Generic Instantiations of the Hidden-Bits Model for Non-interactive Zero-knowledge Proofs for NP

Hamza Abusalah\(^1\) and Eike Kiltz\(^2\)

\(^1\) RWTH-Aachen, Aachen, Germany
hamzaabusalah@gmail.com
\(^2\) Ruhr University Bochum, Bochum, Germany
eike.kiltz@rub.de

Abstract. In this work, we study the necessary and sufficient conditions for implementing the hidden bits model (HBM) for non-interactive zero-knowledge (NIZK) proofs for NP. As a result, we first define collections of \(1\)-\(1\) doubly-enhanced verifiable trapdoor functions \(1\)-\(1\) devTDF for which the \(1\)-\(1\) property is publicly verifiable only for correct indices. Second, we use collections of \(1\)-\(1\) devTDF to construct NIZK proofs for NP by implementing the HBM. Our construction advances the state-of-the-art construction in two ways; first, the implementation of the HBM is perfect in the sense that it imposes no extra soundness error, and second, the use of functions rather than permutations makes the construction more general and opens up new possibilities for instantiating NIZK proofs based on different assumptions. We conclude by instantiating collections of \(1\)-\(1\) devTDF from families of Diffie-Hellman gap groups with explainable domains.

1 Introduction

Goldwasser, Micali, and Rackoff \cite{GMR89} introduced the intriguing notion of interactive zero-knowledge proof systems. Blum, Feldman, and Micali \cite{BFM88} showed that interaction is not essential to the notion of zero-knowledge, and as a result, suggested the notion of non-interactive zero-knowledge (NIZK) proof systems in the common random string model, which will henceforth be referred to as the standard model.

Feige, Lapidot, and Shamir \cite{FLS99} gave the first NIZK proof for an NP-complete problem based on general assumptions. Specifically, they developed a NIZK proof for Hamiltonicity in an unrealistic model they introduced, namely the hidden-bits model (HBM), and then used collections of (strongly certified) trapdoor permutations (TDP) to efficiently transform the proof into the standard model of \cite{BFM88}.

A collection of TDP is said to be (strongly) certified if given a function \(f\), it is possible to efficiently verify whether \(f\) belongs to the collection. This strong sense of certification is unlikely to be efficiently doable for known collections of trapdoor functions such as RSA \cite{RSA78} and Rabin \cite{Rab79}, to mention a few.

Therefore, Bellare and Yung \cite{BY92} showed how to certify, in NIZK, that a map is almost a permutation, i.e., \(1\)-\(1\) on at least all but a polynomial fraction of its designated range, and showed that this weak sense of certification suffices for moderate soundness condition of the NIZK proof of \cite{FLS99}. However, the procedure of \cite{BY92} does not extend to the case of certifying that a function is almost \(1\)-\(1\) and is limited to permutations.

Goldreich \cite{Gol11} pointed out that collections of TDP, as certified by \cite{BY92} do not suffice for constructing NIZK proofs, and therefore, suggested the notion of doubly-enhanced trapdoor permutations (deTDP), which when certified by an extension to the procedure of \cite{BY92}, suffice for constructing NIZK proofs for NP, however with moderate soundness which can be amplified by
repetition. However, this generic construction is limited due to the limited number of candidate permutation collections. Nevertheless, such collections exist. Rabin [Rab79], after being transformed into a permutation by restricting its domain to the quadratic residues, and the RSA collection [RSA78] are examples of deTDP (cf. [Gol11]).

1.1 This Work
The state-of-the-art of NIZK proofs [Gol11] is based on collections of deTDP as certified by [BY92]. Furthermore, all known deTDPs are based on the intractability of factoring, and hence, so are NIZK proofs.

We observe that the doubly-enhanced property is essential for the zero-knowledge, and the 1-1 property of deTDP is required to guarantee that hard-core bits are uniquely opened, which in turn, is essential to the soundness. However, requiring the trapdoor collection to be a permutation, rather than a 1-1 function, is superfluous. Additionally, requiring that the trapdoor collection is certified, as we show in the sequel, is unnecessary.

Therefore, we generalize *doubly-enhanced trapdoor permutations* (deTDP) to *functions*, for which the 1-1 property is publicly verifiable for correct indices only. As a result, we define collections of *1-1 doubly-enhanced verifiable trapdoor functions* (1-1 devTDF). By doing so, we relaxed the superfluous requirement of the trapdoor collection to be a permutation, as well as the certification requirement at once.

Then we use collections of 1-1 devTDF to implement the HBM for NIZK proofs. The novelty of our construction is the use of 1-1 devTDF whose advantage is threefold. First, the implementation of the HBM is perfect in the sense that it imposes no extra soundness error, i.e., the soundness error is the same for both the HBM and the standard model NIZK proof. Second, certification is no longer required. Third, and more importantly, the use of functions, rather than permutations, makes the construction more general and opens up new possibilities for instantiating NIZK proofs based on different assumptions.

Consequently, we instantiate collections of 1-1 devTDF based on families of *Diffie-Hellman gap groups* with *explainable* domains. *Diffie-Hellman gap groups* are groups in which the Decisional Diffie-Hellman Problem is easy while the Computational Diffie-Hellman Problem is hard. The *explainable domains* property implies that given a domain sample, it is possible to efficiently and publicly generate a corresponding uniform sampling randomness.

To stress the generality of our construction, we note that a relaxed version of 1-1 devTDF (cf. Sec 1.2) can be instantiated with deTDP as certified by [BY92]. Therefore our construction extends the class of possible instantiations. In fact, the extension is strict in the sense that our construction of NIZK proofs is not based on factoring.

1.2 Extensions
We relax collections of 1-1 devTDF to *ε*-weak 1-1 devTDF where the main characteristics of 1-1 devTDF are allowed to fail on at most an ε-fraction of the designated range. Then we show how to reach the state-of-the-art results of NIZK proofs by constructing NIZK proofs based on *ε*-weak 1-1 devTDF, and instantiating them based on deTDP as certified by [BY92].

Whereas the use of collections of 1-1 devTDF yields an efficient implementation of the HBM without introducing any extra error, the use of *ε*-weak 1-1 devTDF results in an additional error in both completeness and soundness of the NIZK proof system. However, such additional errors are unavoidable and can be reduced by repetition.
1.3 Organization

In Sec. 2, we introduce some background and notations, including NIZK proofs and the HBM. In Sec. 3, we define collections of 1-1 devTDF and use them to implement the HBM. In Sec. 4, we define and instantiate 1-1 devTDF. In Appendix A, we define collections of (non-certified) deTDP, and finally use them to implement the HBM.

2 Background

In this section, we set up some necessary notational conventions, generalize enhanced and doubly-enhanced trapdoor permutations to functions, and recall the basic definitions of NIZK proofs and the HBM.

2.1 Notational Conventions

Let $X$ be a finite set, then $x \leftarrow X$ denotes the algorithm that samples a uniform $x$ from $X$. Let $M$ be a probabilistic polynomial-time (PPT) algorithm, then $M(x)$ denotes its probability space when run on $x$, and $[M(x)]$ denotes the support of $M(x)$, i.e., the set of outcomes with positive probability, and $R_M$ its randomness space $R_M$ which we require to be of the form $R_M = \{0,1\}^l$ for some polynomial $l(\cdot)$. Furthermore, for a fixed randomness $r$, $M(x;r) = y$ denotes the run of the dual deterministic algorithm of $M$ on input $x$ and randomness $r$ with output $y$. That is, if $M(\cdot)$ is a PPT algorithm, then $M(\cdot;\cdot)$ is its deterministic counterpart.

The notion $Pr[y \leftarrow M(x)]$ denotes the probability with which $M$ samples a uniform element $y$ from $M(x)$, and $Pr[p(x_1,x_2,\ldots) : x_1 \leftarrow X_1; x_2 \leftarrow X_2; \ldots]$ denotes the probability that the predicate $p(x_1,x_2,\ldots)$ is true after the ordered execution of $x_1 \leftarrow X_1$, $x_2 \leftarrow X_2$, etc, where $X_i$ is a finite set.

A function $\nu : \mathbb{N} \rightarrow \mathbb{R}$ is called negligible (in $n$), if for every positive polynomial $p(\cdot)$ and all sufficiently large $n \in \mathbb{N}$, it holds: $\nu(n) \leq \frac{1}{p(n)}$. We use $\text{neg}(n)$ to denote an unspecified negligible function. A strong negation of the notion of negligible function is the notion of noticeable function.

2.2 Enhanced and Doubly-Enhanced Trapdoor Functions

In this section, we generalize collection of enhanced and doubly-enhanced trapdoor permutations of [Gol11] to functions in a natural manner.

