How to Simulate Random Oracles with Auxiliary Input

Yevgeniy Dodis
New York University
dodis@cs.nyu.edu

Aayush Jain
Carnegie Mellon University
aayushja@andrew.cmu.edu

Huijia Lin Ji Luo
University of Washington
{rachel,luoji}@cs.washington.edu

Daniel Wichs
Northeastern University and NTT Research
wichs@ccs.neu.edu

***Abstract*—The *random oracle model* (ROM) allows us to optimistically reason about security properties of cryptographic hash functions, and has been hugely influential in designing practical cryptosystems. But it is overly optimistic against non-uniform adversaries, and often suggests security properties and security levels unachievable by any real hash function. To reconcile with this discrepancy, Unruh [CRYPTO '07] proposed the *auxiliary-input random oracle model* (AI-ROM), where a non-uniform attacker additionally gets a bounded amount of advice about the random oracle.**

Proving security in the AI-ROM is often much more difficult, but a series of works starting with Unruh provided useful technical tools to do so. Although these tools lead to good results in the information-theoretic setting, they are unsatisfactory in the computational setting, where the random oracle is used alongside other computational hardness assumptions. At the most basic level, we did not even know whether it is possible to efficiently simulate random oracle queries given auxiliary input, which has remained as an explicit open problem since the work of Unruh.

In this work, we resolve the above open problem and show how to efficiently simulate auxiliary-input random oracles. Moreover, the simulation has low concrete overhead, leading to small losses in exact security. We use it to prove the security of a broad class of computational schemes in the AI-ROM, including the first non-interactive zero-knowledge (NIZK) scheme in the AI-ROM. As a tool of independent interest, we develop a new notion of ultra-secure pseudorandom functions with fast RAM evaluation, which can achieve 2^λ security while having sublinear $o(\lambda)$ evaluation time.

1. Introduction

Random Oracles. The random oracle model (ROM) [1] is an idealization of cryptographic hash functions into public random functions $\mathcal{O}$ that can be queried on arbitrary inputs. This optimistic modeling attempts to capture the intuition that hash functions can satisfy a wide range of security properties beyond standard ones such as one-wayness or collision resistance. The ROM has been hugely successful and influential in allowing us to easily design and analyze cryptographic schemes that use hash functions. Essentially all cryptosystems used in practice, including Fiat–Shamir or full-domain-hash signatures (e.g., DSA, Schnorr, RSA), CCA encryption (e.g., RSA-OAEP), and Bitcoin, are only shown to be secure in the ROM. Unfortunately, we have no formal proof that security in the ROM implies security in the real world, when the oracle is replaced with a good hash function, and there are contrived counterexamples [2,3] showing that this is not always the case. Nevertheless, security in the ROM is seen as providing very strong heuristic evidence of security in the real world. Moreover, the ROM is not only used to analyze security in the asymptotic sense, but is also used to analyze the *exact security* of a scheme, which is crucial for choosing concrete parameters.

Non-Uniformity. Unfortunately, the ROM does not adequately capture *non-uniform adversaries* (i.e., families of circuits or Turing/RAM machines with advice), and security in the ROM cannot be used as heuristic evidence for real-world security in the non-uniform setting. For example, it is easy to construct seedless collision-resistant hash functions in the ROM, but they are known to be impossible against non-uniform adversaries in the real world. Indeed, a non-uniform attacker can simply hardwire a collision into the (short) advice string.

In a different vein, the ROM gives us incorrect predictions for the security levels that we can expect against non-uniform adversaries. For example, the seminal work of Hellman [4] showed that a non-uniform attacker of size $2^{2\lambda/3}$ can invert a random length-preserving function on λ bits with constant probability, and Fiat and Naor [5] extended this to any function with circuits of size $2^{3\lambda/4}$. In contrast, in the ROM, the λ-bit random oracle itself can easily seen to be $Q/2^\lambda$ hard to invert against attackers making Q random oracle queries, which gives negligible inversion probability for $Q = 2^{3\lambda/4}$. Moreover, this gap is far from theoretical. The famous *rainbow tables* [6] were developed based on these ideas, and are used on a massive scale to crack weak passwords in the real world. As another example of this kind, one can construct a simple pseudorandom generator (PRG)

in the ROM with seed length λ, for which no adversary running in time Q can distinguish the output from uniform with advantage better than $Q/2^{\lambda}$. However, for every real PRG, the works of (e.g.) [7,8] show that there is a simple non-uniform adversary running in time $\mathrm{O}(\lambda)$ that has significantly higher distinguishing advantage $2^{-\lambda/2}$. Similar discrepancies between the security levels achievable in the ROM versus the real world in the non-uniform setting exist for many other primitives (see [9–11]).

To summarize, while the ROM gives good heuristic predictions for the security properties and levels that cryptographic hash functions can achieve against uniform adversaries, it frequently gives wrong, overly optimistic predictions in the non-uniform setting.

Auxiliary-Input ROM. To bridge the above discrepancy, Unruh [9] defined the *auxiliary-input random oracle model* (AI-ROM) as a way of capturing non-uniform attacks in the ROM. In the AI-ROM, a random function $\mathcal{O}$ is chosen initially, but the adversary is then given some S bits of arbitrary auxiliary input $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$ about the function $\mathcal{O}$. The mapping ai can be an arbitrary function and may not be efficient or even computable. The adversary gets the auxiliary input z and oracle access to $\mathcal{O}$, after which it gets to break the cryptosystem by making up to Q additional queries to $\mathcal{O}$. The honest users only get oracle access to $\mathcal{O}$, as in the standard ROM. We can think of S and Q as an arbitrary polynomial when discussing polynomial security, or can put precise bounds when discussing exact security.[1]

Unlike the ROM, the AI-ROM appears to rule out many security properties that are "too good to be true" in the non-uniform setting. For example, one cannot construct seedless collision-resistant hash functions in the AI-ROM since a collision (of any compressing function determined by $\mathcal{O}$) can be provided as part of the auxiliary input z. Moreover, the generic non-uniform attacks against one-way functions, pseudorandom generators, and other primitives discussed above carry over to the AI-ROM, ensuring that for these primitives in the AI-ROM, one cannot achieve exact security levels known to be impossible in the plain model.

Proving Security in AI-ROM. While the AI-ROM prevents us from proving many unrealistic security properties, the flip side is that it makes it difficult to prove anything. Even simple properties that are obvious in the ROM become difficult to prove in the AI-ROM. This is because many useful proof techniques in the ROM do not seem to extend to AI-ROM, including "lazy sampling", (dynamic) programmability of the oracle, and the "unpredictability-to-randomness" conversion. For example, lazy sampling allows one to not commit to the entire truth table of the oracle upfront, and instead answer every fresh query with a fresh random value. Unfortunately, this cannot be done naïvely in the AI-ROM, where the auxiliary input z may be correlated with the entire oracle, and choosing a fresh random value will be easily detectable by the attacker. This leads to the main central question of this work.

> **Main Question:** Develop techniques for showing *tight security bounds* of cryptographic applications in the AI-ROM.

As we will see below, this question has been relatively well understood for what we call *information-theoretic applications* of the AI-ROM. This means that the only computational component in the security of such application comes from the use of cryptographic hash function H. Once H has been abstracted as AI-ROM $\mathcal{O}$ with parameters S, Q, the attacker can be assumed to be otherwise computationally unbounded.[2] Examples of such applications include pseudorandomness, one-wayness, or ("salted") collision resistance of the hash function. While very useful in themselves, in most real-world applications — Fiat–Shamir signatures, full-domain hash, Bitcoin, RSA-OAEP among many many others — the hash function is only part of the larger system, and additional computational assumptions should be used. We will see that the existing state-of-the-art techniques give extremely poor exact security bounds for such *computational* applications, despite the fact we believe all these applications are actually secure. The main result of this work will be a *new AI-ROM simulation technique* to give a *much tighter answer* to our Main Question for *all* such computational applications.

Presampling Technique. The most general known answer to our Main Question comes from the beautiful work of Unruh [9] (which also defined AI-ROM). The technique is called *presampling* and could be thought as a particular AI-ROM simulation method, which effectively reduces AI-ROM to a "friendlier" intermediate model termed *bit-fixing* ROM (BF-ROM) by [11]. Unlike (S, Q)-AI-ROM, which has two parameters S (size of auxiliary input) and Q (number of online queries), the P-BF-ROM has a single parameter P, allowing the attacker to fix the oracle $\mathcal{O}'$ arbitrarily at any set of P points in the domain. After this, the values of $\mathcal{O}'$ on all the remaining points are chosen randomly and independently, just as in the ROM. Intuitively, as long as a given application can avoid using the P "presampled" points for security, traditional ROM techniques (including lazy sampling and others) are back in business.

More formally, Unruh showed that any adversary who gets S-bit auxiliary input $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$ and makes Q oracle queries to $\mathcal{O}$ (call it an (S, Q)-attacker) cannot distinguish interacting with $\mathcal{O}$ from interacting with an oracle $\mathcal{O}'$ in the P-BF-ROM, where the P points of presampling and the oracle outputs are allowed to depend on the function ai and the value $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$. In terms of concrete bounds, the distinguishing advantage ε is upper bounded by $\sqrt{SQ/P}$, and this was later improved by [11] to SQ/P. Moreover, the latter bound $\varepsilon \le SQ/P$ is known to be tight at this level of generality [11].

1. Moreover, in the circuit model we can often view $S = Q$ as the size of the adversary's non-uniform circuit, while in the RAM model it is more convenient to separate the two.

2. In a sense that placing additional computational limitations do not appear to help proving stronger results for such applications.

The work of [11] also showed that one can take fewer presampled points $P = \mathrm{O}(SQ)$ and still ensure small "multiplicative distinguishing advantage" — any event that happens with probability at most δ when the adversary has oracle access to $\mathcal{O}'$ can happen with probability at most 2δ when the adversary has oracle access to $\mathcal{O}$. This is useful for analyzing unpredictability (e.g., one-wayness) rather than indistinguishability (e.g., pseudorandomness) applications. We also note that several other works [8,10,12] explored other approaches to proving specific security properties in the AI-ROM, directly without going through presampling. However, all these techniques suffer even worse limitations than presampling, which we discuss next.

1.1. Limitations of Presampling

Presampling is a very general and powerful technique. For a simple information-theoretic example, presampling allows us to easily prove that a length-doubling random oracle $\mathcal{O} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ is a good PRG in the AI-ROM. We first replace $\mathcal{O}$ by $\mathcal{O}'$ at security loss $\varepsilon = SQ/P$ and then replace $\mathcal{O}'(x)$ for a random seed $x \xleftarrow{\$} \{0,1\}^\lambda$ by a uniformly random output at security loss $(P+Q)/2^\lambda$, reflecting the probability that x falls into the set of the P presampled points or that the adversary queries the oracle on x. Since in this simple example, PRG is our final goal, we can optimize the value of P for the best-possible bound. Concretely, by setting $P = \sqrt{2^\lambda SQ}$, the total distinguishing advantage is $\mathrm{O}\big(\sqrt{SQ/2^\lambda} + Q/2^\lambda\big)$, negligible for polynomial S, Q. Moreover, for polynomial S, Q, it asymptotically matches the known linear-time $2^{-\lambda/2}$ attack on all non-uniform PRG.[3]

Computational Applications? Unfortunately, things start to go astray once we add computational assumptions to the picture. In fact, the difficulty already comes up when we try analyzing schemes that do not make any meaningful use of the AI-ROM! This question is interesting in it own right, as it directly compares the power of AI-ROM versus the standard non-uniform model for *standard-model problems*, which make sense in both models. But it is also must be settled before we even dream to go for the real problem we care about — applications which use cryptographic hash functions *together with* additional computational assumptions, i.e., virtually all real-world applications.

For example, say we want to argue that the decisional Diffie–Hellman (DDH) problem is hard in the AI-ROM. At first, this may appear obvious — the DDH problem does not involve any oracles, so why would it be any easier in the AI-ROM than in the plain model?! But if we want to reduce breaking DDH in the plain model to breaking it in the AI-ROM, we need to *efficiently simulate AI-ROM queries*. In the standard ROM, this is trivial via lazy sampling, but as we already saw, this technique does not work in the AI-ROM. Again, on an intuitive level, such queries should be completely useless in breaking DDH. Formally, though, it is unclear how to do such simulation without the attacker noticing some inconsistency with its advice (and perhaps refusing to break DDH for this reason). Indeed, without the ability to simulate the oracle, we cannot formally argue that the AI-ROM does not give the adversary some unspecified computational power, which may perhaps even suffice to break the DDH assumption!

Unruh's Conjecture. The above issue was already noted by Unruh, who conjectured — but left it as one of the main open problems — that "common sense wins". Namely, efficient simulation of the AI-ROM should be possible [14; Conjecture 15]:

> *Consider an adversary $\mathcal{A}$ that gets some S-bit auxiliary input $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$ and makes oracle queries to $\mathcal{O}$. Is there an efficient non-uniform simulator $\mathcal{S}$ that simulates the view of $\mathcal{A}$ such that no efficient environment can tell apart whether it is interacting with $\mathcal{A}^{\mathcal{O}}(z)$ or $\mathcal{S}$? What is the concrete efficiency of $\mathcal{S}$ compared to that of $\mathcal{A}$?*

The environment does not have access to the oracle. For example, we can think of the environment as the challenger for DDH. In that case, for any attacker $\mathcal{A}^{\mathcal{O}}(z)$ in the AI-ROM that breaks DDH, a similarly efficient non-uniform simulator $\mathcal{S}$ in the plain model should break DDH as well.

One could attempt to use presampling to construct the simulator $\mathcal{S}$. The non-uniform simulator would have $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$ as advice along with the table of P presampled input/output pairs. It would then run $\mathcal{A}$, and answer any oracle query that falls into the table of presampled inputs with the corresponding presampled output and use lazy sampling on all the other inputs. Unfortunately, to achieve negligible distinguishing advantage ε for adversaries $\mathcal{A}$ making Q oracle queries, the non-uniform advice that the simulator needs would be of size SQ/ε, which is super-polynomial! In terms of concrete efficiency, it looks even worse. For example, to argue that DDH with certain parameters has $\varepsilon = 2^{-\lambda}$ security against circuits of size 2^λ (bounding both S, Q by 2^λ) in the AI-ROM, we would need DDH with those parameters to have security $\varepsilon = 2^{-\lambda}$ against much larger circuits of size $2^{4\lambda}$ in the plain model, by setting $P = SQ/\varepsilon = 2^{3\lambda}$ and noting that the above described simulator $\mathcal{S}$ needs to scan through the table of size P on every oracle query made by $\mathcal{A}$.[4]

Perhaps one could hope to get better simulation by improving the analysis of presampling, to allow for fewer presampled values P. Unruh conjectured this to be possible [9; Section 6] with $P = \mathrm{poly}(S, Q)$ and a negligible ε. Unfortunately, the work of [10] showed that this conjecture is false and that $\varepsilon = \Theta(SQ/P)$ is optimal. Therefore, to get better simulation, an entirely new technique is necessary.

We also notice that the situation is also far from satisfactory for unpredictability applications, such as computational Diffie–Hellman problem (CDH), despite the more optimized presampling for such applications [11]. Indeed,

3. Interestingly, the bound is not tight for larger some super-polynomial ranges of S, Q, but this was subsequently settled by [13].

4. This last 2^λ factor can be removed if we consider non-uniform simulation in the RAM model instead of the circuit model.

even though the size of the presampled set is reduced to $P = \mathrm{O}(SQ)$, it is still quadratic in S when $T = \Omega(S)$. Therefore, to convert such $(S, \Omega(S))$-attacker in AI-ROM to the standard non-uniform model, the non-uniform advice (which includes the presampled set) will have quadratic size $S' = \Omega(P) = \Omega(S^2)$. Moreover, the actual simulation (in the circuit model) might take even cubic time $\Omega(S^3)$, as each evaluation inside the presampled set will take time $\Omega(S') = \Omega(S^2)$.[5] Thus, even though this resolves a variant of Unruh's conjecture for unpredictability up to polynomial factors, it does not match our intuition expecting $S' = \mathrm{O}(S)$ to hold.

Need for Better Simulation. To summarize, while presampling works in converting AI-ROM attackers to standard non-uniform ones, the reduction is not efficient in the asymptotic sense (polynomial/negligible) nor in terms of concrete security. In particular, it gives too pessimistic bounds in the non-uniform setting.[6] Put differently, for computational applications of hash functions, the route through the intermediate BF-ROM appears to be too wasteful, and a better simulation technique is needed. In this work, we present such a technique and then use it to prove various computational security notions in the AI-ROM.

1.2. Our Results in a Nutshell

Key Idea: Use a PRF. The main idea of our new technique is to use a pseudorandom function (PRF) to simulate AI-ROM in the standard non-uniform model. The new non-uniform advice z' will be the key K for the PRF, together with the S-bit output of the AI-ROM leakage function ai applied to the *entire truth table V of the PRF*. As such, the adversary $\mathcal{A}$ would not be able to find any inconsistencies of the simulated leakage not matching the evaluation. Of course, this (necessary) step is still insufficient, as our goal will be to make the PRF key K as short as possible (say, $\mathrm{O}(S)$ bits long) and the (computationally unbounded) leakage function ai might be able to tell the truth table V of a PRF from an exponentially long truly random string. Here we will rely on a beautiful sequence of works [15,16], which shows that any "short enough" (in our case, S-bit long) leakage $\mathsf{ai}(\mathcal{O})$ could be replaced by a different leakage function $\mathsf{ai}'(\mathcal{O})$, where ai' is "computationally efficient". Once this is done, we will be able to formally use the PRF security to argue that the actual (efficient) non-uniform attacker should not be able to detect the fact that we replaced $\mathsf{ai}(\mathcal{O})$ by $\mathsf{ai}'(V)$.

Super-Fast PRF. Ignoring some subtle choices of parameter selection, one issue remains — efficiency of the PRF. Namely, in AI-ROM, the cost of Q queries is abstracted out as equal to Q. With our PRF simulation, however, each such call will require evaluating the PRF on some input, and the key K for this PRF will be at least S bits long (otherwise, we cannot apply the leakage simulation technique of [15, 16]). Hence, if done naïvely, we will spend at least time $\Omega(SQ)$ to simulate the Q random oracle queries from the attacker. This is still interesting and will compare favorably with presampling, but we will try to do better. Namely, as a contribution of independent interest, we define and construct a novel kind of PRF with λ bits of security and key length (where think of $\lambda \sim S$), where individual PRF evaluations take time polylogarithmic in λ (and polynomial in the input length). In particular, each individual PRF evaluation only reads a small subset of the key bits.

Applications. Taken together, we will achieve a novel kind of AI-ROM simulation in the standard non-uniform model, where $S' \sim S$ and $Q' \sim Q$ for S' being the size of the new advice and Q' being the time to simulate the oracle queries. This immediately *settles Unruh's conjecture in the affirmative*, with essentially tight simulation, confirming our intuition that AI-ROM should not help with standard-model problems.

Unfortunately, our new simulation by itself is not sufficient for answering our main question — security of computational applications in the AI-ROM. This is because once we simulate AI-ROM by a PRF, we lose all the nice (usually information-theoretic) properties of the random oracle, which are needed to analyze the hash-function part of the applications. For example, if we analyze hashed ElGamal encryption, our simulation will allow us to use the DDH assumption to replace the Diffie–Hellman group element by a random group element, but now it is unclear how to argue that applying the hash function effectively acts as a PRG (from group to strings), as the hash function is already replaced by a PRF whose key is no longer secret.

We develop a relatively general template to this dilemma, by *combining* our new PRF-simulation with presampling. We describe this in more detail in Section 1.3, here only highlighting the main insight. In essence, our template allows us to combine the best features of both simulation methods, without inheriting any of the disadvantages. Concretely, we borrow a fresh-independent random oracle part from presampling, which allows us to use ROM techniques (such as lazy sampling) to tackle the hash-function part or the analysis, provided honest parties avoid evaluating the hash function on the presampled set P. On the other hand, we no longer store the (quite long) truth table of P as an advice, and instead use our more efficient PRF simulation for the behavior of the random oracle inside P (which will likely be used by the attacker). Some subtleties need to be addressed to make it work, but we mention the two main applications where we overcame these subtleties.

- Extending the previous information-theoretic proofs of [10,11] for information-theoretic applications in the AI-ROM, we show that *salting generically defeats preprocessing* for all (even computational) AI-ROM applications, with a tighter reduction than before.
- The first known construction of *non-interactive zero-knowledge (NIZK) proofs in the AI-ROM*. When

5. In the RAM model, we still get a quadratic slow-down.

6. This is the opposite problem to the one we had when adapting standard ROM bounds to the non-uniform setting, where the bounds were unrealistically good.

the hash function is instantiated, this gives the first heuristic NIZK candidate which is secure against preprocessing attacks.

We now describe these results, including the two applications above, in more detail.

