Limits on the Power of Private Constrained PRFs

Mengda Bi
Tsinghua University, Beijing, China

Yaohua Ma
Tsinghua University, Beijing, China

Chenxin Dai
Tsinghua University, Beijing, China

Abstract

Private constrained PRFs are constrained PRFs where the constrained key hides information about the predicate circuit. Although there are many constructions and applications of PCPRF, its relationship to basic cryptographic primitives, such as one-way functions and public-key encryptions, has been unclear. For example, we don't know whether one-way functions imply PCPRFs for general predicates, nor do we know whether 1-key secure PCPRF for all polynomial-sized predicates imply public-key primitives such as public-key encryption and secret-key agreement.

In this work, we prove the black-box separation between a 1-key secure PCPRF for any predicate and a secret-key agreement, which is the first black-box separation result about PCPRF. Specifically, we prove that there exists an oracle relative to which 1-key secure PCPRFs exist while secret-key agreement does not. Our proof is based on the simulation-based technique proposed by Impagliazzo and Rudich (STOC 89). The main technical challenge in generalizing the simulation-based technique to PCPRF is the issue of *unfaithfulness* of Eve's simulation to the real world because our oracle is more complicated than a random oracle. We introduce a new technique which we call the "weighting" technique and show how to leverage it to circumvent the issue of unfaithfulness in the proof framework of Impagliazzo and Rudich.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives

Keywords and phrases Black-box Separations, Private Constrained PRFs, Key Agreement

Digital Object Identifier 10.4230/LIPIcs.ITC.2026.2

Related Version *Full Version*: `https://eprint.iacr.org/2025/1196.pdf`

1 Introduction

1.1 Private Constrained Pseudorandom Functions

Constrained pseudorandom functions (CPRFs) are useful building blocks in cryptography [6, 21, 8]. Suppose $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a pseudorandom function (PRF), F is called a CPRF if it additionally exhibits the following functionality: given a master key $k \in \mathcal{K}$, and a binary predicate $c : \mathcal{X} \to \{0, 1\}$, one can derive a constrained key $k[c]$. $k[c]$ can be used to evaluate $F(k, x)$ if $c(x) = 1$. However, if $c(x) = 0$, $k[c]$ should not reveal non-trivial information of $F(k, x)$.

Since the concept of CPRF was first proposed, there have been constructions of CPRF for various predicate classes under various assumptions. For example, CPRF for the simplest predicate, a point indicator function, also known as puncturable PRF, is shown to be constructable from one-way functions (OWFs) using the classical GGM tree-based PRF [18, 6, 8, 21]. Currently, the most expressive predicate family that can be constructed from OWF are the CNF predicates [13]. Other constructions require structural assumptions such as LWE [11], DDH [2], pairing [6, 5], or even stronger primitives such as indistinguishability obfuscation (iO) [7, 19, 5, 3, 13].

A CPRF can further be *private*[5], if the constrained key k_c reveals no information about the predicate c. A private CPRF is also abbreviated as PCPRF. The notion of privacy varies depending on how many constrained keys the adversary is allowed to obtain. Multi-key secure PCPRF for $\mathsf{P/poly}$ predicates is only known to be constructible from iO [5]. In fact, even 2-key secure PCPRF for a predicate class implies iO for the same circuit class [12]. To the best of our knowledge, without iO, multi-key privacy is only achievable for left-right predicates in the random oracle model from bilinear maps [6]. In contrast, PCPRF with 1-key privacy for all polynomial-sized predicates is known to be constructable from the LWE assumption [10].

Even with just 1-key privacy, PCPRF still has wide applications in cryptography because of its powerful predicate-hiding ability. Known applications include searchable encryption, watermarking, deniable encryption and others in [5], private information retrieval [27, 22], and 1-key secure private-key functional encryption [12]. However, even though PCPRFs seem to be powerful, we still do not know how to construct secret-key agreement – the fundamental primitive in Cryptomania – from 1-key secure PCPRFs. Therefore, it is natural to ask whether 1-key secure PCPRFs fall within Minicrypt or Cryptomania.

1.2 Black-Box separation

In cryptography, it is almost impossible to directly prove separation results in the form of "the existence of a certain primitive P is not sufficient to construct another primitive Q". Instead, great progress has been made in proving the so-called "black-box separations". A black-box separation refers to a cryptographic impossibility result showing that Q cannot be achieved using only black-box access to P. In other words, it demonstrates that there exists no efficient reduction from Q to P when the reduction is restricted to using P as a black-box, without any knowledge of its internal structure. This concept is fundamental in understanding the limitations of cryptographic primitives and provides insights into the relationship between different cryptographic primitives.

One of the most significant black-box separation results is the seminal work of Impagliazzo and Rudich [20]. It shows that OWF cannot be used to construct secret-key agreement (KA) in a black-box way, employing a relativizing technique in which an oracle is constructed such that OWFs exist but KA does not, relative to this oracle. Their result naturally establishes a black-box separation between the following two classes: (1) Minicrypt: comprising primitives that are implied by OWFs, such as pseudorandom generators and pseudorandom functions; (2) Cryptomania: comprising all primitives that imply KA in a black-box way, such as public key encryption and oblivious transfer.

Inspired by [20], researchers have developed black-box separation techniques to obtain various impossibility results, including [25, 28, 15, 16]. In this work, we mainly focus on the impossibility results concerning KA. Significant progress has been made in this field by many researchers: [4] improves the efficiency of the adversary in [20], and [9] simplifies the analysis for perfectly complete KA protocols.

1.3 Motivations and Questions

Although significant work has been devoted to the study of PCPRF, its relationship with OWFs and cryptographic primitives in Cryptomania remains unclear. Firstly, we do not know how to use OWFs to construct PCPRFs for more expressive predicate classes than

CNF (even providing only 1-key security). Secondly, we do not know how to use 1-key secure PCPRFs to construct any primitives in Cryptomania.[1]

Naturally, we aim to establish black-box separation results. Specifically, we propose the following questions:

- Open question 1: *Can we establish or refute black-box separation results between OWFs and more expressive PCPRFs?*
- Open question 2: *Can we establish or refute black-box separation results between 1-key secure PCPRFs and primitives in Cryptomania?*

1.4 Our Result

In this work, we prove the black-box separation between 1-key PCPRF for arbitrary predicate classes and KA, which gives a positive result regarding the second open question. Concretely, we prove the following theorem:

▶ **Theorem 1** (Informal). *There exists an oracle relative to which 1-key PCPRF exists while any KA protocol can be broken. This implies that it is impossible to construct KA using PCPRF in a black-box manner.*

Here, it is worth mentioning the predicate family of the PCPRF relative to our oracle. We use the setting of oracle-aided circuits proposed in [1] in which the circuits are only allowed to access a random oracle H but not the rest of the oracle. As stated in [1], this setting characterizes most of the non-black-boxes constructions such as punctured programming approach of Sahai and Waters [26] and its variants. This is further discussed in Section 2.4.

Our proof framework follows a similar approach to [20]. However, proving our result based on the techniques in [20] is not straightforward. The main challenge is explained in Section 2, where we introduce a new technique to overcome this issue.

2 Technical Overview

In this section, we first review the techniques from [20]. Then, we identify the main reason why the proof in [20] does not directly apply to our problem. Finally, we briefly outline our solution to this obstacle.

2.1 Techniques of [20]

In [20], it is proved that there exists an oracle relative to which a one-way function exists, but KA does not. The oracle consists of two parts: a random function f and a PSPACE-complete oracle. Clearly, f itself is one-way even when the PSPACE-complete oracle is present. Now we review how the attacker Eve in [20] breaks the KA protocol under such an oracle. Suppose Alice and Bob are the two parties of a KA protocol; Eve tries to find all the intersection queries – queries made by both Alice and Bob. It is shown in [20] that finding all the intersection queries suffices to break the KA protocol. During the attack, Eve maintains a list of query-answer pairs, recording the queries and oracle answers she has received so far. Eve's algorithm repeats the following two phases (which is called one "segment") polynomially many times after each message is sent.

[1] In [12] authors construct indistinguishability obfuscators for arbitrary circuits from 2-key secure PCPRF. Since we can construct public key encryption from iO and OWF [26], we already obtain a **non-black-box** construction of primitives in Cryptomania from 2-key secure PCPRF.

- **Simulation Phase**: Eve uniformly samples a simulated view of Alice that is consistent with the transcript and Eve's list (if Bob has just sent a message, she samples Bob's run instead) using the PSPACE-complete oracle. Any simulated query not on Eve's list receives a random answer. In general, such a random answer may differ from the actual one.
- **Update Phase**: For each query in the simulated run, Eve queries the actual oracle f and adds the query-answer pair to her list.

Suppose that Alice has just sent a message. We temporarily fix the view of Bob, the transcript and Eve's list, while only Alice's view is considered unknown. Let BPQ denote the set of "Bob's private queries", which are the queries made by Bob but not Eve. Clearly, the size of BPQ is bounded by the total number of queries made by Bob. We classify each simulated view of Alice into the following two types:

1. **Hit**: this simulated view contains at least 1 query in BPQ.
2. **Miss**: this simulated view does not contain any queries in BPQ.

By the above definition, each hit contains at least one query in BPQ. Then in the following update phase, these queries will be made by Eve. This means the size of BPQ decreases by at least 1 in this segment. Since the size of BPQ is bounded, the number of hits is also bounded. Thus, as long as Eve repeats the simulation sufficiently many times, most of the simulations will be misses.