Definition 1 (Collections of Trapdoor Functions). Let $I \subseteq \{0,1\}^*$ be a set of indices, and $I_n = I \cap \{0,1\}^{p(n)}$, where $p(\cdot)$ is some polynomial. Then the set $\mathcal{F} = \{f_\alpha : D_\alpha \rightarrow R_\alpha\}_{\alpha \in I}$, where $D_\alpha$ and $R_\alpha$ are finite sets, is a collection of trapdoor functions (TDF) with indices in $I$, if there exists a quadruple of PPT algorithms $(G,S,F,F^{-1})$ s.t.,

1. Index and Trapdoor Generator $G$: For every $n$, $G(1^n)$ outputs $(\alpha, \tau) \in I_n \times \{0,1\}^*$, where $\alpha$ is the index and $\tau$ is the trapdoor.
2. Domain Sampler $S$: For every $\alpha \in I_n$, $S(\alpha)$ outputs a uniformly distributed $x \in D_\alpha$.
3. Evaluator $F$: For every $\alpha \in I_n$, and $x \in D_\alpha$, $F(\alpha, x)$ outputs $f_\alpha(x)$.
4. Inverter $F^{-1}$: For every pair $(\alpha, \tau) \in [G(1^n)]$, and every $x \in D_\alpha$, $F^{-1}(\alpha, \tau, f_\alpha(x))$ outputs a preimage $z$ such that $f_\alpha(z) = f_\alpha(x)$.
The security requirements is that for every PPT algorithm \( \mathcal{A} \),
\[
\Pr[f_\alpha(z) = y : \alpha \leftarrow I_n; x \leftarrow D_\alpha; y = f_\alpha(x); z \leftarrow \mathcal{A}(\alpha, y)] \leq \text{neg}(n) .
\]
Moreover, \( \mathcal{G} = (G, S, F, F^{-1}) \) is the generator of the collection.

A trapdoor function is said to be enhanced, if the advantage of any PPT adversary in inverting the trapdoor function on a uniformly chosen image is still negligible even when given additionally the sampling randomness of such an image.

**Definition 2 (Collections of Enhanced Trapdoor Functions).** Let \( \mathcal{F} = \{f_\alpha : D_\alpha \rightarrow R_\alpha\}_{\alpha \in I} \) be a collection of TDF with generator \( \mathcal{G} = (G, S, F, F^{-1}) \). Then \( \mathcal{F} \) is called enhanced (eTDF), if there exists a PPT algorithm \( S_R \) that, on input index \( \alpha \) outputs a uniformly distributed \( y \in R_\alpha \), and for every PPT algorithm \( \mathcal{A} \),
\[
\Pr[f_\alpha(z) = y : \alpha \leftarrow I_n; r \leftarrow R_{S_R}; y = S_R(\alpha; r); z \leftarrow \mathcal{A}(\alpha, r)] \leq \text{neg}(n) .
\]
where \( R_{S_R} \) is the randomness space of \( S_R \). Moreover, \( \mathcal{G} = (G, S_D, S_R, F, F^{-1}) \) is the generator of the collection, where we denoted \( S \) by \( S_D \) for clarity.

We remark that that the adversary \( \mathcal{A} \) can compute the image \( y \) given the randomness \( r \), and hence it is implicit in \( \mathcal{A} \)'s input.

An enhanced trapdoor function is said to be doubly-enhanced, if there exists a PPT algorithm that publicly generates samples of the form \( (x, r) \), such that \( x \) is a preimage of the image whose sampling randomness is \( r \).

**Definition 3 (Collections of Doubly-Enhanced Trapdoor Functions).** A collection of eTDF \( \{f_\alpha : D_\alpha \rightarrow R_\alpha\}_{\alpha \in I} \) with generator \( (G, S_D, S_R, F, F^{-1}) \), is called doubly-enhanced (deTDF), if there exists a PPT algorithm \( S'_R \) that on input \( \alpha \) outputs a pair \( (x, r) \) s.t., \( r \) is identically distributed to \( R_{S_R} \), where \( R_{S_R} \) denotes the randomness space of \( S_R \), and \( x = f_\alpha^{-1}(S_R(\alpha; r)) \). Moreover, \( \mathcal{G} = (G, S_D, S_R, S'_R, F, F^{-1}) \) is the generator of the collection.

Thus the doubly-enhanced property makes it possible to efficiently generate samples of the form \( (x, r) \) publicly, i.e., without relying on the trapdoor, such that \( x = f_\alpha^{-1}(S_R(\alpha; r)) \). This property will play an essential role in proving zero-knowledge.

### 2.3 Hardcore Predicates

A polynomial-time computable boolean predicate \( b \) is said to be hardcore of a function \( f \), if it is infeasible to compute \( b(x) \), given \( f(x) \) only, with a probability that is noticeably better than random guessing.

**Definition 4 (Hardcore Predicates).** A polynomial-time computable predicate \( b : \{0,1\}^n \rightarrow \{0,1\} \) is called hardcore of a function \( f \) if for every PPT algorithm \( \mathcal{A} \),
\[
\Pr[z = b(x) : x \leftarrow \{0,1\}^n; y = f(x); z \leftarrow \mathcal{A}(y)] < \frac{1}{2} + \text{neg}(n) .
\]

Analogous to eTDF, a hardcore predicate is called enhanced if it is hard to approximate even when the adversary is given the sampling random coins.
2.4 Non-Interactive Zero-Knowledge and the Hidden-Bits Model

In this section, we recall the definitions of NIZK proofs and the HBM as given by [Gol11].

**NIZK Proofs.** A NIZK proof system, as introduced by [BFM88], consists of three entities, namely prover, verifier, and simulator operating on a common input, such that the system meets completeness, soundness, and zero-knowledge properties. The common input consists of a statement to be proven \( x \), as well as a common random string (CRS). The statement \( x \) belongs to some predefined language \( L \). The CRS is a uniformly distributed string whose length is polynomial in the size of the statement to be proven \( x \in L \). The CRS can be thought of as being produced by a trusted third party.

A NIZK proof consists of a single message \( \pi \) sent from the prover to the verifier, to which the latter responds by either accepting or rejecting. Whereas completeness requires the prover to be able to convince the verifier of the validity of true statements \( x \in L \), by producing a proof \( \pi \), soundness requires the verifier to protect itself from accepting false proofs; proofs of false statements \( x \notin L \).

With knowledge being viewed as a result of hard computation, a proof is zero-knowledge if it yields no knowledge; that is, whatever can be efficiently obtained from the proof \( \pi \), can be efficiently computed from the assertion itself \( x \in L \) without the proof \( \pi \). When thinking about zero-knowledge, it is best to think of the verifier as an adversary trying to gain knowledge from the proof, and the prover as an entity trying to protect itself from leaking knowledge. In light of this, zero-knowledge is formalized by requiring the existence of an efficient simulator \( S \), such that its output distribution is computationally indistinguishable from the prover’s output distribution.

In the literature, one finds a variety of definitions of NIZK proofs. We stick to the notion of non-adaptive bounded NIZK proof systems with computational zero-knowledge as captures by Def. 5. Furthermore, such proofs are referred to as proofs in the standard model, as opposed to NIZK proofs in the HBM.

**Definition 5 (Non-Interactive Zero-Knowledge Proof Systems).** A triplet of algorithms \((P, V, S)\) is called non-interactive zero-knowledge (NIZK) proof system for a language \( L \), if \( V \) and \( S \) are PPT, and there exists a polynomial \( p(\cdot) \) such that the following three conditions hold:

- **Completeness:** For every \( x \in L \),
  
  \[
  \Pr[V(x, \sigma, \pi) = 1 : \sigma \leftarrow \{0, 1\}^{p(|x|)}; \pi \leftarrow P(x, \sigma)] \geq \frac{2}{3} .
  \]

- **Soundness:** For every \( x \notin L \) and every algorithm \( A \),
  
  \[
  \Pr[V(x, \sigma, \pi) = 1 : \sigma \leftarrow \{0, 1\}^{p(|x|)}; \pi \leftarrow A(x, \sigma)] \leq \frac{1}{3} .
  \]

- **Zero-Knowledge:** The following two ensembles are computationally indistinguishable:
  1. \( \{S(x)\}_{x \in L} \)
  2. \( \{(x, \sigma, \pi) : \sigma \leftarrow \{0, 1\}^{p(|x|)}; \pi \leftarrow P(x, \sigma)\}_{x \in L} \)

The algorithms \( P, V, \) and \( S \) are called prover, verifier, and simulator, respectively. The string \( \sigma \) is called the common random string (CRS).
A couple of remarks is in order. First, whereas soundness holds for all potential provers \( \mathcal{A} \), completeness refers to a specific prover \( P \); the prescribed one. Second, whereas algorithms \( V \) and \( S \) are required to be polynomial-time, the prover, in both completeness and soundness conditions, is computationally unbounded. Third, the error probabilities in both completeness and soundness conditions can be made negligibly small by repeating the process sufficiently many times, using a sequence of independently chosen CRS’s. Therefore, the choice of the error in both conditions to be \( \frac{1}{3} \) is immaterial.

A NIZK proof system is said to be efficient, or equivalently has an efficient prover, if the prover in the completeness condition is PPT. As we study NIZK proofs for NP, the prover is required to be PPT and is fed with an auxiliary input, i.e., a witness for \( x \in L \).

The HBM. The HBM is a fictitious model that is useful for designing and analyzing NIZK proofs. The only difference between the HBM and the standard model is that some of the bits of the CRS are hidden to the verifier, while the prover sees all of the CRS. Specifically, the prover hides specific bits from the view of the verifier.