1.3. Our Results in Detail and Techniques

Consider a random oracle $\mathcal{O} : \{0,1\}^* \to \{0,1\}^\lambda$ mapping arbitrary input strings to λ-bit outputs. Sometimes, we also consider restricted domain $\{0,1\}^n$ with fixed input length n, for some polynomial n in λ.

As mentioned above, the goal is to simulate the random oracle $\mathcal{O}$ and advice $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$, using an efficient simulator $\mathcal{S}$ that receives some (non-uniform) advice w — meaning that $\mathcal{S}^{\mathcal{A}}(w)$ acts indistinguishably from $\mathcal{A}^{\mathcal{O}}(z)$ to a bigger environment (see Main Theorem below). We want the computational overhead of the simulation to be as small as possible, in particular, paying close attention to the time overhead, namely $T_{\mathcal{S}^{\mathcal{A}}}$ compared to $T_{\mathcal{A}}$, as well as the additional amount of advice that $\mathcal{S}$ receives, i.e., $|w| - |z|$. For some applications, the memory overhead of $\mathcal{S}^{\mathcal{A}}$ is also important (e.g., for achieving memory-tight reduction or optimal time-space trade-off). Our simulation is also memory-efficient, though we do not go into details on this front in this work.

Our key lemma presents a simple and generic way of using PRF to simulate AI-ROM. A PRF consists of a key distribution $k \xleftarrow{\$} \mathcal{D}_{\text{prf}}(1^{\lambda_{\text{prf}}})$ for different security parameters λ_{prf}, together with an evaluation algorithm $F(k,x) = y$. We require the PRF to have $(2^{\lambda_{\text{prf}}}, \varepsilon)$-security, meaning that for all $2^{\lambda_{\text{prf}}}$-time distinguishers, oracle access to $F(k,\star)$ with a random k is indistinguishable to the random oracle $\mathcal{O}$. We show that the random oracle with S-bit advice $(\mathcal{O}(\star), z \xleftarrow{\$} \mathsf{ai}(\mathcal{O}))$ can be simulated by the PRF and some advice on its key $(F(k,\star), \tilde{z} \xleftarrow{\$} \widetilde{\mathsf{ai}}(k))$, as long as the PRF is using sufficiently large security parameter $\lambda_{\text{prf}} = S + \mathrm{O}(\lambda)$.

Main Lemma (simulating AI-ROM using PRF)**.** *All variables implicitly depends on the security parameter λ.*

Let $\mathsf{dom} \subseteq \{0,1\}^*$ and $\{0,1\}^\lambda$ be the domain and range of the random oracle. Fix an arbitrary $(2^{\lambda_{\text{prf}}}, \varepsilon)$-secure PRF $(\mathcal{D}_{\text{prf}}, F)$ with the same domain and range, and an arbitrary (randomized) advice function ai with S-bit outputs for $S \le 2^\lambda$, there exists a (randomized)[7] advice function $\widetilde{\mathsf{ai}}$ with S-bit outputs and a threshold $\lambda_{\text{prf},0}$, such that:

- *The threshold is upper bounded by $\lambda_{\text{prf},0} \le S + O(\lambda)$ and,*
- *For every 2^λ-time adversary $\mathcal{A}$, and every PRF security parameter $\lambda_{\text{prf}} > \lambda_{\text{prf},0}$,*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}(\star)}(z) = 1] - \Pr[\mathcal{A}^{F(k,\star)}(\tilde{z}) = 1]\right| \le 2^{-\lambda} + \varepsilon,$$

where $\mathcal{O}$ is a randomly sampled oracle, $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$ an oracle-dependent advice, $k \xleftarrow{\$} \mathcal{D}_{\text{prf}}(1^{\lambda_{\text{prf}}})$ a randomly sampled PRF key, and $\tilde{z} \xleftarrow{\$} \widetilde{\mathsf{ai}}(k)$ a key-dependent advice.

The PRF simulation is conceptually simple and its proof crucially relies on the leakage simulation lemma of [15–17]. The idea is viewing $\mathsf{ai}(\mathcal{O})$ as a short leakage of $\mathcal{O}$, which by prior works can be simulated relatively efficiently $h(\mathcal{O})$ in time $T_h = O(2^S \cdot S \cdot 2^{2\lambda} \cdot T_{\mathcal{A}})$, with $2^{-\lambda}$ advantage. Therefore, as long as PRF is ε-secure against this time, we can switch $\mathcal{O}$ to F_k, and view $h(F_k)$ as the new advice $\mathsf{ai}'(k)$ on k. See Section 3 for the formal proof.

Now to simulate AI-ROM for $\mathcal{A}$, the simulator $\mathcal{S}^{\mathcal{A}}$ should simply run $\mathcal{A}^{F(k,\star)(\tilde{z})}$. It can do so if receiving $w = (k, \tilde{z})$ as non-uniform advice. Overhead of the simulation is directly linked to the efficiency of the PRF, in particular, the PRF key length $|k|$ corresponds to the additional amount of non-uniform advice, and the PRF evaluation time determines the time overhead of simulation. Both of these quantities vary with the PRF security parameter λ_{prf}. More precisely...

Main AI-ROM Simulation Theorem. *Let λ, dom, PRF $(\mathcal{D}_{\text{prf}}, F)$, and ai be defined as in the main lemma. There exist a universal simulator $\mathcal{S}$, and a non-uniform advice function nu, such that:*

- *For every 2^λ-time adversary $\mathcal{A}$, and 2^λ-time machine B with binary output,*

$$\Pr[\mathsf{Out}_B\langle B \leftrightarrow \mathcal{A}^{\mathcal{O}}(z)\rangle = 1] - \Pr[\mathsf{Out}_B\langle B \leftrightarrow \mathcal{S}^{\mathcal{A}}(w)\rangle = 1] \le 2^{-\lambda} + \varepsilon\ ,$$

  where $\mathcal{O}$ is a randomly sampled oracle with advice $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$, and w a non-uniform advice from distribution $w \xleftarrow{\$} \mathsf{nu}(\perp)$.
- *The overhead of the simulation is related to the efficiency of the PRF as follows:*

$$\begin{aligned} &\textit{advice overhead:} && |w| - |z| \le |k| + O(\log(S, |k|))\ , \\ &\textit{time overhead:} && T_{\mathcal{S}^{\mathcal{A}}} - T_{\mathcal{A}} \le T_{\mathcal{A},F}\ , \end{aligned}$$

  where $T_{\mathcal{A}}$ is an upper bound on the running time of $\mathcal{A}$, and $|k|$ the PRF key length when using security parameter $\lambda_{\text{prf}} = S + O(\lambda)$, and $T_{\mathcal{A},F}$ is an upper bound on the total time for evaluating the PRF on all oracles calls made by $\mathcal{A}$.

Clearly, more efficient the PRF is leads to tighter AI-ROM simulation. By plugging in different PRF in the above theorem, we obtain different simulation efficiency as summarized in Table 1, which we explain in detail below.

Subexponential PRF and Resolving Unruh's Conjecture. When plugging in a vanilla subexponentially secure PRF, our theorem resolves Unruh's conjecture affirmatively. A subexponentially secure PRF for n-bit inputs is a $(2^{\lambda_{\text{prf}}}, \varepsilon)$-

7. It is necessary for $\widetilde{\mathsf{ai}}$ to sample random PRF key. Otherwise, $\mathcal{A}$ could hardwire the (fixed) PRF key and indistinguishability would fail. Allowing randomized ai is not technically necessary, yet it makes the lemma more convenient to use.

Table 1. COMPARISON OF METHODS FOR DISTINGUISHING.

method	domain	extra advice	time per query
presampling ([11])	$\{0,1\}^n$	$\Theta(n\lambda SQ/\varepsilon)$	$\Theta(n\lambda SQ/\varepsilon)$
presampling ([11])	$\{0,1\}^*$	$\Theta(T\lambda SQ/\varepsilon)$	$\Theta(T\lambda SQ/\varepsilon)$
generic subexp-PRF (§ 3)	$\{0,1\}^n$	$\mathrm{poly}(S,n)$	$\mathrm{poly}(S,n)$
GGM + subexp-OWF	$\{0,1\}^*$	$\mathrm{poly}(S)$	$\lvert x\rvert\,\mathrm{poly}(S)$
GGM + exp-PRG	$\{0,1\}^*$	$O(S)$	$\lvert x\rvert\,\mathrm{poly}(S)$
GGM + DE-PRG	$\{0,1\}^*$	$\tilde{O}(S)$	$\lvert x\rvert\tilde{O}(S)$
Fast PRF (§ 4)	$\{0,1\}^n$	$n^{1+\gamma}S^{1+\gamma}$	(RAM) $n^{1+\gamma}\times$ $\mathrm{poly}(\log(Sn))\tilde{O}(\lambda)$

Here, λ is a security parameter, as well as the output length. S denotes the length of advice adversary gets in AI-ROM, T the adversary runtime, Q the number of oracle queries. GGM is [18], DE-PRG is doubly efficient PRG. The 4th to 6th rows refers to the GGM construction of PRF for unbounded length inputs, using length doubling PRGs with subexponential (line 4) or exponential security (line 5 and 6), and evaluation time depending polynomially (line 4, 5) or quasilinearly (line 6) on the key length. In the last row $\gamma \in (0,1)$ is an arbitrary small constant. All parameters are for achieve a distinguishing advantage of ε or $\varepsilon + 2^{-\lambda}$ between AI-ROM and simulation.

secure PRF with key length $|k| = \mathrm{poly}(\lambda_{\mathrm{prf}}, n)$.[8] PRF evaluation takes polynomial time $\mathrm{poly}(|x|,|k|,\lambda)$ (recall that λ is the output length). Therefore, when $\lambda_{\mathrm{prf}} = \mathrm{O}(S)$, the advice overhead of our simulation is $|k| = \mathrm{poly}(S,n)$, and the time overhead per oracle query is $\mathrm{poly}(S,n)$ (see line 3 in Table 1). This gives simulation with *polynomial* overhead, independent of the *negligible* distinguishing advantage $(\varepsilon + 2^{-\lambda})$. See Section 5 for more details.

Supporting Unrestricted Domain and Linear Dependency on S. Our simulation technique applies to AI-ROM with unrestricted domain $\{0,1\}^*$, using PRF with the same domain. Concretely, one can instantiate such PRF using AES, or the Goldreich–Goldwasser–Micali (GGM) PRF built from a length-doubling PRG [18], where the PRF key is simply a PRG seed $k = \mathrm{sd}$ and evaluation grows in time linearly with the input length $|x|$. Consider using different PRG...

- A subexponentially secure PRG has seed length $|sd| = \mathrm{poly}(\lambda_{\mathrm{prf}})$ and evaluation time $\mathrm{poly}(\lambda_{\mathrm{prf}})$, which can be based on any subexponentially secure OWF. Using it yields simulation with advice overhead $\mathrm{poly}(S)$ and time overhead $|x|\,\mathrm{poly}(S)$ for an oracle query x.
- An exponentially secure PRG has seed length $|sd| = \mathrm{O}(\lambda_{\mathrm{prf}})$ and evaluation time $\mathrm{poly}(\lambda_{\mathrm{prf}})$, which for example can be based on the exponential hardness of DDH. Plugging it reduces the advice overhead to linear $\mathrm{O}(S)$.
- A "doubly efficient" PRG has both short seeds $|sd| = \tilde{\mathrm{O}}(\lambda_{\mathrm{prf}})$ and fast evaluation time $\tilde{\mathrm{O}}(\lambda_{\mathrm{prf}})$. Such a PRG can be constructed from, for example, the exponential hardness of ring learning with rounding (LWR) with polynomial modulus. The double efficiency translates to both small advice overhead $\tilde{\mathrm{O}}(S)$ and small time overhead $|x| \cdot \tilde{\mathrm{O}}(S)$.

8. In the literature, a subexponentially secure PRF often refers to one that achieves security against $2^{|\lambda'|^\beta}$-time adversaries when using security parameter λ'. This is equivalent to our subexponentially secure PRF, by setting λ' to be $\lambda_{\mathrm{prf}}^{1/\beta}$, which causes the key length to grow polynomially with λ_{prf}.

We note that the last instantiation yields improvement over simulation via presampling in all aspects — smaller dependency on all parameters (observing that $\lambda \cdot S = \Omega(S)$ when $S \le 2^\lambda$).

Even Tighter Simulation. So far, simulation time overhead grows with to the advice length S. This means the overall simulation time $T_{\mathcal{S}}$ is at least $T_{\mathcal{A}} \cdot S$. Can we ensure that $T_{\mathcal{S}} \sim T_{\mathcal{A}}$, while keeping $S_{\mathcal{S}} \sim S$? Since the PRF security parameter needs to be large $\lambda_{\mathrm{prf}} = S + \mathrm{O}(\lambda)$. It is impossible to have PRF evaluation time below $|k| > \lambda_{\mathrm{prf}}$ if evaluation reads the entire key. Nevertheless, we show that in the RAM model, it is possible to construct a PRF whose evaluation reads only a small portion of the key. Our construction is based on a variant of Goldreich's PRG assumption [19], and crucially using the polynomial preprocessing technique of [20] that has recently found application in building doubly efficient PIR [21]. For n-bit inputs and λ-bit outputs, our fast has key length $n^{1+\gamma}{\lambda_{\mathrm{prf}}}^{1+\gamma}$ and evaluation time $n^{1+\gamma}\,\mathrm{poly}\log(\lambda_{\mathrm{prf}} \cdot n)\tilde{\mathrm{O}}(\lambda)$, where γ can be set to an arbitrarily small constant in $(0,1]$. Such a PRF enables AI-ROM simulation where the time overhead is $n^{1+\gamma}\,\mathrm{poly}\log(S \cdot n)\tilde{\mathrm{O}}(\lambda)$ (see last line of Table 1). The overall simulation achieves $T_{\mathcal{S}} \sim T_{\mathcal{A}}$ and $S_{\mathcal{S}} \sim S_{\mathcal{A}}$ (modulo $\mathrm{poly}(\lambda)$ and $\mathrm{o}(S)$ factors).

To the best of our knowledge, our construction gives the first PRF where the evaluation time does not grow with the security parameter — efficiency does not "pay" for stronger security. We believe that such a PRF is of independent interests. See Section 1.4 for an overview of the fast PRF construction.

Applications. We apply our AI-ROM simulation technique in two applications: 1) showing that salting generically defeats preprocessing, with a tighter reduction than prior works [11], and 2) constructing the first NIZK (with polynomial time simulation and negligible distinguishing advantage) in the AI-ROM model.

Salting. Salting is a technique that has been widely used in the practice for defeating preprocessing (going back to password hashing [22]). It uses a random oracle $\mathcal{O}'$ (or real-world hash function) to build another random oracle $\mathcal{O}'_r$, by prepending every oracle call with a public string r, referred to as the salt, i.e., $\mathcal{O}'_r(x) = \mathcal{O}'(r,x)$, chosen randomly after preprocessing. One can generically upgrade a scheme P that is secure in the ROM to a scheme P' secure in the AI-ROM, by using P with the salted random oracle $\mathcal{O}'_r$, i.e., $P'^{\mathcal{O}'} = P^{\mathcal{O}'_r}$. This is shown via a reduction (R, ai) satisfying that if an adversary $\mathcal{A}$ with advice $z \xleftarrow{\$} \mathsf{ai}'(\mathcal{O}')$ can break the security of $P'^{\mathcal{O}'}$, the adversary $B = R^{\mathcal{A}}$ with advice $w \xleftarrow{\$} \mathsf{ai}(\perp)$ can break $P^{\mathcal{O}}$ in ROM. Previously, the

presampling techniques were used to build the reduction [11], which internally simulates AI-ROM for $\mathcal{A}$.

As argued before, reduction via presampling is wasteful. So plug-in our PRF-based simulation instead. For instance, we reduce the computational overhead of B over $\mathcal{A}$ from $\Theta(n\lambda SQ/\varepsilon)$ (using presampling) to $\mathrm{poly}(S,n)$ using subexponentially secure PRF, or $n\tilde{\mathrm{O}}(S)$ using doubly efficient PRG (which in turn can be based on ring LWR with polynomial modulus), or even $n^{1+\gamma}\,\mathrm{poly}\log(S,n)$ using fast PRF (based on a variant of Goldreich's PRG). This establishes a reduction where $T_B \sim T_{\mathcal{A}}$ and $S_B \sim S_{\mathcal{A}}$. See more details in Section 6.1.

NIZK with Preprocessing. To showcase the benefits of our tight simulation, by building the first NIZK secure against preprocessing. Here, efficient simulation (of the cheating verifier's view) is the end goal. It is known that NIZK necessarily relies on some trusted set-up, such as the random oracle, common random/reference string, etc. Unconditionally secure NIZK in ROM and computationally secure NIZK in the CRS model from various assumptions are known. However, it is unknown whether they remain secure when the adversaries possess some preprocessed information of the set-up.

Using salting/simulation technique, we show how to start with a NIZK scheme in the ROM with statistical soundness against time-unbounded cheating prover and upgrade its security to the AI-ROM. We generically "salt" the NIZK scheme, by making the honest prover P generate the proof using a salted random oracle $\mathcal{O}(r,\star)$, and send the tuple (r,π) as the new proof. (Correspondingly, the verifier verifies π w.r.t. $\mathcal{O}(r,\star)$.) Since soundness holds against time-unbounded adversaries, additional advice about $\mathcal{O}$ does not enhance the power of the cheating prover, and statistical soundness remains intact. For zero-knowledge, the task is efficiently simulating the view of the cheating verifier V^*, in a way that is negligibly indistinguishable. The view includes the proof (r,π), the oracle $\mathcal{O}$, and advice $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$. Using our salting/simulation technique, we can first show that the interaction between $P^{\mathcal{O}(r,\star)}$ and ${V^*}^{\mathcal{O}}(z)$ can be simulated by the interaction between $P^{\mathcal{O}'}$ and ${V'}^{\mathcal{O}'}$ in the ROM. The attacker V' internally runs V^* by simulating oracle $\mathcal{O}$ as follows. For every call with input (r,x) it returns $\mathcal{O}'(x)$, and for every call (r',x) with $r' \neq r$, it returns $F(k,(r',x))$. V' also feeds V^* the simulated advice $z' \xleftarrow{\$} \mathsf{ai}'(k)$. Now, the simulator $\mathcal{S}$ guaranteed in the ROM can be used to simulate the ZK proof $\tilde{\pi}$ and oracle $\mathcal{O}'$ to V'. Then, the oracle $\mathcal{O}$ that V^* interacts with and the advice can be simulated, combining our PRF simulation technique.

Thanks to the efficiency of our AI-ROM simulation, the resulting zero-knowledge simulator has polynomial overhead, while achieving negligible distinguishing advantage ε. Moreover, more efficient instantiation of the PRF leads to tighter simulation. See Section 6.2 for more details.

A Four-Step Generic Technique. Both our salting and NIZK applications uses a new technique that combines presampling and our PRF simulation (Main Lemma), which allows us to enjoy the best of both techniques. Consider salting, where we aim show that if a scheme $\Pi^{\mathcal{O}}$ is secure against adversary $B^{\mathcal{O}}$ in the ROM, then the salted scheme $\Pi^{\mathcal{O}(r,\star)}$ is secure against adversaries $\mathcal{A}^{\mathcal{O}}(z)$ in the AI-ROM. Applying our PRF simulation lemma directly does not work. It proves that the interaction $\Pi^{\mathcal{O}(r,\star)} \leftrightarrow A^{\mathcal{O}}(z)$ is indistinguishable to an interaction $\Pi^{F_k(r,\star)} \leftrightarrow A^{F_k}(z'(k))$ where the AI-ROM $(\mathcal{O},z)$ is simulated by the PRF $(F_k, z'(k))$. But this does not allow reducing to security in the ROM. What's worse, once the oracle is replaced by PRF, we cannot apply the many useful tricks in ROM, such as lazy sampling, programming, extraction, etc. Instead, we designed the following 4 step technique.