The key to the proof in [20] is that, conditioned on a miss, the probability distribution of Alice's view in the simulation matches the distribution of her actual view. In other words, Eve's simulation of Alice's view must be faithful to the real one, given a "miss". Since the transcript and Bob's view is fixed, the query that will be made by Bob in the next round is also fixed. If Bob, in the next round, makes an intersection query with high probability, a significant portion of Alice's simulated view would already include this query. As a result, Eve has a high probability of identifying the intersection query. Therefore, even before a query becomes an intersection query, it is highly likely that Eve predicts it and adds it to her list.

2.2 Limitations of [20]

Now we consider how to apply the technique from [20] to our problem of separating 1-key secure PCPRF and KA. For now, we only consider predicates that are not allowed to access the oracle. In other words, they are just binary circuits in the common sense. As it turns out, this is already non-trivial.

The first step is to construct an oracle relative to which PCPRF exists. A natural approach is to define the oracle in the shape of a PCPRF. Let K, K', C be the spaces of master keys, constrained keys and predicates, respectively, and let X, Y be the spaces of inputs and outputs of the PRFs. We then define the oracle Γ to consist of the following components:

- $\mathsf{F} : K \times X \to Y$ serves as the PRFs for master keys;
- $\mathsf{f} : K' \times X \to Y$ serves as the PRFs for constrained keys;
- $\mathsf{Keygen} : K \times C \to K'$ serves as the constrained key generation algorithm (henceforth we denote $k[c] = \mathsf{Keygen}(k, c)$).
- PSPACE-complete oracle, as usual.

The above oracle construction actually implements a multiple-key secure PCPRF. Since 2-key secure PCPRF and OWF already implies KA, we cannot hope to prove that KA does not exist under such an oracle. In order to capture one-key security, a natural idea is to add

another weakening oracle R with the following functionality: given two different constrained keys $k[c_1]$ and $k[c_2]$ of the same master key k, $\mathsf{R}(k[c_1], k[c_2])$ will return k, c_1, c_2. In any other cases, R returns $\perp$. For the R oracle to be well-defined, every element $k' \in K'$ must correspond to **at most one** pair $(k, c) \in K \times C$ satisfying the relation $k' = k[c]$. In other words, the oracle Keygen should be injective.

The correctness of PCPRF states that $\mathsf{F}(k, x) = \mathsf{f}(k[c], x)$ for $c(x) = 1$. This correctness property naturally divides all possible F, f queries into many equivalent classes, and the answer for each equivalent class is sampled uniformly and independently from each other.

Now, we can see the first limitation of [20]. The major idea behind [20] is that: the only way that two parties achieve key agreement in the presence of Eve is essentially by making the same queries to the oracle. However, when we are considering a more sophisticated oracles like Γ instead of a simple random oracle, Alice and Bob can reach an agreement even if they do not make the same queries at all. Consider the following protocol:

- Alice samples $k \in K, c \in C, x \in X$ such that $c(x) = 1$. Then Alice queries $k' = k[c] = \mathsf{Keygen}(k, c)$ and sends k', x to Bob. The output of Alice is $\mathsf{F}(k, x)$.
- Bob receives k', x from Alice and outputs $\mathsf{f}(k', x)$.

The correctness of this key agreement protocol is guaranteed by the correctness of PCPRF: $\mathsf{F}(k, x) = \mathsf{f}(k[c], x)$. However, they make completely different queries.

The above issue is common especially when the internal dependencies exists among the answers of different oracle queries. A common solution is to consider the input elements separately, such as [23, 1]. Specifically, instead of intersection queries, we consider "intersection elements". We say an element in $K \cup K' \cup C \cup X$ appears in a party's view if this element is a part of the input or output of a query in this party's view. (Here, we do not consider elements in Y because they cannot appear as the input of a query.) Analogous to intersection queries, intersection elements are elements that appears in both Alice and Bob's views. Our attacker Eve's goal is to find all the intersection elements. In the above example, Alice's view contains k, c, x, k'; Bob's view contains k', x. Thus, k', x are both intersection elements. If Eve finds k', x, she can make the query $\mathsf{f}(k', x)$ and the answer would be the agreed key.

However, the above problem is not the major challenge we faced trying to apply the technique of [20] to our case. The main reason is that the distribution of Alice's view will depend on Bob's view, even conditioned on "miss". Since Bob's view is invisible to Eve, such dependency cannot be captured by Eve, which causes Eve's simulation to be unfaithful to the real Alice's view. In the following, we use 2 simple examples to demonstrate why the distribution of Alice's simulated view can be unfaithful to Alice's actual view even conditioned on "miss".

- **Example 1**: Consider the following 2-pass protocol. In the first pass, Bob randomly samples $k_B \in K$ and queries $\mathsf{F}(k_B, i) = y_B^i$ for $i = 1, 2, \ldots, n$ where n is a sufficiently large polynomial (such that $|K'|^{-1} > |Y|^{-n}$). No communication between Alice and Bob is made in this round. In the next pass, Alice randomly samples $k'_A \in K'$ and queries $\mathsf{f}(k'_A, i) = y_A^i$ for $i = 1, 2, \ldots, n$. Now we fix Bob's view, i.e. k_B and $y_B^i, i \in [n]$. Let E be the event that $y_A^i = y_B^i$ for all $i \in [n]$. In the real execution, as long as $k'_A = k_B[c]$ for some $c \in C$ such that $c(x) = 1$ for $x \in [n]$, we would have $y_A^i = y_B^i$ for all $i \in [n]$. The probability of Alice picking $k' = k[c]$ for such a c is at least $|K'|^{-1}$. Thus, the probability of E happening in the real execution is at least $|K'|^{-1}$. However, since Eve does not have Bob's view, when simulating Alice's view, each y_A^i are randomly sampled from Y. the probability that $y_A^i = y_B^i$ for all $i \in [n]$ in Eve's simulation is $|Y|^{-n}$. Thus, as long as n is sufficiently large, we would notice that the event E is more likely to happen in the real execution than in Eve's simulation, which means that Eve's simulation is unfaithful to Alice's actual view.

- **Example 2**: Consider the following 2-pass protocol. In the first pass, Bob randomly samples $k_B \in K, c_B \in C$ and queries $\mathsf{Keygen}(k_B, c_B) = k'_B$. In the second pass, Alice randomly samples $k_A \in K$ and $c_A \in C$ and queries $\mathsf{Keygen}(k_A, c_A) = k'_A$. Fix the view of Bob. Let E be the event that $k_A \neq k_B$ and $k'_A = k'_B$. Because Keygen is sampled from injective functions, the probability of E is 0 in the real execution. However, in the distribution of Eve's simulated view of Alice, event E has a non-zero probability if Eve does not know the relation $k'_B = k_B[c_B]$. This means that Eve's simulation is unfaithful to Alice's actual view.

Examples 1 and 2 show that the distribution of Alice's actual view and the distribution of Eve's simulated Alice's view might have very different probabilities on some specific "miss" views. In other words, the distribution of Eve's simulation is unfaithful to Alice's actual view even conditioned on "miss". Even with the weakening oracle R, these two examples can still happen. It should be noted here that these two examples are merely illustrations of the unfaithfulness issue. When we are considering arbitrary protocols, the situation might be more complicated.

To overcome this issue of unfaithfulness, we will modify the proof of [20]. In the proof of [20], the most important step is to prove that Eve's simulation is faithful to Alice's actual view conditioned on "miss", for any **fixed** tuple of Bob's view, Eve's view and the transcript so far. We denote such tuple as G. In our proof, we will analyze the statistical distance between Eve's simulation and Alice's actual view, conditioned on "miss". We show that the expectation of this statistical distance is negligible over a specific distribution of G, which will be enough under the framework of [20]. To do so, we introduce a new technique which we call the **weighting** technique. This is further discussed in the next section.

2.3 Our Approach

In this section we provide a high level overview of our approach. We first give a simplified version of our oracle.

2.3.1 Oracle description

Let K, K', C, X, Y be the space of master keys, constrained keys, predicates, the input and the output of the PCPRF, respectively. The size of these spaces are set as follows: $|K| = |C| = |X| = 2^\kappa, |K'| = 2^{\kappa^2}, |Y| = 2^{\kappa^3}$. Our oracle contains two layers. The first layer contains a random oracle $\mathsf{H} : Z = \{0,1\}^* \to \{0,1\}$ and a PSPACE-complete oracle. The elements in C are viewed as P.P.T.Ms that have access to the first layer. The second layer consists of the following components:

- $\mathsf{F} : K \times X \to Y$ is uniformly sampled from all functions.
- $\mathsf{Keygen} : K \times C \to K'$ is sampled from all injective functions. Henceforth we denote $k[c] = \mathsf{Keygen}(k, c)$.
- $\mathsf{f} : K' \times X \to Y$ is defined as follows: if $k' = k[c]$ and $c(x) = 1$, then $\mathsf{f}(k', x) = \mathsf{F}(k, x)$. Otherwise it is randomly sampled from Y.
- $\mathsf{R} : (K \cup K') \times K' \to C \cup (K \times C \times C) \cup \{\perp\}$ is defined as follows: if the input is $(k, k[c])$ for some $k \in K, c \in C$, then R returns c; if the input is $(k[c_1], k[c_2])$ for some $k \in K, c_1, c_2 \in C$ and $c_1 \neq c_2$, R returns (k, c_1, c_2); in any other cases, R returns $\perp$ as a sign of rejection.