**Definition 6 (NIZK Proof System in the HBM).** A triplet of algorithms \( (P, V, S) \) is a NIZK proof system in the HBM for a language \( L \), if \( V \) and \( S \) are PPT, and there exists a polynomial \( p(\cdot) \), such that the following three conditions hold:

- **Completeness:** For every \( x \in L \),

\[
Pr[V(x, \sigma_R, R, \pi) = 1 : \sigma \leftarrow \{0,1\}^{p(|x|)}; (R, \pi) \leftarrow P(x, \sigma)] \geq \frac{2}{3}.
\]

where \( R = (r_1, \ldots, r_t) \subseteq \{1,2,\ldots,p(|x|)\} \), and \( \sigma_R \) is the substring of \( \sigma \) at positions \( R \), i.e.,

\[
\sigma_R = \sigma_{r_1} \cdots \sigma_{r_t}.
\]

- **Soundness:** For every \( x \notin L \) and every algorithm \( \mathcal{A} \),

\[
Pr[V(x, \sigma_R, R, \pi) = 1 : \sigma \leftarrow \{0,1\}^{p(|x|)}; (R, \pi) \leftarrow \mathcal{A}(x, \sigma)] \leq \frac{1}{3}.
\]

where \( R = (r_1, \ldots, r_t) \subseteq \{1,2,\ldots,p(|x|)\} \), and \( \sigma_R \) is the substring of \( \sigma \) at positions \( R \), i.e.,

\[
\sigma_R = \sigma_{r_1} \cdots \sigma_{r_t}.
\]

- **Zero-Knowledge:** The following two ensembles are computationally indistinguishable:

1. \( \{S(x)\}_{x \in L} \)

2. \( \{(x, \sigma_R, R, \pi) : \sigma \leftarrow \{0,1\}^{p(|x|)}; (R, \pi) \leftarrow P(x, \sigma)\}_{x \in L} \)

where \( R = (r_1, \ldots, r_t) \subseteq \{1,2,\ldots,p(|x|)\} \), and \( \sigma_R \) is the substring of \( \sigma \) at positions \( R \), i.e.,

\[
\sigma_R = \sigma_{r_1} \cdots \sigma_{r_t}.
\]

\( \pi \) is called the certificate, and \( R \) is called the set of revealed bits.

The proof in the HBM consists of a certificate \( \pi \), which serves as the core of the proof, and a set of revealed bits \( \sigma_R \). Note that the prover \( P \), who sees all of \( \sigma \), chooses to reveal a set of its bits specified by \( R \), i.e., \( \sigma_R \). Hence, whatever is not revealed by \( P \) is hidden from the verifier.

The merit of the HBM is that no computational assumptions are needed to construct proofs in it. Computational assumptions are pushed to the second level of implementing the model, i.e., transforming NIZK proofs in the HBM to the standard model. This abstraction makes it possible to concentrate on the proof itself and leave the implementation details to a separate stage.

Efficient implementation of HBM can be achieved by using trapdoor permutations, or variants thereof, such that the standard model prover is provided with the trapdoor as input.
3 Efficient Implementation of the HBM

In this section, we define collections of 1-1 devTDF and use them to efficiently and perfectly implement the HBM.

3.1 1-1 Doubly-Enhanced Verifiable Trapdoor Functions

In the sequel, inspired by verifiable random function of [MRV99], we define collections of 1-1 doubly-enhanced verifiable trapdoor functions, (1-1 devTDF), which will prove essential to the construction of NIZK proof systems.

Collections of 1-1 devTDF enhance collections of deTDF in two pivotal aspects. First, the 1-1 property is publicly verifiable for correct indices, and second, it is impossible to prove it for adversarially chosen indices. This is achieved by proving unique invertibility of images by generating appropriate certificates of unique invertibility. The definition of 1-1 devTDF differs from deTDF by requiring the existence of two additional PPT algorithms, namely certificate generator, and certificate verifier. Informally speaking, the certificate generator is fed with a trapdoor and is required to generate a certificate of unique invertibility for all images. On the other hand, the certificate verifier is required to publicly, i.e., it is given the public index but not the trapdoor, verify the validity of such certificates. Second, it is not possible to prove unique invertibility of images for an adversarially chosen index.

Take the aforementioned properties into account, it is easy to see that collections of certified deTDP imply collections of 1-1 devTDF. However, the former is not essential to NIZK proofs, while the latter is.

Definition 7 (Collections of 1-1 Doubly-Enhanced Verifiable Trapdoor Functions). Let $F = \{f_\alpha : D_\alpha \rightarrow R_\alpha\}_{\alpha \in I}$ be a collection of deTDF with generator $G = (G, S_D, S_R, S'_R, F, F^{-1})$ and $(\alpha, \tau) \in [G(1^n)]$. Then $F$ is a collection of 1-1 Doubly-Enhanced Verifiable Trapdoor Function (1-1 devTDF), if there exists a duplet of PPT algorithms $(\Pi_G, \Pi_V)$. The certificate generator $\Pi_G$, on input trapdoor $\tau$, and $y \in R_\alpha$, outputs a certificate $\pi$ claiming unique invertibility of $y$, i.e., $|f_\alpha^{-1}(y)| = 1$. The certificate verifier $\Pi_V$, on input index $\alpha$, $\pi$, $x$, and a string $y$ either accepts or rejects a certificate $\pi$ claiming that $f_\alpha(x) = y$ and $|f_\alpha^{-1}(y)| = 1$. We require the following:

- Completeness: For all $(\alpha, \tau) \in [G(1^n)]$, and $y \in R_\alpha$, if $\pi \leftarrow \Pi_G(\tau, y)$, then $1 \leftarrow \Pi_V(\alpha, x, y, \pi)$.
- Soundness: For all indices $\alpha$, and all strings $y, x_1, x_2, \pi_1, \pi_2$ such that $x_1 \neq x_2$, then $0 \leftarrow \Pi_V(\alpha, x_j, y, \pi_j)$ for at least one $j \in \{1, 2\}$.

Moreover, $G = (G, S_D, S_R, S'_R, F, F^{-1}, \Pi_G, \Pi_V)$ is the generator of the collection.

The completeness condition requires the certificate generator $\Pi_G$ to be able to generate correct certificates, i.e., certificates that pass verification, for all correct indices $\alpha \in I_\alpha$, and all images $y \in R_\alpha$ possessing unique preimages. However, it does not require that a certificate is unique, that is, $\Pi_G$ may produce more than one distinct correct certificate asserting unique invertibility of an image $y \in R_\alpha$.

The soundness conditions not only requires that the certificate verifier $\Pi_V$ rejects false certificates of unique invertibility, for all indices $\alpha$, even adversarially chosen ones, and all strings $y$, even when $y \notin R_\alpha$, but also guarantees that no two certificates for two distinct preimages of the same image are accepted. This strong soundness is crucial in applications where indices can be adversarially chosen. This is the case for NIZK proof systems.
A couple of remarks is in place. First, although we required the collection to be doubly-enhanced, this requirement is not essential to the 1-1 property being publicly verifiable, i.e., such collections can be defined for trapdoor functions that are not doubly-enhanced. Second, the collection is implicitly assumed to possess an efficient algorithm that tests image membership, i.e., given a string \( y \), tests whether \( y \in R_\alpha \).

### 3.2 Efficient Implementation of the HBM

In this section we show how to efficiently transform NIZK proofs for any language in NP in the HBM into the standard model given a collection of 1-1 devTDF with corresponding polynomial-time computable hardcore predicate.

The essence of implementing the HBM is the ability of the standard model prover to selectively hide bits of the HBM-CRS. To exemplify, given a collection of one-way permutations with corresponding hardcore predicate, and a (standard model) CRS viewed as a sequence of images to the hide bits of the HBM-CRS. To exemplify, given a collection of one-way permutations with corresponding hardcore predicate, and a (standard model) CRS viewed as a sequence of images to the hide bits of the HBM-CRS. To exemplify, given a collection of one-way permutations with corresponding hardcore predicate, and a (standard model) CRS viewed as a sequence of images to the hide bits of the HBM-CRS. To exemplify, given a collection of one-way permutations with corresponding hardcore predicate, and a (standard model) CRS viewed as a sequence of images to the hide bits of the HBM-CRS.

Although we use collections of 1-1 devTDF, Construction [1] is mutatis mutandis identical to the construction of [Gol11] based on collections of deTDP.

**Construction 1** Let \((P, V, S)\) be a NIZK proof system in the HBM for a language \(L \in \text{NP}\). Given a collection of 1-1 devTDF, \(\mathcal{F} = \{ f_\alpha : D_\alpha \rightarrow R_\alpha \} \) with generator \(G = (G, S_D, S_R, S'_R, F, F^{-1}, \Pi_G, \Pi_V)\), s.t. \(R_{S_R} = \{0,1\}^l\) is the randomness space of the range sampler \(S_R\), and a corresponding polynomial-time computable hardcore predicate \(b : D_\alpha \rightarrow \{0,1\}\), then we construct a NIZK proof system \((P', V', S')\) for the same language \(L\) in the standard model \((P', V', S')\). Let \(|x| = n\) for \(x \in L\), \((\alpha, \tau) \in G(1^n)\), and \(p(\cdot)\) be some polynomial.

We quickly introduce some conventions. Variables with hat symbol indicate simulated ones, and those with prime symbol refer to the standard model. For example, whereas \(\sigma\) denotes the HBM-CRS, \(\sigma'\) denotes the standard model CRS, and \(\hat{\sigma}\) denotes the simulated CRS in the standard model. Furthermore, \(f_\alpha(x)\), and \(f_\alpha^{-1}(y)\), are shortcuts for \(F(\alpha, x)\), and \(F^{-1}(\alpha, \tau, y)\), respectively.