- Start from the a real world game $\Pi^{\mathcal{O}(r,\star)} \leftrightarrow A^{\mathcal{O}}(r, z(\mathcal{O}))$ in AI-ROM.
- *Presampling*: Use presampling to go to the bit-fixing ROM model with table P,

$$\Pi^{\mathcal{O}[P](r,\star)} \leftrightarrow A^{\mathcal{O}[P]}(r, z'(P)),$$
$$\text{where } \mathcal{O}[P](x) = \begin{cases} P[x] & \text{if } x \in P \\ \mathcal{O}(x) & \text{o.w.} \end{cases}$$

  Note the advice z' contains only information of P uncorrelated with $\mathcal{O}$.
- *Re-routing relevant queries* by the fact that the salt r is random, it is unlikely that P contains any input starting with r. In such case, we can re-route salted queries $x = (r,x')$ to an independent oracle $\mathcal{O}^*$ independent of $\mathcal{O}$ and P. This brings us to the game

$$\Pi^{\mathcal{O}'(r,\star)} \leftrightarrow A^{\mathcal{O}'}(r, z'(P)),$$
$$\text{where } \mathcal{O}'(x) = \begin{cases} \mathcal{O}^*(x') & \text{if } x = (r,x') \\ \mathcal{O}[P](x) & \text{o.w. } x = (r' \neq r, x') \end{cases}$$

  In other words, all salted queries are "freed".
- *Reverse Presampling:* apply the presampling technique reversely to turn the BF-ROM $\mathcal{O}[P]$ and advice $z'(P)$ back to AI-ROM $(\mathcal{O},z)$, while keeping all salted queries answered by independent oracle $\mathcal{O}'$.

$$\Pi^{\mathcal{O}^*} \leftrightarrow A^{\mathcal{O}''}(r, z(\mathcal{O})),$$
$$\text{where } \mathcal{O}''(x) = \begin{cases} \mathcal{O}^*(x') & \text{if } x = (r,x') \\ \mathcal{O}(x) & \text{o.w. } x = (r' \neq r, x') \end{cases}$$

  At this point, all security relevant queries are answered by $\mathcal{O}^*$, while A additionally has some "meaningless" interaction with $\mathcal{O}$ and has preprocessed information z about it.
- *PRF simulation* by our main lemma, the interaction with $(\mathcal{O},z)$ can be simulated by PRF $(F_k, z''(k))$, allowing us to move to the game

$$\Pi^{\mathcal{O}^*} \leftrightarrow A^{\mathcal{O}'''}(r, z''(k)),$$
$$\text{where } \mathcal{O}'''(x) = \begin{cases} \mathcal{O}^*(x') & \text{if } x = (r,x') \\ F_k(x) & \text{o.w. } x = (r' \neq r, x') \end{cases}$$

Here the adversary $A^{\mathcal{O}'''}(r, z''(k))$ can be emulated by an adversary $B^{\mathcal{O}^*}$ given (k, z'') as non-uniform advice. We can now rely on ROM security, and enjoy the fact that by our PRF simulation $T_B \sim T_{\mathcal{A}}$ and $S_B \sim S_{\mathcal{A}}$.

We remark that our 4-step technique is relatively generic: The key idea is re-routing all security-relevant queries to an independent oracle $\mathcal{O}^*$ for which no preprocessed information is available, and then simulate the rest queries and advice using our PRF simulation. The former is achieved using presampling, and besides from salting, there might be other techniques and applications such that security relevant queries can be re-routed without salting. In this step, all ROM related tricks can be applied. The latter enjoys the efficiency of our PRF simulation. We believe that this technique will find further applications.

1.4. Constructing Fast Pseudorandom Functions

We now come to the question of designing fast pseudorandom functions. Just to be formal, we consider PRF families $F : \{0,1\}^{\ell} \times \{0,1\}^n \to \{0,1\}$ with key length $\ell = \ell(\lambda)$ and input length $n = n(\lambda)$ where the input length is thought of as a varying parameter that is a polynomial in the security parameter λ and the key length is allowed to grow polynomially in both λ and n. We are interested in two properties:

- The PRF is secure against adversaries that take 2^{λ} time adversary,
- The time to evaluate the PRF on any input $x \in \{0,1\}^n$ is quite fast in the RAM model. Namely, each query can be evaluated in time $n^{\mathrm{O}(1)} \operatorname{poly}(\log n, \log \lambda)$. In other words, the evaluation time is effectively a function of the input length and not a function of the level of security. We achieve evaluation time of $n^{1+\gamma} \operatorname{poly}(\log n, \log \lambda)$ for arbitrarily small constant $\gamma > 0$.

We observe that none of the prior constructions of PRF are known to achieve security independent evaluation times in the RAM model.

Leveraging Sparsity of Goldreich's Function. It is clear that we are looking to build functions that accesses $\tilde{O}(n)$ bits of the key on every query as the evaluation must run in that time. Therefore, it is imperative that we look for sparsity related hard problems.

In fact, it was already observed in [23] that Goldreich's function [19] when instantiated with super-constant locality give rise to extremely efficient PRF. We discuss the intuition in a bit but take a slight detour familiarizing the reader with Goldreich's one way function. A familiar reader can skip to the next highlighted paragraph.

Detour: Goldreich's One-Way Function/PRG. Goldreich's one-way function candidate $f_{G,P} : \{0,1\}^{\ell} \to \{0,1\}^m$ is indexed with a local Boolean predicate $P : \{0,1\}^L \to \{0,1\}$ and a hypergraph $G = (S_1, \ldots, S_m)$ with m hyperedges, where each hyperedge $S_i \subset [\ell]$ of size L. Then $f_{G,P}(x) = (y_1, \ldots, y_m)$ where each y_i is computed by applying the predicate P on the bits given by the set S_i (denoted as $y_i = P(x|_{S_i})$). Smaller the locality, more efficiently computable the function is. In particular, if P is L local then given S_i each output bit y_i can be computed in $\tilde{O}(L)$ time. How small can L be? If one is interested in polynomial stretch, one can work with $L \geq 5$ however the function is trivially not pseudorandom when $m \geq n^L$ as then the equations might repeat. This means that with constant locality we cannot hope for a stretch beyond a polynomial. For PRF one might need to support super-polynomial or subexponential stretch. Therefore one has to work with a locality L that is polynomial in the input length. This regime was used by [23] to build an efficient PRF.

Based on a long history [19,24–36] of study culminating in [36], for appropriately chosen hypergraph G satisfying certain expansion criteria and predicate P properly, it is widely believed that such functions should be pseudorandom for $m = n^{\Omega(L)}$. If the predicate has large resiliency and large rational degree (see Section 4.3 for definitions of these properties) and if the hypergraph is chosen so that is $(t, 0.99)$ expanding for a large t (this property says that union of any hyperedges $\{S_i\}_{i \in I}$ for any set $I \subset [m]$ of size t, the size of the union is at least $0.99 \cdot t \cdot L$) then the function $f_{G,P}(x)$ can be conjectured to be psuedorandom against algorithms that run in time $2^{\tilde{O}(t)}$. This expansion property is satisfied by a random hypergraph with high probability however the conjecture holds for arbitrary graphs with expansion. This was backed up by provable lower bounds against various classes of attacks such as sum-of-squares, linear tests and polynomial calculus [36]. The properties regarding resiliency and rational degree is trivially satisfied by the predicate $\mathsf{XOR\text{-}MAJ}_{L'}$ that takes as input $L = 2 \cdot L'$ inputs and computes:

$$P(x_1, \ldots x_{L'}, x_{L'+1}, \ldots, x_L) = x_1 \oplus \ldots x_{L'} \oplus \mathsf{Maj}(x_{L'+1}, \ldots, x_L)$$

This predicate satisfies a resiliency of $L/2$ and rational degree $L/4$. Our assumption therefore can be stated as:

Assumption 1 (Goldreich's Assumption)**.** *Let $n \in \mathbb{N}$ be the input length, $L = 2L' \ll \ell$ be the locality (which should either been seen as some polynomial in seed length ℓ) and let P be the predicate $\mathsf{XOR\text{-}MAJ}_{L'}$. Let $\mathcal{G} = \{G_\ell\}_\ell$ be a family of L regular hypergraph with m hyperedges such that the G_ℓ is $(t(\ell), 0.99)$ expanding. Then it holds that for any p.p.t. adversary running in time $2^{\tilde{O}(t)}$ we have that:*

$$\{f_{G,P}(x) : x \leftarrow \{0,1\}^{\ell}\} \approx_c \{y : y \leftarrow \{0,1\}^m\}$$

Above, the computational indistinguishability holds with advantage bounded by negligible in n.

Efficient Weak PRF. As shown previously in [23], we now describe how one could build highly efficient weak PRF

from Goldreich's PRG (these are PRF where the security definition only allows queries that are random). The high-level idea is that the PRF key consists of a random seed $s \leftarrow \{0,1\}^\ell$ for Goldreich's PRG. On input a random input x, it is interpretted as a random L sized set $S_x \subset [\ell]$ of size L. The evaluation simply outputs a bit $y_x = P(s|_{S_x})$. Such a PRF can be evaluated in time $\tilde{O}(L)$. The security holds because for various queries $S_{x_1}, \ldots, S_{x_q}$ are distributed as random hyperedges and a random graph is going to satisfy the expansion properties needed for Assumption 1 if the parameters and number of queries satisfy a technical condition.

We now ask how does one set the parameters and what's the best possible L (as the evaluation time is nearly linear in L) we can set to support large number of queries and to obtain strong levels of security. Remember, we need to allow $m = 2^n$ queries and obtain security against 2^λ time adversaries where n is a polynomial in λ.

As mentioned previously, we cannot hope for security once the number of samples exceed ℓ^L, so this must be set so that $\ell^L \gg 2^n$. This means that L must be at least n upto polylog factors. Indeed, we will set L to be a large constant multiple of n. For security against 2^λ time adversaries that makes $m = 2^n$ queries we would need the hypergraph $(S_{x_1}, \ldots, S_{x_m})$ to be $(t, 0.99)$ expanding where $t = \Omega(\lambda)$. A routine calculation shows that this can be achieved by setting the key size to be large enough polynomial. In particular, setting (say) $\ell = n\lambda^2$, a random hypergraph with overwhelming probability be $(t, 0.99)$ expanding. This will yield a weak PRF that can be computed in time $\tilde{O}(n)$ time in the RAM model with key size $\mathrm{O}(n\lambda^2)$ and based on Assumption 1 is secure against adversaries of 2^λ size.

Strong PRF via t-Wise Independence. While fast weak PRF can be constructed using Goldreich's PRG readily it is unclear if strong PRF can be built this easily. Indeed, in the previous construction if the adversary picks the L regular hyperedges S_x arbitrarily the induced graph may not be an expander anymore. Weak PRF get around this issue by mandating that the edges are chosen at random.

The issue therefore is how can we have adversarially chosen hyperedges $\{S_x\}_{x\in[m]}$ form an expander? Indeed, this will definitely not be the case.

Our first observation, which is folklore, is that to generate a $(t, 0.99)$ expander with high probability the hyperedges may not neccesarily have to be chosen at random. In fact, if the sets $\{S_x\}_{x\in[m]}$ are chosen so that they form a t wise independent distribution instead of being randomly chosen then it can be shown that with high probability even the hyperedges sampled this way will also form an expander. Thus, a seemingly significantly weaker property suffices. This is the approach we will follow. Note that similar approach was followed by [23], however their construction yielded PRF that are linear time in the key-length. We show how to avoid this dependence on the key-length altogether.

How does one generate a mapping that maps an input $x \in [m]$ to a hyperedge S_x such that the hypergraph $\{S_x\}_{x\in[m]}$ forms a t wise independent distribution? To do this, as a part of the PRF key we sample a random degree t polynomial of the form $h(z) = a_0 + a_1 z + \ldots + a_t z^t \mod p$ for randomly chosen coefficients over $\mathbb{Z}_p$ for an appropriately chosen prime p barely more than $N = \binom{\ell}{L}$ (which happens to be the total possible number of hyperedges of size L). Since the polynomial is a random degree t polynomial the distribution formed by $\{h(x)\}_{x\in[m]}$ is t wise independent. The idea is that on every input $x \in [m]$, we will compute $h(x) \mod p$ first and then map it to a set S_x (it is well known that any string $\sigma \in [1, N]$ for $N = \binom{\ell}{L}$ can be mapped injectively to a set/hyperedge $S \subset [\ell]$ of size L in time $L\,\mathrm{poly}(\log L, \log \ell)$). There are some technical issues such as the number N being not a prime and therefore an evaluation $h(x)$ may not be in the set $[1, N]$ and so the string $h(x)$ might not be map to one of the $N = \binom{\ell}{L}$ sets S_x. We solve this issue by sampling a prime p this very close to N so that the chance of this odd event happening is very small. For the rest of exposition, assume that p is exactly equal to N.

Fast Evaluation via Polynomial Preprocessing. We now analyze the running time of the PRF above. Since the PRG is L local, once the set S_x is derived the PRG can be computed in $\tilde{O}(L)$ time in the RAM model which is our desired complexity. To compute the set S_x for an $x \in [2^n]$ we compute the polynomial $h(x)$. Computing a random degree t polynomial it takes at least $t \cdot \log p$ time which is already prohibitive as we need to set $t > \lambda$ so that the PRF is secure against 2^λ sized adversaries. How can we remove the dependence on λ and replace it with a dependence on n?

We come up with a simple yet powerful idea to address this issue. Namely, we rely on a powerful tool developed in the beautiful work by Kedalaya and Umans [20]. Informally the result says that one could preprocess in polynomial time from the description of a degree t univariate polynomial over $\mathbb{Z}_p$ a data structure of polynomial size such that from there on, in the RAM model the evaluation of the polynomial at any point $x \in \mathbb{Z}_p$ can be computed in time $\mathrm{O}(\log^{1+\gamma} p \cdot \mathrm{poly}(\log t))$ for arbitrarily small constant $\gamma > 0$. We state the theorem below:

Theorem 7 (reinterpreted from Theorem 5.1 in [20])**.** *Let $\mathbb{Z}_p$ be a field of prime order p. Let $f(x)$ be a polynomial of degree t in $\mathbb{Z}_p[x]$. Let $\gamma > 0$ be arbitrary constant. For large enough t, one can compute in time $\mathrm{O}(t^{1+\gamma} \log^{1+\gamma} p)$ a data structure with the property that given any $\alpha \in \mathbb{Z}_p$, $f(\alpha)$ can be computed in time $\mathrm{O}(\log^{1+\gamma} p \cdot \mathrm{poly}(\log t))$ in the RAM model.*

The fast computation time of sets therefore follow directly from preprocessing our polynomial using this result. Since our degree $t \approx \lambda$ and $p \approx \binom{\ell}{L}$ we have that the online evaluation of the polynomial will take time $L^{1+\gamma}\,\mathrm{poly}(\log \lambda, \log n)$. Since $L = \mathrm{O}(n)$ our evaluation time is $n^{1+\gamma}\,\mathrm{poly}(\log \lambda, \log n)$ where one can choose $\gamma > 0$ to be arbitrarily small constant. This concludes finally the claim that our PRF can be evaluated in time $\tilde{\mathrm{O}}(n^{1+\gamma})$ for arbitrarily small constant $\gamma > 0$. Recently, such techniques played important role in constructing doubly efficient PIR schemes from ring LWE [21].

Table 2. SELECT SYMBOLS USED IN THIS WORK.

symbol(s)	meaning
λ	security parameter
C, C_X	absolute constant, X-dependent constant
nu, w, M	standard-model non-uniformity, string, length
ai, z, S	oracle-dependent auxiliary input, string, length
M, T	(concrete) size, (concrete or asymptotic) time
A, E, ε	algorithm, experiment, advantage
n, m	input/output lengths of random oracle
Q	number of queries
L, N	seed length, output length ($N = 2^n$)
d, P	locality, predicate
t, q	degree of independence, modulus
$k, \overline{k}$	preprocessed key, length bound

Relation with [23]. A reader might recall that [23] also constructed fast PRF, however their PRF does not yield efficiency in the sense we want. The goal of that work was to construct PRF with key sizes n, output length n (as opposed to one bit in our case) but those that take $\tilde{\mathrm{O}}(n)$ time to evaluate. They start by constructing a weak PRF, and convert it to a standard PRF similarly to us by instantiating a graph sampler using a independence generator that uses ring-based computation by Miles and Viola [37] (this offers efficiency advantages due to FFT over rings). However, their efficiency is only achieved in an amortized sense, that is, when the output length of the PRG is long enough. The output length itself is linear in the key length and so each PRF output block could take time proportional to the key/output length. Since the size of the key is polynomial in the security parameter, the running time is strongly dependent on the security parameter. Our PRF truly achieves security independent evaluation time in the RAM model.

2. Preliminaries

The security parameter is λ. For $N \in \mathbb{N}$, we write $[N]$ for $\{1, \ldots, N\}$. For $n \in \mathbb{N}$, the set $\{0,1\}^{\leq n}$ consists of all bit-strings of length at most n, and $\{0,1\}^*$ is the set of all bit-strings. Given $p \in \mathbb{N}_{\geq 2}$, we denote by $\mathbb{Z}_p$ the integers modulo p. Logarithms without a base are natural by default. A natural number $n \in \mathbb{N}$ has an easy-to-parse prefix-free encoding $1^{\lceil \log_2(n+1) \rceil} 0 b_{\lceil \log_2(n+1) \rceil - 1} \cdots b_1 b_0$, where $b_{\lceil \log_2(n+1) \rceil - 1} \cdots b_1 b_0$ is the binary form of n without a leading zero. The encoding length is $(2\lceil \log_2(n+1) \rceil + 1)$.

Let X be quantified objects. Every appearance of C_X is a new non-negative number (constant) that depends on X (but not any other object) and makes a proposition or a proof correct. For example, C_{ram} depends on the exact model of RAM programs, $C_{\text{ckt,prf}}$ depends on the exact model of circuits (e.g., AND/OR/NOT or just NAND) and the PRF being used, and C without any subscript is an absolute constant.

Symbols. Table 2 explains select symbols used throughout the paper.

2.1. Auxiliary-Input Random Oracle Model

We consider concrete and asymptotic security against random-access machines (RAM) and circuits, randomized and possibly connected to oracles. Throughout the paper, we fix a universal oracle-aided RAM, which interprets all RAM *programs*. For randomness, at each step, one random bit is provided, which can be saved and retrieved later using random access. Only oracles are put as superscripts and the notation $A(x)$ does not necessitate scanning x in full. We also assume that any input can be hardwired into programs with small overhead. Precisely speaking, there exists a (uniform) linear-time deterministic Turing machine that given a two-input program A and an input z, outputs a single-input program for $A_z(x) = A(z, x)$ that is no more than $(|A| + |z| + C_{\text{ram}} \cdot \log(|A| + |z| + 2))$ bits long and runs in the time of A.[9] We also fix a model of circuits and use RAM programs for generation of circuits.

Definition 1 (advice, algorithm, adversary, experiment, indistinguishability)**.** Denote by X some object about which adversaries get advice. A *concrete X-advice* is a randomized procedure that outputs a bit-string of fixed length given X. An *asymptotic X-advice* is a sequence of concrete X-advices, or equivalently, a randomized procedure that outputs a bit-string of length determined by λ given (λ, X). An asymptotic $\perp$-advice captures standard-model non-uniformity and is denoted by nu. An $\mathcal{O}$-advice, with $\mathcal{O}$ being the random oracle (Definition 3), captures the so-called "auxiliary input" and is denoted by ai. The outputs of nu, ai are w, z. The length of (asymptotic) w is $M(\lambda)$, and that of z is S or $S(\lambda)$.

A *concrete algorithm* A is a distribution[10] over either RAM programs or circuits with uniformly bounded *size* M[11] and *time* T. An *asymptotic algorithm* A is either a RAM program whose time is bounded for each λ or an always-halting Turing machine that generates circuits.[12] For asymptotic circuit algorithms, its *time* $T(\lambda)$ is the maximum time to

9. Suppose programs are written in a real-world-like instruction set and A receives each input as a pair of registers (location and length), then A_z can be storing the location and the length of z in the two registers that A regards as z (i.e., initializing z), followed by the code of A, followed by z itself. If we include the time to initialize each input (e.g., x) in the total time, then the (internal) initialization of z in A_z replaces the (external) initialization of z in A, and the total time does not change.

10. A concrete algorithm has *two* phases of randomness: first, a random RAM program or circuit is sampled from this distribution; then, the RAM program or circuit runs with random bits. The first stage essentially permits randomized standard-model non-uniformity, necessary to formulate simulators reliant on randomness that is not efficiently sampleable. The second stage is preserved primarily for RAM, so that randomness (whose length could be as large as the time) does not "creep into" non-uniformity (its size could be much smaller than time) for accurate accounting.

11. There is no notion of "amount" of standard-model non-uniformity (w) for a concrete algorithm, which only works for a specific value of λ. We regard it as being entirely non-uniform, hence the overloading of M.

12. The machine generating circuits takes as input 1^λ and potentially some advice string. Instead of simply letting the circuits be given, by separating the advice and the efficient circuit generation procedure, our formulation enables accurate accounting of the amount of advice for an asymptotic circuit algorithm, which might be much smaller than its circuit size.

generate a circuit on λ; for asymptotic RAM algorithms, its *time* follows the usual definition.

The term *adversary* loosely means algorithms with or without either of nu, ai, and *experiment*, algorithms that interact with adversaries in black box. For advantage bound ε and adversary class $\mathcal{A}$, *indistinguishability* is denoted by "$\overset{\varepsilon}{\approx}_{\mathcal{A}}$". In *concrete indistinguishability*, $\varepsilon \in [0, +\infty)$ and the inequality must hold for all (concrete) adversaries in $\mathcal{A}$. In *asymptotic indistinguishability*, $\varepsilon : \mathbb{N} \to [0, +\infty)$ and for each (asymptotic) adversary in $\mathcal{A}$, the inequality must hold for all but finitely many λ's.