We denote the whole oracle as $\Gamma = (\mathsf{H}, \mathsf{F}, \mathsf{Keygen}, \mathsf{f}, \mathsf{R})$. Here, we remark that, in our formal definition, the oracle is defined in a more complicated way: the second layer of Γ is sampled for each input length κ. For simplicity, we only consider a fixed input length in this overview.

2.3.2 The Weighting Technique

Before proceeding to our attacking algorithm, we first introduce the weighting technique that is used to address the unfaithfulness issue raised in the previous section. We start by redefining the concept of views by introducing artificial views that might contain collisions of Keygen. Generally speaking, a view is what can be seen by a party (or parties). It includes the randomness of the party, the transcript and the query answer pairs made by the party. Recall that the Keygen oracle is sampled from all injective functions, collisions will never happen in a view. However, we allow a view to contain collisions such as $k_1[c_1] = k_2[c_2] = k'$ where $k_1 \neq k_2$. (Here, by collisions, we do not specifically mean the collision of Keygen queries. R queries can also be a part of collisions. For example, the two query-answer pairs such as $\mathsf{Keygen}(k_1, c_1) = k', \mathsf{R}(k_2, k') = c_2$ also constitute a collision.) That being said, we do not wish a view to contain arbitrary collisions. We say a view V is "valid" if it satisfies the following 2 properties:

1. If two master keys k_1 and k_2 collide at k', that is $k_1[c_1] = k_2[c_2] = k'$ for some c_1, c_2, then k_1 and k_2 will never be a part of another collision in V.
2. There are no self-collisions, i.e. $k[c_1] = k[c_2]$ for $c_1 \neq c_2$.

To better understand the first property, we here give two counterexamples which violate it: (1), one master key simultaneously collides with two other master keys: $k_1[c_1] = k_2[c_2], k_1[c_1'] = k_3[c_3]$; (2), two master keys collide at two different constrained keys: $k_1[c_1] = k_2[c_2], k_1[c_1'] = k_2[c_2']$.

We then define the weight of a valid view. Given a valid view V and a specific instantiation H of the oracle H, define $w_H(V)$ as follows:

- If the H queries made in V are inconsistent with H, $w_H(V) = 0$.
- If the correctness of PCPRF (with H instantiated as H) is violated in V, $w_H(V) = 0$.
- Otherwise, define $w_H(V)$ as follows: each different constrained relation $k[c] = k'$ in V contributes a factor of $|K'|^{-1} = 2^{-\kappa^2}$; each different F or f queries contribute a factor of $|Y|^{-1} = 2^{-\kappa^3}$ However, those F or f queries that can be deduced to be the same from the correctness of PCPRF only count as one contribution. $w_H(V)$ is defined to be the product of all contributions.

Then, the weight of a view V is defined as $w(V) = \frac{1}{2^M} \sum_H w_H(V)$. Here, the summation is taken over all possible instantiations H. M is the size of the possible input space of H. Indeed, the input of H can be arbitrarily long. However, since all parties are P.P.T.Ms, they cannot make H queries with arbitrarily long inputs. Therefore we can view the input space of H as finite. 2^M is the number of possible instantiations of H, which means that $w(V)$ can be viewed as the average value of $w_H(V)$.

We now define two distributions $\mathcal{D}$ and $\mathcal{D}'$ over all valid views. Define $\mathcal{D}$ as the distribution of views generated by first randomly sampling the randomness of all three parties (Alice, Bob and Eve) and oracle Γ, then executing the protocol. $\mathcal{D}'$ is a distribution over all valid views. The probability of a valid view V in $\mathcal{D}'$ is proportional to $w(V)$. Then, we will prove that the statistical distance between $\mathcal{D}$ and $\mathcal{D}'$ is negligible, which is a key lemma in our analysis.

Our Attacking Algorithm

Now we consider how to break KA protocols relative to this oracle. Our attacker Eve is similar to [20]. She locally maintains a list of elements in $K \cup K' \cup C \cup X \cup Z$ and repeats the two phases (i.e. a segment) mentioned in Section 2.1 after each message is sent. Suppose Alice speaks in round $r-1$ and Bob speaks in round r. Then Eve's algorithm in round r will repeat the following two phases sufficiently many times:

- **Simulation Phase**: Eve samples a view of Alice up until the end of round $r-1$, denoted as Sim_A. Sim_A should be consistent with Eve's view and the transcript so far. Furthermore, Sim_A combined with Eve's view V_E (denoted as $\mathsf{Sim}_A \cup V_E$) should form a collision-free valid view. The probability that Sim_A is sampled is proportional to $w(\mathsf{Sim}_A \cup V_E)$.
- **Update Phase:** Eve will add all the elements in the sampled view Sim_A to her list. Then, Eve will try to find out all the information related to the elements in her list with the help of R.

We now briefly explain why our attacking algorithm addresses the unfaithfulness issue raised in Section 2.2. We will show that, for any fixed Bob and Eve's view V_B, V_E, if Sim_A is a "miss" simulation, that is, the intersection elements between Sim_A and V_B all appear in V_E, then the combined view $\mathsf{Sim}_A \cup V_E \cup V_B$ is a valid view which might contain collisions as discussed before. This is exactly why we allow a valid view to contain collisions. Furthermore, we will show that $w(\mathsf{Sim}_A \cup V_E)$ is proportional to $w(\mathsf{Sim}_A \cup V_E \cup V_B)$. Recall that $w(\mathsf{Sim}_A \cup V_E)$ is proportional to the probability that Sim_A is sampled by Eve. By the definition of $\mathcal{D}'$, $w(\mathsf{Sim}_A \cup V_E \cup V_B)$ is proportional to the probability of Alice's view being Sim_A in $\mathcal{D}'$ conditioned on V_B , V_E and the transcript so far. This means, Eve's simulation is actually faithful to Alice's view in $\mathcal{D}'$, conditioned on "miss". After that, we use the result $\mathcal{D} \approx \mathcal{D}'$ to show that Eve's simulation is also faithful to Alice's view in $\mathcal{D}$ with high probability. Then, we can apply the technique of [20] to our problem.

2.4 More on Related Work

Perfect Correctness vs. Imperfect Correctness

In this study, we examine key agreement (KA) protocols where Alice and Bob are allowed to output different keys, a characteristic referred to as **imperfect correctness**. In contrast, KA protocols with **perfect correctness** requires Alice and Bob to always output the same key. When trying to obtain black-box separation results between KA and other primitives, the proof is usually simpler if we restrict the KA protocols to be perfectly correct. For example, when considering the black box separation between OWFs and perfectly correct KA, there exists a proof [9, Section 4.1] much simpler than [20, 4]. There also exist impossibility results only for perfectly correct key-agreement, such as [1, 14].

However, in our case, the proof cannot be greatly simplified if we consider perfectly correct KA protocols. The reason is as follows: under our oracle, Eve's simulated view might not be possible in the real world, as shown in Example 2 in Section 2.2. In these impossible views, Alice and Bob might output different keys. Thus, perfect correctness does not help.

Black-Box Separations Involving Injective Primitives

The key-generation oracle Keygen is defined to be a random injective function. This raises the issue stated in Example 2 discussed in Section 2.2. Here, we acknowledge that similar challenges have appeared when dealing with injective primitives (e.g., injective OWFs [24], TDFs [17]), albeit with different technical details. However, their techniques are not sufficient for solving the problems we have met, because their problems are quite different from the problem we face.

Oracle-Aided Circuits

The predicate class in our work is defined as P.P.T.M.s with access to a lower level random oracle H and the PSPACE-complete oracle. We acknowledge that this is exactly the same as the oracle aided circuits proposed by [1]. This setting rules out constructions of KA in which the predicates might use the lower level primitives (such as OWF) in a non-black box way. We clarify that even in this setting, a 2-key secure PCPRF suffices to imply iO, and sequentially PKE via the punctured programming approach of Sahai and Waters [26]. However, this does not rule out constructions in which the code of the predicates might use a part of PCPRF itself in a non-black box way.

2.5 Paper Organization

The rest of our paper is organized as follows. In Section 3, we give a formal definition of PCPRF. In Section 4, we give a formal definition of our oracle Γ and prove that 1-key secure PCPRF exists under such an oracle Γ. In Section 5, we provide formal definitions of secret-key agreement protocols and views, then define the distribution $\mathcal{D}$. Section 6 is the foundation of our technique. In this section, we give the formal definition of valid views, the weights of valid views, and the distribution $\mathcal{D}'$. Then, we prove our key lemma $\mathcal{D} \approx \mathcal{D}'$. In Section 7, we present the algorithm of Eve that breaks any KA protocol, and our main black-box separation result.

3 Preliminaries

A non-negative function $\mu : \mathbb{N}^* \to \mathbb{R}^+ \cup \{0\}$ is called **negligible** if $\mu(n) = O\left(\frac{1}{n^c}\right)$ for any constant $c \in \mathbb{N}^*$, denoted by $\mu = \mathrm{negl}(n)$.

In this work, we use a notion stronger than negligible: a non-negative function $\mu : \mathbb{N}^* \to \mathbb{R}^+ \cup \{0\}$ is **super-negligible** if $\mu(n) = O\left(\exp\left(-n^{1+\epsilon}\right)\right)$ for some constant $\epsilon > 0$, denoted by $\mu = \mathrm{snegl}(n)$. In the proof, we will use the following fact: $\mathrm{snegl}(\log n) = \mathrm{negl}(n)$.