- **Common Input:** \((x \in L, \alpha, \sigma' \leftarrow \{0,1\}^{p(n)-l})\).
- **CRS** \(\sigma' = \sigma'_1 \cdots \sigma'_{p(n)}\) where \(\sigma'_i \leftarrow \{0,1\}^l\) is viewed as a sequence of randomness used by the range sampler \(S_R\) to sample images from \(R_\alpha\), i.e., \(S_R(\alpha; \sigma'_i) \in R_\alpha\).
- **Prover \(P'\):** Given the trapdoor \(\tau\) as an auxiliary input, \(P'\) does the following:
  1. Compute a HBM-CRS as \(\sigma = \sigma_1 \cdots \sigma_{p(n)}\) with \(\sigma_i = b(f_\alpha^{-1}(S_R(\alpha; \sigma'_i)))\) for \(i = 1, \ldots, p(n)\).
  2. Invoke the HBM prover \(P\) to obtain a HBM proof as \((R, \pi) \leftarrow P(x, \sigma)\), where \(R = (r_1, \ldots, r_t) \subseteq \{1, 2, \ldots, p(n)\}\) is the set of revealed bits and \(\pi\) is a proof certificate.
  3. Let \(\sigma'_R\) be the segments of \(\sigma'\) that correspond to \(R\). Then compute the preimages \(p_R\) of \(\sigma'_R\), as \(p_R = p_1 \cdots p_t\) s.t., \(p_i = f_\alpha^{-1}(S_R(\alpha; \sigma'_i))\).
  4. Compute certificates of unique-invertibility \(\mu_R = \mu_1 \cdots \mu_t\) of the images of \(p_R\) s.t., \(\mu_i \leftarrow \Pi_G(\tau, f_\alpha(p_i))\).
  5. Output \(\rho = (p_R, \mu_R, R, \pi)\).
- **Verifier \(V'\):** On input \(\rho = (p_R, \mu_R, R, \pi)\), \(V'\) does the following:
1. Check the correctness of the preimages $p_R = p_1 \cdots p_t$, i.e., check whether $f_\alpha(p_i) = S_R(\alpha; \sigma_i^R)$ for all $r_i \in R$, and reject and halt in case of a mismatch.

2. Verify the certificates of unique-invertibility $\mu_R$ of the images of $p_R$, i.e., check whether $1 \leftarrow H_V(\alpha, p_i, f_\alpha(p_i), \mu_i)$, and reject and halt in case of a rejection of any certificate.

3. Recover the revealed bits for the HBM verifier as $\sigma_R = \sigma_1 \cdots \sigma_R = b(p_1) \cdots b(p_t)$.

4. Invoke the HBM verifier as $V(x, \sigma_R, R, \pi)$ and accept iff $V$ accepts.

– Simulator $S'$: Given $(x \in L, \alpha, \tau)$ and black-box access to $S$, $S'$ does the following:

1. Invoke $S$ to obtain a HBM-simulated proof as $(x, \hat{\sigma}_R, \hat{R}, \hat{\pi}) \leftarrow S(x)$, where $\hat{R} = (\hat{r}_1, \ldots, \hat{r}_t) \subseteq \{1, 2, \ldots, p(n)\}$ is the set of HBM-simulated revealed bits, and $\hat{\pi}$ is a simulated proof certificate.

2. Simulate $\hat{\sigma}'_R = \hat{\sigma}'_{\hat{r}_1} \cdots \hat{\sigma}'_{\hat{r}_t}$ where $\hat{r}_i \in \hat{R}$, s.t., for each $\hat{r}_i$, repeatedly run the doubly-enhanced sampler $S'_R$ on $\alpha$ to obtain $(\hat{p}_i, u_i) \leftarrow S'_R(\alpha)$ until $f_\alpha(\hat{p}_i) = S_R(\alpha; u_i)$ and $\hat{b}(\hat{p}_i) = \hat{\sigma}_{\hat{r}_i}$, then set $\hat{\sigma}_{\hat{r}_i} = u_i$. Let $\hat{p}_R = \hat{p}_1 \cdots \hat{p}_t$ be the simulated preimages.

3. Simulate the rest of $\hat{\sigma}'$, that is, simulate $\hat{\sigma}'_C = \hat{\sigma}'_{\hat{c}_1} \cdots \hat{\sigma}'_{\hat{c}_t}$, s.t., $\hat{c}_i \in \hat{C} = (\hat{c}_i)_{\hat{c}_i \in \{1, \ldots, p(n)\} \setminus \{\hat{r}_j \mid j \in \hat{R}\}}$ by choosing $\hat{\sigma}'_{\hat{c}_i} \leftarrow \{0, 1\}^t$.

4. Simulate certificates of unique-invertibility $\hat{\mu}_{\hat{R}} = \hat{\mu}_1 \cdots \hat{\mu}_t$ of the images of $\hat{p}_R$ as $\hat{\mu}_i \leftarrow H_C(\tau, f_\alpha(p_i))$.

5. Output $(\hat{p}_R, \hat{\mu}_{\hat{R}}, \hat{R}, \hat{\pi})$.

**Theorem 1.** Construction 2 is perfect and efficient implementation of the HBM, i.e., efficiently transforms an HBM-NIZK proof $(P, V, S)$ into a NIZK proof $(P', V', S')$ in the standard model for the same language with the same completeness and soundness errors.

The proof of Theorem 1 is given in Appendix B.1. However, it would be incomplete to leave it without showing the essentialness of the use of collections of 1-1 devTDF in our construction.

First, soundness presupposes unique openings of the hardcore bits, for otherwise a cheating prover can use selective opening of specific bits to fool the verifier into accepting false proofs. With the use of 1-1 devTDF, unique opening of hardcore bits is guaranteed due to the public verifiability of the 1-1 property, and the soundness of 1-1 devTDF, where adversarially chosen indices are rejected.

Second, the doubly-enhanced property is essential to proving zero-knowledge. Recall that the doubly-enhanced property guarantees generating samples of the form $(x, r)$ such that $x = f_\alpha^{-1}(S_R(\alpha; r))$ publicly. The doubly-enhanced property implies that no PPT adversary exists, such that given polynomially many samples of the form $(x, r)$ can invert the function. The reason is simple. Such an adversary can be converted to another PPT adversary that does not require such samples, for it can publicly generate them on its own. Put differently, the doubly-enhanced property implies that the hardcore bits are enhanced and hence reveal no knowledge (cf. Sec. 2.3). For more details, see how the doubly-enhanced sampler was used in an essential manner to prove that the simulated and the real ensembles are indistinguishable (cf. proof of Lemma 2 of Theorem 1 in Appendix B.1). However, the use of the doubly-enhanced sampler in the construction itself is not essential, as both $P'$ and $S'$ have access to the trapdoor $\tau$, and hence, can invert $f_\alpha$.

Last, providing the simulator with the trapdoor $\tau$ might be prima facie confusing. To unravel such subtlety, it is useful to remember that Theorem 1 is merely an implementation that transforms an already existing zero-knowledge proof system in the HBM into the standard model. Therefore, 

\footnote{Note that what is essential to the soundness is unique openings of hardcore bits. We achieve this by requiring the function to be 1-1, but probably that is not essential, i.e., probably there exists a way to guarantee unique hardcore openings for surjective functions.}
zero-knowledge is already achieved by having an efficient HBM simulator \( S \) in the face of a computationally unbounded HBM prover \( P \), and therefore \( S' \) merely maintains it.

4 Instantiation of 1-1 Doubly-Enhanced Verifiable Trapdoor Functions

In this section, we instantiate collections of 1-1 devTDF, from families of Gap Diffie-Hellman groups with explainable domains.

We start by setting up the preliminary notions and definitions that pave the way to our construction. We first define families of groups, then enhance it with the explainable domains property [PHKW10]. After that, Gap Diffie-Hellman groups are defined [BLS04].

**Definition 8 (Family of Groups).** Let \( I \subseteq \{0,1\}^* \) be a set of indices, and \( I_n = I \cap \{0,1\}^{p(n)} \), where \( p(\cdot) \) is some polynomial. Then the set \( \mathcal{G} = \{G_\alpha\}_{\alpha \in I} \), where \( G_\alpha \) is prime order (multiplicative) finite cyclic group, is called family of groups with indices in \( I \), if there exists a triplet of polynomial-time algorithms \((G,S,M)\) such that,

1. **Index Generator \( G \):** For every \( n \), \( G(1^n) \) outputs an index \( \alpha = (G_\alpha, g_\alpha, p_\alpha) \in I_n \) where \( G_\alpha = \langle g_\alpha \rangle \) and \( |G_\alpha| = p_\alpha \) is prime.
2. **Domain Sampler \( S \):** For every \( \alpha \in I_n \), \( S(\alpha) \) outputs a uniformly distributed element \( e \in G_\alpha \).
3. **Element Multiplier \( M \):** For every \( \alpha \in I_n \), and \( e_1, e_2 \in G_\alpha \), \( M(\alpha, e_1, e_2) \) outputs \( e_1 e_2 \in G_\alpha \).

Moreover, \( \mathcal{G} = (G,S,M) \) is the generator of the family.

The index generator \( G \) outputs an index \( \alpha \) consisting of a prime order cyclic group \( G_\alpha \) alongside with a corresponding generator \( g_\alpha \). The index \( \alpha \) is not required to be uniformly distributed over the set of indices. Given an index \( \alpha \), the domain sampler \( S \) samples a uniformly distributed element of \( G_\alpha \). The element multiplier \( M \) guarantees efficient operation over the group; that is, given two arbitrary group elements, it produces their product efficiently.

Families of groups are said to have explainable domains if for an efficiently samplable domain, there exists an efficient algorithm that, given an arbitrary sample of the domain, produces a uniformly distributed randomness over the set of all possible random coins used by the domain sampler to generate the sample.