We talk about M (length of nu) of an asymptotic adversary, but never its size. Once upper bounds are put on M, T, there are only finitely many derandomized concrete algorithms — this is important when we invoke the leakage simulation lemma.

A security property often involves multiple parameters. It is convenient to unify them into a single one and talk about "bits of security".

Definition 2 (strong bit level of security)**.** Let $\lambda' \in \mathbb{N}$. The indistinguishability of two concrete experiments is *strongly λ'-bit* if the advantage is bounded by $2^{-\lambda'}$ for all adversary whose complexities are bounded by $2^{\lambda'}$.

Let $\lambda' : \mathbb{N} \to \mathbb{N}$ be a function. Two asymptotic experiments has *strongly λ'-bit indistinguishability* if for all adversary whose complexities are bounded by $2^{\lambda'(\lambda)}$ for all but finitely many λ's, the advantage is bounded by $2^{-\lambda'(\lambda)}$ for all but finitely many λ's.

The notion of adversary's complexities is contextual. They can be size, time, number of queries, among others. We intentionally say "strong" in Definition 2 to differentiate it from the folklore version and that in [38]. A popular definition of λ-bit security says that the adversary's complexities must be close to 2^{λ} for the advantage to be $\Omega(1)$. It only tells us when security fails, not when security holds as informed by Definition 2. The version in [38] is more delicate yet too complex for the purpose of this paper. As an example of Definition 2, recall that the one-wayness advantage of a Q-query adversary against a random oracle $\{0,1\}^{2\lambda+2} \to \{0,1\}^{2\lambda+2}$ (without auxiliary input) is bounded by $(Q+2)/2^{2\lambda+2}$, so it has λ strong bits of security.

Definition 3 (AI-ROM)**.** Let dom be a function mapping λ to subsets of $\{0,1\}^*$ (subject to Definition 4) and $m : \mathbb{N} \to \mathbb{N}_+$ a polynomially bounded function. In the *auxiliary-input random oracle model* of type $\mathsf{dom} \to \{0,1\}^m$, on λ, every algorithm is given oracle access to a random function $\mathcal{O} : \mathsf{dom}(\lambda) \to \{0,1\}^{m(\lambda)}$ and adversaries get $z \xleftarrow{\$} \mathsf{ai}(\lambda, \mathcal{O})$ as input (Definition 1). One can also consider AI-ROM for a fixed λ with concrete adversaries and $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$.

Definition 4 (possible domains of AI-ROM)**.** An AI-ROM must use one of the following kinds of domains:

- *fixed-length domain*, $\mathsf{dom}(\lambda) = \{0,1\}^{n(\lambda)}$ for a polynomially bounded $n : \mathbb{N} \to \mathbb{N}$;
- *(bounded-)variable-length domain*, $\mathsf{dom}(\lambda) = \{0,1\}^{\leq n(\lambda)}$ for some $n : \mathbb{N} \to \mathbb{N}$ such that $\log(n+1)$ is polynomially bounded;
- *full domain*, $\mathsf{dom}(\lambda) = \{0,1\}^*$.

A random oracle of type $\{0,1\}^n \to \{0,1\}^m$ can be implemented by (m calls to) one of type $\{0,1\}^{n+\lceil \log_2 m \rceil} \to \{0,1\}$.

2.2. Leakage Simulation

We rely on the leakage simulation lemma of [16].

Lemma 1 (generalized from [16; Theorem 2]; ¶)**.** *There exists a RAM program h subject to the following conditions. Given*

- *a non-empty set $\mathcal{X}$, advice length $S \in \mathbb{N}$, desired error $0 < \varepsilon \leq 1$,*
- *a set $\mathcal{F}$ of measurable functions $\mathcal{X} \times \{0,1\}^S \to \{0,1\}$ (distinguishers), and*
- *a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}^S$,*

there exist

- *a list $\mathcal{L}$ of no more than $C \cdot S\varepsilon^{-2}$ functions from $\mathcal{F}$ and*
- *a string z' of no more than $C \cdot (S\varepsilon^{-2} + (S + \log \varepsilon^{-1})^2 \varepsilon^{-1})$ bits*

such that

- *$(X, Z) \overset{\varepsilon}{\approx}_{\mathcal{F}} (X, \widetilde{Z})$, where $(X, Z) \xleftarrow{\$} \mathcal{D}$ and $\widetilde{Z} \xleftarrow{\$} h^{X, \mathcal{L}}(z')$ and*
- *$h^{x, \mathcal{L}}(z')$ is computed by calling $f(x, z)$ for each $f \in \mathcal{L}$ in its advice and each $z \in \{0,1\}^S$ plus some other processing in no more than $C_{\text{ram}} \cdot 2^S (S+1) \varepsilon^{-2}$ RAM time.*

Alternatively, $h^{x, \mathcal{L}}(z')$ can be computed by $\mathcal{L}^x$-aided circuits, which make exactly $2^S |\mathcal{L}|$ oracle calls and can be generated in time $C_{\text{ckt}} \cdot 2^S (S^2 + 1) \varepsilon^{-2}$.[13]

In our version, $\mathcal{X}$ can be a set of arbitrary cardinality and X is explicitly allowed to be provided as an oracle. It enables us to easily handle "monolithic" random oracles, e.g., $\{0,1\}^* \to \{0,1\}$, whose truth tables reside in a set of continuum cardinality. For completeness, we reprove it in the full version [39].

We remark that considering infinite $\mathcal{X}$ is a matter of presentation, not substance. For example, for monolithic random oracles, a time-bounded adversary cannot query them on an input longer than the time bound, hence we can truncate their truth tables to a suitable finite size and apply the version in [16]. Nevertheless, our exposition removes the doubt whether the simulator's efficiency could degrade with the encoding length of elements in $\mathcal{X}$ — it cannot, inferred from the fact that $\mathcal{X}$ could be infinite!

13. The oracle-aided circuit can be generated by a fixed RAM program taking z' as input. The squaring of S does not make our version worse than [16]. In the circuit model, functions in $\mathcal{F}$ are often also represented by circuits, so each is necessarily of size at least S since it has to read the entire z. Once we replace the oracle gates by their circuits, the oracle calls become the real bottleneck and the overall efficiency is the same as in [16].

2.3. Exponentially Secure Pseudorandom Functions

Exponentially secure PRF is a central tool used in this work.

Definition 5 (PRF)**.** Let InOut be a set whose elements are of the form $(\lambda, \mathsf{dom}, m)$, where $\lambda \in \mathbb{N}$, the domain dom is a subset of $\{0,1\}^*$, and $m \in \mathbb{N}_+$ is the output length. A *pseudorandom function for* InOut consists of two ingredients.

- The *key distribution* $D = \{D_{\lambda,\mathsf{dom},m}\}_{(\lambda,\mathsf{dom},m)\in\mathsf{InOut}}$ is a family (indexed by InOut) of distributions over $\{0,1\}^*$.
- The *evaluation algorithm* $\mathsf{Eval}(k, x)$ is deterministic and efficient. It takes as input k drawn from $D_{\lambda,\mathsf{dom},m}$ and $x \in \mathsf{dom}$, and outputs $y \in \{0,1\}^m$.

The key length must be polynomially bounded. There exists a polynomial $\overline{k}$ such that for all $(\lambda, \mathsf{dom}, m) \in \mathsf{InOut}$, it holds that

$$\Pr\left[\, k \xleftarrow{\$} D_{\lambda,\mathsf{dom},m} \;:\; |k| \le \overline{k}(\lambda, |\mathsf{desc}(\mathsf{dom})|, m) \,\right] = 1,$$

where $|\mathsf{desc}(\mathsf{dom})|$ is the encoding length of dom in Definition 6.

The key distribution need *not* be efficiently sampleable. Our PRF with fast evaluation (Eval running in time sublinear in λ) relies on this relaxation to pick an expander for exponential security.

Definition 6 (possible domains of PRF)**.** A PRF must use one of the possible domains of AI-ROM (Definition 4) and they are encoded as follows:

- *fixed-length domain*, $\mathsf{dom} = \{0,1\}^n$, encoded as $\mathsf{desc}(\{0,1\}^n) = 1^n$;
- *(bounded-)variable-length domain*, $\mathsf{dom} = \{0,1\}^{\le n}$, encoded as $\mathsf{desc}(\mathsf{dom}) = n$ in binary with exactly one leading zero;
- *full domain*, $\mathsf{dom} = \{0,1\}^*$, encoded as a distinguished, fixed bit-string (e.g., $\mathsf{desc}(\{0,1\}^*) = 10$ to be different from the other kinds).

In our applications, we will tune the λ of PRF while keeping dom, m fixed, which are determined by the type of AI-ROM. The following definition captures this by allowing the adversary to choose dom, m, possibly independently of λ.

Definition 7 (exponential security of PRF)**.** Let $\varepsilon : \mathbb{N} \to [0, +\infty)$ be negligible. A PRF is $(2^\lambda, \varepsilon)$-*secure* (in a particular model, circuits or RAM) if for all λ,[14] it holds that $E_{\mathrm{prf},\lambda} \overset{\varepsilon(\lambda)}{\approx}_{\mathcal{A}_\lambda} E_{\$,\lambda}$, where $\mathcal{A}_\lambda$ is the set of concrete adversaries of size and time at most 2^λ, and $E_{\mathrm{prf},\lambda}$ and $E_{\$,\lambda}$ with $A \in \mathcal{A}_\lambda$ work as follows.

- **Setup.** Launch $A()$[15] and receive from it $(\mathsf{dom}, 1^m)$ such that $(\lambda, \mathsf{dom}, m) \in \mathsf{InOut}$ with dom encoded as specified in Definition 6. In $E_{\mathrm{prf},\lambda}$, sample $k \xleftarrow{\$} D_{\lambda,\mathsf{dom},m}$. In $E_{\$,\lambda}$, sample a random function $\mathcal{O} : \mathsf{dom} \to \{0,1\}^m$.
- **Challenge.** Let A continue to run as either $A^{\mathsf{Eval}(k,\cdot)}$ or $A^{\mathcal{O}}$.
- **Guess.** A outputs a single bit, which is the output of the experiment.

We remark that our definition includes the usual version of subexponential security, e.g., security against 2^λ-size/time adversary with λ^2-bit key. In our formulation, λ is the desired security level (in terms of size and time, but not necessarily advantage) and the key length is chosen according to complexity leveraging.

On the other hand, as security is required against 2^λ-size/time adversaries, it can be shown that $|k|$ must be at least roughly λ. Suppose for simplicity that $|k|$ is not randomized, $|\mathsf{desc}(\mathsf{dom})|, m \le \lambda$, $|\mathsf{dom}| > |k|$, $\varepsilon < \frac{1}{2}$. Consider the simple attack of enumerating all strings of $|k|$ bits as the candidate PRF keys, evaluating the PRF with each key on $(|k|+1)$ fixed points, and comparing them with the oracle responses to see whether the oracle is possibly the PRF. Both size and time of the attack are bounded by $2^{|k|}(|k|+1)\,\mathrm{poly}(\lambda)$, and it has advantage at least $\frac{1}{2}$, Therefore, $2^{|k|}(|k|+1)\,\mathrm{poly}(\lambda) \ge 2^\lambda$, which implies $|k| \ge \lambda - C_{\mathrm{model,prf}} \cdot \log(\lambda+2)$. Similarly, any circuit of Eval that handles at least $(\lambda+1)$ inputs has to be of size $\Omega(\lambda/\log\lambda)$.

3. Simulating AI-ROM Using PRF

We present our theorem of simulating AI-ROM using any exponentially secure PRF.

Theorem 2 (concrete simulation; ¶)**.** *For each model (either RAM or circuits), there exists a function*

$$\begin{aligned}&\lambda_{\mathrm{prf},0}(\mathsf{dom}, m, S, M, T, \varepsilon_{\mathrm{leak}})\\ &\le S + C_{\mathrm{model}} \cdot \log \frac{|\mathsf{desc}(\mathsf{dom})| + m + S + M + T + 1}{\varepsilon_{\mathrm{leak}}}\end{aligned}$$

such that for all PRF (D, Eval) *with* $(2^\lambda, \varepsilon_{\mathrm{prf}})$*-security, AI-ROM of type* $\mathsf{dom} \to \{0,1\}^m$*, $\mathcal{O}$-advice* ai *of length S, upper bounds M, T of adversary size and time, leakage simulation error* $0 < \varepsilon_{\mathrm{leak}} \le 1$*, complexity-leveraged* $\lambda_{\mathrm{prf}} \ge \lambda_{\mathrm{prf},0}(\mathsf{dom}, m, S, M, T, \varepsilon_{\mathrm{leak}})$ *such that* $(\lambda_{\mathrm{prf}}, \mathsf{dom}, m) \in \mathsf{InOut}$*, there exists a k-advice* $\widetilde{\mathsf{ai}}$ *of length S so that* $E_{\mathrm{airom}} \overset{\varepsilon}{\approx}_{M,T} E_{\mathrm{prf}}$ *(concrete RAM or circuits) with* $\varepsilon = \varepsilon_{\mathrm{leak}} + \varepsilon_{\mathrm{prf}}(\lambda_{\mathrm{prf}})$*, where the experiments work as follows with A that outputs a single bit.*

- *In* E_{airom}*, sample a random function* $\mathcal{O} : \mathsf{dom} \to \{0,1\}^m$ *then* $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$*. Run and output* $A^{\mathcal{O}}(z)$*.*
- *In* E_{prf}*, sample* $k \xleftarrow{\$} D_{\lambda_{\mathrm{prf}},\mathsf{dom},m}$ *then* $z \xleftarrow{\$} \widetilde{\mathsf{ai}}(k)$*. Run and output* $A^{\mathsf{Eval}(k,\cdot)}(z)$*.*

14. This is not an asymptotic indistinguishability, as the inequality must hold even for small λ's.

15. This is a slight abuse of notation. Here, A represents a sample from the distribution of circuits or RAM programs (also called A).

In the RAM model, with A regarded as an oracle, E_{prf} is a concrete algorithm with[16]

$$\begin{aligned} M_E &\le M_{\mathrm{ram},\mathsf{Eval}} + S + |k| + C_{\mathrm{ram}} \cdot \log(M_{\mathrm{ram},\mathsf{Eval}} + S + |k| + 2), \\ T_E &\le Q_A \cdot T_{\mathrm{ram},\mathsf{Eval}} \quad + C_{\mathrm{ram}} \cdot \log(M_{\mathrm{ram},\mathsf{Eval}} + S + |k| + 2), \end{aligned}$$

where Q_A is the maximum number of queries made by A. In the two models (RAM and circuits), $E_{\mathrm{prf}}(A)$ is a concrete adversary with

$$\begin{aligned} \text{(RAM)} \quad M' &\le M_A + M_{\mathrm{ram},\mathsf{Eval}} + S + |k| \\ &\quad + C_{\mathrm{ram}} \cdot \log(M_A + M_{\mathrm{ram},\mathsf{Eval}} + S + |k| + 2), \\ T' &\le T_A + Q_A \cdot T_{\mathrm{ram},\mathsf{Eval}} \\ &\quad + C_{\mathrm{ram}} \cdot \log(M_A + M_{\mathrm{ram},\mathsf{Eval}} + S + |k| + 2), \\ \text{(circuits)} \quad M' &= T' = M_A + Q_A \cdot M_{\mathrm{ckt},\mathsf{Eval}}. \end{aligned}$$

Theorem 3 (asymptotic simulation; ¶)**.** *The asymptotic version $E_{\mathrm{airom}} \overset{\varepsilon}{\approx}_{M,T} E_{\mathrm{prf}}$ of Theorem 2 holds for some $\lambda_{\mathrm{prf},0}(S,\lambda) \le S(\lambda) + C_{\mathrm{model}} \cdot (\log(S(\lambda)+1) + \lambda + 1)$, any M, T such that $M(\lambda), T(\lambda) \le 2^\lambda$ for sufficiently large λ, and $\varepsilon_{\mathrm{leak}}(\lambda) = 2^{-\lambda}$, where $M(\lambda)$ is now the length of $w \xleftarrow{\$} \mathsf{nu}(\lambda)$ and the experiments also feed 1^λ and $w \xleftarrow{\$} \mathsf{nu}(\lambda)$ into A. In particular, $E_{\mathrm{airom}} \approx E_{\mathrm{prf}}$. In the RAM model, with $A(1^\lambda, w, \cdot)$ regarded as an oracle, E_{prf} is an asymptotic algorithm with*

$$\begin{aligned} M_E &= S + |k| + 2\lceil \log_2(S+1) \rceil + 1, \\ T_E &\le Q_A \cdot T_{\mathrm{ram},\mathsf{Eval}} + C_{\mathrm{ram}} \cdot \log(S + |k| + 2). \end{aligned}$$

In both models (RAM and circuits), the sampling of (w, z, k) in E_{prf} can be combined into $\widetilde{\mathsf{nu}}(\lambda)$, and $E_{\mathrm{prf}}(A)$ is an asymptotic adversary with

$$\begin{aligned} \text{(both)} \quad M' &= M_A + S + |k| \\ &\quad + 2\lceil \log_2(M_A+1) \rceil + 2\lceil \log_2(S+1) \rceil + 2, \\ \text{(RAM)} \quad T' &\le T_A + Q_A \cdot T_{\mathrm{ram},\mathsf{Eval}} + C_{\mathrm{ram}} \cdot \log(M_A + S + |k| + 2), \\ \text{(circuits)} \quad T' &\le C_{\mathrm{ckt}} \cdot (M_A + S + |k| + T_A + Q_A \cdot T_{\mathrm{ckt},\mathsf{Eval}} + 1). \end{aligned}$$

3.1. Comparison with Presampling

Before proceeding to the proofs, we compare our AI-ROM simulation theorems with presampling [9,11,40]. Let's first recall the presampling theorem.

Lemma 4 (presampling with additive error [11; Lemma 1])**.** *For all AI-ROM type $\mathsf{dom} \to \{0,1\}^m$ such that $\mathsf{dom} \ne \{0,1\}^*$, $\mathcal{O}$-advice ai of length S, upper bound Q of number of queries, number $P \in \mathbb{N}_+$ of fixed coordinates, decomposition error $0 < \varepsilon_{\mathrm{decomp}} \le 1$, there exists a z-advice $\mathsf{ai}_{\mathrm{presamp}}$ whose output shall be interpreted as a set $\mathcal{T}$ of no more than P pairs of $(x,y) \in \mathsf{dom} \times \{0,1\}^m$ with distinct x's such that $E_{\mathrm{airom}} \overset{\varepsilon}{\approx}_{\mathcal{A}} E_{\mathrm{presamp}}$ for $\varepsilon = (S + \log_2(1/\varepsilon_{\mathrm{decomp}})) \cdot Q/P + \varepsilon_{\mathrm{decomp}}$, where $\mathcal{A}$ is the set of concrete adversaries making at most Q queries to the oracle and having single-bit output. The experiments are as follows.*

- *E_{airom} is the same as in Theorem 2.*
- *In E_{presamp}, sample two independent random functions $\mathcal{O}, \mathcal{O}_{\mathrm{presamp}} : \mathsf{dom} \to \{0,1\}^m$, then $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O}_{\mathrm{presamp}})$ and $\mathcal{T} \xleftarrow{\$} \mathsf{ai}_{\mathrm{presamp}}(z)$. Define $\mathcal{O}[\mathcal{T}](x)$ to be y if $(x,y) \in \mathcal{T}$ for some y and $\mathcal{O}(x)$ otherwise. Run and output $A^{\mathcal{O}[\mathcal{T}]}(z)$.*

Table 3. COMPARISON BETWEEN OUR THEOREMS AND PRESAMPLING.

aspect	our theorems	presampling
oracle	PRF instance	random except at P points
assumption	exponentially secure PRF	✓ none
adversary	computational	query-bounded
error	✓ could be exponential	inverse polynomial
extra advice	✓ $\mathrm{poly}(S,\lambda,n,m)$	$\Theta((n+m)SQ/\varepsilon)$
per-query size (circuit)	✓ $\mathrm{poly}(S,\lambda,n,m)$	$\Theta((n+m)SQ/\varepsilon)$
per-query time (RAM)	$\mathrm{poly}(S,\lambda,n,m)$	✓ $\Theta(n \log n + m)$

AI-ROM is not efficiently implementable by standard-model algorithms. Both our theorem and presampling provide a way to simulate AI-ROM efficiently. Their major differences are in Table 3. We make the following remarks.

One major advantage of our method is that it achieves the standard definition of simulation (negligible error against polynomial-time distinguishers) in polynomial time, which resolves Unruh's conjecture (Corollary 10) that AI-ROM does not create computational advantage. With presampling, the simulation efficiency grows inverse-polynomially with the error — this growth relation is tight [10,11] — so it will *never* achieve negligible error in polynomial time.

Although our theorems only work with computationally bounded adversaries, the time bound is exponential, covering the practically relevant case. Our simulation efficiency does *not* depend on the number of queries, hence it could enjoy significant savings in some parameter regimes compared to presampling.