We define two notations for simplifying our proof: $a(n) \approx b(n)$ means that $\left|\frac{a(n)}{b(n)} - 1\right| = \mathrm{negl}(n)$, and notation $a(n) \approx_s b(n)$ means that $\left|\frac{a(n)}{b(n)} - 1\right| = \mathrm{snegl}(n)$.

Given a probability distribution $\mathcal{S}$ over the set S, denote $\sup(\mathcal{S})$ as the support set of $\mathcal{S}$. That is, $\sup(\mathcal{S})$ is the set of elements in S that have non-zero probability in $\mathcal{S}$.

3.1 Statistical Distance

▶ **Definition 2.** *Given two (discrete) probability distributions $\mathcal{S}_1$ and $\mathcal{S}_2$ over the set S, the statistical distance between them is defined as*

$$\Delta(\mathcal{S}_1, \mathcal{S}_2) = \frac{1}{2} \sum_{s \in S} \left| \Pr_{s' \leftarrow \mathcal{S}_1}[s' = s] - \Pr_{s' \leftarrow \mathcal{S}_2}[s' = s] \right|.$$

The statistical distance satisfies triangular inequality, i.e., for any distributions $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ over S, we have $\Delta(\mathcal{S}_1, \mathcal{S}_3) \leq \Delta(\mathcal{S}_1, \mathcal{S}_2) + \Delta(\mathcal{S}_2, \mathcal{S}_3)$. Another important fact is that statistical distance does not increase when applying a (possibly randomized) function f, i.e. $\Delta(\mathcal{S}_1, \mathcal{S}_2) \geq \Delta(f(\mathcal{S}_1), f(\mathcal{S}_2))$.

3.2 Private Constrained PRF

In this work, we use a simulation-based definition of 1-key oracle-aided private constrained PRF (PCPRF), our notion is similar to [12] except that both adversary and simulator have access to an oracle $\mathcal{O}$. Since the definition of 1-key security of PCPRF is straightforward to grasp, but a rigorous formulation would be laborious, we therefore omit it here and refer the interested readers to the full version.

4 PCPRF Relative to Oracle Γ

4.1 Oracle Design

Our oracle consists of the following parts:

- A PSPACE-complete oracle.
- $\mathsf{H} : Z = \{0,1\}^* \to \{0,1\}$ is a random oracle, i.e., for arbitrary binary string z, $\mathsf{H}(z)$ is sampled from uniform distribution over $\{0,1\}$.
- For every $\kappa \in \mathbb{N}^*$, we denote PCPRF oracle of length κ as Γ_κ defined below:
  - $K_\kappa = X_\kappa = C_\kappa = \{0,1\}^\kappa, K'_\kappa = \{0,1\}^{\kappa^2}, Y_\kappa = \{0,1\}^{\kappa^3}$. Here elements of C_κ are viewed as arbitrary $\mathsf{P/poly}$ predicates with input length κ. They are allowed to query H and the PSPACE-complete oracle. However, they are only allowed to query H for at most κ times.
  - $\Gamma_\kappa = (\mathsf{F}_\kappa, \mathsf{f}_\kappa, \mathsf{Keygen}_\kappa, \mathsf{R}_\kappa)$ are four oracles sampled by the following way:
    1. $\mathsf{F}_\kappa : K_\kappa \times X_\kappa \to Y_\kappa$ is uniformly sampled from all functions.
    2. $\mathsf{Keygen}_\kappa : K_\kappa \times C_\kappa \to K'_\kappa$ is sampled from all **injective** functions. Denote $k[c] = \mathsf{Keygen}(k, c)$.
    3. $\mathsf{f}_\kappa : K'_\kappa \times X_\kappa \to Y_\kappa$ is defined as follows: given $k' \in K'_\kappa, x \in X_\kappa$, if there exists $k \in K_\kappa, c \in C_\kappa$ s.t. $k' = k[c]$ and $c(x) = 1$, then $\mathsf{f}_\kappa(k', x) = \mathsf{F}_\kappa(k, x)$. The other locations of f_κ are sampled uniformly from Y_κ.
    4. $\mathsf{R}_\kappa : (K_\kappa \cup K'_\kappa) \times K'_\kappa \to C_\kappa \cup (K_\kappa \times C_\kappa \times C_\kappa) \cup \{\bot\}$ is defined as follows: if the input is (k, k') where $k \in K'_\kappa$ and $k' = k[c]$, then $\mathsf{R}_\kappa(k, k') = c$. If the input is (k'_1, k'_2) where $k'_1, k'_2 \in K'_\kappa$ s.t. there exists $k \in K_\kappa$, $c_1, c_2 \in C_\kappa, c_1 \neq c_2$, $k'_1 = k[c_1], k'_2 = k[c_2]$, then $\mathsf{R}_\kappa(k'_1, k'_2) = (k, c_1, c_2)$. In any other cases, R_κ will return $\bot$ as a sign of rejection.
- We define oracle Γ to be the concatenation of Γ_κ for every length, H and the PSPACE-complete oracle, i.e., $\Gamma = (\text{PSPACE-complete oracle}, \mathsf{H}, \{\Gamma_\kappa\}_{\kappa \in \mathbb{N}^*})$, also define K, X, C, K', Y to be the concatenation of all corresponding fields.

Here we remark that the elements of the sets we defined before are different elements, even though they might have the same binary representation. For example, suppose $z \in Z$ is of length κ, then z can only be used as input of H queries. We cannot use z as the input of a Γ_κ query. Similarly, an element $k \in K_\kappa$ can only be used as the first input of F_κ or R_κ queries; an element $x \in X_\kappa$ can only be used as the second input of F_κ or f_κ queries, etc.

In the following, we will often omit the index κ of the oracles in Γ_κ. For example, we might simply write $\mathsf{F}(k, x)$ instead of $\mathsf{F}_\kappa(k, x)$. This does not introduce ambiguity because κ is fully determined by the length of the input.

4.2 Construction of PCPRF

Now we propose our 1-key secure PCPRF construction:

- $\mathbf{Gen}(1^\kappa)$: randomly sample a master key $\mathsf{MSK} \xleftarrow{R} K_\kappa$.
- $\mathbf{Constrain}(\mathsf{MSK}, c)$: answer $\mathsf{Keygen}(\mathsf{MSK}, c)$.
- $\mathbf{Eval}(k, x)$: if k is a master key, i.e., $k \in K_\kappa$, return $\mathsf{F}(k, x)$; otherwise k is a constrain key, i.e., $k \in K'_\kappa$, return $\mathsf{f}(k, x)$.

The perfect correctness of our scheme follows directly the functionality of oracle Γ: whenever $c(x) = 1$, we have

$$\mathbf{Eval}(\mathsf{MSK}, x) = \mathsf{F}(\mathsf{MSK}, x) = \mathsf{f}(\mathsf{Keygen}(\mathsf{MSK}, c), x) = \mathbf{Eval}(\mathsf{CK}, x).$$

Since the oracle Γ os defined in the shape of PCPRFs, the proof of security naturally follows from the definition. Therefore we omit it here and refer the interested readers to the full version. Note that the recovery oracle R does not help in breaking the 1-key privacy because R takes two different constrained keys to generate an output that is not $\perp$.

5 Additional Preliminaries

5.1 Secret-Key Agreement Protocol

5.1.1 Definition

A secret-key agreement protocol consists of two (oracle-aided) P.P.T.M. called Alice and Bob. In addition, they have a common communication tape that both parties can read and write. A run of the protocol is as follows: Alice and Bob both start with input 1^λ, where λ is the security parameter; Alice and Bob run, communicating via the common tape; at last, Alice and Bob both output a string of length l, where $l = l(\lambda)$ is a polynomial over λ. If these two strings are the same, Alice and Bob are said to agree. The entire history of the writes to the communication tape is called the transcript. $\alpha(\lambda)$ will denote the probability that Alice and Bob agree on the same secret with security parameter λ.

We say (oracle-aided) P.P.T.M. Eve breaks a secret-key agreement protocol, if Eve, given only the transcript, can guess the secret with probability $1/\mathrm{poly}(\lambda)$ conditioned on Alice and Bob agree on the same secret. A protocol is secure if no P.P.T.M Eve can break it.

5.1.2 Normal Form

In this section, we will convert arbitrary KA protocol into a protocol satisfying certain conditions, which we call the "normal form". There are many steps in this conversion. In each step, the protocol gains a certain restriction that will facilitate our analysis. The conversion does not compromise the security of the KA protocol and only causes polynomial blow-up in running time.

Restriction 1: Avoid querying Γ_κ with small κ

Let λ be the security parameter of the key agreement protocol. The first step in our conversion is to eliminate any Γ_κ queries with $\kappa \leq \log \lambda$.

We first assume that all three parties have a preparation phase at the beginning of the protocol. In this phase, they make sufficient queries to H so that for every pair of (c, x) where $c \in C_\kappa, x \in X_\kappa, \kappa \leq \log \lambda$, the queries made in the preparation phase is enough to evaluate $c(x)$. For each $\kappa \leq \log \lambda$, $|C_\kappa| = |X_\kappa| = 2^\kappa \leq \lambda$. This means that there are only polynomially many such pairs. For each pair (c, x), to evaluate $c(x)$, at most $\log \lambda$ queries to H need to be made. Thus each party only need to make polynomially many queries to H in the preparation phase. Clearly, adding the preparation phase will not compromise the security of the KA protocol.

