A Computational Category-Theoretic Approach to Cryptography and Average-Case Complexity

Hart Montgomery[1] and Sikhar Patranabis[2, *]

[1]Linux Foundation, hmontgomery@linuxfoundation.org
[2]IBM Research India, sikhar.patranabis@ibm.com

Received: 22nd June 2023 | Accepted: 15th July 2023

Abstract
We study of cryptography and average-case complexity through the lens of what we call computational *category theory, wherein we consider category-theoretic objects and relationships endowed with computational hardness properties. We show that such a computational category-theoretic approach provides many novel insights into the mathematical structure inherent to certain cryptographic primitives and reductions in average-case complexity:*

- *We show how to model cryptographic primitives as computational category-theoretic diagrams, where each such diagram is described using category-theoretic objects and relationships endowed with computational hardness properties. This yields a novel approach to understanding the mathematical structure that is inherent to any given primitive.*
- *We also prove that any "hard" family of problems with a certain type of* randomized self-reduction *implies the existence of a monoid equipped with a natural notion of one-wayness, thus demonstrating that some mathematical structure is inherent to the existence of these kinds of randomized self-reductions.*
- *We further extend these observations to a certain family of collision-resistant hash functions (CRHFs): we show that any CRHF where preimage resistance is reducible to collision resistance with a particular type of reduction also implies a one-way monoid. These statements prove that even certain Minicrypt primitives with "good" reductions also imply mathematical structure, which is (to our knowledge) a novel type of proof.*

Keywords: Category theory, Computational complexity, Cryptographic foundations

2010 Mathematics Subject Classification: 94A60, 11Y40

1 INTRODUCTION

The study of theory of cryptography has in recent years focused on the "space exploration" aspect of research: in other words, what *can* we build from believable assumptions? This has led to some tremendous results: fully homomorphic encryption [38] and indistinguishability obfuscation (iO) [35, 50] are two of the more notable examples. We have a great deal of knowledge about what is (theoretically) possible, and it is not inconceivable to believe that we are approaching the limits of this kind of study, at least for classical models of computation. For instance, we have plausible candidates for iO constructions, but we know that virtual-black box (VBB) obfuscation is impossible [14]; there is only a small gap in our knowledge here of what is possible and what is not.

On the other hand, much less work has been done simplifying constructions *from a theoretical perspective*. While there have been some notable exceptions (e.g. the conceptually simpler FHE scheme presented in [39]), most work and researchers in theory have focused on constructions of cryptographic primitives with novel properties. There has seemingly been even less work done that is focused on the "whys" of cryptography: why can we build constructions from some assumptions but not from others, for instance? While cryptographic separations and reductions between primitives can partially answer these questions (and we discuss these in more detail later), we still do not find many such results intuitive.

Ultimately, we think that *simplifying the underlying principles of theoretical cryptography* and understanding *why things work the way they do* are two of the more neglected areas in cryptographic research. We believe that improving the community's knowledge in these areas would make cryptography more understandable and would benefit practitioners, who often struggle to understand difficult theoretical concepts, which sometimes causes critical mistakes [44]. The machine learning community has an entire subfield focused on explainable AI [26], and we think

*Corresponding Author: sikhar.patranabis@ibm.com

that addressing cryptographic understandability could have a similar impact on things like, say, real-world privacy as explainable AI has had on algorithmic bias.

These are obviously very large problems, and we cannot hope to address them in a single paper. In fact, where should we even begin? Our starting point will be how general theoretical computer scientists attempt to address "why" things are they way they are: complexity theory.

Complexity Theory. The study of complexity theory has enabled computer scientists to understand *why* certain problems are hard, and why others are easy. From the original works on NP completeness [24, 52] to more recent works on fine-grained complexity [70], complexity theory has helped explain why some results are seemingly optimal, and some goals are most likely impossible.

With some notable exceptions [47, 28, 17, 1, 2, 23] [33], most complexity theory results focus primarily on *worst case complexity*. But for many applications where complexity theory might be helpful worst case complexity isn't especially illuminating: we need *average case complexity*. Cryptography is perhaps the most notable example. For instance, rather than investigating whether *there exists* a secure key exchange protocol or constructing a function family where *some* functions are one-way, cryptographers are usually more interested in actually realizing secure key exchange protocols or one-way functions with concrete parameters. But there are many other practical examples (such as industrial SAT solvers [60, 27]) where average case complexity is significantly more important than worst case complexity, and it wouldn't be too much of a stretch to call average case complexity the more "practical" form of complexity theory.

There have been a number of recent results that focus on average-case complexity recently, and some of them have exciting applications to cryptography. For instance, there has been a line of work [9, 45, 66] culminating in a proof that shows the equivalence of the existence of one-way functions and mildly hard-on-average Kolmogorov complexity [54], in a way extending [40], and an exciting recent work extending this to time-bounded Kolmogorov Complexity [55]. To our knowledge, this is the first (and only) characterization of cryptographic primitives in terms of natural, complexity-theoretic assumptions. However, all of these works focus exclusively on one-way functions and completely leave open the problem of naturally characterizing public-key cryptography in a complexity-theoretic way.

In fact, there is almost no complexity-theoretic style of work that focuses on public key primitives. While there has been extensive work on black-box impossibility results in cryptography [49, 14, 63, 13, 57, 30, 36] which has substantially helped cryptographers understand how certain primitives are related, these results have generally been distinct and there has been no way to unify the intuition behind all of these black-box separations. It would be interesting if we could analyze the relationship between cryptographic primitives in some other way than by black-box reductions or separations.

Categorizing Cryptoprimitives by Structure. The idea of separating public-key cryptography from symmetric-key cryptography using mathematical structure has been around for quite some time: Barak mentions this in "The Complexity of public-key cryptography" [12]. As he puts it, "... it seems that you can't throw a rock without hitting a one-way function" but public-key cryptography is somehow "special." Barak implicitly argues that there is some mathematical structure inherent in public-key cryptography: "One way to phrase the question we are asking is to understand what type of structure is needed for public-key cryptography." But putting any formal rigor behind this statement has so far been difficult.

There have been some works which show connections between particular mathematical structures and cryptogrpahy. Hohenberger showed that pseudo-free groups had numerous cryptographic applications [46] which led to several follow-up works [65, 22]. Other works [51, 8] focused on buliding cryptography from "hard" group actions and some papers focusing on Braid group cryptography have led to interesting observations onmathematical structure and cryptography [34, 5].

There have been a few works [7, 6] focused more directly on the categorization of cryptographic primitives by mathematical structure: informally speaking, the authors of these papers show that certain primitives in the world of Minicrypt [47] (i.e., one-way functions, weak unpredictable functions, and weak pseudorandom functions) that are *homomorphic* between the input space (or the key space if it exists) and the output space directly imply the existence of many cryptographic primitives. The power of the homomorphism–essentially, the amount of mathematical structure in the primitive–dictates the power of the cryptographic primitives that can be built: group-homomorphic weak PRFs directly imply essentially any cryptoprimitive that one could build from the DDH assumption, while ring-homomorphic weak PRFs imply more powerful constructions such as fully homomorphic encryption (FHE) [38]. This enables the authors to divide the world of Cryptomania [47] into "continents" based on mathematical structure.

A drawback of these works is that they are purely *constructive*: they establish that simple primitives endowed with extra structure can be used to build powerful cryptographic primitives, but what we would ideally like to know is *if mathematical structure is inherently implied by these powerful cryptographic primitives and thus, strictly*

necessary. But these works do provide a hint towards a possible solution for a complexity-theoretic treatment of public key primitives and assumptions: perhaps mathematical structure is the key.

Analyzing Cryptographic Reductions. Another topic that has received even less complexity-theoretic treatment than public-key cryptography is the structural requirements of cryptographic reductions [31, 32] (or characterizations of problems with cryptographic reductions)[1]. It is obviously highly desirable to base the hardness of cryptographic primitives on problems that have *proofs* of average-case hardness [20]. Proofs of average-case hardness could take a number of forms: some assumptions (such as Learning With Errors [62]) reduce average-case hardness from some worst-case problem. Other assumptions (such as DDH) employ a randomized self-reduction, showing that an adversary that can solve a random instance of a problem can be used to solve any arbitrary instance of the problem.

Currently, many commonly used, practically efficient symmetric-key primitives (e.g., SHA or AES) lack such cryptographic reductions. While it would be desirable to replace these primitives with similarly efficient symmetric-key primitives with reductions (or find reductions for these practically used primitives), basing symmetric-key cryptography on assumptions with known average-case reductions results in much less efficient constructions [41]. A natural question to ask is the following: can we build symmetric-key primitives as efficient (and probably as mathematically *un*structured) as those used in practice today, such that these efficient primitives also have average-case reductions? In other words, what kinds of hard problems permit average-case reductions?

An alternative line of work [10, 43, 42, 58, 48] focuses on showing limits on the power of certain restricted classes of reductions. In particular, there exist works [29, 3, 4, 16, 15] that have examined the implications of average-case reductions on the structure of the underlying hard problems. However, the view of "structure" taken by these works is somewhat different compared to our view of "structure"; in particular, these works view "structure" from a complexity-theoretic perspective (e.g., whether a certain problem is in **AM** ∩ **coAM**), while we view "structure" from a more algebraic perspective. Concretely, we are interested in what kind of algebraic structure is implied by certain classes of average-case reductions, and what kind of mathematical tools can we rely on to formally study such implications?

Cryptography in a Nutshell. All of this discussion leads us to the following core observation: almost all commonly studied public-key cryptographic primitives can be viewed primarily as a combination of two central objects: **mathematical structure** and **average-case complexity**. In other words, if we ignore the semantics (i.e. the application for which a cryptographic construction is designed) and just examine what is required from the perspective of both (mathematical) structure and cryptographic security, we believe that there is a simple unifying viewpoint to the study of public-key cryptographic constructions through the lens of mathematical structure and average-case complexity.

This is how we arrived at *computational category theory*. Category theory is essentially the primary way in which mathematical structure is examined, and attaching computational assumptions to mathematically structured primitives, as we will show later, gives us cryptographic primitives. So we believe that the term "computational category theory" captures the objects we work with in the study of theoretical cryptography.

We are not arguing that this is the only way to study the relationship between mathematical structure and cryptography. We have to bend standard category theoretic conventions in order to make our computational definitions work, and there might be other, better ways that allow for more concise definitions and relationships. But we think, as a whole, this area merits study and this notion of computational category theory is an entirely reasonable starting point.

1.1 CATEGORY THEORY

If we are going to analyze the mathematical structure of various cryptographic objects, then we will need to incorporate the appropriate tools from mathematics. *Category theory*, informally, is the branch of mathematics focused on studying mathematical structure through the lens of collections of objects, arrows between them, and higher-order relations on these collections of objects and arrows. The basic object in category theory is a *category*, which is represented as a set of objects and (directed) arrows between objects in the set.[2]

Functors are maps between categories that preserve structure: functors preserve identity morphisms (i.e., arrows) and composition of morphisms between categories. In other words, a functor $T : C \to B$ between two categories C and B has the property that, for the identity arrows 1_C and 1_B, $T(1_C) = 1_B$, and for arbitrary arrows g and f, we have $Tg \circ Tf = T(g \circ f)$, assuming the composite arrow $g \circ f$ exists (it doesn't necessarily have to, as in the case of forgetful functors). We can similarly define *natural transformations* as structure preserving maps between functors.

[1] There has been some category theoretic work on composing reductions; we discuss that later.

[2] We make a number of simplifying assumptions about category-theoretic concepts in this section (e.g. we assume categories are finite). In the body of the paper, we take a much more rigorous approach and spell out all of our assumptions.

We cannot comprehensively discuss category theory in this paper; for that, we refer the reader to the classic text of Mac Lane [56] or the more recent book of Leinster [53]. We will use almost no complicated or non-basic category theory in this paper, so a reader with even rudimentary knowledge of category theory should be able to understand the content. However, we do define all of the primitives from category theory that are integral to our constructions formally in section 3, and we encourage a reader totally unfamiliar category theory to peruse through those.

1.2 OUR CONTRIBUTIONS

In this work, we formalize a notion of "computational category theory" and show that it can be used to make several novel observations about cryptography and average-case complexity. This framework and these observations have the potential to be useful in the study of cryptography and average-case complexity in a wide variety of different ways.

Modeling Cryptographic Primitives as Diagrams. Our first contribution is formally showing how to model cryptographic primitives as computational category-theoretic *diagrams*. We view "data" in the form of (potentially structured) categories, and computational primitives as functors[3]. These "computational" functors are endowed with particular hardness properties that let us model cryptographic primitives. As mentioned earlier, this yields a novel approach to understanding the mathematical structure inherent to any given primitive. We formally show how to model cryptoprimitives like key exchange, signatures, and identity-based encryption in this computational category-theoretic framework.

The Structure of Randomized Self-Reductions. Our computational category-theoretic framework can also be used in the context of average-case complexity. We prove that any "hard" family of problems (i.e., a one-way function) with a Levin-style[4] *randomized self-reduction* implies the existence of a monoid equipped with a natural notion of one-wayness. A long-standing goal in cryptography has been to construct an efficient symmetric-key encryption primitive (e.g., AES) with a randomized self-reduction (so that we can have more confidence in its security). Our work here shows that any such primitive with a Levin randomized self-reduction must imply mathematically structured hardness. Since mathematically structured primitives tend to be less efficient than unstructured ones, our work here calls into question whether or not such a construction is possible.

CRHFs and Structure. We can extend our result on Levin-style randomized-self reductions to collision-resistant hash functions (CRHFs). Informally speaking, suppose we have a CRHF H such that any adversary $\mathcal{A}$ that breaks the collision resistance of H can also be used to break the one-wayness of H. We show that such an H also implies the existence of a one-way monoid. Our result implies that constructing CRHFs with powerful hardness reductions is likely going to require that the CRHF H is mathematically structured.

1.3 OTHER RELATED WORKS

A number of previous works have attempted to connect category theory and cryptography or security. The work of Pavlovic [61] shows how to categorically model an encryption scheme. Pavlovic uses different formalization methods than we do (while significantly more rigorous from a category-theoretic point of view, it is difficult to see how to generalize his formalism to different cryptographic primitives beyond encryption – our goal here is a general framework that encompasses most commonly used cryptographic primitives), and we think it is interesting future work to apply more category-theoretic rigor to our analysis here. Other interesting uses of category theory include modeling attackers [11], modeling composable security [21, 68], and even mapping finite state machines to zk-SNARK constructions [37].