The table shows efficiency from a generic exponentially secure PRF. By using a PRF with very fast evaluation (Section 4) in the RAM model, the per-query time of our simulator is $mn^{1+\gamma}\,\mathrm{poly}(\log S, \log\lambda, \log n)$, where $\gamma > 0$ is an arbitrary positive constant. Despite still worse than presampling, its significant savings on extra advice come out on top. We depict practical attacks as having S much smaller than Q, so the independence of Q is meaningful. Concretely, our simulator only needs extra advice $S^{1+\gamma}\,\mathrm{poly}(\lambda)$, hence is better as long as Q is not too small.

Unlike presampling, where the correlation is controlled by $\mathcal{T}$, we have little knowledge about the correlation between the PRF used as the oracle and the auxiliary input. This leads to great difficulty in proving security *reliant on* the random oracle, excluding the techniques of programming and extraction. Nevertheless, our simulation could be used to reduce assumptions *unrelated* to the random oracle more tightly to their standard-model versions. The two methods are not mutually exclusive. In applications, for each pair of neighboring hybrids, we can utilize the suitable method.

16. Here, $|k|$ depends on the specific PRF being used and is polynomial in $\lambda_{\mathrm{prf}}, |\mathsf{desc}(\mathsf{dom})|, m$ (hence polynomial in $|\mathsf{desc}(\mathsf{dom})|, m, S$ and polylogarithmic in $M, T, \varepsilon_{\mathrm{leak}}$). The time of E only includes its initialization and responses to A's queries.

3.2. Proofs

We derive our concrete AI-ROM simulation in two steps. For simplicity, let's assume $|\mathsf{desc}(\mathsf{dom})|, m, S, M, T \le 2^\lambda$ and $\varepsilon_{\text{leak}} \ge 2^{-\lambda}$ (the setting useful for asymptotics). The $\mathcal{O}$-advice ai we are given with could be arbitrarily complex — difficult to handle — so we first rely on the leakage simulation lemma to bring it down to $2^{S+\mathrm{O}(\lambda)}$, while keeping $(\mathcal{O}, z)$ indistinguishable to size-M time-T adversaries. We then regard the leakage simulator together with the underlying adversary as an adversary against the PRF, whose overall complexity is dominated by that of the leakage simulator. Because the PRF is secure against exponential-time adversaries, we can adjust the security parameter to some $\lambda_{\text{prf}} = S + \mathrm{O}(\lambda)$. Replacing the random oracle by PRF completes the proof.

Note that in the intermediate step where we invoke PRF security, its adversary runs in *excessive* time as it includes the leakage simulator — a merely polynomial hardness assumption would not survive it (even with complexity leveraging, the adjusted security parameter could become super-polynomial). However, once we arrive at the final stage, the PRF key k and the leakage z can be regarded as (standard-model) advice, the leakage simulator is no longer part of the adversary, and the adversary, with oracle queries answered by PRF, is *again efficient*, only suffering a blow-up polynomial in S, λ.

We now present the formal proof.

Proof (Theorem 2). Consider the RAM version. Define

$$\mathcal{P} = \left\{ A \;\middle|\; \begin{array}{l} A \text{ is a RAM program with single-bit output,} \\ M_A \le M,\ T_A \le T \end{array} \right\},$$
$$\mathcal{A} = \{\, A \mid A \text{ is a distribution over } \mathcal{P} \,\},$$
$$\mathcal{F} = \{\, f_{A,r} : (\mathcal{O}, z) \mapsto A^{\mathcal{O}}(z; r) \mid A \in \mathcal{P}, \text{randomness } r \in \{0,1\}^T \,\}.$$

Let h be the RAM program in Lemma 1, then there exist a list $\mathcal{L}$ of functions in $\mathcal{F}$ and a bit-string z', of promised lengths, such that $(\mathcal{O}, \mathsf{ai}(\mathcal{O})) \overset{\varepsilon_{\text{leak}}}{\approx}_{\mathcal{F}} (\mathcal{O}, h^{\mathcal{O},\mathcal{L}}(z'))$, which implies $(\mathcal{O}, \mathsf{ai}(\mathcal{O})) \overset{\varepsilon_{\text{leak}}}{\approx}_{\mathcal{A}} (\mathcal{O}, h^{\mathcal{O},\mathcal{L}}(z'))$.

Encode $\mathcal{L}$ into a bit-string z'' by writing down first a prefix-free encoding of $|\mathcal{L}|$, next a prefix-free encoding of T, then for each $f_{A,r} \in \mathcal{L}$, a prefix-free encoding of M_A, next A itself, then r itself. By Lemma 1 and the definition of $\mathcal{F}$, the length of z'' is bounded by $C \cdot (S+1)\varepsilon_{\text{leak}}^{-2} \cdot (M+T+1)$. For each $A \in \mathcal{A}$, let $A^{\mathcal{O}}_{\text{prf}}(z', z'')$ first choose dom, m (from the random oracle model it targets) as the PRF challenge lengths, next parse z'' into $\mathcal{L}$, then run $z \xleftarrow{\$} h^{\mathcal{O},\mathcal{L}}(z')$, and lastly run and output $A^{\mathcal{O}}(z)$. We have

$$M_{A_{\text{prf}}} \le \overbrace{C_{\text{ram}}}^{\text{including parsing}} \cdot \Big(\overbrace{|\mathsf{desc}(\mathsf{dom})| + m}^{\mathsf{dom},\, m} + \overbrace{1 + M_A}^{\text{gluing, } h,\, A} + \underbrace{S\varepsilon_{\text{leak}}^{-2} + (S + \log \varepsilon_{\text{leak}}^{-1})^2 \varepsilon_{\text{leak}}^{-1}}_{z'} + |z''| \Big),$$

$$T_{A_{\text{prf}}} \le C_{\text{ram}} \cdot \Big(\overbrace{(|\mathsf{desc}(\mathsf{dom})| + m + |z''|)^{C_{\text{ram}}}}^{\text{parsing, gluing}} + \underbrace{2^S \cdot S\varepsilon_{\text{leak}}^{-2} \cdot T}_{\mathcal{L}^{\mathcal{O}} \text{ in } h} + \underbrace{2^S (S+1)\varepsilon_{\text{leak}}^{-2}}_{\text{other in } h} + T_A \Big).$$

Both are bounded by $2^{\lambda_{\text{prf},0}}$ for some

$$\lambda_{\text{prf},0} \le S + C_{\text{ram}} \log((|\mathsf{desc}(\mathsf{dom})| + m + S + M + T + 1)/\varepsilon_{\text{leak}}).$$

For $\lambda_{\text{prf}} \ge \lambda_{\text{prf},0}$, it follows from PRF security that $\mathcal{O}$ (random) and $\mathsf{Eval}(k, \cdot)$ (with $k \xleftarrow{\$} D_{\lambda_{\text{prf}},\mathsf{dom},m}$) are $\varepsilon_{\text{prf}}(\lambda_{\text{prf}})$-indistinguishable to A_{prf}'s. By definition, this is exactly $(\mathcal{O}, h^{\mathcal{O},\mathcal{L}}(z')) \overset{\varepsilon_{\text{prf}}(\lambda_{\text{prf}})}{\approx}_{\mathcal{A}} (\mathsf{Eval}(k,\cdot), h^{\mathsf{Eval}(k,\cdot),\mathcal{L}}(z'))$.

Defining $\widetilde{\mathsf{ai}}(k)$ as $h^{\mathsf{Eval}(k,\cdot),\mathcal{L}}(z')$, we obtain $E_{\text{airom}} \overset{\varepsilon}{\approx}_{\mathcal{A}} E_{\text{prf}}$ by hybrid argument, where $\varepsilon = \varepsilon_{\text{leak}} + \varepsilon_{\text{prf}}(\lambda_{\text{prf}})$ and subscript "$\mathcal{A}$" means the same as "M, T (for RAM)". To construct the concrete algorithm E_{prf}, recall (Definition 1) that it is a *distribution* over RAM programs. We let the distribution sample $k \xleftarrow{\$} D_{\lambda_{\text{prf}},\mathsf{dom},m}$ and $z \xleftarrow{\$} \widetilde{\mathsf{ai}}(k)$, then output a RAM program with z, k hardwired that calls $A^{\mathsf{Eval}(k,\cdot)}(z)$. The bounds of M_E, T_E follow easily, where the model-dependent terms are due to hardwiring the input and combining Eval with gluing code. The derivation of $E_{\text{prf}}(A)$ (for RAM and circuits) is similar. We only note that for circuits, M_A and $M_{\mathsf{ckt},\mathsf{Eval}}$ already subsume S and $|k|$, so the latter two terms do not appear in M'. □

Proof (Theorem 3). Let (nu, A) be an asymptotic RAM adversary such that for sufficiently large λ it holds that $M_A(\lambda), T_A(\lambda) \le 2^\lambda$. For each λ, let $w \xleftarrow{\$} \mathsf{nu}(\lambda)$ and $A'_\lambda(\cdot) = A(1^\lambda, w, \cdot)$, then A'_λ is a concrete RAM adversary with

$$M'_\lambda \le |A| + \lambda + M_A(\lambda) + C_{\text{ram}} \cdot \log(|A| + \lambda + M_A(\lambda) + 2),$$
$$T'_\lambda = T_A(\lambda).$$

Now, $|\mathsf{desc}(\mathsf{dom}(\lambda))| \le 2^\lambda$, $m(\lambda) \le 2^\lambda$, $M'_\lambda \le 2^{\lambda+1}$, and $T'_\lambda \le 2^\lambda$, all of which hold for sufficiently large λ, for which we apply Theorem 2 with $\mathsf{dom}(\lambda), m(\lambda), S(\lambda), M'_\lambda, T'_\lambda$ and $\varepsilon_{\text{leak}} = 2^{-\lambda}$ and conclude

$$\begin{aligned} \lambda_{\text{prf},0} &\le S(\lambda) + C_{\text{ram}} \cdot \log \frac{2^\lambda + 2^\lambda + S(\lambda) + 2^{\lambda+1} + 2^\lambda + 1}{2^{-\lambda}} \\ &\le S(\lambda) + C_{\text{ram}} \cdot (\log(S(\lambda)+1) + \lambda + 1). \end{aligned}$$

The circuit version is similar. For polynomial indistinguishability, we further rely on $S(\lambda) \le \mathrm{poly}(\lambda) \le 2^\lambda$ for sufficiently large λ.

For the second part. Encode (z, k) by writing down a prefix-free encoding of $S = |z|$ then z, k themselves. This gives the bound of M_E. The algorithm $E^{A(w,\cdot)}_{\text{prf}}(z, k)$ parses its input then calls $A^{\mathsf{Eval}(k,\cdot)}(w, z)$, giving the bound of T_E. The derivation of $E_{\text{prf}}(A)$ is similar. □

4. PRF with Fast RAM Evaluation

We start by formally defining the requirements from a pseudorandom function with fast RAM evaluation.

Definition 8 (Syntax of the PRF)**.** A PRF function family $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval})$ consists of two probabilistic polynomial time algorithms satisfying the following desiderata.

- $\mathsf{PRF.Setup}(1^\lambda, 1^n)$: takes as input a security parameter λ and an input length n (which is a polynomial in λ) and it outputs a key $K \in \mathcal{K}_{\lambda,n}$.
- $\mathsf{PRF.Eval}(K \in \mathcal{K}_{\lambda,n}, x)$ is a deterministic polynomial time algorithm that takes as input the key K and an input $x \in \{0,1\}^n$. It outputs a bit.

Definition 9 (Security of the PRF)**.** A PRF function family $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval})$ is said to be (T, ε) secure if for every $\lambda \in \mathbb{N}$, every polynomial input length $n(\lambda)$, every T time (non-uniform) adversary $\mathcal{A}$ we have that:

$$|\Pr[A^{\mathcal{O}}(1^\lambda) = 1] - \Pr[A^{\mathsf{PRF.Eval}(K,\star)}(1^\lambda) = 1]| \leq \varepsilon$$

Here, $\mathcal{O} : \{0,1\}^n \to \{0,1\}$ is a random oracle and $K \leftarrow \mathsf{PRF.Setup}(1^\lambda, 1^n)$.

In this work, we care about security against $T = 2^\lambda$ time algorithms with advantage ε that are negligible in λ.

Definition 10 (Fast RAM Computation Time)**.** A PRF function family $\mathsf{PRF} = (\mathsf{PRF.Setup}, \mathsf{PRF.Eval})$ is said to have fast Ram computation time if: for any arbitrarily small constant $\gamma > 0$, the size of key is $\tilde{O}(n^{1+\gamma}\lambda^{1+\gamma})$ and in the RAM model each query can be responded to in time $n^{1+\gamma}\,\mathrm{poly}(\log(\lambda \cdot n))$.

In this section, we study and design a pseudorandom function family

4.1. Construction from Goldreich's Function

Our construction of a pseudorandom function with fast RAM evaluation is based on Goldreich's expander-based one-way function [19] that has received years of study [19, 24–35].

Definition 11 (Goldreich's One-Way Function Candidate)**.** Goldreich's one-way function candidate is of the form $f_{G,P} : \{0,1\}^\ell \to \{0,1\}$, indexed with a local Boolean predicate $P : \{0,1\}^L \to \{0,1\}$ for a locality $L = L(\ell)$ and a hypergraph $G = (S_1, \ldots, S_m)$ where each hyperedge $S_i \subset [\ell]$ of size L. The computation is described as follows: $f_{G,P}(x)$ on input $x \in \{0,1\}^\ell$ outputs $y = (y_1, \ldots, y_m)$ where $y_i = P(x|_{S_i})$. Here $x|_{S_i}$ are the bits corresponding to the set S_i in the same order as given by S_i.

The expander graph and the predicate has to be chosen carefully based on insights developed in a long sequence of works [19,24–36]. We discuss these aspects at a high level below. Detailed discussions can be found in Section 4.3.

How to Choose the Predicate P The predicate is typically chosen to be a local function with locality L some constant and the number of outputs are typically some polynomial $m = \ell^{O(L)}$. Since we are interested in building a pseudorandom function, number of outputs must be super-polynomial and therefore we will work with a super-constant locality $L = \ell^\delta$ for some constant $\delta < 1$. Such parameters have also been considered in the past [23] in the context of constructing pseudorandom functions.

Based on the extensive cryptanalysis, culminating in [36] a good choice of predictes are one that simultaneously optimizes resiliency and rational degree (refer to Section 4.3 for a discussion). The gold-standard for the choice of predicate is the predicate $\mathsf{XOR\text{-}MAJ}_{L'}$ that takes as input $L = 2 \cdot L'$ inputs and computes:

$$P(x_1, \ldots x_{L'}, x_{L'+1}, \ldots, x_L) = x_1 \oplus \ldots x_{L'} \oplus \mathsf{Maj}(x_{L'+1}, \ldots, x_L)$$

Namely the predicate first computes an exclusive or of first L' bits and then exclusive ors with majority function applied on the next L' bits.

How to Choose the Graph Roughly speaking, for a randomly chosen graph $(S_1, \ldots, S_m)$, with probability $\ell^{-O(L)}$ two sets (say) S_i and S_j might turn out to be the same in which case the function $f_{G,P}$ will triviably be distinguishable. More generally, one could find "cycles"/"even-covers". These are sets $S_{i_1}, \ldots, S_{i_t}$ such that every variable contained in these sets appear an even number of times in the multi-set union of $S_{i_1}, \ldots, S_{i_t}$. If t is really small, this might cause the outputs $y_{i_1}, \ldots, y_{i_t}$ to be biased leading to a $2^{\tilde{O}(t)}$ time attack. Thus, we must choose the graph so that t is not very small. Such t sized cycles can be ruled out if the graph is sufficiently expanding.

Definition 12 (Expansion of a Hypergraph)**.** We say that a L regular hypergraph $G = (S_1, \ldots, S_m)$ over ℓ nodes is (t, α) expanding if for every non-empty set $T \subseteq [m]$ of size r less than or equal to t, we have that the number of nodes in $|\cup_{i \in T} S_i| > \alpha \cdot L \cdot r$.

It is easy to observe that if $S_{i_1}, \ldots, S_{i_t}$ forms a cycle then the graph is not $(t, \frac{1}{2})$ expanding as every variable in the union of hyperedges/sets appear an even number of times. As we discuss in Section 4.3, lower-bounds against certain classes of algorithms that form the best known attack against Goldreich functions (polynomial calculus, light linear tests, sum-of-squares etc.) can be proven provided α is large enough. We work with a conservative $\alpha = 0.99$ for the following assumption. Similar assumptions were used in [23] for design of efficient pseudorandom functions.

Assumption 1 (Goldreich's Assumption)**.** Let $\delta \in (0,1)$ be a constant, $\ell \in \mathbb{N}$ be the input length, $L = 2L' = \Theta(\ell^\delta)$ be the locality and P be the predicate $\mathsf{XOR\text{-}MAJ}_{L'}$. Let $\mathcal{G} = \{G_\ell\}_\ell$ be a family of L regular hypergraph with m hyperedges such that the G_ℓ is $(t, 0.99)$ expanding. Then it holds that for any p.p.t. adversary running in time $2^{\tilde{O}(t)}$ we have that:

$$\{f_{G,P}(x) : x \leftarrow \{0,1\}^\ell\} \approx_c \{y : y \leftarrow \{0,1\}^m\}$$

Above, the computational indistinguishability holds with advantage bounded by negligible in ℓ.

We note that based on our current understanding we can strengthen the above assumption to work with an advantage $2^{-\tilde{O}(t)}$. Moreover, [23] showed that the above assumption follows from the hardness of search with slightly worse parameters.

We are now ready to describe our construction.

Construction Our construction of a PRF is exactly identical to Golreich's PRG with the xor-majority predicate P of super-constant locality L and seed length ℓ that we will set appropriately (depending on the security parameter λ and input length n of the PRF). The number of outputs for the Goldreich's function will be set to $m = 2^n$.

Ingredient: We will need an efficient (randomized) algorithms $(\mathsf{GraphSetup}, \mathsf{GraphEdge})$ with the following desiderata:

- $\mathsf{GraphSetup}$ takes as input the graph parameters, number of variables/nodes 1^ℓ, number of hyperedges $m = 2^n$, locality 1^L, expansion bound 1^t and it outputs a string pp.
- Algorithm $\mathsf{GraphEdge}(\mathsf{pp}, i)$ is a deterministic algorithm that takes as input pp and an index $i \in [m]$ and outputs a set $S_i \subset [\ell]$ of size L corresponding to the hypergraph $G = (S_1, \ldots, S_m)$. The running time of this algorithm is $L^{1+\gamma}\,\mathrm{poly}(\log \ell)$ in the RAM model for arbitrarily small constant $\gamma > 0$.
- With all but negligible probability in ℓ, the graph produced by these algorithms satisfy $(t, 0.99)$ expansion for some t that we will set later.

We show how to construct such an algorithm in Section 4.2.

Parameters We set various parameters below (all our parameters have a lot of room and may not represent an optimal setting):

- The seed length $\ell = n \cdot \lambda^{1+\gamma}$.
- Locality $L = 100n$.
- Expansion bound $t = \lambda^{1+\gamma/2}$

The constant $\gamma > 0$ can be chosen to be arbitrarily small.

As we show in Section 4.2 in Theorem8, for the parameters above the running time of $\mathsf{GraphSetup}$ is polynomial and the size of preprocessing is bounded by $t^{1+\gamma} n^{1+\gamma} \log^{1+\gamma}(\lambda) = O(\lambda^{1+2\gamma} n^{1+\gamma} \log^{1+\gamma}(\lambda))$. Further, the $\mathsf{GraphEdge}$ calls can be answered in time $n^{1+\gamma}\,\mathrm{poly}(\log(n \cdot \lambda))$ for arbitrarily small constant $\gamma > 0$. Further, the hypergraph output by the graph sampler algorithms satiesfies expansion with all but $\exp(-n \log \lambda)$ probability.

Construction 1. We now describe our construction.

- $\mathsf{PRF.Setup}(1^\lambda, 1^n)$: The algorithm runs $\mathsf{pp} \leftarrow \mathsf{GraphSetup}(1^\ell, m, 1^L, 1^t)$ and $s \leftarrow \{0,1\}^\ell$. The key consists of $K = (\mathsf{pp}, s)$.
- $\mathsf{PRF.Eval}(K, x \in \{0,1\}^n)$: Run $S_x \leftarrow \mathsf{GraphEdge}(\mathsf{pp}, x)$ and then compute and output $y = P(s|_{S_x})$.

One can show that a random hypergraph where each S_i is randomly sampled satisfies $(t, 0.99)$ expansion for $t = \lambda^{1+\gamma/2}$ with all but $\exp(-\Omega(n \log \lambda))$ probability. Our graph sampler does not sample it randomly, but samples it using a k wise independent distribution for $k = t$. We show in Section 4.2 how to implement it with the efficiency related desiderata. It can be shown with an identical analysis that t wise independent distribution will also produce $(t, 0.99)$ expander with similar probability.