Then, we assume that each party also recovers the whole Keygen_κ oracle in the preparation phase. Since the input of Keygen_κ is in the set $|K_\kappa| \times |C_\kappa|$ which has size $2^{2\kappa} \leq \lambda^2$, this step only takes polynomial time. Once Keygen_κ is known to every party, there is no need to query

R_κ in Γ_κ, and all F_κ and f_κ queries are naturally divided into many equivalent classes by the correctness of PCPRF. The queries in each equivalent class have the same answer, and the answer for each equivalent class is individually and independently sampled from Y_κ. We can simulate the F_κ and f_κ oracle by a part of H that is never used in the original protocol. Thus, we may assume that F_κ and f_κ are never queried. As a result, only Keygen_κ is queried in Γ_κ.

Now, instead of letting each party recover Keygen_κ, we let Alice generate a "fake" Keygen_κ oracle and send it to Bob. Whenever a party needs to query Keygen_κ, they will instead use the fake Keygen_κ. As a result, each party will never access Γ_κ.

Restriction 2: Maintain the "completeness" with F, f, Keygen Queries

We first formally define what is the meaning of the "completeness" of a view:

▶ **Definition 3.** *We say a view is* ***complete*** *if for every κ and every pair of $c \in C_\kappa, x \in X_\kappa$ that appears in this view, the H queries made in this view are enough to evaluate $c(x)$.*

We are now ready to describe restriction 2 of the normal form. This restriction ensures that, if a party's view is complete before a $\mathsf{F}, \mathsf{f}, \mathsf{Keygen}$ query, then the view is also complete after this query. Specifically, suppose the next query of a party is $\mathsf{F}(k_0, x_0)$, $\mathsf{f}(k_0', x_0)$ or $\mathsf{Keygen}(k_0, c_0)$. Before making this query, this party will make sure that

- If the next query is $\mathsf{F}_\kappa(k_0, x_0)$ or $\mathsf{f}_\kappa(k_0', x_0)$, for every $c \in C_\kappa$ that has appeared in this party's view, enough H queries must be made in this view to evaluate $c(x_0)$.
- If the next query is $\mathsf{Keygen}_\kappa(k_0, c_0)$, for every $x \in X_\kappa$ that has appeared in this party's view, enough H queries must be made in this view to evaluate $c_0(x)$.

Restriction 3: Repair the completeness after R queries

Suppose a party just made an R_κ query and the answer is not $\perp$. Then, the answer must contain 1 or 2 predicates. Denote them as c_1 (and c_2). This party will first make sure that, for every $x \in X_\kappa$ that has appeared in this view, enough H queries are made to evaluate $c_1(x)$ (and $c_2(x)$).

Note that restriction 2 of the normal form ensures that $\mathsf{F}, \mathsf{f}, \mathsf{Keygen}$ queries cannot break the completeness of a party's view. Clearly, H queries cannot break the completeness of a party's view. Thus, only R queries can break the completeness of a party's view. Indeed, if an R query yields an answer other than $\perp$, then the answer might include 1 or 2 predicates. Denote them as c_1 (and c_2). If c_1 (and c_2) did not appear in this party's view, the completeness of a party's view might be temporarily broken. Restriction 3 ensures that the completeness of this party's view is fixed (after a number of H queries.)

Restriction 4: Find hidden constrained relations among the keys

Suppose the next query of a party is $\mathsf{F}(k_0, x_0)$, $\mathsf{f}(k_0', x_0)$ or $\mathsf{Keygen}(k_0, c_0)$, this party will first make sure that there are no hidden constrained relations among the master keys and constrained keys that have appeared in this view (along with k_0 or k_0'). Specifically, for each $k \in K_\kappa$ and $k' \in K_\kappa'$ that have appeared in this view (along with k_0 or k_0'), it must be clear in this view whether k' is a constrained key generated from k; for each $k_1', k_2' \in K_\kappa'$ appeared in this view (along with k_0'), it must be clear in this view whether they are the constrained keys generated from the same master key k. All of this can be achieved by the recovery oracle R.

The Normal Form

Let $\kappa_{\max}(\lambda)$ be the maximum κ such that Alice or Bob might query Γ_κ. Clearly, $\kappa_{\max}$ is also a polynomial over λ. We assume that the protocol has $n = n(\lambda)$ rounds where $n(\lambda)$ is a polynomial over λ. Alice speaks in odd rounds and Bob speaks in even rounds. Suppose r is even, then in round r:

- Bob first does his own computation. Then he makes an arbitrary query. This query is later referred as the first query of round r. This query can be an H query or Γ_κ query for some $\kappa \geq \log \lambda$. If this query is not an H query, then the input contains two elements. We make the restriction that at least one of the two input elements has already appeared in Bob's view. In other words, only one can be new between the two input elements. Denote the new input element as $q_{r,1}$. (If neither input elements are new or the first query is an H query, let $q_{r,1}$ be the first input element.)
- After the first query, Bob will make $2n\kappa_{\max}$ H queries. Let $q_{r,j}, (2 \leq j \leq 2n\kappa_{\max} + 1)$ be the input of the jth query. Especially, if the first query of round r is an R query and the answer is not $\perp$, then the following $2n\kappa_{\max}$ H queries must be first used to make sure that the completeness of the view still holds as requested by restriction 3. ($2n\kappa_{\max}$ H queries are enough to do this: R query introduces at most 2 new predicates; there are at most n elements in X in Bob's view because at most 1 new elements in X appears in Bob's view in each round; to evaluate $c(x)$ where both c, x appeared in this view, it takes at most $\kappa_{\max}$ H queries. Thus $2n\kappa_{\max}$ H queries are enough.)
- Bob then sends one message Conv^r to Alice. Denote $M^r = (\mathsf{Conv}^1, \mathsf{Conv}^2, \ldots, \mathsf{Conv}^r)$ be the messages sent so far.

For odd r, we only need to change the role of Alice and Bob.

Moreover, before Alice and Bob output their secrets s_A, s_B, we add 2 rounds at the last of the protocol. In these two rounds, the first query of Alice and Bob is $\mathsf{H}(s_A)$ and $\mathsf{H}(s_B)$. These two rounds ensure that, if Alice and Bob agree on the same secret s, both of them would have made the query $\mathsf{H}(s)$.

As we have discussed before, $\mathsf{H}, \mathsf{Keygen}, \mathsf{F}, \mathsf{f}$ queries will not break the completeness of a party's view. Only R queries might temporarily break the completeness. If the completeness of a party's view is broken by an R query, the following $2n\kappa_{\max}$ H queries make sure that the completeness is fixed. Therefore, each party's view is complete at the end of each round.

5.2 Views

Denote V to be the *view* of all three parties at the end of the secret-key agreement protocol. It consists of all three parties' randomness r_A, r_B, r_E, all messages M^n, as well as all the queries and corresponding answers made so far by all three parties to Γ. For now, the oracle behavior in a view can be arbitrary, as long as the same query yields the same answer. We will define what is a valid view in the next section.

Given a view V, if we only consider a part of it, we get a partial view. Let r be a round number, denote $V_A^{r,j}$ as the partial view of Alice in V up to and include the jth query of round r. This does not include the message sent by Alice in round r. Denote V_A^r as the partial view of Alice in V up to the end of round r which includes the message sent by Alice at round r. Same for $V_B^{r,j}$ and V_B^r. The notation is slightly different for Eve, $V_E^{r,i}$ denotes the partial view of Eve in V up to and include segment i of round r. Here, we only need to know that Eve's algorithm in round r includes many segments. In each segment, Eve might make multiple queries. Especially, $V_E^{r,0}$ denotes the view of Eve in V right before Eve's algorithm starts in round r.

Given a (possibly partial) view V, let $P(V)$ be the set of oracle query-answer pairs that appear in V. We also denote $Q(V)$ as the elements of $Z \cup X \cup C \cup K \cup K'$ that appears in this view. Note that we have excluded the elements in Y, because they can only appear as an output of a query.

Conversely, given several partial views, if their common parts are the same, we can combine them together to form another (possibly partial) view. We slightly abuse the notion $\cup$ to denote the combination of partial views. For example, let U, V are two views and M^r are the same in these two views, then we denote $U_A^r \cup V_B^r \cup V_E^{r,i}$ to be the view obtained by replacing the partial view of Alice in V^r to U_A^r.

In this work, we focus on a probability distribution over views:

▶ **Definition 4.** *Define $\mathcal{D}$ as the distribution of views generated by first randomly sampling all parties' randomness and oracle Γ defined in Section 4, then execute the protocol.*

6 Valid View and Indistinguishability Lemma $\mathcal{D} \approx \mathcal{D}'$

Recall that a "view" contains the randomness, messages, and query-answer pairs to Γ made by all parties. A view may include multiple master keys $k \in K$ and constrained keys $k' \in K'$ and many constrained relations $k[c] = k'$. A Keygen query $\mathsf{Keygen}(k, c) = k'$ is equivalent to one such relation: $k[c] = k'$. An R query $\mathsf{R}(k, k') = c$ is equivalent to one such relation $k[c] = k'$. An R query $\mathsf{R}(k'_1, k'_2) = (k, c_1, c_2)$ is equivalent to 2 such relations: $k[c_1] = k'_1, k[c_2] = k'_2$. Since the behavior of the oracles in a view can be arbitrary, $k_1[c_1]$ and $k_2[c_2]$ might be the same for $(k_1, c_1) \neq (k_2, c_2)$. We refer to the scenario where $k_1[c_1]$ and $k_2[c_2]$ are identical despite (k_1, c_1) being different from (k_2, c_2) as a **collision**.