**Definition 9 (Family of Groups with Explainable Domains).** Let \( \mathcal{G} = \{G_\alpha\}_{\alpha \in I} \) be a family of groups with generator \( \mathcal{G} = (G,S,M) \), then \( \mathcal{G} \) is said to have explainable domains iff there exists a PPT algorithm \( E \), that on input \( \alpha \in I_n \) and \( e \in G_\alpha \), outputs a uniformly distributed \( \bar{r} \in \{r \in \mathcal{R}_S : S(\alpha;\bar{r}) = e\} \), where \( \mathcal{R}_S \) denotes the randomness space of \( S \).

Informally, Gap Diffie-Hellman (GDH) groups are groups in which Decisional Diffie-Hellman Problem (DDHP) is easy while Computational Diffie-Hellman Problem (CDHP) is hard.\(^4\) Let \( G \) be a (multiplicative) cyclic group with prime order \( p \), and generator \( g \). Furthermore, let \( a, b, c \in \mathbb{Z}_p \).

On the one hand, the DDHP asks to decide whether \( c \equiv ab \mod p \) given the tuple \( \langle g, g^a, g^b, g^c \rangle \). If \( c \equiv ab \mod p \), then \( \langle g, g^a, g^b, g^c \rangle \) is called Diffie-Hellman tuple. On the other hand, for \( a, b \in \mathbb{Z}_p \), the CDHP asks to compute \( g^{ab} \) given \( \langle g, g^a, g^b \rangle \).

**Definition 10 (Family of Gap Groups).** Let \( \mathcal{G} = \{G_\alpha\}_{\alpha \in I} \) be a family of groups. Then \( \mathcal{G} \) is called a family of gap Diffie-Hellman groups (GDH) if the following hold:

\(^4\) For separation results between DDHP and CDHP, the interested reader is referred to [JN03].
– DDHP is easy: There exists a (deterministic) polynomial-time algorithm $D$, such that for every $\alpha \in I_n$, and $a, b \in \mathbb{Z}_{p_n}$, $D(\alpha, g^a_{\alpha}, g^b_{\alpha}) = 1$ iff $c \equiv ab \mod p_n$.
– CDHP is hard: For every PPT algorithm $A$,

$$\Pr[z \equiv g^a_{\alpha} \mod p_n : \alpha \leftarrow I_n; a, b \leftarrow \mathbb{Z}_{p_n}; z \leftarrow A(\alpha, g^a_{\alpha}, g^b_{\alpha})] \leq \text{neg}(n).$$

We leave the construction of GDH groups abstract. For concrete instantiations, see [BF03, JN03].

Given a collection of GDH groups, we construct collections of 1-1 devTDF. Whereas the hardness of CDHP is essential for one-wayness, the easiness of the DDHP is essential for the public verifiability of 1-1 property of 1-1 devTDF.

**Construction 2** Given a family of gap groups with explainable domains, $\hat{G} = \{G_{\hat{\alpha}}\}_{\hat{\alpha} \in I}$ with generator $\hat{G} = (\hat{G}, \hat{S}, \hat{M}, \hat{E})$, we construct a collection of 1-1 devTDF $\mathbb{F} = \{f_\alpha : D_\alpha \rightarrow R_\alpha\}_{\alpha \in I}$ where $I \subseteq \{0, 1\}^*$ is a set of indices, and $I_n = I \cap \{0, 1\}^{l(n)}$, where $l(\cdot)$ is some polynomial.

– Index and Trapdoor Generator $G$: For every $\alpha \in I_n$, $G$ generates a gap group index as $\hat{\alpha} = (G_{\hat{\alpha}}, g_{\hat{\alpha}}, p_{\hat{\alpha}}) \leftarrow \hat{G}(1^n)$, chooses a trapdoor as $\tau \leftarrow \mathbb{Z}_{p_{\hat{\alpha}}}$, computes $h := g_{\hat{\alpha}}^\tau$, sets the index $\alpha = (\hat{\alpha}, h)$, and outputs an index-trapdoor tuple $(\alpha, \tau) \in I_n \times \{0, 1\}^*$.
– Domain Sampler $S$: For every $\alpha \in I_n$, $S(\alpha)$ chooses $r \leftarrow \mathbb{Z}_{p_{\alpha}}$, and outputs a uniform $x := (x', x'') = (g_{\alpha}^r, h^r) \in D_\alpha$.
– Evaluator $F$: For every $\alpha \in I_n$, and $x = (x', x'') \in D_\alpha$, $F(\alpha, x = (x', x''))$ outputs $y := x' \in R_\alpha$ if $(g_{\alpha}, h, x', x'')$ is a DDH-tuple and ⊥ otherwise.
– Doubly-enhanced Sampler: For every $\alpha \in I_n$, $S'_R$ obtains and image $x \leftarrow S(\alpha)$, and explains its image as $u \leftarrow \hat{E}(\alpha, f_\alpha(x))$ s.t. $S_R(\alpha; u) = f_\alpha(x)$, chooses $r \leftarrow \mathbb{Z}_{p_{\alpha}}$, sets $x = (x', x'') = (f_\alpha(x), f_\alpha(x'))$, and outputs $(x, u)$.
– Inverter $F^{-1}$: For every $(\alpha, \tau) \in [G(1^n)]$, and $y \in R_\alpha$, $F^{-1}_\alpha(\alpha, \tau, y)$ outputs $z := (z', z'') = (y, y')$.
– Certificate Generator $\Pi_G$: Outputs an empty certificate $\pi$.
– Certificate Verifier $\Pi_V$: On input $(\alpha, \pi, x, y)$, $\Pi_V$ accepts iff $y = F_\alpha(\alpha, x)$.

**Theorem 2.** The collection $\mathbb{F} = \{f_\alpha : D_\alpha \rightarrow R_\alpha\}_{\alpha \in I}$ of Const. 2 is a collection of 1-1 devTDF.

The proof of Theorem 2 is given in Appendix B.2. We remark that the bijectivity of the collection is a key to establish it as a collection of 1-1 devTDF. With the preimages themselves treated as certificates of unique invertibility, the certificate generator $\Pi_G$ is asked of nothing, and the certificate verifier $\Pi_V$ checks whether $y = x' = F_\alpha(\alpha, x)$ holds, which implicitly assumes checking that $(g_{\alpha}, h, x', x'')$ constitutes a DDH tuple, which is easy to do publicly, thanks to the easiness of the DDHP.

The explainable domain property was used in an essential manner to prove doubly-enhanced property.

## 5 Conclusion

Although collections of 1-1 devTDF were introduced to construct NIZK proofs, they might find their own applications. In particular, they can be considered certified collections for practical reasons. Certification on the image level as in 1-1 devTDF, rather than on the range level, is not restricted to doubly-enhanced functions, and hence can be applied to other functions in principle.
References


A Efficient Instantiations of the HBM: Relaxed Version

In this section, we relax collections of 1-1 devTDF and define collections of $\epsilon$-weak 1-1 doubly-enhanced verifiable trapdoor functions, and use them to efficiently implement the HBM, however, with relaxed soundness and completeness errors.

A.1 $\epsilon$-Weak 1-1 Doubly-Enhanced Verifiable Trapdoor Functions

We define $\epsilon$-weak 1-1 devTDF, which is a relaxation of 1-1 devTDF, such that completeness and soundness are allowed to fail on at most an $\epsilon$ fraction of the designated range.

**Definition 11 (Collections of $\epsilon$-Weak 1-1 Doubly-Enhanced Verifiable Trapdoor Functions).** Let $\mathbb{F} = \{f_\alpha : D_\alpha \to R_\alpha\}_{\alpha \in I}$ be a collection of deTDF with generator $\mathcal{G} = (G, S_D, S_R, S'_R, F, F^{-1})$ and $(\alpha, \tau) \in [G(1^n)]$. Furthermore, let $\epsilon : \mathbb{N} \to (0,1]$ be polynomial-time computable. Then $\mathbb{F}$ is a collection of $\epsilon$-Weak 1-1 Doubly-Enhanced Verifiable Trapdoor Function, denoted $\epsilon$-weak 1-1 devTDF, if there exists a duplet of PPT algorithms $(\Pi_G, \Pi_V)$ such that $(\Pi_G, \Pi_V)$ is as in Def. 7. We require the following:

- **Completeness:** For all $(\alpha, \tau) \in [G(1^n)]$, and $y \in R_\alpha$, if $\pi \leftarrow \Pi_G(\tau, y)$, then $\Pr[1 \leftarrow \Pi_V(\alpha, f^{-1}_\alpha(y), y, \pi)] \geq 1 - \epsilon$, where the probability is taken over the random choices of $y \in R_\alpha$ and the random coins of $\Pi_G$ and $\Pi_V$.

- **Soundness:** For all indices $\alpha$, and all strings $y, x_1, x_2, \pi_1, \pi_2$ such that $x_1 \neq x_2$, then $\Pr[0 \leftarrow \Pi_V(\alpha, x_j, y, \pi_j)] \leq \epsilon$ for at least one $j \in \{1, 2\}$, where the probability is taken over the random choices of $y \in R_\alpha$ and the random coins of $\Pi_G$ and $\Pi_V$.

Moreover, $\mathcal{G} = (G, S_D, S_R, S'_R, F, F^{-1}, \Pi_G, \Pi_V)$ is the generator of the collection of $\epsilon$-weak 1-1 devTDF.

The same remarks and properties of collections of 1-1 devTDF carry on to collections of $\epsilon$-weak 1-1 devTDF except that they hold for at least a $(1 - \epsilon)$ fraction of the designated range. Although a collection of $\epsilon$-weak 1-1 devTDF with $\epsilon = 0$ is a collection of (exact) 1-1 devTDF, Def. 11 requires $\epsilon > 0$.