Theorem 5. *If* $(\mathsf{GraphSetup}, \mathsf{GraphEdge})$ *satisfies the desiderata described above and for the parameters set above, the construction 1 satisfies fast runtime in the RAM model. Namely, the key can be sampled in polynomial time and is of the size* $O(n^{1+\gamma}\lambda^{1+2\gamma}\log^{1+\gamma}(\lambda \cdot n))$ *and each query can be computed in time* $n^{1+\gamma}\,\mathrm{poly}(\log(n \cdot \lambda))$ *for an arbitrarily small constant* $\gamma > 0$.

Proof. We analyze this based on Theorem 8. Note that $\mathsf{PRF.Setup}$ simply samples a seed s that can be done in time $\ell = n \cdot \lambda^{1+\gamma}$. It also runs $\mathsf{GraphSetup}$ that takes time $O(n^8\,\mathrm{poly}(\log(n \cdot \lambda)) + n^{1+\gamma}\lambda^{1+2\gamma}\,\mathrm{poly}(\log(n \cdot \lambda)))$ time and produces a preprocessing of size at most $O(n^{1+\gamma}\lambda^{1+2\gamma}\,\mathrm{poly}(\log(n \cdot \lambda)))$.

We now analyze the running time in the RAM model for the evaluation algorithm $\mathsf{PRF.Eval}$. The per query time of $\mathsf{GraphEdge}$ is $O(n^{1+\gamma}\,\mathrm{poly}(\log(\lambda \cdot n)))$ in the RAM model to compute the set S_x. Then computing the predicate $P(s|_{S_x})$ takes $\tilde{O}(L)$ time for the predicate $\mathsf{XOR\text{-}Maj}$ as this can be done after looking up s on locations corresponding to the set S_x. As $L = 100n$, the PRF can be evaluated in time $\tilde{O}(n)$ given S_x. Overall, query can be computed in time $n^{1+\gamma}\,\mathrm{poly}(\log(n \cdot \lambda))$. □

Theorem 6. *Assume that* $(\mathsf{GraphSetup}, \mathsf{GraphEdge})$ *satisfies the desiderata described above and let construction 1 use the parameters set above. If Assumption 1 holds then the construction 1 is secure against* 2^λ *sized adversaries.*

Proof. We have that with overwhelming probability (all but $\exp(-\Omega(n \log \lambda))$ probability) the hypergraph induced by $(\mathsf{GraphSetup}, \mathsf{GraphEdge})$ forms a $(t, 0.99)$ expander for $t = \lambda^{1+\gamma/2}$. Further assuming the hypergraph is expanding, Assumption 1 posits that the distribution $f_{G,P}(x)$ is indistinguishable from $y \leftarrow \{0,1\}^m$ against $2^{\tilde{O}(t)}$ time adversaries. Note that $f_{G,P}$ consists of all the 2^n outputs of the PRF. This proves the claim due to a straightforward reduction to Assumption 1. □

4.2. Graph Sampler

In this section we describe our graph sampler. Our high level idea is that that each hyperedge will be computed through a t wise independent hash function.

We will make use of the following powerful lemma from [20].

Theorem 7 (reinterpreted from Theorem 5.1 in [20])**.** *Let* $\mathbb{Z}_p$ *be a field of prime order* p*. Let* $f(x)$ *be a polynomial of degree* d *in* $\mathbb{Z}_p[x]$*. Let* $\gamma > 0$ *be arbitrary constant. For*

large enough d, one can compute in time $O(d^{1+\gamma}\log^{1+\gamma} p)$ a data structure with the property that given any $\alpha \in \mathbb{Z}_p$, $f(\alpha)$ can be computed in time $O(\log^{1+\gamma} p \cdot \mathrm{poly}(\log d))$ in the RAM model.

We will use the Theorem 7 to efficiently generate hyperedges. The main punchline of the theorem is that any degree d polynomial over $\mathbb{Z}_p$ can be computed on any input in time that is much faster to compute the polynomial in the first place (provided a polynomial time pre-processing phase computing a datastructure).

Recall that the hyperedges $(S_1, \ldots, S_m)$ must form a $(t, 0.9)$ expander for Assumption 1 to hold. We will show that a t wise distribution with overwhelming probability yield a $(t, 0.99)$ expander. We will generate such a distribution using random degree t polynomial over an appropriately chosen prime modulus p. Fast implementation follows naturally from Theorem 7.

We describe the construction now.

Ingredients and Parameters: We work with the following choice of parameters:

- The seed length $\ell = n \cdot \lambda^{1+\gamma}$.
- Locality $L = 100n$.
- Expansion bound $t = \lambda^{1+\gamma/2}$.
- $m = 2^n$.

where λ is the security parameter and n is an arbitrary polynomial in λ. $\gamma > 0$ is an arbitrarily small constant greater than 0.

Recall that there are $N = \binom{\ell}{L}$ is the total number of hyperedges. While N is not a prime, using Cramer's conjecture [41] in number theory one could find a prime $p = N + K$ for $K = O(\log^2 N)$ in polynomial time by simply trying out $O(\log^2 N)$ values for K and testing those numbers for primality. We work with this prime modulus. Note that Cramer's conjecture is not essential and we will remark later on how one could remove the reliance on Cramer's conjecture . It is well known that there exists an $\tilde{O}(\log N)$ time computable injective algorithm Set that takes as input a number in $[1, N]$ and outputs one of N sets, $S \subset [\ell]$ of size L.

Construction 2. We are ready to descibe our construction of the graph sampler.

- $\mathsf{GraphSetup}(1^\ell, m, 1^L, 1^t)$: Find a prime p greater than N that is also very close to N, where $N = \binom{\ell}{L}$. By Cramer's conjecture, $p = N + O(\log^2 N)$ and therefore such a prime p can be found out in polynomial time. Sample a random degree t polynomial $h(z) = a_0 + a_1 z + a_2 z_2^2 + \ldots + a_t z^t$ where each a_i is randomly chosen from $\mathbb{Z}_p$. Preprocess the polynomial according to form a data structure σ relying on Theorem 7. Output $\mathsf{pp} = \sigma$
- $\mathsf{GraphEdge}(\mathsf{pp}, x \in [\ell])$: Interpret x as an element in $\mathbb{Z}_p$ and compute $h(x) = r$ by using the fast RAM evaluation process given by Theorem 7. Note that $r \in \mathbb{Z}_p$. With probability $1 - O(\frac{\log^2 N}{N})$, $r \in [1, N]$. If this happens, compute and output the set $S_x = \mathsf{Set}(r)$ otherwise output $\perp$.

We analyze the properties of the graph sampler below.

Theorem 8. *If Cramer's conjecture holds, for the parameters described above, the algorithm* $\mathsf{GraphSetup}$ *takes $O(n^8 \,\mathrm{poly}(\log(n \cdot \lambda)) + n^{1+\gamma}\lambda^{1+2\gamma}\,\mathrm{poly}(\log(n \cdot \lambda)))$ time and produces a preprocessing of size at most $O(n^{1+\gamma}\lambda^{1+2\gamma}\,\mathrm{poly}(\log(n \cdot \lambda)))$. Moreover, per query time of* $\mathsf{GraphEdge}$ *is $O(n^{1+\gamma}\,\mathrm{poly}(\log(\lambda \cdot n)))$ in the RAM model. Above $\gamma > 0$ is an arbitrary constant.*

Proof. Note that assuming Cramer's conjecture [41] the prime p can be found in polynomial time. Note that $N = \exp(O(n \log(n \cdot \lambda)))$. Thus to find p, we have to compute primality testing of $O(n^2 \log(n \cdot \lambda)))$ numbers of bit-length $O(n \log(n \cdot \lambda))$. Primality testing can be done by the AKS algorithm [42] in time $O(n^6 \,\mathrm{poly}(\log(n \cdot \lambda)))$.

Observe that $\mathsf{GraphSetup}$ also computes the datastructure to evaluate a degree $t = \lambda^{1+\gamma/2}$ degree polynomial over $\mathbb{Z}_p$. This can be done in time $t^{1+\delta} \cdot \log^{1+\gamma} p$ based on Theorem 7. The preprocessing consists of the datastructure. This proves the claim about $\mathsf{GraphSetup}$.

Further, $\mathsf{GraphEdge}$ simply computes the polynomial. Based on the Theorem 7 in time $n^{1+\gamma}\,\mathrm{poly}(\log(\lambda \cdot n))$. □

Theorem 9. *If Cramer's conjecture is true, for the parameters above, with all but $\exp(-\Omega(n \cdot \lambda))$ probability, the construction gives a $(t, 0.99)$ expander.*

Proof. We first find an upper bound on the probability that the algorithm $\mathsf{GraphEdge}$ outputs $\perp$ for one of the $x \in [m]$. Let V_x be an indicator variable that takes a value 1 if the algorithm outputs $\perp$ on x (or equivalenly $h(x) > N$) and 0 otherwise. Note that $\mathbb{E}[V_x] = \frac{p-N}{p} = O(\frac{\log^2 N}{N})$. Thus, $\Sigma_x \mathbb{E}[V_x] = O(\frac{m \log^2 N}{N})$. By Markov's inequality, this establishes that the algorithm outputs $\perp$ is at most $O(\frac{m \log^2 N}{N})$. For the parameters set in the theorem $N = \binom{\ell}{L} = \exp(\Omega(n \log \lambda))$ and $m = 2^n$. Therefore we have that the required probability is $\exp(-\Omega(n \log \lambda))$. Let E be the event that the algorithm does not output $\perp$ at any input $x \in [m]$.

We consider the probability that given sets $S_{i_1}, \ldots, S_{i_T}$ for $1 \le T \le t$ for distinct inputs $i_1, \ldots, i_T$ to violate the expansion condition $|S_{i_1} \cup \ldots \cup S_{i_T}| < 0.99 \cdot T \cdot L$. We call this event $Q_{i_1,\ldots,i_T}$ and compute an upper-bound on the probability for the event $Q_{i_1,\ldots,i_T}$ given the event E. Notice that if E holds, the sets $S_{i_1}, \ldots, S_{i_T}$ are, uniformly distributed due to the t wise independence property of h, from the space of all sets of size L in $[\ell]$. Letting $\alpha = 0.99$, this probability can therefore be upper bounded as:

$$\binom{\ell}{\alpha \cdot T \cdot L} \cdot \binom{\alpha \cdot T \cdot L}{L}^T / \binom{\ell}{L}^T$$

The expression is basically counting the number of ways that T hyperedges depend on some $\alpha \cdot T \cdot L$ nodes.

We can already apply union bound at this point and compute an upper bound on the probability of $\bigcup_{T\in[1,T]}(\cup_{i_1,\ldots,i_T}Q_{i_1,\ldots,i_T})$ given E:

$$\sum_{T\in[t]}\binom{m}{T}\binom{\ell}{\alpha\cdot T\cdot L}\cdot\binom{\alpha\cdot T\cdot L}{L}^T/\binom{\ell}{L}^T$$

We analyze each term for $T\in[t]$ and bound them separately.

$$\binom{m}{T}\binom{\ell}{\alpha\cdot T\cdot L}\cdot\binom{\alpha\cdot T\cdot L}{L}^T/\binom{\ell}{L}^T$$

We use the naive bounds for Binomial coefficients $\left(\frac{a}{b}\right)^b\leq\binom{a}{b}\leq e^b\left(\frac{a}{b}\right)^b$. Each term is upper bounded as:

$$e^{T+\alpha\cdot T\cdot L+T\cdot L}\left(\frac{m}{T}\right)^T\cdot\left(\frac{\ell}{\alpha\cdot T\cdot L}\right)^{\alpha\cdot T\cdot L}\cdot\left(\frac{\alpha\cdot T\cdot L}{\ell}\right)^{T\cdot L}$$

$$e^{T+\alpha\cdot T\cdot L+T\cdot L}\left(\frac{m}{T}\right)^T\cdot\left(\frac{\alpha\cdot T\cdot L}{\ell}\right)^{(1-\alpha)\cdot T\cdot L}$$

We now plug various parameters $\alpha=0.99$, $L=100n$, $m=2^n$, $\ell=n\lambda^{1+\gamma}$ and deduce the upper bound as:

$$\exp(O(T\cdot n))\cdot\left(\frac{m}{T}\right)^T\cdot\left(\frac{\alpha\cdot T\cdot L}{\ell}\right)^{(1-\alpha)\cdot T\cdot L}$$

Note that:

$$\left(\frac{m}{T}\right)^T=\exp(O(T\cdot n))$$

Finally,

$$\left(\frac{\alpha\cdot T\cdot L}{\ell}\right)^{(1-\alpha)\cdot T\cdot L}=\left(\frac{99\cdot T\cdot n}{n\cdot\lambda^{1+\gamma}}\right)^{n\cdot T}$$
$$\leq\exp(-\Omega(n\cdot T\cdot\log\lambda)).$$

The last inequality uses the fact that $T\leq t=\lambda^{1+\gamma/2}$.

Combining the above observations we have the required probability as:

$$\exp(-\Omega(n\cdot T\cdot\log\lambda))$$

This concludes that the required probability of upper bound on the probability of $\bigcup_{T\in[1,T]}(\cup_{i_1,\ldots,i_T}Q_{i_1,\ldots,i_T})$ given E as:

$$\sum_{T\in[t]}\exp(-\Omega(n\cdot T\cdot\log\lambda))=\exp(-\Omega(n\cdot T\cdot\log\lambda)).$$

Based on the probability of event E and the probability of violating expansion property, we can therefore finally conclude that with all but $\exp(-n\log\lambda)$ probability the graph sampler outputs a $(t,0.99)$ expanding hypergraph. □

Remark 1 (On the Necessity of Cramer's Conjecture). Cramer's conjecture is not necessary. We use it only for simplifying our presentation. Indeed, another way could be to work with an extension field of $\mathbb{Z}_2$, and produce a t wise independent bit sequence, and use these random bits to map strings to sets. In doing so, we will have to use a sufficiently big finite field so that the order of the field is sufficiently bigger than N. The analysis will only be slightly more involved.

4.3. Discussion on the Assumption

We now discuss cryptanalysis of the assumption. This cryptanlaysis is a highly simplified survey of a long line of works [19,24–35] and in particular the state-of-the-art work of [36]. The two main kinds of attacks on Goldreich's one-way function considered in the work are of the form:

- Linear Attacks: Given a hypergraph $G=(S_1,\ldots,S_m)$ and the output of the function $f_{G,P}=(y_1,\ldots,y_m)$, this attack considers linear attacks of the form $\oplus_{i\in[T]}y_i$ for all sets T. We say that the $f_{G,P}$ fails to be an ε biased generator if there exists a non-empty set T so that $\left|\mathbb{E}_x[\oplus_{i\in T}y_i]-\frac{1}{2}\right|>\varepsilon$. As shown in prior works, one typically shows security by analyzing two different cases. In the first case, one considers *light tests*. These tests consider tests T for which the size of T is less than or equal to t where the hypergraph is $(t,0.99)$ expanding. For such tests for the predicates P that satisfies resiliency of $\frac{L}{2}$ (the property is described later), it can be shown as in [36] $\left|\mathbb{E}_x[\oplus_{i\in T}y_i]-\frac{1}{2}\right|=0$. For *heavier tests*, showing this property holds for a small ε for all tests $T>t$ requires some work. It was proven in [36] if the predicate has large rational degree (we provide the description later) $\Omega(L)$, and if the hypergraph satisfies $(t,0.99)$ expansion for a large enough t, then, ε biased property holds for subexponentially small bias ε. if L is constant [36] prove ε biased property for the $\mathsf{XOR\text{-}Maj}_L$ predicate as long as the hypergraph is expanding. A random graph satisfies such expansion with high probability provided $m=\ell^{\Omega(L)}$. Unfortunately, the bias calculations break down for *heavy linear* tests when L is super-constant or polynomial in the seed length ℓ. Nevertheless, security can still be conjectured against heavy tests. Such a conjecture was implicit in [23] who relied on super-constant locality to realize a PRF.
- Polynomial Calculus: Another really important class of attacks are algebraic attacks. These include Gröbner basis style attacks and other models which manipulate equations algebraically and refute/solve equations algebraically. [36] modelled these equations in Polynomial Calculus [43] proof system and showed lower-bounds in those models. If the predicate P has a rational degree $\frac{L}{4}$ (satisfied by

$\mathsf{XOR\text{-}Maj}$) and if the hypergraph is $(t, 0.99)$ expanding then such Goldreich PRG system cannot be refuted by polynomial calculus proof system of degree $D = t \cdot L \cdot (\frac{1}{16} - \frac{1}{100}) = \Omega(t \cdot L)$. This roughly corresponds to $2^{\tilde{O}(D)}$ time algebraic attacks.

All in all, our current understanding is that to design good PRGs, aside from the graph satisfying $(t, 0.99)$ expansion for large t the predicate with locality L must have large:

- Resiliency: We say that a predicate $P(x_1, \ldots, x_L)$ has a resiliency k if the xor of the predicate with any k variables in $x_1, \ldots, x_L$ is unbiased. For $\mathsf{XOR\text{-}Maj}_L$, it has a resiliency of $L/2$.
- Rational Degree: We say that $P(x_1, \ldots, x_L)$ has a rational degree e if that's the minimum degree for which there exists two non-zero polynomials Q, R of degree less than equal to e so that $PQ = R$. The computation is done over $\mathbb{Z}_2[x_1, \ldots, x_L]$. For $\mathsf{XOR\text{-}Maj}_L$, it has a resiliency of $L/4$. In fact, majority function has the highest rational degree amongst all functions for a given locality.

There are other attacks too such as sum-of-squares/convex optimization based methods. One can prove lower-bounds for those (against versions of attacks that take $2^{\tilde{O}(t)}$) time algorithms as in [35,44,45] based on resiliency and expansion condition alone. We refer to [36] for a survey of other attacks.

5. Direct Applications

5.1. Resolving Unruh's Conjecture

Assuming the existence of any exponentially secure PRF, Theorem 3 resolves Unruh's conjecture that oracle-dependent auxiliary input does not give computational power. We state it in our notation.

Corollary 10 (Conjecture 15 of [14])**.** *Let the AI-ROM be of type* $\mathsf{dom} \to \{0,1\}^m$*. For all efficient adversary* A *and* $\mathcal{O}$*-advice* ai *of polynomial length, there exist an efficient simulator* $\mathcal{S}$ *and a* $\perp$*-advice* nu *of polynomial length such that for all efficient algorithm* B*, it holds that* $E_{\text{airom}} \approx E_{\text{sim}}$*. The experiments work as follows.*

- *In* E_{airom}*, sample a random* $\mathcal{O} : \mathsf{dom} \to \{0,1\}^m$ *then* $z \xleftarrow{\$} \mathsf{ai}(\lambda, \mathcal{O})$*. Let* $A^{\mathcal{O}}(1^\lambda, z)$ *interact with* $B(1^\lambda)$ *and output whatever* B *outputs.*
- *In* E_{sim}*, sample* $w \xleftarrow{\$} \mathsf{nu}(\lambda)$*. Let* $\mathcal{S}(1^\lambda, w)$ *interact with* $B(1^\lambda)$ *and output whatever* B *outputs.*

We note that Theorem 3 is stronger as the simulator has more structure than required by Unruh's conjecture. Namely, nu only depends on ai, and $\mathcal{S}$ is black-box in A.

5.2. Tighter Reductions from Standard-Model Assumptions

Consider a standard-model computational assumption such as factoring and DDH. Does the assumption also hold in AI-ROM? Auxiliary input provides at least the power of standard-model non-uniformity. Formally, it could provide more leverage since the adversary is given oracle access to the random function, correlated with the auxiliary input. Yet it is hard to conceive how such correlation can help with a task unrelated to the random function.

Presampling answers the above question in the positive for polynomial hardness, but its reduction loss is unsatisfactory. Notably, an auxiliary input of S bits could only be simulated by an advice of $\Omega(SQ)$ bits (the bit-fixing truth table), and consequently, in the circuit model, each query suffers a blow-up of $\Omega(SQ)$ in size.

Our methods provide tighter reductions, albeit under computational assumptions. We state them and compare them with presampling, first by computation model, then by type of task (searching or distinguishing). Note that for presampling, lazy sampling outside the bit-fixing table also incurs cost per query. We give it the benefit of discounting such cost — our method is still advantageous or comparable!

Assumptions as Indistinguishability. Let E be an experiment that is black-box and straight-line in the adversary. We require that the size/time of E itself be bounded by $2^{C_{\text{model},E}(\lambda+1)}$. A search assumption can be stated as $E \approx 0$, and a decision assumption, $E \approx \frac{1}{2}$, where $0, \frac{1}{2}$ represent experiments simply outputting 1 with probability $0, \frac{1}{2}$. Let M', T', ε' be the parameters of indistinguishability.

