldots, p_k\}$ in ν form a size-k subset of $[2n-1]$, where $k = \delta n$; **and**
   (b) the t_x contained in τ is equal to t_1, **or** $\mathsf{Ind}(b_1, \ldots, b_t) \cap \{p_1, \ldots, p_k\} = \emptyset$.
7. **(Receiver's Output).** R outputs $Y = \mathsf{t}_1 \| \mathsf{t}_2 \| (\mathsf{P}_{b_1}, \ldots, \mathsf{P}_{b_t}) \| (b_1, \ldots, b_t)$.
8. **(Sender's Output).** S outputs
$$X = (x_1, \ldots, x_n) \| (p_1, v_{p_1}), \ldots, (p_k, v_{p_k}) \| \tau \| \mu \| (b_1, \ldots, b_t).$$

protocol $\Pi_H^{h_i}$ satisfies the similar completeness, soundness and ZK requirement as in Def. 4. We provide a formal definition in the full version [35].

Our Construction. The formal construction is provided in Constr. 2 and Prot. 2. We follow the high-level idea described in Sec. 2.3 with the following modifications. Instead of hashing the $(x_1, \ldots, x_n)$ (contained in x) separately, we build a Merkle tree using them as the leaves. In Constr. 2, P_i denotes the sibling path from leaf x_i to the root; $\mathsf{Ind}(b_1, \ldots, b_t)$ denotes the set of *indices* of the nodes on path $\mathsf{P}_{b_1}, \ldots, \mathsf{P}_{b_t}$ (see the full version [35] for more details). In Prot. 2, the receiver checks t leaves and their corresponding sibling paths. This ensures that there are at least $(n-k)$ "good" leaves, in the sense that there are valid sibling paths from the Merkle root to them. In the **Editing** case, this will allow us to perform preimage editing by planting the v_{p_i} values on the k "bad" paths to obtain a (partial) tree consistent with the root t_1 contained in Y. Note that we also hash the Λ in Step 1c. As explained in Sec. 2.3, this is to prevent the adversary from taking advantage of preimage editing to find collisions.

It is also worth noting that Constr. 2 and Prot. 2 work for an x of fixed length. But since we hash the $\{x_i\}_{i\in[n]}$ part using a Merkle tree, we can handle x with

a various-length $\{x_i\}_{i\in[n]}$ part (which dominates the length of x). To maintain security, we simply include in Y the height of the Merkle tree.

Proof of Security. The security can be proved in a similar manner as for our PB-OWFs. We provide the formal security proof in the full version [35].

References

1. Barak, B., Mahmoody-Ghidary, M.: Merkle's key agreement protocol is optimal: An $O(n^2)$ attack on any key agreement from random oracles. Journal of Cryptology **30**(3), 699–734 (Jul 2017). https://doi.org/10.1007/s00145-016-9233-9
2. Chatterjee, R., Liang, X., Pandey, O.: Improved black-box constructions of composable secure computation. In: Czumaj, A., Dawar, A., Merelli, E. (eds.) ICALP 2020. LIPIcs, vol. 168, pp. 28:1–28:20. Schloss Dagstuhl (Jul 2020). https://doi.org/10.4230/LIPIcs.ICALP.2020.28
3. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the third annual ACM symposium on Theory of computing. pp. 151–158 (1971)
4. Crépeau, C., Kilian, J.: Achieving oblivious transfer using weakened security assumptions (extended abstract). In: 29th FOCS. pp. 42–52. IEEE Computer Society Press (Oct 1988). https://doi.org/10.1109/SFCS.1988.21920
5. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34805-0_39
6. Damgård, I., Ishai, Y.: Constant-round multiparty computation using a black-box pseudorandom generator. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 378–394. Springer, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_23
7. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: 22nd ACM STOC. pp. 416–426. ACM Press (May 1990). https://doi.org/10.1145/100216.100272
8. Garg, S., Gupta, D., Miao, P., Pandey, O.: Secure multiparty RAM computation in constant rounds. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part I. LNCS, vol. 9985, pp. 491–520. Springer, Heidelberg (Oct / Nov 2016). https://doi.org/10.1007/978-3-662-53641-4_19
9. Garg, S., Kiyoshima, S., Pandey, O.: A new approach to black-box concurrent secure computation. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 566–599. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78375-8_19
10. Garg, S., Liang, X., Pandey, O., Visconti, I.: Black-box constructions of bounded-concurrent secure computation. In: Galdi, C., Kolesnikov, V. (eds.) SCN 20. LNCS, vol. 12238, pp. 87–107. Springer, Heidelberg (Sep 2020). https://doi.org/10.1007/978-3-030-57990-6_5
11. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (Feb 2004). https://doi.org/10.1007/978-3-540-24638-1_15
12. Goldreich, O.: Foundations of Cryptography: Basic Tools, vol. 1. Cambridge University Press, Cambridge, UK (2001)
13. Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, Cambridge, UK (2004)

14. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: 21st ACM STOC. pp. 25–32. ACM Press (May 1989). https://doi.org/10.1145/73007.73010
15. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In: 27th FOCS. pp. 174–187. IEEE Computer Society Press (Oct 1986). https://doi.org/10.1109/SFCS.1986.47
16. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th ACM STOC. pp. 291–304. ACM Press (May 1985). https://doi.org/10.1145/22145.22178
17. Goyal, V.: Constant round non-malleable protocols using one way functions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 695–704. ACM Press (Jun 2011). https://doi.org/10.1145/1993636.1993729
18. Goyal, V., Lee, C.K., Ostrovsky, R., Visconti, I.: Constructing non-malleable commitments: A black-box approach. In: 53rd FOCS. pp. 51–60. IEEE Computer Society Press (Oct 2012). https://doi.org/10.1109/FOCS.2012.47
19. Goyal, V., Ostrovsky, R., Scafuro, A., Visconti, I.: Black-box non-black-box zero knowledge. In: Shmoys, D.B. (ed.) 46th ACM STOC. pp. 515–524. ACM Press (May / Jun 2014). https://doi.org/10.1145/2591796.2591879
20. Haitner, I.: Semi-honest to malicious oblivious transfer - the black-box way. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 412–426. Springer, Heidelberg (Mar 2008). https://doi.org/10.1007/978-3-540-78524-8_23
21. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing **28**(4), 1364–1396 (1999)
22. Hazay, C., Venkitasubramaniam, M.: On the power of secure two-party computation. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 397–429. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53008-5_14
23. Hazay, C., Venkitasubramaniam, M.: Round-optimal fully black-box zero-knowledge arguments from one-way permutations. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 263–285. Springer, Heidelberg (Nov 2018). https://doi.org/10.1007/978-3-030-03807-6_10
24. Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo-random generation from one-way functions (extended abstracts). In: 21st ACM STOC. pp. 12–24. ACM Press (May 1989). https://doi.org/10.1145/73007.73009
25. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: 21st ACM STOC. pp. 44–61. ACM Press (May 1989). https://doi.org/10.1145/73007.73012
26. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: Kleinberg, J.M. (ed.) 38th ACM STOC. pp. 99–108. ACM Press (May 2006). https://doi.org/10.1145/1132516.1132531
27. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) 39th ACM STOC. pp. 21–30. ACM Press (Jun 2007). https://doi.org/10.1145/1250790.1250794
28. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (Aug 2008). https://doi.org/10.1007/978-3-540-85174-5_32
29. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of computer computations, pp. 85–103. Springer (1972)

30. Khurana, D., Ostrovsky, R., Srinivasan, A.: Round optimal black-box "commit-and-prove". In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 286–313. Springer, Heidelberg (Nov 2018). https://doi.org/10.1007/978-3-030-03807-6_11
31. Kilian, J.: Founding cryptography on oblivious transfer. In: 20th ACM STOC. pp. 20–31. ACM Press (May 1988). https://doi.org/10.1145/62212.62215
32. Kiyoshima, S.: Round-efficient black-box construction of composable multi-party computation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 351–368. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44381-1_20
33. Kiyoshima, S.: Round-optimal black-box commit-and-prove with succinct communication. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 533–561. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1_19
34. Levin, L.A.: Universal sequential search problems. Problemy peredachi informatsii **9**(3), 115–116 (1973)
35. Liang, X., Pandey, O.: Towards a unified approach to black-box constructions of zero-knowledge proofs. Cryptology ePrint Archive, Report 2021/836 (2021), https://eprint.iacr.org/2021/836
36. Lin, H., Pass, R.: Black-box constructions of composable protocols without set-up. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 461–478. Springer, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_27
37. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34805-0_21
38. Naor, M.: Bit commitment using pseudo-randomness. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 128–136. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34805-0_13
39. Pass, R., Wee, H.: Black-box constructions of two-party protocols from one-way functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 403–418. Springer, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00457-5_24
40. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (Feb 2004). https://doi.org/10.1007/978-3-540-24638-1_1
41. Rosulek, M.: Must you know the code of f to securely compute f? In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 87–104. Springer, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_7
42. Wee, H.: Black-box, round-efficient secure computation via non-malleability amplification. In: 51st FOCS. pp. 531–540. IEEE Computer Society Press (Oct 2010). https://doi.org/10.1109/FOCS.2010.87
43. Wyner, A.D.: The wire-tap channel. Bell system technical journal **54**(8), 1355–1387 (1975)
44. Yerukhimovich, A.B.: A study of separations in cryptography: new results and new models. Ph.D. thesis, University of Maryland, College Park, Maryland, USA (2011)