1.4 PAPER OUTLINE

The rest of the paper is organized as follows. In Section 2, we present an overview of our results on diagramming cryptographic primitives. and the necessary structure for key exchange and some discussion on why we choose category theory. The technical core of our paper begins with Section 3, which presents preliminary background material. In Section 4, we formally show how to model cryptographic primitives as computational category-theoretic diagram. In Section 5, we explain our findings on the relationship between average-case reductions and mathematical structure. Finally, in Section 6 we offer directions for future work.

[3]A category theorist might disagree with this formulation on first sight, but we explain our rationale for this model later.
[4]We formally define this later in the paper.

2 TECHNICAL OVERVIEW

In this section, we provide an overview of how to formally model cryptosystems using computational category-theoretic diagrams. We start with the simple example of two-party non-interactive key exchange, which is illustrative of more complicated protocols.

Informally, key exchange is a protocol between two participants, who we will call Alice and Bob. Given some (randomly) generated public parameters, Alice and Bob each generate some "key share" message which we call s_A and s_B, respectively, and post this to some sort of public message board. They can then at any time "download" each others' public keys. From these shares (and using any "secret" randomness they may have sampled in the course of the protocol), Alice and Bob both can generate a final key k. Informally speaking, security requires that, given the initial public parameters, and the key shares s_A and s_B, no adversary can (efficiently) determine the final shared key k.

We formally outline below all of the components of a two-party non-interactive key exchange protocol:

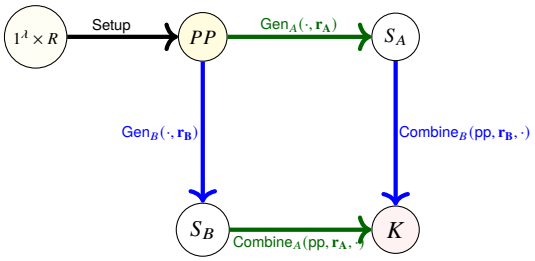
- $\mathsf{Setup} : 1^\lambda \times R \rightarrow PP$.[5]
- $\mathsf{Gen}_A : PP \times R_A \rightarrow S_A$.
- $\mathsf{Gen}_B : PP \times R_B \rightarrow S_B$.
- $\mathsf{Combine}_A : PP \times R_A \times S_B \rightarrow K$.
- $\mathsf{Combine}_B : PP \times R_B \times S_A \rightarrow K$.

In the above description, λ is a security parameter[6], and R, R_A and R_B denote certain efficiently sampleable "randomness spaces." In addition, we need to capture the correctness of the key exchange, ensuring that Alice and Bob can compute the same secret key using two different sequences of computation. This (informally) necessitates the following requirement:

$$\mathsf{Combine}_B\left(\mathsf{pp}, r_B, \mathsf{Gen}_A\left(\mathsf{pp}, r_A\right)\right) = \mathsf{Combine}_A\left(\mathsf{pp}, r_A, \mathsf{Gen}_B\left(\mathsf{pp}, r_B\right)\right),$$

$$\text{where } \mathsf{pp} \in PP, r_A \in R_A, r_B \in R_B.$$

This requirement will be immediately familiar to a category theoretician: it means that Gen_A, Gen_B, $\mathsf{Combine}_A$, and $\mathsf{Combine}_B$ form a *commutative square*. We can represent this using a category-theory style diagram:



We now explicitly formalize this requirement: let PP, R_A, R_B, S_A, S_B and K be *categories* that are sets without any non-identity arrows. Also, let Setup, Gen_A, Gen_B, $\mathsf{Combine}_A$, and $\mathsf{Combine}_B$ be *functors* (more specifically, bifunctors or trifunctors) mapping between the categories, with the property that the square diagram implied by the correctness of the key exchange protocol commutes. We note that some of these categories must be *efficiently sampleable* in order for our above key exchange formalization to actually work. We defer a more detailed discussion on handling such sampleability requirements to Section 4.

It turns out that we can model essentially any cryptographic primitive/protocol as such a collection of categories and functors.[7] In general, certain "objects" in our cryptosystems – messages, data, keys, randomness, etc. – are modeled as categories, and computations are modeled as functors. As we showed with key exchange, most of the transformations from cryptographic system to category-theoretic diagram are pretty straightforward.

Finally, we must address how to model security on these cryptographic/category theoretic diagrams. A natural first approach (mirroring [7]) might be to endow functors with hardness derived from Minicrypt primitives. For

[5]We assume that the public parameters implicitly incorporate the security parameter.

[6]We typically consider our diagrams of categories as an instance of an ensemble of categories indexed by the security parameter.

[7]We leave this statement intentionally vague, and refer the reader to section 4 for a comprehensive discussion on what can and cannot be modeled. Our argument that we can model "everything" is not formal–among other things, we would have to formally define a cryptosystem before we could attempt this–but we find it convincing.

instance, if Gen_A and Gen_B are modeled as one-way functions, and $\mathsf{Combine}_A$ and $\mathsf{Combine}_B$ are modeled as weak pseudorandom functions (with the key r_A and r_B, respectively), could we prove security of the key exchange protocol? Unfortunately (and analogously to why key exchange follows from the computational Diffie-Hellman assumption, but not discrete log), this does not work: we need to define security assumptions that take into account the fact that the functors may be correlated. So our assumption for key exchange follows naturally: given appropriately sampled key shares $s_A \in S_A$ and $s_B \in S_B$, determining the corresponding final k should be hard. We note that it is simple to deduce a hardness assumption for key exchange, but it can be more difficult for complicated primitives.

However, we can rigorously define cryptographic assumptions in a general way using our framework. We explain this in full detail in section 4. We further note that our paper is focused on the traditional classical model of computation. However, it is seemingly simple to extend to, say, quantum computation, where none of our observations on mathematical structure would need to change, but we would need to allow superposition queries in some cases.

Computational Category-Theoretic Modeling of More Primitives. In Section 4, we exemplify our approach by (informally) showing how to model identity-based encryption (IBE) [67, 18], multi-round two-party key exchange, and digital signatures. We present these examples to validate the fact that our framework is generic enough to model most of the commonly used cryptographic primitives, and makes it possible to compare such primitives in terms of their inherent mathematical structure.

Why Category Theory? A natural and immediate question to ask is why we chose to use category theory to model cryptographic primitives. We split this up into two questions, which we then answer:

1. Why is it useful and interesting to model cryptoprimitives and reductions as algebraically structured primitives?
2. Why are category-theoretic notations the right ones for these formalizations?

On (1). There are many good reasons to abstract out cryptoprimitives using mathematical frameworks that ignore "crypto semantics," and we outline many of them at various places in the paper. In particular, we think these mathematical diagrams make it much easier to see the relationship between cryptographic assumptions and cryptoprimitives: they can act as sort of a "narrow waist" between the two (like TCP/IP for the internet). Once it is shown that a computational category-theoretic diagram is sufficient for a cryptoprimitive, then all that is needed to instantiate that cryptoprimitive is show that an assumption is structured enough to securely realize the diagram. In this paper, we focus on a particular application of this approach, namely showing that certain cryptoprimitives can only be built from certain classes of mathematical assumptions (our proof that key exchange requires a certain kind of algebraic structure is an example of this). We hope that computational category-theoretic formulations also have other future applications, including making it easier to build cryptoprimitives from new assumptions and showing new separations (perhaps "unconditional" query-based proofs in the style of generic-group model impossibility results could be used).

Looking ahead, we also believe that modelling reductions (or, more accurately, primitives with certain types of reductions) using these mathematical frameworks provides new insights into their structural requirements, and may yield interesting observations for building OWFs or other Minicrypt primitives with "good" reductions (and it may be extended further–see our comments on Pessiland in Section 5).

On (2). An astute reader might point out that sets and functions can also be used to describe our results. In fact, they can be also used to explain basic category theoretic concepts; MacLane [56] in his textbook introduces categories from a set-theoretic framework. Perhaps our thought process for writing this paper will help explain why we chose category-theoretic concepts. The starting point for this work was [7], but [7] is not expressive enough to let us prove more general equivalence results. After trying to figure out how to best express cryptoprimitives in a mathematical way, we found basic category theoretic concepts to be a natural choice for describing the abstractions that we develop in this paper, while not departing too far from the more standard semantics with which cryptographers are familiar. There probably exists a spectrum of choices for potentially re-phrasing our results, ranging from more classical category-theoretic concepts to more commonly used cryptographic jargon, but we feel that our choice can be seen as a good tradeoff point between these two extremes.

For example, as we discussed in the introduction, [7] showed that endowing, among other things, a weak PRF with a (group) homomorphism between the input space and output space allows us to build "most" primitives that can be built from the DDH assumption. This is easy to model with our category-theoretic notation: a weak PRF is a single functor, and we can model a group-homomorphic weak PRF as a bifunctor from a group and a set (the set being the key space) to a group. Since the functor preserves structure, we don't have to go into any of the details about the homomorphism. This category-theoretic approach also lets us succinctly express the relationship

between public-key encryption and (group) homomorphic encryption: if we start with a category-theoretic diagram that represents PKE and simply make the message and ciphertext spaces groups, then we get group homomorphic encryption. This would be substantially more cumbersome to define without functors.

If we look ahead again, we use a substantial amount of category theory in section 5 to describe reductions. In particular, our observation that reductions are essentially just adjunctions endowed with computational hardness properties seems like a powerful simplification to us.

Why Structure-Preserving Functors? It might seem like a departure from standard conventions in category theory to use ostensibly structure-preserving functors for mapping between categories with no arrows (i.e., structure). The more conventional approach would be to represent all objects in a single category, and transformations between these objects as arrows. Unfortunately, this conventional approach does not suffice for modeling certain "structured" cryptographic primitives, e.g., primitives with homomorphisms. For instance, the discrete log assumption could be modeled as a "one-way functor" $D : K \times X \to Y$ for some categories K, X, and Y. But this representation is somewhat incomplete because it doesn't fully convey the mathematical structure inherent to the discrete log assumption over groups.

A more appropriate category-theoretic modeling of the discrete log assumption over some cyclic group $\mathbb{G}$ of prime order p is a functor of the form $D : \mathbb{G} \times \mathbb{Z}_p \to \mathbb{G}$. Since D is a functor, it preserves all of the relevant structure, and allows us to accurately model the mathematical structure inherent to the discrete log assumption. This exemplifies the fact that our computational category-theoretic framework is capable of modeling structured hardness assumptions using simple category-theoretic tools.

In fact, we can subsume essentially the entire framework of [7] and [6] by using functors. For instance, if F is a functor endowed appropriately with weak pseudorandomness properties, $\mathbb{G}$ is a group, and K is an arbitrary set, then $F : \mathbb{G} \times K \to \mathbb{G}$ can be used to precisely describe an input-homomorphic weak PRF, which is the main building block of [7]. Groups and homomorphic primitives are used extensively in cryptography, so we use separate categories for various objects in the cryptosystems – and functors between them – to formally capture such structured relationships.

3 PRELIMINARIES

We start by defining some basic concepts in category theory. We use slightly modified definitions from the classic MacLane text [56] and note that many of the things we define have multiple equivalent definitions. We note that MacLane defines categories as any interpretation of the category axioms within set theory and uses the notion of *metacategory* to handle classes and other collections that cannot be modelled as sets. Since we are only working with finite collections in this work, this is a restriction that does not impact us in any way. In other words, we will be exclusively working with *small* categories in this paper and thus will sometimes refer to the objects in a category as a set.

Typically in category theory, categories and functors are denoted with uppercase letters, and elements of a category and arrows are denoted with lowercase letters. We will use lowercase letters to sometimes denote functors when it is clear we are not using arrows, as this will help us avoid notation conflicts with some cryptographic primitives (e.g. a hash function "H").

Definition 1 (Directed Graph)**.** *A directed graph is a set O of objects, a set A of arrows, and two functions:*

$$dom : A \to O, \quad cod : A \to O$$

In a graph, we say that the set of composable pairs of arrows is the set

$$A \times A : \{\langle g, f\rangle \in A \text{ such that } dom\ g = cod\ f\}$$

and call this the "product over O."

We can now define a category. Informally, a category is just a directed graph where "identity arrows" exist and all of the arrows compose "nicely" when they are composable.

Definition 2 (Category)**.** *A* category *is a directed graph with two additional functions:*

Identity	Composition
$id : O \to A$	$\odot : A \times A \to A$
$c \to id_c$	$\langle g, f\rangle \to g \odot f$

where we may also write $g \odot f$ as gf such that

$$\begin{aligned} dom\,(id\,a) &= cod\,(id\,a) = a \\ dom\,(g \odot f) &= dom\,f \\ cod\,(g \odot f) &= cod\,g \end{aligned}$$

for all objects $a \in O$ and all composable arrows $\langle g, f\rangle \in A \times A$ such that the unit and associativity laws hold. In other words, for two pairs of composable arrows $\langle h, g\rangle$ and $\langle g, f\rangle$ we have

$$h \odot (g \odot f) = (h \odot g) \odot f$$

and for all arrows $f : a \to b$ and $g : b \to c$ we have

$$id_b \odot f = f \qquad\qquad g \odot id_b = g.$$

For cryptographers who aren't used to category theory, it can be useful to think of a category as a set with some structure indicated by the arrows. With categories in mind, we can define functors.

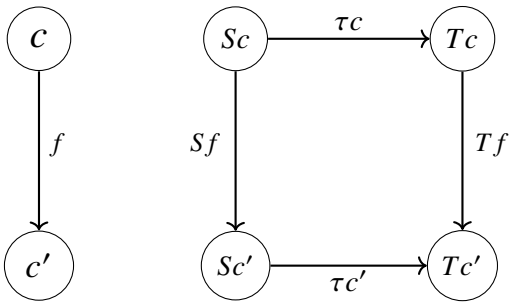
Definition 3 (Functor)**.** *A functor is a morphism of categories. For categories C and B, a functor $T : C \to B$ with domain C and codomain B consists of two related functions, both of which are denoted by T: the* object function*, which assigns to each object $c \in C$ an object $Tc \in B$, and the arrow function which assigns to each arrow $f : c \to c'$ an arrow $Tf : Tc \to Tc'$ of B such that the following holds:*

$$T\,(id_c) = id_{T_c} \text{ and } T\,(g \odot f) = Tg \odot Tf.$$

We emphasize that the latter equation only has to hold whenever the composite $g \odot f$ is defined in C.

In other words, a functor is a mapping between categories that preserves the internal structure of the categories. We can also consider structure preserving mappings of functor to functor. These are called natural transformations.

Definition 4 (Natural Transformation)**.** *Given two functors $S, T : C \to B$, a natural transformation $\tau : S \to T$ is a function which assigns to each object of $c \in C$ an arrow $\tau_c = \tau c : Sc \to Tc$ of B in such a way that every arrow $f : c \to c'$ in C yields a commutative diagram as follows:*



When this holds, we say that τ_c is natural *in c.*

In section 5, we note some similarities between complexity-theoretic reductions and adjunctions, so we define these next.