We can lift E into E_{airom} that works with AI-ROM adversaries. Let the AI-ROM be of type $\mathsf{dom} \to \{0,1\}^m$. The experiment E_{airom} samples $w \xleftarrow{\$} \mathsf{nu}()$, a random $\mathcal{O} : \mathsf{dom} \to \{0,1\}^m$, then $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$, and runs and outputs $E(A^{\mathcal{O}}(w, z))$. Let the level of indistinguishability ($E_{\text{airom}} \approx 0$ or $E_{\text{airom}} \approx \frac{1}{2}$) be S, M, T, Q, ε.

We are interested in the relation between S, M, T, Q, ε and M', T', ε'. Throughout this subsection we assume $M', T', S, M, T, Q \le 2^\lambda$ and $\varepsilon', \varepsilon \ge 2^{-\lambda}$. We also assume a $(2^\lambda, 2^{-\lambda})$-secure PRF (D, Eval), i.e., $\varepsilon_{\text{prf}}(\lambda) \le 2^{-\lambda}$.

Circuits. The bottleneck of reduction efficiency is Eval. Its circuit size is lower-bounded by roughly $\lambda_{\text{prf}} = S + \Omega(\lambda)$ set in Theorem 3, which is the smallest reduction overhead we can hope for. The following notion is more relaxed, yet it is the most efficient one for which we know a candidate.

Definition 13 (quasilinear efficiency of PRF)**.** A PRF (Definition 5) is *quasilinearly efficient* if $|k| \le (\lambda + m) \log^{C_{\text{ckt,prf}}}(\lambda + m + 2)$ and

- (for RAM) the RAM time of Eval,
- (for concrete circuits) the circuit size of Eval with k hardwired, or
- (for asymptotic circuits) the time to generate the circuit of Eval given k

is bounded by $(\lambda + m)(|x| + 1) \log^{C_{\text{ckt,prf}}}(\lambda + m + |x| + 2)$, where $|x|$ is (for RAM) the input length or (for circuits) the maximum length of input that the (generated) circuit must handle.

A candidate of quasilinearly efficient and exponentially secure PRF is GGM tree PRF based on an exponentially secure PRG from ring-LWE.

Corollary 11 (from standard-model to AI-ROM, circuit model)**.** *Assuming the existence of quasilinearly efficient and $(2^\lambda, 2^{-\lambda})$-secure PRF, suppose $E \overset{\varepsilon'}{\approx}_{M',T'} 0$ holds in the standard model, then $E_{\mathsf{airom}} \overset{\varepsilon}{\approx}_{S,M,T,Q} 0$ holds in AI-ROM of type $\mathsf{dom} \to \{0,1\}^m$ if $\varepsilon \geq \varepsilon' + 2^{-\lambda}$ and*

$$\begin{aligned}
\text{(concrete)}\quad & M' \geq M + Q \cdot (S+\lambda+m)(\min\{M,n\}+1) \times \\
& \qquad (\lambda+2)^{C_{\mathsf{ckt},E,\mathsf{airom},\mathsf{prf}}}, \\
\text{(asymptotic)}\quad & M' \geq M + (S+\lambda+m)(\lambda+2)^{C_{\mathsf{ckt},E,\mathsf{airom},\mathsf{prf}}}, \\
& T' \geq C_{\mathsf{ckt},E,\mathsf{airom},\mathsf{prf}} \cdot (M+T), \\
& \qquad + (Q+1) \cdot (S+\lambda+m)(\min\{M,n\}+1) \times \\
& \qquad\quad (\lambda+2)^{C_{\mathsf{ckt},E,\mathsf{airom},\mathsf{prf}}}.
\end{aligned}$$

where n is consistent with $\mathsf{dom} = \{0,1\}^n$ or $\mathsf{dom} = \{0,1\}^{\leq n}$ or $n = +\infty$ if $\mathsf{dom} = \{0,1\}^*$, and $M = T$ and $M' = T'$ for the concrete case. Suppose $E \approx 0$ has λ' strong bits of security (Definition 2) and $\lambda' \leq \lambda$, then $E_{\mathsf{airom}} \approx 0$ has

- *$(\lambda'/3 - C_{\mathsf{ckt},E,\mathsf{airom},\mathsf{prf}} \cdot \log(\lambda+2))$ strong bits of security regardless of n, and*
- *$(\lambda'/2 - C_{\mathsf{ckt},E,\mathsf{airom},\mathsf{prf}} \cdot \log(\lambda+2))$ strong bits of security if $n \leq \mathrm{poly}(\lambda)$.*

The same applies to lifting $E \approx \frac{1}{2}$ to $E_{\mathsf{airom}} \approx \frac{1}{2}$.

Corollary 11 follows from Theorems 2 and 3. Bounds $S, M, T, |x| \leq 2^\lambda$ and $m \leq \mathrm{poly}(\lambda)$ are applied to simplify some terms in M', T'.

Comparison with Presampling. We provide a back-of-envelope calculation of concrete reductions using presampling.[17] Refer to Theorems 1 (Lemma 4) and 2 in [11]. The number of presampled points is P. Each presampled point takes roughly $(\min\{M,n\}+m)$ bits to describe the input/output values.

For search assumption ($E \approx 0$), we set $P = (S+\lambda)Q$ for

$$\begin{aligned}
\varepsilon &= 2\varepsilon' + 2^{-\lambda}, \\
M' &\sim M + Q \cdot (S+\lambda)Q(\min\{M,n\}+m).
\end{aligned}$$

Suppose $E \approx 0$ has λ' strong bits of security, then presampling proves $E_{\mathsf{airom}} \approx 0$ for

- roughly $\lambda'/4$ strong bits of security regardless of n, and
- roughly $\lambda'/3$ strong bits of security if $n \leq \mathrm{poly}(\lambda)$.

For decision assumption ($E \approx \frac{1}{2}$), we set $P = (S+\lambda)Q/\varepsilon''$ for

$$\begin{aligned}
\varepsilon' &= \varepsilon + 2^{-\lambda} + \varepsilon'', \\
M' &\sim M + Q \cdot (S+\lambda)Q(\min\{M,n\}+m)/\varepsilon''.
\end{aligned}$$

17. We do not claim that presampling works with infinite-domain oracles. Instead, we avoid this issue by truncating the oracle to the relevant portion (input length at most M).

Suppose $E \approx \frac{1}{2}$ has λ' strong bits of security, then presampling proves $E_{\mathsf{airom}} \approx \frac{1}{2}$ for

- roughly $\lambda'/5$ strong bits of security regardless of n, and
- roughly $\lambda'/4$ strong bits of security if $n \leq \mathrm{poly}(\lambda)$.

In summary, regardless of the nature of assumption or the domain of AI-ROM, our method saves a constant factor in the loss of bit level of security.

RAM. We rely on PRF with fast evaluation. It currently only works for small domain (input length is *a priori* polynomially bounded).

Corollary 12 (from standard-model to AI-ROM, RAM model)**.** *Assuming the existence of $(2^\lambda, 2^{-\lambda})$-secure PRF with fast evaluation and fixing a positive constant $\gamma > 0$, suppose $E \overset{\varepsilon'}{\approx}_{M',T'} 0$ holds in the standard model, then $E_{\mathsf{airom}} \overset{\varepsilon}{\approx}_{S,M,T,Q} 0$ holds in AI-ROM of type either $\{0,1\}^n \to \{0,1\}^m$ or $\{0,1\}^{\leq n} \to \{0,1\}^m$ for $n \leq \mathrm{poly}(\lambda)$, if $\varepsilon \geq \varepsilon' + 2^{-\lambda}$ and*[18]

$$\begin{aligned}
M' &\geq M + C_{\mathsf{ram},E,\mathsf{airom},\mathsf{prf},\gamma} \cdot (n+1+\log m)^{1+\gamma} \times \\
&\qquad (S+\lambda+1)^{1+\gamma}, \\
T' &\geq T + (Q+1) \cdot m(n+1+\log m)^{1+\gamma} \times \\
&\qquad (\lambda+2)^{C_{\mathsf{ram},E,\mathsf{airom},\mathsf{prf},\gamma}}.
\end{aligned}$$

Suppose $E \approx 0$ has λ' strong bits of security (Definition 2) and $\lambda' \leq \lambda$, then $E_{\mathsf{airom}} \approx 0$ has $(\lambda/(1+\gamma) - C_{\mathsf{ram},E,\mathsf{airom},\mathsf{prf},\gamma} \cdot \log(\lambda+2))$ strong bits of security. The same applies to lifting $E \approx \frac{1}{2}$ to $E_{\mathsf{airom}} \approx \frac{1}{2}$.

Comparison with Presampling. We provide a back-of-envelope calculation of concrete reductions using presampling. Refer to Theorems 1 (Lemma 4) and 2 in [11]. The number of presampled points is P.

In the RAM model, the query time can be reduced by storing a perfect hash table of the presampled points in the advice. The perfect hash table takes $C(P+1)(n+m)$ bits, and hashing an input takes $(n+2)^{C_{\mathsf{ram}}}$ time.

For search assumption ($E \approx 0$), we set $P = (S+\lambda)Q$ for

$$\begin{aligned}
\varepsilon &= 2\varepsilon' + 2^{-\lambda}, \\
M' &\sim M + (S+\lambda)Q(n+m), \\
T' &\sim T + Q \cdot (n+2)^{C_{\mathsf{ram}}}.
\end{aligned}$$

Suppose $E \approx 0$ has λ' strong bits of security, then presampling proves $E_{\mathsf{airom}} \approx 0$ for roughly $\lambda'/2$ strong bits of security.

For decision assumption ($E \approx \frac{1}{2}$), we set $P = (S+\lambda)Q/\varepsilon''$ for

$$\begin{aligned}
\varepsilon' &= \varepsilon + 2^{-\lambda} + \varepsilon'', \\
M' &\sim M + (S+\lambda)Q(n+m)/\varepsilon'', \\
T' &\sim T + Q \cdot (n+2)^{C_{\mathsf{ram}}}.
\end{aligned}$$

18. The bounds simplify to the same form in both concrete and asymptotic scenarios — constants depend on the scenario.

Suppose $E \approx \frac{1}{2}$ has λ' strong bits of security, then presampling proves $E_{\text{airom}} \approx \frac{1}{2}$ for roughly $\lambda'/3$ strong bits of security.

In summary, for small-domain AI-ROM, regardless of the nature of assumption, our method can be made arbitrarily tight, while presampling always loses at least half of the bit level.

Degenerate Cases and Other Remarks. Under our formulation of bit level of security, when the truth table of AI-ROM is below roughly $2^{\lambda'/2}$ bits (e.g., $\{0,1\}^{\lambda'/2} \to \{0,1\}$), neither simulation by PRF nor presampling should be used for circuits. For RAM, the threshold size of truth table can be arbitrarily close to $2^{\lambda'}$ bits. In those cases, the brute-force reduction of hardwiring the entire truth table and the auxiliary information is better.

However, an oracle with such a small truth table often leads to even less bit level of security for primitives constructed from the oracle. In a cryptosystem built from multiple primitives, a mismatch among their security levels often indicates wasteful choices of security parameters. The situation is further complicated if those components suffer different security losses in reductions.

The restriction $\lambda' \le \lambda$ is only there because we have set the statistical error to $2^{-\lambda}$. It can be adjusted for $\lambda' > \lambda$.

6. Combining Presampling and Our Simulation

As discussed in Section 3.1, it is difficult to use our simulation to analyze security reliant on the AI-ROM, since the whole oracle is replaced by a PRF, which has little entropy. We present a simple trick for combining the powers of presampling and our simulation. Oftentimes, the security only depends on the behavior of the random oracle at points *unpredictable* to the adversary but known to the security experiment. We can use presampling to argue that the portion of the oracle relevant to security is essentially independent of the auxiliary input. Let E be the security experiment, A the adversary, $\mathcal{O}, \mathcal{O}_{\text{presamp}}, \mathcal{O}_{\text{ro}} : \mathsf{dom} \to \{0,1\}^m$ three independent random functions, ai the auxiliary input, and (D, Eval) the PRF. The hybrids typically go as follows.

1) E interacts with $A(z)$, where $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$ and E, A have access to $\mathcal{O}$.
2) E interacts with $A(z)$, where $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O}_{\text{presamp}})$ and E, A have access to $\mathcal{O}[\mathcal{T}]$. Here, $\mathcal{T} \xleftarrow{\$} \mathsf{ai}_{\text{presamp}}(\mathcal{O}_{\text{presamp}})$ is the presampled bit-fixing table per Lemma 4 with an appropriate choice of size P.
3) E interacts with $A(z)$, $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O}_{\text{presamp}})$ and E, A have access to
$$\mathcal{O}'(x) = \begin{cases} \mathcal{O}_{\text{ro}}(x), & \text{if } x \text{ is "relevant";} \\ \mathcal{O}[\mathcal{T}](x), & \text{otherwise.} \end{cases}$$
Again, $\mathcal{T} \xleftarrow{\$} \mathsf{ai}_{\text{presamp}}(\mathcal{O}_{\text{presamp}})$ is the presampling. The notion of "relevant" is specific to the application. In addition, whether a point is relevant must be decided by the first time it is queried without consulting $\mathcal{T}$. This transition holds as long as the relevant points hit $\mathcal{T}$ with only negligible probability.
4) E interacts with $A(z)$, where $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$ and E, A have access to
$$\mathcal{O}'(x) = \begin{cases} \mathcal{O}_{\text{ro}}(x), & \text{if } x \text{ is "relevant";} \\ \mathcal{O}(x), & \text{otherwise.} \end{cases}$$
This is just undoing presampling.
5) E interacts with $A(z)$, where $z \xleftarrow{\$} \widetilde{\mathsf{ai}}(k)$ for $k \xleftarrow{\$} D_{\mathsf{dom},m}$ and E, A have access to
$$\mathcal{O}'(x) = \begin{cases} \mathcal{O}_{\text{ro}}(x), & \text{if } x \text{ is "relevant";} \\ \mathsf{Eval}(k, x), & \text{otherwise.} \end{cases}$$
This is our AI-ROM simulation (Theorems 2 and 3).

In the rest of this section, we show how to encapsulate the presampling–rerouting–undoing trick into a notion of salting scheme and demonstrate its applications.

6.1. Salting Defeats Auxiliary Information

In [11], it is shown using presampling that salting generically defeats auxiliary information. However, their reduction is loose due to presampling with large table size P. The reduction can be decomposed into two phases — first an argument that the "salted portion" is independent of the auxiliary information, then a reduction to security in the ROM. The second phase is computational and suffers the tightness issue in simulating the "non-salted portion". Another downside is that due to salting, the oracle domain shrinks.

We encapsulate the first phase into a notion of *salting scheme*, whose security is roughly indifferentiability, possibly with domain extension. It sets a clear boundary between the information-theoretic technique of presampling and our computational simulation with PRF. We remark that *defining* indifferentiability for auxiliary-input models is challenging by itself [46]. Thanks to the presence of salting, our definition circumvents the difficulty and has provably secure instantiations.

Definition 14 (salting scheme)**.** Let $\mathsf{dom}(\lambda) \to \{0,1\}^{m(\lambda)}$ and $\mathsf{dom}_{\text{ro}}(\lambda) \to \{0,1\}^{m_{\text{ro}}(\lambda)}$ be the types of AI-ROM and ROM. A *salting scheme* consists of two efficient algorithms.

- $\mathsf{HashGen}(1^\lambda)$ outputs a hash key hk.
- $\mathsf{Hash}^{\mathcal{O}}(\mathsf{hk}, x_{\text{ro}})$ is deterministic and given oracle access to the AI-ROM. It takes as input hk and $x_{\text{ro}} \in \mathsf{dom}_{\text{ro}}(\lambda)$, and outputs $y_{\text{ro}} \in \{0,1\}^{m_{\text{ro}}(\lambda)}$.

Definition 15 (security of salting scheme)**.** A salting scheme is *secure* if there exists an efficient stateful simulator $\mathcal{S}$ such that $E_{\text{real}} \approx E_{\text{sim}}$, where the experiments with $(\mathsf{nu}, \mathsf{ai}, A)$ work as follows.

- In both, sample $w \xleftarrow{\$} \mathsf{nu}(\lambda)$, random $\mathcal{O} : \mathsf{dom}(\lambda) \to \{0,1\}^{m(\lambda)}$, and $z \xleftarrow{\$} \mathsf{ai}(\lambda, \mathcal{O})$.

- In E_{real}, sample $\mathsf{hk} \xleftarrow{\$} \mathsf{HashGen}(1^\lambda)$. Run and output $A^{\mathcal{O},\mathsf{Hash}(\mathsf{hk},\cdot)}(1^\lambda, w, z, \mathsf{hk})$.
- In E_{sim}, sample random $\mathcal{O}_{\text{ro}} : \mathsf{dom}_{\text{ro}}(\lambda) \to \{0,1\}^{m_{\text{ro}}(\lambda)}$ and $\mathsf{hk} \xleftarrow{\$} \mathcal{S}(1^\lambda)$. Run and output $A^{\mathcal{S}^{\mathcal{O},\mathcal{O}_{\text{ro}}},\mathcal{O}_{\text{ro}}}(1^\lambda, w, z, \mathsf{hk})$.

Security Properties as Indistinguishability. An experiment that is black-box and straight-line in the adversary corresponds to "application" in [11; Definition 3], which captures either unpredictability or indistinguishability (together, a security property) of some primitive implicit in the description of the experiment — for example, an experiment in the ROM might play the IND-CPA security game of RSA-OAEP with the adversary, hence captures the IND-CPA security of RSA-OAEP. Recall that in Section 5.2, we explained how to formulate search/decision assumptions as indistinguishability. The same applies here — unpredictability is modeled as $E \approx 0$ and indistinguishability is modeled as $E \approx \frac{1}{2}$. We show that security properties in the ROM can be lifted to AI-ROM with the help of a salting scheme.

Consider ROM of type $\mathsf{dom}_{\text{ro}} \to \{0,1\}^{m_{\text{ro}}}$ and let $E^{\mathcal{O}_{\text{ro}}}$ be an experiment that is black-box and straight-line in the adversary, which is given the same oracle given to E. Let $M'_A, T'_A, Q'_A, \varepsilon'$ be the parameters of $E \approx 0$ or $E \approx \frac{1}{2}$, and T'_E, Q'_E the complexities of E itself (the experiment is assumed to be uniform).

Let $\mathsf{dom} \to \{0,1\}^m$ be the type of AI-ROM and $(\mathsf{HashGen}, \mathsf{Hash})$ a secure salting scheme. Define $E^{\mathcal{O}}_{\text{airom}}$ working with $(\mathsf{nu}, \mathsf{ai}, A^{\mathcal{O}})$ as follows.

- Sample $w \xleftarrow{\$} \mathsf{nu}()$ and $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$. Run $\mathsf{hk} \xleftarrow{\$} \mathsf{HashGen}()$.
- Run and output $E^{\mathsf{Hash}(\mathsf{hk},\cdot)}(A^{\mathcal{O}}(w, z, \mathsf{hk}))$.

Let the parameters of $E_{\text{airom}} \approx 0$ or $E_{\text{airom}} \approx \frac{1}{2}$ be S, M, T, Q, ε.

Denote by $\mathcal{S}$ the simulator of the salting scheme and $S_{\text{salt}}, M_{\text{salt}}, T_{\text{salt}}, Q_{\text{salt}}, \varepsilon_{\text{salt}}$ the parameters of its security. Let (D, Eval) be a $(2^\lambda, \varepsilon_{\text{prf}})$-secure PRF with key length $|k|$. Let $T_{\mathcal{S}}$ be the per-query time of $\mathcal{S}^{\mathsf{Eval}(k,\cdot),\mathcal{O}_{\text{ro}}}$. As usual, all integer-valued parameters are assumed to be bounded by 2^λ.

Theorem 13 (¶). *In the RAM model, the asymptotic versions of $E_{\text{airom}} \approx 0$ or $E_{\text{airom}} \approx \frac{1}{2}$ can be derived if $\varepsilon \geq \varepsilon_{\text{salt}} + 2^{-\lambda} + \varepsilon_{\text{prf}}(\lambda_{\text{prf}}) + \varepsilon'$, and (for salting),*

$$S_{\text{salt}} \geq S, \quad M_{\text{salt}} \geq M, \quad T_{\text{salt}} \geq T + T'_E, \quad Q_{\text{salt}} \geq Q + Q'_E,$$

and (for simulation using PRF) $\lambda_{\text{prf}} \geq S + C_{\text{ram}}(\lambda + 1)$, and (for reduction to E),

$$\begin{aligned} M'_A &\geq M + S + |k| + C(\lambda + 1), \\ T'_A &\geq T + Q \cdot T_{\mathcal{S}} + C_{\text{ram}}(\lambda + 1), \\ Q'_A &\geq Q. \end{aligned}$$

Compared with [11], we incur less loss in reduction to the (computational) security in ROM, thanks to the efficient simulation of AI-ROM.