A.2 Efficient Implementation of HBM based on $\epsilon$-weak 1-1 devTDF

In this section, we provide an efficient implementation of the HBM based on collections of $\epsilon$-weak 1-1 devTDF with corresponding polynomial-time computable hardcore predicate. Whereas the use of collections of 1-1 devTDF yields an efficient implementation of the HBM without introducing any extra error, the use of $\epsilon$-weak 1-1 devTDF results in an additional error in both completeness and soundness of the NIZK proof system. However, such an additional errors are unavoidable and can be reduced by repetition.

**Theorem 3.** Let $(P, V, S)$ be a NIZK proof system in the HBM for a language $L \in \text{NP}$ as in Def. 6. Then given a collection of $\epsilon$-Weak 1-1 Doubly-Enhanced Verifiable Trapdoor Function, $\epsilon$-weak 1-1 devTDF, $\mathbb{F} = \{f_\alpha : D_\alpha \to R_\alpha\}$ with generator $\mathcal{G} = (G, S_D, S_R, S'_R, F, F^{-1}, \Pi_G, \Pi_V)$, such that $R_{S_R} = \{0,1\}^l$ is the randomness space of the range sampler $S_R$, and a corresponding polynomial-time computable hardcore predicate $b : D_\alpha \to \{0,1\}$. Furthermore, let $|x| = n$ for $x \in L$, $p(\cdot)$ be polynomial, and $\epsilon(\cdot) \leq \frac{1}{p(\cdot)}$. Then $(P, V, S)$ can be transformed into a NIZK proof system $(P', V', S')$ in the standard model for the same language $L$, with completeness, soundness, and zero-knowledge as follows:
\[ \Pr[V'(x, \alpha, \sigma', \pi) = 1 : \sigma' \leftarrow \{0,1\}^{p(n)-1}; \pi \leftarrow P'(x, \alpha, \tau, \sigma')] \geq \frac{2}{3} - \frac{2}{3} \epsilon(n)p(n). \]

- **Completeness:** For every \( x \in L \), and every \( f_\alpha \in \{0,1\}^{|G(1^n)|} \) with \((\alpha, \tau) \in G(1^n)\),

\[ \Pr[V'(x, \alpha, \sigma', \pi) = 1 : \sigma' \leftarrow \{0,1\}^{p(n)-1}; \pi \leftarrow P'(x, \alpha, \tau, \sigma')] \geq \frac{2}{3} - \frac{2}{3} \epsilon(n)p(n). \]

- **Soundness:** For every \( x \notin L \), every \( f_\alpha \) with index \( \alpha \), and every algorithm \( A \),

\[ \Pr[V'(x, \alpha, \sigma', \pi) = 1 : \sigma' \leftarrow \{0,1\}^{p(n)-1}; \pi \leftarrow A(x, \alpha, \sigma')] \leq \frac{1}{3} + \frac{1}{3} \epsilon(n)p(n). \]

- **Zero-Knowledge:** The following two ensembles are computationally indistinguishable:

1. \( \{S'(x)\}_{x \in L} \)
2. \( \{(x, \sigma', \pi) : \sigma' \leftarrow \{0,1\}^{p(n)-1}; \pi \leftarrow P'(x, \sigma', \alpha, \tau)\}_{x \in L} \)

The completeness and soundness errors are adjusted by observing the the dependency between \( \epsilon(n) \) and \( p(n) \). As \( p(n) \) is given from the NIZK proof system of the HBM, which represents the length of the HBM-CRS, the additional errors \( \epsilon(n)p(n) \) can only be made small by choosing a weak 1-1 devTDF for which \( \epsilon \) is small; the smaller the fraction \( \epsilon \), the smaller \( \epsilon(n)p(n) \) can be made. The statements of Theorem 3 and Theorem 1 coincide if \( \epsilon = 0 \). However, by definition, \( \epsilon \in (0,1] \) and hence can not be 0. Hence we have to deal with an additional error.

**Proof.** The proof is, in spirit, similar to the proof of Theorem 1 so we refrain from repeating the similarities and contend by pinpointing the dissimilarities.

The construction of \( (P', V', S') \) is the same as of Const. 1 except that \( P', V', \) and \( S' \) succeed in implementing the HBM with additional error probability due to the use of \( \epsilon \)-weak 1-1 devTDF.

Recall that the CRS is viewed as \( \sigma' = \sigma'_1 \cdots \sigma'_m \), where \( m := p(n) \), and the HBM CRS \( \sigma = \sigma_1 \cdots \sigma_m \), where \( \sigma_i = \beta(f_{\alpha}^{-1}(S_R(\alpha, \sigma'_i))). \)

Let the sample space \( S \) be the set of all strings of length \( ml \), i.e., \( S := \{0,1\}^{ml} \), then \( \sigma' \leftarrow S \) where the (standard model) CRS \( \sigma' \) is viewed as \( \sigma' = \sigma'_1 \cdots \sigma'_m \), and the corresponding HBM CRS \( \sigma = \sigma_1 \cdots \sigma_m \), where \( \sigma_i = \beta(f_{\alpha}^{-1}(S_R(\alpha, \sigma'_i))). \) Then let \( X_{\sigma'} \) be Bernoulli random variable over \( S \) such that \( X_{\sigma'} = 0 \) if there exists an \( i \in \{1, \ldots, m\} \) such that \( \Pr[1 \leftarrow \Pi_V(\alpha, p_i, S_R(\alpha; \sigma'_i), \mu_i)] \leq \epsilon \) where \( \mu_i \leftarrow \Pi_G(\tau, S_R(\alpha; \sigma'_i)) \), and \( X_{\sigma'} = 1 \) if for all \( i \) it holds that \( \Pr[1 \leftarrow \Pi_V(\alpha, f_{\alpha}^{-1}(S_R(\alpha; \sigma'_i)), S_R(\alpha; \sigma'_i), \mu_i)] \geq (1 - \epsilon) \). If \( X_{\sigma'} = 1 \), we call \( \sigma' \) good, otherwise, it is called bad.

Observe that whereas \( \Pr[X_{\sigma'} = 1] \) depicts the probability that \( \Pi_V \) accepts a good \( \sigma' \), \( \Pr[X_{\sigma'} = 0] \) depicts the probability of accepting a bad \( \sigma' \). Furthermore, whereas \( \Pr[X_{\sigma'} = 1] \) captures the probability of acceptance in the completeness condition of \( \epsilon \)-weak 1-1 devTDF, \( \Pr[X_{\sigma'} = 0] \) captures the probability of acceptance in the soundness condition.

To simplify the notion, let \( X_V \) and \( X_{V'} \) be Bernoulli random variables such that \( X_V = 1 \) if and only if \( V \) accepts, that is, \( X_V = V(x, \sigma_R, R, \pi) \), and \( X_{V'} = 1 \) if and only if \( V' \) accepts, that is, \( X_{V'} = V'(x, \alpha, \sigma', \rho) \).

The completeness condition captures an honest prover’s ability to convince the verifier to accept correct proofs. Utilizing our random variables just defined, the completeness condition can be rewritten as,

\[ \Pr[X_{V'} = 1] \geq \Pr[X_V = 1 \cap X_{\sigma'} = 1] \geq \Pr[X_V = 1] \times \Pr[X_{\sigma'} = 1] \geq \frac{2}{3} \times (1 - \epsilon)^m \geq \frac{2}{3} - \frac{2}{3} \epsilon m \]
The inequality of (1) follows from the fact that the prover $V'$ might succeed even if $\sigma'$ is bad; it might not need to prove unique invertibility of some bad image. Equation (2) follows from the independence between $X_{\sigma'}$ and $X_V$. To see this, note that given the revealed bits, the verifier’s acceptance or rejection is oblivious to whether the revealed bits are uniquely opened or not, and hence independent of $X_{\sigma'}$. $Pr[X_V = 1] \geq \frac{2}{3}$ and $Pr[X_{\sigma'} = 1] = (1 - \epsilon)^m$ respectively follow from the completeness condition of the HBM, and that of $\epsilon$-weak 1-1 devTDF. Equation (4) follows from applying Taylor theorem to bound $(1 - \epsilon(n))^m$ for $\epsilon(n) \in [0, 1]$ as:

$$
(1 - \epsilon(n))m \leq (1 - \epsilon(n))^m \leq 1 .
$$

The soundness condition depicts a cheating prover’s ability to convince the verifier to accept false proofs. The verifier accepts a false proof given either a good or bad $\sigma'$. Utilizing our random variables, the soundness condition can be rewritten as,

$$
Pr[X_V' = 1] = Pr[X_V = 1 \cap X_{\sigma'} = 1] + Pr[X_V = 1 \cap X_{\sigma'} = 0]
$$

$$
= Pr[X_V = 1] \times Pr[X_{\sigma'} = 1] + Pr[X_V = 1] \times Pr[X_{\sigma'} = 0]
$$

$$
\leq \frac{1}{3} \times (1 - \epsilon)^m + \frac{1}{3} \times (1 - (1 - \epsilon)^m)
$$

$$
\leq \frac{1}{3} + \frac{1}{3m}
$$

The first summand of (6) depicts the acceptance probability of $V'$ on a false proof with certificates of unique invertibility being honestly generated, and the second summand depicts the acceptance probability of $V'$ on a false proof with certificates not being honestly generated. While $Pr[X_{\sigma'} = 1]$ depicts the honest openings of the hardcore bits, $Pr[X_{\sigma'} = 0]$ depicts a dishonest openings. The transition between (8) and (9) follows from the bound of (5).