Definition 5 (Adjunction)**.** *Let A and X be categories. An adjunction from X to A is a triple $\langle F, G, \psi\rangle : X \to A$ where $F : X \to A$ and $G : A \to X$ are functors while ψ is a function which assigns to each pair of objects $(x, a) \in X \times A$ a bijection between the respective morphism sets; concretely*

$$\psi = \psi_{x,a} : A\,(Fx, a) \cong X\,(x, Ga)\,,$$

where $A\,(Fx, a)$ and $X\,(x, Ga)$ denote the respective morphism sets.

4 REPRESENTING CRYPTOSYSTEMS AS COMMUTATIVE DIAGRAMS WITH HARDNESS

In this section, we discuss how to represent cryptographic primitives as computational category theoretic "diagrams." In general, a cryptographic primitive/protocol is a collection of probabilistic polynomial-time algorithms with certain correctness and security requirements. At a high level, we try to capture the "core" correctness requirements of cryptographic primitives by modeling them as diagrams, where each such diagram consists of a collection of: (1) *categories* and (2) *functors* acting as maps between these categories. In addition, we attempt to capture the security requirements of these primitives by modeling these as natural hardness assumptions associated with the various functors.

One of the key observations is that these "diagrams" should be agnostic to the underlying purpose of the cryptosystem, and only dependent on the mathematical structure and security properties around that structure necessary for the scheme.

4.1 EXAMPLE-1: NON-INTERACTIVE KEY EXCHANGE

It is perhaps best to start with an example rather than immediately going to technical details. So, we exemplify our approach of using computational category theoretic tools to model cryptographic primitives using (two-party) non-interactive key exchange (NIKE). We begin by describing what a NIKE protocol is. Informally, it is a single round protocol executed between a pair of probabilistic polynomial-time algorithms (sometimes informally referred to as "parties") A and B that have access to some common public parameters. At a high level, the protocol proceeds as follows:

- Party A uses the public parameters (say, pp) to generate a secret state r_A and a message s_A, and transmits s_A to party B.
- Simultaneously, party B uses the public parameters pp to generate a secret state r_B and a message s_B, and transmits s_B to party A.
- Party A uses the message s_B received from party B together with its own secret state r_A to locally compute a secret key (say, k_{AB}).
- Similarly, party B uses the message s_A received from party A together with its own secret state r_B to locally compute a secret key (say, k_{BA}).

We note that this is an unusual definition of key exchange: Alice and Bob send their shares to each other rather than posting them to some sort of public "message board" and then "downloading" each other's shares. However, we do note that our definition of key exchange here is exactly equivalent to standard key exchange. The protocol is exactly equivalent if Alice and Bob post s_A and s_B, respectively, in steps one and two above, and then download each others' shares in steps three and four. We chose this definition because it more easily generalizes to multi-round key exchange, on which we spend a substantial amount of time later in the paper.

Correctness (Informal). The "core" correctness requirement of a NIKE protocol is that both parties A and B compute the *same* final secret key, i.e., we have $k_{AB} = k_{BA} = k$, even though each party used a difference sequence of local computations to arrive at this final key.

Security (Informal). At the same time, the security requirement of a NIKE protocol is that no (efficient) adversarial "third" party that observes the transcript of messages exchanged between party A and party B should be able to compute the final secret key k, except with negligible probability. An even stronger requirement that is often considered in the cryptographic literature is that the adversary should be able to *distinguish* the secret key k from a uniformly random bit-string of the same length, except with negligible probability.

Computational Category-Theoretic Representation of NIKE. We now formulate a potential approach for modeling the correctness requirement of a (two-party) NIKE protocol using a computational category-theoretic representation. The key observation here is that the correctness requirement for NIKE (namely, that two parties can compute the same secret key using two different sequences of computation) naturally yields a "commutative" diagram, represented as a collection of nodes and edges:

- At a high level, the nodes (equivalently, "categories") in this diagram are used to represent the sets from which the public parameters, the messages exchanged by the parties, and their own internal randomness/secret states are sampled.

- On the other hand, the edges (equivalently,"functors") in this diagram are used to represent the sequence of function computations that parties A and B follow to arrive at the final secret key.

We make the high-level description of our computational category-theoretic representation for two-party NIKE more explicit below.

Categories/Sets. As mentioned earlier, our category-theoretic representation for NIKE consists of "categories," which are essentially the sets from which the public parameters, the messages exchanged by the parties, and their own internal randomness/secret states are sampled. More specifically, let PP, R, S_A, S_B, R_A, R_B, and K denote *sets* (equivalently, *small categories*), where:

- We let PP denote the set of public parameters and R denote the set of possible random coins used by the setup algorithm to output some public parameters from the set PP.
- We also let S_A and S_B (resp., R_A and R_B) denote the set of possible output shares (resp., the set of possible secret states) for the parties A and B, respectively.
- Finally, we let K denote the set of possible final keys that the parties A and B could agree on at the end of the NIKE protocol.

Functors. Our category-theoretic representation for NIKE additionally consists of "functors". These are used to represent the sequence of function computations that parties A and B follow to arrive at the final secret key. At a high level, these functors act as "maps" between the sets/categories described above. Concretely, we consider the following functors:

- $\mathsf{Setup} : 1^\lambda \times R \to PP$ (λ being a security parameter).
- $\mathsf{Gen}_A : PP \times R_A \to S_A$.
- $\mathsf{Gen}_B : PP \times R_B \to S_B$.
- $\mathsf{Combine}_A : PP \times R_A \times S_B \to K$.
- $\mathsf{Combine}_B : PP \times R_B \times S_A \to K$.

From Correctness to a Commutative Diagram. In order to model the correctness requirement for NIKE, we need to impose the following correctness requirement on these functors: for any $\mathsf{pp} \in PP$, any $r_A \in R_A$ and any $r_B \in R_B$, we have

$$\mathsf{Combine}_A\left(\mathsf{pp}, r_A, \mathsf{Gen}_B\left(\mathsf{pp}, r_B\right)\right) = \mathsf{Combine}_B\left(\mathsf{pp}, r_B, \mathsf{Gen}_A\left(\mathsf{pp}, r_A\right)\right),$$

Observe that this correctness requirement naturally yields the following commutative diagram.

We present a subsequent discussion on how this approach of modeling cryptographic primitives as commutative diagrams naturally generalizes/extends to other well-studied cryptographic primitives.

Modeling Security Requirements. So far, our category-theoretic representation of NIKE captures only correctness requirements. Modeling security requires us to additionally endow the functors in the commutative diagram with certain additional "hardness" properties. For example, the *indistinguishability* security requirement of NIKE[8] naturally yields the following *indistinguishability*-style hardness requirement on the functors described in Figure 1 above:

For any security parameter $\lambda \in \mathbb{N}$, any $\mathsf{pp} \leftarrow PP$, any $r_A \leftarrow R_A$ and any $r_B \leftarrow R_B$, letting

$$s_A = \mathsf{Gen}_A\left(\mathsf{pp}, r_A\right), \quad s_B = \mathsf{Gen}_B\left(\mathsf{pp}, r_B\right), \quad k_{AB} = \mathsf{Combine}_A\left(\mathsf{pp}, r_A, s_B\right),$$

the following holds for any probabilistic polynomial time algorithm $\mathcal{A}$:

$$\mid \Pr[\mathcal{A}(\mathsf{pp}, s_A, s_B, k_{AB}) = 1] - \Pr[\mathcal{A}(\mathsf{pp}, s_A, s_B, k'_{AB}) = 1] \mid < \mathrm{negl}(\lambda),$$

where $k'_{AB} \leftarrow K$ is a key sampled from K by computing the key exchange diagram with "fresh" randomness.

[8]Obviously, key exchange requires only unpredictability, not indistinguishability, but we assume indistinguishability here for the purposes of explanation.

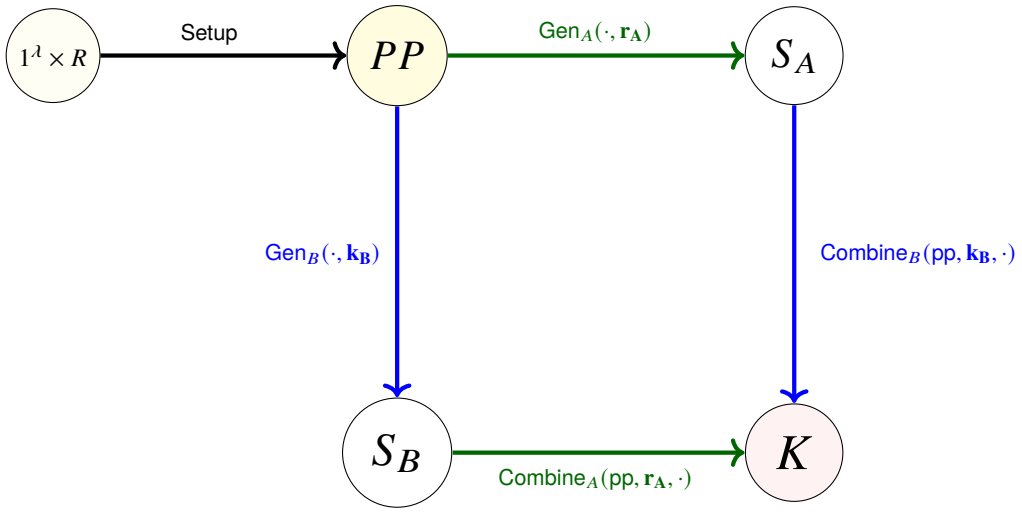


Figure 1: A computational category-theoretic diagram for NIKE.

Remark 1. *Note that this indistinguishability-security requirement that we impose on the collection of functors*

$$(\mathsf{Gen}_A, \mathsf{Gen}_B, \mathsf{Combine}_A, \mathsf{Combine}_B),$$

naturally endows certain hardness assumptions on the individual functors themselves. In particular, each of these functors should be one-way *on the (uniformly random) inputs r_A and r_B for the overall security requirement to hold.*

Remark 2. *In the aforementioned security definition, we assumed that there exist algorithms that can efficiently sample from the uniform distribution over the sets PP, R_A and R_B. While this is true for many NIKE protocols in practice, it need not be true for* any *NIKE protocol.*

4.2 MODELING CRYPTOGRAPHIC PRIMITIVES IN COMPUTATIONAL CATEGORY THEORY

We generalize the ideas exemplified in the above modeling of NIKE to define a system for modeling cryptographic primitives and protocols in terms of computational category-theoretic notions. Our approach will typically involve two stages: (1) modeling the correctness requirement of the cryptographic primitive as a diagram composed of category-theoretic entities (primarily categories and functors) with certain commutative properties, and (2) modeling the security requirement(s) of the primitive by endowing the relevant functors with appropriate hardness properties. Somewhat more concretely, our approach can be described as follows:

- We model cryptographic entities that are either stored locally by parties/algorithms or output/transmitted as messages to other parties as *categories*. We typically focus on *the category of sets* in the sense that almost all of the categories that we consider consist of categories that can be modeled as sets. Examples of entities that we model as categories include public parameters, public/private keys, messages, ciphertexts, etc. These categories serve as “nodes" in our diagram-based models of primitives.

- We model computations (either public or private) on these categories as *functors*. Each functor typically take as input objects belonging to one or more “input categories", and outputs an object belonging to an “output category". These functors serve as the “edges" connecting various categories or “nodes" in our diagram-based models of primitives.

Remark 3. *On the surface, our approach here would seem to violate common conventions of category theory. The conventional approach would be to represent all sets of cryptographic entities as objects in the same category, and to represent the computations between such objects as “arrows". Why are we using functors when they do not initially seem necessary? This is because the earlier approach does not suffice for modeling certain “structured" cryptographic primitives. In particular, certain primitives need maps between sets that preserve some form of mathematical structure (for instance, any homomorphic encryption scheme requires a homomorphism-preserving map between the plaintext set and the ciphertext set). It seems difficult to efficiently capture such structure-preserving maps using only arrows between objects in the same category. On the other hand, functors are structure-preserving by definition, which enables us to naturally capture such mathematically structured primitives. Hence, we opt for a computational category-theoretic approach that models primitives using functor-based representations.*

Remark 4. *When modeling cryptographic primitives, we need to make some assumptions on whether or not a given category supports efficient representations and/or efficiently sampleable distributions. Note we do not necessarily require all categories to have efficient representation and/or sampleability. We specify these properties for categories explicitly as and when we require them.*

Modeling Cryptographic Assumptions. Essentially any cryptographic primitive is associated with certain security properties. In order to model such security requirements, we need to be able to model cryptographic assumptions in a computational category-theoretic manner. At a high level, we model two kinds cryptographic assumptions - *decisional* and *computational*. Below, we provide an informal description of what the computational category-theoretic counterparts of these assumptions look like. We make these descriptions more formal subsequently.