In this section, we will study a special type of view called *valid view* that only allows "mild" collisions. Then we introduce another distribution $\mathcal{D}'$ defined over valid views and show that $\mathcal{D}$ and $\mathcal{D}'$ are statistically close which is the key lemma in proving our main theorem.

6.1 Valid View

In this section, we will define a special type of view called "valid view". Formally, we make the following definition:

▶ **Definition 5.** *A view V is a valid view if it satisfies the following conditions:*

1. *For two distinct master keys $k_1, k_2 \in K_\kappa$, if they collide at k' in V, i.e. $k_1[c_1] = k_2[c_2] = k'$, then there will be no collisions related to k_1, k_2.*
2. *There are no self-collisions. Here, a self-collision means that one master key generates the same constrained key for two different predicates, i.e. $k[c_1] = k[c_2]$ for $c_1 \neq c_2$.*

In a valid view, master keys and constrained keys can be divided into the following 3 types of groups:

1. **Isolated constrained key:** This group only contains 1 constrained key k' which does not appear to be any master key's constrained key in the view. Formally speaking, there exists no k, c such that $k' = k[c]$ in this view.
2. **Normal group:** this group contains one master key k and the constrained keys generated from k and k does not collide with any other master keys.
3. **Hybrid group:** this group can be viewed as two normal groups colliding at a single constrained key. Suppose k_0, k_1 are two different master keys, $\{c_0, c_1, \ldots, c_s\}$ and $\{c'_0, c'_1, \ldots, c'_t\}$ are two sets of predicates (the predicates in each set are different from each other). $k_0[c_0] = k_1[c'_0] = k'$ represents the colliding constrained key. The rest constrained keys $k_0[c_i], k_1[c'_j], i, j \geq 1$ are different from each other, and all different from k'.

To better understand the concept of valid view, we can think of the master keys and constrained keys in a view as nodes. A master key is represented by a filled node and a constrained key is represented by an unfilled node. A constrained relation $k[c] = k'$ is represented by an edge c connecting k and k'. The following example is a valid view that contains a collision.

▶ **Example 6.** Suppose a view V contains the following query-answer pairs: $\mathsf{R}(k'_1, k'_3) = \perp$, $\mathsf{f}(k'_2, x) = y_1$, $\mathsf{R}(k'_4, k'_5) = (k_1, c_2, c_3)$, $\mathsf{Keygen}(k_1, c_1) = k'_3$, $\mathsf{f}(k'_6, x) = y_2$, $\mathsf{R}(k'_6, k'_8) = (k_2, c_4, c_6)$, $\mathsf{R}(k_2, k'_7) = c_5$, $\mathsf{R}(k'_8, k'_9) = (k_3, c_7, c_8)$, $\mathsf{f}(k'_9, x) = y_3$. Then, figure 1 is the graph illustration of the relations between the keys in V. Note that, V contains a collision: $k_2[c_6] = k_3[c_7] = k'_8$, this means that $k_2, k_3, k'_6, k'_7, k'_8, k'_9$ form a hybrid group.

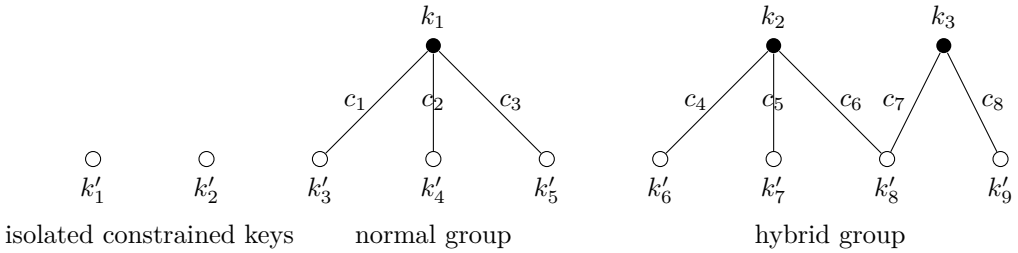

Figure 1 Example of a valid view V which contains 2 isolated constrained keys, 1 normal group and 1 hybrid group.

If a view is not valid, we can easily deduce that at least one of the following 4 complicated collisions must occur:

- Self-collision:

$$k[c_1] = k[c_2] = k'$$

- 3-to-1 collision: three master keys collide at one constrained key:

$$k_1[c_1] = k_2[c_2] = k_3[c_3] = k'$$

- 2-to-2 collision: two master keys collide at two different constrained keys:

$$k_1[c_1] = k_2[c_2] = k'_1, k_1[c'_1] = k_2[c'_2] = k'_2$$

- "W" collision: one master key simultaneously collides with two other master keys at two different constrained keys:

$$k_1[c_1] = k_2[c_2] = k'_1, k_1[c'_1] = k_3[c_3] = k'_2$$

We remark that valid views may contain collisions in the form of hybrid groups, this makes them different from the views in $\mathsf{sup}(\mathcal{D})$ which cannot contain any collisions because Keygen is sampled from injective functions. Thus, valid views can be seen as an extension over $\mathrm{sup}(\mathcal{D})$.

The readers may wonder why we allow a valid view to contain collisions. After all, collisions will never happen in a real execution because Keygen is sampled from all injective functions. The reason is that Eve's simulated view of Alice, combined with the view of Eve and Bob, might contain a collision. Looking ahead, we can see that the simulated view of Alice, combined with the view of Bob, forms a valid view if the simulation is a "miss".

6.2 Equivalent F or f Queries

Notice that in the definition of valid views, the correctness of PCPRF is not considered. Recall the correctness of PCPRF states that $\mathsf{F}(k,x)=\mathsf{f}(k[c],x)$ if $c(x)=1$. The reason that we have not yet considered the correctness of PCPRF is that, in order to evaluate $c(x)$, we might need to make extra H queries that are not in V. Thus, the correctness of PCPRF must be considered under a fixed instantiation H of the oracle H.

Suppose V is a valid view. H is an instantiation of the oracle H that is consistent with the H queries made in V. Suppose $k[c]=k'$ in V. The correctness of PCPRF under V,H states that $\mathsf{F}(k,x)=\mathsf{f}(k',x)$ if $c^H(x)=1$. We say that two F or f queries are equivalent under V and H if they can be deduced to have the same answer from the correctness of PCPRF under V,H.

Still take Example 6 as an illustration. Suppose H is an instantiation of the oracle H. Since no H queries are made in V, H is consistent with V. If $c_4^H(x)=c_6^H(x)=c_7^H(x)=c_8^H(x)=1$, we can say that $\mathsf{f}(k'_6,x)$ and $\mathsf{f}(k'_9,x)$ are equivalent, because the correctness of PCPRF states that $\mathsf{f}(k'_6,x)=\mathsf{F}(k_2,x)=\mathsf{f}(k'_8,x)=\mathsf{F}(k_3,x)=\mathsf{f}(k'_9,x)$. However, if at least one of $c_4^H(x),c_6^H(x),c_7^H(x),c_8^H(x)$ is 0, $\mathsf{f}(k'_6,x)$ and $\mathsf{f}(k'_9,x)$ are not equivalent under V,H. This is clearer in the graph illustration Figure 1. Note that the edges on the shortest path from k'_6 to k'_9 are exactly c_4,c_6,c_7,c_8. Thus, $\mathsf{f}(k'_6,x)$ and $\mathsf{f}(k'_9,x)$ are equivalent under V and H if and only if $c(x)=1$ for every c on this shortest path.

We remark that the answer of $\mathsf{f}(k'_6,x)$ and $\mathsf{f}(k'_9,x)$ in V, i.e. y_2,y_3 are irrelevant to whether $\mathsf{f}(k'_6,x)$ and $\mathsf{f}(k'_9,x)$ are equivalent under V,H.

6.3 Weights of Valid Views and the Indistinguishability Lemma

In this section, we define the notion of *weight* of valid views. With this notion, we may define a new distribution of views called $\mathcal{D}'$ which is distributed over all valid views, thus contains views with hybrid groups. Then, we will prove that $\Delta(\mathcal{D},\mathcal{D}')\le \mathrm{negl}(\lambda)$ (Lemma 8), which is essential to our black-box separation result.

First, even though the input of H can be any string, each party cannot make H queries with arbitrarily long input. Thus, we think of H as finite and denote the size of its domain as M. For a fixed instantiation H of the oracle H and a valid view V, we define $w_H(V)$ as follows:

1. If the H queries made in V are not consistent with H, then $w_H(V)=0$.
2. If two F or f queries made in V are equivalent under V,H, but their answers are different in V, then $w_H(V)=0$.
3. Otherwise, define $w_H(V)$ as follows: each constrained relation $k[c]=k'$ in V contributes a factor of $|K'_\kappa|^{-1}$. Here $k\in K_\kappa, c\in C_\kappa, k'\in K'_\kappa$. Each different F_κ or f_κ query in V contributes a factor of $|Y_\kappa|^{-1}$. However, equivalent F or f queries under V and H are counted only once. $w_H(V)$ is the product of all contributions.

Still take Example 6 as an illustration. Suppose the queries made in V are all in Γ_κ. Suppose H is an instantiation of the oracle H such that $c_4^H(x)=c_6^H(x)=c_7^H(x)=c_8^H(x)=1$. Now we want to compute $w_H(V)$:

1. Since no H queries are made in V, H is naturally consistent with V.
2. There are 3 F or f queries in V: $\mathsf{f}(k'_2,x),\mathsf{f}(k'_6,x),\mathsf{f}(k'_9,x)$. Since k'_2 is an isolated constrained key in V, $\mathsf{f}(k'_2,x)$ is nonequivalent to the other two queries. As we have shown in Section 6.2, $\mathsf{f}(k'_6,x),\mathsf{f}(k'_9,x)$ are equivalent under V, H. Thus, if $y_2\neq y_3$, $w_H(V)=0$. In the following, we assume $y_2=y_3$.