With zero-knowledge being the same as in Theorem 1 this concludes the proof. \(\square\)

### A.3 Instantiation of $\epsilon$-Weak 1-1 Doubly-Enhanced Verifiable Trapdoor Functions

In this section, we show how to instantiate $\epsilon$-weak 1-1 devTDF from (non-certified) deTDP using a certifying procedure due to Bellare and Yung [BY92].

Bellare and Yung [BY92] showed in NIZK how to certify a map to be almost permutation. Specifically, let $p(\cdot)$ be some polynomial, they showed in NIZK, how to certify a map to be 1-1 on at least $(1 - \frac{1}{p(n)})$ fraction of the designated range.

The permutations certified by BY-procedure are assumed to have simple domains, i.e., maps of the form $\gamma : \{0,1\}^n \rightarrow \{0,1\}^n$. This assumption is wlog in the sense that all popular TDPs, such as RSA-based whose domain is $\mathbb{Z}_n^*$, meet this assumption after minor transformations [BY92]. However, such transformations are inefficient. Therefore, we follow [Gol11] and show how to use the procedure of [BY92] to any doubly-enhanced trapdoor permutation, whose domain does not need to be equal to $\{0,1\}^n$.

**Definition 12 ([BY92]).** Let $n > 0$ and $f : \{0,1\}^n \rightarrow \{0,1\}^n$. Let $C(f) = \{ y \in \{0,1\}^n : |f^{-1}(y)| > 1 \}$, and $\epsilon \in [0, 1]$, then $f$ is called $\epsilon$-permutation if $|C(f)| \leq \epsilon 2^n$.

---

5 A possible transformations is due to Yao and is given in [BM92]. Yao’s construction can be applied to any function whose domain size is at least a polynomial fraction of $\{0,1\}^n$ and is efficiently recognizable.
In the following, we state Lemma 1 which is a natural extension to the certification procedure of \[BY92\]. Collections of deTDP passing the certification of Lemma 1 are certified to be \(\epsilon\)-permutations, that is, permutations on at least \((1 - \epsilon)\) fraction of the designated range.

**Lemma 1.** Let \(F = \{f_\alpha : D_\alpha \to R_\alpha\}_{\alpha \in I}\) be a collection of deTDP with an index set \(I\) and \(I_n = \{0,1\}^{p(n)}\) for some polynomial \(p(\cdot)\), and generator \(G = (G, S_D, S_{R}, S'_{R}, F, F^{-1})\) such that \(R_{SR} = \{0,1\}^I\) is the randomness space of the range sampler \(S_R\). Let \(\epsilon : \mathbb{N} \to (0,1]\) be polynomial-time computable and \(\epsilon^{-1}\) be polynomially bounded. Then there exists a triplet of algorithm \((P, V, S)\) such that the following holds:

- **Completeness:** Suppose \(f_\alpha\) is a permutation, with \((\alpha, \tau) \in [G(1^n)]\) then,
  \[
  Pr[V(\alpha, \sigma, \rho) = 1 : \sigma \leftarrow \{0,1\}^{\epsilon^{-1}(l)l}; \rho \leftarrow P(\alpha, \tau, \sigma)] = 1
  \]

- **Soundness:** Suppose \(f_\alpha\) is not an \(\epsilon(l)\)-permutation, then for every algorithm \(A\),
  \[
  Pr[V(\alpha, \sigma, \rho) = 1 : \sigma \leftarrow \{0,1\}^{\epsilon^{-1}(l)l}; \rho \leftarrow A(\alpha, \sigma)] \leq \frac{1}{2}
  \]

- **Zero-knowledge:** Suppose \(f_\alpha\) is a permutation. Then the following distributions are identical:
  1. \([S(\alpha)]_{\alpha \in I_n}\)
  2. \([((\alpha, \sigma, \rho) : \sigma \leftarrow \{0,1\}^{\epsilon^{-1}(l)l}; \rho \leftarrow P(\alpha, \tau, \sigma)]_{\alpha \in I_n}\)

A couple of remarks is in order. First, notice the gap between the completeness and soundness conditions. Whereas in the completeness condition the verifier \(V\) accepts if \(f_\alpha\) is permutation, and rejects in the soundness condition if \(f_\alpha\) is not \(\epsilon(l)\)-permutation, nothing is said about \(f_\alpha\) when it is \(\epsilon(l)\)-permutation. That is, if \(f_\alpha\) is \(\epsilon(l)\)-permutation, Lemma 1 guarantees nothing. Second, whereas the completeness holds with certainty, soundness holds with an error of up to \(\frac{1}{2}\), which can be further reduced to negligible by repetition. Third, zero-knowledge is perfect.

In the sequel, we give a construction (Const. 3) for which Lemma 1 holds such that its proof is mutatis mutandis identical to that of \[BY92\].

**Construction 3** Let \(F, \epsilon, \) and \((P, V, S)\) be as in Lemma 1. Furthermore, let \((\alpha, \tau) \in [G(1^n)]\). Then, following is a construction of the triplet \((P, V, S)\):

- **Common Input:** \((\sigma, \alpha)\).
- **CRS \(\sigma = \sigma_1 \cdots \sigma_{\epsilon^{-1}(l)}\) where \(\sigma_i \leftarrow \{0,1\}^l\). We treat \(\sigma_i\) as the randomness used by \(S_R\) to sample images from \(R_\alpha\), i.e., \(S_R(\alpha; \sigma_i) \in R_\alpha\).**
- **Prover \(P\):** Given \(\tau\) as an auxiliary input, compute preimages of \(\sigma = \sigma_1 \cdots \sigma_{\epsilon^{-1}(l)}\) as \(p_i = f_\alpha^{-1}(S_R(\alpha; \sigma_i))\), set \(p = p_1 \cdots p_{\epsilon^{-1}(l)}\), and output \(p\).
- **Verifier \(V\):** Given \(P\)’s output \(p\), check the correctness of the received preimages \(p = p_1 \cdots p_{\epsilon^{-1}(l)}\), that is, check whether \(f_\alpha(p_i) = S_R(\alpha; \sigma_i)\) for all \(i = 1, \ldots, \epsilon^{-1}(l)\), and reject and halt in case of a mismatch. Otherwise, accept and halt.
- **Simulator \(S\):** Simulate \(\sigma = \sigma_1 \cdots \sigma_{\epsilon^{-1}(l)}\) as \(\hat{\sigma} = \hat{\sigma}_1 \cdots \hat{\sigma}_{\epsilon^{-1}(n)}\) and the preimages \(p = p_1 \cdots p_{\epsilon^{-1}(l)}\) as \(\hat{p} = \hat{p}_1 \cdots \hat{p}_{\epsilon^{-1}(l)}\) s.t. for each \(i = 1, \ldots, \epsilon^{-1}(l)\), run \(S'_{R}\) on \(\alpha\) to obtain \((\hat{p}_i, \hat{\sigma}_i) \leftarrow S'_{R}(\alpha)\). Output \((\hat{\sigma}, \hat{p})\).
Observe that $|\sigma| = \epsilon^{-1}(l) \cdot l$, and hence for $|\sigma|$ to be polynomial (in $l$), $\epsilon^{-1}$ has to be polynomially bounded. This length dependency between $\sigma$ and $\epsilon$ is intuitive in the sense that the larger $\epsilon(l)^{-1}$ is, the larger $|\sigma|$ must be, to certify that $f_\alpha$ is an $\epsilon(l)$-permutation.

Intuitively speaking, if $f_\alpha$ is a permutation, the prover $P$, given trapdoor $\tau$, succeeds with certainty, and hence the completeness condition. On the other hand, if $f_\alpha$ is not an $\epsilon(l)$-permutation, then $A$ fails on at least an $\epsilon$ fraction of all images under $f_\alpha$. Therefore, on uniformly chosen images corresponding to $\sigma$, $A$ succeeds with probability at most $(1 - \epsilon(n))^{\epsilon^{-1}(n)} \leq \frac{1}{2}$, and hence the soundness condition. Moreover, revealing random preimages, as given by the prover, reveals no information about $f_\alpha$, and hence the proof is zero-knowledge.

Last, Const. 3 can be instantiated with any collections of (non-certified) deTDP, and hence with RSA, for example (cf. [Gol11]).

**Proposition 1.** The collection of devTDF $\mathbb{F}$ of Construction 3 is a collection of $\epsilon$-weak 1-1 devTDF.

Lemma 4 guarantees that any collection of deTDP is a collection of $\epsilon$-permutation deTDP, where $\epsilon$ is polynomially bounded, and hence imply that the collection is $\epsilon$-weak 1-1 devTDF, with the preimages serving as certificates $\pi$ of unique invertibility and hence the certificate generator is asked of nothing $\Pi_G$, and the certificate verifier $\Pi_V$ only checks the correctness of the preimages.

Last, we note that collections of deTDP instantiate collections of $\epsilon$-weak 1-1 devTDF in a special way in the sense that the completeness condition holds with certainty, and hence when used to implement the HBM results in no extra completeness error, but with extra soundness error as indicated in Theorem 3 and that is the state-of-the-art for NIZK proofs (cf. [Gol11]).

### B Proofs

#### B.1 Proof of Theorem 4

We proceed by proving completeness, soundness, and zero-knowledge of the triplet $(P', V', S')$ of Const. 4 and hence Theorem 4 follows.