- **Decisional Assumption:** Given a collection of categories $(C_1, \ldots, C_N)$ and a collection of functors of the form $(f_1, \ldots, f_Q)$, where each functor f_q is a map of the form:

  $$f_q : C_{i_1} \times C_{i_2} \times \ldots \times C_{i_\ell} \to C_j,$$

  for some $i_1, \ldots, i_\ell, j \in [N]$, a decisional assumption over the ensemble

  $$((C_1, \ldots, C_N), (f_1, \ldots, f_Q)),$$

  is typically a statement of the following form: for any security parameter $\lambda \in \mathbb{N}$ and for any:

  - vector of "public objects" $\mathbf{x}_{\text{public}}$ and a vector of "private objects" $\mathbf{x}_{\text{private}}$ (where $\mathbf{x}_{\text{public}}$ and $\mathbf{x}_{\text{private}}$ consist of objects sampled from certain well-defined distributions over one or more of the categories in the aforementioned ensemble), and

  - vector of "left challenge-oracle objects" $\mathbf{y}$ (where $\mathbf{y}$ consists of objects from a "left distribution" over one or more of the categories in the aforementioned ensemble, obtained by evaluating a subset of the functors in the aforementioned ensemble on some subset(s) of the objects in $\mathbf{x}_{\text{public}}$ and $\mathbf{x}_{\text{private}}$), and

  - vector of "right challenge-oracle objects" $\mathbf{z}$ (where $\mathbf{z}$ consists of objects from a "right distribution" over one or more of the categories in the aforementioned ensemble, obtained by evaluating a potentially different subset of the functors in the aforementioned ensemble on some subset(s) of the objects in $\mathbf{x}_{\text{public}}$ and $\mathbf{x}_{\text{private}}$),

  and for any probabilistic polynomial time algorithm $\mathcal{A}$, we have

  $$|\Pr[\mathcal{A}(\mathbf{x}_{\text{public}}, \mathbf{y}) = 1] - \Pr[\mathcal{A}(\mathbf{x}_{\text{public}}, \mathbf{z}) = 1]| < \text{negl}(\lambda).$$

- **Computational Assumption:** Given a collection of categories $(C_1, \ldots, C_N)$ and a collection of functors of the form $(f_1, \ldots, f_Q)$, where each functor f_q is a map of the form:

  $$f_q : C_{i_1} \times C_{i_2} \times \ldots \times C_{i_\ell} \to C_j,$$

  for some $i_1, \ldots, i_\ell, j \in [N]$, a decisional assumption over the ensemble

  $$(C_1, \ldots, C_N), (f_1, \ldots, f_Q),$$

  is typically a statement of the following form: for any security parameter $\lambda \in \mathbb{N}$ and for any:

  - vector of "public objects" $\mathbf{x}_{\text{public}}$ and a vector of "private objects" $\mathbf{x}_{\text{private}}$ (where $\mathbf{x}_{\text{public}}$ and $\mathbf{x}_{\text{private}}$ consist of objects sampled from certain well-defined distributions over one or more of the categories in the aforementioned ensemble), and

  - vector of "target objects" $\mathbf{y}$ (where $\mathbf{y}$ consists of objects from a "target distribution" over one or more of the categories in the aforementioned ensemble, obtained by evaluating a subset of the functors in the aforementioned ensemble on some subset(s) of the objects in $\mathbf{x}_{\text{public}}$ and $\mathbf{x}_{\text{private}}$), and

  and for any probabilistic polynomial time algorithm $\mathcal{A}$, we have

  $$\Pr[\mathcal{A}(\mathbf{x}_{\text{public}}) = \mathbf{y}] < \text{negl}(\lambda).$$

Modeling Cryptographic Assumptions with Oracles. We can also support decisional/computational cryptographic assumptions where the probabilistic-polynomial time adversary has oracle-access to certain functor evaluation circuits that are "hardwired" with the private inputs. We note here that the security of many well-studied cryptographic primitives rely on cryptographic assumptions that must hold even when the adversary has access to such oracles.

Consider, for example, the well-studied notion of indistinguishability-based security for public-key encryption schemes against chosen-ciphertext attacks (often abbreviated as CCA security) [25], which requires encryptions of any arbitrary pair of messages to be computationally indistinguishable, even in the presence of a *decryption oracle* that can be used on all messages except the challenge pair. This oracle essentially allows the adversary to evaluate a decryption circuit hardwired with the secret decryption key on ciphertexts of its choice (modulo certain restrictions). From a computational category-theoretic point of view, a decryption key is an example of a "private object" sampled from the category/set of all possible keys. Hence, in order to model such cryptographic primitives, our computational category-theoretic framework should be able to model cryptographic assumptions where the adversary might have oracle-access to certain functor evaluation circuits hardwired with the private inputs. Similar requirements arise in the context of security definitions for advanced cryptographic primitives such as collusion-resistant identity-based encryption (IBE) [67, 18] and functional encryption [19].

Below, we provide an informal description of what the computational category-theoretic counterparts of these oracle-based assumptions look like.

- **Decisional Assumption with Oracle(s):** Given a collection of categories $(C_1, \ldots, C_N)$ and a collection of functors of the form $(f_1, \ldots, f_Q)$, where each functor f_q is a map of the form:

  $$f_q : C_{i_1} \times C_{i_2} \times \ldots \times C_{i_\ell} \rightarrow C_j,$$

  for some $i_1, \ldots, i_\ell, j \in [N]$, a decisional assumption over the ensemble

  $$(C_1, \ldots, C_N), (f_1, \ldots, f_Q),$$

  is typically a statement of the following form: for any security parameter $\lambda \in \mathbb{N}$ and for any:

  - vector of "public objects" $\mathbf{x}_{\mathsf{public}}$ and a vector of "private objects" $\mathbf{x}_{\mathsf{private}}$ (where $\mathbf{x}_{\mathsf{public}}$ and $\mathbf{x}_{\mathsf{private}}$ consist of objects sampled from certain well-defined distributions over one or more of the categories in the aforementioned ensemble), and
  - vector of "left challenge-oracle objects" $\mathbf{y}$ (where $\mathbf{y}$ consists of objects from a "left distribution" over one or more of the categories in the aforementioned ensemble, obtained by evaluating a subset of the functors in the aforementioned ensemble on some subset(s) of the objects in $\mathbf{x}_{\mathsf{public}}$ and $\mathbf{x}_{\mathsf{private}}$), and
  - vector of "right challenge-oracle objects" $\mathbf{z}$ (where $\mathbf{z}$ consists of objects from a "right distribution" over one or more of the categories in the aforementioned ensemble, obtained by evaluating a potentially different subset of the functors in the aforementioned ensemble on some subset(s) of the objects in $\mathbf{x}_{\mathsf{public}}$ and $\mathbf{x}_{\mathsf{private}}$),

  and for any probabilistic polynomial time algorithm $\mathcal{A}$, we have

  $$|\Pr[\mathcal{A}^{O(\mathbf{x}_{\mathsf{private}}, \cdot)}(\mathbf{x}_{\mathsf{public}}, \mathbf{y}) = 1] - \Pr[\mathcal{A}^{O(\mathbf{x}_{\mathsf{private}}, \cdot)}(\mathbf{x}_{\mathsf{public}}, \mathbf{z}) = 1]| < \mathrm{negl}(\lambda),$$

  where $O(\mathbf{x}_{\mathsf{private}}, \cdot)$ (informally) represents a "global" oracle that allows the adversary to see certain "allowed" evaluations of a subset of the functors in the aforementioned ensemble on some subset(s) of the objects in $\mathbf{x}_{\mathsf{private}}$ and any other objects of the adversary's choice.

- **Computational Assumption with Oracle(s):** Given a collection of categories $(C_1, \ldots, C_N)$ and a collection of functors of the form $(f_1, \ldots, f_Q)$, where each functor f_q is a map of the form:

  $$f_q : C_{i_1} \times C_{i_2} \times \ldots \times C_{i_\ell} \rightarrow C_j,$$

  for some $i_1, \ldots, i_\ell, j \in [N]$, a decisional assumption over the ensemble

  $$(C_1, \ldots, C_N), (f_1, \ldots, f_Q),$$

  is typically a statement of the following form: for any security parameter $\lambda \in \mathbb{N}$ and for any:

  - vector of "public objects" $\mathbf{x}_{\mathsf{public}}$ and a vector of "private objects" $\mathbf{x}_{\mathsf{private}}$ (where $\mathbf{x}_{\mathsf{public}}$ and $\mathbf{x}_{\mathsf{private}}$ consist of objects sampled from certain well-defined distributions over one or more of the categories in the aforementioned ensemble), and

– vector of "target objects" $\mathbf{y}$ (where $\mathbf{y}$ consists of objects from a "target distribution" over one or more of the categories in the aforementioned ensemble, obtained by evaluating a subset of the functors in the aforementioned ensemble on some subset(s) of the objects in $\mathbf{x}_{\mathsf{public}}$ and $\mathbf{x}_{\mathsf{private}}$), and

and for any probabilistic polynomial time algorithm $\mathcal{A}$, we have

$$\Pr[\mathcal{A}^{O(\mathbf{x}_{\mathsf{private}},\cdot)}(\mathbf{x}_{\mathsf{public}}) = \mathbf{y}] < \mathrm{negl}(\lambda),$$

where $O(\mathbf{x}_{\mathsf{private}},\cdot)$ again (informally) represents a "global" oracle that allows the adversary to see certain "allowed" evaluations of a subset of the functors in the aforementioned ensemble on some subset(s) of the objects in $\mathbf{x}_{\mathsf{private}}$ and any other objects of the adversary's choice.

To summarize, the computational category-theoretic model of a cryptographic primitive consists of a collection of categories, a collection functors acting as (structure-preserving) maps between these categories, and one or more assumptions modeled as hardness properties endowed on the functors, typically described using either an "indistinguishability" experiment or as an "unpredictability" experiment against probabilistically polynomial-time adversaries. Our definitions and abstractions are designed to be generic enough to model essentially any well-known cryptographic primitive/protocol.

In addition to (two-party)NIKE, we exemplify the generality and usefulness of our framework by showing how to model two more well-studied cryptographic primitives, namely identity-based encryption (IBE) [67, 18] and digital signatures [64, 59]. We think these examples offer good evidence that our models are generic enough to model pretty much any cryptosystem. In an ideal world, we could formally prove that our definitions and abstractions could be used to model *any* cryptosystem, but, as we have mentioned before, this would require a rigorous definition of a cryptosystem. It will be similarly difficult to argue that any definition of a cryptosystem is a correct one (because it encompasses all uses of cryptography in practice), so we do not attempt this and hope that our examples are convincing.

4.3 MORE EXAMPLES OF MODELING CRYPTOGRAPHIC PRIMITIVES

In this subsection, we present some more examples of cryptographic primitives that can be modeled using our computational category-theoretic framework.

4.3.1 IDENTITY-BASED ENCRYPTION (IBE)

In this subsection, we show how to model identity-based encryption (IBE) [67, 18] in our computational category-theoretic framework. For readers not familiar with IBE, we first recall its formal definition.

Definition 6. (**Identity-Based Encryption (IBE)**). *An* IBE *scheme over an identity space $\mathcal{ID}$ and message space $\mathcal{M}$ is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$ *defined as follows:*

- $\mathsf{Setup}(1^\lambda)$*: Given the security parameter λ, outputs the public parameter* pp *and the master secret-key* msk.
- $\mathsf{Ext}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$*: Given the public parameter* pp*, the master-secret-key* msk *and an identity* $\mathsf{id} \in \mathcal{ID}$*, outputs a secret-key* $\mathsf{sk}_{\mathsf{id}}$.
- $\mathsf{Enc}(\mathsf{pp}, \mathsf{id} \in \mathcal{ID}, \mathsf{m})$*: Given the public parameter* pp*, an identity* $\mathsf{id} \in \mathcal{ID}$ *and a message* $\mathsf{m} \in \mathcal{M}$*, outputs a ciphertext* ct.
- $\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct})$*: Given a secret key* $\mathsf{sk}_{\mathsf{id}}$ *and a ciphertext* ct*, outputs a decrypted message* m'.

The following correctness and security properties must be satisfied:

- **Correctness:** If $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, then for all $\mathsf{id} \in \mathcal{ID}$ and all $\mathsf{m} \in \mathcal{M}$, it holds with overwhelming probability over the randomness of Ext and Enc that if $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{Ext}(\mathsf{pp}, \mathsf{msk}, \mathsf{id})$ and $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{id} \in \mathcal{ID}, \mathsf{m})$, then we have:
$$\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) = \mathsf{m}.$$
- **Anonymous-IND-CPA Security:** For $b \in \{0, 1\}$, define the experiment $\mathsf{Expt}_b^{\text{ano-cpa}}$ between a challenger and an adversary $\mathcal{A}$ as in Figure 2. An IBE scheme $(\mathsf{Setup}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$ is said to be anonymous-IND-CPA-secure if for all PPT adversaries $\mathcal{A}$, the views of the adversary in $\mathsf{Expt}_0^{\text{ano-cpa}}$ and $\mathsf{Expt}_1^{\text{ano-cpa}}$ are computationally indistinguishable.

We now show how to model anonymous-IND-CPA-secure IBE in our computational category-theoretic framework.

Experiment $\mathsf{Expt}_b^{\text{ano-cpa}}$:

1. The challenger samples $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and provides pp to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ adaptively issues key-generation queries. For each query identity id, the challenger responds with
$$\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{Ext}(\mathsf{msk}, \mathsf{id}).$$
3. The adversary $\mathcal{A}$ outputs identity-message pairs $\left(\mathsf{id}_0^*, \mathsf{m}_0^*\right)$ and $\left(\mathsf{id}_1^*, \mathsf{m}_1^*\right)$, such that $\mathsf{id}_{b^*}^* \neq \mathsf{id}$ for each identity id queried previously and each $b^* \in \{0, 1\}$. The challenger responds to the adversary $\mathcal{A}$ with the ciphertext
$$\mathsf{ct} \leftarrow \mathsf{Enc}\left(\mathsf{pp}, \mathsf{id}_b^*, \mathsf{m}_b^*\right).$$
4. The adversary $\mathcal{A}$ continues to adaptively issue key-generation queries, subject to the aforementioned restrictions. The challenger responds as above.

Figure 2: Experiment for the Anonymous CPA security of IBE

Categories/Sets. As in the modeling of two-party NIKE, our computational category-theoretic modeling of IBE uses a collection of sets/categories. More concretely, let PP, MSK, R_{Setup}, ID, SK, R_{Ext}, M, CT, and R_{Enc} denote sets/categories, where:

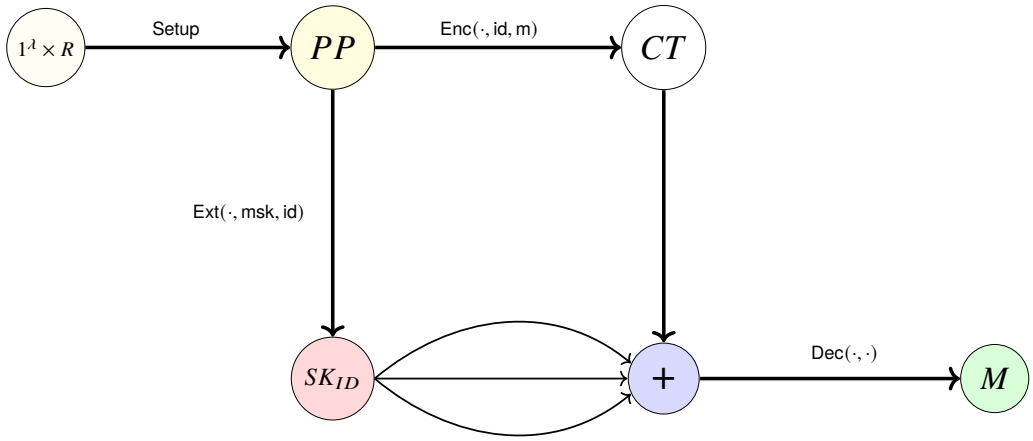
- We let PP and MSK denote the set of public parameters and master private keys, respectively. We let R_{Setup} denote the set of possible random coins used by the setup algorithm to output some public parameter and some master secret key.
- We let ID denote the space of all identities and SK denote the space of all possible secret keys corresponding to identities in ID. We also let R_{Ext} denote the set of possible random coins used by the extraction algorithm to generate the master secret key corresponding to some identity $\mathsf{id} \in ID$.
- Finally, we let M and CT denote the set of possible messages and ciphertexts, respectively. We also let R_{Enc} denote the set of possible random coins used by the encryption algorithm to generate a ciphertext corresponding to some message in M and some identity in ID.