3. There are 8 constrained relations in V, each of which contributes a factor of $|K'_\kappa|^{-1}$. Next, there are 3 F or f queries in V, and two of them are equivalent. Thus, these 3 queries contribute a factor of $|Y_\kappa|^{-2}$ in total. Therefore, $w_H(V) = |K'_\kappa|^{-8}|Y_\kappa|^{-2} = 2^{-2\kappa^3-8\kappa^2}$.

Finally, the weight of a valid view V is defined as

$$w(V) = \frac{1}{2^M}\sum_H w_H(V)$$

Here, the summation is taken over all instantiations of H.

Then, the distribution $\mathcal{D}'$ is defined as follows:

▶ **Definition 7.** *$\mathcal{D}'$ is distributed over all valid views, the probability of a valid view V in $\mathcal{D}'$ is proportional to $w(V)$.*

We can, on some level, think of $w_H(V)$ as an approximated probability of the view V conditioned on H instantiated as H: for each pair of $k \in K_\kappa, c \in C_\kappa$, the constrained key $k[c]$ is sampled from K'_κ. Thus, for a specific k', the probability that $k[c] = k'$ is $|K'_\kappa|^{-1}$. Also, for each pair of $k \in K_\kappa, x \in X_\kappa$, the value of $\mathsf{F}(k, x)$ can be seen as sampled from Y_κ. Thus, for a specific y, the probability that $\mathsf{F}(k, x) = y$ is $|Y_\kappa|^{-1}$. Following this line of thought, we can think of $w(V)$ as an estimated probability of the view V, since $w(V)$ is the average value of $w_H(V)$ over all possible H. Therefore, it is plausible that $\mathcal{D}'$ and $\mathcal{D}$ should be close. Indeed, we can prove the following indistinguishability lemma:

▶ **Lemma 8.** *For any secret-key agreement protocol that each party never queries Γ_κ for $\kappa < \log_2 \lambda$ as we required in Section 5.1.2, it holds that $\Delta(\mathcal{D}, \mathcal{D}') \le negl(\lambda)$.*

The proof of Lemma 8 can be found in appendix B of the full version.

7 Breaking Secret-Key Agreement Protocol Relative to Oracle Γ

In this section, we will construct an adversary Eve who can break any secret-key agreement protocol in the normal form stated in the previous section.

7.1 Eve's Goal

In the normal form of KA protocol (see Section 5.1.2) Alice and Bob queries $\mathsf{H}(s_A)$ and $\mathsf{H}(s_B)$ respectively in the last round, where s_A and s_B are Alice and Bob's output. With probability $\alpha(\lambda)$, $s_A = s_B = s$, then both Alice and Bob would have made the same query $\mathsf{H}(s)$ at the end of the protocol. This means, $s \in Q(V_A) \cap Q(V_B)$, here V_A and V_B denote the view of Alice and Bob at the end of the protocol. Thus, Eve's mission can be reduced to find all the intersection elements $Q(V_A) \cap Q(V_B)$. If Eve accomplishes to include all the intersection elements with probability $1 - \alpha/2$, then she finds all the intersection elements with probability at least $1/2$ conditioned on $s_A = s_B$. Thus, because Eve's view only contains polynomially many elements, Eve can simply choose one element randomly from her view as her guess. The guess would be s with at least $1/\mathsf{poly}(\lambda)$ probability. This means the protocol is broken. In the following, Eve's goal is to find all the intersection elements at the end of the protocol.

7.2 Eve's Algorithm

Suppose r is even, then Bob speaks in round r and Alice speaks in round $r-1$. Eve's algorithm in round r can be seen as running simultaneously with Bob's algorithm in round r. Eve's algorithm in round r consists of many repeated segments. Before segment i, Eve's partial view is $V_E^{r,i-1}$ which is complete. Then, segment i is divided into the following two phases:

- **Simulation Phase**: sample a view of Alice up to the end of round $r-1$: $\mathrm{Sim}_A \leftarrow \mathsf{Sim}(M^r, V_E^{r,i-1})$. Here, the distribution $\mathsf{Sim}(M^r, V_E^{r,i-1})$ is defined as follows: $\mathsf{SIM}(M^{r-1}, V_E^{r,i-1})$ is distributed over all possible partial views of Alice up to the end of round $r-1$ (denoted as Sim_A) that are consistent with M^{r-1}. Besides, $\mathrm{Sim}_A \cup V_E^{r,i-1}$ must form a collision-free valid view. The probability of Sim_A in $\mathsf{SIM}(M^r, V_E^{r,i-1})$ is proportional to $w(\mathrm{Sim}_A \cup V_E^{r,i-1})$.
- **Update Phase**: Eve includes all the elements in $Q(\mathrm{Sim}_A)$ in her view. Here, Sim_A is the sampled view of Alice in the last simulation phase. Then, Eve will try to acquire informations related to $Q(V_E)$ as much as possible. Specifically, for each pair of $k \in K, c \in C$ in her view, she will ask $\mathsf{Keygen}(k, c)$, the answer is automatically included in her view; for each pair of $x \in X, k \in K$ (resp. $k' \in K'$), she will ask $\mathsf{F}(k, x)$ (resp. $\mathsf{f}(k', x)$); Eve always make best efforts to recover the relationships of the master-keys and constrained keys in her view, this can be done by the recovery oracle R. Additionally, Eve will always maintain the completeness of her view. After the update, Eve's view is $V_E^{r,i}$. Naturally, $V_E^{r,i}$ is also complete.

Then, round r proceeds as follows (including Bob and Eve's algorithm):

1. Eve runs m segments described above. (Here $m = m(\lambda)$ is polynomial over λ. Its value is determined later.)
2. Bob makes the first query in round r.
3. We assume that there is a fourth party Charlie. She has the view of all three parties. Charlie is designated for one purpose: if Bob's first query in the last step is an R query and the answer is not $\perp$, then the answer must contain 1 or 2 predicates, say c_1 (and c_2). If c_1 has never appeared in Bob and Eve's view before, Charlie will send c_1 to Eve. Same for c_2. Otherwise Charlie simply sends $\perp$ to Eve.
4. If Charlie's message is not $\perp$ in the last step, Eve will include the received predicates in her view and make her view full, same as the update phase of a segment.
5. Eve runs m segments.
6. Bob makes the rest H queries in round r.
7. Bob sends the message M^r to Alice.

Here, Bob's queries are divided into 2 parts: the first query and the rest $2n\kappa_{\max}$ H queries. Before each part, Eve runs m segments. Between these two parts, we introduces a fourth party Charlie that might send one or two predicates to Eve. If Eve does not receive $\perp$, she will run another update phase to include the received predicates into her view. Indeed, Eve might receive private informations from Charlie, which renders Eve's attack invalid. Therefore, we make the following rule: as long as Charlie sends a message other than $\perp$, Eve is considered failed. In other words, we say Eve fails if (1), Eve does not include all the intersection elements in her view; (2), Charlie sends a message other than $\perp$. We will show that the probability that Eve fails is small. This means that the probability that Eve fails is also small even without Charlie.

7.3 Distribution $\mathcal{D}^{r,i,j}$ and $\mathcal{D}'^{r,i,j}$

The definition of $\mathcal{D}^{r,i,j}, \mathcal{D}'^{r,i,j}$ is the same as $\mathcal{D}, \mathcal{D}'$, except that we make the algorithms of Alice and Bob stop after the j th query of round r and Eve's algorithm stops after segment i of round r. (Especially, if $i = m$, the view sampled from $\mathcal{D}^{r,m,j}$ or $\mathcal{D}'^{r,m,j}$ should also contain the part that Charlie sends messages to Eve and Eve makes updates.) Proof of Lemma 8 also shows $\Delta(\mathcal{D}^{r,i,j}, \mathcal{D}'^{r,i,j}) \leq \text{negl}(\lambda)$.

Recall that the sampling process of $\mathcal{D}$ is as follows: first sample the randomness of three parties r_A, r_B, r_E (Charlie is deterministic) and the whole oracle Γ, then run the protocol. The only difference between $\mathcal{D}$ and $\mathcal{D}^{r,i,j}$ is that the protocol of $\mathcal{D}^{r,i,j}$ stops earlier than $\mathcal{D}$. Thus, we can sample $V^{r,i,j} \leftarrow \mathcal{D}^{r,i,j}$ in the following way: sample $V \leftarrow \mathcal{D}$ first, then output the partial view $V^{r,i,j}$. [2]

7.4 Main Result

Under certain parameter regime, we prove that Eve can break the secret-key agreement protocol with high probability. Let $S(\lambda) = 1/\text{poly}(\lambda)$, we have

▶ **Theorem 9.** *At the end of the protocol, Eve finds all the intersection elements of Alice and Bob's view while Charlie always sends $\perp$ with high probability. Formally, for some sufficiently large polynomial m depending on $S(\lambda)$ (determined later),*

$$\Pr_{V \leftarrow \mathcal{D}}[Q(V_A^n) \cap Q(V_B^n) \subseteq Q(V_E^{n,2m}) \wedge \textit{Charlie always sends } \perp] > 1 - S(\lambda)$$

for sufficiently large λ.

Thus, as long as $S(\lambda) < \alpha(\lambda)/2$, Eve can break the KA protocol as discussed in Section 7.1.