**Proof.** We show that $(P', V')$ is non-interactive proof system, by showing that it emulates $(P, V)$ in a computationally indistinguishable manner. On the one hand, recall that the HBM-CRS $\sigma$ is computed as a sequence of hardcore bits, which by assumptions makes it computationally indistinguishable from random HBM-CRS. On the other hand, $V'$ recovers the exact revealed bits $\sigma_R$. This follows from the unique invertibility of $f_\alpha$. Hence, we conclude that $(P', V')$ emulates $(P, V)$ in a computationally indistinguishable manner, and therefore completeness and soundness of $(P', V')$ follow from completeness and soundness of $(P, V)$ and of 1-1 devTDF.

To prove zero-knowledge, we need to show that the simulated and real ensembles are computationally indistinguishable. Given an HBM simulator $S$, $(\hat{\pi}, \hat{R})$ and $(\pi, R)$ are computationally indistinguishable. Also $p_R$ and $\hat{p}_R$ are random and hence identical. So, it suffices to prove computational indistinguishability of $(\sigma, \sigma')$ and $(\hat{\sigma}, \hat{\sigma}')$. But $(\sigma_R, \sigma'_C)$ and $(\hat{\sigma}_R, \hat{\sigma}'_C)$, where $C = (c_i)_{i \in \{1, \ldots, p(n)\}} \setminus \{r_j: r_j \in R\}$ and $\hat{C} = (\hat{c}_i)_{i \in \{1, \ldots, p(n)\}} \setminus \{\hat{r}_j: \hat{r}_j \in \hat{R}\}$ are random and hence identical. Zero-knowledge reduces to proving computational indistinguishability of $(\sigma_R, \sigma'_R)$ and $(\hat{\sigma}_R, \hat{\sigma}'_R)$, which is captured by the statement of Lemma 2.

**Lemma 2.** Let $b, f_\alpha$ be as in Const. 4 and $l(\cdot)$ be some polynomial. Then define $X_n = (x_i)_{i \in \{1, \ldots, l(n)\}}$ where $x_i \leftarrow \{0, 1\}^b$, $R_n = (r_i)_{i \in \{1, \ldots, l(n)\}}$ where $r_i = b(f_\alpha^{-1}(S_R(\alpha; x_i)))$, and $U_n = (u_i)_{i \in \{1, \ldots, l(n)\}}$ where $u_i \leftarrow \{0, 1\}$. Then the following join distributions are computationally indistinguishable:
1. \(X = \{(X_n, U_n)\}_{n \in N} = \{(x_i, u_i)\}_{i \in \{1, \ldots, l(n)\}}\)

2. \(Y = \{(X_n, R_n)\}_{n \in N} = \{(x_i, r_i)\}_{i \in \{1, \ldots, l(n)\}}\)

**Proof.** The proof of the lemma is by reducibility argument. We show that the existence of an efficient algorithm that distinguishes \((X_n, U_n)\) and \((X_n, R_n)\) with noticeable probability, implies the existence of an efficient algorithm that distinguishes hardcore bits from random bits with noticeable probability, which contradicts the hardcore bit predicate hypothesis. The implication is proven by a hybrid argument as follows.

Suppose, to the contrary, that there exists a PPT distinguisher \(D\) that distinguishes \((X_n, U_n)\) and \((X_n, R_n)\) with noticeable probability, i.e., there exists a polynomial \(p(\cdot)\) such that for sufficiently large \(n\),

\[
\delta(n) := |\Pr[D((X_n, U_n)) = 1] - \Pr[D((X_n, R_n)) = 1]| > \frac{1}{p(n)}.
\]

Then, for every \(i\), with \(i \in \{0, l - 1\}\) where \(l := l(n)\), define the hybrid sequence \(H^i_n\) as

\[
H^i_n := ((x_1, u_1), \ldots, (x_i, u_i), (x_{i+1}, r_{i+1}), \ldots, (x_l, r_l)).
\]

Observe that the hybrid \(H^0_n = ((x_1, r_1), \ldots, (x_l, r_l))\) coincides with \((X_n, R_n)\), and \(H^i_n = ((x_1, u_1), \ldots, (x_l, u_l))\) coincides with \((X_n, U_n)\). By hypothesis, \(D\) distinguishes the extreme hybrids, \(H^0_n\) and \(H^i_n\), with noticeable probability. Now we show how to use \(D\) to construct a PPT algorithm \(D'\) that distinguishes hardcore bits from random bits, with noticeable probability, and hence derive a contradiction to the hardcore predicate hypothesis. Following is the description of \(D'\) that uses the doubly-enhanced sampler \(S'_R\) in an essential manner.

- Input: A tuple \(\gamma = (\gamma_1, \gamma_2)\) such that either \(\gamma \in (X_n, U_n)\) or \(\gamma \in (X_n, R_n)\).
- Choose an index \(i \leftarrow \{0, 1, \ldots, l - 1\}\)
  1. Sample \((x_1, u_1), \ldots, (x_i, u_i)\) such that \((x_j, u_j) \leftarrow \{0, 1\}^n \times \{0, 1\}\).
  2. Sample \((x_{i+2}, r_{i+2}), \ldots, (x_l, r_l)\) such that \((x_j, r_j) = (r, b(x))\) and \((x, r) \leftarrow S'_R(\alpha)\).
- Output \(D(((x_1, u_1), \ldots, (x_i, u_i), \gamma = (\gamma_1, \gamma_2), (x_{i+2}, r_{i+2}), \ldots, (x_l, r_l)))\) and halt.

Algorithm \(D'\) can be implemented efficiently provided \(D\) is efficient and the existence of the doubly-enhanced sampler \(S'_R\). Observe the essential role of \(S'_R\) in generating the sequence \((x_{i+2}, r_{i+2}), \ldots, (x_l, r_l)\).

Now we look at the correctness of \(D'\) as well as its success probability. Observe that \(D'\), when invoked on input \(\gamma\), chooses \(i \leftarrow \{0, \ldots, l - 1\}\), and invokes \(D\) on \(((x_1, u_1), \ldots, (x_i, u_i), \gamma = (\gamma_1, \gamma_2), (x_{i+2}, r_{i+2}), \ldots, (x_l, r_l))\), then it behaves as \(D(H^i_n)\) if \(\gamma \in (X_n, R_n)\) and as \(D(H^{i+1}_n)\) if \(\gamma \in (X_n, U_n)\). This means that \(D'\) can distinguish neighboring hybrids, and hence can distinguish random bits from hardcore bits with the same success probability of \(D\) on neighboring hybrids.

Then we show that \(D\) can distinguish neighboring hybrids with noticeable probability, namely \(\frac{\delta(n)}{l}\), provided it distinguishes extreme hybrids with noticeable probability \(\delta(n)\).

**Claim.** \(|\Pr[D(H^0_n) = 1] - \Pr[D(H^{i+1}_n)]\) = \(|\Pr[D(H^0_n) = 1] - \Pr[D(H^{i+1}_n)]\)| \(> \frac{\delta(n)}{l}\).

**Proof.** By construction, \(D'\) invokes \(D\) on \(((x_1, u_1), \ldots, (x_i, u_i), \gamma, (x_{i+2}, r_{i+2}), \ldots, (x_l, r_l))\) where \(i \leftarrow \{0, 1, \ldots, l - 1\}\), and hence it follows that

\[
|\Pr[D(H^0_n) = 1] - \Pr[D(H^{i+1}_n)]\) = \(\frac{1}{l} \sum_{i=0}^{l-1}(|\Pr[D(H^i_n) = 1] - \Pr[D(H^{i+1}_n)]\) | \(> \frac{\delta(n)}{l}\).
\]

Hence the claim follows. \(\Box\)
This completes the proof of Lemma 2 by reaching a contradiction to the hardcore predicate hypothesis of \( b \), and hence there exists no efficient algorithm that distinguishes \( X \) and \( Y \) with noticeable probability.

This completes the proof of Theorem 1 and shows that \((P', V', S')\) is a NIZK proof system for \( L \) in the standard model.

\( \square \)

\textbf{B.2 Proof of Theorem 2}

\textit{Proof.} We first establish the bijectivity of \( F \). Observe that \( F = \{ f_\alpha : (g_{\alpha}^r, h_r) \rightarrow g_{\alpha}^r \} \) is injective as distinct \( h_r \) is paired with \( g_{\alpha}^r \), and that \( |D_\alpha| = |R_\alpha| \). Hence bijectivity follows.

Given Const. 2, we only need to show that \( F \) is enhanced and one-way. Recall that, for one-wayness to hold, it must be that for every PPT algorithm \( A \),

\[
\Pr[f_\alpha(z) = y : \alpha \leftarrow I_\alpha; x \leftarrow D_\alpha; y = f_\alpha(x); z \leftarrow A(\alpha, y)] \leq \text{neg}(n). \]

Assume to the contrary that \( A \) can invert \( f_\alpha \) with non-negligible probability, then it can solve the CDHP in GDH groups with the same probability, and hence contradicts the hypothesis of GDH groups (Def. 8). Concretely, for \( A \) to succeed in inverting \( f_\alpha \) on \( y = g_{\alpha}^r \), it means that given \((g_{\alpha}, g_{\alpha}^r, h = g_{\alpha}^r)\), it computes \( h_r = g_{\alpha}^{r_r} \) and hence solves the CDHP in \( G_{\hat{\alpha}} \in \hat{G} \). Therefore, no such PPT algorithm exists.

Observe that \( F \) possesses explainable domains, i.e., algorithm \( \hat{E} \), which implies that \( F \) is enhanced in the sense of Def. 2. For otherwise, \( \hat{E} \) can be used to solve CDHP efficiently, which leads to contradiction.

\( \square \)