Functors. Our category-theoretic representation for IBE additionally consists of functors. As in our modeling of NIKE, these functors are used to represent "maps" between the sets/categories described above. Concretely, we consider the following functors:

- $\mathsf{Setup} : 1^\lambda \times R_{\mathsf{Setup}} \rightarrow PP \times MSK$ (λ being a security parameter).
- $\mathsf{Ext} : PP \times MSK \times ID \times R_{\mathsf{Ext}} \rightarrow SK$.
- $\mathsf{Enc} : PP \times ID \times M \times R_{\mathsf{Enc}} \rightarrow CT$.
- $\mathsf{Dec} : SK \times CT \rightarrow M$.

Correctness. In order to model the correctness requirement for IBE, we need to impose the following correctness requirement on these functors: for any $r_{\mathsf{Setup}} \in R_{\mathsf{Setup}}$, any $\mathsf{id} \in ID$, any $r_{\mathsf{Ext}} \in R_{\mathsf{Ext}}$, any $\mathsf{m} \in M$ and any $r_{\mathsf{Enc}} \in R_{\mathsf{Enc}}$, letting
$$(\mathsf{pp}, \mathsf{msk}) = \mathsf{Setup}(1^\lambda, r_{\mathsf{Setup}}), \quad \mathsf{sk}_{\mathsf{id}} = \mathsf{Ext}(\mathsf{pp}, \mathsf{msk}, \mathsf{id}, r_{\mathsf{Ext}}), \quad \mathsf{ct} = \mathsf{Enc}(\mathsf{pp}, \mathsf{id}, \mathsf{m}, r_{\mathsf{Enc}}),$$
where $\mathsf{pp} \in PP$, $\mathsf{msk} \in MSK$, $\mathsf{sk}_{\mathsf{id}} \in SK$, and $\mathsf{ct} \in CT$, we have
$$\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) = \mathsf{m}.$$

Observe that this correctness requirement naturally yields the following diagram. Note that the multiple arrows from SK_{ID} represent the possibility of decryption with each of the keys. We use the "+" symbol in the diagram to indicate that the decryption functor takes as input a tuple consisting of a secret key and a ciphertext, and outputs a message.



Modeling Security Requirements. We now model the anonymous-IND-CPA security requirement of IBE using the following decisional assumption with oracles:

For any security parameter $\lambda \in \mathbb{N}$, any $r_{\mathsf{Setup}} \leftarrow R_{\mathsf{Setup}}$, any $\mathsf{id}_0, \mathsf{id}_1 \in ID$, any $\mathsf{m}_0, \mathsf{m}_1 \in M$, and uniformly sampled $r_{\mathsf{Enc},0}, r_{\mathsf{Enc},1} \leftarrow R_{\mathsf{Enc}}$, letting

$$(\mathsf{pp}, \mathsf{msk}) = \mathsf{Setup}(1^\lambda, r_{\mathsf{Setup}}),$$

and for any probabilistic polynomial time algorithm $\mathcal{A}$, letting

$$\epsilon_b = \Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{Ext}}(\mathsf{msk},\cdot)}(\mathsf{pp}, \mathsf{Enc}(\mathsf{pp}, \mathsf{id}_b, \mathsf{m}_b, r_{\mathsf{Enc},b})) = 1],$$

for $b \in \{0, 1\}$, the following holds:

$$\mid \epsilon_0 - \epsilon_1 \mid < \mathrm{negl}(\lambda),$$

subject to the restriction that for any identity id such that the adversary $\mathcal{A}$ makes an oracle query of the form $\mathcal{O}_{\mathsf{Ext}}(\mathsf{msk}, \mathsf{id})$, we have $\mathsf{id} \neq \mathsf{id}_0^*$ and $\mathsf{id} \neq \mathsf{id}_1^*$.

Remark 5. *Note that this is an example of a decisional assumption with oracles that we model using our category-theoretic framework. In particular, one can view this assumption as a special instance of the generic description regarding how to model any decisional assumption with oracles that we presented earlier.*

Remark 6. *In the aforementioned security definition, we assumed that there exist algorithms that can efficiently sample from the uniform distribution over the sets R_{Setup}, R_{Ext} (inside the oracle $\mathcal{O}_{\mathsf{Ext}}$) and R_{Enc}. While this is true for many IBE schemes in practice, it need not be true for* any *IBE scheme and this assumption was made purely for simplicity of exposition. We can equivalently present a more rigorously formal security definition for IBE in the computational category-theoretic setting where these distributions could be non-uniform.*

4.3.2 DIGITAL SIGNATURES

In this subsection, we show how to model digital signatures [64, 59] in our computational category-theoretic framework. We first recall the formal definition for digital signatures.

Definition 7. (**Digital Signatures**). *A digital signature scheme over a message space $\mathcal{M}$ is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Ver})$ *defined as follows:*

- $\mathsf{Setup}(1^\lambda)$*: Given the security parameter λ, outputs the (secret) signing key* sk *and the (public) verification key* vk.
- $\mathsf{Sign}\,(\mathsf{sk}, \mathsf{m})$*: Given the signing key* sk *and a message* $\mathsf{m} \in \mathcal{M}$*, outputs a signature σ.*
- $\mathsf{Ver}\,(\mathsf{vk}, \mathsf{m}, \sigma)$*: Given the verification key* vk*, a message* $\mathsf{m} \in \mathcal{M}$*, and a signature σ, output either* 1 *(indicating successful verification) or* 0 *(indicating failed verification).*

The following correctness and security properties must be satisfied:

- **Correctness:** If $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}\,(1^\lambda)$, then for all $\mathsf{m} \in \mathcal{M}$, it holds with overwhelming probability that:

$$\mathsf{Ver}\,(\mathsf{vk}, \mathsf{m}, \mathsf{Sign}(\mathsf{sk}, \mathsf{m})) = 1.$$

Experiment $\mathsf{Expt}^{\text{eu-cma}}$:

1. The challenger samples $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and provides vk to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ adaptively issues signing queries. For each query message m, the challenger responds with
$$\sigma \leftarrow \mathsf{Sign}\,(\mathsf{sk}, \mathsf{m})\,.$$
3. Eventually, the adversary $\mathcal{A}$ outputs a message-signature pair $(\mathsf{m}^*, \mathrm{SIG}^*)$ such that no signing query was issued by $\mathcal{A}$ on m previously.

We say that $\mathcal{A}$ wins if $\mathsf{Ver}(\mathsf{vk}, \mathsf{m}^*, \mathrm{SIG}^*) = 1$.

Figure 3: Existential Unforgeability against Chosen-Message Attacks

- **Existential Unforgeability against Chosen-Message Attacks:** Define the experiment $\mathsf{Expt}_b^{\text{eu-cma}}$ between a challenger and an adversary $\mathcal{A}$ as in Figure 3. A signature scheme $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Ver})$ is said to be existentially unforgeable against chosen-message attacks if for all PPT adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the experiment $\mathsf{Expt}^{\text{eu-cma}}$ is negligible.

We now show how to model any digital signature scheme that is existentially unforgeable against chosen-message attacks in our computational category-theoretic framework.

Categories/Sets. As in the modeling of two-party NIKE and IBE, our computational category-theoretic modeling of digital signatures uses a collection of sets/categories. More concretely, let SK, VK, R_{Setup}, M, Σ and R_{Sign} denote sets/categories, where:

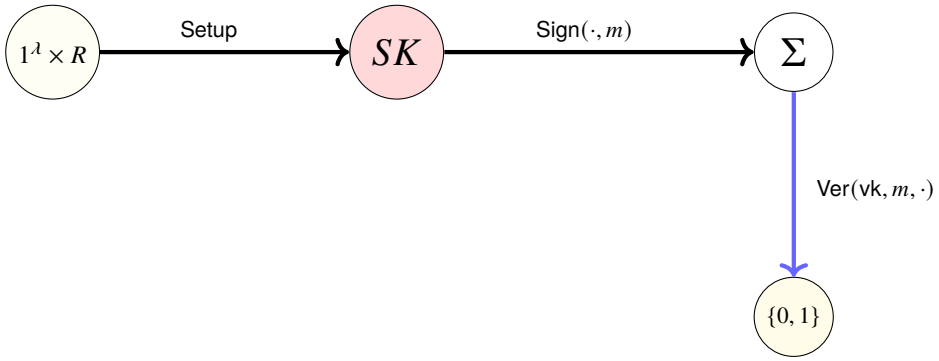
- We let SK and VK denote the set of signing keys and verification keys, respectively. We let R_{Setup} denote the set of possible random coins used by the setup algorithm to output a signing and verification key pair.
- We let M and Σ denote the space of all messages and signatures, respectively. We also let R_{Sign} denote the set of possible random coins used by the signing algorithm to generate a signature in Σ corresponding to some message in M.

Functors. We consider the following functors:

- $\mathsf{Setup} : 1^\lambda \times R_{\mathsf{Setup}} \rightarrow SK \times VK$ (λ being a security parameter).
- $\mathsf{Sign} : SK \times M \times R_{\mathsf{Sign}} \rightarrow \Sigma$.
- $\mathsf{Dec} : VK \times M \times \Sigma \rightarrow \{0, 1\}$.

Correctness. We model the correctness requirement for digital signatures as follows: for any $r_{\mathsf{Setup}} \in R_{\mathsf{Setup}}$, any $\mathsf{m} \in M$, and any $r_{\mathsf{Sign}} \in R_{\mathsf{Sign}}$, letting
$$(\mathsf{sk}, \mathsf{vk}) = \mathsf{Setup}(1^\lambda, r_{\mathsf{Setup}}),$$
where $\mathsf{sk} \in SK$, and $\mathsf{vk} \in VK$, we have
$$\mathsf{Ver}(\mathsf{vk}, \mathsf{m}, \mathsf{Sign}(\mathsf{sk}, \mathsf{m}, r_{\mathsf{Setup}})) = 1.$$

Observe that this correctness requirement naturally yields the following diagram.



Modeling Security Requirements. We now model the notion of existential unforgeability against chosen-message attacks using the following computational assumption with oracles:

For any security parameter $\lambda \in \mathbb{N}$, and any $r_{\mathsf{Setup}} \leftarrow R_{\mathsf{Setup}}$, letting

$$(\mathsf{pk}, \mathsf{vk}) = \mathsf{Setup}(1^\lambda, r_{\mathsf{Setup}}),$$

and for any probabilistic polynomial time algorithm $\mathcal{A}$, letting

$$\epsilon_b = \Pr[\mathcal{A}^{\mathsf{Ext}(\mathsf{msk},\cdot,r_{\mathsf{Ext}})}(\mathsf{pp}, \mathsf{Enc}(\mathsf{pp}, \mathsf{id}_b, \mathsf{m}_b, r_{\mathsf{Enc},b})) = 1],$$

for $b \in \{0, 1\}$, the following holds:

$$\Pr[\mathcal{A}^{O_{\mathsf{Sign}}(\mathsf{sk},\cdot)}(\mathsf{vk}) = (\mathsf{m}^*, \mathrm{SIG}^*) \wedge \mathsf{Ver}(\mathsf{vk}, \mathsf{m}^*, \mathrm{SIG}^*) = 1] < \mathrm{negl}(\lambda),$$

subject to the restriction that for any message m such that the adversary $\mathcal{A}$ makes an oracle query of the form $O_{\mathsf{Sign}}(\mathsf{sk}, \mathsf{m})$, we have $\mathsf{m} \neq \mathsf{m}^*$.

Remark 7. *Note that this is an example of a computational assumption with oracles that we model using our category-theoretic framework. In particular, one can view this assumption as a special instance of the generic description regarding how to model any computational assumption with oracles that we presented earlier.*

Remark 8. *In the aforementioned security definition, we again assumed that there exist algorithms that can efficiently sample from the uniform distribution over the sets R_{Setup} and R_{Sign} (inside the oracle O_{Sign}). Once again, this assumption was made purely for simplicity of exposition; we can equivalently present a more rigorously formal security definition for digital signatures in the computational category-theoretic setting where these distributions could be non-uniform.*

4.4 SUMMARY AND DISCUSSIONS

The aforementioned discussions on IBE and digital signatures serve to exemplify our two-step approach for modeling cryptographic primitives (this was described earlier but we re-iterate it for ease of understanding): (1) modeling the correctness requirement of the cryptographic primitive as a diagram composed of category-theoretic entities (primarily categories and functors) with certain commutative properties, and (2) modeling the security requirement(s) of the primitive by endowing the relevant functors with appropriate hardness properties. In general, the computational category-theoretic model of a cryptographic primitive will consist of a collection of categories, a collection functors acting as maps between these categories, and one or more cryptographic assumptions modeled as hardness properties endowed on the functors, typically described using either an "indistinguishability" experiment or as an "unpredictability" experiment against probabilistically polynomial-time adversaries.

We note here that our approach for modeling of cryptographic primitives and cryptographic assumptions is rather generic. For example, our notion of a decisional/computational *category-theoretic security assumption* allows proving very generic statements of the form: *it is hard for a probabilistic polynomial-time adversary to distinguish between collections of objects that were sampled from two different distributions over a collection of categories*, or it is hard for a probabilistic polynomial-time adversary to predict the output of a certain sequence of functor computations over objects sampled from a certain distributions over a collection of categories. It is possible that one could use our framework to model cryptographic primitives/protocols that are not particularly useful for real-world applications.

However, we do not view this generality of our framework as a drawback; we intentionally opt for a general framework that allows us to encompass as many cryptographic primitives as possible. It also turns out that this general approach provides very meaningful insights into the mathematical structure that might be inherent to a cryptographic primitive in general (as opposed to concrete instantiations of this primitive from concrete hardness assumptions). This is beneficial in terms of inferring/arguing lower bounds on the "amount of structure" necessary to realize certain primitives. In the next section, we present a more in-depth analysis of this observation in the context of two-party NIKE.

5 REDUCTIONS AND MATHEMATICAL STRUCTURE

In this section, we use a computational category-theoretic approach to show that certain types of complexity-theoretic reductions (namely, Levin reductions) imply mathematical structure.[9] More concretely, we show that

[9]We focus exclusively on Levin reductions in this section; extending these arguments to Turing reductions or other, more general, types of reduction is interesting future work. We formally define our exact notion of reduction later in this section.

the existence of a language (or a set of language instances) with a *randomized self-reduction* implies a *one-way monoid*. We then leverage this result to show that a similar mathematical structure is implied by certain families of collision-resistant hash functions. In particular, we show that any (sufficiently compressing) function with collision resistance that is (Levin) reducible from one-wayness also implies a one-way monoid. This latter result has interesting implications on the efficiency of provably secure constructions of CRHFs.