Proof (Theorem 13). Always sample uniformly random $\mathcal{O} : \mathsf{dom} \to \{0,1\}^m$ and $\mathcal{O}_{\text{ro}} : \mathsf{dom}_{\text{ro}} \to \{0,1\}^{m_{\text{ro}}}$. We go through the following hybrids.

- H_0 is E_{airom}, i.e., run and output
$$E^{\mathsf{Hash}(\mathsf{hk},\cdot)}(A^{\mathcal{O}}(w, z, \mathsf{hk}))$$
with $w \xleftarrow{\$} \mathsf{nu}()$, $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$, $\mathsf{hk} \xleftarrow{\$} \mathsf{HashGen}()$.
- In H_1, we replace $\mathsf{Hash}(\mathsf{hk}, \cdot)$ by $\mathcal{O}_{\text{ro}}$. Run and output
$$E^{\mathcal{O}_{\text{ro}}}(A^{\mathcal{S}^{\mathcal{O},\mathcal{O}_{\text{ro}}}}(w, z, \mathsf{hk}))$$
with $w \xleftarrow{\$} \mathsf{nu}()$, $z \xleftarrow{\$} \mathsf{ai}(\mathcal{O})$, $\mathsf{hk} \xleftarrow{\$} \mathcal{S}()$.

  $\mathsf{H}_0 \approx \mathsf{H}_1$ follows from the conditions for salting and incurs advantage error $\varepsilon_{\text{salt}}$.
- In H_2, we simulate $(\mathcal{O}, z)$ using PRF. Sample $k \xleftarrow{\$} D_{\lambda_{\text{prf}},\mathsf{dom},m}$ and run and output
$$E^{\mathcal{O}_{\text{ro}}}(A^{\mathcal{S}^{\mathsf{Eval}(k,\cdot),\mathcal{O}_{\text{ro}}}}(w, z, \mathsf{hk}))$$
with $w \xleftarrow{\$} \mathsf{nu}()$, $z \xleftarrow{\$} \widetilde{\mathsf{ai}}(k)$, $\mathsf{hk} \xleftarrow{\$} \mathcal{S}()$.

  $\mathsf{H}_1 \approx \mathsf{H}_2$ follows from Theorem 3 and incurs advantage error $(2^{-\lambda} + \varepsilon_{\text{prf}}(\lambda_{\text{prf}}))$.

By our choice of parameters, the advantage of $\mathsf{H}_2 \approx 0$ or $\mathsf{H}_2 \approx \frac{1}{2}$ is bounded by ε'. The proof completes by hybrid argument. □

Example Instantiations. The simplest instantiation of a salting scheme in AI-ROM of type $\{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ that creates a random oracle of type $\{0,1\}^\lambda \to \{0,1\}^\lambda$ is as follows.

- $\mathsf{HashGen}()$ samples and outputs $\mathsf{hk} \xleftarrow{\$} \{0,1\}^\lambda$.
- $\mathsf{Hash}^{\mathcal{O}}(\mathsf{hk}, x)$ outputs $\mathcal{O}(\mathsf{hk} \| x)$.

Its security proof follows the presampling–rerouting–undoing blueprint explained at the beginning of this section. The simulator simply chooses to direct query to $\mathcal{O}$ or $\mathcal{O}_{\text{ro}}$ depending on whether the prefix matches hk. Therefore, there is essentially no simulator overhead during reduction to ROM security, implying an arbitrarily tight reduction using PRF with fast evaluation.

Any salting scheme can be composed with an indifferentiable domain extension scheme as needed.

6.2. Non-Interactive Zero-Knowledge in AI-ROM

We show how to construct NIZK in AI-ROM starting from NIZK in ROM with statistical soundness and computational zero-knowledge against polynomial-time adversaries.

Definition 16 (NIZK in ROM or AI-ROM)**.** Let $L \subseteq \{0,1\}^*$ and $\mathcal{O}$ be either AI-ROM or ROM. A *non-interactive zero-knowledge* (NIZK) proof for L consists of two algorithms:

- $P^{\mathcal{O}}(1^\lambda, x)$ takes $x \in L$ as input and outputs a proof $\pi \in \{0,1\}^*$. The algorithm P does not have to be efficient, but π must be polynomially long in $|x|, \lambda$.

- $V^{\mathcal{O}}(1^\lambda, x, \pi)$ takes x, π as input and outputs a single bit. It is efficient.

The scheme must be *complete*, i.e., for all $\lambda \in \mathbb{N}$ and $x \in L$, it holds that $\Pr[V^{\mathcal{O}}(1^\lambda, x, P^{\mathcal{O}}(1^\lambda, x))] = 1$.

The NIZK is *statistically sound* if for all $\lambda \in \mathbb{N}$ and $x \notin L$ and inefficient algorithm $\widetilde{P}^{\mathcal{O}}$, it holds that $\Pr[V^{\mathcal{O}}(1^\lambda, x, \widetilde{P}^{\mathcal{O}}(x)) = 1] \leq 2^{-\lambda}$.

Definition 17 (zero-knowledge in ROM)**.** A NIZK for L in ROM is zero-knowledge if there exists an efficient stateful simulator $\mathcal{S}$ such that $E_{\text{real}} \approx E_{\text{sim}}$, where the experiments with adversary A work as follows.

- Sample $w \xleftarrow{\$} \mathsf{nu}(\lambda)$.
- In E_{real}, launch $A^{\mathcal{O}_{\text{ro}}}(w)$ and receive from it $x \in L$. Run $\pi \xleftarrow{\$} P^{\mathcal{O}_{\text{ro}}}(1^\lambda, x)$ and send π to A.
- In E_{sim}, launch $A^{\mathcal{S}}(w)$ and receive from it $x \in L$. Run $\pi \xleftarrow{\$} \mathcal{S}(x)$ and send π to A.
- A continues with its oracle and outputs some string, which is the output of the experiment.

$E_{\text{real}} \approx E_{\text{sim}}$ is a standard-model indistinguishability, which means the distinguisher taking the output of one of the experiments has no oracle access.

NIZK in ROM can be obtained by taking any NIZK in the common *random* string model, such as [47] and [48], and use the output of the random oracle at a few fixed points as the CRS.

Definition 18 (zero-knowledge in AI-ROM)**.** A NIZK for L in AI-ROM is zero-knowledge if for all $(\mathsf{nu}, \mathsf{ai}, A)$, there exists an efficient simulator $(\widetilde{\mathsf{nu}}, \mathcal{S})$ such that $E_{\text{real}} \approx E_{\text{sim}}$, where the experiments work as follows.

- In E_{real}, sample $w \xleftarrow{\$} \mathsf{nu}(\lambda)$, random $\mathcal{O}$, $z \xleftarrow{\$} \mathsf{ai}(\lambda, \mathcal{O})$. Launch $A^{\mathcal{O}}(1^\lambda, w, z)$ and receive from it $x \in L$. Run $\pi \xleftarrow{\$} P^{\mathcal{O}}(1^\lambda, x)$ and send π to A. The adversary A continues to run with oracle access to $\mathcal{O}$, and eventually outputs some string, which is the output of the experiment.
- In E_{sim}, sample $\widetilde{w} \xleftarrow{\$} \widetilde{\mathsf{nu}}(\lambda)$. Run and output $\mathcal{S}(1^\lambda, \widetilde{w})$.

$E_{\text{real}} \approx E_{\text{sim}}$ is a standard-model indistinguishability, which means the distinguisher taking the output of one of the experiments has no oracle nor oracle-dependent auxiliary input.

We remark that Definition 18 insists that the simulator be standard-model efficient, without the help of AI-ROM.

NIZK in AI-ROM from NIZK in ROM. Let $(P_{\text{ro}}, V_{\text{ro}})$ a NIZK for L in ROM $\{0,1\}^\lambda \to \{0,1\}^\lambda$. The following is a NIZK for L in AI-ROM $\{0,1\}^{3\lambda} \to \{0,1\}^{2\lambda}$.

- $P^{\mathcal{O}}(1^\lambda, x)$ samples $\mathsf{hk} \xleftarrow{\$} \{0,1\}^\lambda$, runs $\pi_{\text{ro}} \xleftarrow{\$} P_{\text{ro}}^{\mathcal{O}(\mathsf{hk}\|\cdot)(1^{2\lambda}, x)}$, and outputs $(\mathsf{hk}, \pi_{\text{ro}})$.
- $V^{\mathcal{O}}(1^\lambda, x, \mathsf{hk}, \pi_{\text{ro}})$ runs $V^{\mathcal{O}(\mathsf{hk}\|\cdot)}(1^{2\lambda}, x, \pi_{\text{ro}})$.

Soundness follows by a union bound. The simulator works as follows.

- Let $\widetilde{\mathsf{nu}}$ be the standard-model advice promised by Theorem 3.
- On input $\widetilde{w} = (w, z, k)$, sample $\mathsf{hk} \xleftarrow{\$} \{0,1\}^\lambda$, launch $\mathcal{S}_{\text{ro}}$, and run $A^{\mathcal{O}}(w, z)$ with

$$\mathcal{O}(\mathsf{hk}' \| y') = \begin{cases} \mathcal{S}_{\text{ro}}(y'), & \mathsf{hk} = \mathsf{hk}'; \\ \mathsf{Eval}(k, \mathsf{hk}' \| y'), & \text{otherwise.} \end{cases}$$

- When A chooses x, send it to $\mathcal{S}_{\text{ro}}$ and receive π_{ro}. Return $(\mathsf{hk}, \pi_{\text{ro}})$ to $\mathcal{A}$.
- Continue A with its oracle and output whatever A outputs.

Acknowledgments. Yevgeniy Dodis was supported by NSF CNS-2055578 and gifts from JP Morgan, Protocol Labs, Stellar, and Algorand Foundation. Aayush Jain was supported by gifts from CyLab of CMU and Google. Huijia Lin and Ji Luo were supported by NSF CNS-1936825 (CAREER), NSF CNS-2026774, a JP Morgan AI Research Award, a Cisco Research Award, and a Simons Collaboration on the Theory of Algorithmic Fairness. Daniel Wichs was supported by NSF CNS-1750795, NSF CNS-2055510, and the JP Morgan Faculty Research Award. The authors thank the anonymous reviewers of FOCS 2024 for their constructive feedback.

References

[1] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *ACM CCS 93*, D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, Eds. ACM Press, Nov. 1993, pp. 62–73.

[2] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited (preliminary version)," in *30th ACM STOC*. ACM Press, May 1998, pp. 209–218.

[3] S. Goldwasser and Y. T. Kalai, "On the (in)security of the Fiat-Shamir paradigm," in *44th FOCS*. IEEE Computer Society Press, Oct. 2003, pp. 102–115.

[4] M. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Transactions on Information Theory*, vol. 26, no. 4, pp. 401–406, 1980. [Online]. Available: https://doi.org/10.1109/TIT.1980.1056220

[5] A. Fiat and M. Naor, "Rigorous time/space tradeoffs for inverting functions," in *23rd ACM STOC*. ACM Press, May 1991, pp. 534–541.

[6] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off," in *CRYPTO 2003*, ser. LNCS, D. Boneh, Ed., vol. 2729. Springer, Heidelberg, Aug. 2003, pp. 617–630.

[7] N. Alon, O. Goldreich, J. Håstad, and R. Peralta, "Simple constructions of almost k-wise independent random variables," in *31st FOCS*. IEEE Computer Society Press, Oct. 1990, pp. 544–553.

[8] A. De, L. Trevisan, and M. Tulsiani, "Time space tradeoffs for attacks against one-way functions and PRGs," in *CRYPTO 2010*, ser. LNCS, T. Rabin, Ed., vol. 6223. Springer, Heidelberg, Aug. 2010, pp. 649–665.

[9] D. Unruh, "Random oracles and auxiliary input," in *CRYPTO 2007*, ser. LNCS, A. Menezes, Ed., vol. 4622. Springer, Heidelberg, Aug. 2007, pp. 205–223.

[10] Y. Dodis, S. Guo, and J. Katz, "Fixing cracks in the concrete: Random oracles with auxiliary input, revisited," in *EUROCRYPT 2017, Part II*, ser. LNCS, J.-S. Coron and J. B. Nielsen, Eds., vol. 10211. Springer, Heidelberg, Apr. / May 2017, pp. 473–495.

[11] S. Coretti, Y. Dodis, S. Guo, and J. P. Steinberger, "Random oracles and non-uniformity," in *EUROCRYPT 2018, Part I*, ser. LNCS, J. B. Nielsen and V. Rijmen, Eds., vol. 10820. Springer, Heidelberg, Apr. / May 2018, pp. 227–258.

[12] R. Gennaro and L. Trevisan, "Lower bounds on the efficiency of generic cryptographic constructions," in *41st FOCS*. IEEE Computer Society Press, Nov. 2000, pp. 305–313.

[13] N. Gravin, S. Guo, T. C. Kwok, and P. Lu, "Concentration bounds for almost k-wise independence with applications to non-uniform security," in *32nd SODA*, D. Marx, Ed. ACM-SIAM, Jan. 2021, pp. 2404–2423.

[14] D. Unruh, "Random oracles and auxiliary input," Cryptology ePrint Archive, Report 2007/168, 2007, https://eprint.iacr.org/2007/168.

[15] D. Jetchev and K. Pietrzak, "How to fake auxiliary input," in *TCC 2014*, ser. LNCS, Y. Lindell, Ed., vol. 8349. Springer, Heidelberg, Feb. 2014, pp. 566–590.

[16] Y.-H. Chen, K.-M. Chung, and J.-J. Liao, "On the complexity of simulating auxiliary input," in *EUROCRYPT 2018, Part III*, ser. LNCS, J. B. Nielsen and V. Rijmen, Eds., vol. 10822. Springer, Heidelberg, Apr. / May 2018, pp. 371–390.

[17] C. Gentry and D. Wichs, "Separating succinct non-interactive arguments from all falsifiable assumptions," in *43rd ACM STOC*, L. Fortnow and S. P. Vadhan, Eds. ACM Press, Jun. 2011, pp. 99–108.

[18] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *Journal of the ACM*, vol. 33, no. 4, pp. 792–807, Oct. 1986.

[19] O. Goldreich, "Candidate one-way functions based on expander graphs," 2000.

[20] K. S. Kedlaya and C. Umans, "Fast polynomial factorization and modular composition," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1767–1802, 2011. [Online]. Available: https://doi.org/10.1137/08073408X

[21] W.-K. Lin, E. Mook, and D. Wichs, "Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE," in *STOC 2023*, 2023, pp. 595–608.

[22] R. Morris and K. Thompson, "Password security: A case history," *Communications of the ACM*, vol. 22, no. 11, pp. 594–597, 1979. [Online]. Available: https://doi.org/10.1145/359168.359172

[23] B. Applebaum and P. Raykov, "Fast pseudorandom functions based on expander graphs," in *TCC 2016-B, Part I*, ser. LNCS, M. Hirt and A. D. Smith, Eds., vol. 9985. Springer, Heidelberg, Oct. / Nov. 2016, pp. 27–56.

[24] M. Cryan and P. B. Miltersen, "On pseudorandom generators in NC^0," in *MFCS 2001*, ser. Lecture Notes in Computer Science, J. Sgall, A. Pultr, and P. Kolman, Eds., vol. 2136. Springer, 2001, pp. 272–284.

[25] U. Feige, "Relations between average case complexity and approximation complexity," in *34th ACM STOC*. ACM Press, May 2002, pp. 534–543.

[26] E. Mossel, A. Shpilka, and L. Trevisan, "On e-biased generators in NC0," in *44th FOCS*. IEEE Computer Society Press, Oct. 2003, pp. 136–145.

[27] U. Feige, J. H. Kim, and E. Ofek, "Witnesses for non-satisfiability of dense random 3CNF formulas," in *47th FOCS*. IEEE Computer Society Press, Oct. 2006, pp. 497–508.

[28] J. Cook, O. Etesami, R. Miller, and L. Trevisan, "Goldreich's one-way function candidate and myopic backtracking algorithms," in *TCC 2009*, ser. LNCS, O. Reingold, Ed., vol. 5444. Springer, Heidelberg, Mar. 2009, pp. 521–538.

[29] A. Bogdanov and Y. Qiao, "On the security of Goldreich's one-way function," in *APPROX/RANDOM 2009*, ser. Lecture Notes in Computer Science, I. Dinur, K. Jansen, J. Naor, and J. D. P. Rolim, Eds., vol. 5687. Springer, 2009, pp. 392–405. [Online]. Available: https://doi.org/10.1007/978-3-642-03685-9_30

[30] B. Applebaum, B. Barak, and A. Wigderson, "Public-key cryptography from different assumptions," in *42nd ACM STOC*, L. J. Schulman, Ed. ACM Press, Jun. 2010, pp. 171–180.

[31] B. Applebaum, A. Bogdanov, and A. Rosen, "A dichotomy for local small-bias generators," in *TCC 2012*, ser. LNCS, R. Cramer, Ed., vol. 7194. Springer, Heidelberg, Mar. 2012, pp. 600–617.

[32] A. Bogdanov and Y. Qiao, "On the security of Goldreich's one-way function," *Computational Complexity*, vol. 21, no. 1, pp. 83–127, 2012.

[33] B. Applebaum, "Pseudorandom generators with long stretch and low locality from random local one-way functions," in *44th ACM STOC*, H. J. Karloff and T. Pitassi, Eds. ACM Press, May 2012, pp. 805–816.

[34] ——, "Pseudorandom generators with long stretch and low locality from random local one-way functions," *SIAM Journal on Computing*, vol. 42, no. 5, pp. 2008–2037, 2013. [Online]. Available: https://doi.org/10.1137/120884857

[35] R. O'Donnell and D. Witmer, "Goldreich's PRG: Evidence for near-optimal polynomial stretch," in *CCC 2014*. IEEE Computer Society, 2014, pp. 1–12. [Online]. Available: https://doi.org/10.1109/CCC.2014.9

[36] B. Applebaum and S. Lovett, "Algebraic attacks against random local functions and their countermeasures," in *48th ACM STOC*, D. Wichs and Y. Mansour, Eds. ACM Press, Jun. 2016, pp. 1087–1100.

[37] E. Miles and E. Viola, "Substitution-permutation networks, pseudorandom functions, and natural proofs," *Journal of the ACM*, vol. 62, no. 6, pp. 46:1–46:29, 2015. [Online]. Available: https://doi.org/10.1145/2792978

[38] D. Micciancio and M. Walter, "On the bit security of cryptographic primitives," in *EUROCRYPT 2018, Part I*, ser. LNCS, J. B. Nielsen and V. Rijmen, Eds., vol. 10820. Springer, Heidelberg, Apr. / May 2018, pp. 3–28.

[39] Y. Dodis, A. Jain, H. Lin, J. Luo, and D. Wichs, "How to simulate random oracles with auxiliary input," 2024, to appear in Cryptology ePrint Archive.

[40] S. Guo, Q. Li, Q. Liu, and J. Zhang, "Unifying presampling via concentration bounds," in *TCC 2021, Part I*, ser. LNCS, K. Nissim and B. Waters, Eds., vol. 13042. Springer, Heidelberg, Nov. 2021, pp. 177–208.

[41] H. Cramér, "On the order of magnitude of the difference between consecutive prime numbers," *Acta Arithmetica*, vol. 2, no. 1, pp. 23–46, 0 1936.

[42] M. Agrawal, N. Kayal, and N. Saxena, "PRIMES is in P," *Annals of Mathematics*, vol. 160, no. 2, pp. 781–793, 2004. [Online]. Available: https://doi.org/10.4007/annals.2004.160.781

[43] M. Clegg, J. Edmonds, and R. Impagliazzo, "Using the Groebner basis algorithm to find proofs of unsatisfiability," in *28th ACM STOC*. ACM Press, May 1996, pp. 174–183.

[44] P. K. Kothari, R. Mori, R. O'Donnell, and D. Witmer, "Sum of squares lower bounds for refuting any CSP," in *49th ACM STOC*, H. Hatami, P. McKenzie, and V. King, Eds. ACM Press, Jun. 2017, pp. 132–145.

[45] S. R. Allen, R. O'Donnell, and D. Witmer, "How to refute a random CSP," in *56th FOCS*, V. Guruswami, Ed. IEEE Computer Society Press, Oct. 2015, pp. 689–708.

[46] Y. Dodis, P. Farshim, S. Mazaheri, and S. Tessaro, "Towards defeating backdoored random oracles: Indifferentiability with bounded adaptivity," in *TCC 2020, Part III*, ser. LNCS, R. Pass and K. Pietrzak, Eds., vol. 12552. Springer, Heidelberg, Nov. 2020, pp. 241–273.

[47] U. Feige, D. Lapidot, and A. Shamir, "Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract)," in *31st FOCS*. IEEE Computer Society Press, Oct. 1990, pp. 308–317.

[48] J. Groth, R. Ostrovsky, and A. Sahai, "Non-interactive zaps and new techniques for NIZK," in *CRYPTO 2006*, ser. LNCS, C. Dwork, Ed., vol. 4117. Springer, Heidelberg, Aug. 2006, pp. 97–111.