7.5 Proof of Theorem 9

In the rest of this section, we will give a proof sketch for our main result Theorem 9.

Recall that Eve's goal is to include all the intersection elements in her view, while Charlie always sends $\perp$. We instead prove a stronger result: before an element q becomes an intersection element, Eve will anticipate q with high probability. We say Eve fails if

1. An element q becomes an intersection element as an input to a query, but Eve's view does not include it.
2. An element q becomes an intersection element as an output of a query, but Eve's view does not include it.
3. Charlie sends a message other than $\perp$.

If Eve never fails through the whole execution of the protocol, all intersection elements must be included in Eve's view and Charlie always sends $\perp$. This means that Eve accomplishes her goal. If Eve fails, there must be a first failure. We first show the probability that Eve's first failure is type 2 or 3 is negligible.

▶ **Lemma 10.** *Suppose V is sampled from $\mathcal{D}$. Then the probability that Eve fails in V and the first failure is type 2 or 3 is negligible.*

[2] However, the same does not necessarily hold for $\mathcal{D}'$ and $\mathcal{D}'^{r,i,j}$ because $\mathcal{D}'$ and $\mathcal{D}'^{r,i,j}$ are defined in a completely different way than $\mathcal{D}$ and $\mathcal{D}'^{r,i,j}$.

Here, we only give a proof sketch of this lemma. We refer the interested readers to the full version. Note that the size of K'_κ is 2^{κ^2}. In K'_κ, there are only $|K_\kappa| \cdot |C_\kappa| = 2^{2\kappa}$ constrained keys. This proportion is negligible as long as $\kappa \geq \log \lambda$. Therefore, it is almost impossible to find a constrained key in $|K'_\kappa|$ without using Keygen oracle. We denote this event as spoof. The key observation is that, as long as Eve fails and the first failure is type 2 or 3, then the event spoof occurs.

With this lemma, we only need to concern type 1 failures. Recall that Alice and Bob can make at most $1+2n\kappa_{\max}$ queries in each round, and there are n rounds in total, Alice and Bob can make at most $n(1+2n\kappa_{\max})$ queries in total. Denote $p(\lambda) = S(\lambda)/n(\lambda)(1+2n(\lambda)\kappa_{\max}(\lambda))$. If we can bound the probability that the input of the jth query of round r is Eve's first failure by p, we can bound the probability that Eve fails. Here, we consider the first query and the rest H queries separately. Formally, we prove the following 2 lemmas:

▶ **Lemma 11.** *For some sufficiently large polynomial $m = m(\lambda)$ depending on $p(\lambda)$, for each round number $r \in [1, n]$ (without loss of generality assume Bob speaks in round r), the probability that either*

1. *$Q(V_A^{r-1}) \cap Q(V_B^{r-1}) \not\subseteq Q(V_E^{r,0})$. (At the end of round $r-1$, Eve failed to include all intersection elements. This is stronger than Eve failing before round r).*
2. *$q_{r,1} \notin Q(V_A^{r-1})$. (For $q_{r,1}$ to be an intersection element, it must have appeared in Alice's view before round r.)*
3. *$q_{r,1} \in Q_E^{r,m}$. (The definition of Eve finding $q_{r,1}$ at the end of the first m segments.)*

is greater than $1 - p(\lambda)/2$ for sufficiently large λ.

▶ **Lemma 12.** *For some sufficiently large polynomial $m = m(\lambda)$ depending on $p(\lambda)$, for each round number $r \in [1, n]$ (without loss of generality assume Bob speaks in round r) and query number $j \in [2, 1 + 2n\kappa_{\max}]$, the probability that either*

1. *$Q(V_A^{r-1}) \cap Q(V_B^{r,j-1}) \not\subseteq Q(V_E^{r,m})$. (Right before the jth query of round r, Eve failed to include all intersection elements with the first m segments. This is stronger than Eve failing before the jth query of round r).*
2. *$q_{r,j} \notin Q(V_A^{r-1})$. (For $q_{r,j}$ to be an intersection element, it must have appeared in Alice's view before round r.)*
3. *$q_{r,j} \in Q(V_E^{r,2m})$. (The definition of Eve finding $q_{r,j}$ at the end of round r.)*

is greater than $1 - p(\lambda)/2$ for sufficiently large λ.

Note that, if Condition 1 $\vee$ Condition 2 $\vee$ Condition 3 has probability greater than $1-p/2$, then $\neg$Condition 1 $\wedge$ $\neg$Condition 2 $\wedge$ $\neg$Condition 3 has probability smaller than $p/2$, and this is weaker than Eve first failing at this query.

The detailed proofs of Lemma 11 and Lemma 12 can be found in appendix C of the full version.

Now Theorem 9 can be proved by Lemma 11 and Lemma 12:

Proof. Combining Lemma 11 and Lemma 12, for each round number r and query number j, the probability that Eve's first failure is the input of the jth query of round r and the failure is type 1 is bounded by $p/2$. Apply a union bound, the probability that Eve fails and the first failure is type 1 is bounded by $(p/2) \cdot n(1 + 2n\kappa_{\max}) = S/2$. Combined with Lemma 10, the probability that Eve fails is bounded by $S/2 + \mathrm{negl}(\lambda) < S$ for sufficiently large λ. This is stronger than Theorem 9. ◀

References

1 Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. *SIAM Journal on Computing*, 45(6):2117–2176, 2016. doi:10.1137/15M1034064.

2 Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained prfs for NC^1 in traditional groups. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 543–574, Cham, 2018. Springer International Publishing.

3 Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively single-key secure constrained prfs for NC^1. In *Public Key Cryptography (2)*, volume 11443 of *Lecture Notes in Computer Science*, pages 223–253. Springer, 2019.

4 Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle puzzles are optimal — an $O(n^2)$-query attack on any key exchange from a random oracle. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, pages 374–390, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-03356-8_22.

5 Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. In *Public Key Cryptography (2)*, volume 10175 of *Lecture Notes in Computer Science*, pages 494–524. Springer, 2017. doi:10.1007/978-3-662-54388-7_17.

6 Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT (2)*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300. Springer, 2013. doi:10.1007/978-3-642-42045-0_15.

7 Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, pages 480–499, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. doi:10.1007/978-3-662-44371-2_27.

8 Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *Public-Key Cryptography – PKC 2014*, pages 501–519, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. doi:10.1007/978-3-642-54631-0_29.

9 Zvika Brakerski, Jonathan Katz, Gil Segev, and Arkady Yerukhimovich. Limits on the power of zero-knowledge proofs in cryptographic constructions. In Yuval Ishai, editor, *Theory of Cryptography*, pages 559–578, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-19571-6_34.

10 Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained prfs (and more) from lwe. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 264–302, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-70500-2_10.

11 Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic prfs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In *TCC (2)*, volume 9015 of *Lecture Notes in Computer Science*, pages 1–30. Springer, 2015. doi:10.1007/978-3-662-46497-7_1.

12 Ran Canetti and Yilei Chen. Constraint-Hiding Constrained PRFs for NC^1 from LWE. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, volume 10210, pages 446–476. Springer International Publishing, Cham, 2017. Series Title: Lecture Notes in Computer Science. doi:10.1007/978-3-319-56620-7_16.

13 Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure constrained pseudorandom functions in the standard model. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 559–589, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-56784-2_19.

14 Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ameer Mohammed. Limits on the power of garbling techniques for public-key encryption. Cryptology ePrint Archive, Paper 2018/555, 2018. URL: https://eprint.iacr.org/2018/555.

15 R. Gennaro and L. Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 305–313, 2000. doi:10.1109/SFCS.2000.892119.

16 Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The relationship between public key encryption and oblivious transfer. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 325–335, 2000. doi:10.1109/SFCS.2000.892121.

17 Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 126–135. IEEE, 2001. doi:10.1109/SFCS.2001.959887.

18 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. doi:10.1145/6490.6503.

19 Dennis Hofheinz, Akshay Kamath, Venkata Koppula, and Brent Waters. Adaptively secure constrained pseudorandom functions. In Ian Goldberg and Tyler Moore, editors, *Financial Cryptography and Data Security*, pages 357–376, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-32101-7_22.

20 R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, pages 44–61, New York, NY, USA, 1989. Association for Computing Machinery. doi:10.1145/73007.73012.

21 Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 669–684, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2508859.2516668.

22 Arthur Lazzaretti and Charalampos Papamanthou. Near-optimal private information retrieval with preprocessing. In Guy Rothblum and Hoeteck Wee, editors, *Theory of Cryptography*, pages 406–435, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-48618-0_14.

23 Mohammad Mahmoody, Hemanta K. Maji, and Manoj Prabhakaran. *On the Power of Public-Key Encryption in Secure Computation*, pages 240–264. Springer Berlin Heidelberg, 2014. doi:10.1007/978-3-642-54242-8_11.

24 Takahiro Matsuda and Kanta Matsuura. On black-box separations among injective one-way functions. In *Theory of Cryptography Conference*, pages 597–614. Springer, 2011. doi:10.1007/978-3-642-19571-6_36.

25 Steven Rudich. The use of interaction in public cryptosystems (extended abstract). In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 242–251. Springer, 1991. doi:10.1007/3-540-46766-1_19.

26 Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 475–484, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2591796.2591825.

27 Elaine Shi, Waqar Aqeel, Balakrishnan Chandrasekaran, and Bruce Maggs. Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 641–669, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-84259-8_22.

28 Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, pages 334–345, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. doi:10.1007/BFB0054137.