5.1 MODELING LANGUAGES AS CATEGORIES

We start by defining a category-theoretic style of representation around NP languages and associated problems. In particular, we model an NP language using what we call a *bipartite* category (i.e., the category can be split into two sets such that all arrows go between the two sets–the natural analogy is a bipartite graph) with two subcategories which we will call X and W. X will contain objects that we refer to as "language instances" and W will contain objects that we refer to as "witnesses." For any $x \in X$ and $w \in W$, we will have an arrow from $x \rightarrow w$ and $w \rightarrow x$ if and only if w is a witness for a particular language instance x. We call the arrows from X to W "solving arrows" and the arrows from W to X "verifying arrows." We formalize this category-theoretic notion with the following definition:

Definition 8 (Language Category)**.** *Let C be a bipartite category with two subcategories X and W, and let L be a language. We say that C is a* language category *if there are* no *arrows between elements of X, no elements between elements of W, and a pair of arrows going back and forth between two elements $x \in X$ and $w \in W$ if and only if w represents a valid witness for the language instance represented by x.*

We note that, as a reader may suspect, to our knowledge this is an entirely new definition and reflects the use of the category rather than any particular underlying structure.

Remark 9. *Note that if it is possible to efficiently tell whether or not a verifying arrow exists from some $w \in W$ to an $x \in X$, then we recover the class NP with this description. We will focus almost entirely on languages in NP in this section, so we assume that such arrows can be efficiently determined unless we explicitly state otherwise.*

5.2 CATEGORY-THEORETIC MODELING OF REDUCTIONS

We model (Levin) reductions as functors between language categories (plus some extra supplemental information). As a warm-up, we will begin by discussing single-instance reductions. We eventually move to reductions between sets of language instances (families of problems), and finally randomized reductions.

Single Instance Reductions. Informally speaking, a reduction between two language instances allows generating a witness for the first language instance given a witness for the second language instance. More precisely, a reduction between a language instance x_1 and a language instance x_2 is an efficiently computable mapping from the set of witnesses of x_2 to the set of witnesses of x_1. Note that this mapping does not need to be one-to-one. However, the reduction should allow us to convert any possible witness for x_2 into a witness for x_1.

From a category theory perspective, suppose we have two set elements x_1 and $x_2 \in X$ that represent language instances, with arrows to and from all of the set elements representing their witnesses $w \in W$. Let C_1 denote the category containing the element x_1, the witness set W, and the appropriate arrows, and C_2 be the same for x_2. We model a reduction as an efficiently computable functor g from C_2 to C_1. Note that since the functor g preserves structure, it maps arrows (corresponding to witness/language instance pairs) in C_2 to arrows in C_1. Since we only have one language instance in each category ($x_1 \in C_1$ and $x_2 \in C_2$), the functor g must map witnesses in C_2 to valid witnesses in C_1, meaning that if we can find a witness for x_2, then the functor g can be used to find one for C_1.

Reductions Between Families of Problems. Single language instance reductions unfortunately are not very useful or interesting. They are really more of an equivalence relation between problems than what a complexity theorist or certainly a cryptographer would consider to be a reduction. However, maps between two sets of language instances (or families of problems, if we use the language of cryptography) are critical for cryptography and complexity theory. It turns out that we can think of these reductions between set/families as collections of reductions between individual problems and continue to view reductions as (structure-preserving) functors–we just need to keep track of the "source" instance in the reduction. However, this necessitates the use of another functor for the "forward" direction of the reduction that composes "naturally" with the functor g.

Once again, consider two language categories $C_1 = \{X_1, W_1\}$ and $C_2 = \{X_2, W_2\}$. Note that our "forward" direction functor f should *not* be able to efficiently map the entire category $C_1 \rightarrow C_2$, as this would enable mapping solutions in the "forward" direction. So we restrict f to be a functor between the (sub)categories X_1 and X_2. In the "reverse" mapping, g still maps from C_2 to C_1, which induces a (sub)functor $\tilde{g}$ from X_2 to X_1.

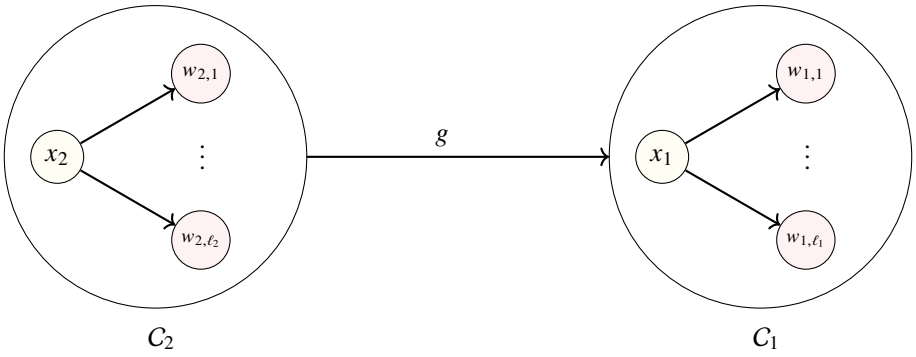


Figure 4: A Computational Category-Theoretic View of a Single Instance Reduction

When $|X_1| = |X_2|$ and the reduction is one-to-one (i.e. the reduction maps every instance $x_1 \in X_1$ to a unique $x_2 \in X_2$), then we can model the reduction using a set of two efficiently computable functors $f : X_1 \to X_2$ and $g : C_2 \to C_1$ (and the corresponding (sub)functor $\tilde{g} : X_2 \to X_1$), with the property that

$$f \circ \tilde{g}\,(x \in X) = x.$$

In category-theoretic terms, this implies the existence of an *adjunction* between the categories C_1 and C_2 defined by the functor f and the (sub)functor $\tilde{g}$.

Informally, an adjunction is a pair of functors mapping between two categories in opposite directions in a "natural" way. The adjunction is defined as a triple $(f, \tilde{g}, \psi)$. f and $\tilde{g}$ are called the *adjoint functors*, and ψ is a function which assigns to each pair of objects $x_1 \in X_1, x_2 \in X_2$ a bijection of sets

$$\psi : X_1\left(\tilde{g}\,(x_2), x_1\right) \cong X_2\left(x_2, f\,(x_1)\right)$$

which is *natural* in x_1 and x_2 (the arrows in the categories compose associatively with the functions). We refer to [56] for a full definition and treatment of adjunctions and adjoint functors. We also state a rigorous definition of an adjunction in section 3.

However, if we think back to our reduction, it is not necessarily the case that our reduction will be one-to-one. In practice, this comes up quite frequently: we cannot usually just forget what the original instance of the problem was in our reduction! In our functor definitions, this would correspond to a functor needing multiple outputs on a single input, which is obviously not well-defined. So we can define a reduction to be a set of functors $f : X \to X'$ and $g : C' \times X \to C$. Note that the extra input $x \in X$ to g allows us to "remember" which problem instance we originally started with, and then make sure we get back to the original problem instance.

Unfortunately, this means that we cannot model a reduction exactly as an adjunction. However, the intuition certainly applies, and it is probably possible to define a more complicated form of adjunction that could handle this case. We leave this as future work for more experienced category theorists.

Randomized Reductions Between Families of Problems. We now focus on randomized reductions between families of problems. Note that our previous reduction definitions have been entirely deterministic: any language instance $x \in X_1$ always maps to the same language instance $x' \in X_2$. This may not be a reasonable restriction in practice; in particular, it does not allow us to model typical reductions in complexity theory and cryptography. In order to additionally model the randomness in each reduction, we add an additional "randomness" category to our model. Each object in the "randomness" category induces a different (deterministic) reduction between the two families of problems. We formalize this notion in the following definition.

Definition 9 (Category-Theoretic Randomized Reduction)**.** *Let $C_1 := \{X_1, W_1\}$ and $C_2 := \{X_2, W_2\}$ be language categories as defined in Definition 8. Let R be a category representing randomness. We define a category-theoretic randomized reduction as a pair of efficiently computable functors:*

$$f : X_1 \times R \to X_2, \quad g : C_2 \times R \times X_1 \to C_1.$$

Informally speaking, for a given randomness object $r \in R$, the functor $f(\cdot, r)$ is a deterministic mapping between problem instances of the form

$$f(\cdot, r) : X \to X'.$$

The functor g allows us to map witnesses in W_2 back to witnesses in W_1 in a structure-preserving way so that any valid witnesses generated in W_2 will map back to valid witnesses in W_1 of our original language instances. Note that we use *the same category R* in both the forward and reverse directions. This is because computing the "reverse" direction of a reduction might require some of the randomness used in the "forward" direction, so we need to give g access to all of the randomness used by f. Of course, g could use some additional randomness objects that are not required by f; for simplicity, we assume that f ignores these randomness objects (for some reductions, R may also be split into subcategories)).

5.3 CATEGORY-THEORETIC MODELING OF RANDOMIZED SELF REDUCTIONS

We now use the aforementioned category-theoretic modeling of (Levin) reductions to show that certain randomized self reductions have some inherent mathematical structure. We assume here that languages have efficient algorithms for sampling a random (instance, witness) tuple. This restricts the languages to the world of Minicrypt [47], but still gives us an interesting class of languages to study, especially since one of our primary goals is to study reductions with cryptographic applications. On the other hand, assuming that languages have efficient algorithms for sampling random instances but *not* witnesses places us in the world of Pessiland [47]. As we observe subsequently, Pessiland does not support interesting notions of average-case hardness in any case.

From a category-theoretic point of view, we encompass this sampleability assumption by incorporating a "sampling" functor $S : \hat{R} \rightarrow X \times W$ that takes in randomness $\hat{r} \leftarrow \tilde{R}$ and outputs a valid problem instance and witness tuple (x, w). It is slightly against standard category theory conventions to call S a functor since we do not anticipate using the structure of the randomness set $\tilde{R}$ in any way, but we will do so anyway because we will compose it with proper functors. We formalize this below.

Definition 10 (Language Instance Sampling Functor)**.** *Let $C := \{X, W\}$ be a language category, and let $\hat{R}$ be a category representing randomness. We define a* language instance sampling functor *as a functor*

$$S := \hat{R} \rightarrow X \times W$$

that takes as input some random coins $\hat{r}$ and outputs an instance-witness tuple of the form (x, w) so that there exist arrows between x and w in the corresponding language category C.

Note that the above definition formalizes the fact that we can efficiently sample a (instance, witness) pair for a language. With this in mind, we are ready to define a randomized self-reduction.

Definition 11 (Randomized Self Reduction)**.** *A randomized self reduction is a randomized reduction as formalized in Definition 9 where the categories C_1 and C_2 are identical. More concretely, let $C = \{X, W\}$ be a language categories as defined in Definition 8 and let R be a category representing randomness. We define a category-theoretic randomized self reduction as a pair of efficiently computable functors:*

$$f : X \times R \rightarrow X, \quad g : C \times R \times X \rightarrow C.$$

with the additional requirement that, for every $x \in X$, sampling $r \leftarrow R$ and computing $f(x, r)$ results in a distribution statistically close to the output distribution of S.

This is very similar to definition 9 except for the fact that we map from C to itself (and, by corollary, X to itself) and have an additional requirement on the output distribution of f.

Let's now examine the structure of randomized self-reductions more closely. R defines two functors that are automorphisms (in different directions): one, from $X \rightarrow X$ in the forward direction, and one, from $C \leftarrow C$, in the opposite direction. Each "reduction" can be defined by some $r \in R$. We explained before that these functors f and g defined something similar to an adjunction earlier. Here, we have the special case that they map to and from the same category.

From Randomized Self Reduction to One-Way Monoid Action. We now use our category-theoretic modeling to show that any randomized self reduction that fits our model implies a one-way monoid action. We begin by describing the monoid in this implication.

Let R be a category representing randomness as in Definition 11 with compact representation. We define the *randomness concatenation monoid* M_R to be the monoid such that elements of M_R are concatenations of string-representations of objects in R, and the monoid operation is concatenation. M_R also has a special element $e_M = \varepsilon$, which is an empty string and represents the identity element with respect to string concatenation. We note here that string concatenation (more precisely, concatenation of elements of any finite set) is known to be a monoid [56].

The Monoid Action. Consider a randomized self-reduction parameterized by a tuple of functors (f, g) as in Definition 11 on a language category $C := \{X, W\}$ with randomness space R. We define the action $\star : M_R \times X \to X$ in the following way:

$$\star(e_M = \varepsilon, x) := x,$$
$$\star(m_R = (r_\ell \| \ldots \| r_1), x \in X) := \underbrace{f\left(\ldots f\left(f\left(x, r_1\right) r_2\right) \ldots r_\ell\right)}_{\ell-\text{times}}.$$

Note that m_R represents the randomness used in many different reductions concatenated together. The operation $\star$ decomposes the randomness from all of the reductions into randomness for many single reductions, and then applies them in sequence to a problem instance. So $\star$ essentially just computes one or more of the "forward" directions of the reduction.

Lemma 1. *$(M_R, X, \star)$ is a monoid action.*

Proof. As already mentioned, M_R is a monoid by the fact that string concatenation (more precisely, concatenation of elements of any finite set) is known to be a monoid [56]. Also, the action satisfies identity by definition. Finally, it is immediate to see that for any $m_1, m_2 \in M_R$ and any $x \in X$,

$$m_2 \star (m_1 \star x) = (m_2 \| m_1) \star x.$$

To see this, choose positive integers ℓ and ℓ', and let $m_1 = a_\ell \| \ldots \| a_1$ and $m_2 = b_{\ell'} \| \ldots \| b_1$ for all $a_i, b_i \in R$. Note that both of these operations equate to

$$g\left(\ldots g\left(g\left(\ldots g\left(x, a_1\right), \ldots a_\ell\right), b_1\right) \ldots b_{\ell'}\right)$$

□

Distributional One-Way Monoid Action. We now show that $(M_R, X, \star)$ is what we call a distributional *one-way* monoid action.

Definition 12 (Distributional One-Way Monoid Action)**.** *A monoid action $(M, X, \star)$ such that the set X supports efficient representation, and such that the "action operation" $\star$ is efficiently computable is said to satisfy distributional one-wayness with respect to some efficiently sampleable distribution $\mathcal{D}_M$ over M with respect to some security parameter λ if for any $g \leftarrow \mathcal{D}_M$, any $x \in X$ and any probabilistic polynomial-time adversary $\mathcal{A}$, we have*

$$\Pr[\mathcal{A}(x, g \star x) = g' \text{ such that } g' \star x = g \star x.] < \text{negl}(\lambda).$$

Lemma 2. *Consider a randomized self-reduction parameterized by a tuple of functors (f, g) as in Definition 11 on a language category $C := \{X, W\}$ with randomness space R. Let $(M_R, X, \star)$ be the monoid action as described above. Also, assume that there exists another sampling functor $S : \hat{R} \to X \times W$ that takes in randomness $\hat{r} \leftarrow \hat{R}$ and outputs a random valid problem instance and witness tuple (x, w). Then the monoid action $(M_R, X, \star)$ satisfies distributional one-wayness as per Definition 12.*

Proof. Suppose that the monoid action $(M_R, X, \star)$ does not satisfy distributional one-wayness as per Definition 12. In other words, suppose there exists an adversary $\mathcal{A}$ that breaks the distributional one-wayness of $(M_R, X, \star)$. We present a reduction that uses this adversary to construct an algorithm that, given a problem instance $x \in X$, outputs a valid witness w for x with non-negligible probability. This reduction is sufficient to prove Lemma 2.

The reduction is relatively straightforward. Given a problem instance $x \in X$, we sample an instance of randomness $\hat{r} \in \hat{R}$ and compute $S(\hat{r}) = (x', w')$. We then give the adversary $\mathcal{A}$ the tuple of set elements (x, x') as the challenge in the distributional one-wayness experiment as described in Definition 12. By assumption, the adversary $\mathcal{A}$ breaks the distributional one-wayness of $(M_R, X, \star)$ and hence, with non-negligible probability, outputs a monoid element $m_R \in M_R$ such that $m_R \star x = x'$.

We now proceed as follows. We first compute a decomposition of this monoid element m_R into elements of R. Then, given an equality $m_R = r_\ell \| r_{\ell-1} \| \ldots \| r_2 \| r_1$, we first re-trace the "forward" sequence

$$f\left(\ldots f\left(f\left(x, r_1\right), r_2\right) \ldots, r_\ell\right)$$

making sure to note down the corresponding problem instance $x'_i \in X$ at every step. This just ensures we have all of the necessary information to compute the reduction in the "reverse" direction (we discussed this earlier in our definition of a randomized self-reduction).

Given this, we can then trace the "backward" sequence to compute the tuple

$$(x, w) = g\left(\ldots g\left(g\left(\{x', w'\}, r_\ell, x'_\ell\right), r_{\ell-1}, x'_{\ell-1}\right) \ldots r_1, x'_1\right)$$

which, by the correctness of g, gives us a valid witness w for x. This completes the proof of Lemma 2. □

Interestingly, this reduction works at the "natural transformation" layer of the protocol. Rather than show that particular language instances are structured, or even reductions between language instances are structured, we show that *reductions between reductions between language instances are structured*. To our knowledge, this style of argument is original, and we think it merits further study and application.

Finally, putting everything together allows us to state the following theorem:

Theorem 1. *Any randomized self-reduction implies the existence of a one-way monoid action.*

In summary, we have the following (informal) observation: there is some mathematical structure inherent to a randomized self-reduction.

5.4 DISCUSSION

In this section, we provide some more informal observations about the structure of average-case reductions. We think that formalizing these concepts would be excellent future work.

On the Structure of the Randomness. One interesting thing to note is the effect of the choice of the category R on the problem. We can apply reductions in sequence, so we can reach any $m_R \in M_R$ that is spanned by the elements of R, which we can view as a generating set. There are some interesting issues around leftover hashing here and geenrating sets over monoids that might lead to further insight into the problem, but this gets into topics that are far beyond the scope of this work. Let's illustrate what can be impacted by the structure of R with a couple of examples.

On one hand, suppose that R contains only values that map to a single element. In other words, R contains $r_1, \ldots, r_n$ such that r_i maps all $x \in X$ to some x_i. So each r_i can be viewed as inducing a map with many source nodes and only one sink node. This is even a "perfect" randomized self reduction, since, over the choice of randomness, reducing to each language instance is equally likely. However, a language with such a randomized self-reduction is not very useful for cryptographic applications

Now, given r_i and x_i, if we can find a witness w_i for x_i, then we can use r_i to find witnesses for all of the language instances in X since we can apply the reverse direction of the reduction back to whatever instance we want! In other words, if we can find a witness for a *single* random instance, then we can immediately use that witness to generate witnesses for all of the other problems in X. This means that X isn't a very desirable set of language instances (or problems), and would be really problematic for cryptography (unless, of course, the reader works for a governmental agency–there are always matters of perspective). The mathematical structure for R here is also interesting: R is a monoid, but for every $r_1, r_2 \in R$, we have $r_1 \oplus r_2 = r_1$.

On the other hand, suppose that R contains r elements $r_1, \ldots, r_n$ such that each r_i maps each problem instance x_i to completely different problem instances in X. This gives us much better properties–if we find or generate a witness for one arbitrary instance in X, we are unlikely to be able to use the solution to find witnesses for other instances in X immediately, outside of the one instance we used to generate the randomized self-reduction. So this is much more useful type of reduction from a cryptographic perspective.

The structure of R in this case is interesting as well. Since it has no obvious kernels, it should contain at least some subgroup that has inverses (although the whole set R does not have to be a group). This is obviously a much nicer structure than before, and, if we require that $|R| = |X|$, then we effectively guarantee inverses, getting a one-way *group* action.

In summary, what we have here is the following observation: the mathematical structure of a randomized self-reduction is not too dissimilar from that of key exchange (which implies an *abelian* unpredictable monoid action) although slightly weaker, but we also have the fact that, at least seemingly, the more structured the randomness set R of the randomized self-reduction is is, the "better" the underlying problem is for things like cryptography. This is a very informal statement, but we think a formalization of this could yield very interesting results.

What Happens In Pessiland? Our finding that any randomized self-reduction implies the existence of a one-way monoid action is predicated on the fact that we can sample instances of hard problems in conjunction with their corresponding witnesses. This, of course, implies the existence of one-way functions. But what if this is not the case–what if we can only sample hard problems without solutions? In other words, what happens when we are in the world of Pessiland [47]?

In this case, suppose we consider any reduction that shows average-case hardness. By our starting assumption, we know that these reductions cannot be one-way: given two language instances x_1 and x_2, if it is possible to create a reduction between them, then it must be possible to solve for the randomness of the reduction given x_1 and x_2 with non-negligible probability. Otherwise, a one-way function exists. This means that for any class of language instances that are reducible to each other (or from a single instance), there exists an algorithm to generate a witness for *any* language instance in the set given a witness to any one of the language instances.

To summarize, it appears that Pessiland is unlikely to support randomized self-reductions that could be be useful for cryptographic applications. These facts on Pessiland may be known or may be folklore, but we were unable to find a reference. We leave it as an interesting open question to investigate if our category-theoretic modeling techniques might be useful for further illuminating properties of Pessiland, about which very little is known [69].

5.5 COLLISION-RESISTANT HASH FUNCTIONS

We now show that a similar mathematical structure is implied by certain families of collision-resistant hash functions (CRHFs). Informally, a CRHF is a function $f : X \to Y$ such that it is hard to find two elements $x_1, x_2 \in X$ such that $f(x_1) = f(x_2)$. CRHFs are one of the most widely used cryptographic primitives, which makes it interesting to study what kind of mathematical structure might be inherent to CRHFs.

Our main observation in this section is the following: we show that any (sufficiently compressing) function family F that has a reduction from collision resistance to one-wayness implies the existence of a one-way monoid action. In other words, it seems unlikely that we can build provably secure CRHFs with collision resistance reducible to one-wayness from unstructured ad-hoc assumptions, such as those used by practically efficient CRHFs (e.g., SHA). We note here that almost all known provably secure CRHFs rely on the same set of assumptions for collision resistance and one-wayness. Hence, our results seemingly indicate some lower bounds on the practical efficiency of provably secure CRHFs.

Let $F = F_\lambda$ be a family of functions with input space X and output space Y (λ being a security parameter). Let $\mathcal{D}_F$ and $\mathcal{D}_X$ denote some efficiently sampleable distributions over F and X, respectively.

Definition 13 (One-Way Function Family)**.** *We say that F is one-way with respect to the distributions $(\mathcal{D}_F, \mathcal{D}_X)$ if for any $f \leftarrow \mathcal{D}_F$, any $x \leftarrow \mathcal{D}_X$, and for any PPT adversary $\mathcal{A}$, we have*

$$\Pr[\mathcal{A}(f, f(x)) = x' \text{ such that } f(x) = f(x')] < \text{negl}(\lambda).$$

Definition 14 (Collision Resistant Function Family)**.** *We say that F is collision resistant with respect to the distribution $\mathcal{D}_F$ if for any $f \leftarrow \mathcal{D}_F$, and for any PPT adversary $\mathcal{A}$, we have*

$$\Pr[\mathcal{A}(f) = (x, x') \text{ such that } f(x) = f(x')] < \text{negl}(\lambda).$$

Lemma 3. *Let $F = F_\lambda$ be a family of functions with input space X and output space Y such that for any $f \in F$ and an overwhelmingly large fraction of $x \in X$, $f(x)$ has super-polynomially many preimages. Then collision resistance of F implies one-wayness of F. In other words, there is a reduction from the one-wayness of F to the collision resistance of F.*

This is a folklore result for which the proof is relatively simple; so we omit it (the folklore proof assumes uniform distributions but can be adapted in a straightforward manner to work for non-uniform distributions as well). The basic idea is just to sample a random input value $x \in X$, compute $y = f(x)$, and then send y as a challenge to the one-way function adversary. If the adversary finds a solution $x' \in X$ such that $y = f(x')$, we have that $x \neq x'$ with high probability due to the density of inputs per output, giving us a collision.

We now state our main observation.

Lemma 4. *Let $F = F_\lambda$ be a family of functions with input space X and output space Y such that for any $f \in F$ and an overwhelmingly large fraction of $x \in X$, $f(x)$ has super-exponentially many preimages. If there exists a reduction from the collision resistance of F to the one-wayness of F, then there exists a one-way monoid action.*

Proof. To prove this statement, it suffices to show the existence of a randomized self-reduction. Observe that by Lemma 3, there is a reduction from the one-wayness of F to the collision resistance of F. Also, by assumption, there exists a reduction from the collision resistance of F to the one-wayness of F. By chaining these (randomized) reductions, we get a randomized self-reduction on the one-wayness of F. At this point, we can invoke Theorem 1 to argue the existence of a one-way monoid action.

The rigorously formal proof again uses a category-theoretic modeling of the one-wayness of F. This can be done by defining a category $C := \{F \times Y, X\}$ such that each object in C is of the form $\{(f, y), x\}$ such that $f(x) = y$ (here (f, y) is an instance of the problem, and the preimage x is the witness). However the details are relatively straightforward, and are hence avoided. □

6 FUTURE WORK

In this paper, we show that analyzing cryptography and average-case complexity through a computational category-theoretic lens yields interesting results on the role of mathematical structure in these two fields. We briefly mention some potential future directions below.

New Separation Results. In our opinion, one of the most exciting facets of this new computational category-theoretic lens is the potential for new black-box separations in cryptography based on mathematical structure. As we explained earlier, we can model essentially all cryptosystems as category-theoretic diagrams. If the diagram of one cryptosystem contains another cryptosystem (and the "projection" of the computational assumptions from the larger diagram to the smaller one is appropriate), then we can obviously build the second cryptosystem from the first in a black-box way. Can we show any more complicated reductions, or, in particular, lower bounds?

The celebrated result of Impagliazzo and Rudich [49] is the main result we can currently exploit: in our language, it shows that we cannot generically build a "computationally hard" commutative square (key exchange) from a single functor (one-way function). Can we use topology-style proofs to turn [49] into a stronger statement that can be applied to many different kinds of commutative diagrams? Can we show other lower bounds on different commutative diagrams? Taking a category-theoretic view of cryptosystems seems to offer a lot of potential for black-box lower bounds in cryptography.

Classifying Cryptographic Primitives. In [7], the authors conjecture that public key cryptoprimitives can be classified by mathematical structure and propose a potential hierarchy for classification. They only "measure" mathematical structure in terms of algebraic objects such as groups and rings, so intuitively we should be able to do better by using the arbitrary structures provided by category theory. Can we do this, and provide a more concrete hierarchical view of Cryptomania primitives?

Average-Case Complexity. We show in this paper that mathematical structure can be helpful for understanding what kinds of problems have average-case reductions. But this barely scratches the surface in terms of what we would like to know about average-case complexity (which is much less than what we know about worst-case complexity). Can we use mathematical structure to help achieve a better understanding of average-case complexity?

Key Exchange from New Concrete Assumptions. Given our exact characterization of the mathematical structure necessary for key exchange, we hope that it becomes easier to identify new (and plausibly post-quantum secure) concrete assumptions to build key exchange. We think looking at assumptions with very minimal structure might be quite interesting, particularly because these kinds of assumptions are more likely to be quantum-resistant.

Proving Indistinguishability Obfuscation (iO) is "Crypto-Complete." Many cryptographers have referred to iO as a "crypto-complete" primitive. Can this be shown in any kind of formal way?

We expect that it would be straightforward to instantiate any kind of category-theoretic diagram using virtual black-box (VBB) obfuscation, but this is not particularly useful since VBB is known to be uninstantiable in general [14]. In particular, suppose that given a cryptographic diagram, we VBB-obfuscate each functor in this diagram. This essentially is equivalent to giving an adversary oracle access to the functors and, by definition, does not leak any information beyond what an adversary is "allowed" to learn from the diagram. Now, assuming that the diagram correctly captures the security properties of the underlying cryptoprimitive, we achieve a secure realization of this primitive from VBB-obfuscation. This is, of course, an informal intuition and we do not claim any formal results related to this statement, but we expect a construction and proof in this vein to be straightforward.

In particular, a "compiler" showing how to build computational category-theoretic diagrams from iO would seemingly be an interesting statement on the power of iO.

Improvements to Our Framework and Techniques. We emphasize that we approached the computational category-theoretic framework described in this paper from the perspective of cryptographers. Hence, some of our formalization may not adhere to traditional conventions in category-theory. We leave it open to improve upon our framework and techniques using more technical formalizations from a complexity-theoretic or category-theoretic point of view.

ACKNOWLEDGEMENT

The authors would like to thank Navid Alamati for helpful discussions on an earlier version of this paper.

REFERENCES

[1] Dimitris Achlioptas and Ricardo Menchaca-Mendez. "Exponential lower bounds for DPLL algorithms on satisfiable random 3-CNF formulas". In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer. 2012, pp. 327–340.

[2] Dimitris Achlioptas et al. "Bounds for random constraint satisfaction problems via spatial coupling". In: *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2016, pp. 469–479.

[3] Adi Akavia et al. "On basing one-way functions on NP-hardness". In: *38th ACM STOC*. Ed. by Jon M. Kleinberg. ACM Press, May 2006, pp. 701–710. DOI: 10.1145/1132516.1132614.

[4] Adi Akavia et al. "Erratum for: on basing one-way functions on NP-hardness". In: *42nd ACM STOC*. Ed. by Leonard J. Schulman. ACM Press, June 2010, pp. 795–796. DOI: 10.1145/1806689.1806797.

[5] Gorjan Alagic, Stacey Jeffery, and Stephen P Jordan. "Partial-indistinguishability obfuscation using braids". In: *arXiv preprint arXiv:1212.6458* (2012).

[6] Navid Alamati, Hart Montgomery, and Sikhar Patranabis. "Symmetric Primitives with Structured Secrets". In: *CRYPTO 2019, Part I*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11692. LNCS. Springer, Heidelberg, Aug. 2019, pp. 650–679. DOI: 10.1007/978-3-030-26948-7_23.

[7] Navid Alamati et al. *Minicrypt Primitives with Algebraic Structure and Applications*. Cryptology ePrint Archive, Report 2019/108. https://eprint.iacr.org/2019/108. 2019.

[8] Navid Alamati et al. "Cryptographic Group Actions and Applications". In: *ASIACRYPT 2020, Part II*. LNCS. Springer, Heidelberg, Dec. 2020, pp. 411–439. DOI: 10.1007/978-3-030-64834-3_14.

[9] Eric Allender and Bireswar Das. "Zero knowledge and circuit minimization". In: *Information and Computation* 256 (2017), pp. 2–8.

[10] Benny Applebaum, Boaz Barak, and David Xiao. "On Basing Lower-Bounds for Learning on Worst-Case Assumptions". In: *49th FOCS*. IEEE Computer Society Press, Oct. 2008, pp. 211–220. DOI: 10.1109/FOCS.2008.35.

[11] Georgios Bakirtzis, Fabrizio Genovese, and Cody H Fleming. "Yoneda hacking: the algebra of attacker actions". In: *arXiv preprint arXiv:2103.00044* (2021).

[12] Boaz Barak. "The Complexity of Public-Key Cryptography". In: *Tutorials on the Foundations of Cryptography*. Ed. by Yehuda Lindell. Springer International Publishing, 2017, pp. 45–77. ISBN: 978-3-319-57047-1. DOI: 10.1007/978-3-319-57048-8. URL: https://doi.org/10.1007/978-3-319-57048-8.

[13] Boaz Barak and Mohammad Mahmoody-Ghidary. "Merkle Puzzles Are Optimal - An $O(n^2)$-Query Attack on Any Key Exchange from a Random Oracle". In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Heidelberg, Aug. 2009, pp. 374–390. DOI: 10.1007/978-3-642-03356-8_22.

[14] Boaz Barak et al. "On the (Im)possibility of Obfuscating Programs". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Heidelberg, Aug. 2001, pp. 1–18. DOI: 10.1007/3-540-44647-8_1.

[15] Nir Bitansky, Omer Paneth, and Daniel Wichs. "Perfect Structure on the Edge of Chaos - Trapdoor Permutations from Indistinguishability Obfuscation". In: *TCC 2016-A, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. LNCS. Springer, Heidelberg, Jan. 2016, pp. 474–502. DOI: 10.1007/978-3-662-49096-9_20.

[16] Andrej Bogdanov and Christina Brzuska. "On Basing Size-Verifiable One-Way Functions on NP-Hardness". In: *TCC 2015, Part I*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9014. LNCS. Springer, Heidelberg, Mar. 2015, pp. 1–6. DOI: 10.1007/978-3-662-46494-6_1.

[17] Andrej Bogdanov and Luca Trevisan. "Average-case complexity". In: *arXiv preprint cs/0606037* (2006).

[18] Dan Boneh and Matthew K. Franklin. "Identity-Based Encryption from the Weil Pairing". In: *CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, Heidelberg, Aug. 2001, pp. 213–229. DOI: 10.1007/3-540-44647-8_13.

[19] Dan Boneh, Amit Sahai, and Brent Waters. "Functional Encryption: Definitions and Challenges". In: *TCC 2011*. Ed. by Yuval Ishai. Vol. 6597. LNCS. Springer, Heidelberg, Mar. 2011, pp. 253–273. DOI: 10.1007/978-3-642-19571-6_16.

[20] Dan Boneh and Victor Shoup. *A graduate course in applied cryptography*. 2020.

[21] Anne Broadbent and Martti Karvonen. "Categorical composable cryptography". In: *arXiv preprint arXiv:2105.05949* (2021).

[22] Dario Catalano, Dario Fiore, and Bogdan Warinschi. "Adaptive Pseudo-free Groups and Applications". In: *EUROCRYPT 2011*. Ed. by Kenneth G. Paterson. Vol. 6632. LNCS. Springer, Heidelberg, May 2011, pp. 207–223. DOI: 10.1007/978-3-642-20465-4_13.

[23] Amin Coja-Oghlan. "Belief propagation guided decimation fails on random formulas". In: *Journal of the ACM (JACM)* 63.6 (2017), pp. 1–55.

[24] Stephen A Cook. "The complexity of theorem-proving procedures". In: *Proceedings of the third annual ACM symposium on Theory of computing*. 1971, pp. 151–158.

[25] Ronald Cramer and Victor Shoup. "Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption". In: *EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. LNCS. Springer, Heidelberg, Apr. 2002, pp. 45–64. DOI: 10.1007/3-540-46035-7_4.

[26] Derek Doran, Sarah Schulz, and Tarek R Besold. "What does explainable AI really mean? A new conceptualization of perspectives". In: *arXiv preprint arXiv:1710.00794* (2017).

[27] Niklas Eén and Niklas Sörensson. "An extensible SAT-solver". In: *International conference on theory and applications of satisfiability testing*. Springer. 2003, pp. 502–518.

[28] Uriel Feige. "Relations between average case complexity and approximation complexity". In: *34th ACM STOC*. ACM Press, May 2002, pp. 534–543. DOI: 10.1145/509907.509985.

[29] Joan Feigenbaum and Lance Fortnow. "Random-self-reducibility of complete sets". In: *SIAM Journal on Computing* 22.5 (1993), pp. 994–1005.

[30] Marc Fischlin. "Black-Box Reductions and Separations in Cryptography (Invited Talk)". In: *AFRICACRYPT 12*. Ed. by Aikaterini Mitrokotsa and Serge Vaudenay. Vol. 7374. LNCS. Springer, Heidelberg, July 2012, pp. 413–422.

[31] Marc Fischlin and Nils Fleischhacker. "Limitations of the Meta-reduction Technique: The Case of Schnorr Signatures". In: *EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. LNCS. Springer, Heidelberg, May 2013, pp. 444–460. DOI: 10.1007/978-3-642-38348-9_27.

[32] Nils Fleischhacker, Tibor Jager, and Dominique Schröder. "On Tight Security Proofs for Schnorr Signatures". In: *ASIACRYPT 2014, Part I*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. LNCS. Springer, Heidelberg, Dec. 2014, pp. 512–531. DOI: 10.1007/978-3-662-45611-8_27.

[33] David Gamarnik. "The overlap gap property: A topological barrier to optimizing over random structures". In: *Proceedings of the National Academy of Sciences* 118.41 (2021).

[34] David Garber. *Braid Group Cryptography*. 2008. arXiv: 0711.3941 [cs.CR].

[35] Sanjam Garg et al. "Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits". In: *54th FOCS*. IEEE Computer Society Press, Oct. 2013, pp. 40–49. DOI: 10.1109/FOCS.2013.13.

[36] Sanjam Garg et al. "Limits on the Power of Garbling Techniques for Public-Key Encryption". In: *CRYPTO 2018, Part III*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10993. LNCS. Springer, Heidelberg, Aug. 2018, pp. 335–364. DOI: 10.1007/978-3-319-96878-0_12.

[37] Fabrizio Genovese, Andre Knispel, and Joshua Fitzgerald. *Mapping finite state machines to zk-SNARKS Using Category Theory*. 2019. arXiv: 1909.02893 [cs.CR].

[38] Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *41st ACM STOC*. Ed. by Michael Mitzenmacher. ACM Press, May 2009, pp. 169–178. DOI: 10.1145/1536414.1536440.

[39] Craig Gentry, Amit Sahai, and Brent Waters. "Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based". In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Heidelberg, Aug. 2013, pp. 75–92. DOI: 10.1007/978-3-642-40041-4_5.

[40] Mikael Goldmann, Per Grape, and Johan Håstad. "On average time hierarchies". In: *Information processing letters* 49.1 (1994), pp. 15–20.

[41] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to construct random functions". In: *Journal of the ACM (JACM)* 33.4 (1986), pp. 792–807.

[42] Iftach Haitner, Mohammad Mahmoody, and David Xiao. "A new sampling protocol and applications to basing cryptographic primitives on the hardness of NP". In: *2010 IEEE 25th Annual Conference on Computational Complexity*. IEEE. 2010, pp. 76–87.

[43] Iftach Haitner, Alon Rosen, and Ronen Shaltiel. "On the (Im)Possibility of Arthur-Merlin Witness Hiding Protocols". In: *TCC 2009*. Ed. by Omer Reingold. Vol. 5444. LNCS. Springer, Heidelberg, Mar. 2009, pp. 220–237. DOI: 10.1007/978-3-642-00457-5_14.

[44] Nadia Heninger et al. "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices". In: *USENIX Security 2012*. Ed. by Tadayoshi Kohno. USENIX Association, Aug. 2012, pp. 205–220.

[45] Shuichi Hirahara. "Non-Black-Box Worst-Case to Average-Case Reductions within NP". In: *59th FOCS*. Ed. by Mikkel Thorup. IEEE Computer Society Press, Oct. 2018, pp. 247–258. DOI: 10.1109/FOCS.2018.00032.

[46] Susan Rae Hohenberger. "The cryptographic impact of groups with infeasible inversion". PhD thesis. Massachusetts Institute of Technology, 2003.

[47] R. Impagliazzo. "A personal view of average-case complexity". In: *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*. June 1995, pp. 134–147. DOI: 10.1109/SCT.1995.514853.

[48] Russell Impagliazzo. "Relativized separations of worst-case and average-case complexities for NP". In: *2011 IEEE 26th Annual Conference on Computational Complexity*. IEEE. 2011, pp. 104–114.

[49] Russell Impagliazzo and Steven Rudich. "Limits on the Provable Consequences of One-Way Permutations". In: *21st ACM STOC*. ACM Press, May 1989, pp. 44–61. DOI: 10.1145/73007.73012.

[50] Aayush Jain, Huijia Lin, and Amit Sahai. "Indistinguishability obfuscation from well-founded assumptions". In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 2021, pp. 60–73.

[51] Zhengfeng Ji et al. "General Linear Group Action on Tensors: A Candidate for Post-quantum Cryptography". In: *TCC 2019, Part I*. LNCS. Springer, Heidelberg, Mar. 2019, pp. 251–281. DOI: 10.1007/978-3-030-36030-6_11.

[52] Richard M Karp. "Reducibility among combinatorial problems". In: *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[53] Tom Leinster. *Basic Category Theory*. 2016. arXiv: 1612.09375 [math.CT].

[54] Yanyi Liu and Rafael Pass. "On One-way Functions and Kolmogorov Complexity". In: *61st FOCS*. IEEE Computer Society Press, 2020, pp. 1243–1254. DOI: 10.1109/FOCS46700.2020.00118.

[55] Yanyi Liu and Rafael Pass. "On the Possibility of Basing Cryptography on $\mathsf{EXP} \neq \mathsf{BPP}$". In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual: Springer, Heidelberg, Aug. 2021, pp. 11–40. DOI: 10.1007/978-3-030-84242-0_2.

[56] Saunders Mac Lane. *Categories for the working mathematician*. Vol. 5. Springer Science & Business Media, 2013.

[57] Mohammad Mahmoody and Rafael Pass. "The Curious Case of Non-Interactive Commitments - On the Power of Black-Box vs. Non-Black-Box Use of Primitives". In: *CRYPYO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. LNCS. Springer, Heidelberg, Aug. 2012, pp. 701–718. DOI: 10.1007/978-3-642-32009-5_41.

[58] Mohammad Mahmoody and David Xiao. "On the power of randomized reductions and the checkability of SAT". In: *2010 IEEE 25th Annual Conference on Computational Complexity*. IEEE. 2010, pp. 64–75.

[59] Ralph C. Merkle. "A Certified Digital Signature". In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, Heidelberg, Aug. 1990, pp. 218–238. DOI: 10.1007/0-387-34805-0_21.

[60] Matthew W Moskewicz et al. "Chaff: Engineering an efficient SAT solver". In: *Proceedings of the 38th annual Design Automation Conference*. 2001, pp. 530–535.

[61] Dusko Pavlovic. "Chasing diagrams in cryptography". In: *Categories and Types in Logic, Language, and Physics*. Springer, 2014, pp. 353–367.

[62] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *37th ACM STOC*. Ed. by Harold N. Gabow and Ronald Fagin. ACM Press, May 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.

[63] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. "Notions of Reducibility between Cryptographic Primitives". In: *TCC 2004*. Ed. by Moni Naor. Vol. 2951. LNCS. Springer, Heidelberg, Feb. 2004, pp. 1–20. DOI: 10.1007/978-3-540-24638-1_1.

[64] Ronald L Rivest, Adi Shamir, and Leonard Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[65] Ronald L. Rivest. "On the Notion of Pseudo-Free Groups". In: *TCC 2004*. Ed. by Moni Naor. Vol. 2951. LNCS. Springer, Heidelberg, Feb. 2004, pp. 505–521. DOI: 10.1007/978-3-540-24638-1_28.

[66] Rahul Santhanam. "Pseudorandomness and the Minimum Circuit Size Problem". In: *ITCS 2020*. LIPIcs, Jan. 2020, 68:1–68:26. DOI: 10.4230/LIPIcs.ITCS.2020.68.

[67] Adi Shamir. "Identity-Based Cryptosystems and Signature Schemes". In: *CRYPTO'84*. Ed. by G. R. Blakley and David Chaum. Vol. 196. LNCS. Springer, Heidelberg, Aug. 1984, pp. 47–53.

[68] Eftychios Theodorakis and John C. Mitchell. *Semantic Security Invariance under Variant Computational Assumptions*. Cryptology ePrint Archive, Report 2018/051. https://ia.cr/2018/051. 2018.

[69] Hoeteck Wee. “Finding Pessiland”. In: *TCC 2006*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. LNCS. Springer, Heidelberg, Mar. 2006, pp. 429–442. DOI: 10.1007/11681878_22.

[70] Virginia Vassilevska Williams and Ryan Williams. “Subcubic Equivalences between Path, Matrix and Triangle Problems”. In: *51st FOCS*. IEEE Computer Society Press, Oct. 2010, pp. 645–654. DOI: 10.1109/FOCS.2010.67.